СРАВНИТЕЛЬНЫЙ АНАЛИЗ РАЗЛИЧНЫХ АРХИТЕКТУР НЕЙРОННЫХ СЕТЕЙ ДЛЯ ЗАДАЧ РЕГРЕССИИ

В.Н. Комличенко, *к.т.н., доцент, Белорусский государственный университет информатики и радиоэлектроники, v.komlitchenko@gmail.com;*
В.А. Федосенко, *к.т.н., доцент, Белорусский государственный университет информатики и радиоэлектроники, fedosenko@bsuir.by;*
А.С. Купрейчик, *Белорусский государственный университет информатики и радиоэлектроники, kuprejcikaleksandra@gmail.com.*

УДК 004.032.26

Аннотация. В статье представлено сравнительное исследование различных архитектур нейронных сетей, применяемых для решения задач регрессии. Анализ охватывает популярные архитектуры с использованием реальных наборов данных. Результаты исследования позволяют определить сильные и слабые стороны каждой архитектуры в зависимости от специфики данных и целей задачи.

Ключевые слова: нейронные сети; глубокое обучение; регрессия; сравнительный анализ; архитектура; *MLP; CNN; RNN; LSTM; Transformer; MSE; MAE; R-squared.*

COMPARATIVE ANALYSIS OF DIFFERENT NEURAL NETWORK ARCHITECTURES FOR REGRESSION PROBLEMS

V.N. Komlichenko, *Candidate of technical Science, Associate Professor, Belarusian State University of Informatics and Radioelectronics;*
V.A. Fedosenko, *Candidate of Technical Science, Associate Professor, Belarusian State University of Informatics and Radioelectronics;*
A.S. Kupreichyk, *Belarusian State University of Informatics and Radioelectronics.*

Annotation. The article presents a comparative study of various neural network architectures used to solve regression problems. The analysis covers popular architectures using real data sets. The results of the study allow us to identify the strengths and weaknesses of each architecture, depending on the specifics of the data and the objectives of the task.

Keywords: neural networks; deep learning; regression; comparative analysis; architecture; *MLP; CNN; RNN; LSTM; Transformer; MSE; MAE; R-squared.*

Введение

Аппарат регрессионного анализа предоставляет ряд статистических методов, позволяющих на заданном уровне значимости решать задачи классификации. Регрессия как метод машинного обучения применим для предсказания как дискретной, так и непрерывной переменной на основе ряда предопределенных факторов (непрерывных, дискретных, лаговых эндогенных).

В контексте машинного обучения под регрессионным анализом понимается процесс построения математической модели, описывающей зависимость некоторой целевой характеристики объекта или процесса от других его характеристик [1].

В задаче регрессионного анализа всегда есть обучающая выборка, состоящая из входных параметров и откликов, а также начальная параметрическая модель, в самом простом случае – линейная, однако не обязательно таковая.

Существуют различные способы решения задач машинного обучения, например, такие программы, как *Excel* или *Wolfram Mathematica*, или даже сведение к математической модели задачи математического программирования. Однако профессионалы в области машинного обучения обычно используют либо *Python*, либо язык *R* [2]. В табл. 1 приведены основные библиотеки, написанные на *Python* для решения задач машинного обучения, и их возможности.

Таблица 1.

NumPy	Поддержка многомерных массивов и математических функций для работы с ними
Pandas	Группировка, комбинирование, фильтрация данных. Анализ временных рядов
Scikit-learn	Решение задач классическими алгоритмами МО
Matplotlib	Визуализация изображений

Нейронные сети являются эффективными инструментами для решения задач регрессии, обеспечивая высокую гибкость и способность к моделированию сложных зависимостей в данных.

Различные архитектуры нейронных сетей предназначены для решения специфических задач, и каждая из них имеет свои сильные стороны и ограничения, что делает выбор оптимальной архитектуры критическим шагом при решении конкретных задач [3].

Архитектуры нейронных сетей

Далее будет рассмотрено несколько популярных архитектур нейронных сетей, а также охарактеризованы их преимущества и недостатки.

1. Многослойные персептроны (*MLP*) – это базовая нейронная сеть, состоящая из нескольких слоев нейронов, соединенных между собой. Каждый нейрон в слое получает входные данные от нейронов предыдущего слоя и применяет нелинейную функцию активации для вычисления своего выхода.

MLP обучается с помощью алгоритма обратного распространения ошибки (*backpropagation*). В процессе обучения веса и смещения нейронов корректируются для минимизации ошибки между предсказанными и фактическими значениями [4].

2. Сверточные нейронные сети (*CNN*) – это тип нейронной сети, специализированной для обработки пространственных данных, таких как изображения. Она использует сверточные фильтры, которые скользят по входным данным и выделяют определенные пространственные паттерны.

CNN обучается с использованием алгоритма обратного распространения ошибки, аналогично *MLP*. Однако в *CNN* используются дополнительные операции, такие как пулинг (*pooling*) и нормализация (*normalization*), которые помогают предотвратить переобучение и извлечь более устойчивые признаки [5].

3. Рекуррентные нейронные сети (*RNN*) – это тип нейронной сети, разработанный для обработки последовательных данных, таких как тексты, временные ряды и речь. В отличие от *MLP* и *CNN*, *RNN* обладает памятью, которая позволяет учитывать прошлые состояния для прогнозирования будущих состояний.

RNN обучается с использованием алгоритма обратного распространения ошибки, но с модификациями, которые учитывают циклическую природу сети. В *RNN* используются механизмы, такие как *LSTM* (*Long Short-Term Memory*) или *GRU* (*Gated Recurrent Unit*), которые помогают сети сохранять информацию о долгосрочных зависимостях в данных [6].

4. Долгая краткосрочная память (*LSTM*) – это специализированный тип *RNN*, который решает проблему исчезающего градиента, возникающую при обучении глубоких *RNN*. Он использует специальные механизмы памяти для запоминания информации на протяжении длительных временных интервалов.

В процессе обучения данная архитектура анализирует последовательные данные и запоминает контекстные зависимости, что делает *LSTM* особенно полезной для задач, связанных с временными рядами или обработкой естественного языка [7].

5. Трансформер (*Transformer*) – это архитектура нейронных сетей, которая произвела революцию в обработке естественного языка (*NLP*) и достигла впечатляющих результатов в различных задачах, таких как машинный перевод, анализ текста, автоматическое составление текстов и других.

Обучение трансформеров – это сложный процесс, который включает в себя несколько этапов: подготовку данных, обучение, оценку и использование. Сначала данные, обычно текст, преобразуются в формат, понятный для нейронной сети, включающий токенизацию, нормализацию и встраивание слов. Затем данные делятся на обучающую, валидационную и тестовую выборки. Обучение происходит с помощью алгоритма обратного распространения, где модель делает предсказания и корректирует свои веса, чтобы минимизировать ошибку. В процессе обучения применяются различные методы оптимизации и регуляризации для улучшения точности и предотвращения переобучения. После обучения модель оценивается на тестовой выборке, чтобы определить ее эффективность и выбрать наилучшую модель. Ключевыми моментами обучения трансформеров являются использование большого количества данных, вычислительные ресурсы и правильный подбор гиперпараметров и методов регуляризации [8].

Для работы с *MLP, CNN, RNN, LSTM* и *Transformer* обычно используются библиотеки *TensorFlow/Keras, PyTorch*.

TensorFlow/Keras – наиболее популярный выбор, предлагающий богатый набор инструментов для создания и обучения нейронных сетей. *PyTorch* предоставляет гибкий интерфейс и динамическое вычисление графов. *Hugging Face Transformers* специально разработана для работы с моделями *Transformer* и предоставляет доступ к множеству предобученных моделей для задач обработки естественного языка, таких как *BERT, GPT* и другие [9].

При работе с задачами регрессии важно правильно оценивать качество моделей. Существуют различные метрики, которые помогают понять, насколько хорошо модель предсказывает целевые значения. Численная оценка результатов прогнозирования проведена с помощью сравнения следующих параметров:

Среднеквадратичная ошибка (*Mean Squared Error, MSE*) –

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2;$$

Корень из среднеквадратичной ошибки (*Root Mean Squared Error, RMSE*) –

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2};$$

Средняя абсолютная погрешность (*Mean Absolute Error, MAE*) –

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|;$$

Коэффициент детерминации (*Coefficient of determination,* R^2) –

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\bar{y}_i - y_i)^2}.$$

Для сравнительного анализа использовались не только разные архитектуры, но и наборы данных. А именно:

1. *Global Superstore Transaction Data* (2011-2014) [10], в котором содержится информация об операциях купли-продажи в розничном магазине, действовавшем с 2011 по 2014 г. Данные включают: информацию о клиенте (идентификатор, сегмент, местоположение) и продукте (идентификатор, категория, название, продажи, прибыль), а также детали заказа (дата, идентификатор, доставка).

2. *Predict Conversion in Digital Marketing Dataset* [11]. Эти данные предоставляют исчерпывающую информацию о взаимодействии клиентов с цифровыми маркетинговыми кампаниями. Они включают в себя демографические данные, маркетинговые показатели, показатели вовлеченности клиентов и исторические данные о покупках.

Оба набора данных прошли несколько шагов предобработки данных. Были исключены неподходящие признаки, пропущенные значения и дубликаты, преобразованы типы данных. Также прошел отбор признаков, которые содержат релевантные данные. Преобразованы даты в соответствующий формат с помощью *pd.to_datetime*. Затем дата конвертируется в числовой формат: отнимается дата 1 января 1970 г. (начало эпохи *Unix*), и результатом становится количество дней с этой даты. Это необходимо для того, чтобы алгоритмы машинного обучения могли обрабатывать даты как числовые значения [12].

Также было выполнено кодирование категориальных данных с использованием *One-Hot Encoding*. Данный способ основывается на создании бинарных признаков, которые показывают принадлежность к уникальному значению. Столбцы преобразуются в множество двоичных (0/1) признаков.

После создаются словари, сопоставляющие уникальные идентификаторы заказов, продуктов и клиентов с их индексами в наборах данных. Это необходимо, чтобы заменить длинные строковые значения на более компактные целочисленные индексы.

На заключительном этапе происходит замена оригинальных значений идентификаторов в датасете на соответствующие им индексы из созданных словарей. Это обеспечивает лучшую производительность и подходит для большинства алгоритмов машинного обучения, поскольку они работают лучше с числовыми значениями.

После чего данные были разделены на обучающую и тестовую выборки. Сначала удаляется столбец, который будет использоваться как целевая переменная, и сохраняется в переменную *X*. Затем этот столбец сохраняется в переменную *y*. Далее, с помощью функции *train_test_split* из библиотеки *sklearn.model_selection*, датасет делится на обучающую и тестовую выборки, где 20% данных используются для тестирования, а оставшиеся 80% – для обучения. Для воспроизводимости разделения данных устанавливается «случайное число» *random_state*=42. В результате получаются четыре переменные: *X_train* и *y_train* – обучающая выборка, *X_test* и *y_test* – тестовая выборка. Обучающая выборка будет использоваться для обучения модели регрессии, а тестовая выборка – для оценки ее производительности.

Также в коде выполняется процесс масштабирования числовых признаков с использованием стандартного масштабирования (*Standard Scaling*) [13].

Метод *fit_transform* сначала вычисляет среднее значение и стандартное отклонение для указанных признаков и затем применяет трансформацию, чтобы привести данные к стандартному виду (т.е. вычитает среднее и делит на стандартное отклонение). Полученные значения заменяют оригинальные данные в соответствующих столбцах в обучающем наборе данных *X_train.*

Сразу после этого выполняется масштабирование тех же признаков в тестовом наборе данных *X_test*. Важный момент заключается в том, что здесь используется только метод *transform*, а не *fit_transform*. Это связано с тем, что отсутствует необходимость в пересчете средних и стандартных отклонений на тестовых данных во избежание утечки информации из тестового набора в обучающий. Вместо этого следует использовать те же параметры, которые были получены на обучающем наборе данных, и применить их к тестовым данным.

Далее была проведена настройка гиперпараметров для каждой модели.

MLP с помощью *Sequential()* создает последовательную модель, в которой слои добавляются один за другим. Были установлены два полносвязных слоя с 128 и 64 нейронами соответственно и функцией активации *ReLU*, а также выходной слой с одним нейроном (для предсказания одного значения).

CNN через *Conv1D* формирует сверточный слой для одномерных данных с 32 фильтрами и размером ядра 3. После происходит уменьшение размерности выходных данных через слой подвыборки (*MaxPooling1D*) и преобразование многомерного тензора в одномерный вектор перед подачей в полносвязный слой (*Flatten*).

RNN использует рекуррентный слой со 128 нейронами, который обрабатывает последовательные данные, учитывая предшествующие значения.

LSTM на первом слое возвращает последовательности и позволяет передать выходные данные следующему слою. Первый слой имеет 128 нейронов, второй – 64.

Transformer, применяя *Embedding,* преобразует целочисленные индексы в плотные векторы фиксированной длины, что полезно для обработки категориальных данных. Остальные шаги аналогичны *CNN.*

Все модели используют оптимизатор *Adam* и функцию потерь *MSE* и метрику (*MAE*), что типично для задач регрессии.

Оптимизатор – это алгоритм, используемый для незначительного изменения параметров, таких как веса и скорость обучения, чтобы модель работала правильно и быстро.

Adam – один из самых эффективных алгоритмов оптимизации в обучении нейронных сетей. Он сочетает в себе идеи *RMSProp* и оптимизатора импульса. Вместо того чтобы адаптировать скорость обучения параметров на основе среднего первого момента (среднего значения), как в *RMSProp*, *Adam* также использует среднее значение вторых моментов градиентов [14]. В частности, алгоритм вычисляет экспоненциальное скользящее среднее градиента и квадратичный градиент, а параметры *beta*1 и *beta*2 управляют скоростью затухания этих скользящих средних [15].

Анализ полученных результатов

Результат обучения моделей на наборе данных *Predict Conversion in Digital Marketing Dataset* представлен в табл. 2. В ней явно видно, что при количестве эпох, равном 10, *RNN* показала наилучшие результаты по всем метрикам: наименьшее значение *MSE* (0,0843), *MAE* (0,1792) и *RMSE* (0,2903), а также наивысшее значение R^2 (0,2091). Это свидетельствует о высокой способности модели к объяснению вариации в целевой переменной.

CNN и *LSTM* демонстрируют достойные результаты, но не дотягивают до показателей *RNN*. Особенно выделяется *LSTM* с самым низким *MAE* (0,1491), что может свидетельствовать о хорошей предсказательной способности модели в этом контексте. *MLP* достигает худших результатов среди представленных моделей. Хотя его показатели *MSE* и *RMSE* достаточно близки к другим архитектурам, значение R^2 (0,1051) указывает на худшее объяснение дисперсии.

