ДИСЦИПЛИНА	**Интеллектуальные системы и технологии**
	(полное наименование дисциплины без сокращений)
ИНСТИТУТ	**информационных технологий**
КАФЕДРА	**корпоративных информационных систем**
	полное наименование кафедры
ВИД УЧЕБНОГО МАТЕРИАЛА	**Лекция**
	(в соответствии с пп.1-11)
ПРЕПОДАВАТЕЛЬ	**Демидова Лилия Анатольевна**
	(фамилия, имя, отчество)
СЕМЕСТР	**1 семестр (осенний), 2024 – 2025 учебный год**
	(семестр обучения, учебный год)

ЛЕКЦИЯ 7
РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

Рекуррентные нейронные сети – подкласс нейронных сетей с обратной связью между различными слоями нейронов. Их характерной особенностью является передача сигналов с выходного (output layer) или скрытого слоя (hidden layer) нейронной сети во входной слой (input layer). Как и любая другая нейронная сеть, рекуррентная нейронная сеть может состоять из любого числа слоев.

Главная особенность, выделяющая эти сети среди других, – динамические зависимости на каждом этапе функционирования. Вследствие обратной связи типа «один ко многим», изменение состояния одного нейрона ведет к изменению состояния всей сети в целом.

Для обучения рекуррентных нейронных сетей применяется алгоритм обратного распространения ошибки по времени, который является вариантом алгоритма обратного распространения ошибки, используемый для нейронных сетей прямого распространения сигнала. Данный алгоритм определяет стратегию подбора весов нейронной сети с применением градиентных методов оптимизации. При этом решается задача минимизации суммарных потерь на всех объектах обучающей выборки:

$$Q(W) = \sum_{t=1}^{n} \mathcal{L}_t(W) \to \min_W \qquad (1)$$

где W – набор весов нейронной сети; n – число объектов в обучающей выборке; $\mathcal{L}_t(W)$ – функция потерь на t-м объекте x_t $(t = \overline{1,n})$; $x_t \in \Re^L$.

Функция потерь $\mathcal{L}_t(W)$ на t-м объекте x_t $(t = \overline{1,n})$ может быть представлена как квадратичная функция вида:

$$\mathcal{L}_t(W) = \frac{1}{2} \cdot \sum_{m=1}^{M} (h_t^{(m)}(W) - y_t^{(m)})^2, \qquad (2)$$

где M – число выходных нейронов; $y_t^{(m)}$ – ожидаемое значение на m-м нейроне для t-го объекта; $h_t^{(m)}(W)$ – фактическое значение на m-м нейроне для t-го объекта ($t = \overline{1,n}$).

Уточнение весов нейронной сети может проводиться после предъявления каждого обучающего объекта либо однократно после предъявления всех объектов, участвующих в цикле обучения.

Цель обучения заключается в таком определении значений весов нейронов каждого слоя сети, чтобы при заданном входном векторе x_i сети, соответствующем *i*-му объекту, получить на j-м выходном нейроне значение $h_t^{(m)}(W)$, совпадающее с требуемой точностью с ожидаемым значением $y_t^{(m)}$ ($t = \overline{1,n}$; $m = \overline{1,M}$).

Пусть нейронная сеть имеет K нейронов в скрытом слое, M нейронов в выходном слое, а объект x_t во входном слое нейронной сети имеет L характеристик.

Обучение сети с использованием алгоритма обратного распространения ошибки проходит следующим образом.

На каждом t-м шаге обучения нейронной сети предъявляется некоторый входной вектор x_t, для которого известно ожидаемое выходное значение y_t. Для этого входного вектора x_t рассчитываются веса нейронов скрытого слоя (скрытых слоев, если их несколько), а затем – веса нейронов выходного слоя, с учетом значений которых и вычисляется значение функции (1), которая должна быть минимизирована.

Выходной вектор k_t ($k_t \in \Re^K$) скрытого слоя нейронной сети на каждом t-м шаге обучения вычисляется как:

$$k_t = \sigma_k(W_k \cdot x_t + U_k \cdot k_{t-1} + b_k), \qquad 3)$$

где x_t ($x_t \in \Re^L$) – входной вектор нейронной сети на t-м шаге; W_k ($W_k \in \Re^{L \times K}$) – матрица весов для входного вектора x_t; U_k ($U_k \in \Re^{K \times K}$) – матрица весов для нейронов скрытого слоя; b_k ($b_k \in \Re^K$) – вектор смещения скрытого слоя (при этом матрицы W_k, U_k и вектор b_k вычисляются в процессе обучения); σ_k – функция активации для скрытого слоя.

На рисунке 1 показана простая рекуррентная нейронная сеть, а также ее представление в развернутом виде.

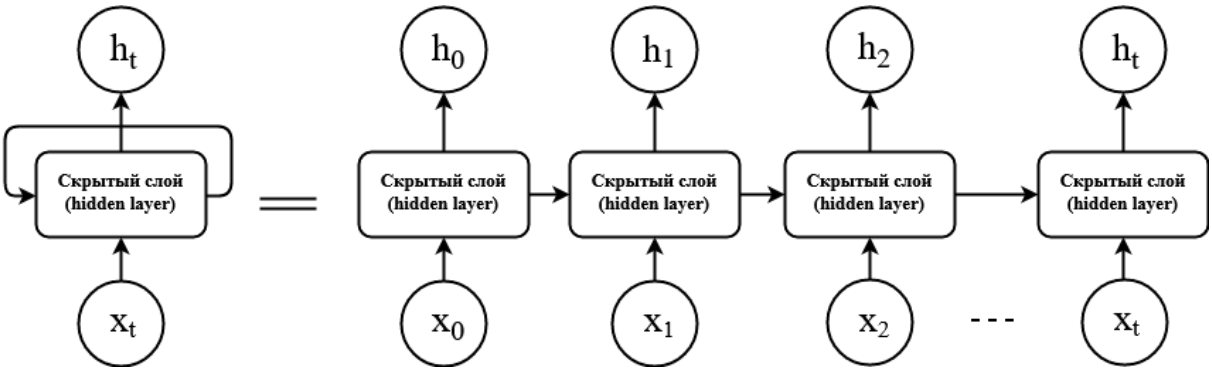

Рисунок 1 – Архитектура рекуррентной нейронной сети с одним входным нейроном, одним скрытым нейроном и одним выходным нейроном и ее развернутое представление

В простой RNN-сети выходной вектор h_t ($h_t \in \Re^M$) на t-м шаге вычисляются как:

$$h_t = \sigma_h(W_h \cdot k_t + b_h) \quad (4)$$

где W_h ($W_h \in \Re^{M \times K}$) – матрица весов для нейронов выходного слоя; b_h ($b_h \in \Re^M$) – вектор весов для нейронов выходного слоя; σ_h – функция активации для выходного слоя.

Функция активации – функция, принимающая некоторую взвешенную сумму S как аргумент. Значение функции активации и будет являться иско-

мым значением на выходном нейроне. В таблице 1 представлены примеры некоторых функций активации.

Таблица 1 – Функции активации нейронов

Название	Формула	Область значений
Пороговая	$\sigma(S) = \begin{cases} 0, S < \theta \\ 1, S \geq \theta \end{cases}$	$(0;\ 1)$
Линейная	$\sigma(S) = aS$	$(-\infty;\ +\infty)$
Сигмоидальная	$\sigma(S) = \frac{1}{1 + e^{-aS}}$	$(0;\ 1)$
Гиперболический тангенс	$\sigma(S) = \frac{e^{aS} - e^{-aS}}{e^{aS} + e^{-aS}}$	$(-1;\ 1)$

Память, реализуемая в архитектуре RNN-сети, является короткой, так как на каждом шаге обучения информация в памяти смешивается с новой информацией и через несколько шагов полностью перезаписывается.

LSTM-сеть

Нейронная сеть с долгой краткосрочной памятью (LSTM – long short-term memory) – LSTM-сеть – представляет собой одну из разновидностей рекуррентных нейронных сетей. Схематично архитектура LSTM-сети представлена на рисунке 2.

Ключевой компонент LSTM-сети – это так называемое состояние ячейки (cell state), которому сопоставлена горизонтальная линия в верхней части центрального блока на рисунке 2. Состояние ячейки участвует в нескольких линейных преобразованиях. Состояние ячейки отвечает за процесс обучения, обратное распространение ошибки и обновление весов.

LSTM-сеть имеет возможность удалять и добавлять информацию в состояние ячейки, используя вентили (гейты, gates).

Вентили позволяют пропускать информацию на основании некоторых условий. Вентили состоят из сигмоидального слоя (то есть предполагают использование сигмоидальной функции активации) и операции поточечного умножения. Сигмоидальный слой возвращает число в диапазоне от «0» до «1», определяющее, какую долю каждого блока информации следует пропустить дальше по сети. Число «0» означает «не пропускать ничего», а число «1» – «пропустить все».

