|  |  |
| --- | --- |
| |  | | --- | | МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | |

Институт Информационных Технологий

Кафедра корпоративных информационных систем (КИС)

**ОТЧЕТ**

по Практической работе № 13-16

на тему

по дисциплине

«Интеллектуальные системы и технологии»

Выполнил студент группы ИКМО-03-22 Килюп Л.А.

Принял к.т.н., доцент Демидова Л.А.

Выполнено «\_\_\_» \_\_\_\_\_\_\_\_\_2022 г.

Зачтено «\_\_\_» \_\_\_\_\_\_\_\_\_2022 г.

**СОДЕРЖАНИЕ**

[Задание 3](#_Toc91268135)

[Ход работы 4](#_Toc91268136)

[Выводы 13](#_Toc91268137)

# Задание

Выполнить исследование и сравнительный анализ возможностей RNN, LSTM и GRU на примере предлагаемого кода.

[**https://github.com/Azure/lstms\_for\_predictive\_maintenance/blob/master/Deep%20Learning%20Basics%20for%20Predictive%20Maintenance.ipynb**](https://github.com/Azure/lstms_for_predictive_maintenance/blob/master/Deep%20Learning%20Basics%20for%20Predictive%20Maintenance.ipynb)

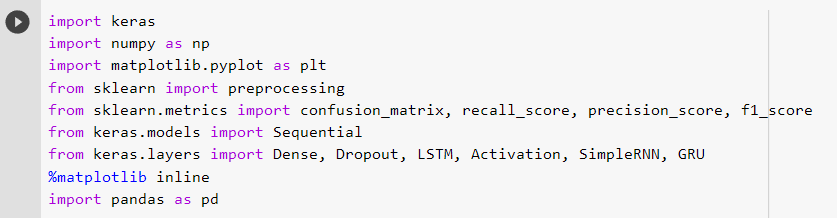
Выполнить работу по варианту, соответствующему номеру с id авиационного двигателя в наборе данных.

1. Сравнить полученные нейронные сети по Accuracy, Precision, Recall, F1, Loss на train и test.
2. Выполнить исследования на примере фрагмента кода с заменой LSTM на RNN и GRU.
3. Исследовать, как определяется число параметров Param в каждом слое.
4. Изучить и описать назначение используемых методов и параметров.
5. Вывести графические зависимости для Loss и Accuracy на train и val (на обучающей и валидационной подвыборках).
6. Оценить время разработки классификаторов с CPU.
7. Оценить время разработки классификаторов с GPU (в Google Colab).

# Ход работы

В данной практической работе мы будем использовать библиотеку Keras, которая представляет собой надстройку над фреймворком TensorFlow. Данная библиотека позволяет работать с искусственными нейронными сетями. Работать будем в среде Google Colab.

