Федеральное государственное автономное образовательное

учреждение высшего образования «Московский государственный институт

международных отношений (университет)

Министерства иностранных дел Российской Федерации»

Факультет финансовой экономики

Кафедра цифровой экономики и искусственного интеллекта

РАБОТА

Направление подготовки Бизнес-информатика

Профиль (магистерская программа) Искусственный интеллект

Тема: «Распознавание изображений»

Выполнил:

студент(ка) Глоба Виктория

2 курс

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись студента

|  |  |
| --- | --- |
|  | Преподаватель:  Якушин Алексей Валерьевич  Кандидат пед. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  /подпись научного руководителя/ |

Москва – 2022

Оглавление

1. Анализ предметной области
2. Постановка задачи
3. Описание метода машинного обучения, который используется
4. Техническое задание
5. Текст программы с комментариями
6. Контрольные примеры работы программы (10 шт,)
7. Выводы

**Анализ предметной области**

E-commerce рынок развивается стремительными темпами. Крупные маркеплейсы развивают портфель товаров и поставщиков. Конкуренция в области продаж и борьбы за клиента возрастает, так как одни и те же товары могут быть предоставлены различными поставщиками.

Качественная категоризация товаров крайне важна для крупных маркетплейсов типа Amazon, Yoox Net-A-Porter и т.д. Текущий процесс размещения товаров выполняется сотрудниками компании вручную, либо на основании заполненных профилей, предоставленных поставщиками. При этом может быть осуществлена ошибка, которая в итоге может привести к тому, что товар окажется не в той категории и его поиск потенциальным покупателем будет неэффективен.

Для размещения товаров к каталоге необходимо определить категоризацию товаров, чтобы подобные товары попадали в одинаковые каталоги. А также чтобы избежать Fraud cо стороны поставщиков. Например, один и тот же товар размещен в различных категориях, продажа более дешевого в категории дорогого и т.д.

Для размещения товаров по фото в соответствующие категории может быть использована система на базе ML алгоритма.

**Постановка задачи**

Заказчик является крупнейшей компанией он-лайн ритейла одежды для мужчин и женщин. Товары на сайте одежды доступны в более чем в 100 странах, на 10 языках и в 7 валютах, компания обеспечивает удобство покупок на мобильных устройствах, планшетах и ​​компьютерах, простой возврат и многоязычную поддержку клиентов 24 часа в сутки, 7 дней в неделю, 365 дней в году.

На вход система получает набор данных [Fashion MNIST](https://github.com/zalandoresearch/fashion-mnist) который содержит 70,000 монохромных изображений в 10 категориях. На каждом изображении содержится по одному предмету одежды в низком разрешении (28 на 28 пикселей).

На выходе система должна определить к какой категории товаров относится изображение. При этом автоматически должен использоваться алгоритм обучения, предлагающий наибольшую точность с предложенным датасетом.

**Техническое задание**

**1 Общие сведения**

**1.1 Полное наименование системы и ее условное обозначение**

Полное наименование системы – «Программное обеспечение автоматизации распознавания изображений одежды и категоризации их». Краткое наименование - ПО «Распознавание изображений».

**1.2 Наименование предприятия (объединения) разработчика, заказчика (пользователя) разработки и их реквизиты**

Заказчик системы: Якушин Алексей Валерьевич

Разработчик системы: Глоба Виктория Александровна

**1.3 Перечень документов, на основании которых создается разработка**

Данная система создается на основании задания на выполнение и заявки заказчика.

**1.4 Плановые сроки начала и окончания работы по созданию системы**

Начало работ по созданию: 1.12.2021

Окончание работ по созданию: 20.01.2022

**1.5 Порядок оформления и предъявления заказчику результатов работ по созданию разработки программного комплекса**

Результаты работ по созданию работы оформляются в виде документа Word и предъявляются заказчику после защиты перед комиссией.

**2 Назначение и цели создания ПО «Распознавание изображений»**

**2.1 Назначение ПО «Распознавание изображений»**

Система разрабатывается и предназначена для категоризации одежды по фото

**2.2 Цели и задачи создания ПО «Распознавание изображений»»**

Основной целью создания ПО «Распознавание изображений» является категоризация одежды по фото с помощью алгоритма машинного обучения.