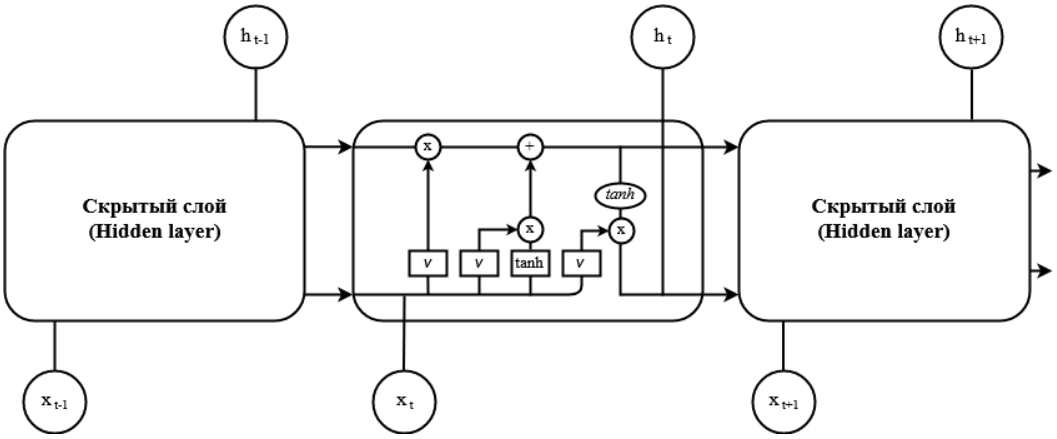

Рисунок 2 – Архитектура LSTM-сети в развернутом представлении

Пусть имеется LSTM-сеть с сигмоидальной функцией активации.

Работу LSTM-сети можно описать следующей последовательностью шагов.

На первом шаге обучения LSTM-сеть определяет, какую информацию можно удалить из состояния ячейки. Это решение принимает сигмоидальный слой, называемый forget gate layer – «слой вентиля забывания».

Он «анализирует» значения входного вектора x_t ($x_t \in \Re^L$) и выходного вектора h_{t-1} ($h_{t-1} \in \Re^M$), определяя выходной вектор f_t ($f_t \in \Re^K$) слоя вентиля забывания как:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right), \quad (5)$$

где h_{t-1} ($h_{t-1} \in \Re^M$) – выходной вектор нейронной сети на $(t-1)$-м шаге; x_t ($x_t \in \Re^L$) – входной вектор нейронной сети на t-м шаге; σ – функция акти-

вации; W_f ($W_f \in \Re^{L\times K}$) – матрица весов для входного вектора x_t; U_f ($U_f \in \Re^{K\times K}$) – матрица весов для выходного вектора h_{t-1} нейронной сети на $(t-1)$-м шаге; b_f ($b_f \in \Re^K$) – вектор смещения слоя вентиля забывания.

На втором шаге LSTM-сеть определяет, какая новая информация будет храниться в состоянии ячейки. Этот шаг состоит из двух этапов.

На первом этапе сигмоидальный слой input layer gate решает, какую информацию следует обновить, определяя выходной вектор i_t ($i_t \in \Re^K$) как:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \quad 6)$$

где h_{t-1} ($h_{t-1} \in \Re^M$) – выходной вектор нейронной сети на $(t-1)$-м шаге; x_t ($x_t \in \Re^L$) – входной вектор нейронной сети на t-м шаге; σ – функция активации; W_i ($W_i \in \Re^{L\times K}$) – матрица весов для входного вектора x_t; U_i ($U_i \in \Re^{K\times K}$) – матрица весов для выходного вектора h_{t-1} нейронной сети на $(t-1)$-м шаге; b_i ($b_i \in \Re^K$) – вектор смещения.

На втором этапе *tanh*-слой вычисляет новый вектор состояния ячейки C'_t ($C'_t \in \Re^K$):

$$C'_t = tanh(W_c x_t + U_c h_{t-1} + b_c), \quad 7)$$

где h_{t-1} ($h_{t-1} \in \Re^M$) – выходной вектор нейронной сети на $(t-1)$-м шаге; x_t ($x_t \in \Re^L$) – входной вектор нейронной сети; $tanh$ – тангенсальная функция активации; W_c ($W_c \in \Re^{L\times K}$) – матрица весов для входного вектора x_t; U_c ($U_i \in \Re^{K\times K}$) – матрица весов для выходного вектора h_{t-1} нейронной сети на $(t-1)$-м шаге; b_c ($b_c \in \Re^K$) – вектор смещения *tanh*-слоя.

На третьем шаге LSTM-сеть изменяет старое состояние ячейки C_{t-1} ($C_{t-1} \in \Re^K$) на новое состояние C_t ($C_t \in \Re^K$):

$$C_t = f_t * C_{t-1} + i_t * C'_t,$$

8)

где f_t ($f_t \in \Re^K$) – выходной вектор слоя вентиля забывания, i_t ($i_t \in \Re^K$) – выходной вектор слоя входного вентиля.

На четвертом шаге на основе состояния ячейки C_t, к которому применены вентили, вычисляется выходной вектор h_t ($h_t \in \Re^M$).

При этом сначала сигмоидальный слой решает, какая информация будет выводиться, вычисляя выходной вектор o_t ($o_t \in \Re^K$) сигмоидального слоя:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o), \quad 9)$$

где h_{t-1} ($h_{t-1} \in \Re^M$) – выходной вектор значений нейронной сети на $(t-1)$-м шаге, x_t – входной вектор значений нейронной сети, C_t ($C_t \in \Re^K$) – текущее состояние ячейки, σ – функция активации, W_o ($W_o \in \Re^{L \times K}$) – матрица весов для входного вектора x_t; U_o ($U_o \in \Re^{K \times K}$) – матрица весов для выходного вектора h_{t-1} нейронной сети на $(t-1)$-м шаге; b_o ($b_o \in \Re^K$) – вектор смещения сигмоидального слоя.

Далее состояние ячейки C_t ($C_t \in \Re^K$) проходит через *tanh*-слой (для получения значений в диапазоне от –1 до 1), и искомый вектор h_t ($h_t \in \Re^M$) вычисляется как:

$$h_t = o_t * tanh(C_t), \quad 10)$$

где o_t ($o_t \in \Re^K$) – выходной вектор сигмоидального слоя ; C_t – новое состояние ячейки.

Вычисленные таким образом векторы h_t и C_t передаются на вход нейронной сети на шаге $(t+1)$.

GRU-сеть

Управляемый рекуррентный блок (GRU – Gated Recurrent Unit) механизм вентилей для рекуррентных нейронных сетей. Схематично архитектура GRU-сети представлена на рисунке 3.

В управляемом рекуррентном блоке используется вентиль обновления и вентиль сброса. Вентиль обновления отбирает информацию из предыдущих шагов, которая должна быть передана дальше, а вентиль сброса решает, какую информацию из предыдущих шагов следует удалить.

Выходной вектор z_t ($z_t \in \Re^K$) вентиля обновления определяет, какая информация должна храниться в новом состоянии ячейки C_t ($C_t \in \Re^K$):

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{11}$$

где h_{t-1} ($h_{t-1} \in \Re^M$) – выходной вектор значений нейронной сети на предыдущем шаге, x_t – входной вектор значений нейронной сети, σ – функция активации, W_z ($W_z \in \Re^{L \times K}$) – матрица весов для входного вектора x_t; U_z ($U_z \in \Re^{K \times K}$) – матрица весов для выходного вектора h_{t-1} нейронной сети на $(t-1)$-м шаге; b_z ($b_z \in \Re^K$) – вектор смещения слоя вентиля обновления.

Аналогично выходной вектор r_t ($r_t \in \Re^K$) вентиля сброса определяет, какую часть информации необходимо удалить:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \tag{12}$$

где h_{t-1} ($h_{t-1} \in \Re^M$) – выходной вектор нейронной сети на $(t-1)$-м шаге, x_t – входной вектор нейронной сети, σ – функция активации, W_r ($W_r \in \Re^{L \times K}$) – матрица весов для входного вектора x_t; U_r ($U_r \in \Re^{K \times K}$) – матрица весов для выходного вектора h_{t-1} нейронной сети на $(t-1)$-м шаге; b_r ($b_r \in \Re^K$) – вектор смещения слоя вентиля сброса.

