| Министерство науки и высшего образования Российской Федерации | | |
| --- | --- | --- |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования | | |
| **ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ** | | |
|  | Институт информационных технологий и анализа данных |  | |
| наименование института | | |
|  | | |

| Допускаю к защите |  |  |
| --- | --- | --- |
| Руководитель |  | Ж.С. Афанасьева |
|  |  | И.О. Фамилия |

| Прогнозирование землетрясений с помощью рекуррентных нейронных |
| --- |
| сетей |
| наименование темы |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА |
| к курсовой работе по дисциплине |
| Теоретические основы искусственного интеллекта |
| 1.003.00.00 – ПЗ  обозначение документа |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | ИИТм-23-1 |  |  |  | Д.Е. Вовиков |
|  | шифр группы |  | подпись |  | И.О. Фамилия |
| Нормоконтроль |  |  |  |  | Ж.С. Афанасьева |
|  |  |  | подпись |  | И.О. Фамилия |

|  |  |  |
| --- | --- | --- |
| Курсовая работа защищена с оценкой |  |  |

Иркутск 2024 г.

|  |
| --- |
| Министерство науки и высшего образования Российской Федерации |
| Федеральное государственное бюджетное образовательное учреждение высшего образования |
| **ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ** |

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

По курсу Теоретические основы искусственного интеллекта

Студенту Вовикову Даниилу Евгеньевичу

Тема работы Прогнозирование землетрясений с помощью рекуррентных нейронных сетей

Цель работы: Сравнение архитектур рекуррентных нейронных сетей и оценка возможности их применения для прогнозирования землетрясений

Исходные данные: Набор данных с радоновых датчиков

Курсовая работа выполнена в соответствии с требованиями СТО 005-2020 «Система менеджмента качества. Учебно-методическая деятельность. Оформление курсовых проектов (работ) и выпускных квалификационных работ технических направлений подготовки и специальностей».

Рекомендуемая литература:

1. Шолле Ф. Глубокое обучение на Python. СПб: Питер, 2018. 400 c.
2. Гафаров Ф.М. Искусственные нейронные сети и приложения: учеб. пособие/ Ф.М. Гафаров, А.Ф. Галимянов. Казань: Изд-во Казан. ун-та, 2018.121 с.
3. Ростовцев В.С. Искусственные нейронные сети: учебник. Санкт-Петербург: Лань, 2019. 213 с.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | | «13» февраля 2024 г. | | | | |
| Задание получил |  | | |  | Д.Е. Вовиков | | | | |
|  | подпись | | |  | И.О Фамилия. | | | | |
| Дата представления работы руководителю | | | | | | «    » мая 2024 г. | | | | |
| Руководитель курсовой работы | | |  | | | | |  | Ж.С. Афанасьева | |
|  | | | подпись | | | | |  | И.О Фамилия. | |

**Содержание**

[Введение 4](#_Toc167818342)

[1 Основные принципы построения рекуррентных нейронных сетей 5](#_Toc167818343)

[1.1 Принцип построения модели RNN 5](#_Toc167818344)

[1.2 Принцип построения модели LSTM 6](#_Toc167818345)

[1.3 Принцип построения модели GRU 7](#_Toc167818346)

[1.4 Принцип построения модели Bidirectional RNN 7](#_Toc167818347)

[1.5 Принцип построения модели Deep RNN 8](#_Toc167818348)

[Выводы по первой главе 8](#_Toc167818349)

[2 Проверка применимости рекуррентных нейронных сетей для прогнозирования землетрясений 10](#_Toc167818350)

[2.1 Подготовка данных 10](#_Toc167818351)

[2.2 Построение моделей нейронных сетей 12](#_Toc167818352)

[2.2.1 Построение модели RNN 12](#_Toc167818353)

[2.2.2 Построение модели LSTM 14](#_Toc167818354)

[2.2.3 Построение модели GRU 17](#_Toc167818355)

[2.2.4 Построение модели Bidirectional RNN 19](#_Toc167818356)

[2.2.5 Построение модели Deep RNN 21](#_Toc167818357)

[2.3 Тестирование моделей 23](#_Toc167818358)

[Выводы по второй главе 26](#_Toc167818359)

[Заключение 27](#_Toc167818360)

[Список использованных источников 28](#_Toc167818361)

Введение

Применение нейронных сетей для прогнозирования землетрясений является активно развивающимся направлением в области геофизики и искусственного интеллекта [1]. Нейронные сети способны обрабатывать и анализировать большие объемы данных, выявлять сложные закономерности и делать прогнозы. Это позволяет ученым и исследователям использовать эти модели для анализа сейсмической активности и предсказания вероятности и мощности будущих землетрясений.

Традиционные методы прогнозирования, основанные на статистическом анализе и физических моделях, не всегда обеспечивают достаточную точность и своевременность прогнозов. В связи с этим, активно развивается направление применения методов искусственного интеллекта, в частности, рекуррентных нейронных сетей (РНС), для прогнозирования землетрясений. Эти методы предлагают новые подходы к обработке и анализу данных, что может привести к улучшению точности прогнозов [2].

Результаты этого исследования могут быть полезны для создания более точных и оперативных систем прогнозирования землетрясений, что в свою очередь способствует повышению безопасности и снижению рисков для населения в зонах сейсмической активности. Это особенно важно в свете того, что землетрясения могут привести к значительным материальным убыткам и потере жизней.

При этом, применение современных методов искусственного интеллекта может значительно улучшить качество прогнозов и сделать их более доступными для широкого использования. В конечном итоге, это может привести к созданию более эффективных систем предупреждения о землетрясениях, что поможет снизить риски и улучшить безопасность населения.

**Целью курсовой работы** является сравнение архитектур рекуррентных нейронных сетей и оценка возможности их применения для прогнозирования землетрясений

**Задачи для достижения поставленной цели**:

1. изучить принципы построения рекуррентных нейронных сетей (РНС);
2. сравнить различные архитектуры РНС;
3. применить наиболее подходящие архитектуры для прогнозирования землетрясений.

1 Основные принципы построения рекуррентных нейронных сетей

1.1 Принцип построения модели RNN

Модель рекуррентной нейронной сети (RNN) строится на основе нескольких ключевых принципов, которые позволяют эффективно обрабатывать последовательные данные.

RNN обрабатывает входные данные последовательно, учитывая контекст предыдущих элементов последовательности. Это позволяет модели улавливать зависимости в данных и работать с различными типами последовательных входов, такими как текст, звук, временные ряды и другие.

Одной из ключевых особенностей RNN является использование внутреннего состояния для передачи информации между шагами обработки. Это позволяет модели запоминать предыдущие состояния и учитывать их при обработке новых входных данных.

Веса и смещения нейронной сети в RNN используются на всех шагах обработки последовательности. Это обеспечивает общую способность модели к улавливанию шаблонов и зависимостей в данных на протяжении всей последовательности.

Для обучения RNN применяется алгоритм обратного распространения ошибки во времени (BPTT), который учитывает временную структуру данных. Это позволяет модели корректировать веса и улучшать свою способность к предсказанию и генерации последовательностей [3].

Классическая модель рекуррентной нейронной сети (RNN) основана на принципе использования внутренней памяти для обработки последовательностей переменной длины. Основное отличие RNN от традиционных нейронных сетей заключается в том, что они могут учитывать предыдущее состояние системы при обработке текущего входа. Это делает их идеальными для задач, связанных с временными рядами или естественным языком.

В RNN каждый нейрон или ячейка принимает входные данные не только от предыдущего слоя, но и от собственного предыдущего состояния. Это позволяет сети сохранять информацию о предыдущих данных в последовательности и использовать её для формирования текущего вывода (см. рисунок 1.1).

