с помощью своего приватного ключа.

7. Боб отправляет другую часть Алисе.

8. Алиса объединяет оба сообщения Боба и расшифровывает их с помощью своего приватного ключа.

УДК 004.89

Студ. И.А. Старовойтов
Науч. рук. доц., канд. техн. наук Г.Л. Тимонович
(Кафедра информационных систем и технологий, БГТУ)

ОБНАРУЖЕНИЕ И ИЗБЕЖАНИЕ ПЕРЕОБУЧЕНИЯ В НЕЙРОННЫХ СЕТЯХ

Переобучение представляет собой распространенную проблему при обучении нейронных сетей. Когда модель слишком хорошо запоминает обучающие данные, она может плохо обобщать на новые, ранее не виданные примеры. Это может проявляться в форме высокой точности на обучающем наборе, но низкой точности на тестовых данных. Для примера создадим модель, не содержащую инструментов, способных предотвратить переобучение. Ее код представлен на рисунке 1.

```
model_overfit = tf.keras.Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(512, activation='relu'),
    layers.Dense(512, activation='relu'),
    layers.Dense(10, activation='softmax')
])
 model_overfit.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
                       metrics=['accuracy'])
```

Рисунок 1 – Модель с переобучением

После обучения модели создадим график, демонстрирующий потери на тестовом и тренировочном наборах данных, представленный на рисунке 2.

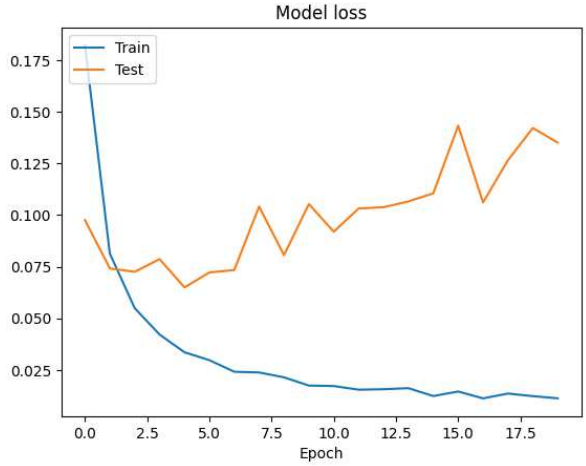

Рисунок 2 – График потерь модели

На графике можно заметить, что модель переобучена. Это можно заметить по тому, что потери на наборах данных начинают расходиться в противоположные стороны после пересечения.

Для устранения проблемы переобучения было использовано несколько различных методик:

1. Добавление Dropout-слоев, позволяющих упростить строение модели;

2. Добавление l2-регуляризации;

3. Создание функции ранней остановки, используемой в случае уменьшения точности модели.

Код, используемый для построения модели, представлен на рисунке 3.

```
model_regularized = tf.keras.Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(256, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    layers.Dropout(0.1),
    layers.Dense(256, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
    layers.Dropout(0.1),
    layers.Dense(10, activation='softmax')
])
early_stopping = tf.keras.callbacks.EarlyStopping(patience=2)
```

Рисунок 3 – Оптимизированная модель

После обучения данной модели был получен график потерь, представленный на рисунке 4.

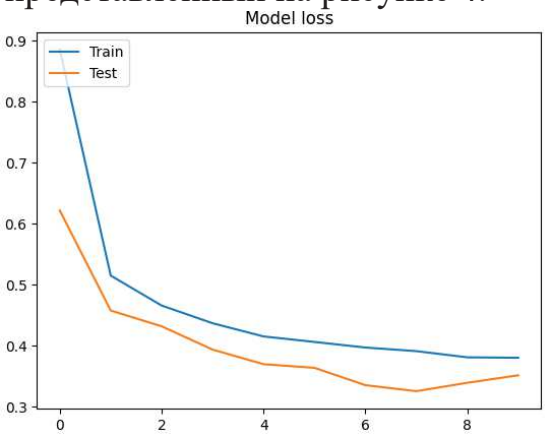

Рисунок 4 – График потерь с использованием инструментов предотвращения переобучения

В отличие от переобученной сети, потери на наборах данных не пересекались между собой, что указывает на оптимальное обучение и устранение рассматриваемой проблемы. На данном этапе данная нейронная сеть будет показывать наилучший результат, что существенно повышает ее эффективность. Существует большое количество различных инструментов, использование которых позволит избежать проблем переобучения, возникающих при обучении в нейронных сетях. При этом минимальное их использование значительно упрощает процесс, позволяя уменьшить количество затрачиваемых ресурсов на обучение нейронной сети.