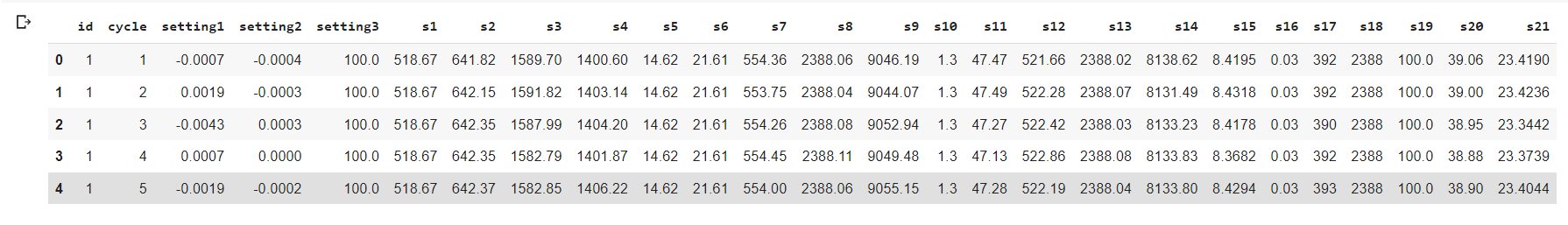
**Рисунок 1 – импорт библиотек**



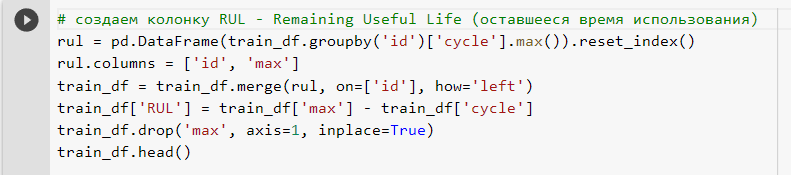
Работать будем с данными по обслуживанию авиационных двигателей, предлагаемый датасет состоит из тренировочной выборки, тестовой выборки и выборки для проверки обученной модели (train, test, ground truth). Данные обучения состоят из нескольких многомерных временных рядов с «циклом» в качестве единицы времени, вместе с показаниями 21 датчика для каждого цикла. В обучающей выборке не указано когда именно произошел сбой двигателя. Наша задача – протестировать различные модели искусственной нейронной сети, которая будет предсказывать произойдет ли сбой двигателя или нет, то есть решается задача бинарной классификации. Модели будем строить с использованием различных слоев LSTM, GRU, RNN.

Часть тренировочной выборки отображена на рисунке 2: в первой колонке – номер двигателя (id), во второй – номер цикла его работы. В других колонках – показания датчиков и настроек.

**Рисунок 2 – общий вид тренировочного набора данных**

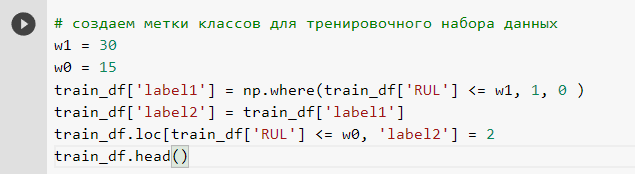


Для начала подготовим данные к обработке. Добавим колонку оставшегося времени работы двигателя (рис. 3).



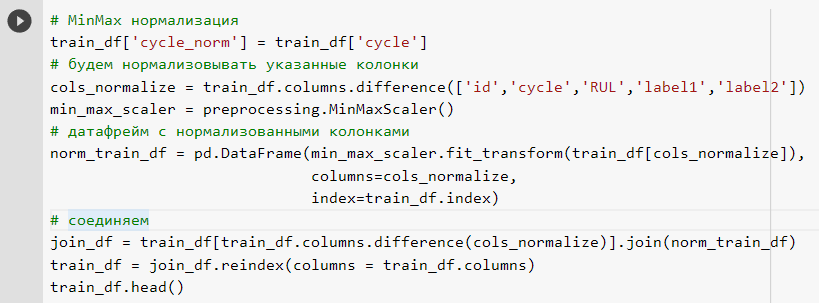
**Рисунок 3**

Затем необходимо создать метки классов для тренировочного набора данных (рис.4).



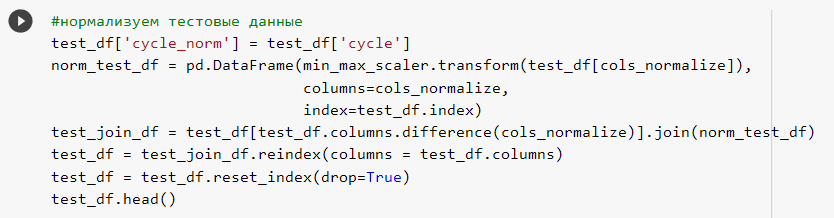
**Рисунок 4**

Далее нормализуем тренировочный набор данных при помощи MinMax нормализации (рис. 5).



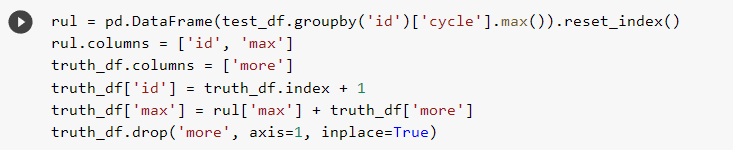
**Рисунок 5**

Нормализуем также тестовую выборку (рис. 6).

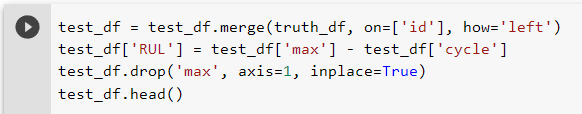


**Рисунок 6**

Далее – создадим колонки max и RUL для тестовой выборки (оставшееся время работы двигателя), используя проверочную выборку (ground truth) (рис. 7, рис.8).

