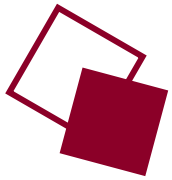
Machine Learning

14장 최적화를 위한 점검

고려대학교 통계학과
박유성



Contents

CALLBACKS

- keras에서는 callbacks 라이브러리를 이용하여 앞에 열거한 사항을 점검할 수 있다.

```
from keras.callbacks import EarlyStopping, ModelCheckPoint, ReduceLROnPlateau
```

- EarlyStopping: 검증데이터의 정밀도를 기준으로 일정 에폭수가 반복되더라도 정밀도의 향상이 없을 때 에폭의 반복을 멈추기 위해 사용
- ModelCheckPoint: 매 에폭마다 추정된 모수를 저장할 수 있으며 검증데이터의 손실함수 값이 가장 작을 때의 모수만을 저장할 수도 있다. EarlyStopping과 같이 사용한다.

CALLBACKS

- ReduceLROnPlateau: 일정 에폭이 진행되었는데도 불구하고 검증데이터의 정밀도가 향상되지 않을 경우에 학습률을 조정하고자 할 때 사용된다.
- 이러한 callbacks은 list 형태로 model.fit의 옵션으로 사용하게 된다.

CALLBACKS

```
from keras.callbacks import EarlyStopping, ModelCheckPoint, ReduceLROnPlateau
callback_list=[EarlyStopping(monitor='acc', patience=2)
ModelCheckPoint(filepath='~/model_c.h5', monitor='val_loss', save_best_only='True')}
model_c.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
model_c.fit(x,y,epochs=20, batch_size=64, callbacks=callback_list, validation_data=(x_val,y_
val))
```

- EarlyStopping의 monitor='acc'는 검증데이터의 metric=['acc']를 이용하여 에폭이 20번에 도달하기 이전에 멈출 수 있으며, patient=2에 의해 연속해서 3번 이상 검증데이터의 'acc'가 향상되지 않으면 에폭을 멈추게 된다.
- 그러므로 model_c.compile에서 metric=['acc']로 선택되어야만 한다.

CALLBACKS

- ModelCheckPoint에 의해 지정한 path에 'model_c.h5' 파일에 추정된 모수가 저장된다.
- save_best_only='True'를 부여함으로써 최종적으로 가장 우수한 모수만을 저장한다.
- 모수의 우수성은 검증데이터의 손실함수 값(monitor='val_loss') 기준이다.
- EarlyStopping과 ModelCheckPoint를 사용하기 위해서는 반드시 model.fit에 검증데이터가 포함되어 있어야 한다.

CALLBACKS

```
callback_list=[ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=10)]
model.fit(x,y, batch_size=64, callbacks=callback_list, validation_data=(x_val,y_val))
```

- 연속해서 11번의 에폭 동안 검증데이터의 손실함수 값이 줄어들지 않으면 학습률을 1/10수준으로 낮추라는 의미
- 국소최소가 의심되면 factor를 1보다 크게, 1보다 작게 주어 두 가지 경우를 모두 해보는 것을 권장한다. 국소최소를 벗어나는데 학습률의 증가 혹은 감소 중 어느 것이 좋은 지 일반적으로 알기 어렵기 때문이다.

TensorBoard

TensorBoard를 통해

1. 딥러닝모형의 학습과정에서 손실함수와 정밀도 등의 metrics를 점검할 수 있으며
2. 딥러닝모형의 아키텍처를 시각화할 수 있으며
3. 활성함수 출력값 등에 대한 히스토그램을 볼 수 있으며
4. 기타 word embedding, 이미지 자료 등에 대한 추가적 정보를 제공하여 준다.

