УДК 004.89

Мурадян Д. С., Алимов П. Г.

Реализация и внедрение архитектуры Gated Recurrent Unit в пакет нейросетевой аппроксимации дифференциальных уравнений DEGANN

Рекомендовано к публикации доцентом Гориховским В. И.

1. Введение. Дифференциальные уравнения играют ключевую роль в самых разных областях науки и техники. Решение таких уравнений – задача, требующая больших вычислений. С усложнением и увеличением размерности задачи, традиционные численные методы решения ДУ начинают сталкиваться с проблемами, связанными с длительным временем вычислений и недостаточной точностью [1].

Это заставляет задуматься об альтернативных способах решения таких задач. Одним из них является использование нейронных сетей, которые могут аппроксимировать функции, описывающие ДУ. Такой подход используется в пакете нейросетевой аппроксимации ДУ DEGANN [2].

Функциональность этого пакета, на данный момент, ограничена одной архитектурой НС – многослойным перцептроном (MLP – Multi-Layer Perceptron). Эта архитектура может быть недостаточно эффективна в некоторых задачах аппроксимации ДУ, так как MLP не достаточно хорошо улавливает последовательные зависимости в данных [3]. В отличие от MLP, рекуррентные нейронные сети (RNN – Recurrent Neural Network) [4] специально предназначены для работы с последовательностями, которые очень важны в анализе нелинейных ДУ.

Мурадян Денис Степанович – Автор, студент бакалавриата, Санкт-Петербургский государственный университет; e-mail: muradyan.denis@inbox.ru

Алимов Павел Геннадьевич – Соавтор, преподаватель, Санкт-Петербургский государственный университет; e-mail: st076209@student.spbu.ru

Гориховский Вячеслав Игоревич – доцент, Санкт-Петербургский государственный университет; e-mail: gorihovskyvyacheslav@gmail.com

Работа выполнена при поддержке СПБГУ (Pure ID 116636233).

Однако RNN имеют известную проблему затухания и взрыва градиента, из-за чего обучение становится затруднительным на длинных последовательностях [5]. Архитектура (GRU – Gated Recurrent Unit) является усовершенствованной версией RNN и частично решает эту проблему благодаря использованию элементов управления потоком данных.

Внедрение GRU в пакет DEGANN может помочь оптимизировать и улучшить качество аппроксимации ДУ.

2. Реализация и интеграция в пакет.

Для реализации архитектуры `Gated Recurrent Unit` в пакете DEGANN разработан класс `TensorflowGRUNet`, который наследуется от `tf.keras.Model`

Первоначально, создаётся список GRU-слоёв, с заданными функциями активации (основная — *tanh*, рекуррентная — *sigmoid*) и инициализаторами весов и смещений. Последовательная обработка входных данных этими слоями передаётся в полносвязный выходной слой с линейной активацией, что обеспечивает регрессионное предсказание.

Логика компиляции модели выполняется с помощью реализованного метода `custom_compile` — он отвечает за настройку оптимизатора, функции потерь, метрик и так далее, а метод `to_dict`, экспортирует конфигурацию модели в виде словаря для дальнейшей интеграции с другими компонентами DEGANN. В методе `call` реализован прямой проход через нейросеть, а вычисление градиентов производится стандартными средствами TensorFlow. Разработанная оболочка обеспечивает более гибкую настройку и унифицированный интерфейс для работы модели в системе.

Основным элементом данной реализации является класс `TensorflowGRUNet`, который осуществляет следующее этапы:

- Построение и инициализация GRU-слоёв, каждый из которых имеет число нейронов, определяемое параметром **block_size**,
- Настройка ключевых параметров:
  - **input_size** – размер входных данных (количество признаков);
  - **output_size** – размер выхода;

  - **activation_func** и **recurrent_activation** – функции активации для основного и рекуррентного состояний (по умолчанию *tanh* и *sigmoid*);
  - **weight** и **biases** – инициализаторы весов и смещений.