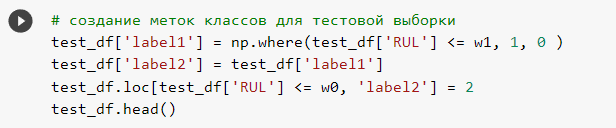


**Рисунок 7**



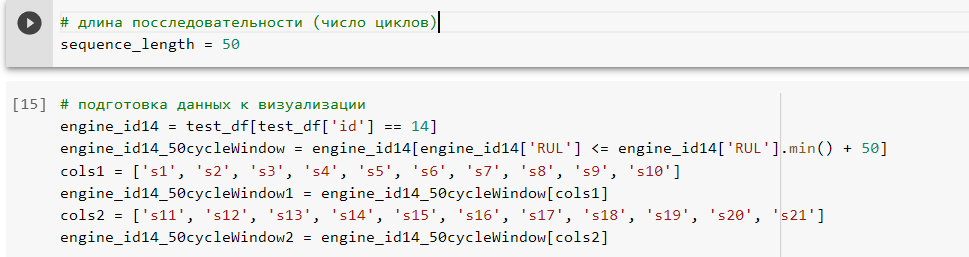
**Рисунок 8**

Создадим метки классов для тестовой выборки (рис. 9).



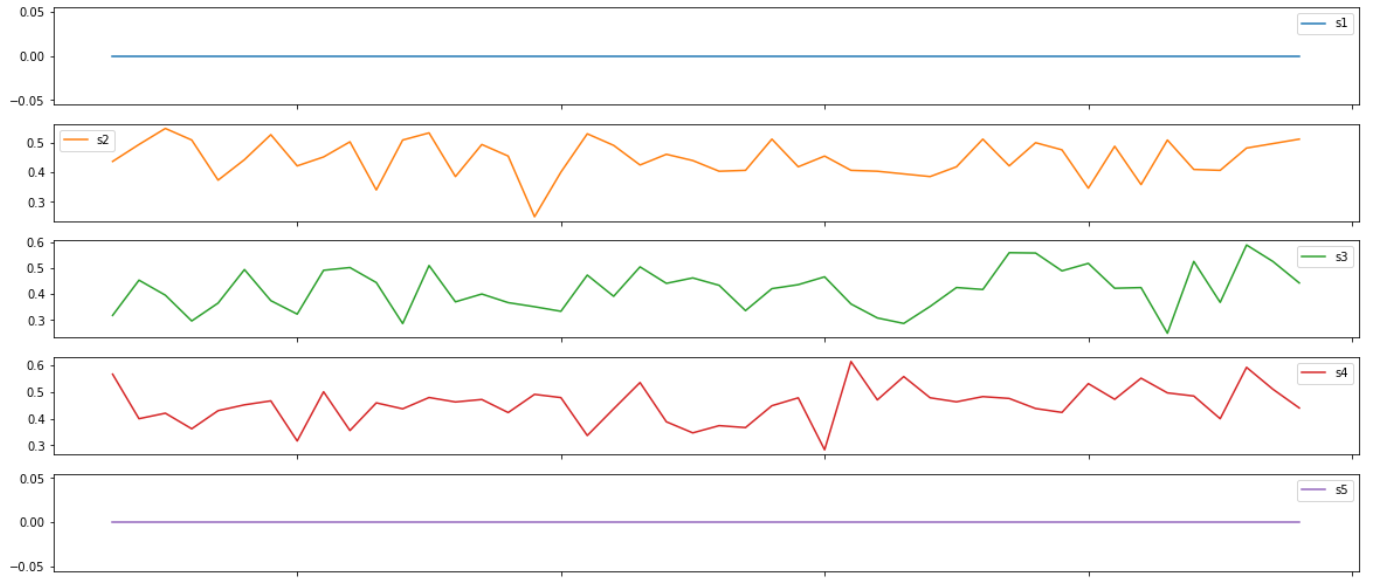
**Рисунок 9**

Прежде чем создавать модель искуственной нейронной сети определимся с размером входных параметров. Возьмем окно, состоящее из 50 циклов обслуживания двигателя. Двигатель возьмем с id под номером 14 (рис. 10). Визуализируем данные.