В соответствии с поставленной целью, создаваемая система должна уметь решать следующие задачи:

* Выполнять категоризацию товаров по фото с точностью работы системы не менее 85%

**3 Характеристика объекта автоматизации**

Современная конфигурация вычислительной техники и программное обеспечение все больше усовершенствуется, поэтому растет спрос рынка средств вычислительной техники.

Независимо от сложности компьютера и периферийного оборудования, процедура категоризации товаров состоит из четырех основных этапов:

1. Предобработка данных перед обучением нейросети
2. Компиляция модели
3. Обучение модели  ассоциированию изображений с правильными классами
4. Предсказание классов для новых данных

**4 Требования к системе**

**4.1 Требования к системе в целом**

**4.1.1 Требования к структуре и функционированию ПО «Распознавание изображений»**

**4.1.1.1 Перечень подсистем**

Вся система должна состоять из 3х подсистем:

- подсистема ввода и обработки исходных данных;

- подсистема обучения модели;

- подсистема оценки точности;

-  подсистема формирования результата.

* + - 1. **Требования к функционированию ПО «Распознавание изображений»**

Общение пользователя с системой должно происходить путем сохранения изображений с соответсвующие папки на компьютере сотрудника. Оттуда данные будут импортироваться в систему для дальнейшей обработки и определения категорий тора по фото.

Всего существует 10 возможных категорий товаров:

|  |  |
| --- | --- |
| **Label** | **Class** |
| 0 | T-shirt/top |
| 1 | Trouser |
| 2 | Pullover |
| 3 | Dress |
| 4 | Coat |
| 5 | Sandal |
| 6 | Shirt |
| 7 | Sneaker |
| 8 | Bag |
| 9 | Ankle boot |

Для корректного функционирования ПО «Распознавание изображений» необходимо наличие облачного хранилища на базе облачной платформы Azure и алгоритм на языке Python.

**4.1.2 Требования к численности и квалификации работников и режиму его работы**

С системой работает один или несколько человек, которые должны обладать знаниями и навыками работы с персональным компьютером, компонентами ПК, иметь знания по технике безопасности при работе с электроустановками, иметь навыки работы в среде Azure Microsoft IBM.

Режим работы персонала определяется предприятием, эксплуатирующим систему, и должен соответствовать требованиям по эргономике и безопасности труда, определенных в стандартах.

**4.1.3 Требования к безопасности**

Требования по безопасности должны соответствовать разделу 2 ГОСТ 24.104-85. Неправильные действия персонала не должны приводить к аварийной ситуации. Требования по безопасности средств вычислительной техники должны соответствовать ГОСТ 25.861-83. Все внешние элементы технических средств, находящиеся под напряжением, должны иметь защиту от случайного прикосновения, а сами технические средства иметь защитное заземление в соответствии с ГОСТ 12.1.030-81 и “Правилами устройства электроустановок”.

**4.1.4 Требования к эксплуатации, техническому обслуживанию, и ремонту**

Пользователь, работающий с системой, должен обладать знаниями и навыками работы с персональным компьютером, ИБП, должен знать правила техники безопасности при работе с электроустановками, иметь навыки работы с данной системой.

Режим эксплуатации системы круглосуточный, с учетом специфики работы.

Для размещения комплекса технических средств и персонала необходима площадь не менее 2 кв.м.

Регламент обслуживания технических средств предусматривает проведение профилактических обследований и ремонтов; замену вышедших из строя деталей.

**4.1.5 Требования по сохранности информации при авариях**

В случае аварий информация должна сохраняться. С этой целью используется система создания резервных копий, обеспечение безопасности хранимых данных от несанкционированного доступа.

**4.2 Требования к функциям и задачам ПО «Распознавание изображений»**

В состав разрабатываемой системы будут входить следующие подсистемы:

- подсистема, которая тренирует модель на датасете. Результат работы системы- натренированная нейронная сеть, которая будет использоваться в real time для класификации фото

- Web server, который принимает запросы из браузера со всеми фото, которые планирует классифицировать пользователь.

- yейронная сеть, которая выполняет вычисление prediction.

- html таблица, которую можно открыть в браузере

**4.3 Требования к видам обеспечения**

**4.3.1 Требования к математическому обеспечению**

Для реализации функций системы будет использоваться современные системы машинного обучения с использованием нейронных сетей.

**4.3.2 Требования к информационному обеспечению**

Входными данными будут являться изображения товаров, которые сохранены в соответствующую папку.

