МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Искусственные нейронные сети»

Тема: Распознавание рукописных символов

Студентка гр. 8383 _____ Гречко В.Д.

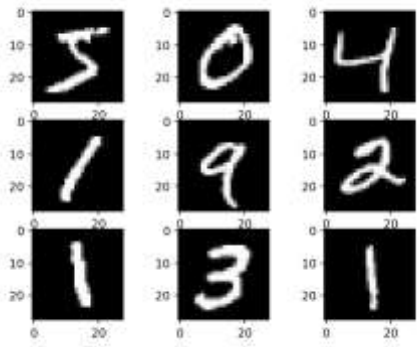
Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Реализовать классификацию черно-белых изображений рукописных цифр (28x28) по 10 категориям (от 0 до 9).



Набор данных содержит 60,000 изображений для обучения и 10,000 изображений для тестирования.

Задачи и требования.

- Ознакомиться с представлением графических данных
- Ознакомиться с простейшим способом передачи графических данных нейронной сети
- Создать модель
- Настроить параметры обучения
- Написать функцию, позволяющая загружать изображение пользователи и классифицировать его

- Найти архитектуру сети, при которой точность классификации будет не менее 95%
- Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения
- Написать функцию, которая позволит загружать пользовательское изображение не из датасета

Выполнение работы.

1. *Поиск архитектуры сети, при которой точность классификации будет не менее 95%*

   Первоначальная архитектура:

```
model = Sequential()
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=128)
```

   Точность: 97.939% - как видно из полученного результата, мы достигли требуемой точности.

2. *Исследование влияние различных оптимизаторов, а также их параметров, на процесс обучения*

Adam:

это метод стохастического градиентного спуска, основанный на адаптивной оценке моментов первого и второго порядков.

learning_rate=0.001,beta_1=0.9, beta_2=0.999, epsilon=1e-7	test_acc in %: 97.64999
learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-7	test_acc in %: 97.43
learning_rate=0.0001, beta_1=0.8, beta_2=0.996, epsilon=1e-9	test_acc in %: 94.29
learning_rate=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-8	test_acc in %: 94.3499
learning_rate=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-8	test_acc in %: 85.5099
learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-8	test_acc in %: 97.79

RMSprop:

Суть RMSProp заключается в том, чтобы:

- Поддерживать скользящее (дисконтированное) среднее значение квадрата градиентов
- Разделите градиент на корень этого среднего

Эта реализация RMSProp использует простой импульс, а не нестеровский импульс. Центрированная версия дополнительно поддерживает скользящее среднее градиентов и использует это среднее значение для оценки дисперсии

learning_rate=0.001, rho=0.9, momentum=0.0, epsilon=1e-7	test_acc in %: 97.64999
learning_rate=0.01, rho=0.9, momentum=0.0, epsilon=1e-7	test_acc in %: 96.45
learning_rate=0.0001, rho=0.9, momentum=0.0, epsilon=1e-9	test_acc in %: 94.45999
learning_rate=0.001, rho=0.8, momentum=0.0, epsilon=1e-9	test_acc in %: 97.6599
learning_rate=0.001, rho=0.7, momentum=0.0, epsilon=1e-9	test_acc in %: 97.69999
learning_rate=0.01, rho=0.7, momentum=0.0, epsilon=1e-8	test_acc in %: 97.64

SGD:

это метод стохастического градиентного спуска, основанный на адаптивной оценке моментов первого и второго порядков.

learning_rate=0.01, momentum=0.0, nesterov=False	test_acc in %: 91.009
learning_rate=0.01, momentum=0.0, nesterov=True	test_acc in %: 91.119
learning_rate=0.001, momentum=0.3, nesterov=True	test_acc in %: 84.17
learning_rate=0.0001, momentum=0.0, nesterov=True	test_acc in %: 32.78
learning_rate=0.1, momentum=0.0, nesterov=True	test_acc in %: 96.02
learning_rate=0.1, momentum=0.6, nesterov=True	test_acc in %: 97.1899

Adadelta:

это стохастический метод градиентного спуска, основанный на адаптивной скорости обучения на измерение для устранения двух недостатков:

- Постоянное снижение скорости обучения на протяжении всего обучения
- Необходимость в выбранном вручную глобальном уровне обучения

Adadelta - это более надёжное расширение Adagrad, которое адаптирует скорость обучения на основе движущегося окна обновлений градиента, а не накапливает все прошлые градиенты. Таким образом, Adadelta продолжает учиться, даже когда было сделано много обновлений.

learning_rate=0.001, rho=0.95, epsilon=1e-07	test_acc in %: 53.619
learning_rate=1.0, rho=0.95, epsilon=1e-07	test_acc in %: 97.86
learning_rate=1.2, rho=0.95, epsilon=1e-09	test_acc in %: 94.9299
learning_rate=1.0, rho=0.89, epsilon=1e-07	test_acc in %: 97.64
learning_rate=1.3, rho=0.73, epsilon=1e-07	test_acc in %: 97.509
learning_rate=1.3, rho=0.99, epsilon=1e-07	test_acc in %: 97.790

