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

ОТЧЕТ

по практической работе №2

по дисциплине «Искусственные нейронные сети»

Тема: Создание простой нейронной сети с использованием библиотеки Keras

Студент гр. 8383 _____ Федоров И.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Задача.

Необходимо дополнить предоставленный фрагмент кода моделью ИНС, которая способна провести бинарную классификацию по сгенерированным данным.

Вариант 3: Функция genData():

```
def genData(size=500):
    data = np.random.rand(size, 2)*2 - 1
    label = np.zeros([size, 1])
    for i, p in enumerate(data):
        if (p[0]+0.2)**2 + (0.6*p[1])**2 >= 0.25:
            label[i] = 0.
        else:
            label[i] = 1.
    div = round(size*0.8)
    train_data = data[:div, :]
    test_data = data[div:, :]
    train_label = label[:div, :]
    test_label = label[div:, :]
    return (train_data, train_label), (test_data, test_label)
```

В функции с помощью функции rand() создается массив данных указанной формы (size, 2) случайных чисел, равномерно распределенных (до преобразования от 0 до 1).

Была создана следующая модели, способная провести бинарную классификацию:

```
model = models.Sequential()
model.add(layers.Dense(16, activation='relu',
input_shape=(train_data.shape[1],)))
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

Конечной функцией является функция "sigmoid", выводящая вероятность. При компиляции в качестве функции оптимизации используется "rmsprop", в качестве функции потер - бинарная кросс-энтропия (функция, которая в основном используется при бинарной классификации).

```
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])
```

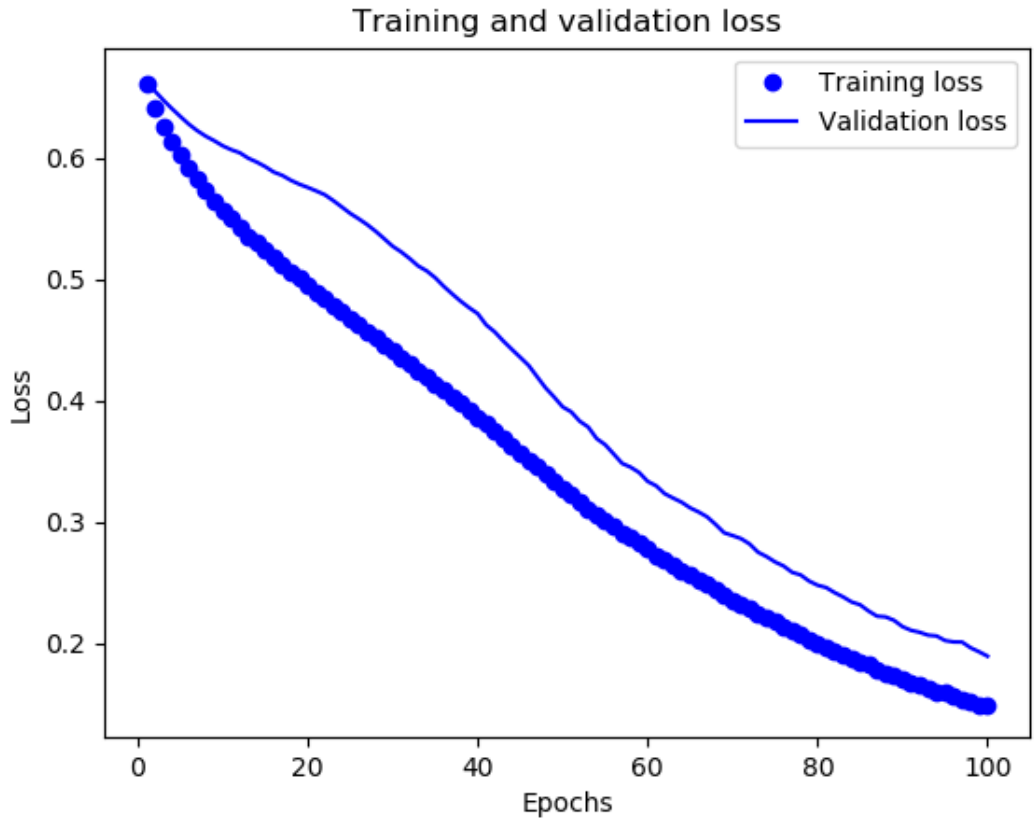
Для контроля точности модели в процессе обучение выделим проверочный набор, выбрав 20% от набора для обучения:

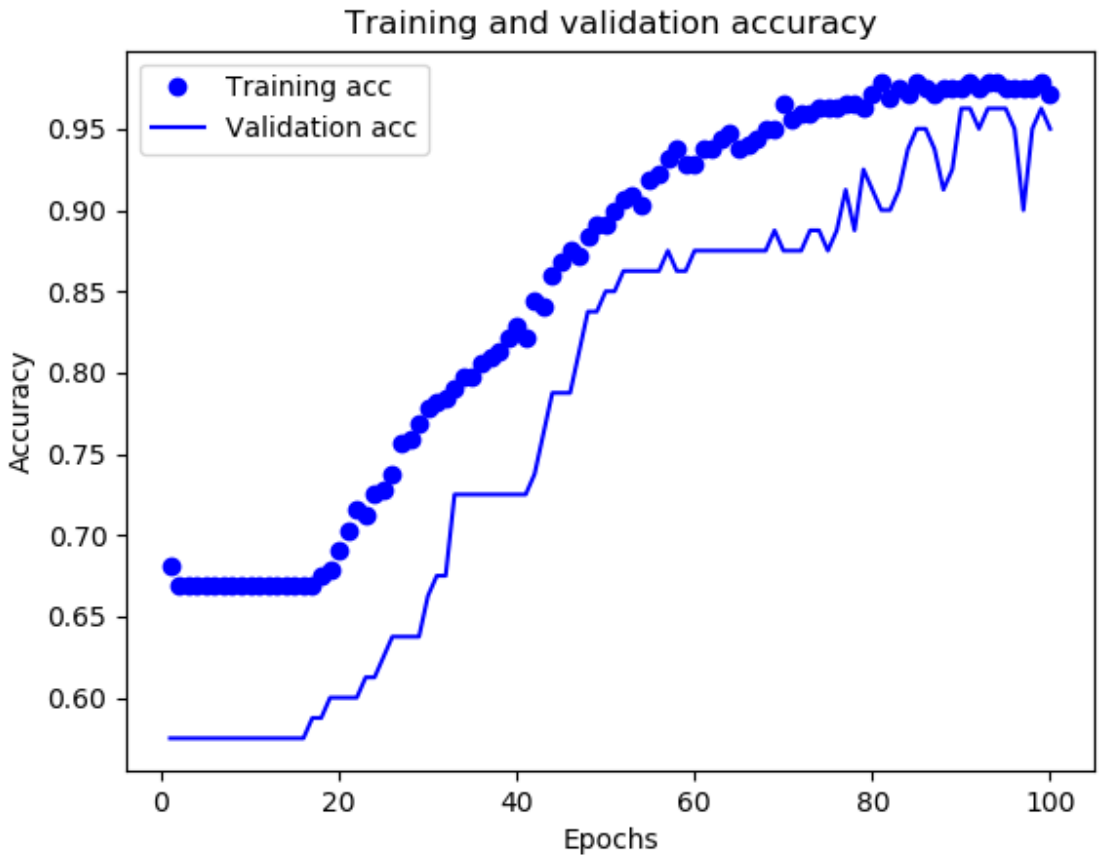
```
train_size = round(len(train_data) * 0.8)
val_data_x = train_data[train_size:, :]
part_x_train = train_data[:train_size, :]
val_data_y = train_label[train_size:]
part_y_train = train_label[:train_size]
```

Было произведено обучение модели и получен класс History, который используется для отображения графиков ошибок и точности:

```
H = model.fit(part_x_train, part_y_train,
                                epochs=80,
                                batch_size=50,
                                validation_data=(val_data_x, val_data_y),
                                verbose=2)
```

Ниже представлены графики ошибок и точности:

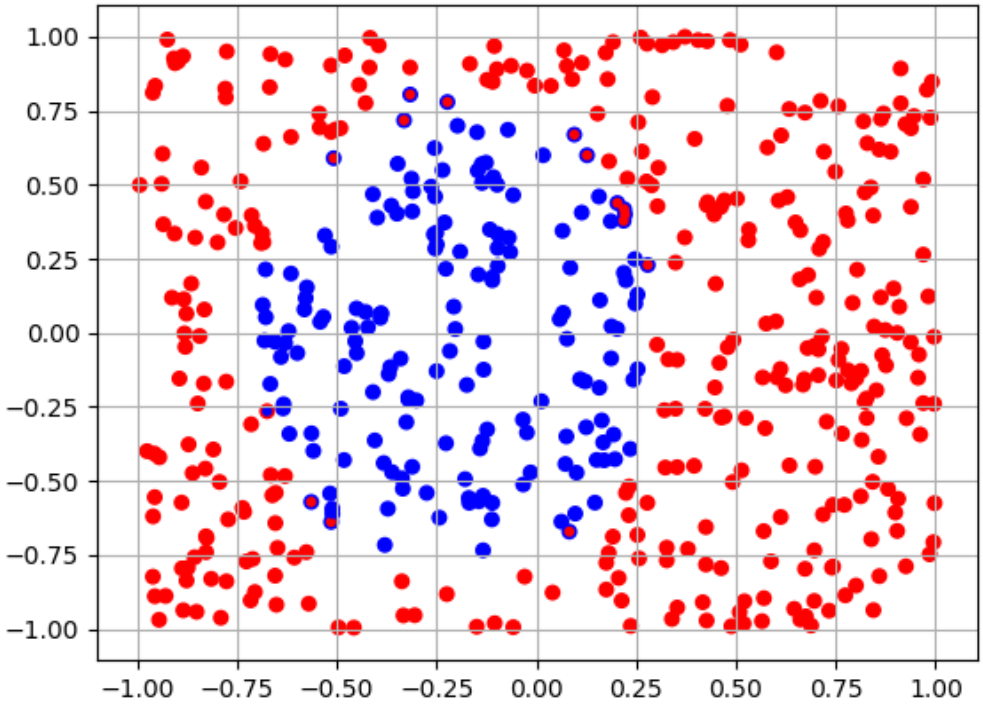




Была выполнена оценка модели на контрольных данных:

```
results = model.evaluate(test_data, test_label)
print(results)
#Результат [0.1642639 0.94]
```

Предоставленная функция drawResults() позволяет визуализировать результаты и ошибки (точки предсказанные моделью с помощью функции predict() и точки данных разного размера, благодаря чему видно где произошло наложение):



При большем объеме данных: