Министерство образования Республики Беларусь
Учреждение образования
"Белорусский Государственный университет информатики
и радиоэлектроники"

Лабораторная работа №2

“**Реализация глубокой нейронной сети”**

по учебной дисциплине “Машинное обучение”

Выполнил: Студент гр. 956241 Дубовик Н.О.

Минск 2020

Данные: В работе предлагается использовать набор данных notMNIST, который состоит из изображений размерностью 28×28 первых 10 букв латинского алфавита (A … J, соответственно). Обучающая выборка содержит порядка 500 тыс. изображений, а тестовая – около 19 тыс.

Данные можно скачать по ссылке:

- https://commondatastorage.googleapis.com/books1000/notMNIST_large.tar.gz (большой набор данных);
- https://commondatastorage.googleapis.com/books1000/notMNIST_small.tar.gz (маленький набор данных);

Описание данных на английском языке доступно по ссылке:
http://yaroslavvb.blogspot.sg/2011/09/notmnist-dataset.html

Результат выполнения заданий опишите в отчете.

Задание 1.

Реализуйте полносвязную нейронную сеть с помощью библиотеки Tensor Flow. В качестве алгоритма оптимизации можно использовать, например, стохастический градиент (Stochastic Gradient Descent, SGD). Определите количество скрытых слоев от 1 до 5, количество нейронов в каждом из слоев до нескольких сотен, а также их функции активации (кусочно-линейная, сигмоидная, гиперболический тангенс и т.д.).

Была создана нейронная сеть с двумя скрытыми слоями, в каждом из которых 128 нейронов с функцией активации ReLU. В выходном слое 10 нейронов с функцией активации “softmax”.

Также был использован оптимизатор Adam и функция потерь “sparse_categorical_crossentropy”.

Реализация представлена ниже, здесь количество эпох равняется 100:

```
model=tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(img_height, img_width)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
  ])

model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=epochs)
    model.evaluate(x_test, y_test)
```

Задание 2.

Как улучшилась точность классификатора по сравнению с логистической регрессией?

По итогам обучения были достигнуты следующие результаты: 94.09% и 90.71% для обучающей и тестовой выборки соответственно. Таким образом, точность классификатора по сравнению с логистической регрессией повысилась на 7%.

Задание 3.

Используйте регуляризацию и метод сброса нейронов (dropout) для борьбы с переобучением. Как улучшилось качество классификации?

Для каждого слоя были добавлены регуляризации с коэффициентом 0.0001 и метод сброса с коэффициентом 0.2:

```
model=tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(img_height, img_width)),
    tf.keras.layers.Dense(128,
activation='relu',kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(128,
activation='relu',kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
  ])

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

  model.fit(x_train, y_train, epochs=epochs)
  model.evaluate(x_test, y_test)
```

Задание 4.

Воспользуйтесь динамически изменяемой скоростью обучения (learning rate). Наилучшая точность, достигнутая с помощью данной модели составляет 97.1%. Какую точность демонстрирует Ваша реализованная модель?

Получилось достигнуть 87.34% процента точности на обучающей и 88.82% процента на тестовой выборках.