TensorFlow

Основные функции и возможности TensorFlow

- Мультиплатформенность:
- Высокая производительность
- Гибкость
- Визуализация и отладка
- Обработка данных
- Готовые модели и инструменты

Поддержка различных моделей машинного обучения

- нейронные сети,
- рекуррентные сети,
- сверточные сети,
- градиентный бустинг,
- деревья решений
- и др.

Процесс создания моделей с использованием TensorFlow

- Импорт библиотек
- Загрузка данных
- Подготовка данных
- Определение модели
- Компиляция модели
- Обучение
- Оценка
- Прогнозирование

Функции потерь TensorFlow

- tf.keras.losses.MeanSquaredError
- tf.keras.losses.BinaryCrossentropy
- tf.keras.losses.CategoricalCrossentropy
- tf.keras.losses.SparseCategoricalCrossentropy
- tf.keras.losses.Hinge

Оптимизаторы TensorFlow

- tf.keras.optimizers.SGD
- tf.keras.optimizers.Adam
- tf.keras.optimizers.RMSprop
- tf.keras.optimizers.Adagrad
- tf.keras.optimizers.Adadelta

Процесс создания моделей с использованием TensorFlow

```
import tensorflow as tf
from tensorflow.keras import layers
```

```
(train_images, train_labels), (test_images, test_labels) =

                tf.keras.datasets.cifar10.load_data()
```

Процесс создания моделей с использованием TensorFlow

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

Процесс создания моделей с использованием TensorFlow

```
model = tf.keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10)
])
```

Процесс создания моделей с использованием TensorFlow

```
model.compile(optimizer='adam’,

loss=tf.keras.losses.SparseCategoricalCrossentropy(from
_logits=True),

metrics=['accuracy’]

)
```

Процесс создания моделей с использованием TensorFlow

```
model.fit(train_images, train_labels,
epochs=10, batch_size=32)


test_loss, test_acc = model.evaluate(test_images,
test_labels, verbose=2)

print('Test accuracy:', test_acc)
```

Процесс создания моделей с использованием TensorFlow

```
predictions = model.predict(test_images)
```

Использование TensorFlow для обработки изображений с помощью предварительно обученной модели

```
# Загружаем предварительно обученную модель
base_model =
tf.keras.applications.MobileNetV2(weights='imag
enet', include_top=False)

# Создаем новую модель на основе предварительно
обученной
model = tf.keras.models.Sequential([
base_model,
tf.keras.layers.GlobalAveragePooling2D(),
tf.keras.layers.Dense(num_classes,
activation='softmax')
])
```

```
# Замораживаем веса предварительно обученной модели
base_model.trainable = False

# Компилируем модель и загружаем данные для обучения
model.compile(optimizer=tf.keras.optimizers.Adam(),
loss='sparse_categorical_crossentropy',
metrics=['accuracy’])

# Обучаем модель на тренировочных данных
model.fit(x_train, y_train, epochs=10,
batch_size=32)
```

Обнаружение объектов на изображении с помощью модели TensorFlow Object Detection API

```
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as
viz_utils

# Загружаем предварительно обученную модель и метки классов

model = tf.saved_model.load('path_to_saved_model’)

label_map = label_map_util.load_labelmap('path_to_label_map’)

categories =
label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=5, use_display_name=True)

category_index =
label_map_util.create_category_index(categories)
```

```
# Загружаем изображение для обнаружения объектов
image_path = 'path_to_image’

image_np = cv2.imread(image_path)

input_tensor = tf.convert_to_tensor(image_np)

input_tensor = input_tensor[tf.newaxis, ...]

detections = model(input_tensor)

num_detections = int(detections.pop('num_detections’))

detections = {key: value[0, :num_detections].numpy() for
key, value in detections.items()}

detections['num_detections'] = num_detections
```

```
# Показываем обнаруженные объекты на изображении

viz_utils.visualize_boxes_and_labels_on_image_array(
image_np,
detections['detection_boxes’],

detections['detection_classes’],

detections['detection_scores’],

category_index,

use_normalized_coordinates=True,

max_boxes_to_draw=5,

min_score_thresh=0.5,

agnostic_mode=False)
```

```
cv2.imshow('Object Detection',
cv2.resize(image_np, (800, 600)))

cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
tf.keras.models.save_model(model,
'path/to/save/model.h5’)
```

```
h5 (Hierarchical Data Format)
TensorFlow SavedModel
TFLite
```

```
loaded_model =
tf.keras.models.load_model('path/to/save/model
.h5’)

predictions = loaded_model.predict([...])
```

