Мониторинг моделей глубокого обучения средствами библиотеки Keras

Многие аспекты обучения модели нельзя предсказать заранее. Например, нельзя предсказать заранее количество эпох, обеспечивающее оптимальное значение потерь на проверочном наборе. Лучшего останавливать обучение, как только выяснится, что оценка потерь на проверочном наборе перестала улучшаться. Это можно реализовать с использованием механизма обратных вызовов в Keras. Обратный вызов — это объект (экземпляр класса, реализующего конкретные методы), который передается в модель через вызов fit и который будет вызываться моделью в разные моменты в процессе обучения. Он имеет доступ ко всей информации о состоянии модели и ее качестве и может предпринимать следующие действия: прерывать обучение, сохранять модели, загружать разные наборы весов или как-то иначе изменять состояние модели.

Вот несколько примеров использования обратных вызовов:

- фиксация состояния модели в контрольных точках — сохранение текущих весов модели в разные моменты в ходе обучения;
- ранняя остановка — прерывание обучения, когда оценка потерь на проверочных данных перестает улучшаться (и, конечно, сохранение лучшего варианта модели, полученного в ходе обучения);
- динамическая корректировка значений некоторых параметров в процессе обучения, например шага обучения оптимизатора;
- журналирование оценок для обучающего и проверочного наборов данных в ходе обучения или визуализация представлений, получаемых моделью, по мере их обновления

Обратные вызовы ModelCheckpoint и EarlyStopping

Обратный вызов EarlyStopping можно использовать для прерывания процесса

обучения, если находящаяся под наблюдением целевая метрика не улучшалась

на протяжении заданного количества эпох. Например, этот обратный вызов позволит прервать обучение после наступления эффекта переобучения и тем самым избежать повторного обучения модели в течение меньшего количества эпох. Этот обратный вызов обычно используется в комбинации с ModelCheckpoint , который позволяет сохранять состояние модели в ходе обучения (и, при необходимости, сохранять только лучшую модель: версию модели, достигшую лучшего качества к концу эпохи). Пример данных вызовов:

```
import keras
callbacks_list = [
    keras.callbacks.EarlyStopping(
        monitor='val_acc',
        patience=1,
    ),
    keras.callbacks.ModelCheckpoint(
        filepath='my_model.h5',
        monitor='val_loss',
        save_best_only=True,
    )
]

model.compile(optimizer='rmsprop',
```

```
    loss='binary_crossentropy',
    metrics=['acc'])

model.fit(x, y,
    epochs=10,
    batch_size=32,
    callbacks=callbacks_list,
    validation_data=(x_val, y_val))
```

Обратный вызов ReduceLROnPlateau

Этот обратный вызов можно использовать для снижения скорости обучения, когда потери на проверочных данных перестают уменьшаться. Уменьшение или увеличение скорости обучения в точке перегиба кривой потерь — эффективная стратегия выхода из локального минимума в ходе обучения:

```
callbacks_list = [
    keras.callbacks.ReduceLROnPlateau(
        monitor='val_loss'
        factor=0.1,
        patience=10,
    )
]

model.fit(x, y,
    epochs=10,
    batch_size=32,
    callbacks=callbacks_list,
    validation_data=(x_val, y_val))
```

Разработка своего обратного вызова

Если в ходе обучения потребуется выполнить какие-то особые действия, не предусмотренные ни одним из встроенных обратных вызовов, можно написать свой обратный вызов. Обратные вызовы реализуются путем наследования класса keras.callbacks.Callback . Вы можете реализовать любые из следующих методов с говорящими именами, которые будут вызываться в соответствующие моменты в ходе обучения:

- on_epoch_begin
- on_epoch_end
- on_batch_begin
- on_batch_end
- on_train_begin
- on_train_end

Все эти методы вызываются с аргументом logs — словарем, содержащим информацию о предыдущем пакете, эпохе или цикле обучения: метрики обучения и проверки и т. д. Кроме того, обратный вызов имеет доступ к следующим атрибутам:

- self.model — экземпляр модели, вызвавшей этот обратный вызов
- self.validation_data — значение, переданное методу fit в качестве проверочных данных.