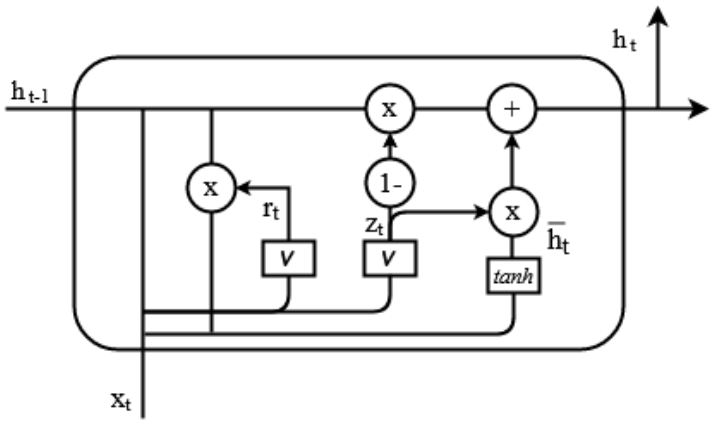

Рисунок 3 – Архитектура GRU-сети в развернутом представлении

Для вычисления выходного вектора h_t ($h_t \in \Re^K$) нейронной сети необходимо найти выходной вектор h'_t ($h'_t \in \Re^K$), позволяющий определить какую информацию на $(t-1)$-м шаге следует отбросить:

$$h'_t = tanh(W_h x_t + r_t \odot U_h h_{t-1} + b_h), \quad (13)$$

где r_t ($r_t \in \Re^K$) – выходной вектор вентиля сброса, h_{t-1} ($h_{t-1} \in \Re^M$) – выходной вектор нейронной сети на $(t-1)$-м шаге, W_h ($W_h \in \Re^{L \times K}$) – матрица весов для входного вектора x_t; U_h ($U_h \in \Re^{K \times K}$) – матрица весов для выходного вектора h_{t-1} нейронной сети на $(t-1)$-м шаге; b_h ($b_h \in \Re^K$) – вектор смещения слоя *tanh*-слоя.

Выходной вектор нейронной сети определяется на основе векторов h_{t-1} ($h_{t-1} \in \Re^M$), h'_t ($h'_t \in \Re^M$) и выходного вектора z_t ($z_t \in \Re^K$) вентиля обновления как:

$$h_t = (1 - z_t) * h_{t-1} + z_t * h'_t, \quad (14)$$

где z_t ($z_t \in \Re^K$) – выходной вектор вентиля обновления, h_{t-1} ($h_{t-1} \in \Re^M$) – выходной вектор нейронной сети на $(t-1)$-м шаге, h'_t ($h'_t \in \Re^M$) – выходной вектор нейронной сети на t-м шаге.

Несмотря на то, что GRU-сеть эффективнее в вычислительном плане из-за меньшего числа вентилей, он все равно стоит на втором месте после LSTM-сеть в плане исполнения. В связи с этим GRU-сеть целесообразно использовать тогда, когда необходимо быстро обучить модель при нехватке вычислительных мощностей.

Формулировка типичных задач

Типичными примерами задач проактивного интеллектуального обслуживания являются следующие задачи, упомянутые ниже. Первые две из этих проблем относятся к сфере технической диагностики, третья проблема относится к сфере медицинской диагностики.

Первая задача сформулирована с целью прогнозирования оставшегося срока полезного использования авиационных двигателей. Созданный шаблон должен использовать значения датчиков самолета, чтобы предсказать, когда двигатель самолета выйдет из строя в будущем, чтобы можно было заранее планировать техническое обслуживание.

Экспериментальный набор данных, который можно использовать для тестирования новых методов обслуживания в этой сфере, содержит идентификатор двигателя для 100 двигателей, номер цикла конкретного двигателя (при этом номер последнего цикла является точкой отказа стабильной работы оборудование); начальный износ и производственные различия (всего 3 параметра), показания 23 датчиков во время работы двигателя в течение рабочего цикла. При этом каждый двигатель может иметь свой последний цикл, описывающий точку отказа стабильной работы оборудования.

Вторая задача сформулирована с целью разработки методов раннего обнаружения сбоев для жестких дисков, которые позволили бы улучшить до-

ступность систем хранения и избежать потери данных. Прогнозирование отказов в таких обстоятельствах позволило бы сократить затраты на простои за счет ожидаемой замены диска. Многие из методов, предложенных до сих пор, в основном выполняют начальное обнаружение отказов, таким образом, не позволяя должным образом планировать задачи обслуживания. Другие работают хорошо только при ограниченном горизонте прогнозирования. Существенный интерес представляет разработка методов оценки оставшегося полезного ресурса для жестких дисков. Эти методы могут быть основаны на параметрах SMART и должны быть способны прогнозировать сбои как в долгосрочных, так и в краткосрочных интервалах.

Экспериментальный набор данных, который можно использовать для тестирования новых методов обслуживания в этой сфере, состоит из данных ежедневного наблюдения за 92348 жесткими дисками в течение нескольких лет. Эти наблюдения содержат информацию о серийном номере, модели, емкости, неисправности и 90 атрибутах SMART каждого устройства. Согласно компании Backblaze, устройство помечается как неисправное, если оно перестает работать (не включается или не получает команды) или если самопроверка SMART не выполняется для атрибутов 5, 187, 188, 197 или 198. Большинство моделей жестких дисков не сообщают обо всех атрибутах SMART. В этом случае не сообщенные значения остаются пустыми. Кроме того, разные производители и модели устройств могут сообщать разные атрибуты. Вся эта информация консолидирована в экспериментальном наборе данных.

Третья задача сформулирована с целью предсказания типов терапевтического воздействия на пациентов. При изучении различных заболеваний гетерогенность среди пациентов обычно приводит к различным стадиям прогрессирования и может потребовать различных видов терапевтического вмешательства. Поэтому важно изучить подтипы пациентов, которые объединяют пациентов в типы, характеризующие подтипы. Выявление подтипа на основе сложных данных пациента является нетривиальной задачей из-за неоднородности информации и временной динамики. Очевидно, что LSTM- и

GRU-нейроны могут успешно применяться для анализа продольных записей пациентов. Однако из ограничений стандартных LSTM- и GRU-нейронов является то, что они не могут работать с нерегулярными временными интервалами. Но нерегулярность времени обычна во многих приложениях здравоохранения. Чтобы проиллюстрировать это, можно рассмотреть записи пациентов, где временной интервал между последовательными посещениями или госпитализацией варьируется от дней до месяцев, а иногда и года. Такие переменные временные промежутки могут указывать на определенные угрожающие условия болезни. Например, частые госпитализации могут указывать на серьезную проблему со здоровьем, а записи о таких посещениях служат источником для изучения состояния. Если между двумя последовательными записями проходят месяцы, зависимость от предыдущих записей в медицинской карте пациента не должна играть активную роль в прогнозировании текущего результата.

Как уже было отмечено, стандартные LSTM- и GRU-нейроны предназначены для обработки данных с постоянными интервалами времени между последовательными элементами последовательности. Учитывая, что промежуток времени между последовательными элементами в записях пациентов может варьироваться от дней до месяцев, использование базовых (традиционных) LSTM- и GRU-нейронов может привести к неоптимальным результатам. В связи с этим был предложен новый LSTM-нейрон под названием Time-Aware LSTM-нейрон (T-LSTM) для обработки нерегулярных временных интервалов в продольных записях пациентов. Авторы статьи рассматривают подпространственное разложение памяти ячейки, которое позволяет затуханию времени дисконтировать содержимое памяти в соответствии с прошедшим временем. При этом модель подтипа пациента использует предложенный T-LSTM-нейрон в автокодере, чтобы реализовать мощное единое представление для последовательной записи пациентов, которые затем используются для объединения пациентов в клинические подтипы. Архитекту-

ра T-LSTM-нейрона фиксирует базовые структуры в последовательностях с временными нерегулярностями.

Экспериментальный набор данных, который можно использовать для тестирования новых методов ведения в этой сфере, описывает так называемую «инициативу маркеров прогрессии Паркинсона» (PPMI), которая представляет собой обсервационное клиническое и продольное исследование, включающее оценки людей с болезнью Паркинсона (БП) с данными для людей с высоким риском и данными для людей, которые здоровы.

PPMI направлена на выявление биомаркеров прогрессирования болезни Паркинсона. Данные PPMI содержат клинические и поведенческие оценки, данные визуализации и биологические образцы, поэтому PPMI представляет собой уникальный архив БП. PPMI представляет собой продольный набор данных с неструктурированным прошедшим временем. Были исследованы данные по 654 пациентов с идиопатической БП или без БП. При этом в наборе данных присутствуют вмененные пропущенные значения и используется форма однозначного признака для категориальных значений и кодировка отклонения данных как 1 и 0. Итоговый набор данных содержит 15 636 записей для 654 пациентов со средней длиной последовательностей, равной 25 (минимальная длина последовательности составляет 3). Также выполнена классификация данных на признаки и цели, где признаки связаны с характеристиками пациента, а цели соответствуют прогрессированию БП.

При этом определено 319 признаков таких, как двигательные симптомы/осложнения, когнитивные функции, вегетативные симптомы, психотические симптомы, проблемы со сном, депрессивные симптомы и шкалы тревожности и депрессии в больнице, 82 цели связаны с моторными признаками, моторными симптомами, познанием и другими немоторными факторами. В этом наборе данных прошедшее время измеряется в месяцах, а разрыв между последовательными записями пациентов составляет от 1 месяца до почти 2 лет.

Можно обучить и оценить регрессионные модели, бинарные или мультиклассовые модели классификации для всех этих задач.

В дальнейшем именно первая задача будет рассмотрена детально.

Пример. Диагностика авиационных двигателей

Для большинства сложных систем, определение класса вероятной ошибки, приводящей к отказу оборудования, само по себе является сложной задачей. Проактивное техническое обслуживание включает в себя множество направлений: прогнозирование отказов, диагностика отказов (анализ первопричин), обнаружение отказов, классификация типов отказов и рекомендации по смягчению последствий и т.д.

