ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 131 за 2019 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Чащин А.В., Бочев М.А.,
Оселедец И.В., Овчинников Г.В.

Предсказание эволюции динамических систем остаточными нейронными сетями

Рекомендуемая форма библиографической ссылки: Предсказание эволюции динамических систем остаточными нейронными сетями / А.В.Чащин [и др.] // Препринты ИПМ им. М.В.Келдыша. 2019. № 131. 26 с. http://doi.org/10.20948/prepr-2019-131
URL: http://library.keldysh.ru/preprint.asp?id=2019-131

ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. КЕЛДЫША
Российской академии наук

А.В. Чащин, М.А. Бочев , И.В. Оселедец, Г.В. Овчинников

Предсказание эволюции динамических систем остаточными нейронными сетями

Москва — 2019

Чащин А.В., Бочев М.А., Оселедец И.В., Овчинников Г.В.

Предсказание эволюции динамических систем остаточными нейронными сетями

Предсказание временных рядов и зависящих от времени данных является важной задачей, возникающей во многих приложениях. Типичным примером является решение систем обыкновенных дифференциальных уравнений (ОДУ) $\dot{x} = F(x)$. Часто функция правой части $F(x)$ в явном виде неизвестна, а система ОДУ описывается значениями решения в разные моменты времени. В этом случае использование стандартных пакетов для решения систем ОДУ невозможно. В препринте предлагается основанный на данных подход для обучения нейросетей эволюции динамических систем. Показано, как тренировкой остаточных нейросетей (ResNet) на образцах решения можно построить модели для предсказания эволюции решения во времени. Тесты предложенного подхода на трёх системах ОДУ демонстрируют, что нейросетевые модели в состоянии качественно верно предсказывать динамику систем. Кроме того, предсказываемые решения устойчивы на более длинных временных интервалах, чем в других известных из литературы подходах.

Ключевые слова: динамические системы, нейронные сети, остаточные нейросети, глубокое обучение

Artem Chashchin, Mikhail Botchev, Ivan Oseledets, George Ovchinnikov

Predicting dynamical system evolution with residual neural networks

Forecasting time series and time-dependent data is a common problem in many applications. One typical example is solving ordinary differential equation (ODE) systems $\dot{x} = F(x)$. Oftentimes the right hand side function $F(x)$ is not known explicitly and the ODE system is described by solution samples taken at some time points. Hence, ODE solvers cannot be used. In this paper, a data-driven approach to learning the evolution of dynamical systems is considered. We show how by training neural networks with ResNet-like architecture on the solution samples, models can be developed to predict the ODE system solution further in time. By evaluating the proposed approaches on three test ODE systems, we demonstrate that the neural network models are able to reproduce the main dynamics of the systems qualitatively well. Moreover, the predicted solution remains stable for much longer times than for other currently known models.

Key words: dynamical systems, residual networks, deep learning

Работа третьего автора выполнена при поддержке гранта РФФИ №18-31-20069-мол-а-вед.

1. Введение

Нейросетевые методы становятся важным инструментом анализа зависящих от времени массивов данных и многомерных временных рядов. Типичной задачей является реконструкция решений системы обыкновенных дифференциальных уравнений (ОДУ) $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$ путём аппроксимации её правой части $F(\boldsymbol{x})$ соответствующей нейронной сетью. В работе [31] для такой задачи представлены интересные результаты, полученные с помощью нейросети с одним скрытым слоем (shallow network), обученной функции $F(\boldsymbol{x})$ и затем трансформированной в рекуррентную нейросеть (recurrent neural network, RNN).

В [2] данная проблема формулируется в качестве задачи разреженной регрессии. $F(\boldsymbol{x})$ представляется в виде линейной комбинации функций из определённого множества, и по известным значениям $\boldsymbol{x}$ и $\dot{\boldsymbol{x}}$ (или по значениям $\boldsymbol{x}$ и оценкам $\dot{\boldsymbol{x}}$) с помощью алгоритмов sequential thresholded least squares или LASSO (least absolute shrinkage and selection operator) восстанавливаются коэффициенты разложения. Авторы отмечают, что в случае динамических систем с дискретным временем их подход напоминает разложение по динамическим модам (Dynamic Mode Decomposition, DMD) [32].

Однако в прикладных задачах правая часть $F(\boldsymbol{x})$ может быть не известна явно, а сама система может описываться выборкой из значений решения в нескольких точках. В таком случае подходы [31, 2] неприменимы, но можно воспользоваться методами машинного обучения, которые позволяют строить математические модели по данным. Возможным решением данной проблемы будет построение модели, рассматривающей данные в качестве временного ряда, и её обучение восстановлению уравнения или предсказанию эволюции системы через время $\Delta t > 0$ после текущего момента t. Если во втором случае рекурсивно применить модель к данным k раз, то мы ожидаем предсказание эволюции на k шагов по времени вперёд в интервале $[t, t + k\Delta t]$.

Как при восстановлении $F(\boldsymbol{x})$, так и при предсказывании решения ОДУ возникает вопрос точности аппроксимации нейронными сетями.

В частности, если система хаотическая, будет сложно добиться малых ошибок для долгосрочных предсказаний сетей ввиду проблем с устойчивостью: даже изначально малые различия между исходными данными и предсказанием заметно вырастают со временем. В таком случае будет разумным потребовать от нейросетевой аппроксимации запоминания хотя бы основной динамики системы, чтобы общий вид предсказанных траекторий не так сильно отличался от настоящего решения.

В [22] представлены обещающие результаты предсказаний на умеренные времена путём восстановления правой части системы $F(\boldsymbol{x})$ по алгоритму. В случае одномерных и двумерных задач при долгосрочном прогнозе наблюдается благоприятное поведение ошибки предсказания. Однако в случае трёхмерной си-

стемы Лоренца с хаотической динамикой предсказанные и исходные траектории начинают заметно отличаться даже на относительно малых временах $t \approx 1$.

Теперь рассмотрим второй подход, когда решение ОДУ предсказывается без предположений насчёт правой части системы $F(\boldsymbol{x})$. Его можно применять в случаях, когда явное восстановление уравнений не является основной целью и важнее просто получить предсказание решения. На выходе аппроксимирующей модели мы получаем предсказание $\tilde{\boldsymbol{x}}(t + \Delta t)$ решения ОДУ $\boldsymbol{x}(t + \Delta t)$ в виде $\tilde{\boldsymbol{x}}(t + \Delta t) = g(\boldsymbol{x}(t), \boldsymbol{\theta})$, где $\boldsymbol{\theta}$ — вектор параметров модели, а $\boldsymbol{x}(t)$ — входные данные (т.е. состояние системы в момент t). Функция g определяется выбором конкретной архитектуры для аппроксимирующей модели. Одним из известных подходов является разложение по динамическим модам (Dynamic Mode Decomposition, DMD), в рамках которого ищется такая матрица эволюции системы A, что $\tilde{\boldsymbol{x}}(t + \Delta t) = A\boldsymbol{x}(t)$. После формулировки метода в [28] были предложены его различные модификации, которые, например, предлагают другой способ обработки входных данных [17] или применяют теорию оператора Купмана для получения более точных аппроксимаций [36]. Ещё одним примером служит использование резервуарных вычислений (reservoir computing) в [24, 20] для подсчёта экспонент Ляпунова нескольких хаотических процессов по выборке значений решения. Полученная модель затем используется для воспроизведения динамики системы. Для уравнений из системы Лоренца предсказание, полученное на основе такого подхода, начинает отклоняться от истинной траектории после $t \approx 7$, но авторы статьи ничего не сообщают об ошибках предсказания [24, 20]. Также следует упомянуть [34], где для предсказания будущих состояний хаотических систем высокой размерности применяются сети долгой краткосрочной памяти (long short-term memory network, LSTM).

В данном препринте мы рассматриваем другую архитектуру, а именно остаточные нейронные сети (residual network, ResNet), применительно к задаче предсказания эволюции системы нейросетевыми моделями. Известно, что остаточные сети помогают в решении проблемы исчезающего и взрывного градиента и успешно применяются к задачам классификации изображений. Мы показываем их работу на трёх системах ОДУ, две из которых хаотические, и сравниваем с результатом, полученным обычной сетью прямого распространения.

В [25] также рассматривается подход, тесно связанный с архитектурой ResNet. Авторы статьи исследуют систему Лоренца, но не сообщают численных результатов. Отличие от нашей постановки эксперимента заключается в том, что их начальные условия системы выбираются на аттракторе, что сильно упрощает задачу для нейронных сетей.

Препринт построен следующим образом. В разделе 2 даётся краткий обзор теории нейронных сетей, в том числе приводится информация об остаточных сетях, и описываются архитектуры, используемые нами для предсказания эволю-

ции систем. В разделе 3 описываются рассматриваемые динамические системы. В разделе 4 представлены детали проведённых экспериментов, а их результаты описаны в разделе 5. В разделе 6 подводятся итоги и обсуждаются направления дальнейшей работы.

2. Основные концепции нейронных сетей

В данном разделе мы рассказываем о нейронных сетях, обсуждаем основные принципы их построения и рассматриваем их особый тип, остаточные сети, который используется в наших экспериментах.

2.1. Архитектура нейронных сетей. Несмотря на то что нейронные сети являются фундаментальным концептом глубокого обучения, им довольно сложно дать определение, причём сделать это можно разными способами. Как правило, они описываются основными свойствами и структурой. В данной статье мы рассматриваем *нейронные сети прямого распространения (feedforward neural networks)* или *многослойные перцептроны (multilayer perceptrons, MLPs)*. Согласно [9], нейронные сети прямого распространения — это модели, строящие аппроксимацию $g(\boldsymbol{x}, \theta)$ некоторой функции $\bar{g}(\boldsymbol{x})$ и выучивающие значения параметров θ, дающие наилучшую аппроксимацию. В настоящее время эти модели, придуманные по аналогии с человеческим мозгом, широко используются в различных областях науки и промышленности. Их структура выглядит следующим образом.

Основным элементом сети прямого распространения является *нейрон*. Он описывается набором весов $\boldsymbol{w} = (w_0, w_1, \ldots, w_N)^T$. Нейрон принимает на вход N-мерный вектор $\boldsymbol{x} \in \mathbb{R}^N$, вычисляет линейную комбинацию его компонент и применяет к ней некоторую функцию $G : \mathbb{R} \to \mathbb{R}$. Таким образом, на выходе нейрона получается число $y = G(w_0 + w_1 x_1 + \ldots + w_N x_N) = G(\boldsymbol{w}^T \hat{\boldsymbol{x}})$, где $\hat{\boldsymbol{x}} = (1, x_1, \ldots, x_N)^T \in \mathbb{R}^{N+1}$. На протяжении статьи мы будем использовать обозначения с шапкой для векторов с добавленной в начало единичной компонентой.

Нейроны группируются в слои. *Полносвязный (или плотный)* слой из K нейронов принимает на вход такой же вектор $\boldsymbol{x}$ и вычисляет K выходов отдельных нейронов: $\boldsymbol{y} = (G(\boldsymbol{w}^{1T}\hat{\boldsymbol{x}}), \ldots, G(\boldsymbol{w}^{KT}\hat{\boldsymbol{x}}))^T$, где $\boldsymbol{w}^i = (w_0^i, w_1^i, \ldots, w_N^i)^T$, $i = 1, \ldots, K$, - веса i-го нейрона. G называется *функцией активации*. Она одинакова для всех нейронов слоя, в то время как их веса независимы друг от друга. Для удобства предыдущее выражение можно переписать в виде $\boldsymbol{y} = G(W\hat{\boldsymbol{x}})$, где $W = (\boldsymbol{w}^1|\boldsymbol{w}^2|\ldots|\boldsymbol{w}^K)^T \in \mathbb{R}^{K\times(N+1)}$ — матрица с весами нейронов слоя, а G применяется покомпонентно.

Слои нейронных сетей также могут использовать другие операции преобразования входных данных: свёртка, пулинг и т.д. Ещё одна разновидность

слоёв, используемых в нашей работе, а именно слой пакетной нормировки, будет обсуждаться в следующем подразделе.

Наконец, слои нейронов объединяются в нейронную сеть. Важным свойством нейронной сети прямого распространения является отсутствие циклов, т.е. выходные данные каждого слоя не подаются ему на вход [9]. Для примера рассмотрим сеть прямого распространения с L полносвязными слоями. Её первый слой принимает на вход $\boldsymbol{x} \in \mathbb{R}^N$ и вычисляет вектор $\boldsymbol{G}_1 = G_1(W_1\hat{\boldsymbol{x}})$. Второй слой принимает выход первого слоя $\boldsymbol{G}_1$ себе на вход и возвращает $\boldsymbol{G}_2 = G_2(W_2\hat{\boldsymbol{G}}_1)$.

Процесс рекурсивно продолжается в виде $\boldsymbol{G}_{i+1} = G_{i+1}(W_{i+1}\hat{\boldsymbol{G}}_i)$ до последнего слоя, выход которого $\boldsymbol{y} = G_L(W_L\hat{\boldsymbol{G}}_{L-1})$ считается выходом сети.

В данном случае параметрами θ, которые надо подобрать для получения наилучшей аппроксимации $\bar{g}$, являются веса слоёв сети $W_1, \ldots, W_L$, количество слоёв и их размер, функции активации и т.д. Под архитектурой сети подразумевают определённый набор её параметров за исключением весов. Она выбирается в зависимости от решаемой задачи. Чем масштабнее задача, тем больше слоёв используется и тем больше размер каждого слоя. Активации G_i обычно выбираются нелинейными, чтобы сеть могла аппроксимировать более сложные функции.

Свойства активаций детально обсуждаются в [6, 23]. Из наиболее широко используемых функций можно отметить следующие:

- сигмоида $G(x) = 1/(1 + e^{-x})$;
- гиперболический тангенс $G(x) = \tanh x = (e^x - e^{-x})/(e^x + e^{-x})$;
- усечённое линейное преобразование (rectified linear unit, ReLU) $G(x) = \max(0, x)$.

Способность нейронных сетей аппроксимировать широкий класс функций $\bar{g}(\boldsymbol{x})$ подтверждается *универсальной аппроксимационной теоремой*. Она встречается в нескольких формулировках и обобщениях [13, 5, 12, 21], но основной результат, необходимый для данной статьи, заключается в том, что нейронные сети прямого распространения с конечным числом нейронов могут аппроксимировать непрерывные функции на компактном носителе в $\mathbb{R}^N$ с произвольной точностью.

2.2. Обучение и переобучение. Как только архитектура сети зафиксирована, её веса подбираются с помощью методов оптимизации. Строится *функция потерь*, которая показывает, насколько нейросетевая аппроксимация g с весами W «близка» к моделируемой функции $\bar{g}$, на основе их значений в точках из множества $X = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^M\}$. Типичным примером такой функции является среднеквадратичная ошибка (mean squared error, MSE) $\mathcal{L} = \frac{1}{M}\sum\limits_{i=1}^{M}\|g(\boldsymbol{x}^i, \theta) - \bar{g}(\boldsymbol{x}^i)\|^2$, где параметры θ включают в себя веса W. Процесс подбора весов нейронной сети

путём минимизации функции потерь по W называется *обучением* сети. Нейронные сети обучаются методами стохастического градиентного спуска (stochastic gradient descent), пакетного градиентного спуска (batch gradient descent) и их модификаций [1, 15]. На каждом шаге такие алгоритмы оценивают градиент функции потерь либо по всей выборке сразу, либо по её частям (*мини-пакетам, mini-batches*), либо по одному её элементу. Обновление весов происходит согласно алгоритму обратного распространения ошибки: ввиду особой структуры функции g существует эффективный способ подсчёта градиентов $\mathcal{L}$ с помощью цепного правила дифференцирования сложной функции [9].

Множество элементов выборки X_{train}, которое используется для оценки функции потерь во время обучения, называется *обучающей выборкой*. Может случиться так, что обученная нейронная сеть показывает высокую точность на X_{train}, но её качество работы заметно снижается на других элементах, не включённых в X_{train}. Это происходит потому, что сеть научилась лишь отображать входные данные из обучающей выборки в соответствующие выходные, но не приобрела способности к обобщению, т.е. к качественной работе на данных, которые не видела ранее. Такое явление называется *переобучением* [3, 10].

Его можно наблюдать на примере полиномиальной регрессии. Если взять большую степень полинома n, то в известных точках его значения будут близки к истинным, но в целом точность будет достаточно низкая. С другой стороны, уменьшение степени приведёт к снижению точности на известной выборке, но повысится общее качество аппроксимации.

Возможным решением этой проблемы будет оценка функции потерь на другой, *тестовой*, выборке X_{test} из элементов, не представленных в X_{train}. Наблюдение за значениями $\mathcal{L}(W)$ на тестовой выборке во время обучения позволяет обнаружить момент переобучения, заранее остановить обучение сети и выбрать веса с предыдущих итераций (*эпох*), дающие наименьшие потери. Такой метод называется *ранней остановкой* [3, 26, 37]. Соотношение между обучающей и тестовой выборками эвристически выбирают как $80/20$ или $70/30$. Иногда для итоговой оценки производительности модели также используют дополнительную, *отложенную*, выборку, а во время обучения следят за ошибкой на тестовой выборке.

В общем случае способы борьбы с переобучением называются *регуляризацией*. Она определяется в [9] как «любая модификация алгоритма обучения, предпринятая с целью уменьшить его ошибку обобщения, не уменьшая ошибку обучения». Такие модификации могут включать внесение ограничений на значения параметров и добавление дополнительных слагаемых в функцию потерь. В

данной статье мы используем вторую стратегию. Используя функцию потерь

$$\mathcal{L}_{\mathrm{L2}}(W) = \mathcal{L}(W) + \lambda \sum_{i=1}^{L} \sum_{j=0}^{K_i} \|W_i^j\|^2,$$

где λ — небольшая константа (например, `1e-8`), L — число слоёв, K_i — размер i-го слоя, мы штрафуем модель за сложность и накладываем ограничения на веса сети, делая их «не слишком большими». Такой подход называется L_2 *регуляризацией* (также известна как регуляризация Тихонова). Он часто приводит к упрощению модели и улучшению обобщающих свойств.

Ещё один метод, который используется нами в экспериментах, называется *пакетной нормировкой* (*batch normalization*). В [14] она предлагается как решение проблемы ковариационного сдвига. Предположим, что нейронная сеть обучена различать кошек и собак на картинках, но выборка изображений с котами несбалансирована: в ней содержатся только фотографии чёрных котов. После обучения на таких данных качество работы сети на котах других окрасов будет плохим. Другими словами, распределения данных в обучающей и тестовой выборках отличаются. Такое явление называют *проблемой ковариационного сдвига* (*the covariate shift problem*).

Логичным решением проблемы было бы создание более сбалансированной выборки путём добавления изображений с котами разных окрасов. Однако это может не помочь, поскольку поменяется только распределение данных входного слоя. В скрытых (промежуточных) слоях по-прежнему может сохраниться внутренний ковариационный сдвиг, потому что входные данные для этих слоёв резко меняются с каждым обновлением значений весов. Это приводит к медленному обучению сетей и необходимости использования маленького коффициента скорости обучения (learning rate).

Известное решение данной проблемы, пакетная нормировка, нормирует вход каждого слоя по мини-пакету, тем самым снижая внутренний ковариационный сдвиг. Для мини-пакета $B = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \subset \mathbb{R}^N$ вычисляются среднее значение $\boldsymbol{\mu}_B = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{x}_i$ и стандартное отклонение $\boldsymbol{\sigma}_B^2 = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{x}_i - \boldsymbol{\mu}_B)^2$. Затем происходит нормировка мини-пакета $\tilde{\boldsymbol{x}}_i = (\boldsymbol{x}_i - \boldsymbol{\mu}_B)/\sqrt{\boldsymbol{\sigma}_B^2 + \varepsilon}$, где ε — маленькая константа, а операции деления и взятия корня выполняются поэлементно.

Наконец, мы применяем операции масштабирования и сдвига для получения выходных значений слоя $\boldsymbol{y}_i = \boldsymbol{\gamma}\tilde{\boldsymbol{x}}_i + \boldsymbol{\beta}$, где $\boldsymbol{\gamma} \in \mathbb{R}^N$ и $\boldsymbol{\beta} \in \mathbb{R}^N$ — выучиваемые параметры, а умножение также выполняется покомпонентно. Это приводит к перепараметризации выходных значений слоёв глубоких нейронных сетей, благодаря чему снижается влияние операций с предыдущих слоёв на выход с текущего и уменьшается внутренний ковариационный сдвиг. Пакетная

нормировка также выступает в качестве небольшой регуляризации для модели, поскольку нормировка по мини-пакету добавляет шум в выходные данные.

2.3. Остаточные сети. В данном подразделе мы рассматриваем остаточные нейронные сети (residual networks, ResNets). Эта архитектура используется нами для выучивания динамики систем ОДУ. Такой тип сетей был придуман, чтобы бороться с проблемой исчезающего и взрывного градиента, встречающейся в очень глубоких архитектурах. Если сеть состоит из большого числа скрытых слоёв, градиенты, вычисленные методом обратного распространения ошибки, могут быть очень большими или очень маленькими. Это, в свою очередь, приводит к неэффективным обновлениям значений весов на каждом шаге оптимизации и к сложностям в нахождении их оптимальных значений [7]. Для преодоления этой проблемы [11] предлагает заменить часть умножений в формулах выходов слоёв на сложения. Основная идея состоит в использовании «блоков» слоёв, которые будут выучивать разницу между входными данными и желаемым выходом, а потом прибавят эту разницу ко входу для получения нужного результата.

Рассмотрим один такой блок со следующей структурой:

$$FC \to BN \to \mathit{ReLU} \to FC, \tag{1}$$

где FC — полносвязный слой, BN означает пакетную нормировку, а ReLU — функция усечённого линейного преобразования, применённая покомпонентно. Если обозначить за $\boldsymbol{x}_n$ входные данные блока, а за $G(\boldsymbol{x}_n)$ его выход (то есть выход второго полносвязного слоя), то суммарный выход определяется как

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + G(\boldsymbol{x}_n).$$

Этот результат затем передаётся следующему блоку с такой же структурой. Таким образом, выход нейронной сети, состоящей из M блоков, можно записать как

$$\boldsymbol{x}_M = \boldsymbol{x}_1 + \sum_{i=1}^{M-1} G(\boldsymbol{x}_i).$$

Это выражение определяет нашу нейросетевую аппроксимацию эволюции дискретизованной динамической системы

$$\tilde{\boldsymbol{x}}(t + \Delta t) = g(\boldsymbol{x}(t), \theta),$$

где $\boldsymbol{x}_1 = \boldsymbol{x}(t)$, $\boldsymbol{x}_M = \tilde{\boldsymbol{x}}(t + \Delta t)$, а θ включает в себя вышеупомянутую архитектуру сети и веса её слоёв. Как говорилось ранее, такой подход снижает количество умножений, необходимых для обучения сети и вычисления градиентов функции потерь, в результате чего снижается вероятность исчезновения или взрыва градиентов.

Несмотря на то что в исходной статье про архитектуру ResNet не упоминаются численные методы, другие авторы [35, 19, 4] справедливо заметили, что структура сети напоминает явный метод Эйлера решения дифференциальных уравнений. Действительно, решение задачи

$$\dot{\boldsymbol{x}}(t) = F(\boldsymbol{x}(t)), \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_0,$$

может быть найдено с помощью явной схемы Эйлера в моменты времени $t_n = t_0 + nh$ итеративно в виде

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + hF(\boldsymbol{x}_n),$$

где $h > 0$ — размер шага сетки. В этом заключается наша мотивация применения ResNet к запоминанию динамики систем ОДУ. На основе этой идеи мы применяем несколько архитектур нейронных сетей. Они различаются числом слоёв в блоке, размером каждого слоя и глубиной сети. Стоит отметить, что мы не ставили целью найти оптимальную архитектуру, которая привела бы к наименьшей ошибке прогнозирования, а хотели показать качество работы разных сетей и их возможность предсказывать эволюцию. Для экспериментов использовались следующие четыре архитектуры:

- **RN1**: наиболее простая из архитектур. Она состоит из четырёх блоков ResNet, каждый из которых имеет структуру (1). Выход каждого блока суммируется с его входными данными. Размер полносвязных слоёв — 10.
- **RN2**: имеет такую же структуру, как RN1 (ср. (1)), но размер каждого полносвязного слоя увеличен до 50.
- **RN3**: более глубокая сеть с шестью блоками ResNet и размером слоя FC, равным 50.
- **RN4**: сеть с более глубокой структурой блока ResNet: $FC \to BN \to ReLU \to FC \to BN \to ReLU \to FC$. Как и первые две сети, она состоит из четырёх блоков, но размер полносвязного слоя равен 15.

Мы также сравниваем качество работы ResNet с архитектурой MLP $FC \to BN \to ReLU \to FC \to BN \to ReLU \to FC \to BN \to ReLU \to FC \to Sigmoid$ на одной из систем ОДУ. Для всех описанных архитектур входные данные нормируются так, чтобы они лежали в отрезке $[0, 1]$.

3. Динамические системы

В этом разделе описываются динамические системы, на которых сравниваются нейронные сети. Это осциллятор Ван дер Поля, система Лоренца и система Рёсслера. Ниже даётся описание систем ОДУ вместе с их уравнениями. Также в таблице 1 приводятся параметры уравнений, которые использовались для экспериментов.

3.1. Осциллятор Ван дер Поля. Для осциллятора, описанного Б. ван дер Полем в [33], исходное уравнение записывается в виде

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0,$$

где $x(t)$ обычно означает положение точки (но в зависимости от задачи возможны и другие интерпретации), а μ — скалярный параметр, характеризующий затухание колебаний. В экспериментах мы задавали значение $\mu = 3$. Если выделить производную по времени отдельной переменной $y(t) = \dot{x}(t)$, то уравнение можно свести к системе дифференциальных уравнений первого порядка $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$, где $\boldsymbol{x}(t) = [x(t), y(t)]^T$:

$$\begin{aligned} \dot{x}(t) &= y, \\ \dot{y}(t) &= \mu(1 - x^2)y - x. \end{aligned}$$

Траектории предельного цикла для этой системы в фазовом пространстве изображены на рис. 1a.

Данная система служит примером довольно «простой» динамики, поэтому будет логичным проверить работу наших моделей сначала на такой задаче, а потом переходить к более «сложным» системам. Прежде чем подавать данные нейронной сети, мы нормируем их на максимальное по модулю значение по всем компонентам вектора, так что каждая компонента нормированного вектора лежит в отрезке $[0, 1]$.

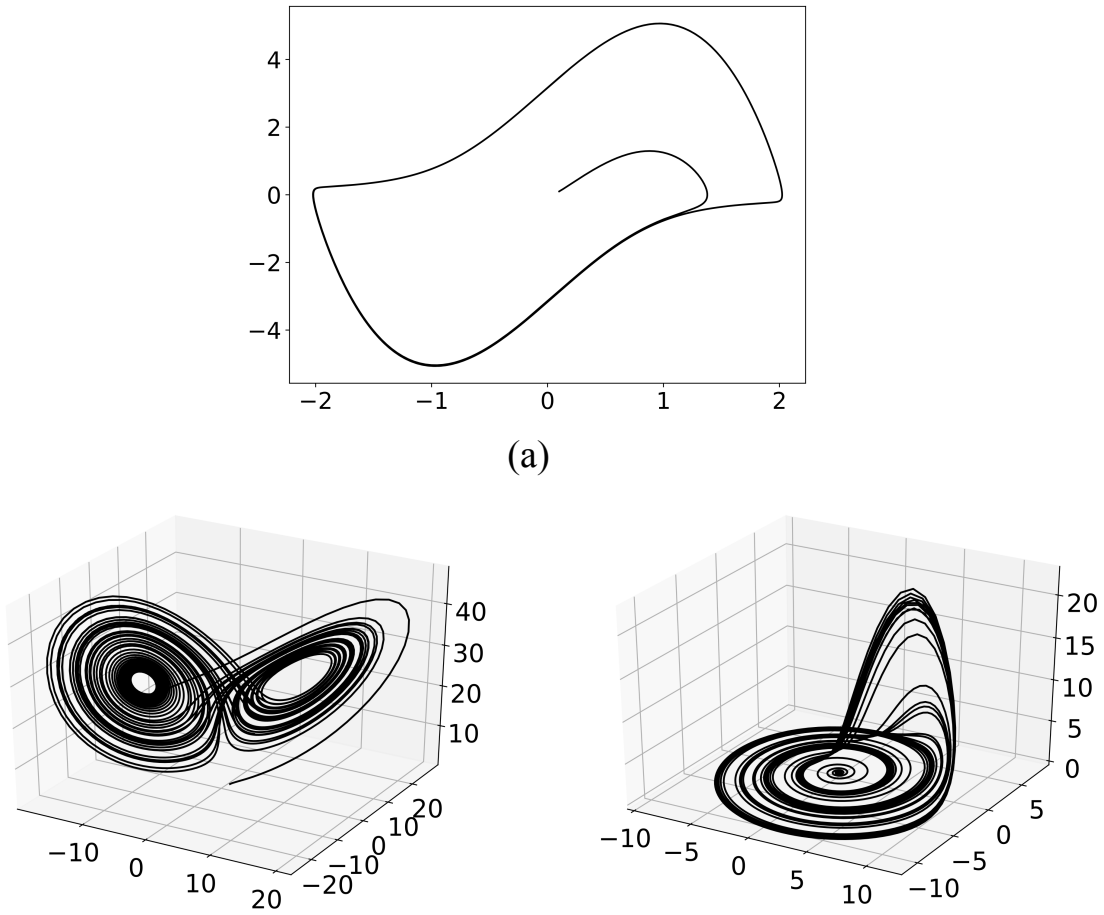
3.2. Система Лоренца. Обозначив $\boldsymbol{x}(t) = [x(t), y(t), z(t)]^T$, систему ОДУ Лоренца можно представить в виде $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$ как

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= x(\rho - z) - y, \\ \dot{z} &= xy - \beta z, \end{aligned}$$

где использовались значения параметров $\sigma = 10$, $r = 28$, $b = 8/3$. Эта система дифференциальных уравнений описана Э. Лоренцем в [18]. Она демонстрирует хаотическое поведение, а множество её решений, которое называют аттрактором, имеет форму бабочки (рис. 1b). Для этой системы мы также нормируем выборку значений решения, но способом, отличным от двух других систем. Вместо нормировки на максимальные значения выборки мы воспользуемся теоретическим результатом о границах решения из [38]. Обозначим

$$m = \sigma + \rho,$$

$$r_1 = \sqrt{\frac{-\beta m^2}{4\max\{-\sigma, -1, -\beta\}}},$$



(a)

(b) (c)

Рис. 1. Траектории решений динамических систем: (a) осциллятор Ван дер Поля, (b) система Лоренца, (c) система Рёсслера

$$r_2^2 = \begin{cases} \dfrac{\beta^2 m^2}{4\sigma(\beta-\sigma)}, & \text{if } \sigma \le 1, \sigma \le \beta/2 \\ \dfrac{\beta^2 m^2}{4(\beta-1)}, & \text{if } 1 < \sigma, 1 \le \beta/2 \\ m^2, & \text{if } \sigma > \beta/2, 1 > \beta/2 \end{cases},$$

$$R = \max\{r_1 + |m|/2, r_2\}.$$

Тогда выполняется следующий результат [38]. Если начальные условия лежат в сфере

$$\Omega = \{(x, y, z) \mid x^2 + y^2 + (z - m)^2 \le R^2\},$$

то дальнейшая траектория системы тоже целиком содержится в Ω [38]. В нашем случае $m = 38$ и $R \approx 50.0$.

3.3. Система Рёсслера. Теперь рассмотрим систему ОДУ вида $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$ из статьи [27], опубликованной в 1976 г. О. Рёсслером. Она имеет вид

$$\begin{aligned}\dot{x} &= -y - z,\\ \dot{y} &= x + ay,\\ \dot{z} &= b + z(x - c),\end{aligned}$$

где a, b, c — действительные параметры. Мы используем исходные значения параметров из статьи Рёсслера: $a = 0.2$, $b = 0.2$, $c = 5.7$. Как и в случае с системой Лоренца, система Рёсслера демонстрирует хаотическую динамику. На рис. 1c изображён её аттрактор.

Системы Лоренца и Рёсслера служат отличными примерами моделей малой размерности с достаточно сложной динамикой. Обе системы очень чувствительны к начальным условиям ввиду их хаотической природы. Поэтому предсказание их динамики будет хорошей задачей для наших нейросетевых моделей.

4. Постановка эксперимента

В экспериментах, представленных в этом разделе, мы оцениваем качество работы четырёх архитектур нейронных сетей, описанных ранее, на выборках решений трёх систем ОДУ и сравниваем результаты с работой архитектуры MLP. Сбор данных для каждой системы проходит следующим образом. Сначала мы семплируем $N = 12\,000$ точек $\boldsymbol{x}_i(0)$ из $\mathcal{N}(0, 2I)$, где $i = 1, \ldots, N$. Отметим, что в зависимости от задачи распределение берётся двумерным или трёхмерным. Затем мы решаем дифференциальные уравнения, соответствующие каждой задаче, взяв эти точки $\boldsymbol{x}_i(0)$ в качестве N начальных условий. Тем самым мы получаем $12\,000$ траекторий $\boldsymbol{x}_1(t), \ldots, \boldsymbol{x}_N(t)$. Для решения уравнений используется решатель LSODA, вызываемый функцией `odeint` библиотеки SciPy языка Python и работающий со стандартными допустимыми отклонениями `atol` = `1.49012e-8`, `rtol` = `1.49012e-8`. Решатель автоматически выбирает шаг по времени и вычисляет решение в 2500 моментах времени 0, Δt, $2\Delta t$, …, $T - \Delta t$, T, где T — параметр, задающий конечное время, а $\Delta t = T/2500$. Для каждой системы мы выбираем такое T, что траектории обходят аттрактор или предельный цикл достаточное количество раз, чтобы отразить эволюцию системы в точках выборки. Значения параметров приведены в таблице 1. Также мы приводим в ней время Ляпунова для каждой из трёх динамических систем [29, 30]. Стоит отметить, что, поскольку мы рассматриваем нехаотический случай осциллятора Ван дер Поля, его время Ляпунова формально равно бесконечности, в то время как для двух других систем оно конечно.

После получения численных данных для экспериментов мы оцениваем на них работу нейронных сетей RN1–RN4. Для каждой из $12\,000$ траекторий мы

Таблица 1. Описания и численные значения некоторых параметров

Параметр	Описание	Ван дер Поль	Лоренц	Рёсслер
—	Параметры уравнений	$\mu = 3$	$\sigma = 10$, $r = 28$, $b = 8/3$	$a = 0.2$, $b = 0.2$, $c = 5.7$
N	Количество точек	12 000	12 000	12 000
N_1	Размер обучающей выборки	8 000	8 000	8 000
N_2	Размер тестовой выборки	2 000	2 000	2 000
N_3	Размер отложенной выборки	2 000	2 000	2 000
T	Конечное время	25	25	125
T_L	Время Ляпунова	∞	1.1	14.0
$\Delta t = T/2500$	Шаг интегрирования по времени	0.01	0.01	0.05

делим выборку значений решения на пары $(\boldsymbol{x}(t), \boldsymbol{x}(t + \Delta t))$, $t = 0, \ldots, T - \Delta t$. Это означает, что для каждого $\boldsymbol{x}(t)$ нейронная сеть должна вычислить вектор $\tilde{\boldsymbol{x}}(t + \Delta t)$, близкий к решению ОДУ $\boldsymbol{x}(t + \Delta t)$.

Из всех N траекторий мы берём $N_1 = 8000$ в качестве обучающей выборки и $N_2 = 2000$ в качестве тестовой выборки. Оставшиеся $N_3 = N - N_1 - N_2 = 2000$ будут отложенной выборкой, на которой мы измерим точность предсказаний обученных моделей. Для обучения используется алгоритм Adam [16] с размером мини-пакета 2048.

В качестве функции потерь мы используем среднеквадратичную ошибку с L2 регуляризацией

$$\mathcal{L}_{\text{L2}} = \frac{1}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} \|\boldsymbol{x}_i(t_j) - \tilde{\boldsymbol{x}}_i(t_j)\|^2 + \lambda \sum_{i=1}^{L} \sum_{j=0}^{K_i} \|W_i^j\|^2,$$

где n — количество траекторий, m — количество рассматриваемых пар для каждой траектории, обозначения в слагаемом, отвечающем за регуляризацию, взяты из раздела 2.2, $\lambda = 1e - 10$. Заметим, что конкретные значения n и x_i зависят от выбора множества, на котором мы рассматриваем функцию (обучающая, тестовая или отложенная выборки); значение m зависит от шага обучения, который вводится далее в препринте.