Вот простой пример нестандартного обратного вызова, который сохраняет на диск (как массивы Numpy) активации всех слоев модели после окончания каждой эпохи, вычисленные по первому образцу в проверочном наборе:

```
import keras
import numpy as np

class ActivationLogger(keras.callbacks.Callback):
    def set_model(self, model):
        self.model = model
        layer_outputs = [layer.output for layer in model.layers]
        self.activations_model = keras.models.Model(model.input,layer_outputs)

    def on_epoch_end(self, epoch, logs=None):
        if self.validation_data is None:
            raise RuntimeError('Requires validation_data.')
        validation_sample = self.validation_data[0][0:1]
        activations = self.activations_model.predict(validation_sample)
        f = open('activations_at_epoch_' + str(epoch) + '.npz', 'wb')
        np.savez(f, activations)
        f.close()
```

Введение в TensorBoard: фреймворк визуализации TensorFlow

Основное назначение TensorBoard — помочь визуально наблюдать за происходящим внутри модели в процессе обучения. Отслеживая больший объем информации, нежели просто окончательные потери модели, можно получить более четкое представление о том, что делает или чего не делает модель, и быстрее добиться прогресса. TensorBoard открывает доступ к некоторым замечательным возможностям через самое обычное окно браузера:

- визуальный мониторинг метрик в ходе обучения
- визуализация архитектуры модели
- · вывод гистограмм активаций и градиентов
- исследование векторных представлений в трехмерном пространстве

Обратите внимание: его можно использовать для исследования моделей Keras, только когда в качестве низкоуровневого механизма обработки тензоров Keras использует TensorFlow.

Для сохранения информации обратных вызовов для TensorBoard создадим папку, например, *my_log_dir* .

Теперь запустим обучение, передав экземпляр TensorBoard в качестве обратного вызова. Этот обратный вызов будет записывать события на диск в указанный каталог:

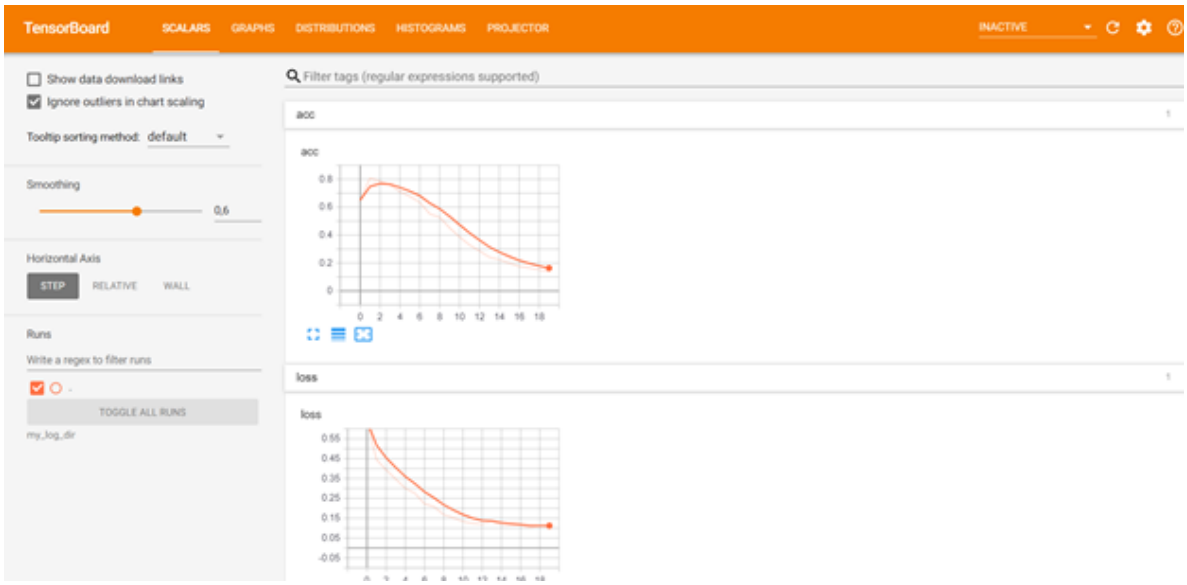
```
callbacks = [
    keras.callbacks.TensorBoard(
        log_dir='my_log_dir',
        histogram_freq=1,
        embeddings_freq=1,
        embeddings_data = x_train[:100],
    )
]

history = model.fit(x_train, y_train,
                    epochs=20,
                    batch_size=128,
                    validation_split=0.2,
```

```
                    callbacks=callbacks)
```

