

 ilikeev 17 октября в 19:09

Нейронная сеть с использованием TensorFlow: классификация изображений

Python, Машинное обучение

Из песочницы

Привет, Хабр! Представляю вашему вниманию перевод статьи "[Train your first neural network: basic classification](#)".



Это руководство по обучению модели нейронной сети для классификации изображений одежды, таких как кроссовки и рубашки. Для создания нейронной сети используем python и библиотеку TensorFlow.

Установка TensorFlow

Для работы нам понадобятся следующие библиотеки:

1. numpy (в командной строке пишем: `pip install numpy`)
2. matplotlib (в командной строке пишем: `pip install matplotlib`)
3. keras (в командной строке пишем: `pip install keras`)
4. jupyter (в командной строке пишем: `pip install jupyter`)

С помощью pip: в командной строке пишем `pip install tensorflow`
Если у вас возникает ошибка, то можно [скачать .whl файл](#) и установить с помощью pip: `pip install путь_к_файлу\название_файла.whl`

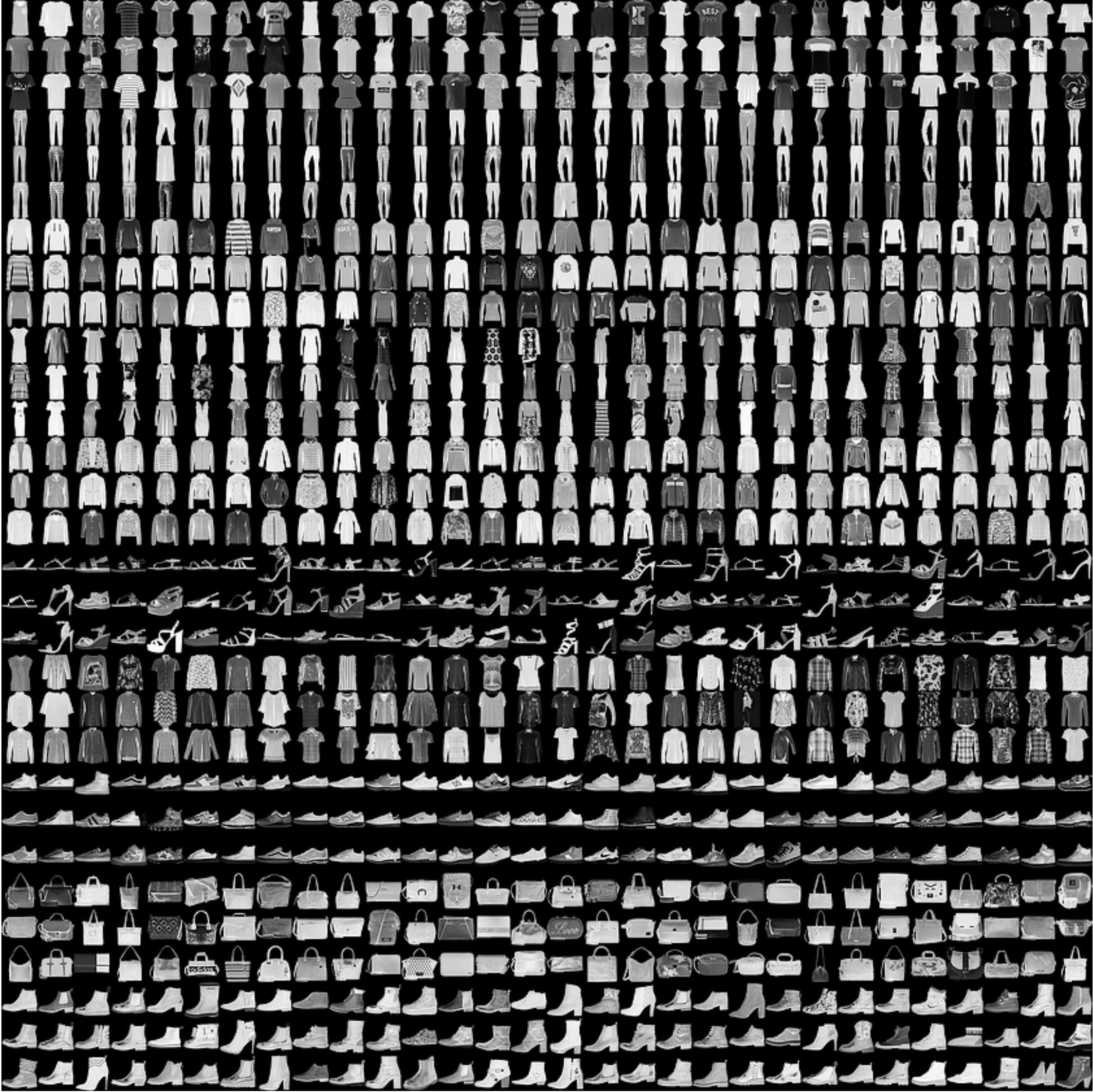
[Официальное руководство по установке TensorFlow \(на англ.\)](#)