Для ускорения обучения сети мы используем несколько приёмов. Во-первых, вместо всех N_1 траекторий используется только их часть: из всех пар $(\boldsymbol{x}(t), \boldsymbol{x}(t +$

$\Delta t)$) для обучения берётся каждая пятая. Во-вторых, мы начинаем с небольшого размера обучающей и тестовой выборок, а затем итеративно добавляем в них элементы через фиксированное число эпох обучения. В частности, обозначим множество входных элементов обучающей выборки за X_{in}, а множество ожидаемых выходов за X_{out}. В самом начале обучения оба множества пустые. Будем называть i-м шагом обучения следующую последовательность действий:

1. Добавление каждой пятой пары во множества, т.е. добавление $\boldsymbol{x}(5(i-1))$ в X_{in} и $\boldsymbol{x}(5(i-1)+1)$ в X_{out}.
2. Инициализация весов сети значениями с предыдущего шага обучения (на первом шаге веса инициализируются случайными значениями).
3. Обучение сети на парах из X_{in} и X_{out}. Оно останавливается, как только произойдёт одно из событий: сеть выполнит 5000 эпох обучения или её относительная ошибка предсказания на тестовой выборке

$$\varepsilon_{\text{i}} = \frac{1}{iN_2} \sum_{j=1}^{N_2} \sum_{k=1}^{i} \frac{\|\boldsymbol{x}_j(5(k-1)+1) - \tilde{\boldsymbol{x}}_j(5(k-1)+1)\|}{\|\boldsymbol{x}_j(5(k-1)+1)\|},$$

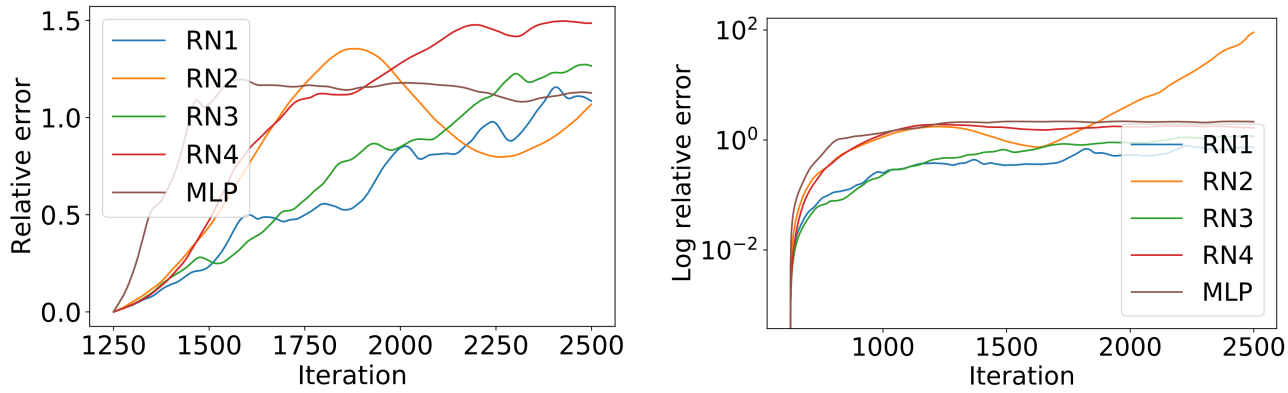
   где суммирование выполняется по всем парам из N_2 траекторий, добавленных за первые i шагов, x_j взяты из тестовой выборки, станет меньше заданного значения (в наших экспериментах это 0.0015).
4. Сохранение весов после обучения (они будут использоваться для инициализации на следующем шаге).

Такой алгоритм приводит к более эффективному использованию данных и ускоренному обучению. Стоит отметить, что аналогичные размеры обучающей и тестовой выборок (и, следовательно, аналогичную сложность вычислений) на каждом шаге можно получить, если использовать пятикратный шаг по времени $5\Delta t$ и рассмотреть все данные с траекторий вместо каждой пятой пары. Однако это не будет равносильно нашей процедуре и не даст такой же эффект, поскольку данные будут хуже описывать эволюцию системы и шаг предсказания сетью будет в пять раз больше. Критерий остановки мотивирован предварительными экспериментами с системой Лоренца: мы обнаружили, что относительная ошибка предсказания уменьшается до ≈ 0.0015 после 5000 эпох обучения, что является для нас приемлемой точностью.

Для оценки качества предсказаний четырёх сетей мы проверяем их работу в двух ситуациях:

- обучение сетей на точках из интервала $t \in [0, T/2]$ и предсказание траекторий из отложенной выборки на интервале $t \in [T/2, T]$;
- обучение сетей на точках из интервала $t \in [0, T/4]$ и предсказание траекторий из отложенной выборки на интервале $t \in [T/4, T]$.

Обучение выполняется согласно процедуре, описанной выше. После него обе сети получают координаты траекторий из отложенной выборки только в один



(a) (b)

Рис. 2. Пошаговые ошибки предсказания ResNet и MLP для осциллятора Ван дер Поля с момента $T/2$ до T (a) и с $T/4$ до T (b)

момент времени, $\boldsymbol{x}(T/4)$ или $\boldsymbol{x}(T/2)$. Они итеративно предсказывают решение в следующий момент времени на шаг Δt и принимают результат на вход для дальнейшего прогнозирования. В обоих случаях нами измеряется средняя относительная ошибка предсказания по отложенной выборке

$$\varepsilon_{\text{avg}} = \frac{1}{MN_3} \sum_{i=1}^{N_3} \sum_{j=1}^{M} \frac{\|\boldsymbol{x}_i(t_j) - \tilde{\boldsymbol{x}}_i(t_j)\|}{\|\boldsymbol{x}_i(t_j)\|},$$

где M — число шагов предсказания ($M = 250$ для $t \in [T/2, T]$ и $M = 375$ для $t \in [T/4, T]$), t_j — моменты времени из интервала предсказания, взятые с шагом Δt, x_i взяты из отложенной выборки. Мы также измеряем относительную ошибку предсказания в момент $t = T$

$$\varepsilon_T = \frac{1}{N_3} \sum_{i=1}^{N_3} \frac{\|\boldsymbol{x}_i(T) - \tilde{\boldsymbol{x}}_i(T)\|}{\|\boldsymbol{x}_i(T)\|}.$$

Аналогичная процедура производится над сетью MLP, описанной в разделе 2.3.

5. Обсуждение результатов

Для системы осциллятора Ван дер Поля все четыре сети показывают благоприятные результаты (рис. 2). Они с хорошей точностью воспроизводят эволюцию системы. По результатам, представленным в таблице 2, можно видеть, что наименьшие ошибки предсказания на промежутке $[T/2, T]$ достигаются сетями RN1 ($\varepsilon_{\text{avg}} \approx 0.61$) и RN2 ($\varepsilon_T \approx 1.07$). На промежутке $[T/4, T]$ лучший результат по обеим ошибкам достигается RN1: $\varepsilon_{\text{avg}} \approx 0.42$ и $\varepsilon_T \approx 0.73$ (таблица 3). Для наглядной демонстрации качества реконструкции мы сравниваем предсказания решателя ОДУ и ResNet на 2 000 траекториях из отложенной выборки в момент

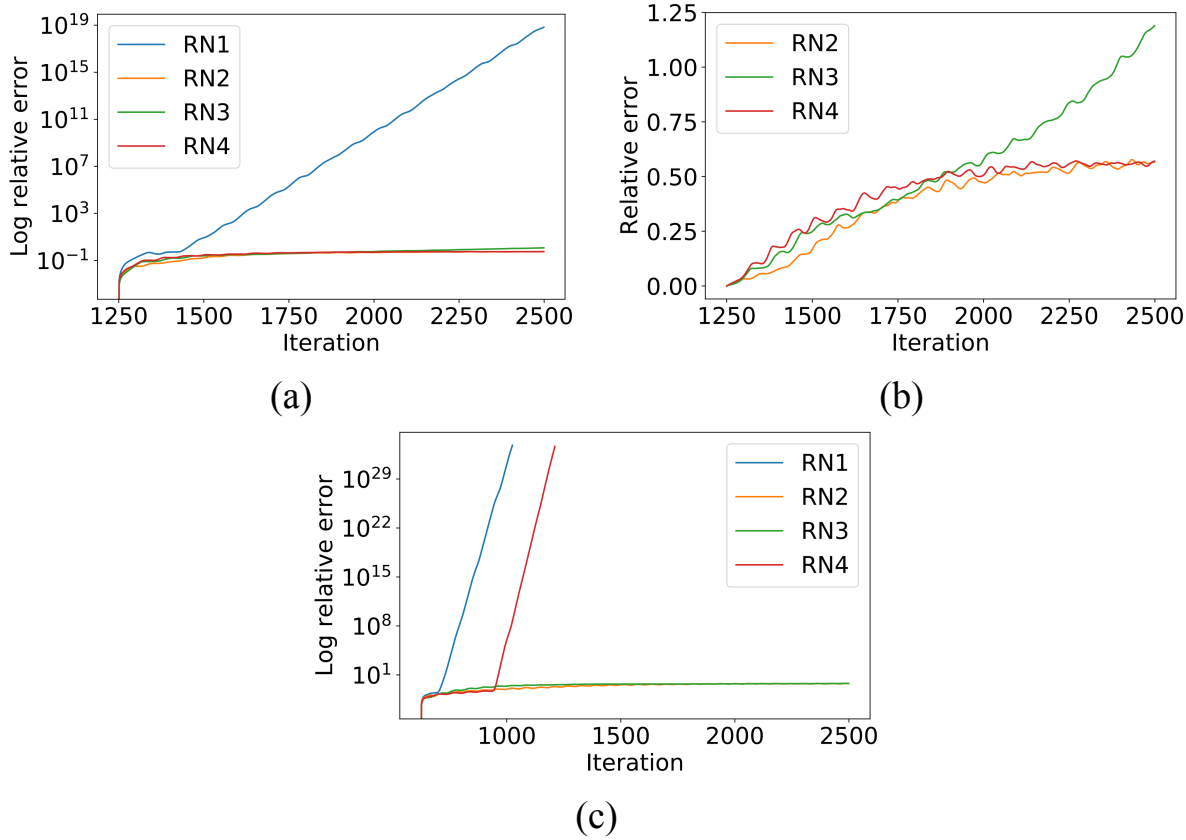


Рис. 3. Пошаговые ошибки предсказания ResNet для системы Лоренца с момента $T/2$ до T (a,b) и с $T/4$ до T (c)

