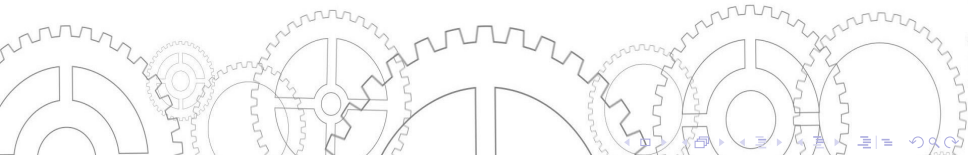




Применение нейронных сетей для улучшения решения уравнения переноса

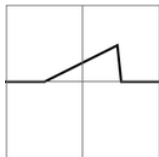
Выполнил: Климов О. Д., ФН2-71Б
под руководством д. ф.-м. н. Галанина М. П.



Формулировка

- 1 Необходимо для одномерного уравнения переноса реализовать алгоритм улучшения решения на основе искусственных нейронных сетей.
- 2 Протестировать работу программы на системе из 5 тестов (рис.1): левый и правый треугольники, прямоугольник, косинус, зуб.

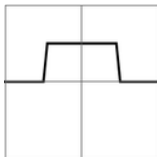
► Приложение 1



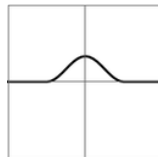
(a) Тест 1



(b) Тест 2



(c) Тест 3



(d) Тест 4



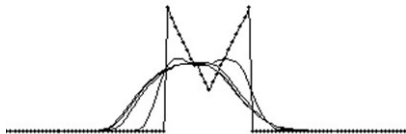
(e) Тест 5

Рис. 1: Система тестов

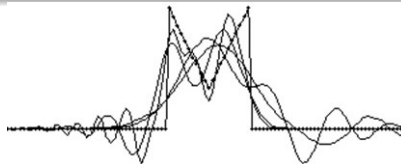
Задача Коши для уравнения переноса

$$\begin{cases} \frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = 0, \\ u(x, 0) = u_0(x), \end{cases} \quad \text{где } a = \text{const} > 0, \quad t \in (0, T). \quad (1)$$

Решение имеет вид $u = u_0(x - at)$ и заключается в сдвиге неизменного профиля по характеристикам. Численные схемы имеют ряд проблем.



(a) Пример схемы с диссипацией



(b) Пример схемы с дисперсией

Рис. 2: Пример точного (буква М) и численного решения уравнения переноса при некоторых различных числах Куранта. Иллюстрации взяты из [1, с.416]

Технологии нейронных сетей

Понятие нейронной сети

Нейронная сеть — это модель основанная на принципе организации биологических нейронных сетей. Ее можно интерпретировать совершенно по-разному.

Нейрон

Нейрон — единица обработки информации в нейронной сети, который можно представить функцией [2, с.28], [▶ Приложение 2](#)

$$y_k = \varphi(u_k + b_k), \quad u_k = \sum_{j=1}^m w_{kj} x_j \quad (2)$$

где x_1, x_2, \dots, x_m — входные сигналы,
 $w_{k1}, w_{k2}, \dots, w_{km}$ — веса связей нейрона k ,
 u_k — линейная комбинация входных воздействий, b_k — порог,
 φ — функция активации, y_k — выходной сигнал.

Функции активации

- $\varphi(x) = \frac{1}{1 + \exp^{-x}}$ — Сигмоида
- $\varphi(x) = \max(0, x)$ — ReLU

Структура сети

В нейронной сети нейроны упорядочено располагаются по слоям и связываются между собой направленными связями. Каждый нейрон принимает входные сигналы от нейронов предыдущего слоя, преобразует их с помощью весов и функции активации и передает результат на следующий слой.

- **Входной слой** — принимает $x = (x_1, x_2, \dots, x_m)$
- **Скрытые слои** — выполняют основную обработку информации
- **Выходной слой** — представляет результат $y = (y_1, y_2, \dots, y_n)$

Обучение сети

Обучением нейронной сети называется процесс настройки весов w_{ij} таким образом, минимизировать ошибку между результатом сети и целевым значением на обучающем наборе данных. [3, с.85]

Вводят функцию потерь $L(y_{pred}, y_{true}) = \frac{1}{n} \sum_{i=1}^n (y_{pred,i} - y_{true,i})^2$

Алгоритм метода обратного распространения ошибки

1 Прямой ход:

для входных данных x вычисляется результат y

2 Обратный ход: вычисляется

$\delta_k = \frac{\partial L}{\partial y_k} \cdot \varphi'(u_k)$ — ошибка на выходном слое

$\delta_j = \sum_{k=1}^n \delta_k w_{kj} \cdot \varphi'(u_j)$ — ошибка на скрытых слоях

3 Обновление параметров: $w_{kj} := w_{kj} - \eta \cdot \frac{\partial L}{\partial w_{kj}}$,

где $\frac{\partial L}{\partial w_{kj}} = \delta_k \cdot x_j$, η — скорость обучения

Сверточная нейронная сеть

Сверточная нейронная сеть — особый тип сетей, который характеризуется наличием операций свертки и пуллинга.

Свертка и пулинг ► Приложение 3

Операция свертки для матрицы x и ядра свертки w с размером $h \times h$ определяется следующим образом:

$$y_{ij} = \sum_{m=0}^{h-1} \sum_{n=0}^{h-1} x_{i+m, j+n} \cdot w_{mn}.$$

Для сохранения исходной размерности добавляют нулевой **паддинг** — обрамление изображение нулями. Без паддинга размер уменьшается на $(h - 1)$ пикс. по каждому измерению.

Слои **пуллинга** (подвыборки) производят функцию уменьшения размерности данных при сохранении признаков.

