Вестник Сыктывкарского университета.
Серия 1: Математика. Механика. Информатика.
Выпуск 2 (39). 2021

УДК 004.032.26 **DOI: 10.34130/1992-2752_2021_2_04**

ПРОЕКТИРОВАНИЕ НЕЙРОННОЙ СЕТИ ДЛЯ РАСПОЗНАВАНИЯ РУКОПИСНЫХ КИРИЛЛИЧЕСКИХ СИМВОЛОВ

И. И. Голенев, А. В. Ермоленко

В данной работе подробно описывается моделирование сверточной нейронной сети (CNN). Модель разрабатывалась на языке Python 3.8 с использованием библиотек TensorFlow и Keras.
Ключевые слова: сверточные нейронные сети, распознавание символов, глубокое обучение.

1. Введение

Распознавание образов, в том числе рукописных символов, является актуальной задачей, которая регулярно возникает в разных областях деятельности человека. Одна из наиболее часто встречающихся задач распознавания образов — *распознавание рукописных символов*. Для решения таких задач часто применяют искусственные нейронные сети (ИНС).

Задача *распознавания* кириллических букв *сложнее*, чем, например, распознавание латинских букв. Это связано с большим количеством классов (количеством букв), а также большим количеством похожих символов: «ь» - «ъ» - «ч»; «д» - «з» - «у»; «к» - «н» и другие.

В качестве входных данных взяты 14,2 тыс. изображений символов. Соотношение тренировочной, валидационной и тестовой выборок равны соответственно 80 %, 10 %, 10 % [1].

© Голенев И. И., Ермоленко А. В., 2021.

2. Постановка задачи

Задачу распознавания сигналов или образов (речевых или зрительных) можно свести к следующей математической постановке:

$$X \to Y,$$

где X — *вектор входных* сигналов; Y — *выходной* результат. Решить такую задачу — значит найти *функциональное отображение*, при котором для любого вектора X будет сформирован правильный результат Y.

Функциональное отображение задается множеством пар «вход» — «выход», или, иными словами, *обучающей выборкой.*

Так, например, при решении задач распознавания рукописного символа вектор X будет представлен в виде *формализованного множества*, а Y может быть представлен в виде *вероятностного вектора*, содержащего порядковый номер класса символа и *вероятность принадлежности* входного символа каждому из классов.

Таким образом, требуется провести обучение ИНС для настройки *коэффициентов весов* — связей между нейронами. Настройка весов проводится таким образом, при котором в отображении $X \to Y$ *минимизируется ошибка* обучения для каждого элемента обучающей выборки.

3. Входные данные. Предобработка данных

Для решения задачи распознавания образов, как правило, обрабатывают *нормализованные растры* изображений. В нашем случае образом является изображение рукописного символа (буквы) размером 32×32 пикселов, которые в дальнейшем конвертируются в 3d массив. Общее число пикселов внутри одного изображения $32 \times 32 = 1024$. Каждое значение цвета находится в диапазоне [0, 255]. Для нормализации изображения необходимо значение цвета каждого пиксела представить в виде типа данных float, после чего разделить это значение на 255. Так мы получим данные в диапазоне [0, 1.0], на которых CNN сходится быстрее [1].

Также используется *унитарное кодирование* [1], после которого изображения в обучающей выборке либо принадлежат определенному классу, либо не принадлежат: не должно быть промежуточного значения.

Для того чтобы избежать *переобучения* (overfitting) используется *dropout* — случайное исключение нейронов из обучения и *генерация новых* изображений. Генерация основывается на обучающей выборке,

применяют к ней такие манипуляции, как повороты, сдвиги и т. п. Генерация реализуется с помощью класса *keras.preprocessing.image.DataGenerator* [2].

Перед тем как начать строить модель, необходимо убедиться в том, что входные данные *сбалансированы* в обучающей выборке. Если классы в обучающей выборке несбалансированы, модель будет максимизировать точность «наибольших» классов, игнорируя другие классы, что неизбежно приведет к менее точным прогнозам на «меньших» классах.

Классы в нашем наборе данных *идеально сбалансированы*, каждому классу принадлежит 430 изображений.

4. Выходные данные

В качестве выходных данных CNN возвращает вероятностный вектор Y, содержащий порядковый *номер класса* символа и *вероятность принадлежности* входного символа каждому из классов. Всего классов 33 (количество букв). Пример: А — 0,5 %; Б — 3 %; В — 95 %; ... ; Я — 0,00001 %.

5. Построение модели CNN

Первый слой нашей CNN — *сверточный* (Conv2D). Он формирует «карту признаков», необходимую для понимания различных частей изображения. Количество каналов (фильтров) 32, а размер фильтров 5×5. Такой размер фильтра на первом сверточном слое позволит «запомнить» больше признаков изображения, нежели часто используемый фильтр 3×3. Число фильтров принято увеличивать с последующими слоями. В нашей модели используется двойная свертка (два сверточных слоя подряд).