В момент времени *t* узлы с рекуррентными ребрами получают входные данные от текущей входного , а также от значений скрытых слоев в предыдущем состоянии сети. Выходной слой в каждый момент времени *t* вычисляется с учетом значений скрытых слоев в момент времени *t*. Входной слой в момент времени *t - 1* может влиять на выходной слой в момент времени *t* посредством рекуррентных связей.

Два уравнения определяют все необходимые вычисления для расчета на каждом временном шаге в простой рекуррентной нейронной сети (1, 2):

|  |  |
| --- | --- |
|  | **(1)** |
|  | **(2)** |

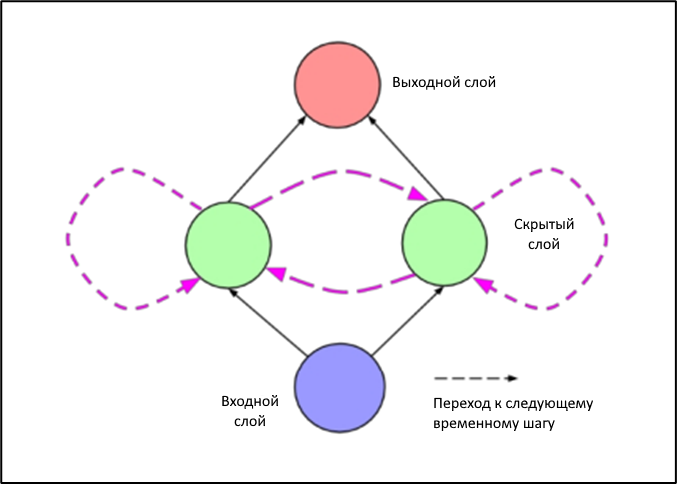


Рисунок 1.1 – Простая рекуррентная сеть

Здесь – матрица обычных весов между входом и скрытым слоем, а – матрица рекуррентных весов между скрытым слоем и самим собой на соседних временных шагах. Векторы являются параметрами смещения [4].

1.2 Принцип построения модели LSTM

Модель LSTM (Long Short-Term Memory) — это разновидность рекуррентной нейронной сети (РНС), которая эффективно обрабатывает последовательные данные и запоминает долгосрочные зависимости.

LSTM состоит из ячеек памяти, которые хранят информацию о предыдущих состояниях. Каждая ячейка имеет состояние ячейки (cell state) и скрытое состояние (hidden state), которые передаются между временными шагами.

LSTM использует три типа вентилей для управления информацией в ячейках памяти:

1. Вентиль забывания (forget gate) определяет, какую информацию из предыдущего состояния ячейки следует сохранить.
2. Вентиль входа (input gate) решает, какую новую информацию из текущего входа и предыдущего скрытого состояния следует добавить в состояние ячейки.
3. Вентиль выхода (output gate) определяет, какую часть состояния ячейки следует использовать для генерации выхода на текущем шаге.

На каждом временном шаге LSTM обновляет состояние ячейки, используя вентили для управления информацией. Это позволяет модели эффективно запоминать долгосрочные зависимости.

LSTM обучается с помощью алгоритма обратного распространения ошибки во времени (BPTT), который вычисляет градиенты ошибки по отношению к параметрам модели на каждом временном шаге. Это позволяет модели адаптировать свои параметры для улучшения производительности на последовательных задачах [5].

1.3 Принцип построения модели GRU

Принцип построения модели GRU (Gated Recurrent Unit) основан на использовании специальных ячеек памяти, которые позволяют эффективно обрабатывать последовательные данные, такие как текст или речь.

Ключевыми особенностями GRU являются:

1. Обновляющие ворота (update gate) – определяют, какая информация из предыдущего состояния ячейки должна быть сохранена в текущем состоянии. Это позволяет модели решать, какую информацию следует запомнить или забыть.
2. Сбрасывающие ворота (reset gate) – контролируют, какая информация из предыдущего состояния должна быть использована для вычисления текущего состояния. Это помогает модели игнорировать несущественные предыдущие состояния.
3. Состояние ячейки (cell state) – представляет собой информацию, которая передается из одного временного шага в другой. Состояние ячейки обновляется с помощью обновляющих и сбрасывающих ворот.
4. Выходное состояние (output state) – вычисляется на основе текущего состояния ячейки и входных данных. Выходное состояние используется для генерации выходных данных или для передачи в следующую ячейку [6].

GRU использует обновляющие и сбрасывающие ворота для контроля потока информации в ячейке памяти, что позволяет эффективно обрабатывать последовательные данные и решать задачи, связанные с обработкой естественного языка, распознаванием речи и другими приложениями, требующими обработки последовательностей.

Отличие модели GRU (Gated Recurrent Unit) от LSTM (Long Short-Term Memory) заключается в структуре и функциональности их ячеек памяти.

В GRU есть два типа ворот – обновляющий и сбрасывающий, которые помогают модели принимать решения о том, какая информация должна быть сохранена или забыта.

GRU имеет меньше параметров, чем LSTM, что делает ее более простой в обучении и менее затратной по памяти.

GRU обычно лучше справляется с проблемой затухания градиента, так как она имеет механизм обновления состояния ячейки [7].

1.4 Принцип построения модели Bidirectional RNN

Принцип построения модели Bidirectional RNN основан на использовании рекуррентных нейронных сетей (RNN), которые способны учитывать контекст информации как в прошлом, так и в будущем (см. рисунок 1.2). В случае Bidirectional RNN (BRNN), информация обрабатывается как в прямом, так и в обратном направлении, что позволяет модели учитывать как предшествующий контекст, так и последующий контекст для более глубокого понимания последовательности данных [4].

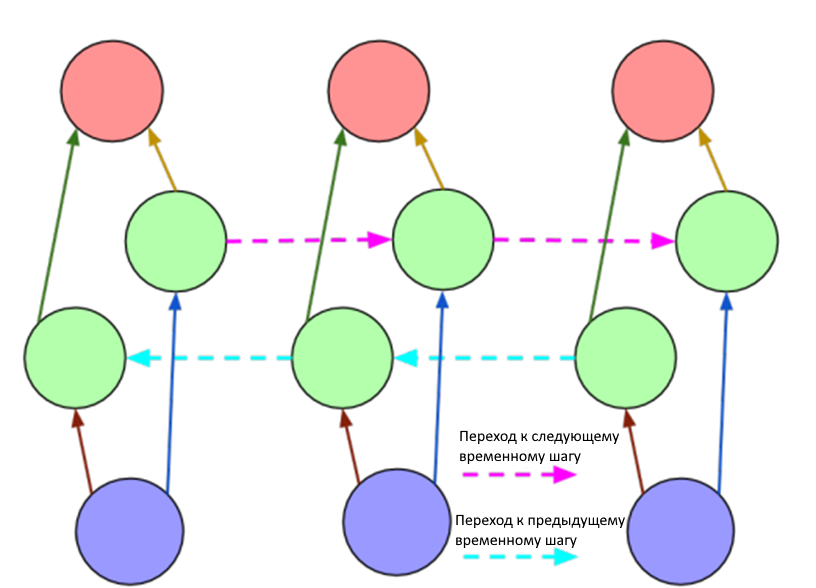


Рисунок 1.2 – Bidirectional RNN

BRNN состоит из двух слоев: прямого и обратного. Прямой слой обрабатывает последовательность данных в привычном порядке, а обратный слой работает в обратном порядке. Это позволяет модели улавливать зависимости как от предыдущих, так и от последующих элементов последовательности, что полезно для задач, где контекст имеет важное значение [8].

1.5 Принцип построения модели Deep RNN

Deep RNN сочетает в себе преимущества глубоких нейронных сетей и рекуррентных нейронных сетей, что делает ее мощным инструментом для решения задач, связанных с обработкой последовательных данных, таких как машинный перевод, распознавание речи, анализ текстов и других задачах.

