

# Классификация изображений

Курсовая работа  
студентки 351 группы М. М. Крулевой

Саратовский государственный университет  
им. Н. Г. Чернышевского

Кафедра математической кибернетики  
и компьютерных наук

Научный руководитель: к. ф.-м. н., доцент Иванов А. С.

2021г.

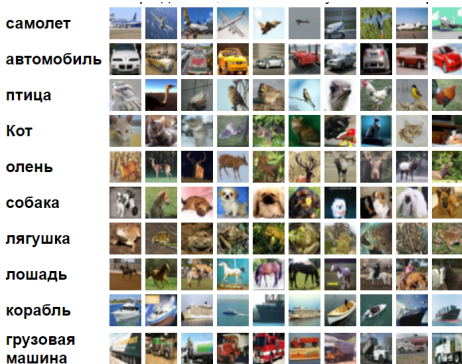
Разработать веб-приложения для классификации случайных изображений из определенного набора данных с помощью сверточной нейронной сети.

Для этого нужно:

- 1 построить и обучить модель для классификации изображений;
- 2 протестировать обученную модель на случайных изображениях из тестового набора;
- 3 отобразить результаты в браузере.

Для реализации поставленных задач будут использованы:

- Python
- Tensorflow и Keras
- CIFAR-10



- Разбитие датасета на обучающий и тестовый наборы.
- Перемешивание их в случайном порядке.
- Нормализация входных данных в диапазон значений  $[0; 1]$ .
- Группировка набора данных по 64 образцам.

```
1 ds_train, info = tfds.load("cifar10", with_info=True,  
  ↪ split="train", as_supervised=True)  
2 ds_test = tfds.load("cifar10", split="test",  
  ↪ as_supervised=True)  
3 ds_train =  
  ↪ ds_train.shuffle(1024).map(preprocess_image).batch(64)  
4 ds_test =  
  ↪ ds_test.shuffle(1024).map(preprocess_image).batch(64)
```

Используется стек слоев `Sequential`.

Первый слой модели — сверточный. Он будет принимать входные данные и запускать на них сверточные фильтры. Функция `MaxPooling2D()` используется для уменьшения размера ввода и извлечения важной информации.

Слой отсева `Dropout()` применяется в нейронных сетях для решения проблемы переобучения.

Аналогично выполняется создание сверточных слоев с 64 и 128 фильтрами.

```
1 model = Sequential()
2 model.add(Conv2D(filters=32, kernel_size=(3, 3),
   ↪ padding='same', input_shape=input_shape,
   ↪ activation='relu'))
3 model.add(MaxPooling2D(pool_size=(2, 2)))
4 model.add(Dropout(0.25))
```

- 1 Сглаживание данных.
- 2 Создание полносвязного слоя.

Функция активации softmax выбирает нейрон с наибольшей вероятностью в качестве своего выходного сигнала, считая, что изображение принадлежит к этому классу.

```
1 model.add(Flatten())  
2 model.add(Dense(1024, activation="relu"))  
3 model.add(Dropout(0.5))  
4 model.add(Dense(num_classes, activation="softmax"))
```

Теперь осталось скомпилировать модель.  
Оптимизатор нужен для того, чтобы  
приблизиться к точке наименьших потерь.

```
1 model.compile(loss="sparse_categorical_crossentropy",  
  ↪ optimizer="adam", metrics=["accuracy"])
```

На заключительном этапе запустим процесс  
обучения на 30 эпохах. Точность классификации  
составила 81.5%.

```
1 model.fit(ds_train, validation_data=ds_test,  
  ↪ epochs=epochs, batch_size=batch_size)
```

Для того, чтобы проверить работу обученной модели, выбираются 4 случайных изображений из тестового набора и для них составляется прогноз.

Тестирование производится через пользовательский интерфейс в браузере. Для его разработки была использована библиотека Flask.

```
1 for i in range(4):
2     sample_image = data_sample[0].numpy()[i]
3
4     ↪ sample_label.append(categories[data_sample[1].numpy()[i]
5     prediction =
6     ↪ np.argmax(model.predict(sample_image.reshape(-1,
7     ↪ *sample_image.shape))[0])
8     predict_label.append(categories[prediction])
9     plt.imsave("static/img/" + str(i) + ".jpg",
10    ↪ sample_image)
```



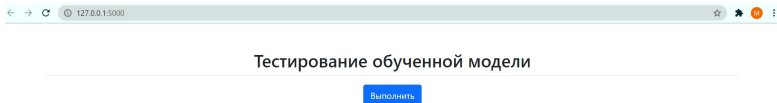


Рис.: Стартовая страница

← → ↺ 127.0.0.1:5000/classifier ☆ 🌟 🔍

**Точность модели составляет 81.5%**





Ожидаемый результат <b>Грузовик</b>	Ожидаемый результат <b>Лягушка</b>	Ожидаемый результат <b>Автомобиль</b>	Ожидаемый результат <b>Лягушка</b>
			
<b>Грузовик</b>	<b>Лягушка</b>	<b>Автомобиль</b>	<b>Лягушка</b>
Классифицировано верно	Классифицировано верно	Классифицировано верно	Классифицировано верно

Рис.: Страница результатов тестирования модели

- С помощью сверточной нейронной сети была построена, обучена и протестирована на случайных изображениях модель для классификации изображений.
- Разработано веб-приложение, которое предоставляет возможность протестировать обученную модель на четырех случайных изображениях из тестового набора и отобразить результаты работы.



<https://www.tensorflow.org/>

Библиотека машинного обучения TensorFlow



<https://keras.io/>

Библиотека машинного обучения Keras



Ф.М. Гафаров, А.Ф. Галимянов

Искусственные нейронные сети и их приложения

*Казань: Издательство Казанского университета, 2018. — 121 р.*



Dr. A. Rosebrock

Deep Learning for Computer Vision with Python

*Philadelphia: PYIMAGESEARCH, 2017. — 330 р.*



Д. Форсайт, Д. Понс

Компьютерное зрение. Современный подход

*М.: Вильямс, 2004. — 928 с.*



Ш. Нишант

Машинное обучение и TensorFlow

*СПБ.: Питер, 2019. — 336 р.*



К. Symoniam, A. Zisserman

Very deep convolutional networks for largescale image recognition

*ICLR. — 2015. — Рр. 1–16.*



Л. М. Ха

Свёрточная нейронная сеть для решения задачи классификации

*ТРУДЫ МФТИ. — 2016. — Т. 8, № 3. — С. 91–97.*



С. Хайкин

Нейронные сети: полный курс

*М.: Вильямс, 2016. — 1103 р.*



P. Goyal, S. Pandey, K. Jain

Deep Learning for Natural Language Processing

*New York: Apress, 2018. — 277 р.*



Я. Гудфеллоу, И. Бенджио, А. Курвилль

Глубокое обучение

*М.: ДМК Пресс, 2018. — 652 с.*



F. Chollet

Deep Learning with Python

*New York: Manning Publications Co., 2018. — 361 p.*

СПАСИБО ЗА ВНИМАНИЕ!