Т. 9. №12. 2023
https://doi.org/10.33619/2414-2948/97

УДК 004.032.26 https://doi.org/10.33619/2414-2948/97/06

ПРИМЕНЕНИЕ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РАСПОЗНАВАНИЯ ОБРАЗОВ

©Павлов Е. М., *ORCID: 0009-0002-0028-8270, SPIN-код: 3158-0223, Национальный исследовательский университет «МЭИ», г. Москва, Россия, PavlovyEgM@mpei.ru*
©Рыжов А. В., *ORCID: 0009-0003-4729-9505, Национальный исследовательский университет «МЭИ», г. Москва, Россия, RyzhovAVl@mpei.ru*
©Баланев К. С., *ORCID: 0009-0002-9722-7262, SPIN-код: 8192-1861, Национальный исследовательский университет «МЭИ», г. Москва, Россия, BalanevKS@mpei.ru*
©Крепков И. М., *канд. техн. наук, Национальный исследовательский университет «МЭИ», г. Москва, Россия, KrepkovIM@mpei.ru*

APPLICATION OF NEURAL NETWORKS FOR PATTERN RECOGNITION

©Pavlov E., *ORCID: 0009-0002-0028-8270, SPIN-code: 3158-0223, National Research University Moscow Power Engineering Institute, Moscow, Russia, PavlovyEgM@mpei.ru*
©Ryzhov A., *ORCID: 0009-0003-4729-9505, National Research University Moscow Power Engineering Institute, Moscow, Russia, RyzhovAVl@mpei.ru*
©Balanev K., *ORCID: 0009-0002-9722-7262, SPIN-code: 8192-1861, National Research University Moscow Power Engineering Institute, Moscow, Russia, BalanevKS@mpei.ru*
©Krepkov I., *Ph.D., National Research University Moscow Power Engineering Institute, Moscow, Russia, KrepkovIM@mpei.ru*

Аннотация. Рассматривается процесс создания и обучения нейронной сети для задачи классификации изображений собак и кошек с использованием TensorFlow и архитектуры MobileNetV2. Описывается подготовка и предобработка данных, включая изменение размеров и нормализацию изображений. Приводятся детали интеграции предварительно обученной модели MobileNetV2, демонстрируется процесс дообучения модели на специфических данных, а также методы построения и оптимизации модели, включая добавление и настройку дополнительных слоев. Отдельное внимание уделено практическому применению обученной модели для классификации новых изображений, включая загрузку, обработку, предсказание и визуализацию результатов.

Abstract. This article details the process of creating and training a neural network for the task of classifying dog and cat images using TensorFlow and the MobileNetV2 architecture. Data preparation and preprocessing, including image resizing and normalization, are described. Details of the integration of the pre-trained MobileNetV2 model are given, the process of pre-training the model on specific data is demonstrated, as well as methods of model construction and optimization, including the addition and tuning of additional layers. Special attention is paid to practical application of the trained model for classification of new images, including loading, processing, prediction and visualization of results.

Ключевые слова: нейронные сети, распознавание образов, машинное обучение.

Keywords: neural networks, pattern recognition, machine learning.

Т. 9. №12. 2023
https://doi.org/10.33619/2414-2948/97

В эпоху цифровизации огромное количество данных требует эффективных и автоматизированных методов анализа, особенно в области распознавания изображений. Прогресс в разработке и использовании нейронных сетей значительно продвинул границы возможностей в этой сфере, открывая новые горизонты в самых разнообразных областях — от автоматического распознавания лиц и объектов до точной медицинской диагностики и обработки спутниковых снимков [1].

Нейронные сети, благодаря их способности обучаться на больших объемах данных и извлекать сложные закономерности из визуального контента, уже демонстрируют результаты, превосходящие человеческую точность в некоторых задачах. Это открывает широкие перспективы для повышения эффективности и автоматизации рутинных процессов в различных отраслях. Примером может служить медицина, где нейросети помогают в распознавании рентгеновских и МРТ снимков, повышая точность диагностики заболеваний. В промышленности и безопасности эти технологии способствуют развитию систем видеонаблюдения и анализа технического состояния оборудования. А в сфере автомобильного транспорта нейронные сети лежат в основе разработки систем автономного вождения.