Deep RNN состоит из нескольких слоев рекуррентных нейронных сетей, что позволяет извлекать более сложные и абстрактные признаки из входных данных.

Для преодоления проблемы исчезающего или взрывного градиента в стандартных RNN, в Deep RNN применяются специальные ячейки, такие как LSTM (Long Short-Term Memory) или GRU (Gated Recurrent Unit). Эти ячейки позволяют эффективно запоминать и использовать долгосрочные зависимости в последовательных данных.

Как и другие глубокие нейронные сети, Deep RNN требует больших объемов размеченных данных для эффективного обучения. Чем больше данных, тем лучше модель способна обобщать и решать сложные задачи [9].

Выводы по первой главе

Модели рекуррентных нейронных сетей (RNN), такие как LSTM и GRU, представляют собой инструменты для обработки последовательных данных и учета долгосрочных зависимостей в информации. GRU отличается от LSTM своей структурой, включая два типа ворот – обновляющий и сбрасывающий, что делает ее более простой в обучении и менее затратной по памяти.

Модель Bidirectional RNN (BRNN) позволяет учитывать как предшествующий, так и последующий контекст для более глубокого понимания последовательности данных.

Deep RNN объединяет преимущества глубоких нейронных сетей и RNN, позволяя извлекать сложные признаки из входных данных и эффективно учитывать долгосрочные зависимости.

Для обучения RNN, включая LSTM и GRU, применяется алгоритм обратного распространения ошибки во времени (BPTT), который помогает модели адаптировать свои параметры для улучшения производительности на последовательных задачах.

2 Проверка применимости рекуррентных нейронных сетей для прогнозирования землетрясений

2.1 Подготовка данных

В качестве набора для обучения и проверок моделей использовались данные о радоновых выбросах, полученные от датчиков, расположенных неподалеку от деревни Тырган Иркутской области (см. рисунок 2.1).

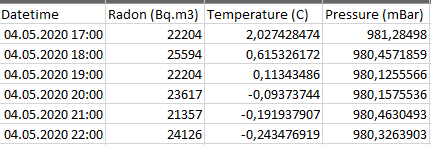


Рисунок 2.1 – Фрагмент набора данных

Данный набор включает в себя следующие столбцы:

1. Datetime – столбец, содержащий временные метки, указывающие, когда были сделаны измерения.
2. Radon (Bq.m3) – столбец измерения концентрации почвенного радона, выраженной в беккерелях на кубический метр.
3. Temperature – столбец, содержащий измерения температуры воздуха в градусах Цельсия во время каждого измерения радона.
4. Pressure (mBar) – столбец с измерениями атмосферного давления в миллибарах во время каждого измерения радона.

Данные из CSV-файла были загружены в DataFrame data. Полученный DataFrame был проверен на наличие пустых значений (см. рисунок 2.2) в данных и тип столбцов (см. рисунок 2.3).

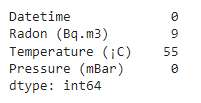


Рисунок 2.2 – Результат проверки DataFrame на пустоты

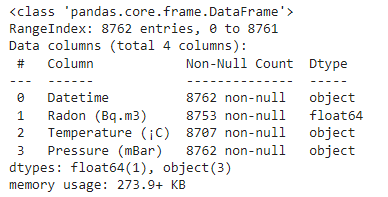


Рисунок 2.3 – Результат проверки DataFrame на тип данных

Столбцы «Temperature (°C)» и «Pressure (mBar)» были преобразованы из строкового формата в числовой путем замены запятых на точки и преобразования результатов в тип float.

data['Temperature (¡C)'] = data['Temperature (¡C)'].str.replace(',', '.').astype(float)

data['Pressure (mBar)'] = data['Pressure (mBar)'].str.replace(',', '.').astype(float)

Пропущенные значения в числовых столбцах были заменены средними значениями по соответствующим столбцам для сохранения целостности данных и избегания искажения результатов анализа [10].

# Замена пустых значений средними по каждому столбцу

for column in data.columns:

# Проверяем, содержит ли столбец числовые данные

if pd.api.types.is\_numeric\_dtype(data[column]):

# Заменяем пустые значения средними, только если столбец числовой

data[column].fillna(data[column].mean(), inplace=True)

else:

pass # или пропустить столбец

Столбец «Datetime» был преобразован в формат datetime с использованием функции pd.to\_datetime(). Затем данный столбец был установлен в качестве индекса датафрейма. Таким образом набор данных был преобразован во временной ряд (см. рисунок 2.4).

# Преобразование столбца с датой и временем в формат datetime

data['Datetime'] = pd.to\_datetime(data['Datetime'], format='%d.%m.%Y %H:%M')

# Установка столбца с датой и временем в качестве индекса

data.set\_index('Datetime', inplace=True)

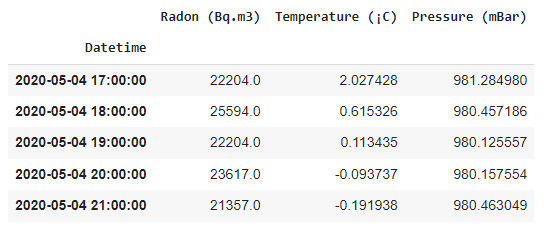


Рисунок 2.4 – Результат преобразования данных

Далее выполнено разделение данных на обучающий, валидационный и тестовый наборы с использованием функции train\_test\_split(). Это стандартная практика в машинном обучении, которая позволяет оценить, насколько хорошо модель будет работать на новых данных.

train\_data, test\_data = train\_test\_split(data, test\_size=0.1, random\_state=42)

train\_data, val\_data = train\_test\_split(train\_data, test\_size=0.15, random\_state=42)

С помощью MinMaxScaler() данные нормализованы в диапазон от 0 до 1. Это делается для улучшения производительности и стабильности алгоритмов машинного обучения.

scaler = MinMaxScaler(feature\_range=(0, 1))

scaler.fit(train\_data)

train\_scaled = scaler.fit\_transform(train\_data)

val\_scaled = scaler.transform(val\_data)

test\_scaled = scaler.transform(test\_data)

Для каждого из наборов данных созданы соответствующие наборы X и y с использованием функции create\_dataset(), которая создает последовательности данных с заданным шагом времени и выполнено преобразование данных в форму для модели RNN.

time\_step = 5

X\_train, y\_train = create\_dataset(train\_scaled, time\_step)

X\_val, y\_val = create\_dataset(val\_scaled, time\_step)

X\_test, y\_test = create\_dataset(test\_scaled, time\_step)

# Изменение формы входных данных для модели RNN

X\_train = X\_train.reshape(X\_train.shape[0], X\_train.shape[1], data.shape[1])

X\_val = X\_val.reshape(X\_val.shape[0], X\_val.shape[1], data.shape[1])

X\_test = X\_test.reshape(X\_test.shape[0], X\_test.shape[1], data.shape[1])

2.2 Построение моделей нейронных сетей

2.2.1 Построение модели RNN

Модель рекуррентной нейронной сети построена следующим образом (см. рисунок 2.5):

1. Создан экземпляр класса Sequential(), который представляет собой линейный стек слоев нейронной сети.
2. Добавлен слой SimpleRNN с 50 нейронами. Параметр return\_sequences=True указывает, что этот слой должен возвращать последовательность выходных данных, а не только последний выход. Это необходимо, потому что следующий слой в модели — это еще один слой RNN. Параметр input\_shape определяет форму входных данных, которую ожидает модель.
3. Добавлен слой Dropout с параметром 0.2, что означает, что во время обучения 20% нейронов будут случайным образом игнорироваться. Это помогает предотвратить переобучение.
4. Добавлен еще один слой SimpleRNN с 50 нейронами. Поскольку это последний слой RNN в модели, return\_sequences по умолчанию установлено в False, что означает, что этот слой возвращает только последний выход.
5. Добавлен слой Dense с одним нейроном. Этот слой выдает окончательный прогноз модели.