Инструменты для визуализации

1. TensorBoard

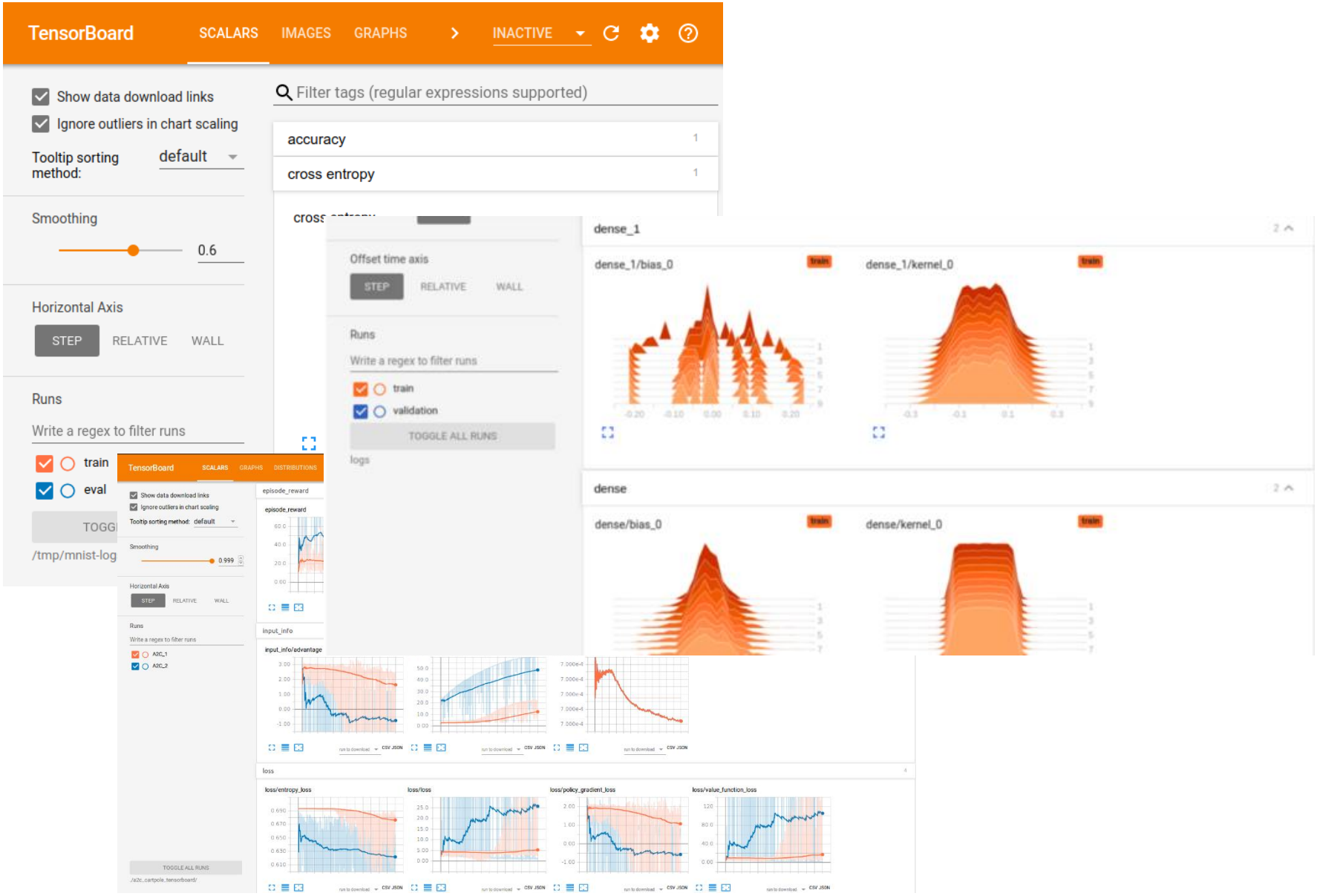
2. TensorFlow Datasets

3. TensorFlow Projector

4. TensorFlow Model Analysis

5. TensorFlow Lite Visualizer

6. TensorFlow Probability


TensorBoard
SCALARS
IMAGES
GRAPHS
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method:
default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
eval
/tmp/mnist-log
Filter tags (regular expressions supported)
accuracy
1
cross entropy
1
Offset time axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
validation
TOGGLE ALL RUNS
logs
dense_1
dense_1/bias_0
dense_1/kernel_0
dense
dense/bias_0
dense/kernel_0
episode_reward
input_info
input_info/advantage
loss
loss/entropy_loss
loss/loss
loss/policy_gradient_loss
loss/value_function_loss

```
# Создание модели
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])

# Компиляция модели
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Создание обратного вызова TensorBoard для записи журналов
log_dir = "logs/fit/"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,
histogram_freq=1)

# Обучение модели с использованием обратного вызова TensorBoard
model.fit(x_train, y_train, epochs=10, callbacks=[tensorboard_callback])
```

Открыть терминал и указать путь к проекту с моделью TensorFlow

Запустить **TensorBoard**, используя команду:
tensorboard --logdir logs/fit

Открыть браузер и перейти по ссылке, указанной в выводе команды, например: http://localhost:6006

```
import tensorflow as tf