- Организация прямого прохода данных через сеть с помощью метода `call`. В этом методе входные данные последовательно обрабатываются всеми GRU-слоями, а затем результат передается в выходной полносвязный слой с линейной активацией, что оптимально для регрессионных задач.

Интеграция архитектуры GRU в DEGANN осуществляется посредством параметра `net_type`, передаваемого при создании модели через базовый класс `IModel`. При установке `net_type="GRUNet"` внутри `IModel` вызывается метод, создающий экземпляр класса `TensorflowGRUNet`. Такой подход позволил сохранить единообразный интерфейс работы с моделями различных типов (например, с DenseNet или другими), не нарушая общую модульную структуру проекта. Это обеспечивает возможность использования стандартных методов обучения, тестирования и экспорта моделей.

3. Валидация решения.

3.1. Параметры эксперемента.

В данном разделе представлено экспериментальное сравнение эффективности двух архитектур, реализованных в DEGANN: GRUNet (на основе рекуррентных слоёв GRU) и DenseNet (традиционный многослойный перцептрон, MLP).

Цель эксперимента — продемонстрировать, что при одинаковых гиперпараметрах (3 слоя, 30 нейронов в каждом, функция потерь MSE, метрики MAPE, R^2) и фиксированном времени обучения архитектура GRU обеспечивает более точное приближение целевых функций за счёт своей способности учитывать последовательные зависимости в данных.

Конфигурация тестирующей системы.

CPU: Intel i5 1240p | GPU: Iris Xe Graphics G7 80EUs | RAM: 16 gb

Выбор тестовых функций и их обоснование.

Для проверки моделей были выбраны следующие функции, встроенные в DEGANN, которые представляют собой достаточно сложные для аппроксимации зависимости.

- Функция $f_{\sin}(x) = \sin(10x)$ демонстрирует периодическую нелинейность, что позволяет проверить способность модели захватывать колебательные процессы.

- Функция $f_{\text{hyperbol}}(x) = \dfrac{x^2 + 0.5}{x + 0.1}$ обладает неоднородной зависимостью, где квадратичный рост сочетается с особенностями в малых значениях x, что требует от модели адаптивного поведения в различных областях определения.

- Функция $f_{\text{hardsin}}(x) = \sin\Big(\ln\big(x^{\sin(10x)}\big)\Big)$, включающая логарифмическое преобразование и экспоненциальные эффекты, ещё более усложняет задачу аппроксимации за счёт усиленной нелинейности.

Такой набор тестовых функций позволяет всесторонне оценить эффективность сравниваемых архитектур, поскольку каждая из них предъявляет уникальные требования к способности модели моделировать сложные, неоднородные зависимости.

Параметры моделей.
Для каждой архитектуры эксперименты проводятся при фиксированном времени обучения. В частности, результаты тестирования собираются для трёх вариантов времени обучения: 15, 30 и 45 секунд. При каждом варианте фиксируется функция потерь – Mean Squared Error (MSE), метрики – Mean Absolute Percentage Error (MAPE) и коэффициент детерминации (R^2), а так же затраты по памяти.

Таким образом, основная задача эксперимента состоит в сравнении показателей качества аппроксимации для GRUNet и DenseNet при идентичных условиях обучения. При этом, важно отметить, что использование одинаковых гиперпараметров позволяет провести объективное сравнение, где разница в результатах обусловлена внутренней структурой моделей, а не различиями в настройках.

3.2. Результаты эксперемента.

Ниже приведены три таблицы, демонстрирующие структуру представления результатов эксперимента для каждого из выбранных интервалов времени обучения.