Запускаем Jupyter. Для запуска в командной строке пишем `jupyter notebook`.

Начало работы

```
#Подключаем библиотеки
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
```

В этом руководстве используется набор данных Fashion MNIST, который содержит 70 000 изображений в оттенках серого в 10 категориях. На изображениях показаны отдельные предметы одежды с низким разрешением (28 на 28 пикселей):



Мы будем использовать 60 000 изображений для обучения сети и 10 000 изображений, чтобы оценить, насколько точно сеть научилась классифицировать изображения. Вы можете получить доступ к Fashion MNIST непосредственно из TensorFlow, просто импортировав и загрузив данные:

```
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

Загрузка набора данных возвращает четыре массива NumPy:

1. Массивы `train_images` и `train_labels` — это данные, которые использует модель для обучения
2. Массивы `test_images` и `test_labels` используются для тестирования модели

Изображения представляют собой NumPy массивы 28x28, значения пикселей которых находятся в диапазоне от 0 до 255. Метки (labels) представляют собой массив целых чисел от 0 до 9. Они соответствуют классу одежды:

Метка	Класс
0	T-shirt (Футболка)
1	Trouser (Брюки)
2	Pullover (Свитер)
3	Dress (Платье)
4	Coat (Пальто)
5	Sandal (Сандали)

6	Shirt (Рубашка)
7	Sneaker (Кроссовки)
8	Bag (Сумка)
9	Ankle boot (Ботильоны)

Имена классов не включены в набор данных, поэтому прописываем сами:

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Исследование данных

Рассмотрим формат набора данных перед обучением модели.

```
train_images.shape #В обучающем наборе имеется 60 000 изображений, каждое изображение представлено как 28 x 28 пикселей

test_images.shape #В тестовом наборе имеется 10 000 изображений, каждое изображение представлено как 28 x 28 пикселей

len(train_labels) #В учебном наборе 60 000 меток

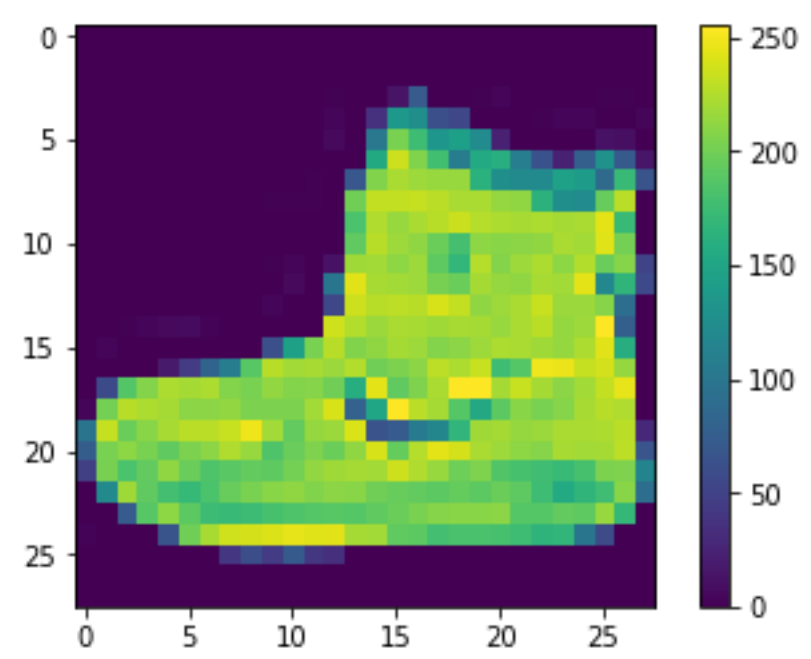
len(test_labels) #В тестовом наборе 10 000 меток