Выходными данными являются: присвоении категории товара каждому изображению.

**4.3.3 Требования к программному обеспечению**

Для успешного проектирования и эксплуатации системы необходимы следующие минимальные требования к программному обеспечению:

- операционная система Linux и выше;

- браузер Chrome 97.0.4692.71;

- Python 3 и выше.

**4.3.4 Требования к техническому обеспечению**

Для успешного функционирования системы необходимы следующие минимальные требования к техническому обеспечению:

- 64 giga ram

- 12 16 cpus

- 1tb disk drive (ssd)

- Windows

**5 Состав и содержание работ по созданию системы**

Разработка ПО «Распознавание изображений» должна содержать следующие этапы:

* обзор и анализ существующих методов и систем;
* разработка общесистемных решений;
* разработка информационного обеспечения;
* разработка математического обеспечения;
* разработка программного обеспечения;
* разработка технического обеспечения;
* разработка организационного обеспечения;
* оформление документов.

Эти работы должны быть выполнены в срок с 1.12.2021 по 20.01.2022.

**6 Порядок контроля и приемки разработки**

Контроль и приемка работы должна происходить на 10 контрольных примерах, обеспечивающих просмотр работоспособности всех функций разработки. Готовая разработка представляется преподавателю и сопровождается отчетом.

**7 Требования к документированию**

Пояснительная записка к работе должна содержать совокупность документов, структура и содержание которых регламентируется руководящим документом по стандартизации РД 50-34.698-90 (введенным 01.01.92) МЕТОДИЧЕСКИЕ УКАЗАНИЯ. ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ. АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ. ТРЕБОВАНИЯ К СОДЕРЖАНИЮ ДОКУМЕНТОВ.

Схемы алгоритмов, программ данных и систем, условные обозначения и правила выполнения регламентируются ГОСТ 19.701-90.

Общие требования к текстовым документам регламентируются ГОСТ 2.105-95

Техническое задание на создание автоматизированной системы выполняется на основании ГОСТ 34.602-89.

**Текст программы с комментариями**

import tensorflow as tf

from tensorflow import keras

import numpy as np

import matplotlib.pyplot as plt

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

print(tf.\_\_version\_\_)

fashion\_mnist = keras.datasets.fashion\_mnist

(train\_images, train\_labels), (test\_images, test\_labels) = fashion\_mnist.load\_data()

class\_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',

'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

print(train\_images.shape)

print("Train labels")

print(len(train\_labels))

print(train\_labels)

print(test\_images.shape)

print(len(test\_labels))

print("----")

plt.imshow(train\_images[0])

print(train\_labels[0])

plt.colorbar()

plt.grid(False)

plt.show()

2.6.2

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz>

32768/29515 [=================================] - 0s 0us/step

40960/29515 [=========================================] - 0s 0us/step

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz>

26427392/26421880 [==============================] - 0s 0us/step

26435584/26421880 [==============================] - 0s 0us/step

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz>

16384/5148 [===============================================================================================] - 0s 0us/step

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz>

4423680/4422102 [==============================] - 0s 0us/step

4431872/4422102 [==============================] - 0s 0us/step

(60000, 28, 28)

Train labels

60000

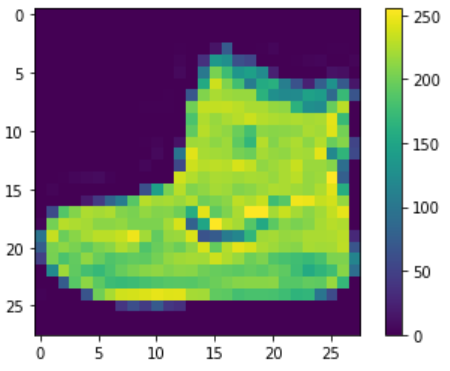
[9 0 0 ... 3 0 5]

(10000, 28, 28)

10000

----

9



train\_images = train\_images / 255.0

test\_images = test\_images / 255.0

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]])

plt.show()

model = keras.Sequential([

keras.layers.Flatten(input\_shape=(28, 28)),

keras.layers.Dense(128, activation='relu'),

keras.layers.Dense(10, activation='softmax')

])