Первый этап создания нейросети для распознавания изображений включает в себя подготовку и настройку рабочего окружения, что включает импорт следующих библиотек:

1. NumPy — это основная библиотека для научных вычислений в Python. Она предоставляет поддержку больших многомерных массивов и матриц, вместе с широким набором высокоуровневых математических функций для операций с этими массивами. В контексте обработки изображений NumPy используется для манипуляций и трансформаций данных изображений, которые часто представлены в форме массивов.

2. TensorFlow — это один из наиболее популярных фреймворков для глубокого обучения. Он предлагает гибкие инструменты и библиотеки для построения и обучения различных видов нейронных сетей. TensorFlow обеспечивает поддержку как для исследовательской работы, так и для развертывания производственных систем [3].

3. TensorFlow Datasets — это утилита, предоставляющая большое количество стандартизированных, готовых к использованию наборов данных. Она полезна для получения и предварительной обработки данных перед их использованием для обучения нейросетей.

4. Keras (в TensorFlow): Keras — это высокоуровневый API для создания и обучения моделей глубокого обучения. Он интегрирован прямо в TensorFlow и упрощает многие задачи, делая код более читаемым и лаконичным.

5. Слои в Keras — это основные строительные блоки нейронных сетей. В частности:

Dense — полносвязный слой нейронной сети, где каждый нейрон связан со всеми нейронами в предыдущем слое.

GlobalAveragePooling2D — слой, который усредняет карты признаков по пространственным измерениям, уменьшая количество параметров и упрощая модель.

Dropout — слой, который помогает предотвратить переобучение в нейронных сетях путем «выключения» случайного набора активаций в слоях во время обучения.

6. Matplotlib — это графическая библиотека, используемая для визуализации данных, что может быть полезным для отображения изображений, графиков обучения, результатов тестирования и т. д.

Второй этап процесса создания нейросети для распознавания изображений заключается в загрузке и подготовке тестовых данных. Рассмотрим этот этап на примере

Т. 9. №12. 2023
https://doi.org/10.33619/2414-2948/97

использования набора данных «cats_vs_dogs»:

```
train, info = tfds.load('cats_vs_dogs', split=['train[:100%]'], with_info=True,
as_supervised=True)
```

Загрузка набора данных "cats_vs_dogs" включает в себя изображения котов и собак, широко применяемые для задач бинарной классификации в области компьютерного зрения. В этом примере используется параметр split=['train[:100%]'], указывающий на загрузку всего 100% набора данных, предназначенного для тренировки, что означает использование всего доступного набора для обучения нейросети. Благодаря флагу with_info=True получается дополнительная информация о наборе данных, включая его структуру, количество классов, размер выборки и другие метаданные, что важно для настройки процесса обучения и валидации. Включение опции as_supervised=True означает загрузку данных в формате кортежей, где входные данные представляют собой изображения, а целевая переменная — метки классов (коты или собаки), что делает формат удобным для непосредственного использования в процессе обучения, так как четко разделяет изображения и соответствующие им метки.

Третий этап в создании нейросети для распознавания изображений включает адаптацию входных данных под спецификации предварительно обученной модели. Одним из ключевых моментов на этом этапе является изменение размера и нормализация изображений, чтобы они соответствовали требованиям предварительно обученной нейросети, которая эффективно работает с изображениями размером 224×224 пикселей. Для начала необходимо преобразовать изображений в формат **float32** [5], который является общепринятым для работы с данными в нейронных сетях. После этого нужно адаптировать размер каждого изображения к требуемым параметрам (224×224 пикселей), чтобы они могли быть корректно обработаны предварительно обученной нейронной сетью. Затем выполняется приведение значений пикселей к диапазону от 0 до 1 для улучшения эффективности и стабильности процесса обучения. Заключительным этапом функция возвращает обработанные изображения вместе с их метками, обеспечивая готовность данных к использованию в процессе обучения.

Четвертый этап является критически важной стадией в процессе создания нейросети для распознавания изображений, на этом этапе происходит является подготовка обработанных данных к обучению [2, 3]. Данные, которые уже были изменены в размерах и нормализованы, группируются в пакеты (batches) для более эффективного и управляемого процесса обучения.

```
train_resized = train[0].map(resize_image)
train_batches = train_resized.shuffle(1000).batch(16)
```

Применение функции изменения размера включает использование функции resize_image на каждом изображении в наборе данных train для обеспечения соответствия всех изображений необходимым параметрам размера и нормализации. Далее, для улучшения обучения и предотвращения переобучения на конкретном порядке данных, используется перемешивание данных с помощью shuffle(1000), работающего с буфером размером в 1000 элементов. Это следует группировкой данных в пакеты при помощи batch(16), создавая таким образом пакеты по 16 элементов и позволяя эффективно обрабатывать группы образцов за

Т. 9. №12. 2023
https://doi.org/10.33619/2414-2948/97

один проход нейронной сети, тем самым оптимизируя шаг градиентного спуска. В итоге формируются пакеты данных, готовые к использованию в процессе обучения модели, что помогает улучшить управляемость и эффективность обучения, позволяя модели обрабатывать сразу целую группу изображений, а не каждое в отдельности.

Пятый этап в разработке нейросети для распознавания изображений — интеграция и настройка предварительно обученной модели, как основы для дальнейшего обучения через трансферное обучение. Хотя часто используются проверенные архитектуры, такие как MobileNetV2, важно понимать, что этот этап не обязателен. Если у вас есть достаточно данных и ресурсов, вы можете разрабатывать свою нейросеть с нуля, создавая уникальную архитектуру без использования предварительно обученных моделей, что может быть полезным в случаях, когда доступные предварительно обученные модели не подходят или требуется большая гибкость и контроль над архитектурой сети.

```
base_layers = tf.keras.applications.MobileNetV2(input_shape=(SIZE, SIZE, 3),
include_top=False)
base_layers.trainable = False
```

В этом фрагменте описывается начальный этап создания нейронной сети, включающий в себя загрузку базовой модели и заморозку ее весов. Для начала выбирается модель MobileNetV2, известная своей эффективностью и легкостью, что делает ее идеальным выбором для мобильных устройств и встраиваемых систем. Параметр input_shape устанавливается таким образом, чтобы соответствовать размерам подготовленных изображений, а параметр include_top=False исключает верхние слои модели, обеспечивая таким образом ее гибкость для разнообразных задач. Далее происходит заморозка весов базовой модели с помощью установки base_layers.trainable = False. Это означает, что веса базовой модели остаются неизменными в процессе дальнейшего обучения, позволяя использовать уже извлеченные признаки без риска их искажения. Такой подход способствует ускорению обучения и снижает риск переобучения на специфических данных.

Шестой этап создания нейросети для распознавания изображений включает в себя составление и компиляцию модели. Этот этап критически важен, так как здесь происходит конфигурирование слоев нейросети и определение параметров обучения.

```
model = tf.keras.Sequential([
base_layers,
GlobalAveragePooling2D(),
Dropout(0.2),
Dense(1)
])
model.compile(loss = tf.keras.losses.BinaryCrossentropy(from_logits=True))
```

Добавление базовой модели использует предварительно обученную модель MobileNetV2 для извлечения признаков из изображений. Применяется слой GlobalAveragePooling2D, который уменьшает размерность данных, сохраняя ключевые признаки, что помогает упростить модель и снизить риск переобучения. Регуляризация с помощью слоя Dropout случайным образом «выключает» некоторые нейроны во время обучения, улучшая обобщающую способность модели и предотвращая переобучение. Классификационный слой, выполненный с помощью одного нейрона в слое Dense,

Т. 9. №12. 2023
https://doi.org/10.33619/2414-2948/97

выполняет финальное решение о том, присутствует на изображении кошка или собака. Компиляция модели включает выбор функции потерь BinaryCrossentropy для двоичной классификации (кошки против собак), измеряющей точность модели на обучающих данных и направляющей процесс обучения.

Седьмой этап в создании нейросети для распознавания изображений заключается в обучении модели:

```
model.fit(train_batches, epochs=1)
```

В этом шаге модель проходит процесс обучения, используя подготовленные и предобработанные пакеты данных *(train_batches).* Задача этого этапа — адаптировать веса модели таким образом, чтобы она могла точно классифицировать изображения на категории «кошки» или «собаки». Параметр *epochs=1* указывает, что модель должна пройти через весь набор тренировочных данных один раз. Этот однократный проход помогает определить начальное качество и эффективность модели, а также выявить первоначальные тенденции в процессе обучения. В дальнейшем количество эпох может быть увеличено для улучшения точности модели.