Проактивное техническое обслуживание предполагает использование в качестве диагностических признаков таких параметров системы, наблюдение которых даёт возможность контролировать глубинные причины деградации факторов стабильности системы. Данный вид технического обслуживания широко применяется, например, в авиационной отрасли для повышения безопасности полётов воздушных судов. Встроенные бортовые устройства регистрации информации самолётов последнего поколения позволяют получить дополнительные данные о показателях работы бортовых систем. На основе полученных с таких устройств данных можно сформировать «цифровую историю» работы конкретной системы, вплоть до наступления того или иного отказа. Имея подобную информацию можно сформировать соответствующие наборы временных рядов (ВР) и попытаться на их основе классифицировать потенциальный отказ работы системы во времени, а затем спрогнозировать возможное наступление отказа в работе схожих систем в течение циклов эксплуатации.

Для решения подобной задачи, которая может быть отнесена к задачам классификации, целесообразно обратиться к машинному обучению. В настоящее время различные задачи классификации успешно решаются с применением классификаторов на основе SVM-алгоритма, решающих деревьев,

нейронных сетей и т.п. Как показывает анализ литературных источников, в контексте работы с ВР в последние годы активно применяются рекуррентные нейронные сети (RNN, Recurrent Neural Network), при этом различные модификации моделей рекуррентных нейронных сетей, такие как LSTM-сеть (Long Short-Term Memory) и GRU- сеть (Gated Recurrent Unit), позволяют более эффективно работать с памятью сети, и, в частности, решать так называемую проблему исчезающего градиента, что в свою очередь позволяет им более эффективно обучаться и решать задачи классификации, прогнозирования и анализа.

Для того чтобы определить вероятный класс отказа работы авиационного двигателя необходимо разработать модель нейронной сети и обучить ее на наборе данных, содержащем как информацию о бесперебойной работе двигателя, так и информацию о том, что во время эксплуатации произошел сбой.

Для обучения нейронной сети использовался набор данных PM (Predictive Maintenance dataset), представленный в открытом доступе NASA Ames Research Center и содержащий:

– обучающий набор PM_train с данными о работе двигателя до отказа;

– тестовый набор PM_test с данными о работе двигателя без регистрации события отказа;

– GTD-набор (Ground Truth Data) с информацией об оставшихся циклах до отказа для каждого двигателя тестового набора.

Обучающая и тестовая выборки имеют сходную структуру.

Фрагмент тестового набора PM_test представлен в таблице 2.

В таблице 2 представлены:

– id – идентификационный номер двигателя ($id = \overline{1,100}$);

– *cycle* – номер цикла работы конкретного двигателя (при этом номер последнего цикла является точкой отказа стабильной работы оборудования);

– *setting*_1, *setting*_2, *setting*_3 – степени начального износа и производственных различий;

– *s*1 – *s*21 – показания датчиков во время эксплуатации двигателя в течении цикла работы.

Таблица 2 – Фрагмент тестового набора PM_test

id	***cycle***	***setting*_1**	***setting*_2**	***setting*_3**	***s*1**	***s*2**	***s*3**	**…**	***s*19**	***s*20**	***s*21**
1	1	0,0023	0,0003	100	518,67	643,02	1585,29		100	38,86	23,3735
1	2	-0,0027	-0,0003	100	518,67	641,71	1588,45		100	39,02	23,3916
1	3	0,0003	0,0001	100	518,67	642,46	1586,94		100	39,08	23,4166
1	4	0,0042	0	100	518,67	642,44	1584,12		100	39	23,3737
1	5	0,0014	0	100	518,67	642,51	1587,19		100	38,99	23,413
…											
7	22	0,0018	0,0002	100	518,67	642,61	1587,18		100	39,04	23,3756
7	23	-0,0004	0	100	518,67	642,09	1578,49		100	39,19	23,4363
7	24	-0,0021	-0,0003	100	518,67	642,04	1577,27		100	38,99	23,3413
7	25	-0,0029	-0,0001	100	518,67	642,12	1575,94		100	39,07	23,3753
7	26	0,003	-0,0001	100	518,67	642,07	1584,32		100	39,02	23,4244
7	27	-0,0007	-0,0001	100	518,67	641,65	1585,65		100	39,04	23,3983
…											
100	198	0,0013	0,0003	100	518,67	642,95	1601,62		100	38,7	23,1855

Обучающий набор PM_train содержит информацию из нескольких многомерных временных рядов, сформированных по показаниям различных датчиков. Предполагается, что каждый двигатель запускается с различными степенями начального износа и производственных различий, и эта информация пользователю неизвестна. В обучающем наборе PM_train предполагается, что двигатель работает нормально в начале каждого временного ряда. В некоторый момент времени в течении рабочих циклов показания работы двигателя начинают ухудшаться. При достижении установленного порогового значения двигатель считается небезопасным для дальнейшей работы. Иными словами, каждый последний цикл в каждом временном ряду можно рассматривать как точку отказа соответствующего двигателя.

Тестовый набор PM_test также содержит информацию из нескольких многомерных временных рядов, сформированных по показаниям различных датчиков. При этом также предполагаются различные степени износа и отклонения на старте работы двигателя. В начале каждого временного ряда двигатель работает штатно. В некоторый момент времени показатели работы двигателя начинают ухудшаться, но последний рабочий цикл для конкретного двигателя необязательно является точкой отказа (то есть в этом наборе не показано, сколько циклов может работать этот двигатель, прежде чем он выйдет из строя). Нейронная сеть должна научиться определять состояние, когда показания работы двигателя будут считаться небезопасными для дальнейшей эксплуатации.

GTD-данные определяют число оставшихся рабочих циклов до отказа для двигателей в тестовой наборе.

Разработка модели нейронной сети и ее обучение производились на языке Python с использованием библиотеки Keras. Данный язык широко применяется для анализа данных, а библиотека Keras разработана специально для работы с нейронными сетями.

При этом в рамках эксперимента решалась задача бинарной классификации, в которой с учетом имеющейся информации в обучающем наборе данных необходимо дать ответ на вопрос: «Выйдет ли из строя двигатель того или иного типа в течение q-го цикла (q=1, 2, 3,…) ?».

Предварительно на основе обучающего набора данных PM_train была сформирована обучающая выборка Train посредством дополнения обучающего набора данных PM_train значениями по дополнительной характеристике: label, определяющей метку класса («0» – двигатель в норме, «1» – отказ двигателя). Значения дополнительных характеристик в обучающей выборке Train определялись с учетом информации о том, что последний цикл определяет точку отказа двигателя.

Аналогичным образом была создана тестовая выборка Test (таблица 3).

Таблица 3 – Фрагмент тестовой выборки Test

id	***cycle***	***setting_1***	***setting_2***	***setting_3***	***s*1**	***s*2**	***s*3**	**…**	***s*19**	***s*20**	***s*21**	***label***
1	1	0,0023	0,0003	100	518,67	643,02	1585,29		100	38,86	23,3735	0
1	2	-0,0027	-0,0003	100	518,67	641,71	1588,45		100	39,02	23,3916	0
1	3	0,0003	0,0001	100	518,67	642,46	1586,94		100	39,08	23,4166	0
1	4	0,0042	0	100	518,67	642,44	1584,12		100	39	23,3737	0
1	5	0,0014	0	100	518,67	642,51	1587,19		100	38,99	23,413	0
…												
7	22	0,0018	0,0002	100	518,67	642,61	1587,18		100	39,04	23,3756	0
7	23	-0,0004	0	100	518,67	642,09	1578,49		100	39,19	23,4363	0
7	24	-0,0021	-0,0003	100	518,67	642,04	1577,27		100	38,99	23,3413	0
7	25	-0,0029	-0,0001	100	518,67	642,12	1575,94		100	39,07	23,3753	1
7	26	0,003	-0,0001	100	518,67	642,07	1584,32		100	39,02	23,4244	1
7	27	-0,0007	-0,0001	100	518,67	641,65	1585,65		100	39,04	23,3983	1
…												
100	198	0,0013	0,0003	100	518,67	642,95	1601,62		100	38,7	23,1855	1

Далее была выполнена нормировка данных обучающей и тестовой выборок по характеристикам *settings*_1 – *settings*_3 и *s*1 – *s*21 по формуле:

$$\hat{a}_l = \frac{\hat{a}_l - \min\limits_{k=1,K}(a_k)}{\max\limits_{k=1,K}(a_k) - \min\limits_{k=1,K}(a_k)} \qquad (15)$$

где $\hat{a}_l$ – нормированное *l*-е значение характеристики; a_l – исходное *l*-е значение характеристики ($l = \overline{1,K}$); K – число значений по характеристике.

Эксперимент 1.

В ходе исследований был проведен сравнительный анализ двух рекуррентных нейронных сетей: LSTM- и GRU-сетей.