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

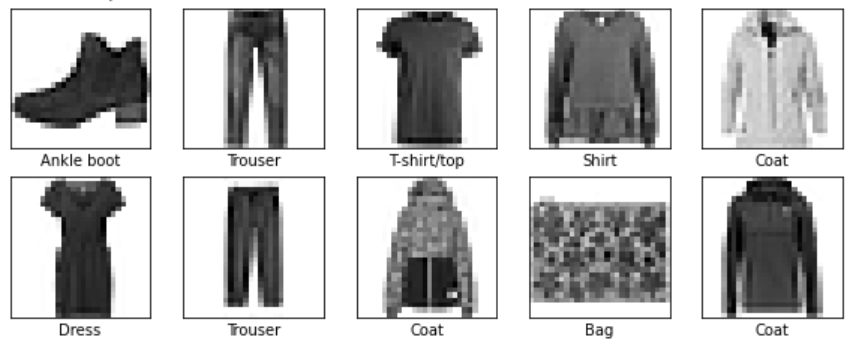
metrics=['accuracy'])

model.fit(train\_images, train\_labels, epochs=10)

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels, verbose=2)

print('\nТочность на проверочных данных:', test\_acc)





User settings:

KMP\_AFFINITY=granularity=fine,verbose,compact,1,0

KMP\_BLOCKTIME=0

KMP\_SETTINGS=1

KMP\_WARNINGS=0

Effective settings:

KMP\_ABORT\_DELAY=0

KMP\_ADAPTIVE\_LOCK\_PROPS='1,1024'

KMP\_ALIGN\_ALLOC=64

KMP\_ALL\_THREADPRIVATE=128

KMP\_ATOMIC\_MODE=2

KMP\_BLOCKTIME=0

KMP\_CPUINFO\_FILE: value is not defined

KMP\_DETERMINISTIC\_REDUCTION=false

KMP\_DEVICE\_THREAD\_LIMIT=2147483647

KMP\_DISP\_NUM\_BUFFERS=7

KMP\_DUPLICATE\_LIB\_OK=false

KMP\_ENABLE\_TASK\_THROTTLING=true

KMP\_FORCE\_REDUCTION: value is not defined

KMP\_FOREIGN\_THREADS\_THREADPRIVATE=true

KMP\_FORKJOIN\_BARRIER='2,2'

KMP\_FORKJOIN\_BARRIER\_PATTERN='hyper,hyper'

KMP\_GTID\_MODE=3

KMP\_HANDLE\_SIGNALS=false

KMP\_HOT\_TEAMS\_MAX\_LEVEL=1

KMP\_HOT\_TEAMS\_MODE=0

KMP\_INIT\_AT\_FORK=true

KMP\_LIBRARY=throughput

KMP\_LOCK\_KIND=queuing

KMP\_MALLOC\_POOL\_INCR=1M

KMP\_NUM\_LOCKS\_IN\_BLOCK=1

KMP\_PLAIN\_BARRIER='2,2'

KMP\_PLAIN\_BARRIER\_PATTERN='hyper,hyper'

KMP\_REDUCTION\_BARRIER='1,1'

KMP\_REDUCTION\_BARRIER\_PATTERN='hyper,hyper'

KMP\_SCHEDULE='static,balanced;guided,iterative'

KMP\_SETTINGS=true

KMP\_SPIN\_BACKOFF\_PARAMS='4096,100'

KMP\_STACKOFFSET=64

KMP\_STACKPAD=0

KMP\_STACKSIZE=8M

KMP\_STORAGE\_MAP=false

KMP\_TASKING=2

KMP\_TASKLOOP\_MIN\_TASKS=0

KMP\_TASK\_STEALING\_CONSTRAINT=1

KMP\_TEAMS\_THREAD\_LIMIT=4

KMP\_TOPOLOGY\_METHOD=all

KMP\_USE\_YIELD=1

KMP\_VERSION=false

KMP\_WARNINGS=false

OMP\_AFFINITY\_FORMAT='OMP: pid %P tid %i thread %n bound to OS proc set {%A}'

OMP\_ALLOCATOR=omp\_default\_mem\_alloc

OMP\_CANCELLATION=false

OMP\_DEFAULT\_DEVICE=0

OMP\_DISPLAY\_AFFINITY=false

OMP\_DISPLAY\_ENV=false

OMP\_DYNAMIC=false

OMP\_MAX\_ACTIVE\_LEVELS=1

OMP\_MAX\_TASK\_PRIORITY=0

OMP\_NESTED: deprecated; max-active-levels-var=1

OMP\_NUM\_THREADS: value is not defined

OMP\_PLACES: value is not defined

OMP\_PROC\_BIND='intel'