from tensorflow.contrib.tensorboard.plugins import projector


# Загружаем набор данных MNIST и создаем TensorFlow сессию:

mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

sess = tf.Session()

# Преобразуем изображения в 2D массивы пикселей и нормализуем их
значения
x_train_2D = x_train.reshape(-1, 784)
x_test_2D = x_test.reshape(-1, 784)
x_train_normalized = x_train_2D / 255.0
x_test_normalized = x_test_2D / 255.0
```

```
# Создаем переменные для вектора признаков и меток классов:

features = tf.Variable(x_test_normalized, name='features')
labels = tf.constant(y_test, name='labels')


# Создаем проектор и указываем вектор признаков и метки
классов

projector = projector.ProjectorConfig()
embedding = projector.embeddings.add()
embedding.tensor_name = features.name
embedding.metadata_path = 'metadata.tsv'


# Запускаем TensorFlow сессию и инициализируем переменные:

sess.run(tf.global_variables_initializer())
```

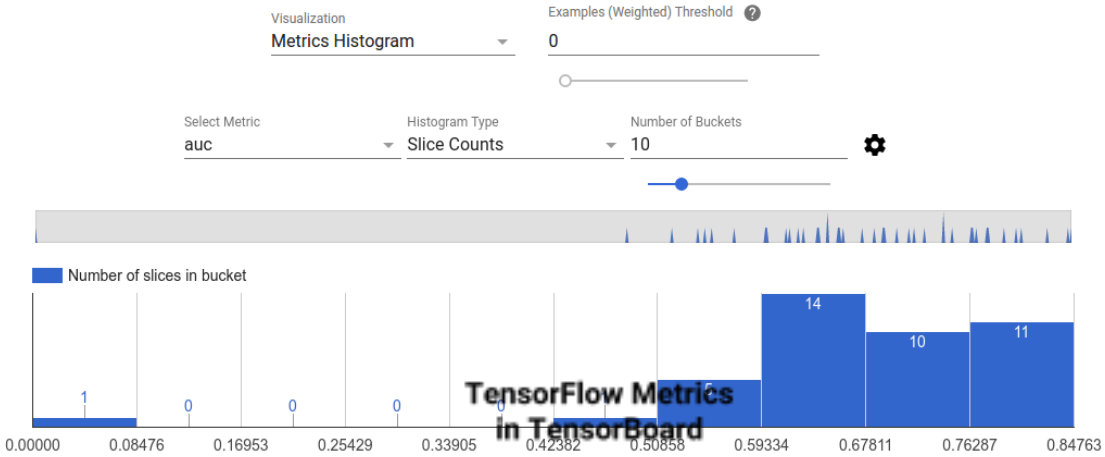
```
# Сохраняем метаданные (метки классов) в файл metadata.tsv

with open('metadata.tsv', 'w') as metadata_file:
    for label in y_test:
        metadata_file.write('{}\n'.format(label))


# Создаем TensorFlow.summary.FileWriter для сохранения данных в
TensorBoard

summary_writer = tf.summary.FileWriter('logdir')


# Сохраняем проектор и записываем вычисленные векторы признаков

projector.visualize_embeddings(summary_writer, projector)
saver = tf.train.Saver()
saver.save(sess, 'logdir/model.ckpt')
```