train_labels #Каждая метка представляет собой целое число от 0 до 9 (Показывается первые 3 метки и последние 3 метки)
```

Предварительная обработка данных

Перед подготовкой модели данные должны быть предварительно обработаны. Если вы проверите первое изображение в тренировочном наборе, вы увидите, что значения пикселей находятся в диапазоне от 0 до 255:

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
```



Мы масштабируем эти значения до диапазона от 0 до 1:

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

Отобразим первые 25 изображений из тренировочного набора и покажем имя класса под каждым изображением. Убедимся, что данные находятся в правильном формате.

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
```



```
plt.yticks([])
plt.grid(False)
plt.imshow(train_images[i], cmap=plt.cm.binary)
plt.xlabel(class_names[train_labels[i]])
```



Построение модели

Построение нейронной сети требует настройки слоев модели.

Основным строительным блоком нейронной сети является слой. Большая часть глубокого обучения состоит в объединении простых слоев. Большинство слоев, таких как `tf.keras.layers.Dense`, имеют параметры, которые изучаются во время обучения.

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

Первый слой в сети `tf.keras.layers.Flatten` преобразует формат изображений из 2d-массива (28 на 28 пикселей) в 1d-массив из $28 * 28 = 784$ пикселей. У этого слоя нет параметров для изучения, он только переформатирует данные.

Следующие два слоя это `tf.keras.layers.Dense`. Это плотно связанные или полностью связанные нейронные слои. Первый слой `Dense` содержит 128 узлов (или нейронов). Второй (и последний) уровень — это слой с 10 узлами `tf.nn.softmax`, который возвращает массив из десяти вероятностных оценок, сумма которых равна 1. Каждый узел содержит оценку, которая указывает вероятность того, что текущее изображение принадлежит одному из 10 классов.

Скомпилирование модели

Прежде чем модель будет готова к обучению, ей потребуется еще несколько настроек. Они добавляются во время этапа компиляции модели

- Loss function (функция потерь) — измеряет насколько точная модель во время обучения
- Optimizer (оптимизатор) — это то, как модель обновляется на основе данных, которые она видит, и функции потерь
- Metrics (метрики) — используется для контроля за этапами обучения и тестирования

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Обучение модели

Обучение модели нейронной сети требует следующих шагов:

1. Подача данных обучения модели (в этом примере — массивы train_images и train_labels)
2. Модель учится ассоциировать изображения и метки
3. Мы просим модель сделать прогнозы о тестовом наборе (в этом примере — массив test_images). Мы проверяем соответствие прогнозов меток из массива меток (в этом примере — массив test_labels)

Чтобы начать обучение, вызовите метод model.fit:

```
model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5
60000/60000 [=====] - 5s 90us/step - loss: 0.5001 - acc: 0.8253
Epoch 2/5
60000/60000 [=====] - 5s 88us/step - loss: 0.3761 - acc: 0.8644
Epoch 3/5
60000/60000 [=====] - 5s 89us/step - loss: 0.3384 - acc: 0.8759
Epoch 4/5
60000/60000 [=====] - 5s 89us/step - loss: 0.3124 - acc: 0.8857
Epoch 5/5
60000/60000 [=====] - 5s 86us/step - loss: 0.2946 - acc: 0.8922
```

При моделировании модели отображаются показатели потерь (loss) и точности (acc). Эта модель достигает точности около 0,88 (или 88%) по данным обучения.

Оценка точности

Сравним, как модель работает в тестовом наборе данных:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