Структура LSTM-сети была определена следующим образом:

– LSTM-слой со 100 нейронами;

– Dropout-слой с параметром 0,2 для решения проблемы переобучения (определяет долю исключаемых нейронов, то есть нейронов, не вносящих свой вклад в процесс обучения);

– Dense-слой с одним нейроном и сигмоидальной функции активации для решения задачи классификации.

При этом были использованы:

– алгоритм *Adam* в качестве алгоритма оптимизации;

– показатель *Accuracy* в качестве целевой функции.

– функция бинарной перекрестной энтропии *binary-crossentropy*, возвращающая ошибку классификации в качестве логистической функции потерь *Loss*:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i) + (1 - \hat{y}_i) \cdot \log(1 - \hat{y}_i) \qquad (16)$$

где y_i – истинная метка класса; $\hat{y}_i$ – ответ классификатора (вычисленная метка класса) на i-м объекте; N – число примеров.

В таблице 4 приведены результаты тестирования 30 моделей LSTM-сети, полученных в течение 100 эпох обучения; с размером окна (то есть периодом времени, используемым для каждого случая обучения), равным 50; скоростью обучения, равной 0,001; общим числом обучающих объектов (batch-size), равным 200; параметром генератора случайных чисел seed, начальное значение которого равно 5 и увеличивается при каждом новом запуске процесса обучения на 5 (изменение параметра необходимо для генерации новых последовательностей обучающих и валидационных подвыборках из сформированной выборки).

Также в таблице 4 указано время, затраченное на обучение в секундах, и результаты обучения нейронной сети с разным значением параметра seed.

Таблица 4 – Результаты тестирования модели LSTM-сети

№	**Accuracy**	**F1-score**	**Network**	**Precision**	**Recall**	**activation**	**seed**	**time, *c***
1	0,989247	0,979592	LSTM	1	0,96	sigmoid	5	1381,91
2	1	1	LSTM	1	1	sigmoid	10	1363,39
3	0,978495	0,961538	LSTM	0,925926	1	sigmoid	15	1357,66
4	0,956989	0,92	LSTM	0,92	0,92	sigmoid	20	1371,83
5	0,978495	0,96	LSTM	0,96	0,96	sigmoid	25	1357,63
	…							
25	0,989247	0,979592	LSTM	1	0,96	sigmoid	130	1360,79
26	0,978495	0,96	LSTM	0,96	0,96	sigmoid	135	1371,94
27	0,989247	0,979592	LSTM	1	0,96	sigmoid	140	1373,20
28	0,978495	0,958333	LSTM	1	0,92	sigmoid	145	1375,78
29	0,978495	0,958333	LSTM	1	0,92	sigmoid	150	1366,73
30	0,978495	0,961538	LSTM	0,925926	1	sigmoid	155	1378,83

Для оценки качества обучения нейронной сети могут быть использованы различные показатели качества, в частности, такие показатели, как точность (*Accuracy*), точность (*Precision*), полнота (*Recall*), *F*-мера.

В случае бинарной классификации на основе матрицы неточностей (ошибок) *Errors* можно составить таблицу размером 2x2 (таблица 5), в которой использованы следующие обозначения: TP – истинно-положительное решение; TN – истинно-отрицательное решение; FP – ложно-положительное решение; FN – ложно-отрицательное решение.

Показатель *Accuracy* определяет долю объектов, по которым классификатор принял правильное решение:

$$Accuracy = \frac{P}{N} = \frac{TP + TN}{FP + FN + TP + TN}, \quad (17)$$

где P – число объектов, по которым нейронная сеть приняла правильное решение; а N – число объектов в обучающей выборке.

Показатель *Precision* в пределах класса определяет долю объектов, верно отнесенных классификатором к классу, к общему числу объектов, отнесенных классификатором к классу в обучающей (тестовой) выборке. Чем выше значение показателя *Precision*, тем меньше ложно-положительное решений.

Показатель *Recall* в пределах класса определяет долю объектов, верно отнесенных классификатором к классу к числу объектов этого класса в обучающей (тестовой) выборке. Чем выше значение показателя *Recall*, тем меньше ложно-отрицательное решений.

В случае бинарной классификации показатель *Precision* и показатель *Recall* определяются как:

$$Precision = \frac{TP}{TP + FP}, \quad (18)$$

$$Recall = \frac{TP}{TP + FN}, \quad (19)$$

В простейшем случае F-мера определяется как гармоническое среднее между показателями *Precision* и *Recall*:

$$F = 2 \cdot \frac{Precision * Recall}{Precision + Recall}. \quad (20)$$

Таблица 5 – Матрица неточностей (ошибок)

Классификация		Метки классов в наборе данных	
		Метка положительного класса	Метка отрицательного класса
Метки классов, выставленные нейронной сетью	Метка положительного класса	TP	FP
	Метка отрицательного класса	FN	TN

В таблице 6 представлены усредненные значения показателей результатов обучения модели, а также значения математического ожидания и дисперсии по показателю *Accuracy*.

На рисунках 4 и 5 приведены графические представления результатов обучения одной из моделей LSTM-сети на 100 эпохах по показателю *Accuracy* и по логистической функции потерь *Loss* соответственно.

Таблица 6 – Усредненные значения показателей тестирования модели LSTM-сети

Accuracy	F1-score	Precision	Recall	Mean time, c	Mean Accuracy	Dispersion Accuracy
0,97849	0,96154	0,92593	1	1357,66	0,97983	0,0000087

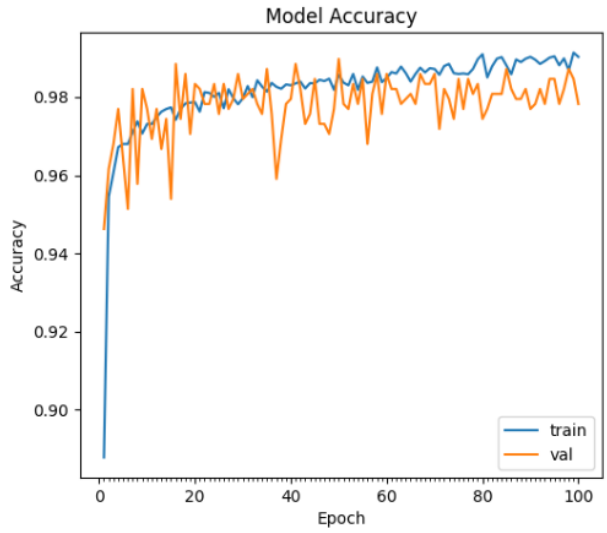

Рисунок 4 – Результаты обучения модели LSTM-сети по показателю *Accuracy*

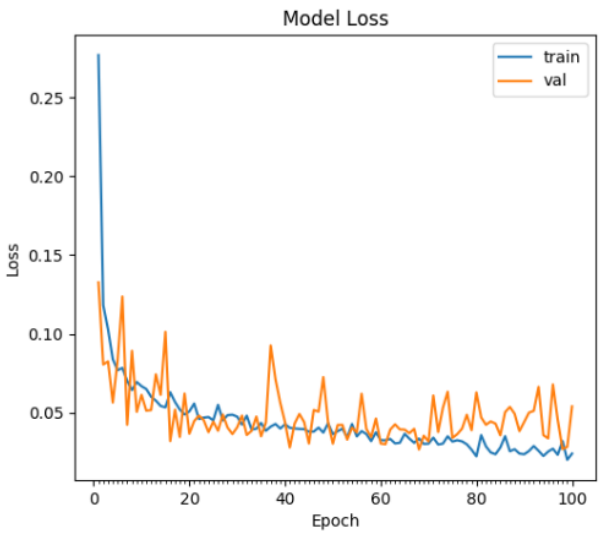

Рисунок 5 – Результаты обучения модели LSTM-сети по логистической функции потерь *Loss*

Таблица 7 – Результаты тестирования модели GRU-сети

№	**Accuracy**	**F1-score**	**Network**	**Precision**	**Recall**	**activation**	**seed**	**time, c**
1	1	1	GRU	1	1	sigmoid	5	1001,41
2	0,989247	0,979592	GRU	1	0,96	sigmoid	10	1003,674628
3	0,956989	0,923077	GRU	0,888889	0,96	sigmoid	15	1003,653628
4	0,989247	0,979592	GRU	1	0,96	sigmoid	20	1003,5053
5	0,967742	0,941176	GRU	0,923077	0,96	sigmoid	25	1005,891989
	…							
25	0,989247	0,980392	GRU	0,961538	1	sigmoid	130	1007,414239
26	0,967742	0,93617	GRU	1	0,88	sigmoid	135	1005,549091
27	0,989247	0,979592	GRU	1	0,96	sigmoid	140	1007,583889
28	0,967742	0,941176	GRU	0,923077	0,96	sigmoid	145	1010,307497
29	0,967742	0,941176	GRU	0,923077	0,96	sigmoid	150	1009,367884
30	0,978495	0,958333	GRU	1	0,92	sigmoid	155	1009,297234

Таблица 8 – Усредненные значения показателей тестирования модели GRU-сети

Accuracy	F1-score	Precision	Recall	Mean time, c	Mean Accuracy	Dispersion Accuracy
0,97849	0,96154	0,92593	1	1011,84	0,98279	0,00011