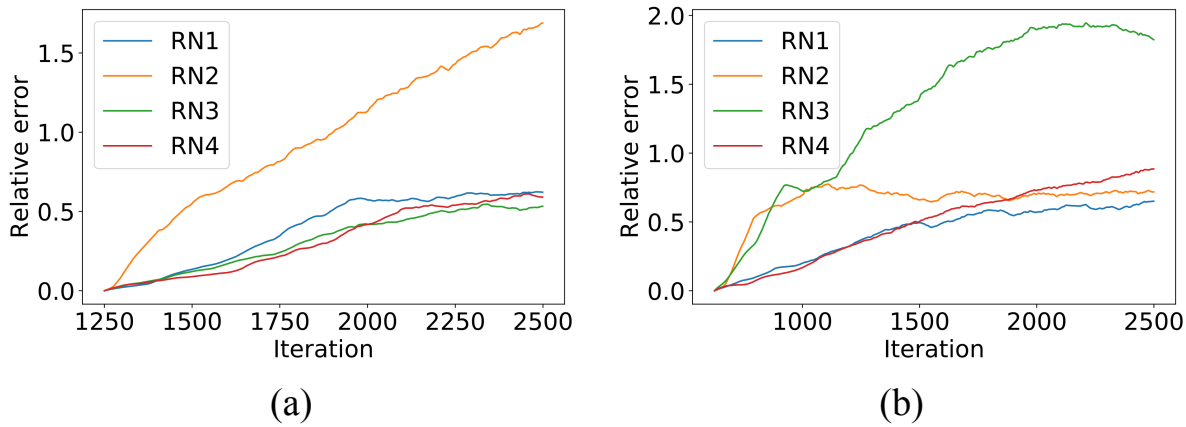


Рис. 4. Пошаговые ошибки предсказания ResNet для системы Рёсслера с момента $T/2$ до T (a) и с $T/4$ до T (b)

времени T (рис. 5). Как видно, сети успешно справляются с воспроизведением основной кривой за исключением небольших выбросов слева.

Как оказалось, сложнее всего сетям было выучить систему Лоренца. По рис. 3 и таблицам 2-3 видно, что в части экспериментов наблюдаются большие ошибки предсказания и даже переполнения (в таблицах обозначены как «NaN»). Возможной причиной такого явления может быть хаотическая природа

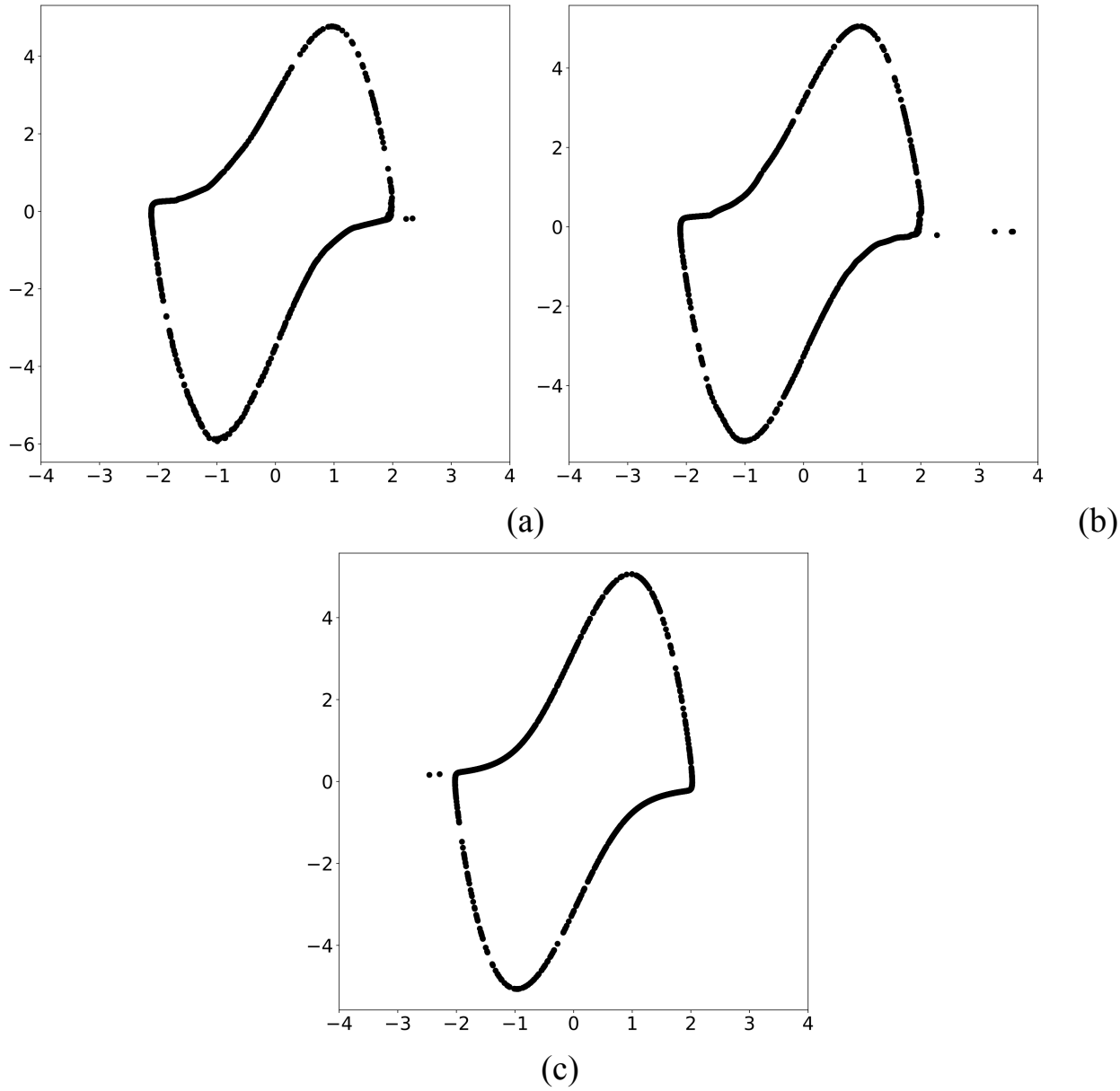


Рис. 5. Предсказания 2000 траекторий из отложенной выборки в момент T для системы Ван дер Поля: (a) и (b) получены RN1 для начальных времён $T/2$ и $T/4$ соответственно, (c) — решение, полученное решателем ОДУ

системы, из-за которой сетям тяжело прогнозировать её динамику с высокой точностью. Ошибка накапливается со временем, и предсказанные траектории заметно отличаются от тех, что получены словером. Тем не менее некоторые архитектуры смогли показать приемлемое качество работы. Предсказания на интервале $[T/2, T]$ с наименьшими ошибками даёт RN2, $\varepsilon_{\text{avg}} \approx 0.38$ и $\varepsilon_T \approx 0.57$. Сеть RN2 также показала лучший результат на $[T/4, T]$, $\varepsilon_{\text{avg}} \approx 0.34$ и $\varepsilon_T \approx 0.58$. Сравнение результатов предсказаний решателя и ResNet на рис. 6 наглядно иллюстрирует обозначенные ранее проблемы. Предсказания RN2 с момента $T/4$ улавливают вращение вокруг аттрактора, но не могут воспроизвести основную