OMP\_SCHEDULE='static'

OMP\_STACKSIZE=8M

OMP\_TARGET\_OFFLOAD=DEFAULT

OMP\_THREAD\_LIMIT=2147483647

OMP\_WAIT\_POLICY=PASSIVE

KMP\_AFFINITY='verbose,warnings,respect,granularity=fine,compact,1,0'

Epoch 1/10

1875/1875 [==============================] - 5s 2ms/step - loss: 0.5046 - accuracy: 0.8232

Epoch 2/10

1875/1875 [==============================] - 4s 2ms/step - loss: 0.3768 - accuracy: 0.8648

Epoch 3/10

1875/1875 [==============================] - 5s 3ms/step - loss: 0.3390 - accuracy: 0.8757

Epoch 4/10

1875/1875 [==============================] - 4s 2ms/step - loss: 0.3121 - accuracy: 0.8852

Epoch 5/10

1875/1875 [==============================] - 4s 2ms/step - loss: 0.2985 - accuracy: 0.8907

Epoch 6/10

1875/1875 [==============================] - 5s 2ms/step - loss: 0.2845 - accuracy: 0.8946

Epoch 7/10

1875/1875 [==============================] - 5s 2ms/step - loss: 0.2713 - accuracy: 0.8992

Epoch 8/10

1875/1875 [==============================] - 5s 2ms/step - loss: 0.2626 - accuracy: 0.9027

Epoch 9/10

1875/1875 [==============================] - 4s 2ms/step - loss: 0.2515 - accuracy: 0.9060

Epoch 10/10

1875/1875 [==============================] - 5s 3ms/step - loss: 0.2430 - accuracy: 0.9087

313/313 - 1s - loss: 0.3334 - accuracy: 0.8826

Точность на проверочных данных: 0.8826000094413757

predictions = model.predict(test\_images)

i = 0

img = test\_images[i]

print(img.shape)

img = (np.expand\_dims(img,0))

print(img.shape)

predictions\_single = model.predict(img)

print(predictions\_single[0])

plot\_value\_array(0, predictions\_single, test\_labels)

\_ = plt.xticks(range(10), class\_names, rotation=45)

np.argmax(predictions\_single[0])

(28, 28)

(1, 28, 28)

[4.4344966e-07 1.5697911e-08 2.8105960e-09 1.1634359e-09 3.4340117e-09

8.5656578e-04 7.0517285e-07 4.4118538e-03 1.1145851e-07 9.9473023e-01]

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(test\_images[i], cmap=plt.cm.binary)

plt.xlabel(class\_names[test\_labels[i]])

plt.show()

for i in range(25):

print("Index", " ", i, " ", predictions[i], np.argmax(predictions[i]), class\_names[test\_labels[i]], class\_names[np.argmax(predictions[i])])

print(predictions[0])

print(np.argmax(predictions[0]))

print(test\_labels[0])

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')

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)

plt.show()

i = 12

plt.figure(figsize=(6,3))

plt.subplot(1,2,1)

plot\_image(i, predictions, test\_labels, test\_images)

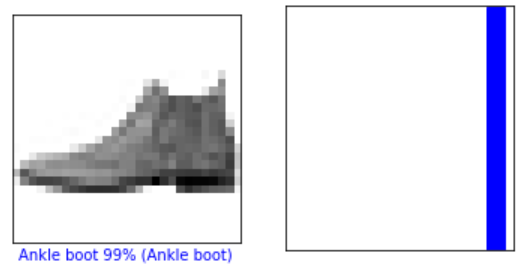
plt.subplot(1,2,2)

plot\_value\_array(i, predictions, test\_labels)

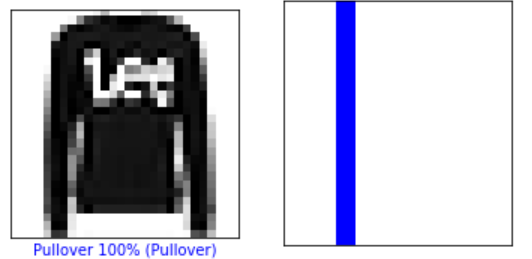
plt.show()