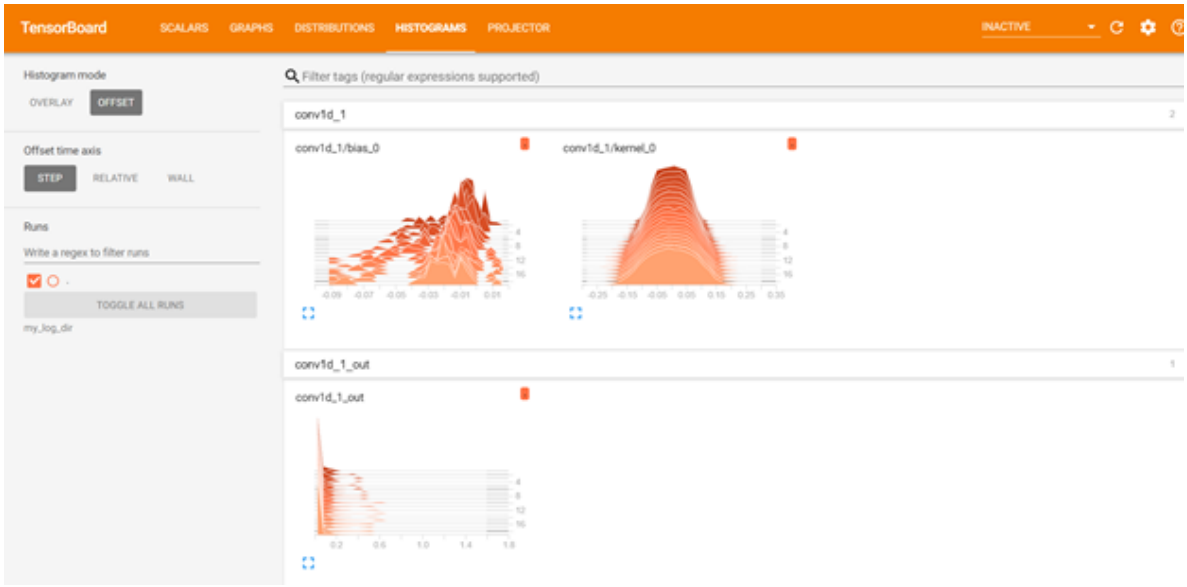
После этого можно запустить сервер TensorBoard из командной строки, указав, что тот должен читать журналы, которые в настоящий момент записывает обратный вызов. Утилита tensorboard должна автоматически установиться вместе с фреймворком TensorFlow: *tensorboard --**logdir=my log **dir*

После этого можно запустить браузер, перейти по адресу **http://localhost:6006** и посмотреть, как протекает процесс обучения модели.

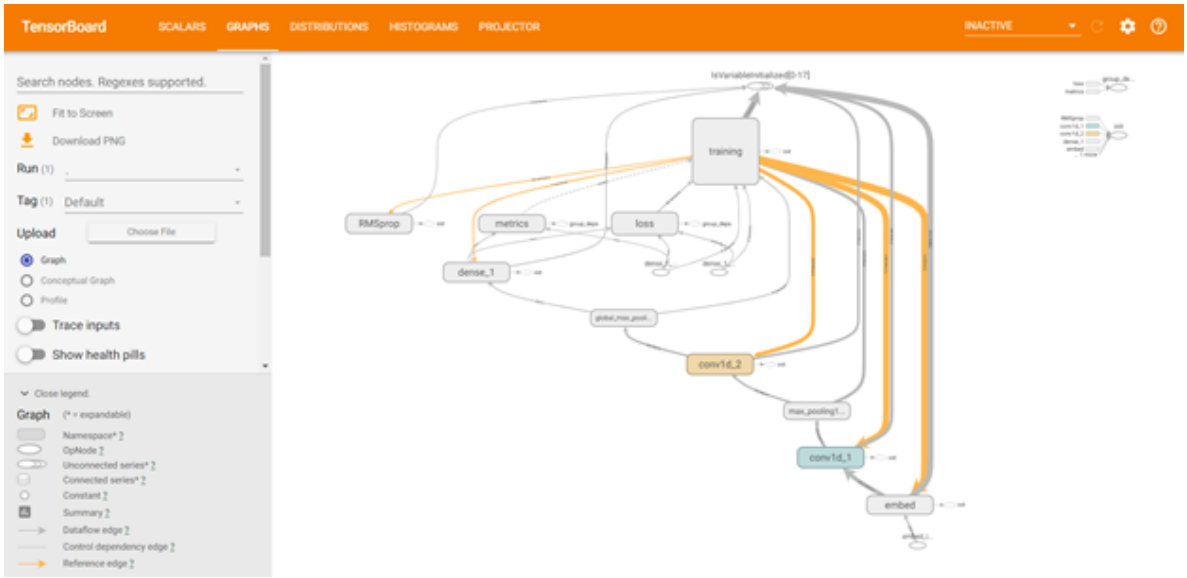
Мониторинг измнения метрик в ходе обучения



Гистограммы активаций



На вкладке Graphs (Диаграммы) изображены интерактивные диаграммы низкоуровневых операций, выполняемых фреймворком TensorFlow в ходе обучения модели Keras


TensorBoard
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
PROJECTOR
INACTIVE
Search nodes. Regexes supported.
Fit to Screen
Download PNG
Run (1)
Tag (1)
Default
Upload
Choose File
Graph
Conceptual Graph
Profile
Trace inputs
Show health pills
Close legend.
Graph (* = expandable)
Namespace* ?
OpNode ?
Unconnected series* ?
Connected series* ?
Constant ?
Summary ?
Dataflow edge ?
Control dependency edge ?
Reference edge ?
training
RMSprop
metrics
loss
dense_1
conv1d_2
conv1d_1
embed