RN =21153.0
RN =42837.0
RN =41113.0
RN =27565.0
RN =26321.0

These pseudorandom numbers are the initial numbers for the main 53-bit RNG for the signal processes. For example, for the second block with the initial number RN = 42837.0 first ten numbers of the sequence are the following: >> gen53

RDNO = 856745
RDNO =4283725
RDNO =21418625
RDNO =107093125
RDNO = 535465625
RDNO = 2.6773e+009
RDNO =1.3387e+010
RDNO =6.6933e+010
RDNO =3.3467e+011
RDNO =1.6733e+012

The auxiliary generator should differ from the main RNG in principle of formation of the sequence, otherwise processes can be intercorrelated. During the sessions initial numbers should depend on the number of session. The operator sets only one initial number and this number combine with the number of the communication session in any manner. This algorithm of pseudorandom number sequences generation allow simultaneously and bypass implement the processes in communication channel simulation with the use of only one initial number.

Список цитированных источников

1. Меньших, Т.Ю. Генераторы псевдослучайных чисел для криптографической защиты канала связи // Соврем. пробл. математики и выч. техники: матер. IX Республ. науч. конф. молодых ученых и студентов, Брест, 19–21 ноября 2015 г. – Брест: БрГТУ , 2015. – С. 31–34.

УДК 004.4:004.032.26

KERAS: БИБЛИОТЕКА ГЛУБОКОГО ОБУЧЕНИЯ PYTHON

Хацкевич М.В.
Брестский государственный технический университет, г. Брест
Научный руководитель: Головко В.А., д. т. н., профессор

Keras - мощный простой в использовании API(Application Programming Interface), разработанный на языке программирования Python и способный работать с библиотеками: TensorFlow, CNTK или Theano. Keras применяется для разработки и оценки моделей глубокого обучения. Основная цель Keras – предоставить возможность быстрой разра-

ботки различных моделей нейронных сетей. Keras позволяет разрабатывать сверточные нейронные сети (convolutional neural network, CNN) и рекуррентные нейронные сети (recurrent neural network; RNN), а также разрабатывать комбинированные модели нейронных сетей на базе сверточных нейронных сетей и рекуррентных нейронных сетей. Простота разработки нейронной сети обеспечивает возможность быстрого экспериментирования разработанных моделей нейронных сетей. Современные системы обучения нейронных сетей поддерживают GPU (Graphics Processing Unit) для сокращения времени обучения нейронной сети в несколько раз. Библиотеки Theano и Tensorflow работают с GPU.

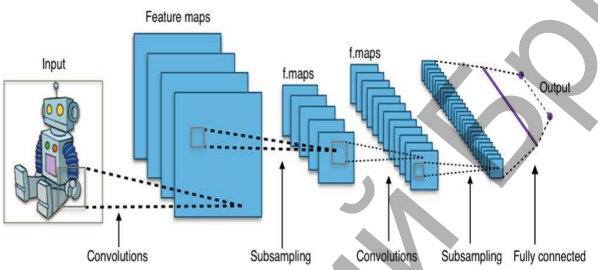

Рисунок 1 – Общая структура сверточной нейронной сети

Для разработки сверточной нейронной сети необходимо следующее: настроить свою среду, установить Keras, импортировать библиотеки и модули, загрузить данные из базы данных (например, CIFAR-10), выполнить предварительную обработку входных данных для Keras, разработать модель нейронной сети, скомпилировать модель, обучать модель на обучающей выборке, оценить модель по тестовым данным.

Общая структура сверточной нейронной сети приведена на рисунке 1.

Простейшим типом модели является последовательная модель, линейная совокупность слоев. Пример создания последовательной модели:

```
# Создаем последовательную модель
from keras.models import Sequential
model = Sequential()
```

Для реализации модели сверточной нейронной сети используются следующие методы:

1) Conv2D - 2D сверточный слой (например, пространственная свертка над изображениями).

Общий формат без списка параметров: keras.layers.Conv2D()

Пример: model.add(Conv2D(32, (3, 3),input_shape=x_train.shape[1:]))

Здесь filters - число выходных фильтров в свертке; kernel_size - ширина и высота окна 2D свертки; input_shape – входные данные;

2) MaxPooling2D – слой пулинга (иначе подвыборки, субдискретизации) представляет собой нелинейное уплотнение карты признаков.

Общий формат без списка параметров:keras.layers.pooling.MaxPooling2D()

Пример: model.add(MaxPooling2D(pool_size=(2, 2))) , где pool_size – размер фильтра;

3) Dropout - слой регуляризации.

Пример: model.add(Dropout(0.25))

Переобучение — это излишне точное соответствие нейронной сети конкретному набору обучающих примеров, при котором сеть теряет способность к обобщению.

dropout с параметром p за одну итерацию обучения проходит по всем нейронам определенного слоя и с вероятностью p полностью исключает их из сети на время итерации. Это заставит сеть обрабатывать ошибки и не полагаться на существование определенного нейрона (или группы нейронов), а полагаться на “единое мнение” (consensus) нейронов внутри одного слоя;

4) Flatten - слой преобразования данных из 2D представления в плоское

Пример: model.add(Flatten())

5) Dense - полносвязный слой для классификации.