# Создание модели RNN

model = Sequential()

model.add(SimpleRNN(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1], X\_train.shape[2])))

model.add(Dropout(0.2))

model.add(SimpleRNN(units=50))

model.add(Dense(1))

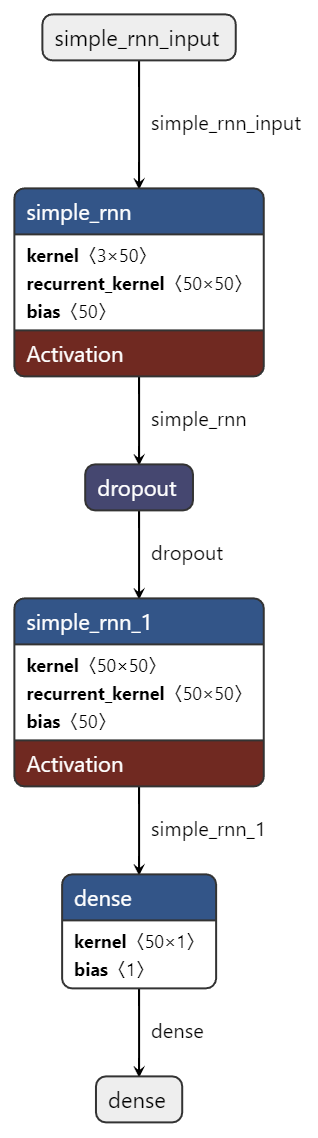


Рисунок 2.5 – Визуализация модели RNN

Был создан экземпляр оптимизатора SGD (стохастический градиентный спуск) с коэффициентом обучения 0.01. Этот оптимизатор был использован при компиляции модели, где в качестве функции потерь была выбрана среднеквадратичная ошибка (mean\_squared\_error).

sgd = SGD(learning\_rate=0.01)

model.compile(loss='mean\_squared\_error', optimizer=sgd)

После компиляции модели был запущен процесс обучения с использованием метода fit(). Входные данные для обучения (X\_train и y\_train) и валидации (X\_val и y\_val) были переданы в этот метод. Количество эпох обучения было установлено равным 20, а размер пакета (batch\_size) - равным 1. Параметр verbose был установлен равным 2, что означает, что в процессе обучения выводилась информация о прогрессе обучения после каждой эпохи.

history = model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=20, batch\_size=1, verbose=2)

Результат обучения представлен на рисунке 2.6.

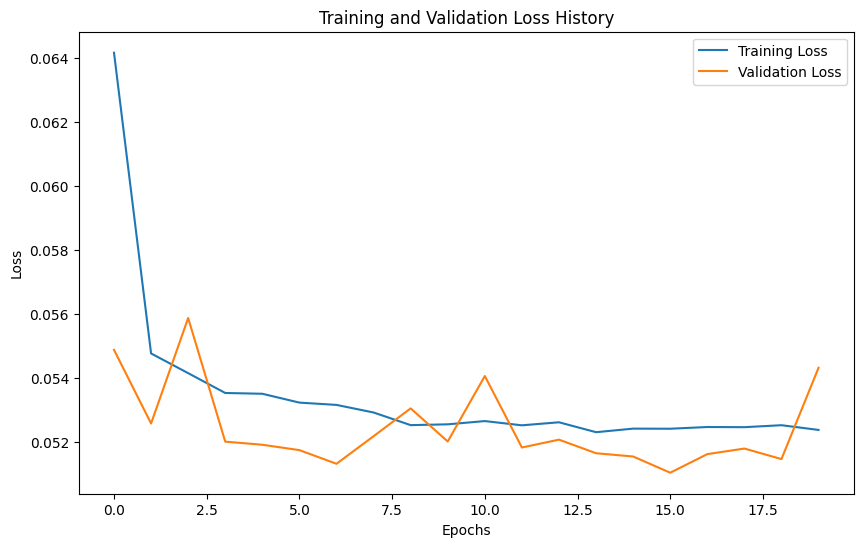


Рисунок 2.6 – Визуализация обучения модели RNN

Значение функции потерь на обучающей выборке начинается примерно с 0.064 и затем резко снижается. После этого оно продолжает колебаться, но в целом показывает тенденцию к снижению, заканчиваясь чуть выше 0.054.

Значение функции потерь на валидационной выборке начинается чуть ниже 0.058, затем во вторую эпоху резко возрастает, превышая начальное значение обучающей потери. После этого оно также показывает тенденцию к снижению с колебаниями, но остается выше значения обучающей потери на протяжении всех показанных эпох.

Эти наблюдения могут указывать на то, что модель обучается и улучшает свою производительность на обучающих данных, но она может быть переобучена, поскольку потеря на валидационных данных остается выше, чем на обучающих данных. Это может означать, что модель может плохо обобщать на новые, ранее не виденные данные.

2.2.2 Построение модели LSTM

LSTM-модель рекуррентной нейронной сети представлена следующими слоями (см. рисунок 2.7):

1. Создан экземпляр класса Sequential(), который представляет собой линейный стек слоев нейронной сети.
2. Добавлен слой LSTM с 50 нейронами. Параметр return\_sequences=True указывает, что этот слой должен возвращать последовательность выходных данных, а не только последний выход. Это необходимо, потому что следующий слой в модели — это еще один слой LSTM. Параметр input\_shape определяет форму входных данных, которую ожидает модель.
3. Добавлен слой Dropout с параметром 0.2, что означает, что во время обучения 20% нейронов будут случайным образом игнорироваться. Это помогает предотвратить переобучение.
4. Добавлен еще один слой LSTM с 50 нейронами. Поскольку это последний слой LSTM в модели, return\_sequences по умолчанию установлено в False, что означает, что этот слой возвращает только последний выход.
5. Добавлен слой Dense с одним нейроном. Этот слой выдает окончательный прогноз модели.

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1], X\_train.shape[2])))

model.add(Dropout(0.2))

model.add(LSTM(units=50))

model.add(Dense(1))

Оптимизатор, функция потерь и количество эпох обучения использованы аналогичные предыдущей модели.

sgd = SGD(learning\_rate=0.01)

model.compile(loss='mean\_squared\_error', optimizer=sgd)

history = model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=20, batch\_size=1, verbose=2)

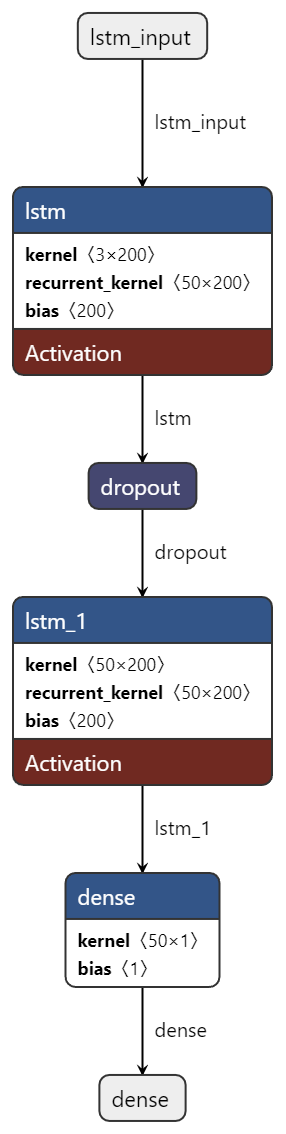


Рисунок 2.7 – Визуализация модели LSTM

Результат обучения представлен на рисунке 2.8.

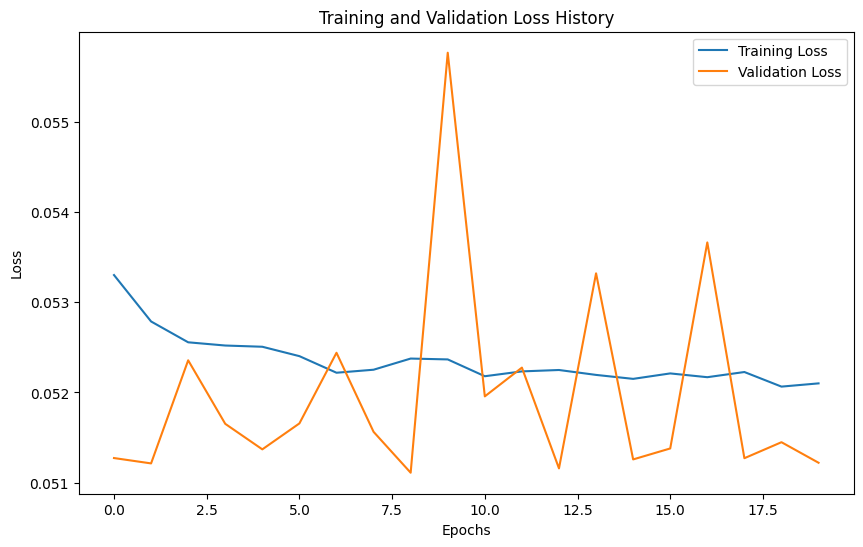


Рисунок 2.8 – Визуализация обучения модели LSTM

Значение функции потерь на обучающей выборке начинается примерно с 0.055 и затем колеблется, но в целом показывает тенденцию к снижению.

Значение функции потерь на валидационной выборке начинается чуть выше 0.051, затем колеблется, но остается ниже значения обучающей потери на протяжении всех показанных эпох.

Эти наблюдения могут указывать на то, что модель обучается и улучшает свою производительность на обучающих данных.

2.2.3 Построение модели GRU

Модель нейронной сети Gated Recurrent Unit представлена следующими слоями (см. рисунок 2.9):

1. Создан экземпляр класса Sequential(), который представляет собой линейный стек слоев нейронной сети.
2. Добавлен слой GRU с 50 нейронами. Параметр return\_sequences=True указывает, что этот слой должен возвращать последовательность выходных данных, а не только последний выход. Это необходимо, потому что следующий слой в модели — это еще один слой GRU. Параметр input\_shape определяет форму входных данных, которую ожидает модель.
3. Добавлен слой Dropout с параметром 0.2, что означает, что во время обучения 20% нейронов будут случайным образом игнорироваться. Это помогает предотвратить переобучение.
4. Добавлен еще один слой GRU с 50 нейронами. Поскольку это последний слой GRU в модели, return\_sequences по умолчанию установлено в False, что означает, что этот слой возвращает только последний выход.
5. Добавлен слой Dense с одним нейроном. Этот слой выдает окончательный прогноз модели.

Оптимизатор, функция потерь и количество эпох обучения использованы аналогичные предыдущим моделям.

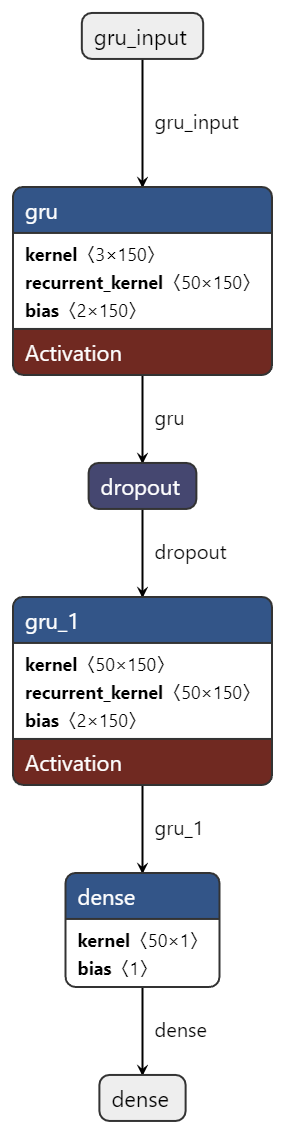


Рисунок 2.9 – Визуализация обучения модели GRU

Результат обучения представлен на рисунке 2.10.

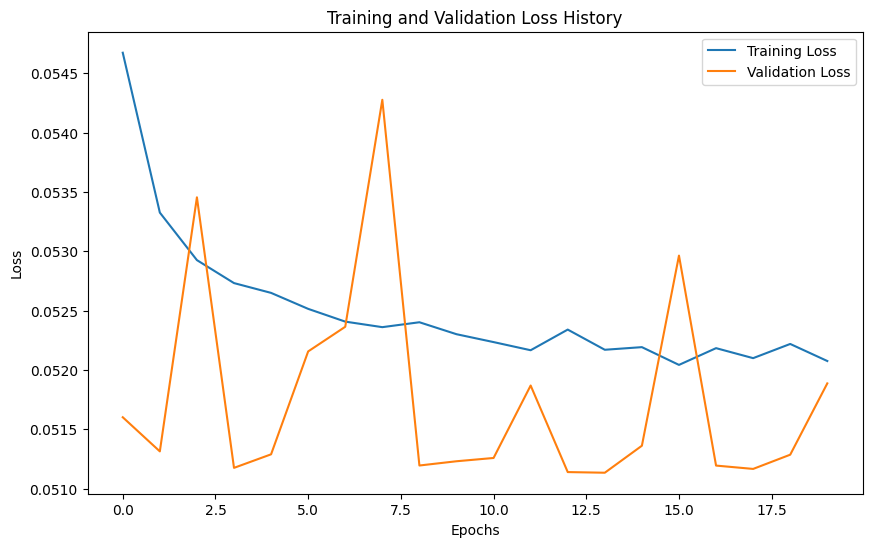


Рисунок 2.10 – Визуализация обучения модели GRU

Значение функции потерь на обучающей выборке начинается примерно с 0.0546 и затем колеблется, но в целом показывает тенденцию к снижению.

Значение функции потерь на валидационной выборке начинается чуть выше 0.051, затем колеблется, но остается ниже значения обучающей потери на протяжении всех показанных эпох.

Эти наблюдения могут указывать на то, что модель обучается и улучшает свою производительность на обучающих данных, но может быть переобучена, поскольку потеря на валидационных данных остается выше, чем на обучающих данных.

2.2.4 Построение модели Bidirectional RNN

Bidirectional RNN представлена следующими слоями (см. рисунок 2.11):

1. Создан экземпляр класса Sequential(), который представляет собой линейный стек слоев нейронной сети.
2. Добавлен слой Bidirectional, в который вложен слой SimpleRNN с 50 нейронами. Параметр return\_sequences=True указывает, что этот слой должен возвращать последовательность выходных данных, а не только последний выход. Это необходимо, потому что следующий слой в модели — это еще один слой Bidirectional. Параметр input\_shape определяет форму входных данных, которую ожидает модель.
3. Добавлен слой Dropout с параметром 0.2.
4. Добавлен еще один слой Bidirectional, в который вложен слой SimpleRNN с 50 нейронами. Поскольку это последний слой RNN в модели, return\_sequences по умолчанию установлено в False, что означает, что этот слой возвращает только последний выход.
5. Добавлен слой Dense с одним нейроном. Этот слой выдает окончательный прогноз модели.