Заключительный этап процесса заключается в анализе и классификации загруженных изображений на предмет того, изображен ли на них кот или собака.

```
image_files = [file for file in uploaded_files if file.lower().endswith(('.jpeg', '.jpg'))]
for img_file in image_files:
    img = load_img(img_file)
    img_array = img_to_array(img)
    img_resized, _ = resize_image(img_array, _)
    img_expanded = np.expand_dims(img_resized, axis=0)
    prediction = model.predict(img_expanded)[0][0]
    pred_label = 'КОТ' if prediction < 0.5 else 'СОБАКА'
    plt.figure()
    plt.imshow(img)
    plt.title(f'{pred_label} {prediction}')
```

На этом этапе сначала происходит фильтрация списка файлов так, чтобы оставить только изображения с расширениями .jpeg или .jpg. Затем, для каждого изображения в отфильтрованном списке выполняется следующий ряд действий: изображение загружается, преобразуется в массив данных, затем размер изображения изменяется, оно нормализуется и расширяется для подачи в модель. После этих подготовительных шагов изображение подается в модель нейронной сети, которая выдает предсказание [4]. Это предсказание используется для определения, изображен ли на картинке кот (если значение предсказания меньше 0,5) или собака (если значение равно или больше 0,5). Модель, скомпилированная с BinaryCrossentropy(from_logits=True), предсказывает «логиты» — значения перед применением сигмоидной функции. Сигмоидная функция преобразует логиты в вероятности, принимающие значения между 0 и 1. В контексте бинарной классификации (коты против собак), если предсказанное значение ближе к 0, это указывает на большую вероятность принадлежности к первому классу (здесь "КОТ"), а если значение ближе к 1 — ко второму классу («СОБАКА»). Таким образом, установка порога в 0.5 является логичной, так как это середина между 0 и 1, что позволяет сбалансированно классифицировать изображения на две

Т. 9. №12. 2023
https://doi.org/10.33619/2414-2948/97

группы. Если предсказание модели меньше 0,5, мы решаем, что на изображении скорее всего кот (так как значение ближе к 0), и наоборот, если значение равно или превышает 0.5, предполагаем, что на изображении собака (так как значение ближе к 1).

Наконец, результат предсказания визуализируется: на экран выводится само изображение вместе с предсказанной меткой и числовым значением предсказания (Рисунок).