```
10000/10000 [=====] - 1s 53us/step
Test accuracy: 0.8777
```

Оказывается, точность в тестовом наборе данных немного меньше точности в тренировочном наборе. Этот разрыв между точностью обучения и точностью тестирования является примером переобучения. Переобучение — это когда модель машинного обучения хуже работает с новыми данными, чем с данными обучения.

Прогнозирование

Используем модель для прогнозирования некоторых изображений.

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

Здесь модель предсказала метку для каждого изображения в тестовом наборе. Давайте посмотрим на первое предсказание:

```
predictions[0]
```

```
array([4.0652740e-06, 6.9819279e-08, 2.5388722e-06, 1.3390627e-07,
       1.1847248e-07, 2.9022932e-02, 2.0918555e-06, 6.4492501e-02,
       9.1468155e-06, 9.0646631e-01], dtype=float32)
```

Предсказание представляет собой массив из 10 чисел. Они описывают «уверенность» модели в том, что изображение соответствует каждому из 10 разных предметов одежды. Мы можем видеть, какая метка имеет наибольшее доверительное значение:

```
np.argmax(predictions[0]) #9
```

Таким образом, модель наиболее уверена в том, что это изображение — Ankle boot (Ботильоны), или class_names [9]. И мы можем проверить тестовую метку, чтобы убедиться, что это правильно:

```
test_labels[0]  #9
```

Напишем функции для визуализации этих предсказаний

```
def plot_image(i, predictions_array, true_label, img):
    predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])

    plt.imshow(img, cmap=plt.cm.binary)

    predicted_label = np.argmax(predictions_array)
    if predicted_label == true_label:
        color = 'blue'
    else:
        color = 'red'

    plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                         100*np.max(predictions_array),
                                         class_names[true_label]),
              color=color)

def plot_value_array(i, predictions_array, true_label):
    predictions_array, true_label = predictions_array[i], true_label[i]
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    thisplot = plt.bar(range(10), predictions_array, color="#777777")
    plt.ylim([0, 1])
    predicted_label = np.argmax(predictions_array)

    thisplot[predicted_label].set_color('red')
    thisplot[true_label].set_color('blue')
```

Давайте посмотрим на 0-е изображение, предсказания и массив предсказаний.

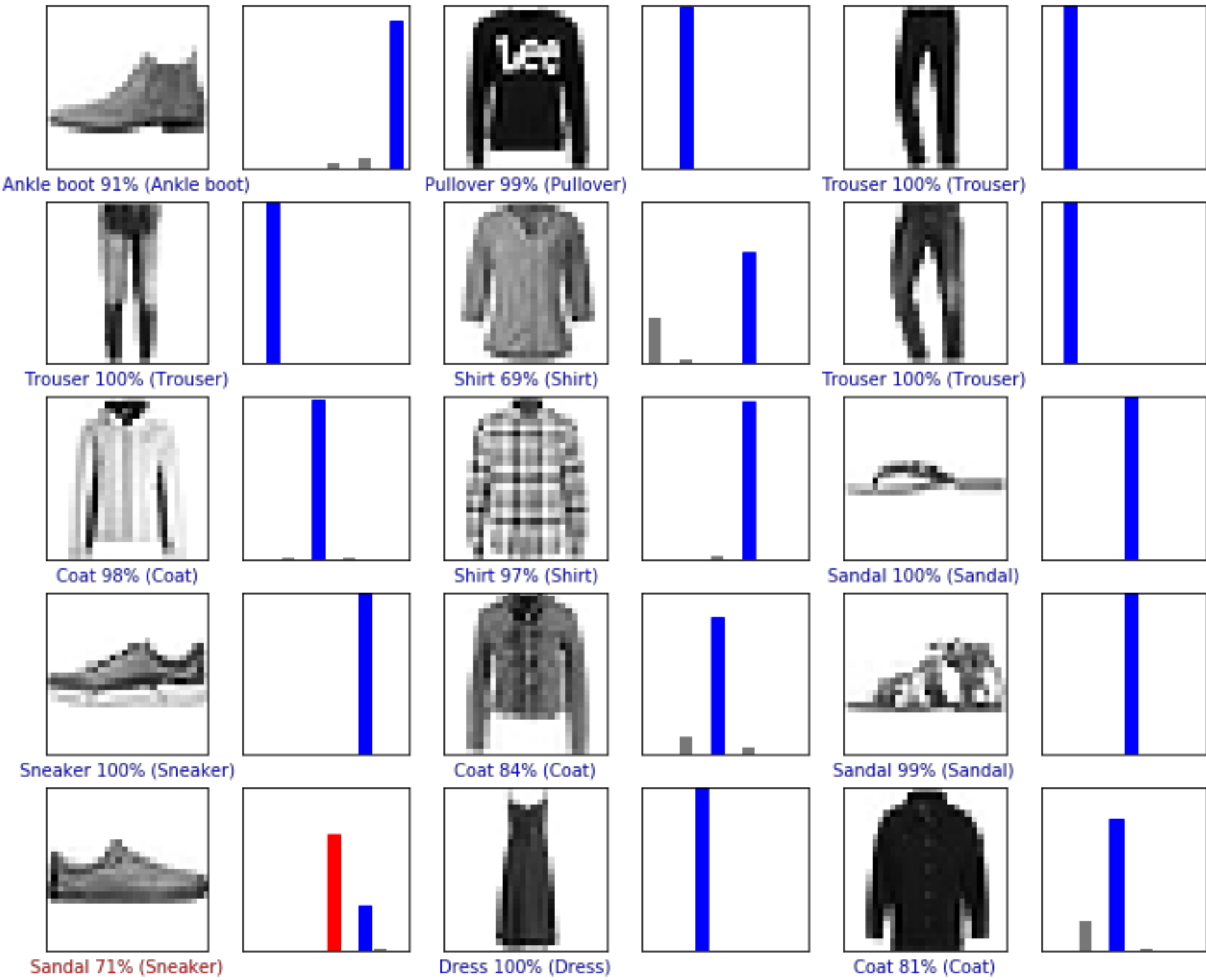
```
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions, test_labels)
```



Построим несколько изображений с их прогнозами. Правильные метки прогноза — синие, а неправильные метки прогноза — красные. Обратите внимание, что это может быть неправильно, даже когда он очень уверен.

```
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
```

```
plt.subplot(num_rows, 2*num_cols, 2*i+1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(num_rows, 2*num_cols, 2*i+2)
plot_value_array(i, predictions, test_labels)
```



Наконец, используем обученную модель, чтобы сделать предсказание об одном изображении.

```
# Возьмём изображение из тестового набора данных
img = test_images[0]
```

Модели `tf.keras` оптимизированы для того, чтобы делать прогнозы на пакеты (batch) или коллекции (collection). Поэтому, хотя мы используем одно изображение, нам нужно добавить его в список:

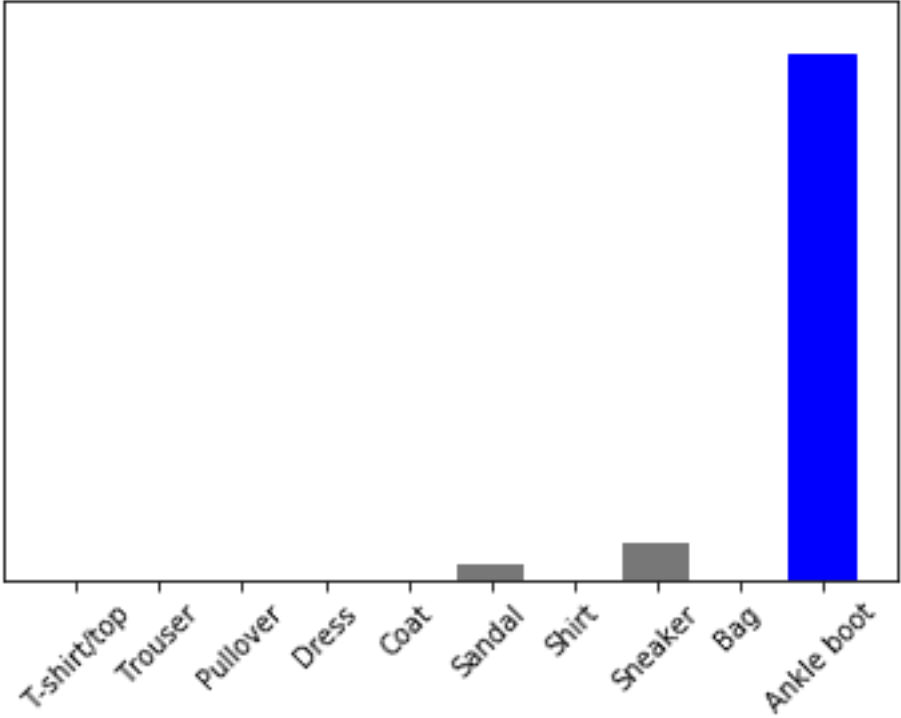
```
#Добавим изображение в пакет, где он является единственным членом
img = (np.expand_dims (img, 0))
```

Прогноз для изображения:

```
predictions_single = model.predict(img)
print(predictions_single)
```

```
[[4.0652740e-06  6.9819272e-08  2.5388672e-06  1.3390652e-07  1.1847247e-07
  2.9022938e-02  2.0918596e-06  6.4492591e-02  9.1468228e-06  9.0646625e-01]]
```

```
plot_value_array(0, predictions_single, test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
```



```
np.argmax(predictions_single[0])
```

Как и прежде, модель предсказывает метку 9.

Если есть вопросы, пишите в комментариях или в личные сообщения.

Метки: [перевод](#), [машинное обучение](#), [нейронная сеть](#), [python](#), [tensorflow](#)

↑

+32

↓

🔖


199

👁

6k

💬

14



11,0

Карма

25,6

Рейтинг

0

Подписчики

@ilikeev

Пользователь

ПОХОЖИЕ ПУБЛИКАЦИИ

1 июля 2016 в 10:02

Драм-машина на нейронной сети

↑

+11

👁

11,1k

🔖

64

💬

17

13 августа 2015 в 13:52

Аугментация (augmentation, “раздутие”) данных для обучения нейронной сети на примере печатных символов

↑

+9

👁

13,6k

🔖

82

💬

11

13 января 2015 в 09:39

Введение в машинное обучение с помощью Python и Scikit-Learn

↑

+47

👁

210k

🔖


906

💬

13

ВАКАНСИИ


Мой круг



Python Programmer

Poteha Labs • Возможна удаленная работа


от 60 000 до 200 000 ₽



Старший аналитик

Out of Cloud • Москва


от 150 000 ₽



Technical lead ML/CV

Facechain • Санкт-Петербург

от 200 000 до 300 000 ₽




Software Engineer - Computer Vision

AIFactory.app • Санкт-Петербург

от 100 000 до 180 000 ₽

Комментарии 14

- 

Stas911


17.10.18 в 20:48

#

🔖

Вот тут говорят, что ставить ТФ через PIP не очень хорошо для производительности: towardsdatascience.com/stop-installing-tensorflow-using-pip-for-performance-sake-5854f9d9eb0c

↑

+2
- 

Dark_Daiver

18.10.18 в 08:38

#


🔖

📄

👤

Учить на CPU боль, так или иначе. А для GPU версии tf MKL не используется, как я понимаю

↑

+1
- 

ilikeev

18.10.18 в 17:02

#


🔖

📄

👤

Спасибо за замечание. Можно скачать Anaconda с [официального сайта](#). Для установки TensorFlow используем команду: `conda install tensorflow`

↑

0
- 

Tihon_V

18.10.18 в 17:21

#


🔖

📄

👤

Для AMD (CGN1.3 и выше) — можно использовать [это](#).

↑

+1
- 

Daniyar94


18.10.18 в 02:07

#

🔖

Для тех кто хочет попробовать сам, есть MNIST для рукописных цифр. Подход точно такой же

↑

+1
- 

perfect_genius


18.10.18 в 12:38

#

🔖

Не разбираюсь в нейросетях, но почему нельзя их обучать без программирования? Вон как DeepFakes App — есть интерфейс, скармливаешь данные обучаешь... Т.е. что мешает вывести нейросети в массы?

↑

0
- 

Tyler

18.10.18 в 16:06

#


🔖

📄

👤

можно. кнопочки накидал на форму и связал их со строчками кода.
просто зачем это надо.
там программирования как такового и нет, просто кодом быстрее: накидать сетку, поправить параметры, вывести картинку и тд. Чем для каждой такой мимолетной хотелки, кнопку рисовать.
ну... они и так уже в массах давно уже.

↑

0
- 

Akon32

18.10.18 в 16:16

#


🔖

📄

👤