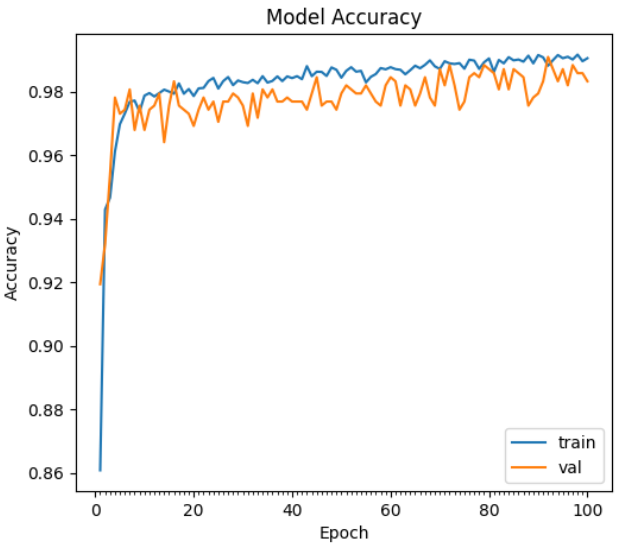

Рисунок 4 – Результаты обучения модели GRU-сети по показателю *Accuracy*

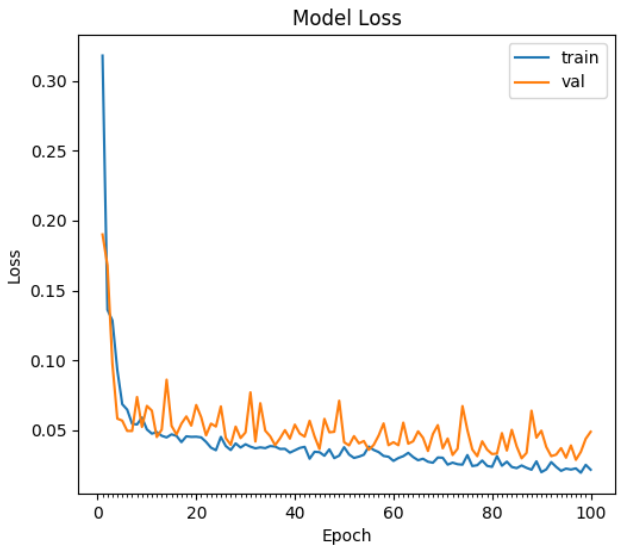

Рисунок 5 – Результаты обучения модели GRU-сети по логистической функции потерь *Loss*

При этом на рисунке 4 графические зависимости для показателя *Accuracy* на обучающих и валидационных подвыборках (полученных на основе обучающей выборки PM_train случайным разбиением в пропорции 80:20) обозначены как train и val, а на рисунке 5 аналогичным образом обозначены графические зависимости для логистической функции потерь *Loss*.

Структура GRU-сети была разработана аналогично структуре LSTM-сети, только вместо LSTM-слоя был использован GRU-слой.

В таблице 7 приведены результаты обучения 30 моделей GRU-сети в течение 100 эпох с размером окна, равным 50; скоростью обучения, равной 0,001; общим числом обучающих объектов (batch-size), равным 200; параметром генератора случайных чисел seed, начальное значение которого равно 5 и увеличивается при каждом новом запуске процесса обучения на 5.

Также в таблице 7 указано время, затраченное на обучение в секундах, и результаты обучения нейронной сети с разным значением параметра seed.

В таблице 8 представлены усредненные значения показателей результатов обучения модели, а также значения математического ожидания и дисперсии по показателю *Accuracy*. На рисунках 6 и 7 приведены графические

представления результатов обучения одной из моделей GRU-сети по 100 эпохам по показателю *Accuracy* и по логистической функции потерь *Loss* соответственно.

При этом на рисунке 6 графические зависимости для показателя *Accuracy* на обучающих и валидационных подвыборках (полученных на основе обучающей выборки PM_train случайным разбиением в пропорции 80:20) обозначены как train и val, а на рисунке 7 аналогичным образом обозначены графические зависимости для логистической функции потерь *Loss.*

Сравнительный анализ результатов обучения нейронных сетей показывает, что, несмотря на равные значения показателя *Accuracy* (таблицы 6 и 8), скорость обучения GRU-сети превосходит скорость обучения LSTM-сети. Анализ графических зависимостей показывает, что качество обучения LSTM-сети нестабильно и подвергнуто, начиная примерно с 60-й эпохи, переобучению, в то время как GRU-сеть не имеет проблемы переобучения на 100 эпохах, при этом качество обучения по показателю *Accuracy* постепенно возрастает.

Как видно из результатов экспериментальных исследований, LSTM- и GRU-сети демонстрируют свою высокую эффективность в решении задачи классификации вероятного класса ошибки в работе оборудования в сложных технических системах. В дальнейшем планируется исследовать возможность использования таких сетей для решения задач прогнозирования отказов в работе оборудования в сложных технических системах. При этом для повышения качества классификационных и прогностических решений будут рассмотрены вопросы определения архитектуры нейронной сети с выбором числа и типа слоёв, а также вопросы выбора значений гиперпараметров (например, момента и скорости обучения, числа нейронов в скрытых слоях, параметров регуляризации нейронной сети и т.п.), размера окна (длины ВР, подаваемого на вход нейронной сети).

Эксперимент 2.

В ходе экспериментов был проведен сравнительный анализ различных рекуррентных конфигураций нейронных сетей. Были рассмотрены одноуров-

невые и двухуровневые конфигурации (таблица 9). Рисунки 6 – 15 показывают графические зависимости для показателя *Accuracy* модели и *функции* Loss модели для этих конфигураций.

Структура любой нейронной сети была определена следующим образом:

– рекуррентный слой со 100 нейронами;

– Dropout-слой с параметром 0,2 для решения проблемы переобучения (определяет долю исключаемых нейронов, то есть нейронов, не вносящих свой вклад в процесс обучения);

– Dense-слой с одним нейроном и сигмоидальной функции активации для решения задачи классификации.

При необходимости добавляется еще один рекуррентный слой и связанный с ним Dropout-слой.

Рекуррентный слой может быть определен с использованием RNN-, LSTM- или GRU-нейронов.

Только одна модель, двухслойная модель LSTM + LSTM (100 + 50 нейронов), имеет 50 нейронов на втором слое. На рисунках 15 и 16 показана структура этой модели и форма выходной формы с количеством параметров на каждом слое соответственно.

При этом были использованы:

– алгоритм *Adam* в качестве алгоритма оптимизации;

– показатель *Accuracy* в качестве целевой функции;

– функция бинарной перекрестной энтропии *binary-crossentropy*, возвращающая ошибку классификации в качестве логистической функции потерь *Loss*.