**Контрольные примеры работы программы**

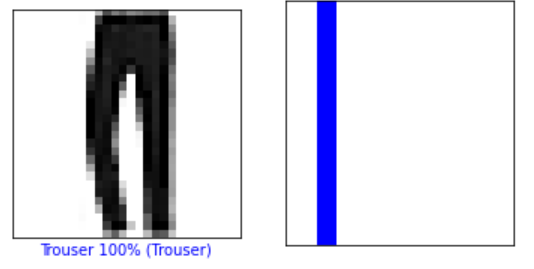
Пример 1:



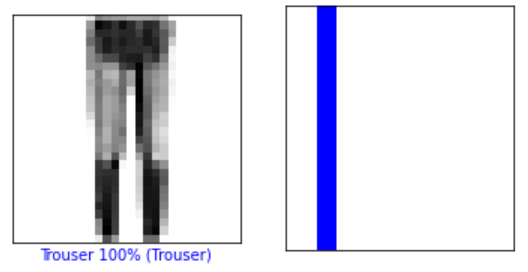
Пример 2:



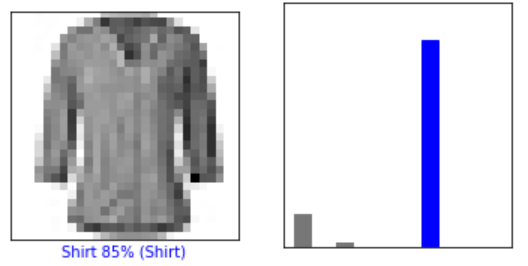
Пример 3:



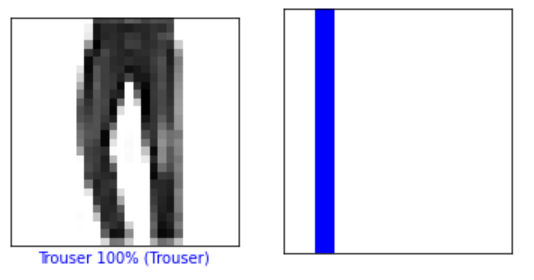
Пример 4:



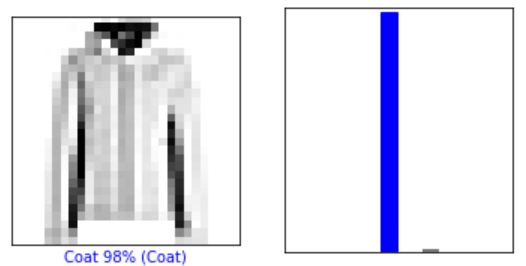
Пример 5:



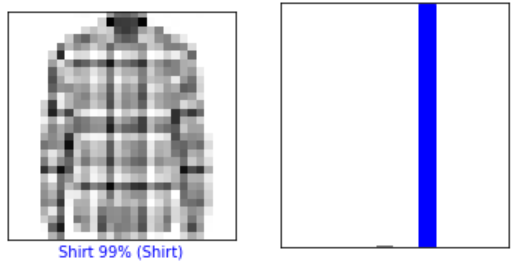
Пример 6:



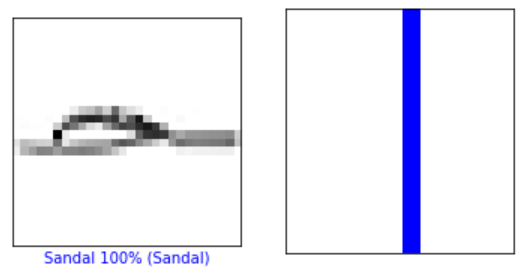
Пример 7:



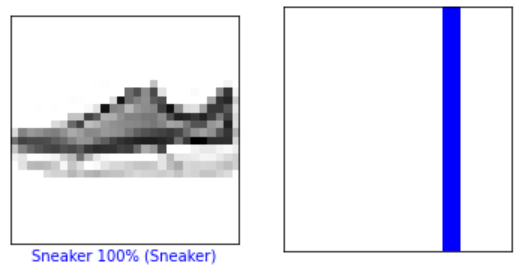
Пример 8:



Пример 9:



Пример 10:



**Выводы**

Были использовали примерно 70,000 монохромных изображений для обучения нейросети. В результате наших тестов нам удалось достичь точности, близкой к 90%. Учитывая, что речь идёт о категоризации одежды на маркетплейсе, данная точность приемлема для поставленных целей.

Сеть обучена, ввод принимается программой без ошибок. Программа исправна.