Во многих случаях (за исключением банальных) подготовка данных нетривиальна. Проще всего её сделать, написав какой-то не очень сложный код. Универсальный графический интерфейс для обучения (под все возможные задачи) сделать пока невозможно.

↑

0
- 

perfect_genius

18.10.18 в 16:57

#


🔖

📄

👤

Дать нейросети выявить общие признаки в пачке изображений и потом распознавать их в новых — на такое уже есть интерфейс?

↑

0
- 

Akon32

19.10.18 в 12:24

#


🔖

📄

👤

Может, для такой задачи и есть.

↑

+1
- 

perfect_genius

21.10.18 в 16:34

#

🔖

📄

👤

Но чаще набор данных нужно разметить таким образом, чтобы выявление признаков было удобно для нейросети. Можно, конечно, и не стараться, но сеть не обучится или будет обучаться медленно. Разметку удобнее делать программно.
Например, при обучении AlphaGo использовались слои с признаками "цвет камня" (тут всё понятно), "число степеней свободы", "захват группы" (а вот тут простой алгоритм справится лучше, чем нейросеть) и т.д.
В нейросетях для сегментации (типа Mask R-CNN) нужно разрезать изображение на фрагменты, масштабировать их и распределить по классам, а затем на фрагментах применять нейросеть. При обучении сети программно сделать это намного удобнее, а на этапе работы сети — просто необходимо.
Вариантов программной обработки слишком много, чтобы для всех их сделать GUI. Для следующей (новой) задачи жёстко заданный GUI, вероятно, будет бесполезен.

↑

0

Наверно, знаете вот такое от Гугла: teachablemachine.withgoogle.com
Обучается хорошо, но хотелось таки подсунуть свои картинки. Спецпрограммой подсунул их и обучил различать художника картинки.
Жаль что нет десктопной версии чего-то подобного.

 **uhf** 18.10.18 в 20:54  

↑ +2

Думаю, стоит упомянуть, что tensorflow с поддержкой GPU (CUDA) ставится так: `pip install tensorflow-gpu`

 **Vinci** 22.10.18 в 09:30  

↑ 0

- 1 — указывайте что это перевод.
- Это ноутбук с сайта документации tensorflow.
- 2 — Вы собираетесь переводитть всю их документацию (по крайней мере вводные мануалы)? Это было бы круто.

 **ilikeev** 22.10.18 в 12:25    

↑ 0

Здравствуйте. В первой строке я указал что это перевод статьи [Train your first neural network: basic classification](#)


Только [полноправные пользователи](#) могут оставлять комментарии. [Войдите](#), пожалуйста.


САМОЕ ЧИТАЕМОЕ


- Сутки
- Неделя
- Месяц

Red Hat будет поглощен IBM

↑ +75

 32k

 35

 196

Как Яндекс пытался скопировать мой сервис тепловых карт

↑ +264


 47,6k


 76


 201

Мессенджеры vs соцсети vs ... — анонс нового проекта

↑ +44

 24k

 64

 158

Памятка по разновидностям фишинга

↑ +42

 13,5k

 127

 20

Первокурснику: Вуз.Инструкция 1.0

↑ +60

 16,8k

 113

 114

Аккаунт

- Войти
- Регистрация

Разделы

- Публикации
- Хабы
- Компании
- Пользователи
- Песочница

Информация

- Правила
- Помощь
- Документация
- Соглашение
- Конфиденциальность

Услуги

- Реклама
- Тарифы
- Контент
- Семинары

Приложения

 Загрузите в **App Store**

 доступно в **Google Play**