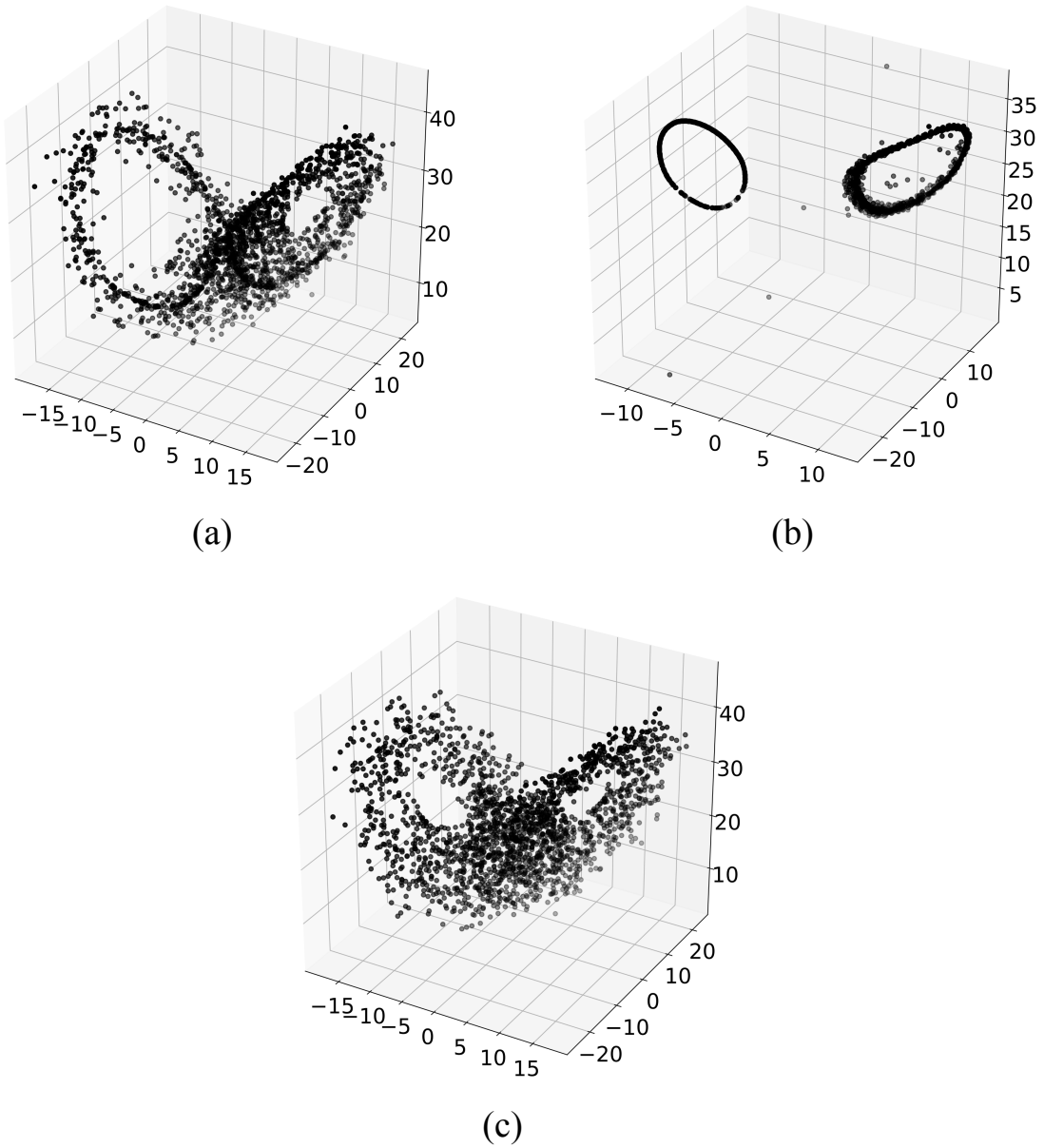


Рис. 6. Предсказания 2000 траекторий из отложенной выборки в момент T для системы Лоренца: (a) и (b) получены RN2 для начальных времён $T/2$ и $T/4$ соответственно, (c) — решение, полученное решателем ОДУ

динамику системы в форме бабочки. Однако в случае предсказания с момента $T/2$ RN2 показывает гораздо лучшие результаты, очень похожие на предсказания решателя.

Наконец, мы оцениваем работу сетей на аттракторе Рёсслера. Все архитектуры показывают стабильные результаты, что вполне ожидаемо, поскольку эволюция системы гораздо проще, чем в случае с системой Лоренца. RN3 даёт наименьшую ошибку для $[T/2, T]$, $\varepsilon_{\text{avg}} \approx 0.31$ и $\varepsilon_T \approx 0.53$, в то время как RN1 лучше всех работает на интервале $[T/4, T]$, $\varepsilon_{\text{avg}} \approx 0.42$ и $\varepsilon_T \approx 0.65$. Визуальный

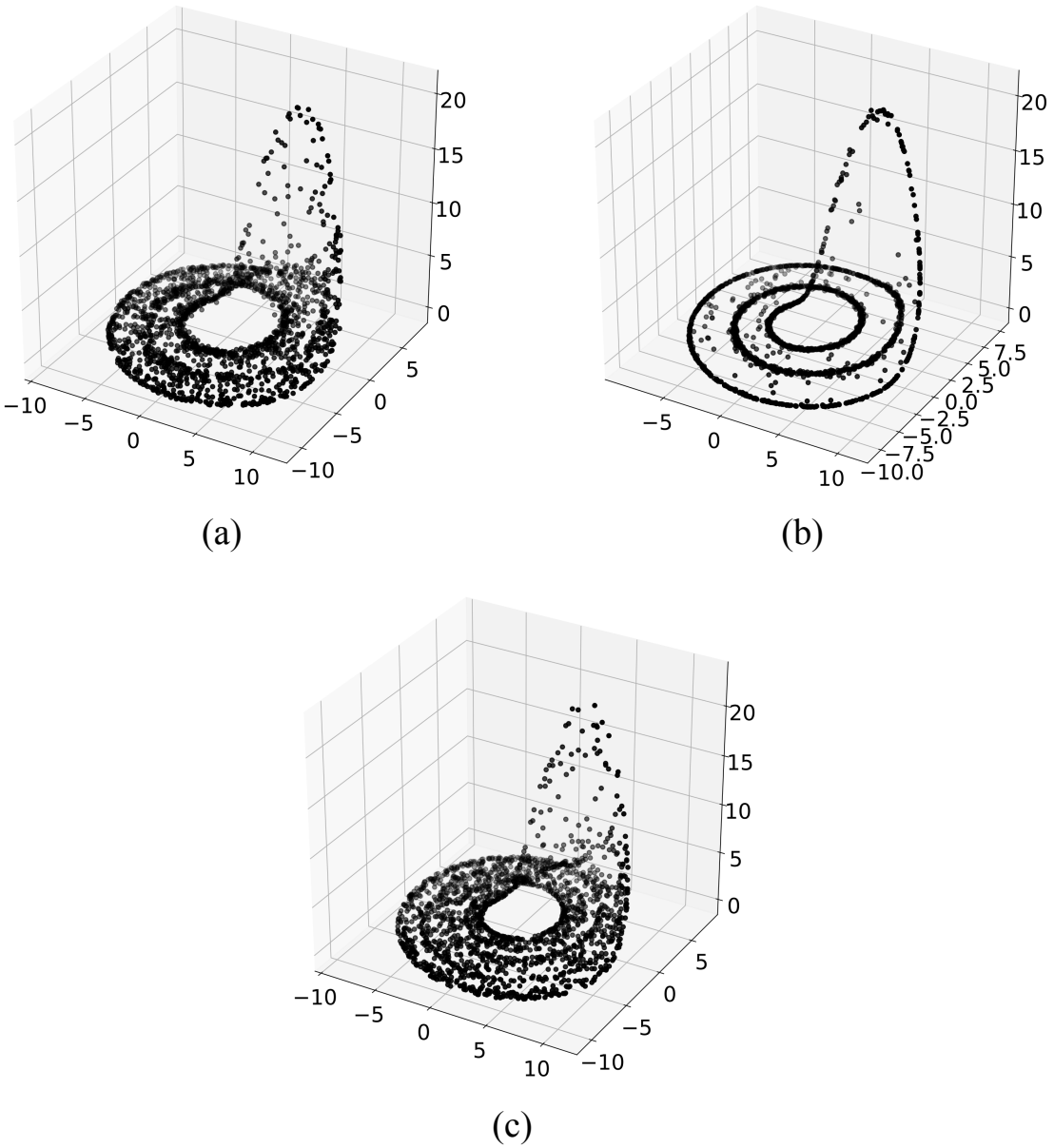


Рис. 7. Предсказания 2000 траекторий из отложенной выборки в момент T для системы Рёсслера: (a) получено RN3 для начального времени $T/2$, (b) получено RN1 для начального времени $T/4$, (c) — решение, полученное решателем ОДУ

анализ предсказаний в момент T (рис. 7) показывает, что в обоих экспериментах сети успешно восстанавливают форму аттрактора, но в случае предсказаний с момента $T/4$ точки проявляют неверную тенденцию и концентрируются около горизонтальной спирали.