Пример: model.add(Dense(512))

6) Dense - выходной полносвязный слой

Пример: model.add(Dense(num_classes)),где num_classes- количество классов

Пример модели сверточной нейронной сети приведен на рисунке 2.

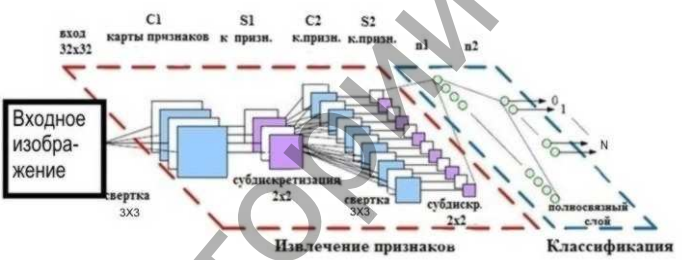

Рисунок 2 – Модель сверточной нейронной сети

Наконец, для обучения модели используют следующие методы:

1) compile – начинает процесс обучения.

Пример, model.compile(loss='categorical_crossentropy', optimizer= 'adam', metrics=['accuracy'])

Здесь loss(имя целевой функции; используем перекрестную энтропию в качестве функции потерь); optimizer='adam' (оптимизатор Адама для градиентного спуска);

metrics (список показателей, которые будут оцениваться моделью во время обучения и тестирования:accuracy(точность));

2) fit - обучает модель фиксированное число эпох.

Метод fit принимает на вход обучающую выборку вместе с метками - x_trainи y_train, размером батча batch_size, который ограничивает количество примеров, подаваемых за раз, количеством эпох epochs, а также тем, какую долю обучающей выборки отдать под валидацию - validation_split(которые будут использоваться в качестве данных проверки достоверности). shuffle(нужно ли перетасовывать образцы в каждую эпоху). Пример: model.fit(x_train,y_train,batch_size=batch_size,epochs=epochs,validation_data=(x_test, y_test), shuffle=True)

3) evaluate - получает на вход тестовую выборку вместе с метками для нее.

Пример: score = model.evaluate(x_test, y_test, batch_size=batch_size)

Здесь x_test – тестовая выборка; y_test- метки; batch_size – количество обучающих образцов, обрабатываемых одновременно за одну итерацию алгоритма.

Результат работы программы после обучения на тестовых данных (CIFAR-10) составляет 76.29% (верных ответов).

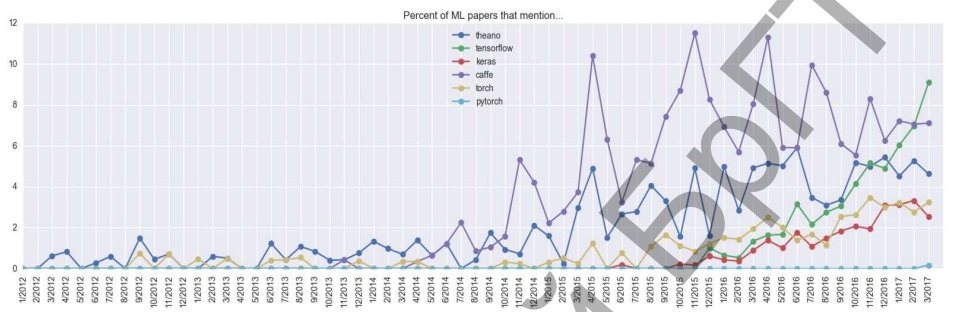

Рисунок 3 – График

Достоинства Keras: удобство и простота использования, модульность, простота модификации, отсутствие отдельных файлов конфигурации. Keras становится все более и более популярным (см рисунок 3).

Список цитированных источников

1. Головко, В.А. Семантическое кодирование на основе глубоких автоассоциативных нейронных сетей / В.А. Головко, А.А. Крощенко, М.В. Хацкевич // Открытые семантические технологии проектирования интеллектуальных систем: Материалы VI Международной научно-технической конференции (Open Semantic Technologies for Intelligent Systems). – Минск : БГУИР, 2016. – с. 313–318.

2. Головко, В.А. Теория глубокого обучения : конвенциальный и новый подход / В.А.Головко, А.А. Крощенко, М.В. Хацкевич // Вестник Брестского государственного технического университета. – 2016. – № 5(101): Физика, математика, информатика. – С. 7 – 15.

УДК 004.032.26

КЛАССИФИКАЦИЯ ФУНКЦИЙ АКТИВАЦИИ В НЕЙРОННЫХ СЕТЯХ

Юхимук Т.Ю.

Брестский государственный технический университет, г. Брест

Научный руководитель: Махнист Л.П., к. т. н., доцент

Всякая нейронная сеть принимает на входе и выдает на выходе числовые значения. Взвешенная сумма S – сумма входных сигналов, умноженная на соответствующие им веса. Функция активации $F(S)$, принимающая взвешенную сумму как аргумент, для