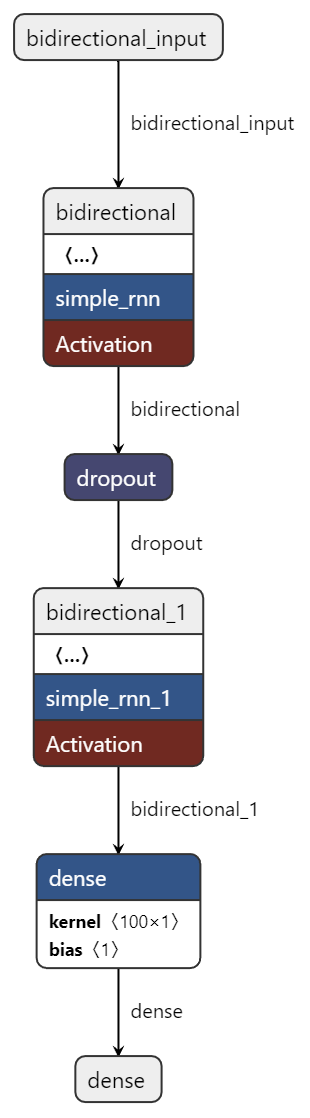


Рисунок 2.11 – Визуализация обучения модели Bidirectional RNN

Результат обучения представлен на рисунке 2.12.

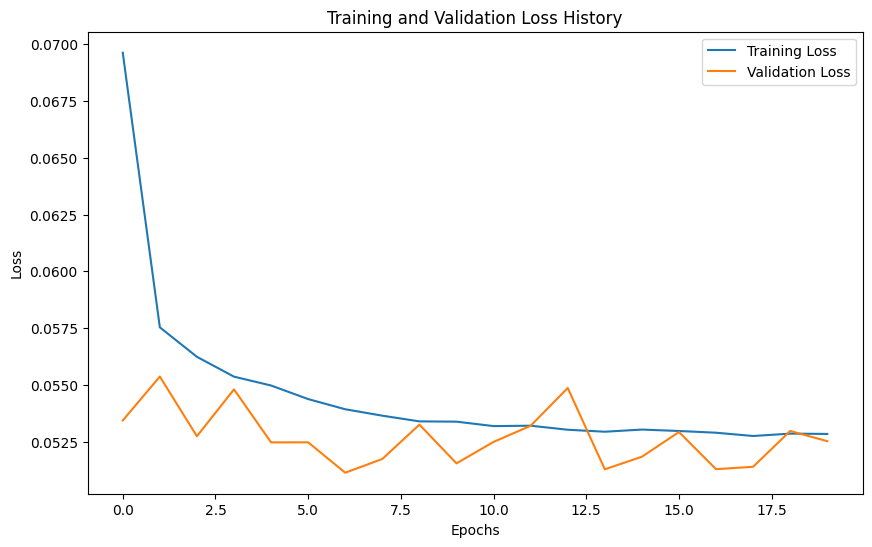


Рисунок 2.12 – Визуализация обучения модели Bidirectional RNN

Значение функции потерь на обучающей выборке начинается примерно с 0.0700 и затем резко снижается, после чего продолжает снижаться более плавно, заканчиваясь чуть выше 0.0525.

Значение функции потерь на валидационной выборке начинается чуть ниже 0.0550, затем колеблется, но остается выше значения обучающей потери на протяжении некоторых показанных эпох.

2.2.5 Построение модели Deep RNN

Deep RNN модель нейронной сети представлена следующими слоями (см. рисунок 2.13):

1. Создан экземпляр класса Sequential(), который представляет собой линейный стек слоев нейронной сети.
2. Добавлен первый слой SimpleRNN с 50 нейронами. Параметр return\_sequences=True указывает, что этот слой должен возвращать последовательность выходных данных, а не только последний выход. Это необходимо, потому что следующий слой в модели — это еще один слой SimpleRNN. Параметр input\_shape определяет форму входных данных, которую ожидает модель.
3. Добавлен слой Dropout с параметром 0.2.
4. Добавлен второй слой SimpleRNN с 50 нейронами. Этот слой также возвращает последовательность выходных данных, что позволяет добавить еще один слой SimpleRNN после него.
5. Добавлен еще один слой Dropout с параметром 0.2 для предотвращения переобучения.
6. Добавлен третий слой SimpleRNN с 50 нейронами. Поскольку это последний слой RNN в модели, return\_sequences по умолчанию установлено в False, что означает, что этот слой возвращает только последний выход.
7. Добавлен слой Dense с одним нейроном. Этот слой выдает окончательный прогноз модели.

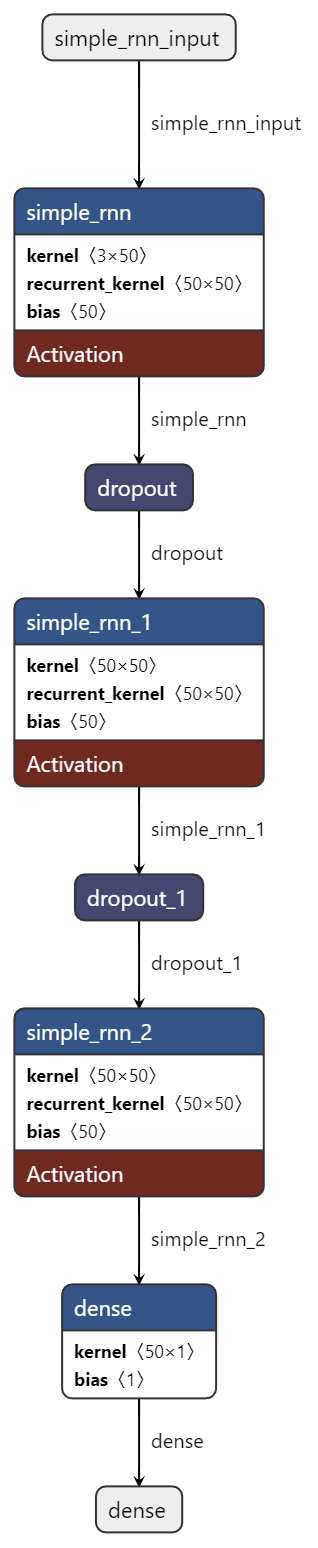


Рисунок 2.13 – Визуализация обучения модели Bidirectional RNN

Результат обучения представлен на рисунке 2.14.

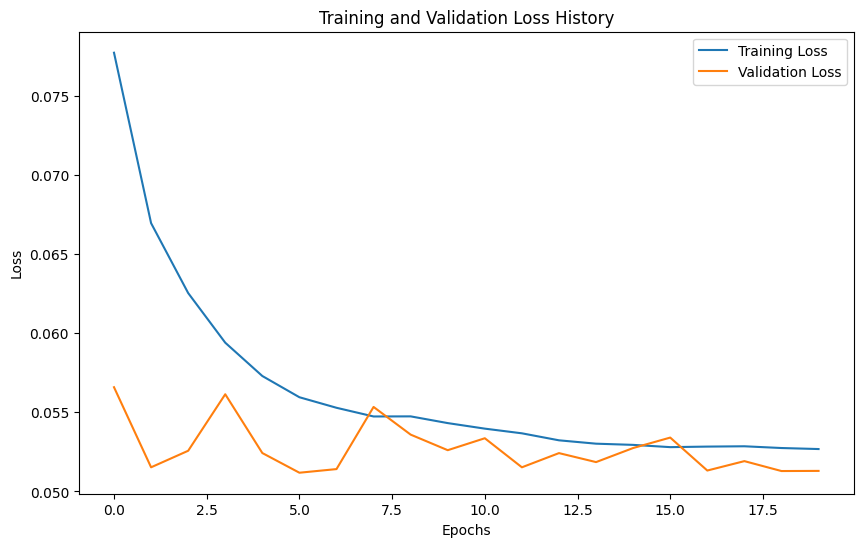


Рисунок 2.14 – Визуализация обучения модели Deep RNN

График говорит о том, что модель хорошо обучается на тренировочных данных (что подтверждается уменьшением потерь на обучении), а валидационные потери дают представление о том, насколько хорошо модель обобщает новые данные.