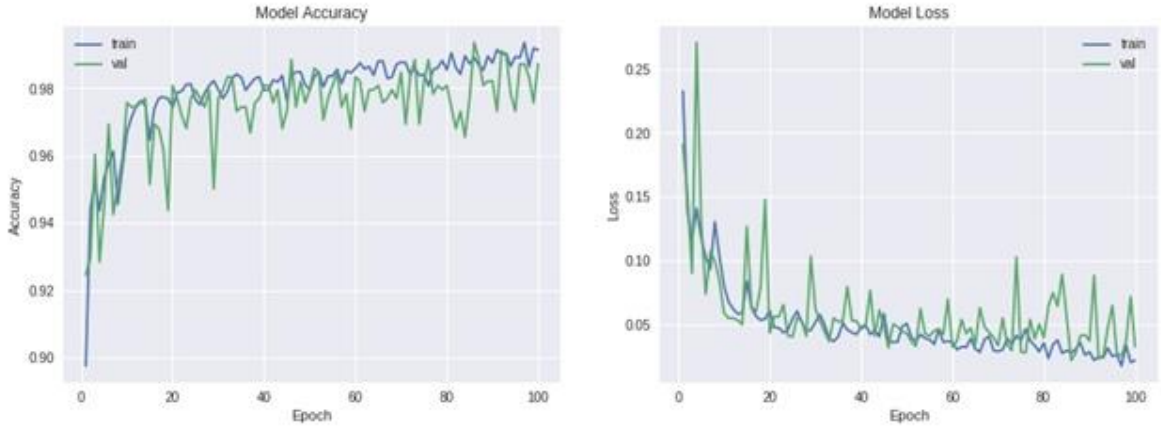

Рисунок 6 – Графические зависимости для однослойной RNN сети

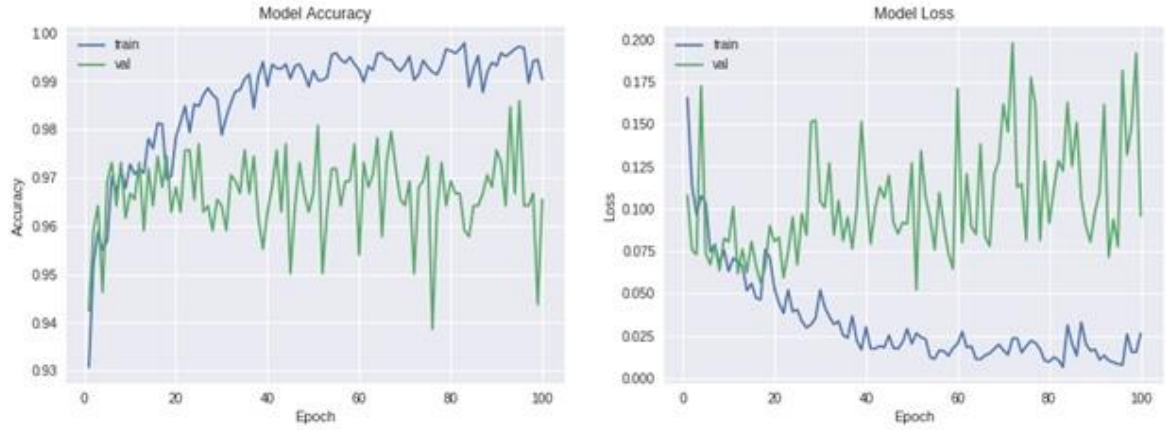

Рисунок 7 – Графические зависимости для двухслойной RNN сети

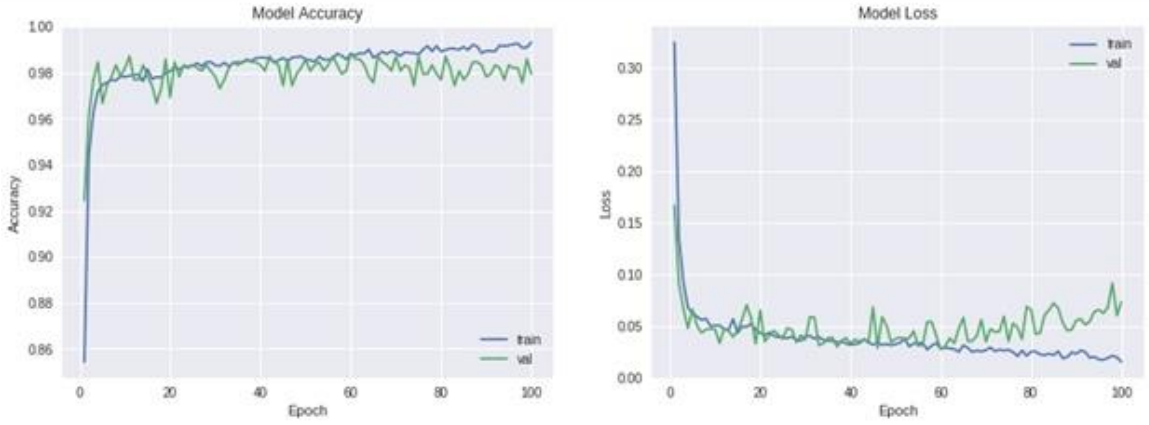

Рисунок 8 – Графические зависимости для однослойной GRU сети

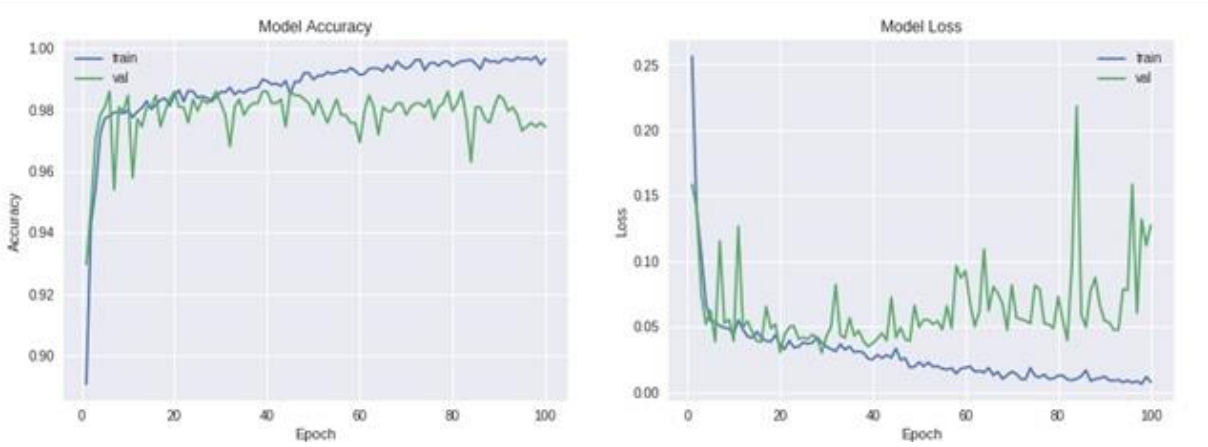

Рисунок 9 – Графические зависимости для двухслойной GRU сети

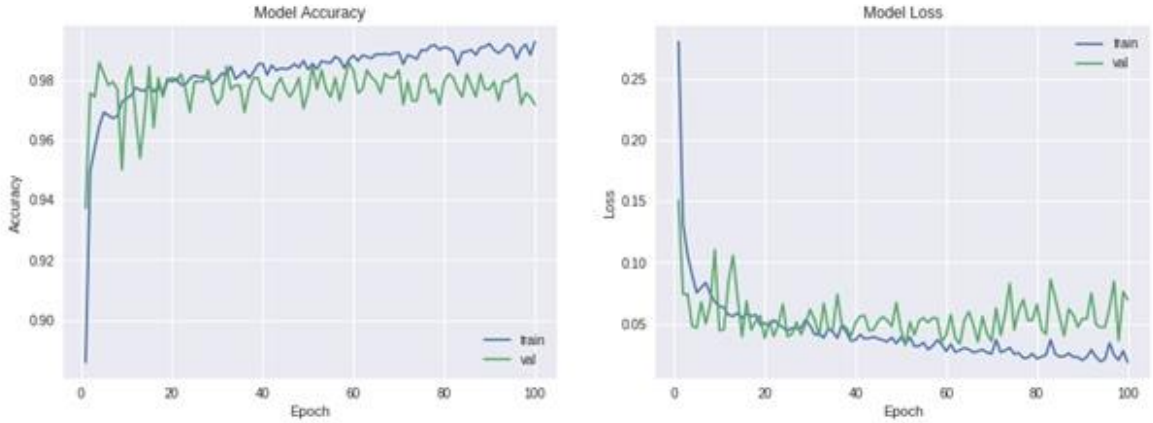

Рисунок 10 – Графические зависимости для однослойной LSTM сети

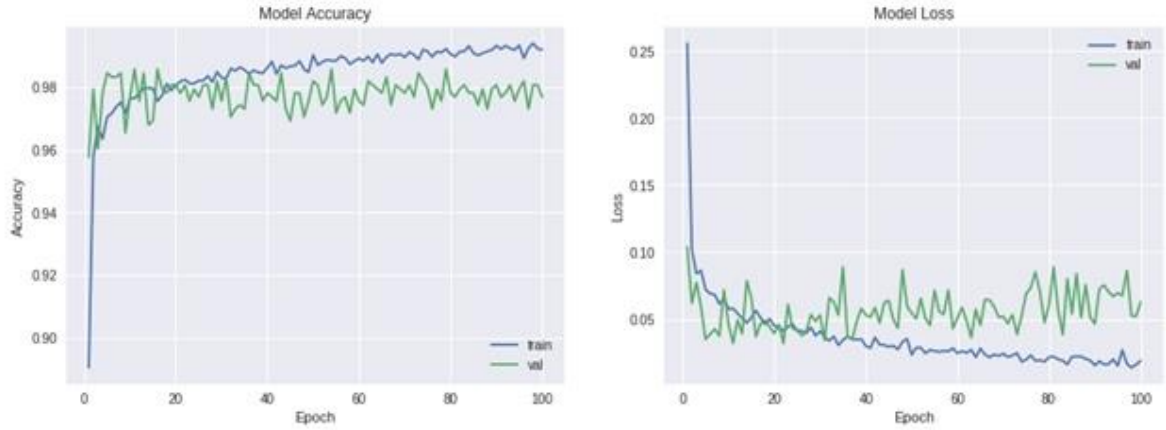

Рисунок 11 – Графические зависимости для двухслойной LSTM сети (100+50)

