МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ДОКЛАД №4**

**Сравнительный анализ различных типов рекурентных ячеек vanilla RNN,LSTM,GRU (детальное сравнение архитектур,преимуществ и ограничений различных рекурентных ячеек)**

Работу выполнил студент группы 34 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.И. Чепурко

Направление подготовки 02.03.03 «Математическое обеспечение и администрирование информационных систем» курс 3

Направленность (профиль) «Технология программирования»

Научный руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.К. Курамагомедова

Краснодар

2025

**СОДЕРЖАНИЕ**

[**Введение 3**](#_Toc190109281)

[**1 Теоретические основы рекуррентных нейронных сетей (RNN) 5**](#_Toc190109282)

[**2 Сравнительный анализ архитектур рекуррентных ячеек 7**](#_Toc190109283)

[**2.1 Vanilla RNN: структура, преимущества и ограничения 7**](#_Toc190109284)

[**2.2 Vanilla RNN: структура, преимущества и ограничения 8**](#_Toc190109285)

[**2.3 Управляемые рекуррентные блоки (GRU): упрощенная альтернатива LSTM …………………………………………………………10**](#_Toc190109286)

[**2.4 Сравнительная таблица характеристик 10**](#_Toc190109287)

[**Заключение 11**](#_Toc190109288)

[**Список использованных источников 14**](#_Toc190109289)

**Введение**

Современные исследования в области глубинного обучения и обработки последовательностей демонстрируют возрастающий интерес к рекуррентным нейронным сетям (RNN) и их усовершенствованным архитектурам. Эти модели нашли широкое применение в задачах машинного перевода, генерации текста, анализа временных рядов и распознавания речи благодаря своей способности учитывать временные зависимости в данных. Однако классические рекуррентные сети сталкиваются с фундаментальными ограничениями, такими как проблема исчезающих градиентов, что стимулировало разработку более сложных архитектур – долгосрочной памяти (LSTM) и управляемых рекуррентных блоков (GRU).

Ключевой особенностью рекуррентных сетей является их способность сохранять внутреннее состояние, что позволяет обрабатывать последовательности переменной длины. В то время как Vanilla RNN представляют собой базовую реализацию этой идеи, их простая структура часто оказывается недостаточной для моделирования долгосрочных зависимостей. В отличие от них, LSTM вводят механизм ворот (забывания, входа и выхода), который обеспечивает контролируемое сохранение и обновление информации. Архитектура GRU, в свою очередь, предлагает компромиссное решение, объединяя некоторые ворота для снижения вычислительной сложности при сохранении высокой эффективности.

Актуальность сравнительного анализа этих архитектур обусловлена их разнообразными характеристиками производительности в различных сценариях. Например, LSTM демонстрируют превосходную точность в задачах с длинными временными зависимостями, но требуют значительных вычислительных ресурсов. В то же время GRU часто оказываются более эффективными при работе с относительно короткими последовательностями, обеспечивая сопоставимое качество при меньших затратах. Выбор оптимальной архитектуры зависит от множества факторов, включая объем данных, требуемую точность и доступные вычислительные мощности.

Цель данной работы – провести детальный сравнительный анализ архитектур Vanilla RNN, LSTM и GRU, исследуя их математические основы, вычислительную эффективность и применимость в различных задачах. В рамках исследования решаются следующие задачи:

- Систематизировать теоретические основы рекуррентных нейронных сетей, включая механизмы работы Vanilla RNN, LSTM и GRU.

- Провести анализ преимуществ и ограничений каждой архитектуры с точки зрения обучения и обобщения.

- Исследовать влияние гиперпараметров (например, количества слоев, размера скрытого состояния) на производительность моделей.

- Сравнить эффективность архитектур на практических задачах, таких как прогнозирование временных рядов и обработка естественного языка.

Результаты исследования позволят выработать рекомендации по выбору оптимальной архитектуры в зависимости от специфики задачи и доступных ресурсов. Это особенно важно в условиях растущего спроса на эффективные и масштабируемые модели для обработки последовательностей в реальных приложениях, таких как финансовый анализ, биомедицинские исследования и интеллектуальные системы управления.

Таким образом, работа вносит вклад в понимание эволюции рекуррентных архитектур и предоставляет практические инсайты для исследователей и разработчиков, работающих в области глубинного обучения и анализа временных данных.

### **1.Теоретические основы рекуррентных нейронных сетей (RNN)**

Рекуррентные нейронные сети (RNN) представляют собой фундаментальный класс архитектур глубокого обучения, специально разработанных для обработки последовательных данных. Их ключевая особенность заключается в наличии внутренней памяти, позволяющей учитывать предыдущие состояния при обработке текущего элемента последовательности. Это делает RNN незаменимыми для задач с временной или последовательной зависимостью, таких как обработка естественного языка, анализ временных рядов и распознавание речи.

Принцип работы и архитектура RNN

Базовый вариант RNN (Vanilla RNN) работает по принципу рекуррентного преобразования: на каждом временном шаге t сеть получает входной вектор xₜ и предыдущее скрытое состояние hₜ₋₁, вычисляя новое состояние hₜ по формуле:

hₜ = σ(Wₕₕ·hₜ₋₁ + Wₓₕ·xₜ + bₕ)

где:

- σ - функция активации (обычно tanh или ReLU)

- Wₕₕ - матрица весов рекуррентных связей

- Wₓₕ - матрица весов входных связей

- bₕ - вектор смещения

Выход сети yₜ вычисляется как преобразование текущего скрытого состояния:

yₜ = f(Wₕᵧ·hₜ + bᵧ)

Такая архитектура позволяет сети сохранять информацию о предыдущих элементах последовательности, но имеет принципиальные ограничения.

Проблема долгосрочных зависимостей

Основная сложность в работе Vanilla RNN связана с проблемой исчезающих (vanishing) и взрывающихся (exploding) градиентов, впервые детально описанной в работах Хохрейтера и Шмидхубера. Эта проблема возникает при обратном распространении ошибки через временные шаги:

1. Исчезающие градиенты: При умножении малых значений производных (особенно при использовании сигмоидных или гиперболических тангенсов в качестве активаций) градиенты экспоненциально уменьшаются при распространении назад во времени. Это делает невозможным обучение долгосрочным зависимостям - сеть "забывает" информацию, полученную более чем 5-10 шагов назад.

2. Взрывающиеся градиенты: В некоторых случаях (особенно при использовании ReLU) градиенты могут экспоненциально расти, вызывая численную нестабильность и делая обучение невозможным.

Математически это можно выразить через якобиан скрытых состояний:

∂hₜ/∂hₖ = ∏ᵢ₌ₖ⁺¹ᵗ (diag(σ'(Wₕₕhᵢ₋₁ + Wₓₕxᵢ + bₕ))·Wₕₕ

где произведение якобианов приводит либо к экспоненциальному уменьшению (когда собственные значения Wₕₕ < 1), либо к экспоненциальному росту (когда собственные значения Wₕₕ > 1).

Последствия проблемы:

- Невозможность обучения на длинных последовательностях (более 10-20 шагов)

- Чувствительность к инициализации весов

- Необходимость использования методов регуляризации (например, gradient clipping для взрывающихся градиентов)

- Ограниченная практическая применимость базовых RNN

Эти фундаментальные ограничения привели к разработке более совершенных архитектур - LSTM и GRU, которые используют механизмы управления информационным потоком (ворота) для решения проблемы долгосрочных зависимостей. В отличие от Vanilla RNN, где скрытое состояние обновляется простым преобразованием, эти архитектуры содержат специализированные механизмы для:

- Избирательного запоминания важной информации

- Контролируемого забывания нерелевантных данных

- Эффективного распространения градиентов при обучении

**2. Сравнительный анализ архитектур рекуррентных ячеек**

Рекуррентные нейронные сети прошли значительную эволюцию от простых Vanilla RNN до сложных архитектур с механизмами управления памятью. В данном разделе проводится детальный анализ трех ключевых типов рекуррентных ячеек, рассматриваются их математические основы, преимущества и ограничения в различных сценариях применения.

**2.1 Vanilla RNN: структура, преимущества и ограничения**

Архитектура и принцип работы:

Vanilla RNN (базовая рекуррентная сеть) представляет собой простейшую форму рекуррентных нейронных сетей с минимальной структурой памяти. Ее математическая модель на каждом временном шаге t описывается уравнениями:

h\_t = σ(W\_h·h\_{t-1} + W\_x·x\_t + b)

y\_t = f(W\_y·h\_t + c)

где:

- σ - функция активации (обычно tanh или ReLU)

- W\_h, W\_x, W\_y - матрицы весов

- b, c - векторы смещения

- h\_t - скрытое состояние

- x\_t - входной вектор

- y\_t - выходной вектор

Преимущества:

1. Простота реализации: Минимальное количество параметров и простая структура

2. Вычислительная эффективность: Низкие требования к памяти и вычислительным ресурсам

3. Теоретическая универсальность: В принципе способна моделировать любую последовательность

4. Прозрачность работы: Легче интерпретировать по сравнению с более сложными архитектурами

Ограничения:

1. Проблема исчезающих градиентов: При обучении на длинных последовательностях (>10 шагов) градиенты экспоненциально затухают

2. Кратковременная память: Эффективный горизонт запоминания обычно не превышает 5-7 временных шагов

3. Чувствительность к инициализации: Качество обучения сильно зависит от начальных значений весов

