**Слайд 1. Титульный слайд**  
Добрый день, уважаемые коллеги!  
Меня зовут Денис Мурадян. Сегодня представляется работа «Реализация и внедрение архитектуры GRU в пакет нейросетевой аппроксимации дифференциальных уравнений DEGANN». В ходе доклада будут освещены теоретические предпосылки, этапы реализации и результаты экспериментов, демонстрирующие работу GRU в данном пакете.

**Слайд 2. 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 начинают сталкиваться с проблемами, связанными с длительным временем вычислений и недостаточной точностью.

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 DEGANN.

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. В отличие от MLP, рекуррентные нейронные сети (RNN – Recurrent Neural Network) специально предназначены для работы с последовательностями, которые очень важны в анализе нелинейных ДУ.

Однако RNN имеют известную проблему затухания и взрыва градиента, из-за чего обучение становится затруднительным на длинных последовательностях.

В связи с этим была выбрана архитектура (GRU – Gated Recurrent Unit) которая 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.

**Слайд 3. Реализация и интеграция в DEGANN**  
Для реализации архитектуры Gated Recurrent Unit в пакете DEGANN разработан класс TensorflowGRUNet, который наследуется от tf.keras.Model .

Среди ключевых этапов реализации можно выделить:  
– **Создание списка GRU-слоёв,** где каждый слой имеет (тангенсальную) tanh функцию активации для основного состояния и (сигмоидную) sigmoid для рекуррентного – **Далее** Входная информация последовательно проходит через созданные GRU-слои, после чего результат поступает на полносвязный выходной слой с линейной активацией для регрессионной задачи.  
– **Для настройки параметров**, экспорта конфигурации, а также прямого прохода данных через нейронную сеть были реализованы методы **custom\_compile**, отвечает за компиляцию модели и выбор оптимизатора, функции потерь, метрик и тд. Метод **to\_dict** для экспорта конфигурации модели в виде словаря для дальнейшей интеграции с другими компонентами пакета. И метод **call** – который реализует прямой проход через нейросеть, а вычисление градиентов производится стандартными средствами TensorFlow.  
–  **Интеграция архитектуры** 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.

**Слайд 4. Валидация решения: Экспериментальная установка**

Для проверки работы архитектуры в пакете был поставлен эксперимент.

**основная задача эксперимента** состоит в сравнении показателей качества аппроксимации двух архитектур 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 настройка

Для эксперимента были заданы условия:  
– Три слоя, по 30 нейронов в каждом,  
– Использовалась функция потерь Mean Squared Error (MSE)  
- метрики: Mean Absolute Percentage Error (MAPE) и коэффициент детерминации (R2),  
– Время обучения фиксировалось на интервалах 15, 30 и 45 секунд.  
  
обоснование:  
**Функция потерь (MSE):**  
MSE (среднеквадратичная ошибка) является стандартным выбором для задач регрессии, поскольку она эффективно штрафует большие отклонения между предсказанными и истинными значениями. Это способствует более точной настройке модели в процессе оптимизации.

**Метрика MAPE:**  
MAPE (средняя абсолютная процентная ошибка) позволяет оценить относительную погрешность модели, что особенно полезно при анализе ошибок в различных диапазонах значений целевой функции. Она помогает понять, насколько в процентном соотношении модель точна в своих предсказаниях.

**Коэффициент детерминации (R²):**R² показывает, какую долю дисперсии (разброса) зависимой переменной (например, Y) модель объясняет с помощью независимых переменных (например, X). Высокое значение R² свидетельствует о том, что модель хорошо описывает данные, что является важным индикатором качества аппроксимации. Значение 1 указывает на идеальное соответствие, когда модель идеально отражает взаимосвязь между переменными

**Слайд 5. Выбор тестовых функций**  
Для эксперимента над моделями были выбраны тестовые функции, встроенные в DEGANN, которые представляют собой достаточно сложные для аппроксимации зависимости

– **fsin(x) = sin(10x).** периодическая функция с фиксированной частотой, позволяющая проверить способность модели улавливать синусоидальные колебания   
– **fhyperbol(x) = (x² + 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.

– **fhardsin(x) = sin(ln(x · sin(10x))).** функция объединяет синусоидальные колебания с логарифмическим масштабированием, что приводит к сложной нелинейной динамике и усложняет задачу аппроксимации.

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

**Слайд 6. Результаты эксперимента и анализ**  
В ходе эксперимента были получены результаты по метрикам при трёх интервалах времени обучения. Для более наглядного анализа результатов, были подсчитаны средние показатели метрик для каждой из архитектур на соответсвенном временном промежутке обучения: (сейчас вы можете видеть их на слайде)

* **При 15 секундах обучения:**  
   • GRUNet: MAPE ≈ 6.78, R2 ≈ 0.908, затраты памяти ~112 МБ;  
   • DenseNet: MAPE ≈ 15.51, R2 ≈ 0.844, затраты памяти ~37 МБ.
* **При 30 секундах обучения:**  
   • GRUNet: MAPE ≈ 3.90, R2 ≈ 0.983, затраты памяти ~115 МБ;  
   • DenseNet: MAPE ≈ 6.20, R2 ≈ 0.969, затраты памяти ~40 МБ.
* **При 45 секундах обучения:**  
   • GRUNet: MAPE ≈ 2.81, R2 ≈ 0.991, затраты памяти ~118 МБ;  
   • DenseNet: MAPE ≈ 5.41, R2 ≈ 0.978, затраты памяти ~45 МБ.

Анализ полученных значений демонстрирует, что архитектура GRU при идентичных гиперпараметрах и фиксированном времени обучения обеспечивает более высокое качество аппроксимации целевых функций по сравнению с архитектурой MLP. 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. А GRU демонстрирует более быструю сходимость, обеспечивая лучшее качество аппроксимации за меньшее время.**Слайд 7. Заключение**  
Подводя итог, можно отметить следующее:  
– Внедрение архитектуры GRU в DEGANN позволяет существенно повысить точность аппроксимации, что критически важно при решении сложных дифференциальных уравнений.  
– Архитектура GRU в рамках пакета DEGANN демонстрирует более быструю сходимость модели, что сокращает время обучения при сохранении высокой точности.  
– Несмотря на немного более высокие затраты памяти, их увеличение не является существенным в современных вычислительных машинах, а преимущества в качестве аппроксимации делают выбор архитектуры оптимальным.

Таким образом, разработка и внедрение архитектуры GRU открывают перспективы для дальнейшего повышения эффективности нейросетевой аппроксимации в рамках пакета DEGANN.

На этом у меня все, Благодарю за внимание.

Готов ответить на вопросы, если таковы имеюся…