Теперь сравним работу ResNet с MLP на задаче предсказания траекторий осциллятора Ван дер Поля. По рис. 2 видно, что значения относительных ошибок по порядку сопоставимы с результатами для RN1–RN4. Численные значения следующие: $\varepsilon_{\text{avg}} \approx 1.03$, $\varepsilon_{\text{T}} \approx 1.13$ для предсказаний с момента $T/2$ и $\varepsilon_{\text{avg}} \approx 1.81$,

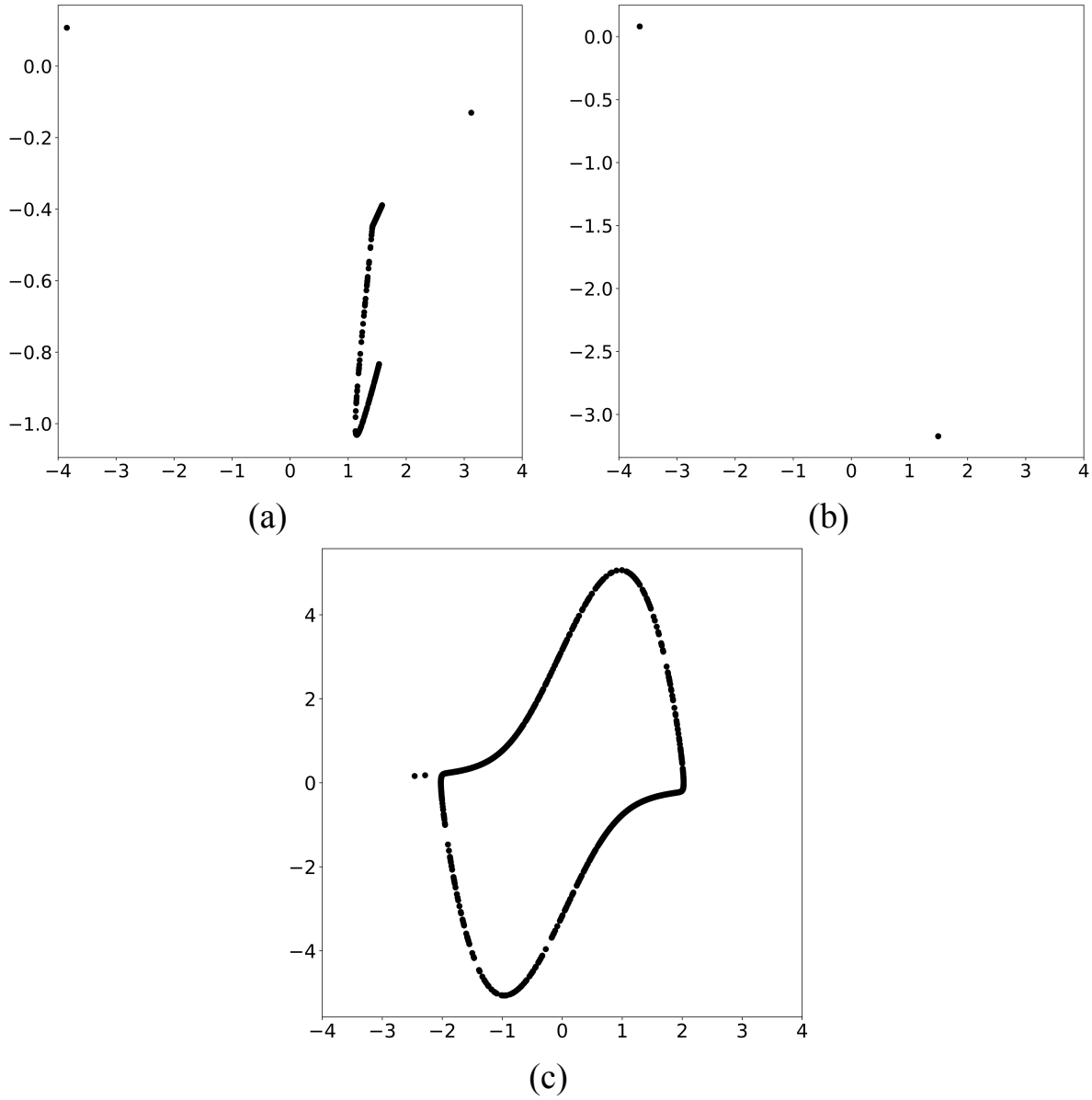


Рис. 8. Предсказания 2000 траекторий из отложенной выборки в момент T для системы ван дер Поля: (a) и (b) получены MLP для начальных времён $T/2$ и $T/4$ соответственно, (c) — решение, полученное решателем ОДУ

$\varepsilon_{\mathrm{T}} \approx 2.15$ для предсказаний с момента $T/4$. Основное отличие заключается в том, что в случае с MLP ошибка быстро растёт до определённого значения и дальше осциллирует вокруг него. Однако графики с предсказаниями явно показывают неспособность архитектуры MLP прогнозировать эволюцию на длинные временные интервалы. Кроме небольшого числа точек, которые всё ещё перемещаются к моменту T, основная их часть собралась в нескольких кластерах и остаётся практически без движения (рис. 8). Такое отсутствие движения в предсказанных MLP траекториях объясняет стагнирующую ошибку, упомянутую ранее.

Таблица 2. Ошибки предсказания на интервале $[T/2, T]$. Наименьшая ошибка в каждом столбце выделена жирным

	Ван дер Поль		Лоренц		Рёсслер	
	ε_{avg}	ε_T	ε_{avg}	ε_T	ε_{avg}	ε_T
RN1	**0.61**	1.09	1.60e17	6.80e18	0.39	0.62
RN2	0.83	**1.07**	**0.38**	**0.57**	0.97	1.69
RN3	0.70	1.27	0.52	1.19	**0.31**	**0.53**
RN4	1.01	1.49	0.42	0.57	0.32	0.59

Таблица 3. Ошибки предсказания на интервале $[T/4, T]$. Наименьшая ошибка в каждом столбце выделена жирным

	Ван дер Поль		Лоренц		Рёсслер	
	ε_{avg}	ε_T	ε_{avg}	ε_T	ε_{avg}	ε_T
RN1	**0.42**	**0.73**	NaN	NaN	**0.42**	**0.65**
RN2	7.84	90.51	**0.34**	**0.58**	0.65	0.72
RN3	0.63	1.16	0.41	0.59	1.33	1.82
RN4	1.47	1.66	NaN	NaN	0.49	0.88

6. Заключение

В данном препринте мы показали, что нейронные сети можно успешно использовать для предсказания решений систем ОДУ вида $\dot{\boldsymbol{x}}(t) = F(\boldsymbol{x}(t))$ по выборке их значений, т.е. без знания функции правой части системы $F(\boldsymbol{x})$. Ключевая особенность подхода заключается в применении остаточных нейронных сетей (ResNet), которые были созданы для борьбы с исчезающими и взрывными градиентами в глубоких сетях и представляются нам подходящей архитектурой для нашей задачи. Проверка качества работы ResNet на трёх тестовых нелинейных системах ОДУ с хаотическим поведением показала их способность достаточно хорошо обучаться динамике систем. Также эксперименты показывают отличные свойства устойчивости остаточных сетей, что позволяет делать прогнозы на более долгие временные интервалы по сравнению с существующими подходами машинного обучения.

Можно обозначить несколько направлений дальнейших исследований. Во-первых, остаточные сети разумно будет протестировать на более сложных прикладных задачах. Во-вторых, будет полезно рассмотреть различные модификации остаточных сетей, предложенные в последнее время, например RevNet [8]. Как показывают некоторые свойства таких сетей, с их помощью можно добиться повышения точности предсказания.

Список литературы

[1] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[2] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[3] R. Caruana, S. Lawrence, and C. L. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*, pages 402–408, 2001.

[4] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6572–6583, 2018.

[5] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[6] B. Ding, H. Qian, and J. Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841. IEEE, 2018.

[7] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[8] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. In *Advances in neural information processing systems*, pages 2214–2224, 2017.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[10] D. M. Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[13] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[14] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[15] J. Kiefer, J. Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] J. N. Kutz, X. Fu, and S. L. Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.

[18] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[19] Y. Lu, A. Zhong, Q. Li, and B. Dong. Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations. In *International Conference on Machine Learning*, pages 3282–3291, 2018.

[20] Z. Lu, B. R. Hunt, and E. Ott. Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6):061104, 2018.

[21] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

[22] M. Mai, M. D. Shattuck, and C. S. O'Hern. Reconstruction of ordinary differential equations from time series data. *arXiv preprint arXiv:1605.05420*, 2016.

[23] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[24] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

[25] S. Pawar, S. Rahman, H. Vaddireddy, O. San, A. Rasheed, and P. Vedula. A deep learning enabler for nonintrusive reduced order modeling of fluid flows. *Physics of Fluids*, 31(8):085101, 2019.

[26] G. Raskutti, M. J. Wainwright, and B. Yu. Early stopping and non-parametric regression: an optimal data-dependent stopping rule. *The Journal of Machine Learning Research*, 15(1):335–366, 2014.

[27] O. E. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[28] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

[29] J. C. Sprott. *Chaos and time-series analysis*, volume 69. Citeseer, 2003.

[30] J. C. Sprott. *Elegant chaos: algebraically simple chaotic flows*. World Scientific, 2010.

[31] A. P. Trischler and G. M. D'Eleuterio. Synthesis of recurrent neural networks for dynamical system simulation. *Neural Networks*, 80:67–78, 2016.

[32] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: theory and applications. *Journal of Computational Dynamics*, 1(2), 2014.

[33] B. Van der Pol. LXXXVIII. On "relaxation-oscillations". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):978–992, 1926.

[34] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.

[35] E. Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[36] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, Dec 2015.

[37] Y. Yao, L. Rosasco, and A. Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[38] Z. Yu and Z. Xiao-Dan. Estimating the bound for the generalized Lorenz system. *Chinese Physics B*, 19(1):010505, 2010.

Оглавление

1 Введение 3
2 Основные концепции нейронных сетей 5
3 Динамические системы 10
4 Постановка эксперимента 13
5 Обсуждение результатов 16
6 Заключение 22
Список литературы 23