**Рисунок 10**

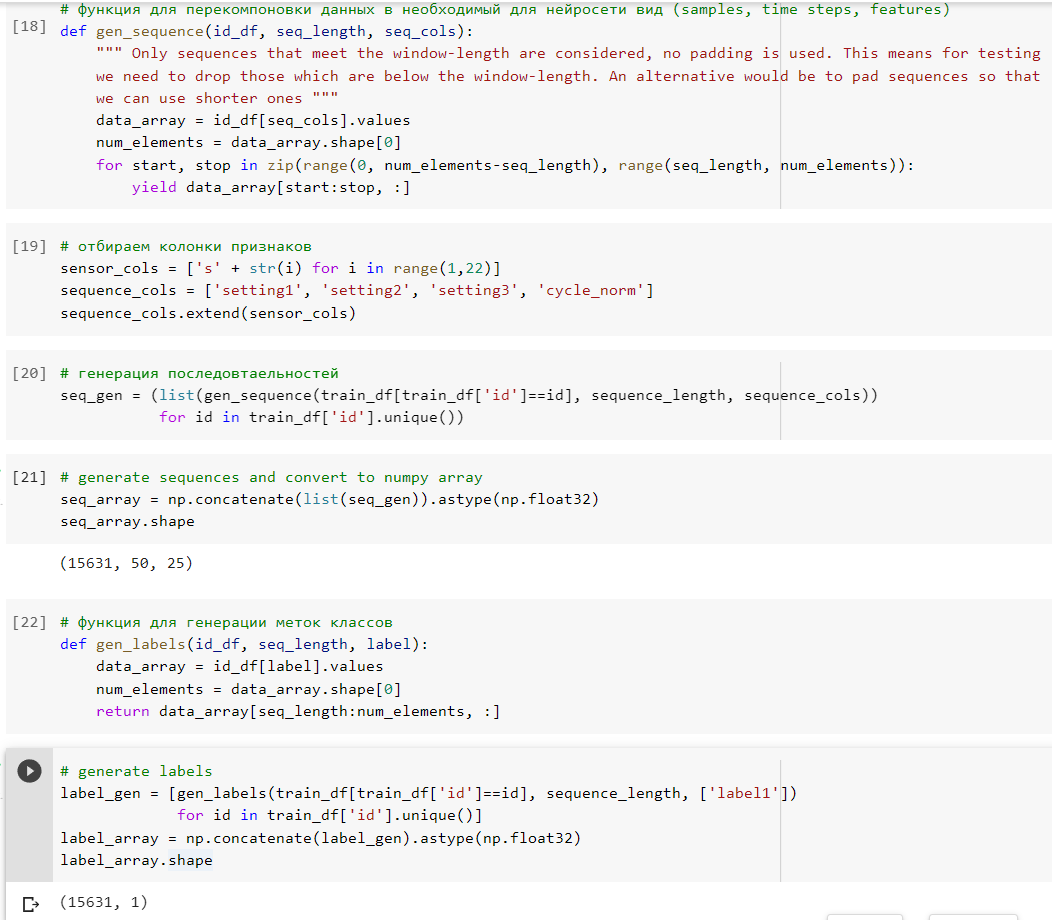
В качестве примера для визуализации возьмем результаты измерений первых 5 датчиков (рис. 11). По оси x – цикл обслуживания двигателя, по оси y – результат измерения.



**Рисунок 11**

Слои нейронной сети Keras ожидают на входе данные в виде массива из 3-х измерений (образцы (samples), временные шаги, признаки (features)), где образцы - это те последовательности, которые будут поставляться в нейронную сеть, временные шаги - это окно ретроспективного обзора или длина последовательности, а признаки – результаты измерений (отдельные характеристики образца).

Поэтому создадим вспомогательные функции для подготовки данных для искусственной нейросети (рис. 12).

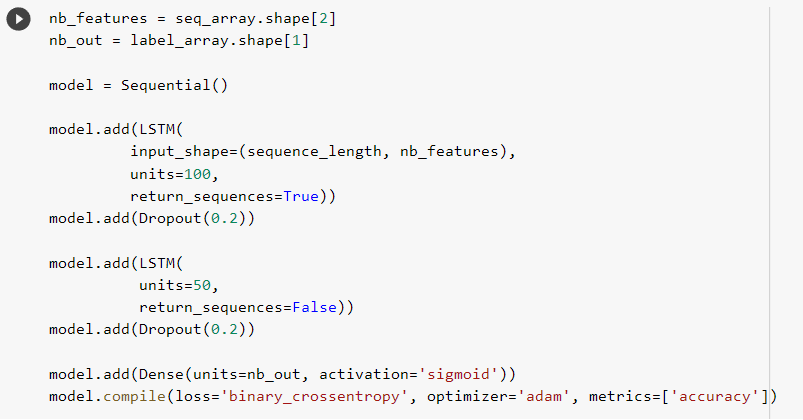


**Рисунок 12**

Применив функцию gen\_sequence получим массив seq\_array, который затем будет использоваться при обучении нейросети, как видим получившийся массив имеет размерность (15631,50,25). Таким образом мы получили 15631 результат измерений, которые затем будут подаваться в нейронную сеть по 50 результатов за 1 раз. Каждый результат измерений будет иметь 25 признаков (показания датчиков).

