УДК 004.423

ПОДГОТОВКА ИЗОБРАЖЕНИЙ ПРИ ОБУЧЕНИИ СВЁРТОЧНОЙ НЕЙРОСЕТИ. СОЗДАНИЕ ГЕНЕРАТОРА ИЗОБРАЖЕНИЙ

Кулинча П.В.

студент направления подготовки информатика и вычислительная техника,

Хакасский государственный университет имени Н.Ф. Катанова,

г. Абакан, Россия [1]

Аннотация: В статье рассматриваются методы подготовки изображений для обучения свёрточной нейросети, включая использование генераторов изображений, аугментации, разбиения на батчи и one-hot кодирования, с примерами реализации на Python с использованием TensorFlow и Keras.

Ключевые слова: свёрточные нейросети, генератор изображений, аугментация, обучение нейросети, батч, one-hot кодирование, TensorFlow, Keras

IMAGE PREPARATION WHEN TRAINING A CONVOLUTIONAL NEURAL NETWORK. CREATING AN IMAGE GENERATOR

Kulincha P.V.

student of computer science and computer engineering department,

N.F. Katanov Khakass State University,

Abakan, Russia

Abstract: This article explores methods for image preprocessing in convolutional neural network training, including the use of image generators, data augmentation,

[1] Научный руководитель: Спирин Д.В. доцент кафедры ЦТиД, Хакасский государственный университет имени Н.Ф. Катанова, г. Абакан, Россия

batch processing, and one-hot encoding, with implementation examples in Python using TensorFlow and Keras.

Keywords: convolutional neural networks, image generator, augmentation, neural network training, batch, one-hot encoding, TensorFlow, Keras

Введение

Современные методы машинного обучения, особенно в области компьютерного зрения, активно используют свёрточные нейронные сети (CNN, Convolutional Neural Networks) для решения задач классификации, детекции и сегментации изображений. Однако эффективность таких моделей во многом зависит от качества и разнообразия обучающих данных. В условиях ограниченного объёма исходных изображений важно использовать специальные методы подготовки данных, позволяющие расширить датасет и повысить обобщающую способность модели.

Одним из ключевых инструментов такой подготовки являются генераторы изображений, способные автоматически обрабатывать и подавать данные в нейросеть с учетом заданных параметров. Они позволяют не только нормализовать изображения, но и применять аугментацию — искусственное увеличение объёма обучающей выборки путём трансформаций изображений. Также важными аспектами подготовки данных являются разбиение на батчи (batch processing) и кодирование меток в формате one-hot, необходимые для корректной подачи информации в модель.Исторический аспект сверточных нейросетей

Теоретический аспект генераторов изображений

Генераторы изображений играют важную роль при обучении нейросетей, особенно в задачах компьютерного зрения, где объём данных может быть ограничен, а требования к качеству высоки. Генератор — это специальный объект, реализующий итеративную подачу данных в модель по мере

необходимости, без необходимости загружать весь датасет в оперативную память. Это особенно важно при работе с большими объёмами изображений.

Одним из наиболее популярных инструментов для создания генераторов в экосистеме TensorFlow является класс ImageDataGenerator из библиотеки keras.preprocessing.image. Он позволяет не только организовать поток изображений с жёсткого диска, но и на лету выполнять **предобработку** изображений (например, нормализацию) и **аугментацию** — то есть генерацию новых вариантов изображений на основе исходных. Такие вариации могут включать повороты, масштабирование, сдвиги, зеркальные отражения, изменения яркости и многое другое [1].

Использование генераторов позволяет:

- Повысить **обобщающую способность** модели;
- Снизить **риск переобучения**;
- Эффективно использовать ресурсы памяти;
- Упростить **работу с данными** разной структуры и форматов.

Таким образом, генераторы являются неотъемлемой частью современных пайплайнов обучения нейросетей, обеспечивая как удобство, так и повышение качества итоговой модели.

Аугментация изображений и её значение

Аугментация изображений (от англ. *augmentation* — увеличение, расширение) — это метод искусственного увеличения объёма обучающих данных за счёт создания новых изображений путём применения различных трансформаций к уже имеющимся. Этот подход позволяет значительно повысить качество обучения нейросети без необходимости вручную собирать дополнительные данные [2].

К основным видам аугментации относятся:

- **Повороты (rotation)** — изображение поворачивается на случайный угол, обычно в пределах ±15–30°.

- **Сдвиги по ширине и высоте (width/height shift)** — изображение немного сдвигается в стороны или вверх-вниз.
- **Масштабирование (zoom)** — увеличивает или уменьшает изображение.
- **Сдвиги с деформацией (shear)** — наклон изображения по вертикали или горизонтали.
- **Зеркальное отражение (horizontal_flip)** — особенно полезно для распознавания лиц или объектов, где симметрия допустима.
- **Изменения яркости (brightness_range)** — позволяет обучить модель работать при различных условиях освещения.