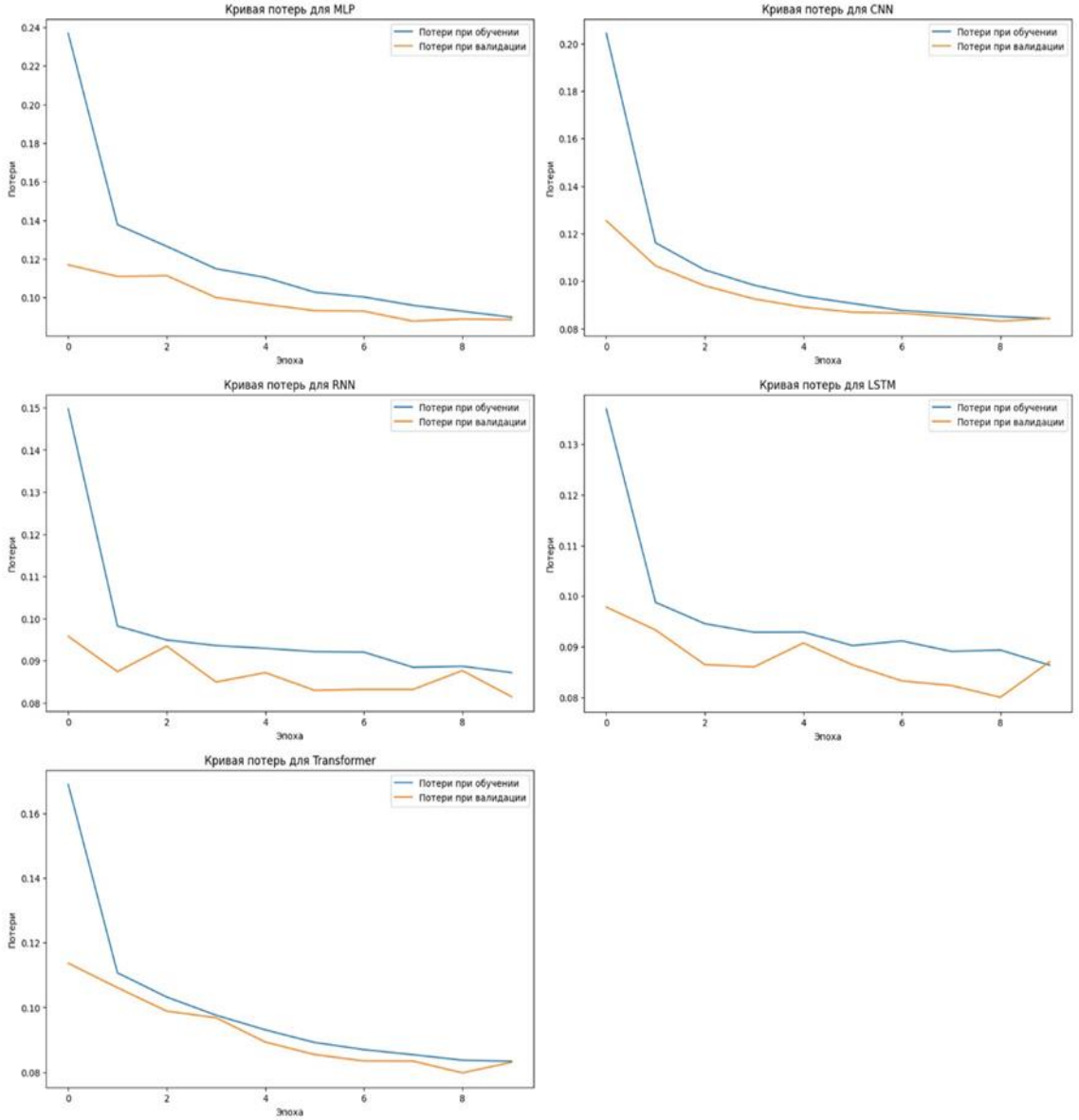

Рисунок 1

Источник: составлено авторами

Таблица 2.

Архитектура	*MSE*	*MAE*	*RMSE*	R^2
10 эпох *Predict Conversion in Digital Marketing Dataset*				
MLP	0,0953	0,2302	0,3088	0,1051
CNN	0,0889	0,1903	0,2982	0,1657
RNN	0,0843	0,1792	0,2903	0,2091
LSTM	0,0883	0,1491	0,2972	0,1708
Transformer	0,0864	0,1934	0,2939	0,1893
100 эпох *Predict Conversion in Digital Marketing Dataset*				

Архитектура	*MSE*	*MAE*	*RMSE*	R^2
MLP	0,0989	0,2514	0,3144	0,0720
CNN	0.0821	0,2002	0,2865	0,2297
RNN	0,0826	0,1527	0,2874	0,2248
LSTM	0,0805	0,1360	0,2837	0,2444
Transformer	0,0820	0,1980	0,2864	0,2303

Источник: составлено авторами

На рис. 1 предоставлены кривые потерь для каждой из архитектур при количестве эпох, равном 10. На них никаких явных признаков переобучения или недообучения не наблюдается. Обучение является оптимальным, ведь сократились потери как при обучении, так и при валидации.

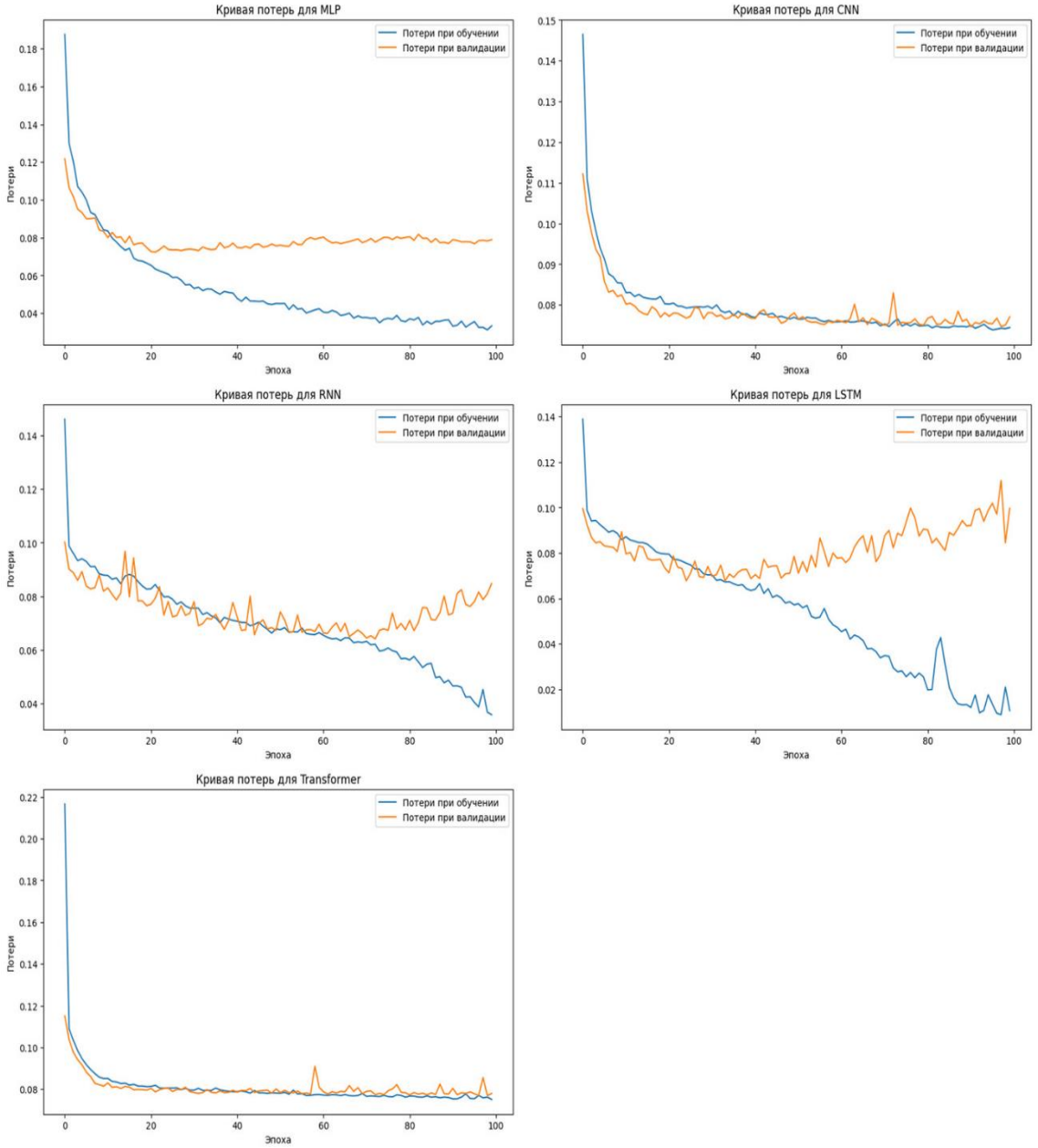

Рисунок 2

Источник: составлено авторами

Для 100 эпох результаты изменяются. *RNN* и *LSTM* все еще показывают хорошие результаты, однако ценность R^2 (0,2444 для *LSTM*) увеличивается, что указывает на лучшее объяснение изменений целевой переменной. *CNN* на 100 эпохах по-прежнему демонстрирует конкурентные значения – *MSE* (0,0821) и R^2 (0,2297) показывают, что эта архитектура все еще справляется с задачей. *MLP* демонстрирует ухудшение в производительности по сравнению с 10 эпохами, где его R^2 уменьшилось до 0,0720.

На рис. 2 отображены кривые потерь для каждой архитектуры при 100 эпохах. По ним можно заметить, что возможно переобучение.

Переобучение – это явление, когда модель слишком хорошо учится на обучающей выборке и не способна обобщать свои знания на новые данные [16]. Это проявляется в том, что потери на обучающей выборке продолжают уменьшаться, а потери на валидационной выборке начинают увеличиваться. Другими словами, модель «запоминает» обучающие данные, но не может правильно предсказывать новые данные.

Результаты, полученные при обучении на наборе данных *Global Superstore Transaction Data* (2011-2014), предоставлены в табл. 3.

Таблица 3.