В качестве функции активации используется ReLu, поскольку при построении глубоких обучающих сетей с большим количеством слоев *сигмоидальные* функции активации будут быстро «застаиваться» [3]. Это связано с тем, что максимальное значение производной сигмоидальной функции примерно равно 0,25. Следовательно, после большого числа слоев, произведения чисел меньше 1 на градиент устремляются к нулю.

После активации данные проходят через объединяющий слой (MaxPooling2D). Объединение *сжимает информацию*, которую предоставляют предыдущие слои. Процесс объединения позволяет лучше распознавать объекты и делает сеть более гибкой [1]. Слой объединения

содержит фильтр 2×2, что позволяет отсеить $3/4$ информации. Важно, чтобы в модели не было много объединяющих слоев, в противном случае можно потерять важные данные о распознаваемом объекте.

После строится исключающий слой (Dropout) [3; 4]. Эксперимент показал, что для распознавания рукописных кириллических символов оптимальным значением dropout'а является 0,2.

После нескольких циклов сверток, объединений и dropout'ов нужно сжать данные и передавать их полносвязным слоям. Для этого используется функция Flatten, которая подает на выход вектор с 4096 значениями.

Следующий слой (Dense) будем называть *слоем образов букв*. Будем считать, что в идеализированном случае каждый нейрон из слоя Dense соотносится с одним из компонентов на рис. 1. Тогда, после передачи изображения с определенным признаком буквы, существует определенный нейрон, чья активация станет ближе к единице. Будем считать, что таких признаков около 256.

В конце используется функция активации softmax, которая выбирает нейрон с наибольшей вероятностью, полагая, что входное изображение принадлежит этому классу.

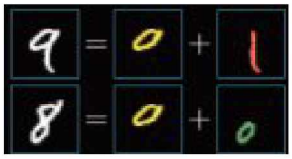

Рис. 1. Признаки символа

6. Обучение

В настройке *модели обучения* можно выделить три важных понятия:

- функция потерь (ошибка обучения);
- градиентный спуск;
- обратное распространение.

Функция потерь дает оценку прогнозам нашей модели. Ее можно формализовать следующим образом: Loss = F (фактическое значение, предсказанное значение). Нейросеть *минимизирует потери*, т. е. улучшает прогноз, приближая его к фактическому значению. Значение

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	2432
conv2d_1 (Conv2D)	(None, 32, 32, 32)	25632
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 64)	0
dropout_1 (Dropout)	(None, 8, 8, 64)	0
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 256)	1048832
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 33)	8481

Рис. 2. Модель CNN

функции потерь изменяется *при изменении весов* сети. Так как решается задача мультиклассовой классификации ($>$ 2 классов), используется функция categorical_crossentropy.

Модель ведет поиск лучших весов с помощью *градиентного спуска*. Чтобы сделать «один шаг» по методу градиентного спуска (одно изменение весов), необходимо [5]:

1) подать на вход весь обучающий набор данных,

2) вычислить ошибку для каждого объекта,

3) вычислить необходимую коррекцию,

4) после подачи всех данных рассчитать сумму градиентов,

5) провести коррекцию весов.

Величина изменения весов определяется *скоростью обучения*.

Важной функцией на этапе обучения является *оптимизатор*. Данная функция итеративно улучшает параметры (значения сверточных фильтров, веса, смещения нейронов, ...), чтобы минимизировать функцию потерь (ошибку). В работе была выбрана *RMSProp (root mean*

square propagation), так как это очень эффективная функция оптимизации [1; 6]. Можно было использовать оптимизатор стохастического градиентного спуска sgd, но он сходится медленнее, чем RMSProp.

В качестве метрики (оценки производительности модели) используется *accuracy* (точность), так как входные данные идеально сбалансированы [3; 4].

7. Вывод

Построенная последовательная модель Sequential() из Keras [2] (рис. 2, 3) обладает следующими характеристиками: Train: 0,997, Valid: 0,971, Test: 0,970, где Train — точность на тренировочной выборке, Valid — точность на валидационной выборке и Test — точность на тестовой выборке. В некоммерческих проектах точность определения символов не превышает 96 % [7; 8].

1. ИНС имеет высокий процент точности (accuracy) на идеально сбалансированных данных.

2. Сеть не переобучена, что видно на графике (рис. 3) и подтверждено собственными тестами.

3. Подтвердилось предположение, что похожие символы («д» – «з» – «у») трудно распознаются. В тестовом предложении из 15 букв верно распознано 13. Неправильно распознаны буквы «у» и «к».