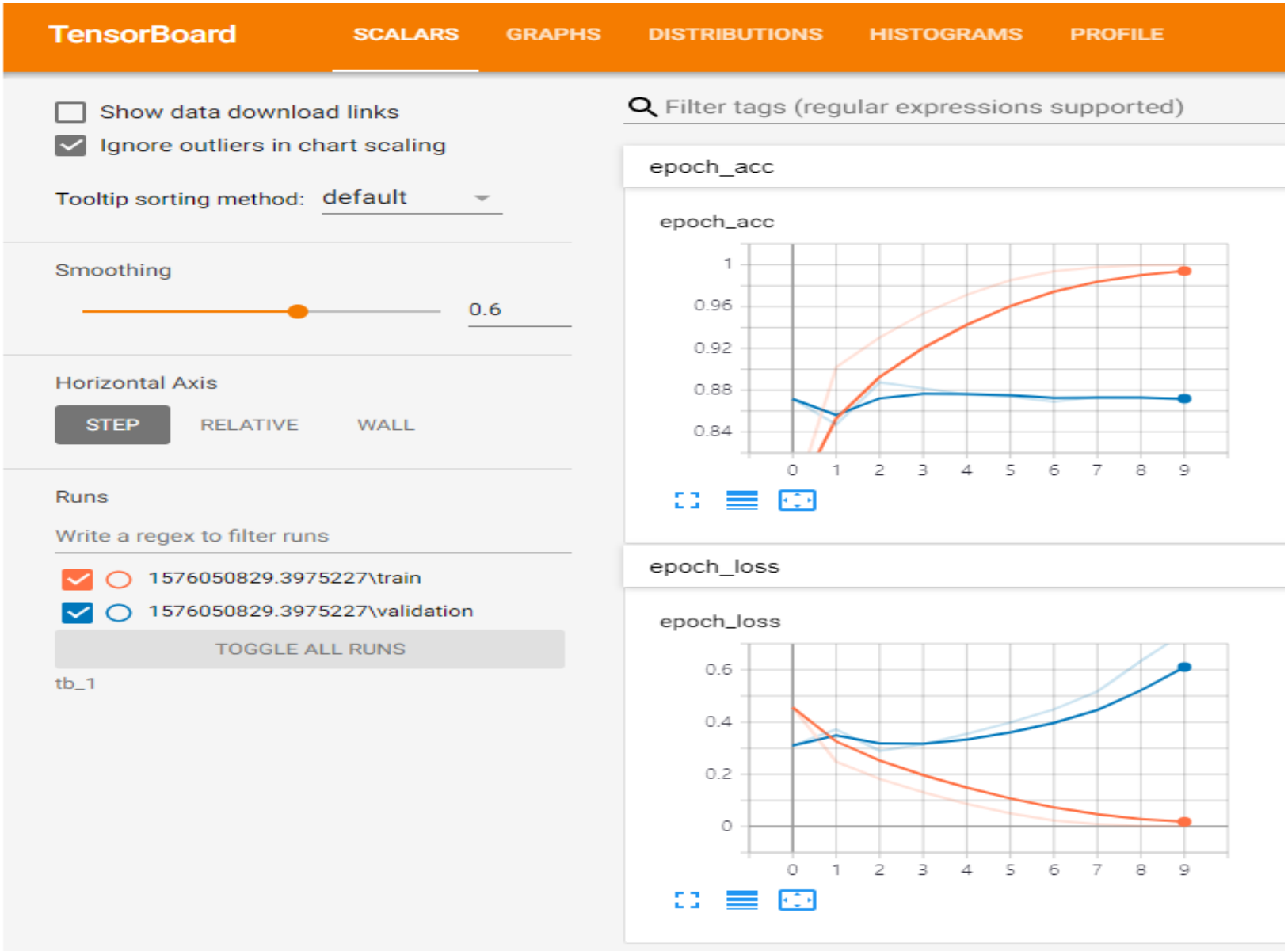
- TensorBoard는 브라우저를 통해 구현되므로 http://localhost:6006에 브라우즈(browse)하여 시각화를 할 수 있다.