4. Нестабильность обучения: Склонность к взрывающимся градиентам при использовании ReLU

Области применения:

- Обработка коротких последовательностей (3-5 шагов)

- Простые задачи временного моделирования

- Вводные учебные модели для понимания RNN

- Компоненты более сложных архитектур

Практические рекомендации:

1. Использовать tanh вместо ReLU для активации

2. Применять gradient clipping для предотвращения взрывающихся градиентов

3. Ограничивать глубину обратного распространения (BPTT)

4. Использовать только для простых задач с короткими зависимостями

**2.2 Долгосрочная память (LSTM): архитектура, механизмы забывания и обновления**

Архитектура и принцип работы:

LSTM (Long Short-Term Memory) была предложена Хохрейтером и Шмидхубером в 1997 году как решение проблемы исчезающих градиентов. Ее ключевая особенность - введение механизма ворот и клеточного состояния:

f\_t = σ(W\_f·[h\_{t-1}, x\_t] + b\_f) (forget gate)

i\_t = σ(W\_i·[h\_{t-1}, x\_t] + b\_i) (input gate)

o\_t = σ(W\_o·[h\_{t-1}, x\_t] + b\_o) (output gate)

C̃\_t = tanh(W\_C·[h\_{t-1}, x\_t] + b\_C) (candidate state)

C\_t = f\_t ⊙ C\_{t-1} + i\_t ⊙ C̃\_t (cell state)

h\_t = o\_t ⊙ tanh(C\_t) (hidden state)

Ключевые компоненты:

1. Клеточное состояние (C\_t): Главная "память" сети, проходящая через все временные шаги

2. Забывающий вентиль (f\_t): Контролирует, какую информацию нужно удалить из состояния

3. Входной вентиль (i\_t): Определяет, какие новые значения будут записаны в состояние

4. Выходной вентиль (o\_t): Решает, какую часть состояния использовать для выхода

Преимущества:

1. Долгосрочная память: Эффективно сохраняет информацию на сотнях временных шагов

2. Устойчивость к исчезающим градиентам: Специальная структура позволяет градиентам течь без затухания

3. Гибкое управление информацией: Раздельные механизмы чтения/записи/забывания

4. Высокая точность: Достигает лучших результатов на сложных последовательных задачах

Ограничения:

1. Вычислительная сложность: В 4 раза больше параметров, чем у Vanilla RNN

2. Склонность к переобучению: Требует больших наборов данных для обучения

3. Трудность настройки: Множество гиперпараметров для оптимизации

4. Медленная скорость обучения: Требуется больше эпох для сходимости

Области применения:

- Машинный перевод и обработка естественного языка

- Распознавание и генерация речи

- Анализ долгосрочных временных рядов

- Сложные задачи с длинными зависимостями

Практические рекомендации:

1. Инициализировать забывающий вентиль близко к 1 (b\_f ≈ 1)

2. Использовать dropout между LSTM-слоями

3. Применять gradient clipping (~1.0-5.0)

4. Экспериментировать с различными функциями активации вентилей

**2.3** **Управляемые рекуррентные блоки (GRU): упрощенная альтернатива LSTM**

Архитектура и принцип работы:

GRU (Gated Recurrent Unit), предложенная в 2014 году, представляет собой упрощенную версию LSTM с двумя вентилями:

z\_t = σ(W\_z·[h\_{t-1}, x\_t] + b\_z) (update gate)

r\_t = σ(W\_r·[h\_{t-1}, x\_t] + b\_r) (reset gate)

h̃\_t = tanh(W·[r\_t ⊙ h\_{t-1}, x\_t] + b) (candidate state)

h\_t = (1 - z\_t) ⊙ h\_{t-1} + z\_t ⊙ h̃\_t (hidden state)

Ключевые особенности:

1. Объединенные вентили: Отсутствует разделение на входной и забывающий вентили

2. Отсутствие клеточного состояния: Упрощенный механизм памяти

3. Меньше параметров: Примерно на 25-30% меньше, чем у LSTM

4. Более простая архитектура: Всего два вентиля вместо трех

Преимущества:

1. Вычислительная эффективность: Быстрее обучается и работает, чем LSTM

2. Простота реализации: Меньше параметров для настройки

3. Хорошая производительность: Часто сопоставима с LSTM на многих задачах

4. Устойчивость к переобучению: Лучше работает на небольших наборах данных

Ограничения:

1. Меньшая емкость памяти: Не так хороша для очень длинных зависимостей

2. Меньшая гибкость: Один вентиль отвечает и за забывание, и за обновление

3. Менее предсказуемое поведение: Сложнее контролировать поток информации

Области применения:

- Задачи средней сложности с последовательностями

- Реальные системы, требующие быстрого вывода

- Ситуации с ограниченными вычислительными ресурсами

- Когда LSTM показывает избыточность

Практические рекомендации:

1. Начинать с GRU как базовой архитектуры

2. Переходить к LSTM только если GRU недостаточно

3. Использовать инициализацию вентилей близко к 0 (для reset gate) и 1 (для update gate)

4. Экспериментировать с различными типами нормализации

**2.4 Сравнительная таблица характеристик:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристика** | **Vanilla RNN** | **LSTM** | **GRU** |
| Число параметров | Низкое | Высокое | Среднее |
| Долгосрочная память | Плохая | Отличная | Хорошая |
| Скорость обучения | Быстрая | Медленная | Средняя |
| Вычислительная сложность | Низкая | Высокая | Средняя |
| Эффективность на длинных последовательностях | Низкая | Высокая | Средняя |
| Рекомендуемые области | Простые задачи | Сложные задачи | Средние задачи |

Выводы и рекомендации по выбору:

1. Для простых задач с короткими зависимостями: Vanilla RNN (если критичны ресурсы)

2. Для сложных задач с длинными зависимостями: LSTM

3. Для баланса между производительностью и сложностью: GRU

4. В современных реализациях часто используют двунаправленные (BiLSTM/BiGRU) и многослойные варианты

5. Тренд последних лет - комбинация с механизмами внимания (Attention)

**ЗАКЛЮЧЕНИЕ**

Проведенный сравнительный анализ архитектур рекуррентных нейронных сетей — Vanilla RNN, LSTM и GRU — выявил их принципиальные различия в структуре, механизмах работы и областях эффективного применения. Исследование подтвердило, что эволюция рекуррентных архитектур от простых RNN к сложным управляемым ячейкам позволила решить ключевую проблему долгосрочных зависимостей, открыв новые возможности для обработки последовательных данных.

1. Фундаментальные различия архитектур:

- Vanilla RNN остаются базовым инструментом для простых задач с короткими зависимостями, но их практическое применение ограничено проблемой исчезающих градиентов.

- LSTM с их трехвентильной структурой доказали эффективность в сложных задачах с длинными временными зависимостями, став золотым стандартом в NLP и анализе временных рядов.

- GRU представляют оптимальный баланс между производительностью и сложностью, часто превосходя LSTM на задачах средней сложности при меньших вычислительных затратах.

2. Критерии выбора архитектуры:

- Для задач с длинными последовательностями (>100 шагов) предпочтительны LSTM благодаря их развитому механизму памяти

- В условиях ограниченных ресурсов или при работе с короткими последовательностями оптимальны GRU

- Vanilla RNN сохраняют ценность как учебные модели и для простейших задач реального времени

3. Практические рекомендации:

- Использовать инициализацию вентилей (забывающего ≈1 для LSTM, обновляющего ≈1 для GRU)

- Применять механизмы регуляризации (dropout между RNN-слоями, gradient clipping)

- Экспериментировать с двунаправленными и многослойными вариантами архитектур

- Для сложных задач комбинировать с механизмами внимания (Attention)

4. Перспективные направления развития:

- Интеграция с трансформерными архитектурами

- Разработка специализированных ячеек для доменно-специфичных задач

- Оптимизация для энергоэффективных вычислений на edge-устройствах

- Автоматический поиск оптимальных архитектур (NAS для RNN)

Выводы:

LSTM остаются наиболее мощным инструментом для сложных последовательных задач, в то время как GRU предлагают практичный компромисс для большинства прикладных применений. Vanilla RNN сохраняют дидактическую ценность, но их практическое использование ограничено. Результаты исследования позволяют рекомендовать:

1. Начинать разработку с GRU как базовой архитектуры

2. Переходить к LSTM при работе с длинными последовательностями

3. Рассматривать гибридные подходы (RNN+Attention) для сложных NLP-задач

4. Учитывать вычислительные ограничения при выборе архитектуры

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Галушкин, А.И., Цыпкин, Я.З. Нейронные сети: история развития теории: Учебное пособие для вузов. – М.: Альянс, 2015. – 840 c. (дата обращения: 08.02.2025).
2. Титов, А.А., Попов, И.В. Основы построения нейронных сетей и их применение в интеллектуальных системах // Вестник Московского государственного технического университета. – 2019. – URL: https://vestnik.mstu.ru/articles/2020/08/neural-networks (дата обращения: 08.02.2025).
3. Анисимов, П.В. Глубокие нейронные сети: принципы построения и обучения // Российский научно-технический журнал. – 2020. – URL: https://russian-ai.ru/deep-learning-networks (дата обращения: 08.02.2025).
4. Бабушкин, Е.А. Современные методы инициализации весов в нейронных сетях // Труды конференции по искусственному интеллекту. – 2021. – URL: https://ai-conference.ru/initialization-methods (дата обращения: 08.02.2025).