Архитектура	*MSE*	*MAE*	*RMSE*	R^2
Global Superstore Transaction Data (2011-2014) 10 эпох				
MLP	90400,9178	136,6024	300,6675	0,5961
CNN	93950,4256	160,9294	306,5133	0,5802
RNN	223819,9833	262,0479	473,0962	0,0001
LSTM	223822,2828	261,9388	473,0986	0,0001
Transformer	155949,5275	224,3144	394,9045	0,3032
Global Superstore Transaction Data (2011-2014) 100 эпох				
MLP	63967,5028	104,4016	252,9180	0,7142
CNN	77402,4549	131,9994	278,2130	0,6541
RNN	76968,3010	122,2236	277,4316	0,6561
LSTM	223815,3553	262,3036	473,0913	0,0001
Transformer	424995,6801	311,6859	651,,169	0,0001

Источник: составлено авторами

На этапе обучения с 10 эпохами *MLP* демонстрирует лучшие результаты среди всех архитектур, имея минимальные значения *MSE* (90400,9178), *MAE* (136,6024) и *RMSE* (300,6675). Значение R^2 составляет 0,5961, что указывает на то, что модель объясняет почти 60% вариаций целевой переменной. *CNN* показывает чуть худшие результаты, *MSE* составила 93950,4256, а *MAE* достигла 160,9294. Несмотря на это, *CNN* все же продемонстрировала относительно приемлемую производительность. *RNN* и *LSTM* показывают значительно высокие значения *MSE* (223819,9833 и 223822,2828 соответственно), а их R^2 близко к нулю (0,0001), что свидетельствует о плохой способности этих моделей объяснять вариации в данных. *Transformer* показывает некоторое улучшение по сравнению с *RNN* и *LSTM*, с *MSE* 155949,5275, однако его производительность остается ниже *MLP*.

На рис. 3 предоставлены кривые потерь для каждой из архитектур при количестве эпох, равном 10. Для *MLP*, *CNN*, *RNN* и *LSTM* обучение является оптимальным, в то время как кривая потерь *Transformer* отражает возможность переобучения.

При обучении на 100 эпохах результаты меняются. *MLP* продолжает оставаться лучшей моделью с *MSE* 63967.5028, *MAE* 104,4016 и *RMSE* 252,9180,

что подтверждает его эффективность и позволяет предположить, что модель значительно улучшила свои показатели по сравнению с 10 эпохами. R^2 в этом случае составило 0,7142, что говорит о том, что модель теперь объясняет 71% вариаций. *CNN* также показывает улучшение, с *MSE* 77402,4549 и *MAE* 131,9994, но по-прежнему находится на втором месте после *MLP*. *RNN* демонстрирует стойкое улучшение с *MSE* 76968,3010 и *MAE* 122,2236, что дает надежду на то, что модель находит свое применение. *LSTM*, напротив, не демонстрирует прогресса и сохраняет высокие показатели, как и ранее. *Transformer* показывает резкое ухудшение, с *MSE* 424995,6801.