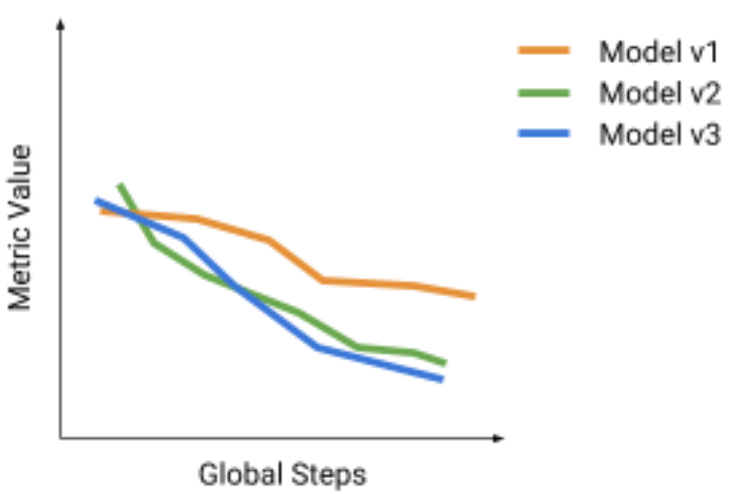
```
# Запускаем TensorBoard

tensorboard --logdir=logdir
```

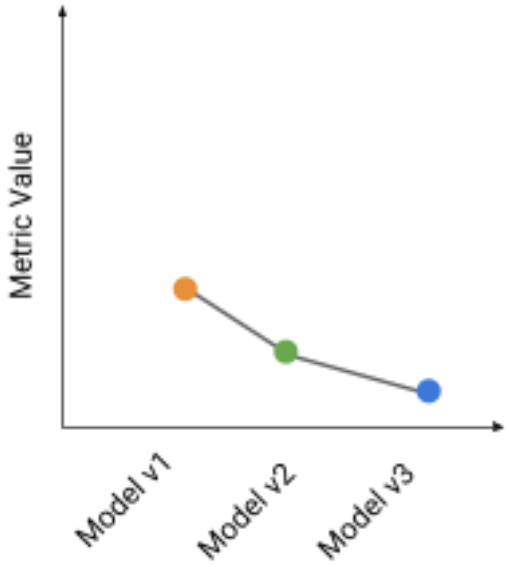
После запуска TensorBoard открыть веб-браузер и перейти по адресу localhost:6006.

В разделе "EMBEDDINGS" должны отображаться рукописные цифры, которые можно вращать и масштабировать


TensorBoard
SCALARS
GRAPHS
HPARAMS
TIME SERIES
PROJECTOR
DATA
17 runs found
3 tensors found
feature_vecs.tsv
Label by
label
Color by
No color map
Edit by
label
Tag selection as
Load
Download
Label
Sphereize data
UMAP
T-SNE
PCA
CUSTOM
X
Component #1
Y
Component #2
Z
Component #3
PCA is approximate.
Total variance described: 31.1%.
Points: 1500 | Dimension: 1280
Search
label
BOOKMARKS (0)


Visualization
Metrics Histogram
Examples (Weighted) Threshold
0
Select Metric
auc
Histogram Type
Slice Counts
Number of Buckets
10
Number of slices in bucket
1
0
0
0
14
10
11
0.00000
0.08476
0.16953
0.25429
0.33905
0.42382
0.50858
0.59334
0.67811
0.76287
0.84763
TensorFlow Metrics
in TensorBoard


Model v1
Model v2
Model v3
Metric Value
Global Steps


TensorFlow Metrics
in TensorFlow Model Analysis
Metric Value
Model v1
Model v2
Model v3