РАСПОЗНАВАНИЕ ОБРАЗОВ ОБЪЕКТОВ НА ИЗОБРАЖЕНИЯХ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

Бруй Н. М., аспирант, e-mail: brujnm@bsuir.by
Сыс А.Д., аспирант, e-mail: sys@bsuir.by
Калита О.В., аспирант, e-mail: olgakalitav@gmail.com

2022

Белорусский государственный университет информатики и радиоэлектроники

Ключевые слова: нейронная сеть, обучающие множества, классификация данных

Аннотация**.** С помощью обучающих множеств на языке программирования Python создана модель, состоящая из трёх слоёв. При тестировании установлено, что модель совершает предсказания с точностью до 88%. Обоснована актуальность нейронных сетей в области машинного обучения.

Введение

Нейронные сети, также известные как искусственные нейронные сети или имитированные нейронные сети, являются подмножеством машинного обучения и лежат в основе алгоритмов глубокого обучения [1]. Их название и структура основаны на работе человеческого мозга и имитируют способ передачи сигналов биологическими нейронами друг другу.

Искусственные нейронные сети состоят из входного слоя, одного или нескольких скрытых слоев и выходного слоя. Так как каждый узел, или искусственный нейрон, имеет индивидуальное влияние на принятие решения, он имеет свой параметр веса и смещение, которое регулирует конечное значение узла и с помощью заданного порогового значения определяет, активируется ли нейрон. Каждый нейрон соединяется с другим, и, если его выход превышает заданное пороговое значение, этот узел активируется, отправляя данные на следующий уровень сети. В противном случае данные не передаются на следующий уровень сети.

В данной статье обосновывается выбор параметров для создания нейронной сети, описывается дальнейшая разработка модели, а также её обучение и тестирование. Показано, что математический алгоритм может заменить человека в вопросе распознавания данных. Созданы модели нейронной сети с использованием обучающих множеств и протестированы с использованием новых образцов.

Основная часть

Для обучения нейронной сети выбран набор данных *Fashion MNIST dataset*, состоящий из 60 000 изображений для обучения модели и 10 000 изображений для её тестирования. Данные промаркированы 10 категориями одежды в оттенках серого, каждый образец имеет размер 28x28 пикселей [2].

Каждое изображение – это двумерный массив из 28 строк и 28 колонок, который необходимо преобразовать в список из 784 пикселей со значениями от 0 (абсолютно чёрный) до 255 (абсолютно белый), для увеличения скорости модели значение каждого пикселя следует разделить на 255. Таким образом, входной слой модели состоит из 784 нейронов, каждый из которых соответствует определённому пикселю изображения.

Так как исходный набор данных охватывает 10 категорий одежды, выходной слой включает 10 нейронов, каждый из которых коррелирует с определённым классом одежды и предсказывает, вероятность того, что изображение на текущей картинке соответствует классу текущего нейрона. В идеальной модели 9 из 10 нейронов имеют значение 0 или «нулевое соответствие классу одежды» и 1 нейрон имеет значение 1, то есть «на текущем изображении данный класс одежды».

Скрытый слой должен иметь 15 – 20% от входного слоя, в текущей модели он состоит из 128 нейронов, каждый из которых связан со всеми нейронами входного и выходного слоёв.

Создание модели. Для создания модели использованы платформы машинного обучения *TensorFlow* и *Keras* и язык программирования *Python* [3]. На этапе подготовки к созданию модели происходит загрузка набора данных, создание кортежей для обучения и тестирования с изображениями и соответствующими им маркерами, а также создание массива маркеров.

Как описано ранее, модель состоит из 3 слоёв: входного, разглаженного в список; скрытого и выходного с плотно связанными нейронами.

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

Перед обучением модели задаются исходные данные, которые включают оптимизатор для изменения параметров нейронной сети, таких как веса и скорость обучения, с целью уменьшения потерь [4]; функцию потерь для определения разницы между ожидаемым и полученным результатом в целях совершенствования модели [5] и функцию для оценки производительности модели [6].

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

Модель обучается в 10 циклов, в каждом из которых данные используются один раз.

```
model.fit(train_images, train_labels, epochs=10)
```

Для оценки точности модели используются образцы для тренировки, после чего на экран выводится протестированная точность.

```
text_loss, test_acc = model.evaluate(test_images, test_labels)
print("Tested accuracy >> ", test_acc)
```

Оценка производительности модели. Для оценки модели была использована функция, описанная в разделе выше. Модель тренируется в 10 циклов за 36 секунд и достигает точности, равной 88,49%. Для изображения работы модели создан цикл, который выводит 60 первых изображений с реальным и предсказанным маркером. В процессе тестирования 6 из 60 маркеров не совпало, что при округлении подтверждает полученную точность модели. Нейронная сеть имеет точность классификации 88% при высокой скорости вычислений.

Заключение

Для классификации одежды на языке программирования *Python* создана нейронная сеть, состоящая из трёх слоёв. Входной слой принимает значения цветов всех пикселей изображения, скрытый слой с помощью подобранных значений веса и смещения производит вычисления и передаёт данные выходному слою, который выдаёт вероятность соответствия изображения одной из 10 категорий одежды. Точность предсказаний модели достигает 88% с сохранением скорости вычислений.

Список использованных источников

1. Various Optimization Algorithms For Training Neural Network [electronic resource] – https://towardsdatascience.com/ optimizers-for-training-neural-network-59450d71caf6

2. An Introduction to Neural Network Loss Functions [electronic resource] – https://programmathically.com /an-introduction-to-neural-network-loss-functions/

3. Metrics [electronic resource] – https://keras.io/api/metrics/

4. Jost Tobias Springenberg, Martin Riedmiller (2013), ”Improving Deep Neural Networks with Probabilistic Maxout Units”, ICML 2013