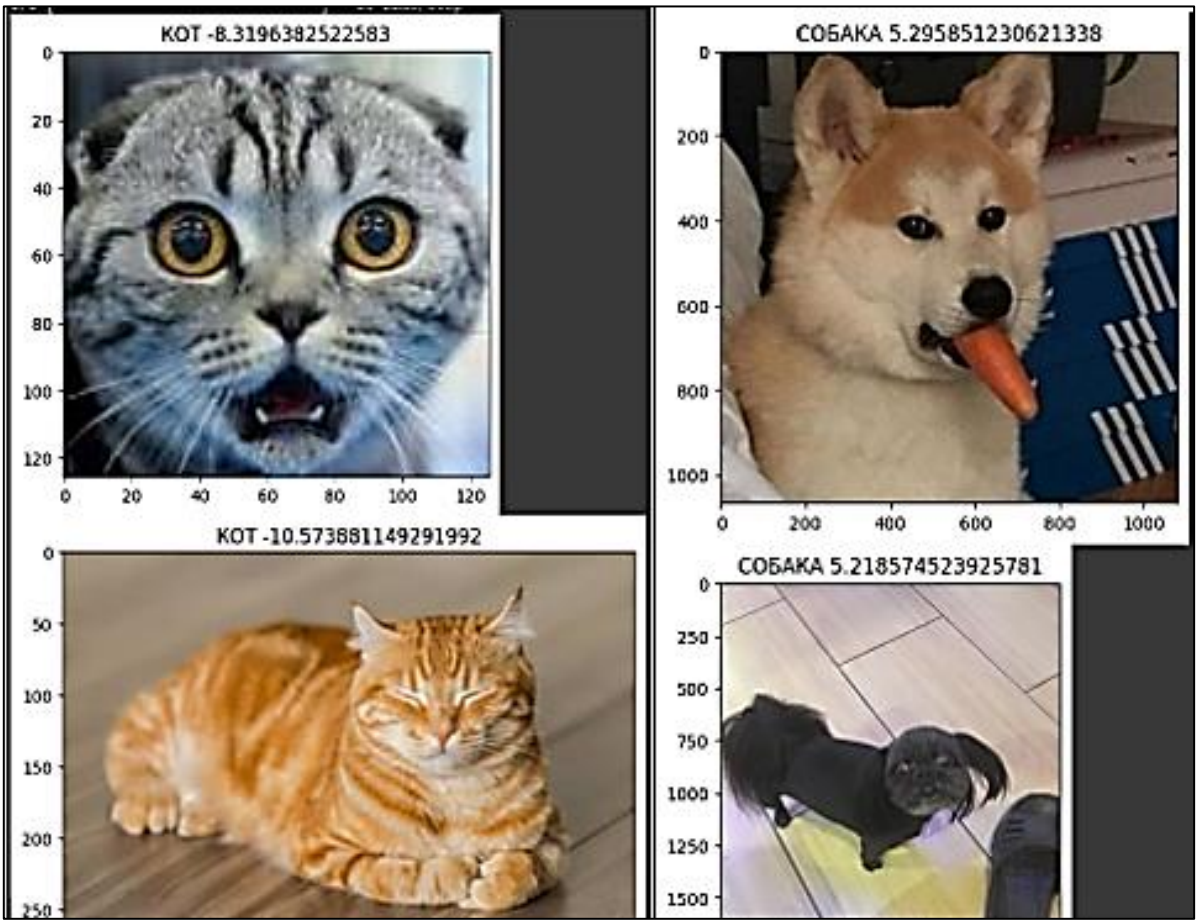

Рисунок. Полученное предсказание на основе переданных тестовых данных

Этот процесс показывает, как мощные инструменты машинного обучения могут быть использованы для решения относительно сложных задач, таких как распознавание изображений, с использованием сравнительно простого и интуитивно понятного кода. Используя методы глубокого обучения, мы можем достичь значительной точности в таких задачах, опираясь на существующие разработки и архитектуры, что существенно ускоряет и упрощает процесс разработки интеллектуальных систем.

Список литературы:

1. Потопов А.С. Автоматический анализ изображений и распознавание образов. М.: LAP Lambert Academic Publishing, 2017. 292 с.

2. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. М.: Вильямс, 2017. 480 с.

3. Паттанаяк С. Глубокое обучение и TensorFlow для профессионалов. Математический подход к построению систем искусственного интеллекта на Python. М.: Диалектика, 2019. 480 с.

4. Емельянова С. В. Информационные технологии и вычислительные системы. Вычислительные системы. Компьютерная графика. Распознавание образов // Математическое моделирование. 2015. Т. 100.

5. Хеллман Д. Стандартная библиотека Python 3. М.; СПб : Вильямс, 2018. 1376 с.

References:

1. Potopov, A. S. (2017). Avtomaticheskii analiz izobrazhenii i raspoznavanie obrazov. Moscow. (in Russian).

2. Myuller, A., & Gvido, S. (2017). Vvedenie v mashinnoe obuchenie s pomoshh'yu Python. Moscow. (in Russian).

3. Pattanayak, S. (2019). Glubokoe obuchenie i TensorFlow dlja professionalov. Matematicheskij podhod k postroeniju sistem iskusstvennogo intellekta na Python. Moscow. (in Russian).

4. Emelyanova, S. V. (2015). Informatsionnye tekhnologii i vychislitel'nye sistemy. Vychislitel'nye sistemy. Komp'yuternaya grafika. Raspoznavanie obrazov. In *Matematicheskoe modelirovanie,* 100. (in Russian).

5. Hellman, D. (2018). Standartnaya biblioteka Python 3. Moscow. (in Russian).

Работа поступила
в редакцию 09.11.2023 г.

Принята к публикации
20.11.2023 г.

Ссылка для цитирования:

Павлов Е. М., Рыжов А. В., Баланев К. С., Крепков И. М. Применение нейронных сетей для распознавания образов // Бюллетень науки и практики. 2023. Т. 9. №12. С. 52-58. https://doi.org/10.33619/2414-2948/97/06

Cite as (APA):

Pavlov, E., Ryzhov, A., Balanev, K., & Krepkov, I. (2023). Application of Neural Networks for Pattern Recognition. *Bulletin of Science and Practice, 9*(12), 52-58. (in Russian). https://doi.org/10.33619/2414-2948/97/06