Таблица 1. Результаты эксперимента (время обучения: 15 секунд)

Архитектура	**Тестовая функция**	**Память (MB)**	**Loss MSE**	**MAPE**	R^2
GRUNet	hyperbol	112.460	0.002	1.536	0.993
GRUNet	sin	111.360	0.003	4.149	0.994
GRUNet	hardsin	112.450	0.045	14.647	0.738
DenseNet	sin	35.000	0.091	23.095	0.812
DenseNet	hardsin	41.770	0.017	10.100	0.902
DenseNet	hyperbol	33.990	0.051	13.346	0.819

Таблица 2. Результаты эксперимента (время обучения: 30 секунд)

Архитектура	**Тестовая функция**	**Память (MB)**	**Loss MSE**	**MAPE**	R^2
GRUNet	hardsin	112.62	0.004	4.008	0.976
GRUNet	hyperbol	116.27	0.005	3.508	0.980
DenseNet	sin	37.52	0.012	7.509	0.975
DenseNet	hyperbol	36.95	0.011	5.989	0.961
DenseNet	hardsin	44.78	0.005	5.101	0.971
GRUNet	sin	115.39	0.003	4.169	0.994

Таблица 3. Результаты эксперимента (время обучения: 45 секунд)

Архитектура	**Тестовая функция**	**Память (MB)**	**Loss MSE**	**MAPE**	R^2
GRUNet	sin	120.59	0.003	4.101	0.995
GRUNet	hardsin	116.84	0.002	3.158	0.987
GRUNet	hyperbol	117.79	0.002	1.174	0.992
DenseNet	sin	38.98	0.007	7.339	0.985
DenseNet	hyperbol	47.98	0.004	3.138	0.987
DenseNet	hardsin	47.71	0.006	5.741	0.962

3.3. Анализ.

Средние значения метрик MAPE, коэффициента детерминации (R^2) и потребляемой памяти для каждой архитектуры и временного интервала представлены ниже:

- **Время обучения: 15 секунд**
  GRUNet: MAPE = 6.777, R^2 = 0.908, Память = 112.09 MB
  DenseNet: MAPE = 15.514, R^2 = 0.844, Память = 36.92 MB

- **Время обучения: 30 секунд**
  GRUNet: MAPE = 3.895, R^2 = 0.983, Память = 114.76 MB
  DenseNet: MAPE = 6.2, R^2 = 0.969, Память = 39.75 MB

- **Время обучения: 45 секунд**
  GRUNet: MAPE = 2.811, R^2 = 0.991, Память = 118.41 MB
  DenseNet: MAPE = 5.406, R^2 = 0.978, Память = 44.89 MB

Анализ проведённых экспериментов демонстрирует, что архитектура GRUNet (go метрикам MAPE и коэффициенту детерминации (R^2)) при идентичных гиперпараметрах и фиксированном времени обучения обеспечивает более высокое качество аппроксимации целевых функций по сравнению с DenseNet.

3.4. Вывод.

Несмотря на несколько более высокие затраты памяти, разница в потреблении ресурсов (порядка 80 МБ) не является значительной для пользователей в большинстве практических конфигураций, в отличие от времени. GRUNet демонстрирует более быструю сходимость, обеспечивая лучшее качество аппроксимации за меньшее время. Это делает данную архитектуру перспективной для задач аппроксимации дифференциальных уравнений в рамках пакета DEGANN, где важна эффективность обучения при сохранении высокой точности результатов.

Литература

1. Ernst Hairer, Gerhard Wanner Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. 978-3-540-60452-5 изд. Гейдельберг: Springer-Verlag, 1996 . 629 с.

2. Пакет нейросетевой аппроксимации дифференциальных уравнений // DEGANN URL: https://pypi.org/project/degann/ (дата обращения: 10.03.2025).

3. Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, David Duvenaud Neural Ordinary Differential Equations // Advances in Neural Information Processing Systems. 2018. №31. С. 6571–6583.

4. SStryker C. What is a Recurrent Neural Network? // IBM . 2024.

5. Carlos Mosquera, Greg Van Houdt, Gonzalo Napoles A review on the long short-term memory model // Artificial Intelligence Review. 2020 . №53. С. 5929–5955.