Однако, несмотря на общий нисходящий тренд, валидационные потери имеют некоторый разброс, что может указывать на некоторую степень переобучения. Это может быть связано с тем, что модель слишком хорошо подстраивается под тренировочные данные и теряет способность к обобщению.

2.3 Тестирование моделей

Для тестирования моделей использовалось несколько способов:

1. Сравнение моделей с помощью метрик.
2. Предсказание нескольких значений и сравнение их с истинными значениями тестового набора.
3. Сравнение показателей потерь на валидационных выборках.

Для сравнения обученных моделей использовались три метрики: MSE, MAE и R2 (см. рисунок 2.15).

Среднеквадратичная ошибка (MSE) – это среднее значение квадратов ошибок. Ошибкой является разница между фактическим значением и прогнозируемым значением.

Средняя абсолютная ошибка (MAE) – это среднее значение абсолютных ошибок.

Коэффициент детерминации (R2) – это статистическая мера, которая показывает, насколько близки данные к подогнанной регрессионной линии. Он также известен как коэффициент детерминации или коэффициент множественной детерминации для множественной регрессии.

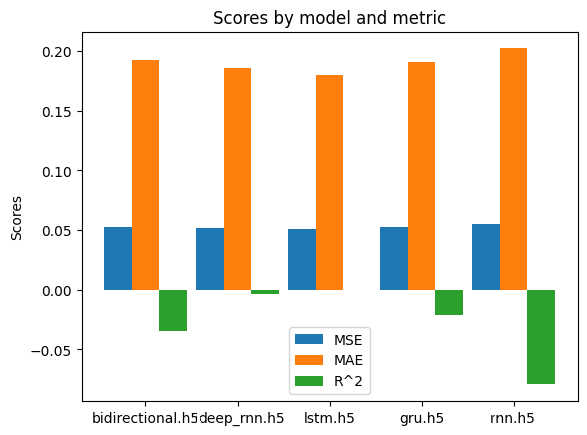


Рисунок 2.15 – Диаграмма сравнения моделей

MSE (синие столбцы) – значения MSE для всех моделей примерно одинаковы.

MAE (оранжевые столбцы) – значения MAE также примерно одинаковы для всех моделей и достаточно высоки.

R2 (зелёные столбцы) – значения R2 варьируются, причём все значения, кроме rnn.h5, близки к нулю или даже отрицательны. Это указывает на то, что модели плохо объясняют вариацию данных. Особенно высокое отрицательное значение наблюдается у модели RNN.

В целом, все модели нуждаются в улучшении для повышения точности предсказаний.

Для визуальной проверки точности предсказания каждая модель предсказала десять значений на временном отрезке тестовой выборки (см. рисунок 2.16).

На изображении представлены пять линейных графиков, каждый из которых представляет производительность различной предсказательной модели: модели RNN, модели LSTM, модели GRU, модели Bidirectional RNN и модели Deep RNN. Каждый график отображает две линии: одну для предсказанных значений и одну для истинных значений по ряду точек данных. Графики предназначены для сравнения того, насколько тесно предсказания каждой модели соответствуют истинным значениям.

Анализируя результаты предсказаний моделей по сравнению с истинными значениями:

1. Модель RNN (Recurrent Neural Network) показывает некоторое совпадение между предсказанными и истинными значениями, но с заметными отклонениями.
2. Модель LSTM (Long Short-Term Memory) имеет более тесную подгонку к истинным значениям с меньшим количеством отклонений, чем базовая RNN.
3. График модели GRU (Gated Recurrent Unit) указывает на аналогичную производительность с LSTM, с тесным отслеживанием истинных значений, но небольшими расхождениями.
4. График модели Bidirectional RNN демонстрирует улучшенную подгонку по сравнению с простыми RNN, но все же есть области, где предсказание отличается от фактических данных.
5. График модели Deep RNN показывает, что, хотя он хорошо отражает некоторые тренды в точках данных, есть значительные участки, где предсказания не совпадают с фактическими значениями.

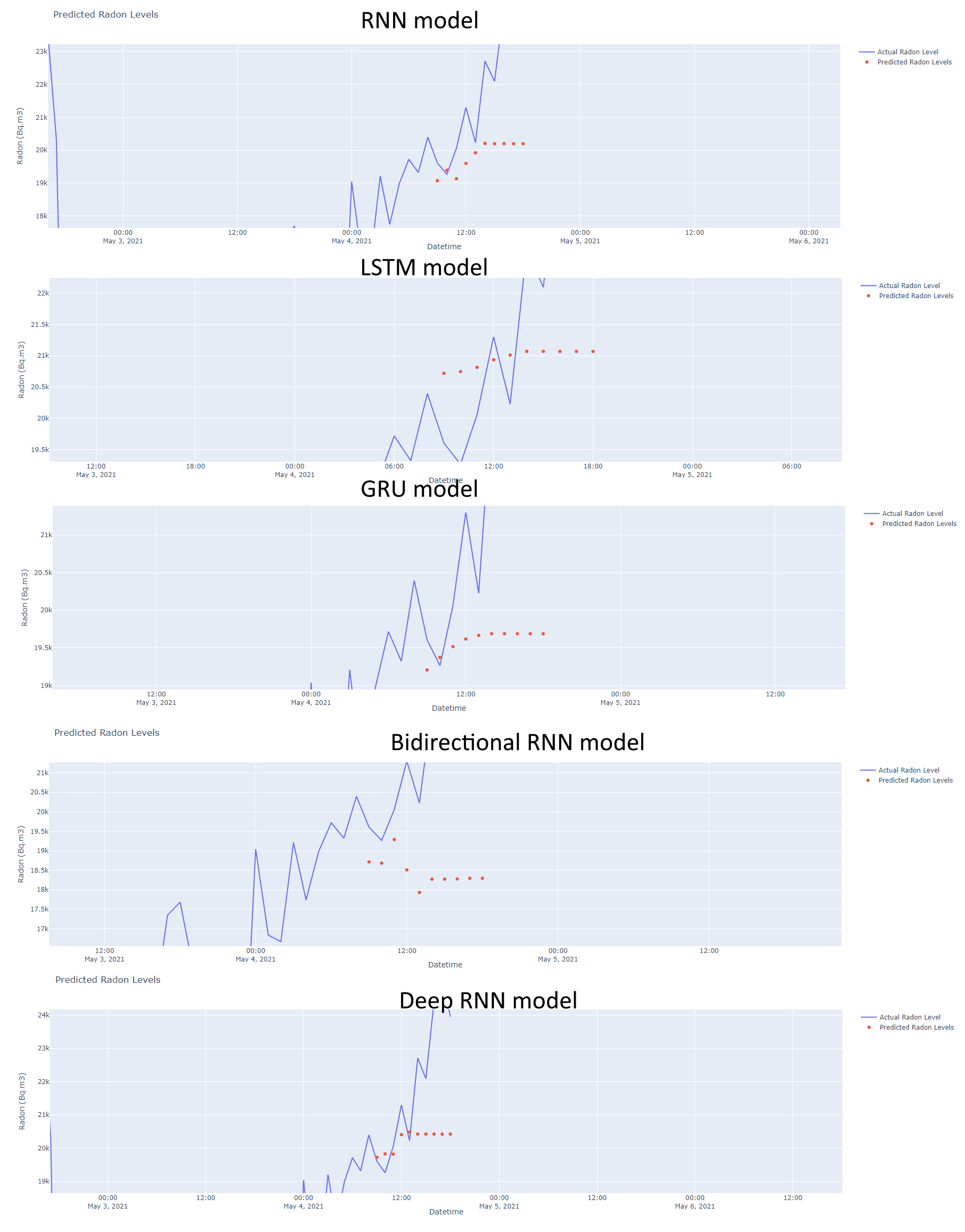


Рисунок 2.16 – Визуализация предсказаний моделей

На графике «Validation Loss over Epochs» представлены валидационные потери пяти различных моделей: rnn (желтый), bidirectional (синий), deep\_rnn (красный), gru (зеленый), lstm (фиолетовый) в процессе обучения (см. рисунок 2.17).