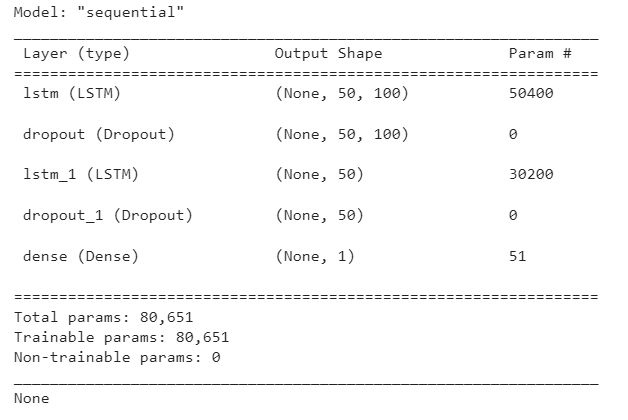
Функция gen\_labels выделит нам метки классов в отдельный массив. Данные метки нужны будут при обучении нейросети.

Затем при помощи библиотеки Keras построим модель нейронной сети (рис. 13).



**Рисунок 13**

Посмотрим её характеристики: Первый слой имеет размерность 50 на входе и 100 на выходе. 2-й слой – размерность 50 на выходе. Выходной слой имеет размерность 1 на выходе. Также после каждого слоя на выходе применяется отсев (Dropout) (кроме последнего слоя), который позволяет контролировать переобучение (запоминание тренировочной выборки). В данном случае первые два слоя имеют тип LSTM – рекуррентные слои с долгой краткосрочной памятью (рис. 14).

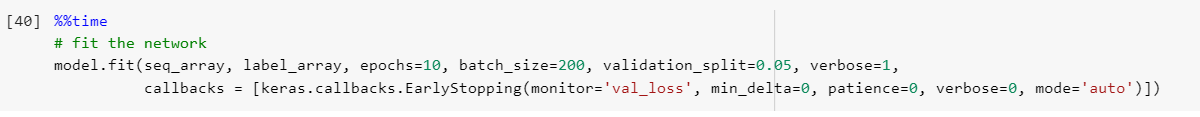


**Рисунок 14**

В качестве функции потерь выбрана бинарная кросс-энтропия (применяется для решения задачи бинарной классификации), в качестве функции оптимизации – модификация градиентного спуска – Adam. Эти функции отвечают непосредственно за обучение нейросети. Функция потерь показывает ошибку нейронной сети, а Adam позволяет делать шаг в сторону убывания функции потерь. Обучить нейронную сеть означает минимизировать функцию потерь.

При помощи функции fit запустим процесс обучения получившейся модели(рис.15). На вход данная функция принимает тренировочную выборку, массив меток классов.

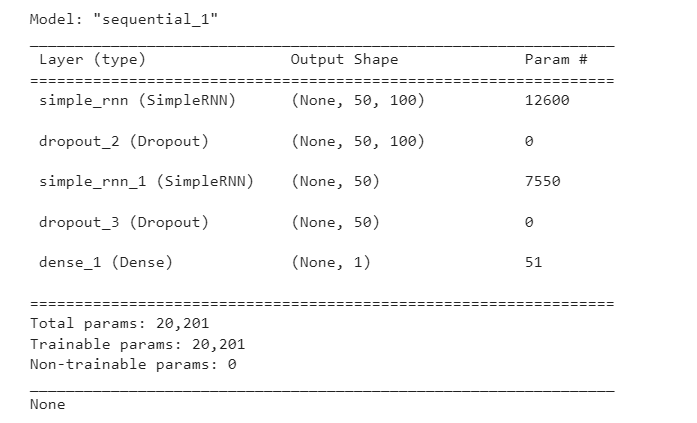
Также есть другие параметры, отметим наиболее важные из них: epochs – количество эпох обучения. За одну эпоху нейронная сеть просматривает всю тренировочную выборку; Поскольку вся тренировочная выборка может занимать много места в памяти, данные в нейронную сеть подаются отдельными блоками – батчами (batch size), а не целиком.