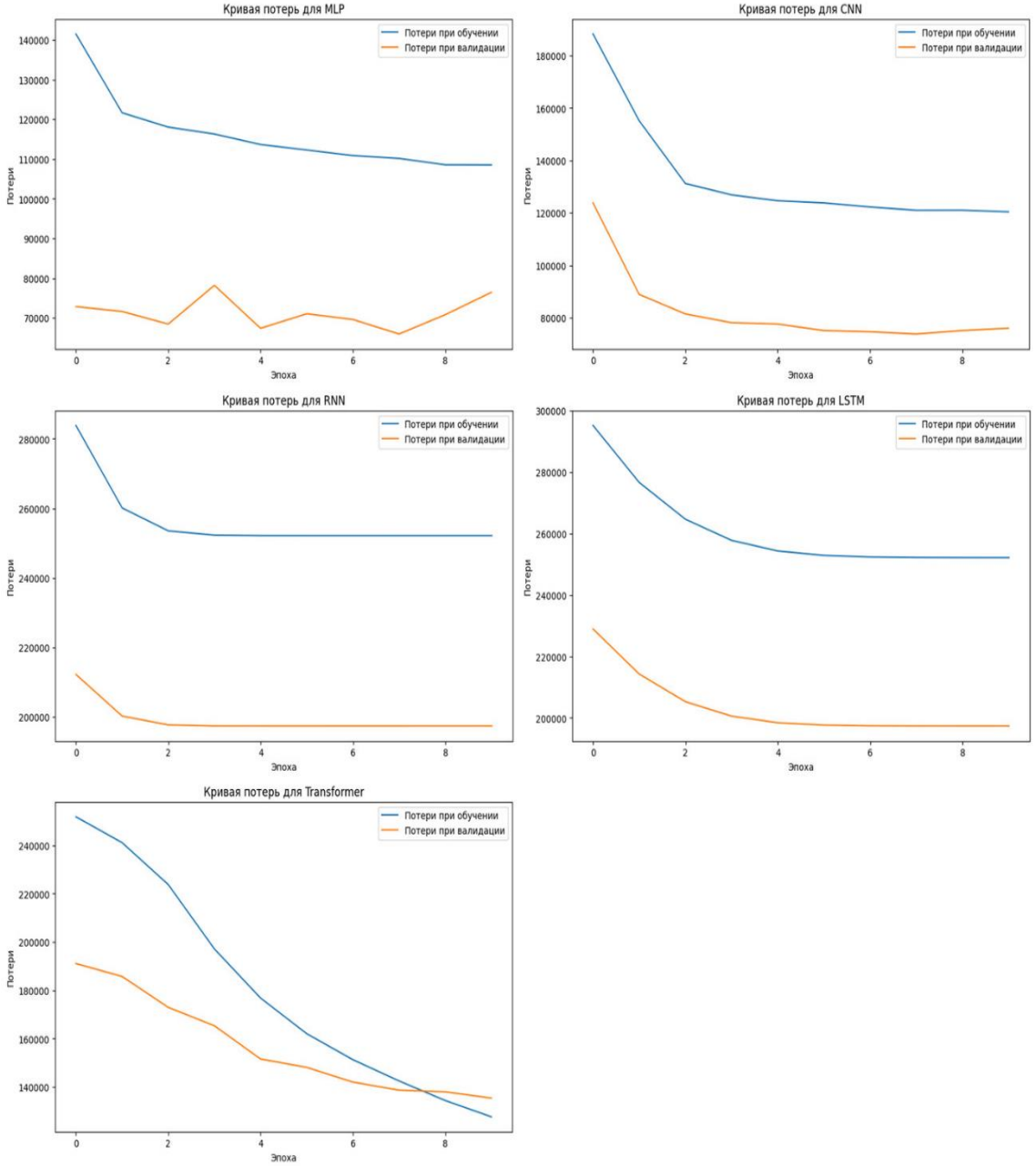

Рисунок 3

Источник: составлено авторами

На рис. 4 изображены кривые потерь при 100 эпохах. Видно, что для *Transformer* потери при валидации значительно выше, чем потери при обучении. Это явный признак переобучения. Модель не может хорошо предсказывать новые данные, так как слишком специализируется на данных из обучающей выборки. Для остальных архитектур обучение можно назвать оптимальным.