В приведённом в работе коде для обучающей выборки использована комплексная аугментация, включающая:

- повороты до 15°;
- сдвиги до 20% по ширине и высоте;
- масштабирование и искажения;
- изменение яркости;
- горизонтальные отражения.

Целью аугментации является **повышение устойчивости** модели к вариациям входных данных, что в конечном итоге улучшает **точность на валидационных и тестовых данных**, а также **уменьшает переобучение**. Это особенно важно, когда датасет ограничен по объёму или содержит изображения, сделанные в разных условиях [2].

Что такое батчи (batch) и зачем они нужны

В машинном обучении термин **«batch»** (батч) означает **пакет данных,** обрабатываемый моделью за одну итерацию обучения. Вместо подачи нейросети всех изображений сразу (что может быть слишком ресурсоёмко), данные делятся на меньшие порции — батчи.

Например, если в наборе 3200 изображений, и размер батча равен 32, модель обработает весь датасет за 100 итераций (3200 / 32 = 100). Эти 100 итераций составляют **одну эпоху** обучения.

Преимущества использования батчей:

- **Снижение потребления памяти**: не нужно загружать весь датасет в память.
- **Ускорение обучения**: можно использовать векторные операции и ускорение на GPU.
- **Более стабильное обновление весов**: усреднение градиентов по батчу делает обучение устойчивее, чем при обновлении на каждом отдельном примере (что называется "stochastic gradient descent").

В нашем коде генератора данных задан параметр batch_size=32, что означает, что модель будет обучаться, получая по 32 изображения за раз. Такой размер батча является универсальным компромиссом между скоростью обучения и стабильностью.

One-hot кодирование

В задачах классификации, таких как распознавание лиц, каждая фотография относится к одному из нескольких классов (например, человек A, человек B и т.д.). Чтобы нейросеть могла корректно воспринимать эти классы, они должны быть представлены в числовом виде. Один из наиболее распространённых способов — **one-hot кодирование** [3].

Это метод представления категориальных (нечисловых) меток в виде бинарных векторов. Допустим, у нас есть 3 класса: A, B, C. Они будут представлены следующим образом:

- A → [1, 0, 0]
- B → [0, 1, 0]
- C → [0, 0, 1]

То есть для каждого класса создаётся вектор длины, равной количеству классов, где единица стоит только на позиции, соответствующей нужной категории.

В параметре flow_from_directory используется настройка: class_mode='categorical'.

Она указывает генератору данных автоматически применять one-hot кодирование к меткам классов. Это особенно важно при использовании **функции активации softmax** на выходном слое нейросети и **функции потерь categorical_crossentropy**, которые требуют именно такого формата меток.

Таким образом, благодаря one-hot кодированию, нейросеть может корректно соотнести вероятность каждого класса с правильным ответом и обучаться на множестве различных классов.

Реализация на практике

Все изображения будут храниться в основной директории face_dataset, которая будет содержать три подкаталога: train, validation и test. Эти директории предназначены для разделения данных на обучающую, валидационную и тестовую выборки (рисунок 1).

```
# Папки с обучающими, валидационными и тестовыми данными
train_dir = '/content/face_dataset/train'
val_dir = '/content/face_dataset/validation'
test_dir = '/content/face_dataset/test'
```

Рисунок 1 – Создание переменных с путями хранения данных для обучения нейросети [разработано автором]

В данной работе используется класс «ImageDataGenerator» из библиотеки «tensorflow.keras.preprocessing.image». В нем задаются параметры для случайных преобразований изображений при каждой итерации обучения (рисунок 1).

```
# Создание генераторов данных с аугментацией для train
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    brightness_range=[0.9, 1.1],
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Рисунок 2 – Создание параметров для тренировочного генератора изображений [разработано автором]

Эти преобразования применяются **только к обучающим изображениям**. Для валидации и тестирования используется другой генератор — val_test_datagen, в котором указана лишь нормализация (рисунок 2).

```
# Для валидации и теста аугментация не нужна, только нормализация
val_test_datagen = ImageDataGenerator(rescale=1./255)
```

Рисунок 3 – Создание параметров для тестового и валидационного генератора изображений [разработано автором]

Это сделано для того, чтобы на этапе проверки качества обучения использовать **неизменённые изображения**, позволяющие объективно оценить способность модели к генерализации.

Для создания генераторов используется метод flow_from_directory, который помогает организовать загрузку данных, разделённых по каталогам, в зависимости от меток классов. В нашем случае изображения делятся на три папки: **train**, **validation** и **test**. Далее приведены детали работы с каждым генератором.

Для обучающих данных применяется аугментация с помощью ранее описанного объекта train_datagen. Изображения будут масштабироваться, вращаться, сдвигаться и подвергаться другим преобразованиям (рисунок 3).