Adagrad:

это оптимизатор с определенными параметрами скорости обучения, которые адаптируются относительно того, как часто параметр обновляется во время обучения. Чем больше обновлений получает параметр, тем меньше обновлений.

learning_rate=0.001, initial_accumulator_value=0.1, epsilon=1e-07	test_acc in %: 87.5

learning_rate=0.001, initial_accumulator_value=0.3, epsilon=1e-07	test_acc in %: 84.8999
learning_rate=0.01, initial_accumulator_value=0.01, epsilon=1e-07	test_acc in %: 95.34
learning_rate=0.001, initial_accumulator_value=0.001, epsilon=1e-08	test_acc in %: 91.7599
learning_rate=0.1, initial_accumulator_value=0.01, epsilon=1e-07	test_acc in %: 98.0199
learning_rate=0.1, initial_accumulator_value=0.1, epsilon=1e-07	test_acc in %: 97.63

Adamax:

это вариант Адама, основанный на норме бесконечности. Adamax иногда превосходит Adam, особенно в моделях с вложениями.

learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	test_acc in %: 96.3599
learning_rate=0.001, beta_1=0.8, beta_2=0.888, epsilon=1e-07	test_acc in %: 97.64
learning_rate=0.01, beta_1=0.8, beta_2=0.779, epsilon=1e-07	test_acc in %: 96.10
learning_rate=0.01, beta_1=0.9, beta_2=0.999, epsilon=1e-08	test_acc in %: 97.68
learning_rate=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-08	test_acc in %: 92.11999
learning_rate=0.01, beta_1=0.5, beta_2=0.999, epsilon=1e-08	test_acc in %: 97.99

Nadam:

Подобно тому, как Adam по существу является RMSProp с импульсом, Nadam – это Adam с нестеровским импульсом.

learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07	test_acc in %: 97.5199
learning_rate=0.001, beta_1=0.7, beta_2=0.999, epsilon=1e-08	test_acc in %: 97.829
learning_rate=0.001, beta_1=0.7, beta_2=0.777, epsilon=1e-08	test_acc in %: 97.9399
learning_rate=0.001, beta_1=0.6, beta_2=0.666, epsilon=1e-08	test_acc in %: 97.61
learning_rate=0.01, beta_1=0.8, beta_2=0.787, epsilon=1e-08	test_acc in %: 96.9399
learning_rate=0.001, beta_1=0.8, beta_2=0.787, epsilon=1e-08	test_acc in %: 98.00

Ftrl:

Алгоритм " Follow the Regularized Leader" проистекает из настройки онлайн-обучения, где процесс обучения является последовательным. В этой ситуации онлайн-игрок принимает решение в каждом раунде и терпит поражение. В частности, множество решений является выпуклым множеством в евклидовом пространстве, которое мы можем обозначить через $\mathcal{M} \in \mathbb{R}^n$, а потери-выпуклыми функциями над $\mathcal{M}$.

learning_rate=0.001, learning_rate_power=-0.5, initial_accumulator_value=0.1, l1_regularization_strength=0.0, l2_regularization_strength=0.0, l2_shrinkage_regularization_strength=0.0,beta=0.0	test_acc in %: 18.960
learning_rate=0.01, learning_rate_power=-0.5, initial_accumulator_value=0.6, l1_regularization_strength=0.7, l2_regularization_strength=0.7, l2_shrinkage_regularization_strength=0.9, beta=0.4	test_acc in %: 11.349
learning_rate=0.0001, learning_rate_power=-10.0001, initial_accumulator_value=1.9, l1_regularization_strength=19.000001, l2_regularization_strength=0.0000007,	test_acc in %: 9.65

l2_shrinkage_regularization_strength=8.9, beta=0.6	

Результаты крайне низки, поэтому дальнейшее исследование было прервано.

3. *Написать функцию, которая позволит загружать пользовательское изображение не из датасета*

   Функция predictNumber получает путь к файлу (вводимый пользователем).

   Полученные результаты:

Цифра на картинке	Результат
1	[1] Second: [1]
2	[2] Second: [2]
3	[3] Second: [3]
4	[4] Second: [4]
5	[5] Second: [5]
6	[6] Second: [6]
7	[7] Second: [7]
8	[8] Second: [8]
9	[9] Second: [9]

Выводы.

В ходе выполнения работы была создана искусственная нейронная сеть, осуществляющая распознавание черно-белых изображений цифр от 0 до 9. Также было проведено исследование влияния различных оптимизаторов и их параметров на работу нейронной сети.