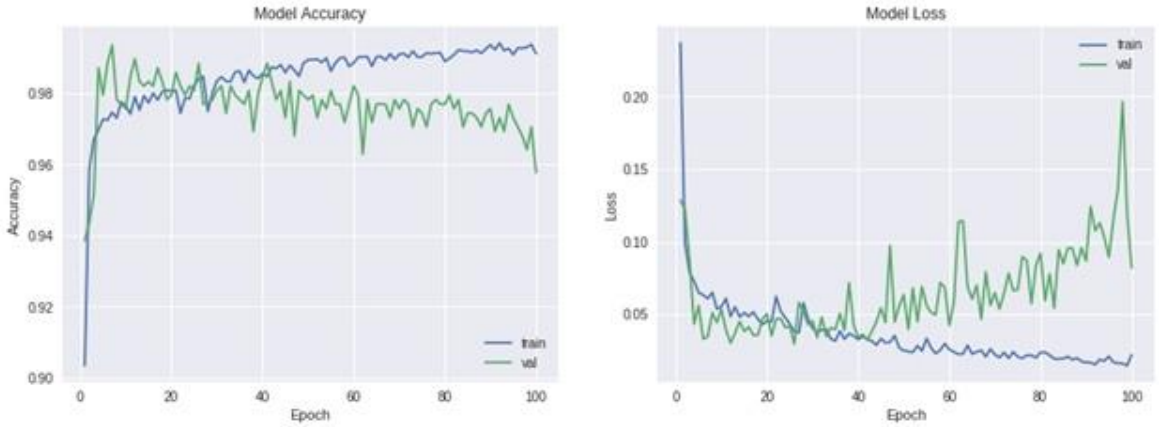

Рисунок 12 – Графические зависимости для двухслойной LSTM сети (100+100 нейронов)

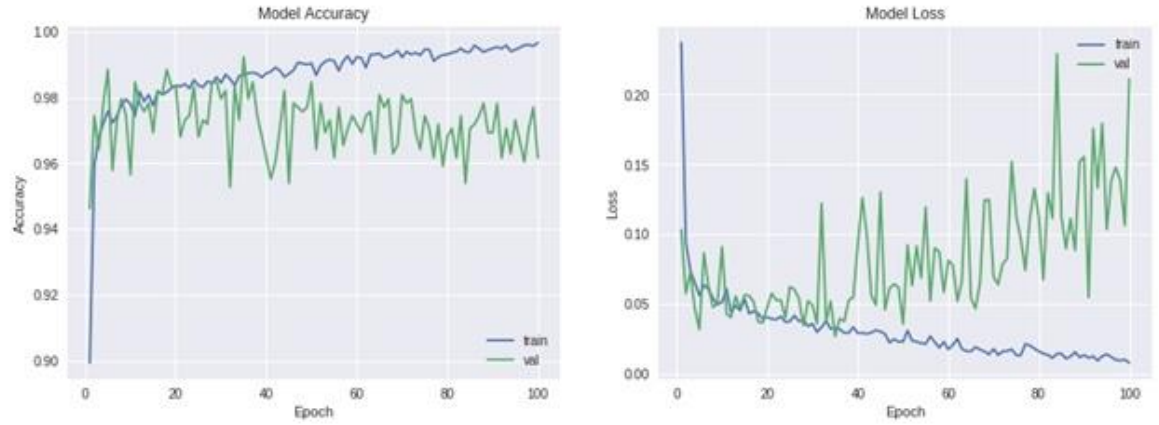

Рисунок 13 – Графические зависимости для двухслойной GRU+LSTM сети

Было реализовано 100 эпох, чтобы наблюдать за процессом разработки модели.

В таблице 6 приведены структуры модели, временные затраты и значения показателя точности на обучающем и тестовом (проверяющем) наборе.

Известно, что результаты обучения нейронной сети существенно зависят от правильной инициализации весов нейронов. При этом требуется, чтобы найденное решение всегда было близко к субоптимальному.

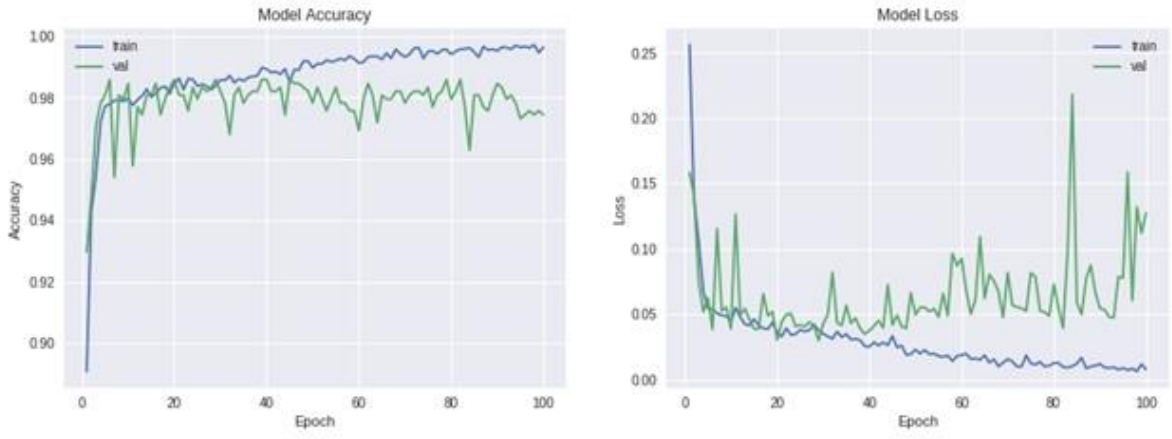

Рисунок 14 – Графические зависимости для однослойной LSTM+GRU сети

Минимальное время разработки, как и ожидалось, соответствует одноуровневой и двухслойной моделям RNN сети (рисунки 6 и 7). При этом очень сложно отследить для однослойной RNN сети (рисунок 6), когда происходит переобучение, и сказать, когда целесообразно выполнить раннюю остановку процесса обучения. В этом контексте однослойные модели GRU (рисунок 8) и LSTM (рисунок 10) сетей ведут себя заметно лучше. Двухуровневая модель RNN сети (рисунок 7) также ведет себя лучше в этом контексте, чем модель одноуровневой RNN сети (рисунок 6), но она имеет одно из худших значений показателя *Accuracy* на тестовом наборе.

Двухслойная модель GRU сети (рисунок 9) позволяет сделать остановку раньше, чем однослойная модель GRU сети (рисунок 8), вдвое сократив число эпох (примерно 30 эпох против 60).

Для моделей LSTM сети требуется меньше эпох для ранней остановки: около 40 эпох для однослойной модели LSTM сети (рисунок 10) и около 30 эпох для двухслойной модели LSTM сети (рисунки 11 и 12). При этом временные затраты увеличиваются незначительно. Выбор оптимального числа нейронов во втором слое (рисунки 11 и 12) требует дополнительного внимания: важно минимизировать число нейронов (и, следовательно, затраты времени на разработку модели) без ущерба для показателя *Accuracy* модели.

Модели гибридной сети (рисунки 13 и 14) работают аналогичным образом в контексте точности, но, на первый взгляд, двухслойная модель LSTM + GRU сети (рисунок 14) допускает более раннюю остановку, однако необходимы дополнительные исследования в это направлении.

Таблица 6 – Модели и их характеристики

Структура модели	Время	Accuracy (Train)	Accuracy (Train)
one-layer RNN (100 neurons)	6 min 36 s	0.995650	0.989247
two-layer RNN+RNN (100+100 neurons)	11 min 10 s	0.995584	0.9677419
one-layer GRU(100 neurons)	12 min 43 s	0.994242	0.978495
two-layer GRU+GRU (100+100 neurons)	26 min 15 s	0.996865	0.978495
one-layer LSTM (100 neurons)	19 min 46 s	0.985158	0.978495
two-layer LSTM+LSTM (100+50 neurons)	30 min 30 s	0.993922	0.989247
two-layer LSTM+LSTM (100+100 neurons)	30 min 45 s	0.979720	0.967742
two-layer LSTM+GRU (100+100 neurons)	28 min 56 s	0.9968652	0.978495
two-layer GRU+LSTM (100+100 neurons)	28 min 43 s	0.995970	0.978495

Сравнительный анализ результатов обучения моделей нейронных сетей показывает равные или близкие значения показателя точности для моделей GRU- и LSTM-сетей. При этом скорость обучения модели, основанной на GRU-нейронах, превышает скорость обучения модели, основанной на LSTM-нейронах (как для однослойной сетевой модели, так и для двухслойной модели).

Более того, эти нейронные сети решают проблему переобучения и ранней остановки во время обучения нейронной сети.

```
# build the network
nb_features = seq_array.shape[2]
nb_out = label_array.shape[1]

model = Sequential()

model.add(LSTM(
         input_shape=(sequence_length, nb_features),
         units=100,
         return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(
          units=50,
          return_sequences=False))
model.add(Dropout(0.2))

model.add(Dense(units=nb_out, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Рисунок 15 – Форма выхода с числом параметров на каждом слое двухслойной модели LSTM + LSTM (100 + 50 нейронов)

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 50, 100)           50400
_________________________________________________________________
dropout_1 (Dropout)          (None, 50, 100)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                30200
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 80,651
Trainable params: 80,651
Non-trainable params: 0
_________________________________________________________________
None
```

Рисунок 16 – Структура двухслойной модели LSTM + LSTM (100 + 50 нейронов)

Ссылки

1. Saxena A and Goebel K NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/tech/dash/pcoe/prognostic-data-repository/), NASA Ames Research Center, Moffett Field, CA.
2. https://github.com/Azure/lstms_for_predictive_maintenance
3. https://gallery.azure.ai/Collection/Predictive-Maintenance-Template-3
4. https://www.backblaze.com/b2/hard-drive-test-data.html