TensorBoard

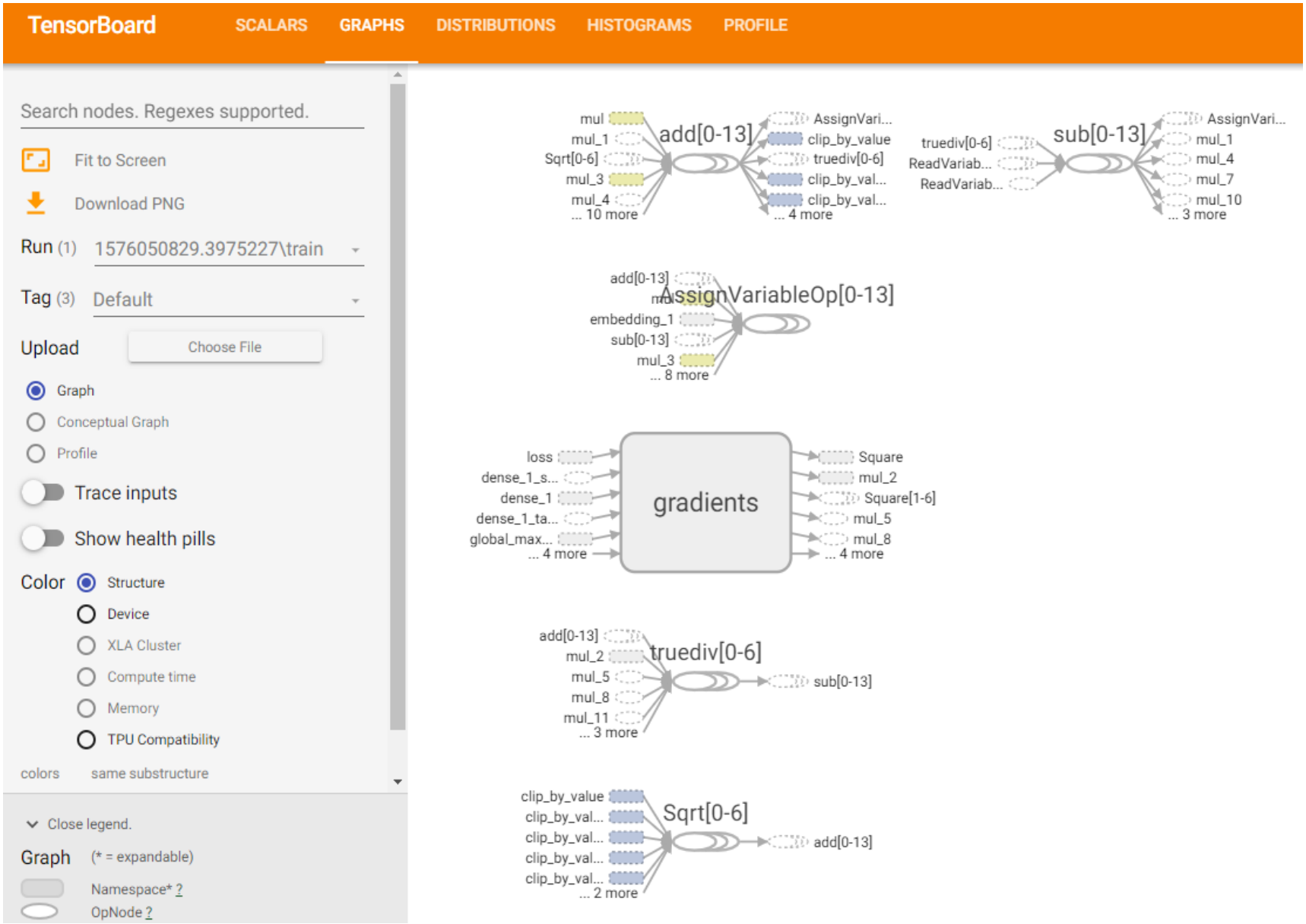
```
from keras.callbacks import TensorBoard
from time import time
import os
tensorboard = TensorBoard(log_dir = 'tb_1₩{}'.format(time()), histogram_freq=1)
m_tb.compile(optimizer='rmsprop',loss='binary_crossentropy',metrics=['acc'])
m_tb.fit(x_train,y_train, epochs=10, batch_size=64, validation_split=0.2, callbacks=[tensorboard])
```

- Anaconda prompt에서 반드시 base를 path로 만든 후 "tensorboard --logdir=tb_1"을 입력한 후 http://localhost:6006에 들어가면 Tensorboard를 볼 수 있다.
- 예를 들어 "A/B/C/tb_1"으로 path A/B/C에 tb_1이 저장되어 있다면 반드시 Anaconda prompt의 base를 A/B/C로 변경한 후, "tensorboard —logdir=tb_1"를 입력해야 한다.
- "tensorboard —logdir=A/B/C/tb_1"를 입력하면 tensorboard가 작동않됨.