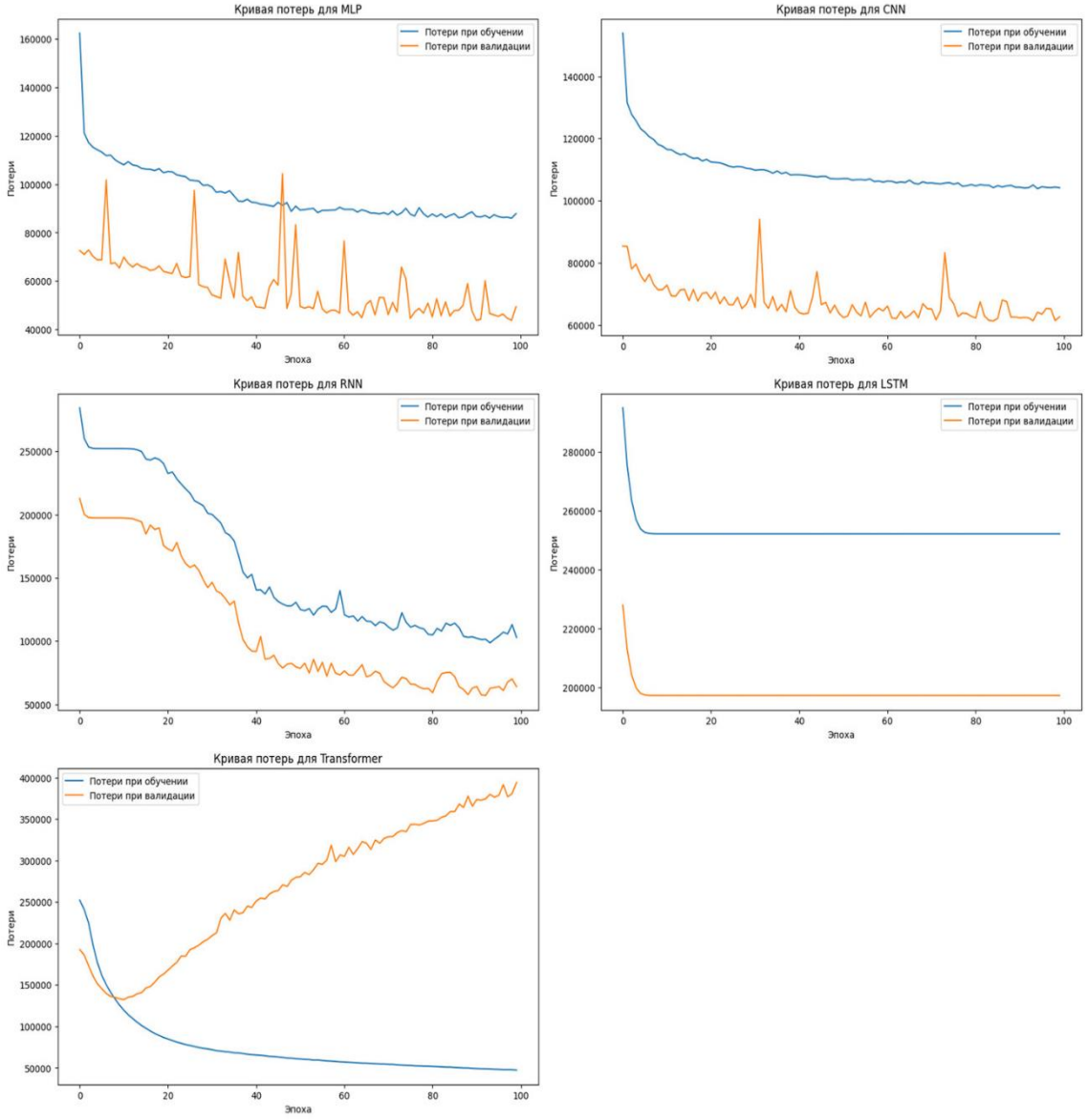

Рисунок 4

Источник: составлено авторами

Заключение

Проведенный анализ различных архитектур нейронных сетей для задач регрессии на двух разных наборах данных подтверждает, что решение задач регрессии требует гибкости и адаптивности как со стороны методологии, так и со стороны выбора модели. Оптимизация на основе характеристик конкретного набора данных может привести к значительным улучшениям в качестве модели.

Результаты показывают, что выбор архитектуры нейронной сети существенно влияет на качество предсказаний. В случае набора данных *Global Superstore Transaction Data* наилучшей моделью оказалась *MLP*. Для набора данных *Predict Conversion in Digital Marketing Dataset*, в свою очередь, наилучшие результаты показала *RNN*, что подчеркивает, как разные архитектуры могут эффективно работать в зависимости от задачи.

Также можно отметить, что увеличение количества эпох в основном способствовало улучшению результатов для *MLP*, в то время как *RNN* показала относительное превосходство на 100 эпохах в другом наборе данных. Тем не менее, некоторые архитектуры, такие как *LSTM* и *Transformer*, продемонстрировали слабые результаты, особенно в задачах с небольшими количествами эпох, а также не проявили значительных улучшений при увеличении эпох.

Анализ подчеркивает важность тщательного выбора и настройки архитектуры нейронной сети в зависимости от специфики задачи и доступных данных. Использование различных подходов к предобработке данных и экспериментирование с настройками гиперпараметров также играют критическую роль в повышении производительности моделей.

Литература

1. Ярушкина Н.Г. Интеллектуальный анализ временных рядов: учебное пособие / Н.Г. Ярушкина, Т.В., Афанасьева., И.Г. Перфильева. – М.: ИД «ФОРУМ»: ИНФРА-М, 2012. – 160 с.
2. Peng H. Application of Artificial Intelligence in Computer Network Technology in the Big Data Era. Electronic Technology and Software Engineering, 2018. – № 14 (20). – С. 169-172.
3. Горбуля Д.С. Вариант применения нейросетей для прогнозирования изменения параметров качества арендуемых информационных потоков // Экономика и качество систем связи, 2024. – № 1 (31). – С. 90-98.
4. Саймон Хайкин. Нейронные сети: полный курс = Neural Networks: A Comprehensive Foundation. 2-е изд. – М.: Вильямс, 2006.
5. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы; [пер. с польск. И.Д. Рудинского]. – М.: Горячая линия – Телеком, 2006. – 452 с.
6. Круглов В.В., Дли М.И., Годунов Р.Ю. Нечеткая логика и искусственные нейронные сети. – М.: Физматлит, 2001. – 224 с.
7. Coinspot: электрон. ресурс. Mirocana – нейронная сеть для торговли на крипторынке. 2017 URL: https://coinspot.io/technology/mirocana-nejronnaya-set-dlyatorgovli-na-kriptorynke/ (дата обращения 10.01.2024).
8. Ma Y.H. Analysis of the application of artificial intelligence in computer network technology. Contemporary Educational Science, 2015. – № 11 (20). – С.77-79.
9. Yuan Q.Ch. Artificial Intelligence and Its Application in Computer Network Technology. Computer Programming Skills and Maintenance, 2016. – № 12 (13). – С.70-73.
10. URL https://data.world/asepetruk/global-superstore (дата обращения – октябрь 2024 г.).
11. URL https://www.kaggle.com/datasets/rabieelkharoua/predict-conversion-in-digital-marketing-dataset (дата обращения – октябрь 2024 г.).
12. Xu W.T. Artificial Intelligence and Its Application in Computer Network Technology. Information and Computer (Theoretical Edition), 2017. – № 23. – С.146-147 (2017). (in Chinese).

13. Meyer Michael D. (January 2007). «Artificial Intelligence in Transportation Information for Application». Transportation Research Circular.
14. Pan Y. Application analysis of artificial intelligence in computer network technology. Computer fan, 2018. – № 12 (11) – C. 22-24.
15. Nami M.R., Bertels K. A survey of autonomic computing systems, in: Third international conference on autonomic and autonomous systems (ICAS'07), IEEE, 2007. – pp. 26-26.
16. Catalao J.P.S., Mariano S.J.P.S., Mendes V.M.F., Ferreira L.A.F.M. Short-Term Electricity Prices Forecasting In A Competitive Market: A Neural Network Approach. Electric Power Systems Research. 2007. – Vol. 77. – P. 1297-1304.