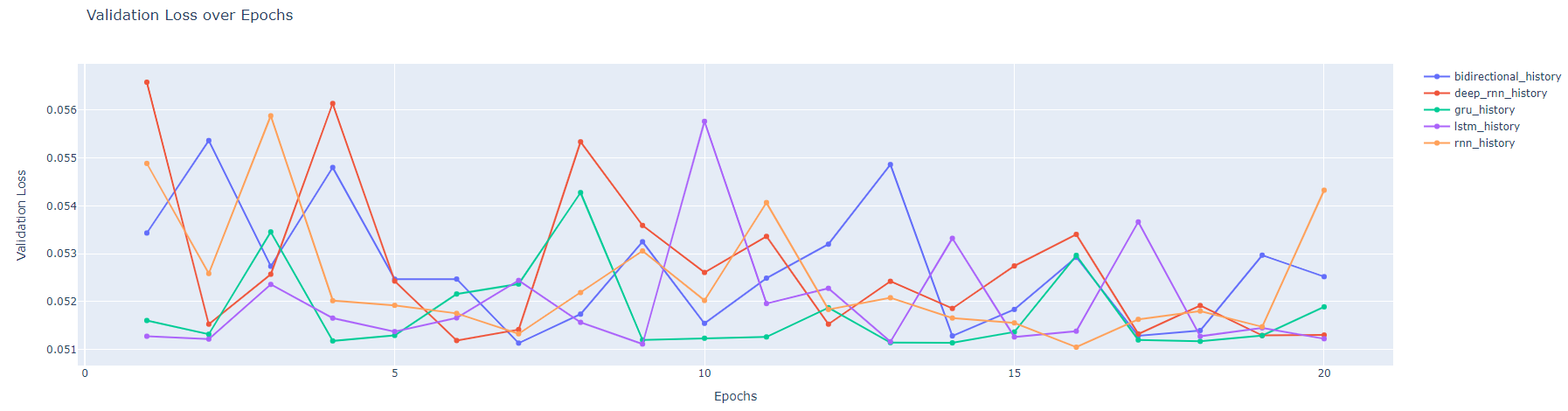


Рисунок 2.17 – Визуализация валидационных потерь моделей

Судя по графику, кажется, что модель GRU (зеленая линия) в целом имеет более низкие значения на протяжении большинства эпох по сравнению с другими моделями, что может указывать на то, что она обучилась лучше с точки зрения поддержания более низких валидационных потерь.

Таким образом, судя по графику, модель rnn (желтый) обучилась лучше всего. Однако, необходимо учесть, что все представленные кривые на графике имеют заметные колебания, что свидетельствует о возможном переобучении моделей.

Выводы по второй главе

В ходе работы с моделями и их тестировании, были представлены ключевые этапы оценки производительности моделей рекуррентных нейронных сетей для прогнозирования землетрясений на основе данных о радоновых выбросах. Основные моменты включают сравнение моделей с использованием различных метрик, предсказание значений и анализ показателей потерь на валидационных выборках.

Эффективное тестирование моделей позволяет оценить их способность обобщения на новые данные и выявить возможное переобучение. Результаты тестирования помогают выбрать наиболее подходящую модель для задачи прогнозирования землетрясений на основе радоновых выбросов.

Таким образом, работа с моделями и их тестирование играют важную роль в разработке и оптимизации алгоритмов машинного обучения для прогнозирования землетрясений, обеспечивая точность и надежность результатов прогнозирования.

Заключение

Прогнозирование землетрясений является одной из критически важных задач, от решения которой зависит безопасность и благополучие людей.

Современные методы искусственного интеллекта, такие как рекуррентные нейронные сети (РНС), предоставляют уникальные возможности для анализа данных и прогнозирования сейсмической активности.

В ходе выполнения работы были изучены основные принципы построения РНС, что позволило понять механизм их работы и специфику применения в задаче прогнозирования землетрясений.

Проведенное сравнение различных архитектур РНС позволило выявить различия и уникальные возможности каждой из них.

В результате применения различных архитектур РНС для прогнозирования землетрясений были получены результаты, которые могут быть использованы в дальнейшем исследовании.

Рекуррентные нейронные сети, такие как LSTM (точность модели – 0,9468) и GRU (точность модели – 0,9487), показывают более точные результаты при прогнозировании по сравнению с базовыми RNN (точность модели – 0,9438) моделями.

Модель Bidirectional RNN (точность модели – 0,9471) демонстрирует улучшенную подгонку к истинным значениям, однако все еще имеет области с отклонениями от фактических данных.

Глубокие рекуррентные нейронные сети Deep RNN (точность модели – 0,9456) способны отражать некоторые тренды в данных, но имеют значительные участки, где предсказания не совпадают с реальными значениями.

Исходя из вышеперечисленных результатов, можно сделать вывод, что выбор оптимальной архитектуры рекуррентной нейронной сети играет ключевую роль в прогнозировании землетрясений на основе данных о радоновых выбросах.

Таким образом, выполнение поставленных задач позволило не только изучить и сравнить различные архитектуры рекуррентных нейронных сетей, но и оценить их применимость для прогнозирования землетрясений на основе данных о радоновых выбросах.

Список использованных источников

1. Суханова, Н. В. разработка нейросетевой модели для прогнозирования вероятности землетрясений / Н. В. Суханова // Автоматизация и моделирование в проектировании и управлении. – 2023. – № 2(20). – С. 40-49. – DOI 10.30987/2658-6436-2023-2-40-49. – EDN EDIMKN.
2. F. Azam ARTIFICIAL INTELLIGENCE BASED TECHNIQUES FOR EARTHQUAKE PREDICTION: A REVIEW / F. Azam, M. Yasmin, S. Mohsin, M. Sharif // Engineering, Environmental Science, Geology, Computer Science. - 2014. - №26(4). - С. 1495-1502.
3. Разработка математической модели для построения имитационной модели системы передачи информации / Е. В. Заргарян, Ю. А. Заргарян, А. Д. Коринец, И. М. Малышенко // Современная техника и технологии. – 2015. – № 4(44). – С. 53-56. – EDN UAEVIL.
4. Использование рекуррентных сетей для прогнозирования потребления электроэнергии / С. Л. Подвальный, М. А. Лихотин, А. В. Михайлусов, А. К. Донских // Вестник Воронежского государственного технического университета. – 2022. – Т. 18, № 3. – С. 45-50. – DOI 10.36622/VSTU.2022.18.3.005. – EDN SGJIWZ.
5. Y. Yu A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures / Y. Yu, X. Si, C. Hu, J. Zhang // Neural Computation. - 2019. - №31 (7). - С. 1235–1270.
6. J. Chung Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling / J. Chung, C. Gulcehre, K. Cho, Y. Bengio // Arxiv. - 2014. - №1. - С. 1-9.
7. S. Obeta A Comparative Study of Long Short-Term Memory and Gated Recurrent Unit / S. Obeta, E. Grisan, C. V. Kalu // e Data and Learning Machine ,Intelligence Artificial of J. - 2023. - №1(1). - С. 1-9.
8. Y. Su On Extended Long Short-term Memory and Dependent Bidirectional Recurrent Neural Network / Y. Su, Y. Huang, C.-C. Jay Kuo // Neurocomputing. - 2018. - №356. - С. 1-31.
9. Yaser S. Abu Mostafa, Malik Magdon Ismail, Hsuan Tien Lin Learning from data: a short course. - 1 изд. - NYC: AMLBook.com, 2012. - 215 с.