```
# Генератор обучающих данных
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical'
)
```

Рисунок 4 – Создание генератора для тренировочных изображений [разработано автором]

Здесь параметр target_size=(150, 150) указывает, что все изображения будут изменены до размера 150x150 пикселей. Параметр batch_size=32 определяет, что на каждом шаге будет подано по 32 изображения. Параметр class_mode='categorical' позволяет нам использовать **one-hot кодирование** для меток классов.

На рисунке 4 представлен код для создания тестовых и валидационыых генераторов.

```
# Генератор валидационных данных
validation_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical'
)
print(f"✅ Генератор validation загружен: {validation_generato
# Генератор тестовых данных
test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical'
)
print(f"✅ Генератор test загружен: {test_generator.samples} и
```

Рисунок 5 – Создание генераторов для тестовых и валидационных изображений [разработано автором]

Использование метода flow_from_directory упрощает процесс загрузки и предварительной обработки изображений. Оно позволяет легко и эффективно управлять данными, особенно когда изображения уже организованы по каталогам, соответствующим классам. Генераторы в сочетании с аугментацией

значительно улучшат обучение нейросети, обеспечивая её устойчивость к различным вариациям входных данных и предотвращая переобучение.

Для демонстрации результата работы генератора изображений и успешного разбиения данных, после выполнения кода, можно визуализировать несколько примеров изображений, которые были загружены и подготовлены для обучения (рисунок 6).

```
[ ]  import matplotlib.pyplot as plt
     img, label = next(train_generator)
     plt.imshow(img[0])  # вывести первое изображение
     plt.title(f"Label: {label[0]}")
     plt.show()
```

Рисунок 6 – Проверка работоспособности генератора [разработано автором]

Этот код позволяет проверить работу генератора, выводя одно случайное изображение из обучающего набора, а также отображая метку, которая соответствует данному изображению.

- Функция next(train_generator) извлекает следующий батч данных из генератора. Батч включает как изображения (в переменной img), так и их соответствующие метки (в переменной label).
- img[0] представляет собой первое изображение из текущего батча, которое выводится с помощью plt.imshow().
- label[0] отображает метку этого изображения, которая в случае использования class_mode='categorical' будет представлять собой вектор one-hot кодирования, где значение 1 указывает на соответствующий класс (человека).

Результат выполнения данной программы представлен на рисунке 7.

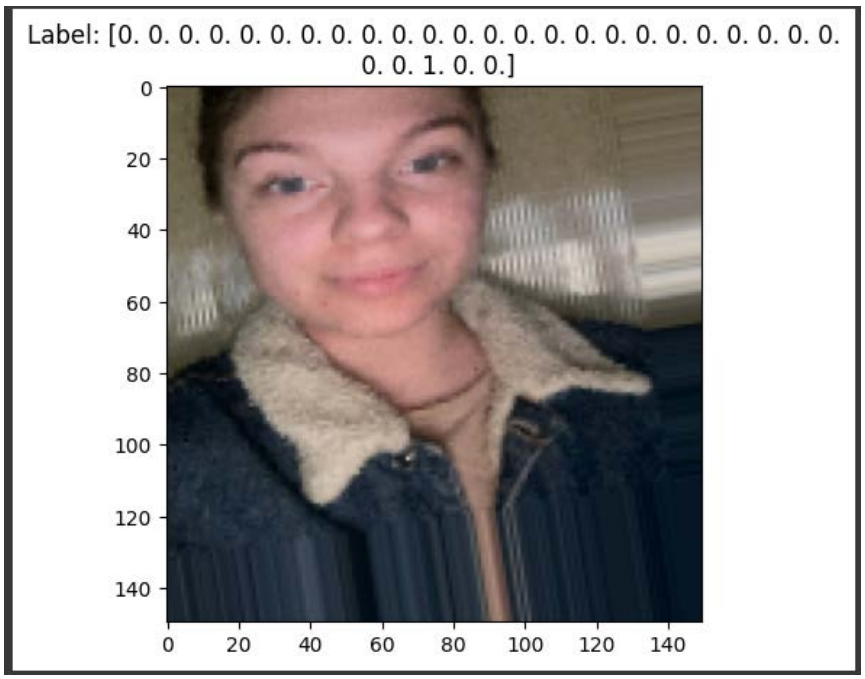

Рисунок 7 – Результат работы генератора [разработано автором]

Заключение

В статье рассмотрены основные аспекты подготовки данных для обучения сверточных нейросетей, включая использование генераторов изображений и аугментацию данных для улучшения качества обучения и предотвращения переобучения. Описан процесс создания генераторов с аугментацией, применением техник, таких как повороты, сдвиги и изменение яркости, а также использование one-hot кодирования для меток классов.

Библиографический список:

1. Chollet, F. (2017). Deep Learning with Python. Manning Publications.
2. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1), 60.
3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NIPS), 25, 1097–1105.

Оригинальность 76%