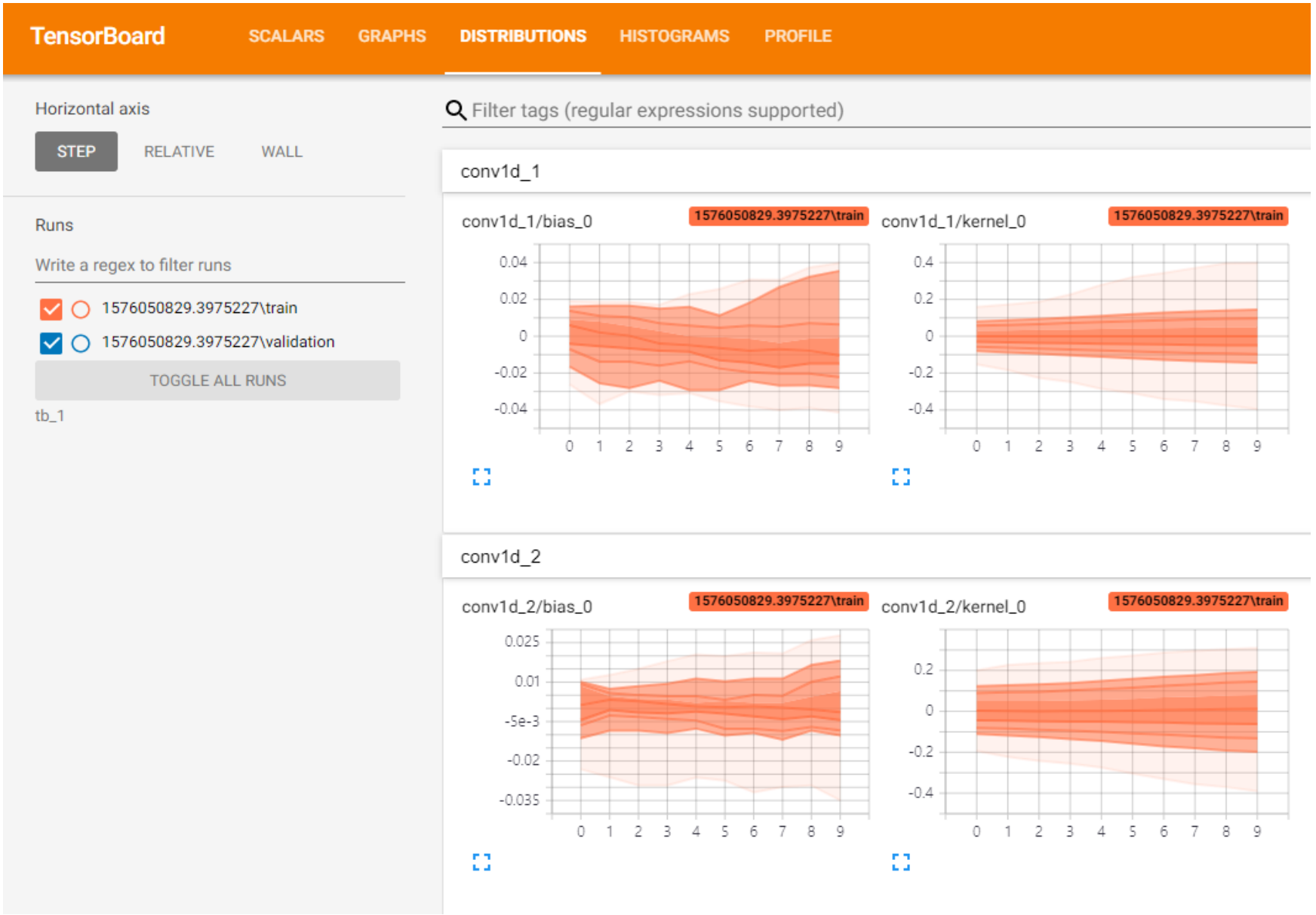
TensorBoard



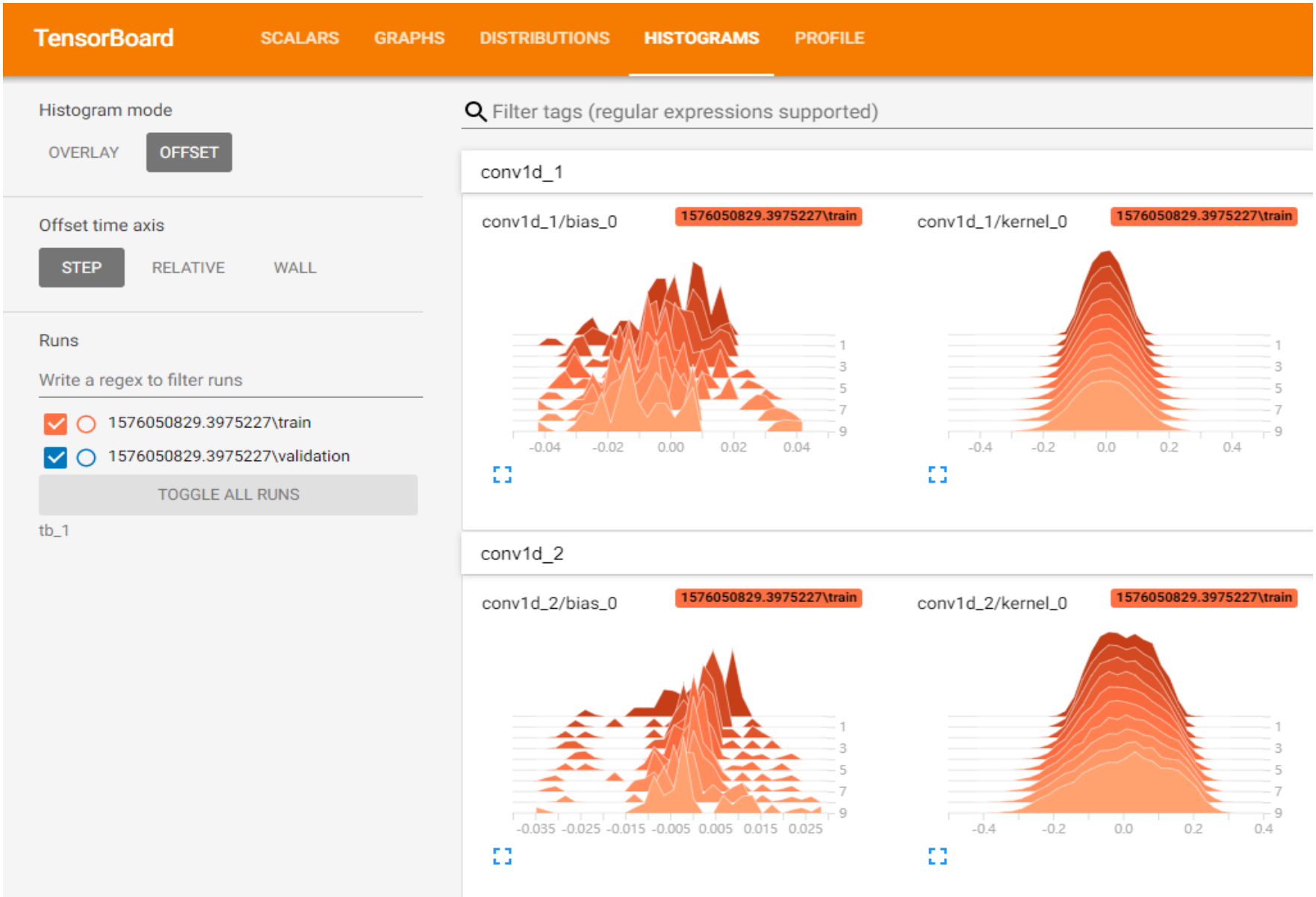
TensorBoard



TensorBoard



TensorBoard



모형의 시각화, 저장 그리고 불러오기

- 딥러닝모형의 아키텍처는 keras.utils.plot_model에 의해 좀 더 선명하게 시각화할 수 있다.
- 이를 위해 "pip install pydot", "pip install pydot-ng", 그리고 "pip install graphviz"를 차례로 anaconda prompt에 입력하여 설치하거나 jupyter notebook에서 pip앞에 !를 추가하면 된다.

```
!pip install pydot
!pip install pydot-ng
!pip install graphviz
```

모형의 시각화, 저장 그리고 불러오기

- keras에서 학습된 모형을 저장하는 것은 save() 함수를 이용하여 다음과 같이 간단하게 저장할 수 있다. 확장자는 h5를 사용하여야 한다.

```
model_tb.save('path/model_tb.h5')
```

- 저장된 모형을 불러내어 새로운 데이터(아래의 경우, xhat)에 적용할 수 있다.

```
from keras.models import load_model
loadedmodel=load_model('path/model_tb.h5')
yhat=loadedmodel.predict(xhat)
yhat=loadedmodel.predict_classes(xhat)
```

모형의 시각화, 저장 그리고 불러오기

- 저장된 모형은 load_model 함수를 이용하여 불러낼 수 있으며
- 회귀이거나 클래스에 속할 확률값을 예측하고자 할 때 loadedmodel.predict를 사용하고,
- 분류문제이면서 클래스 label를 예측하고자 할 때 predict_classes를 사용한다. 예를 들어, 클래스 label이 6이면 array[6]을 출력하게 된다.

Q & A