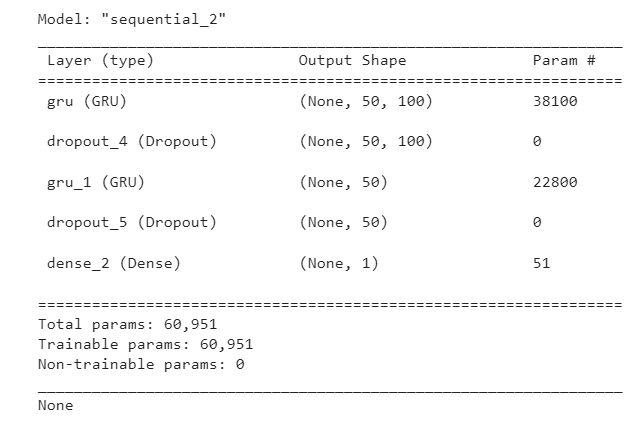
**Рисунок 15**

Параметр validation\_split позволяет создать валидационную выборку из тренировочной, чтобы можно было во время обучения отслеживать значения точности, валидационная выборка не используется для обучения нейросети.

Затем повторим то же самое, заменив LSTM слои на GRU (управляемые рекуррентные блоки), а затем на RNN – простая рекуррентная нейронная сеть (рис. 16, рис. 17).

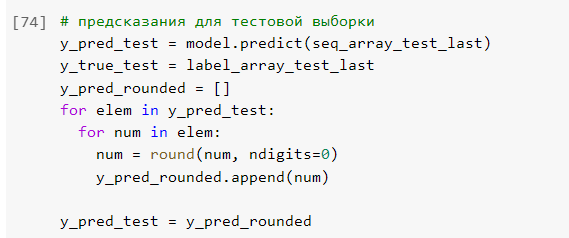


**Рисунок 16**



**Рисунок 17**

Теперь проверим модели на тестовой выборке и сравним результаты предсказания метки класса полученными моделями (рис. 18).



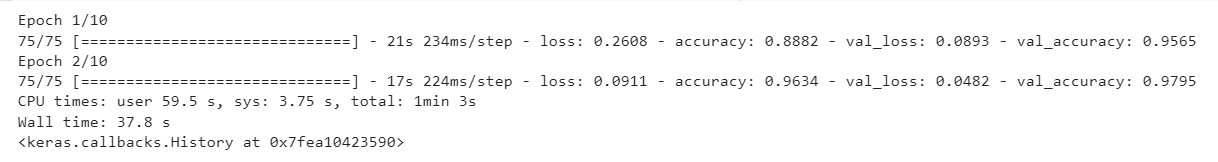
**Рисунок 18**

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Precision | Recall | F1-Score |
| **LSTM** | 0.989247 | 0.961538 | 1.0 | 0.980392 |
| **GRU** | 0.946237 | 0.954545 | 0.84 | 0.893617 |
| **SimpleRNN** | 0.978495 | 0.960000 | 0.96 | 0.960000 |
|  |  |  |  |  |
|  |  |  |  |  |

Как видно из таблицы 1, лучшие показатели метрик оказались у нейросети с LSTM-слоями.

Отметим, что в каждом случае обучение автоматически завершалось на 2-й эпохе, поэтому график со значениями функции потерь на каждой эпохе визуализировать нет необходимости (рис. 19).



**Рисунок 19**

Сравним время обучения нейросетей при их запуске на GPU и CPU (таблица 2).

Таблица 2

|  |  |  |
| --- | --- | --- |
|  | CPU | GPU |
| **LSTM** | 63 сек | 9,13 сек |
| **GRU** | 57,9 сек | 8,58 сек |
| **SimpleRNN** | 20,8 сек | 31,3 сек |

Время обучения LSTM и GRU моделей гораздо быстрее выполняется на графическом процессоре. Обучение модели с RNN было быстрее на центральном процессоре.

# Выводы

В рамках практической работы было проведено сравнение трех моделей нейросетей с различными типами слоев – LSTM, GRU, RNN. В процессе обучения решалась задача бинарной классификации для предсказания выхода из строя двигателя c id равным 14. Лучшие показатели точности, полноты и F1-меры оказались у нейросети с LSTM слоями. Также было выявлено, что вычисления на графическом процессоре (GPU) происходят значительно быстрее у нейросетей с LSTM, GRU слоями чем на CPU.