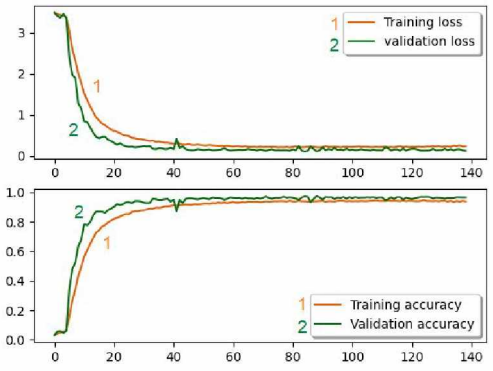

Рис. 3. Точность модели

Список литературы

1. **Гудфеллоу Я., Бенджио И., Курвилль А.** Глубокое обучение / пер. с анг. А. А. Слинкина. М.: ДМК Пресс, 2018. 652 с.

2. Keras: the Python deep learning API [Electronic resource] // Keras official site. URL: https://keras.io (дата обращения: 28.04.2021).

3. Глубокое обучение для новичков: тонкая настройка нейронной сети [Электронный ресурс] // Блог компании Wunder Fund, алгоритмы, машинное обучение. URL: https://habr.com/ru/company/wunderfund/blog/315476/ (дата обращения: 05.05.2021).

4. **Бабенко В. В., Котелина Н. О., Тельнова О. П.** Программно-информационное обеспечение палеопалинологической задачи // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2021. Вып. 1 (38). С. 26–40.*

5. Градиентный спуск [Электронный ресурс] // Свободная энциклопедия Википедия. URL: https://ru.wikipedia.org/wiki/Градиентный_спуск (дата обращения: 17.05.2021).

6. Методы оптимизации нейронных сетей [Электронный ресурс] // @Siarshai. URL: https://habr.com/ru/post/318970/ (дата обращения: 15.05.2021).

7. **Валеев Д. И.** Разработка системы обработки математических рукописных формул с применением нейросетевых технологий // ВКР. Челябинск, 2018. 43 с.

8. **Кулакова О. А., Воронова Л. И.** Распознавание рукописных букв с помощью нейронных сетей // *Материалы IX Международной студенческой научной конференции «Студенческий научный форум»*. URL: https://scienceforum.ru/2017/article/2017033009 (дата обращения: 10.05.2021).

Summary

Golenev I. I., Yermolenko A. V. Designing a neural network for recognizing handwritten cyrillic symbols

This paper deals with the modeling of a convolutional neural network (CNN). The model was developed in Python 3.8 using the TensorFlow and Keras.

Keywords: convolutional neural networks, character recognition, deep learning.

References

1. **Goodfellow I., Bengio Y., Courville A.** *Glubokoe obuchenie* [Deep learning] / transl. A. A. Slinkina, M.: DMK Press, 2018, 652 p.

2. Keras: the Python deep learning API [Electronic resource] / Keras official site. Available at: https://keras.io (Accessed: 28.04.2021).

3. Deep learning: image recognition with convolutional neural networks [Electronic resource] / *Blog kompanii Wunder Fund, algoritmy, mashinnoe obuchenie* [Wunder Fund company blog, Algorithms, machine learning]. Available at: https://habr.com/ru/company/wunderfund/blog/314872/ (Accessed: 05.05.2021).

4. **Babenko V. V., Kotelina N. O., Telnova O. P.** Software and information support of the paleopalinological problem, *Vestnik Syktyvkarskogo universiteta. Ser. 1: Matematika. Mekhanika. Informatika* [Bulletin of Syktyvkar University. Series 1: Mathematics. Mechanics. Informatics], 2021, 1 (38), pp. 26-40.

5. Gradient descent [Electronic resource] / *Svobodnaya enciklopediya Vikipediya* [Wikipedia The Free Encyclopedia]. Available at: https://en.wikipedia.org/wiki/Gradient_descent (Accessed: 17.05.2021).

6. Neural network optimization methods [Electronic resource] / @Siarshai. Available at: https://habr.com/ru/post/318970/ (Accessed: 15.05.2021).

7. **Valeev D. I.** Development of a system for processing mathematical handwritten formulas with using neural network technologies, *VKR*, Chelyabinsk, 2018, 43 p.

8. **Kulakova O. A., Voronova L. I.** Handwritten letters recognition using neural network, *Materialy IX Mezhdunarodnoj studencheskoj nauchnoj konferencii «Studencheskij nauchnyj forum»* [Materials of the IX International Student Scientific Conference «Student Scientific Forum»]. Available at: https:// scienceforum.ru/2017/article/ 2017033009 (Accessed: 10.05.2021).

Для цитирования: Голенев И. И., Ермоленко А. В. Проектирование нейронной сети для распознавания рукописных кириллических символов // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2021. Вып. 2 (39). С. 4–12. DOI: 10.34130/1992-2752_2021_2_04*

For citation: Golenev I. I., Yermolenko A. V. Designing a neural network for recognizing handwritten cyrillic symbols, *Bulletin of Syktyvkar University. Series 1: Mathematics. Mechanics. Informatics*, 2021, 2 (39), pp. 4–12. DOI: 10.34130/1992-2752_2021_2_04

СГУ им. Питирима Сорокина *Поступила 23.05.2021*