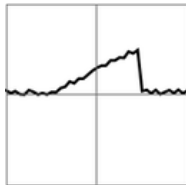
Реализация алгоритма

Методика подготовки наборов данных

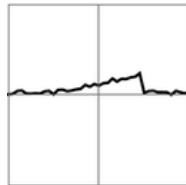
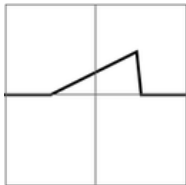
- Пары изображений «шумное-идеальное» созданы с помощью Wolfram Mathematica.
- 128x128 px — размер изображения, канал ч/б
- 3 теста — левый и правый треугольник, прямоугольник, тесты 4, 5 оставлены для тестирования
- 2 обучающих набора данных: однородные и неоднородные фигуры соответственно. Набор №1 — 500 пар, набор №2 — 2000 пар

Способ реализации программы

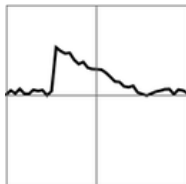
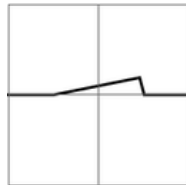
- Язык программирования Python, фреймворк TensorFlow (TF)
- Использование функций создания моделей и их обучения из TF
- Собственная реализация функций работы с наборами данных, их визуализации и расчета оценок
- Использование графического процессора для расчетов



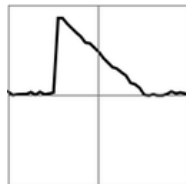
(a) Тест 1 из набора №1



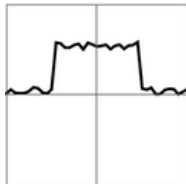
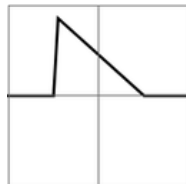
(d) Тест 1 из набора №2



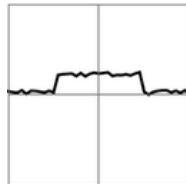
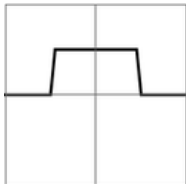
(b) Тест 2 из набора №1



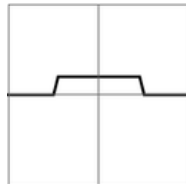
(e) Тест 2 из набора №2



(c) Тест 3 из набора №1



(f) Тест 3 из набора №2



Архитектура модели

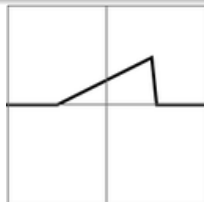
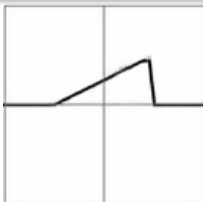
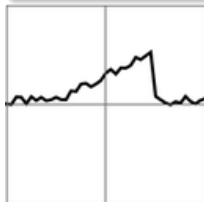
► Приложение 4

- **Входной слой:** Принимает изображения размером 128×128 с одним каналом (градации серого).
- **Сверточные слои:** Используются 4 последовательные свертки с ядром размером 3×3 . Каждая свертка сопровождается функцией активации ReLU.
- **Уменьшение размерности (MaxPooling):** После сверточных слоев применяется слой пулинга с блоком 2×2 .
- **Декодирующие слои (UpSampling):** На этапе восстановления разрешения используются транспонированные свертки с ядром 3×3 и операцией увеличения размера (UpSampling) для восстановления изображения до исходного размера.
- **Выходной слой:** Завершающий слой с функцией активации sigmoid, который возвращает выходное изображение.

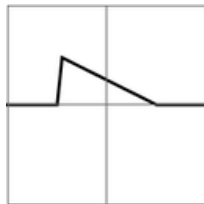
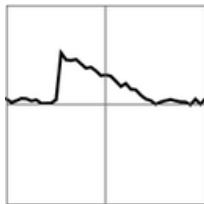
Результаты для набора №1

Расчет оценок

$\varkappa = L_{in} - L_{out}$, $\psi = L_{in}/L_{out}$, где $L_{in} = L(x, y_{true})$, $L_{out} = L(y_{pred}, y_{true})$

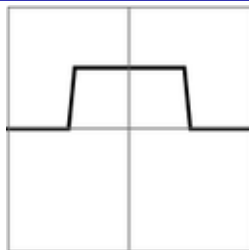
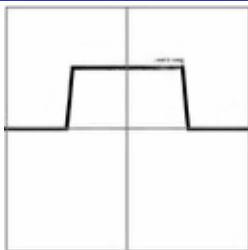
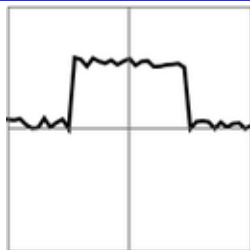


(a) Тест 1 для набора №1: $L_{in} = 0.0198$, $L_{out} = 0.0003$, $\varkappa = 0.0194$, $\psi = 60.07$

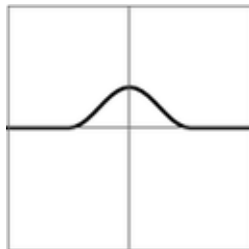
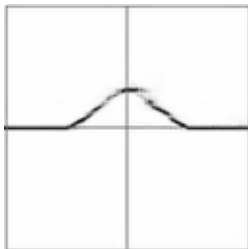
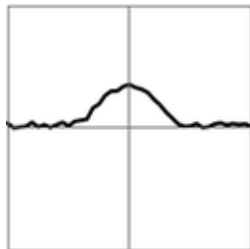


(b) Тест 2 для набора №1: $L_{in} = 0.0205$, $L_{out} = 0.0001$, $\varkappa = 0.0204$, $\psi = 254.75$

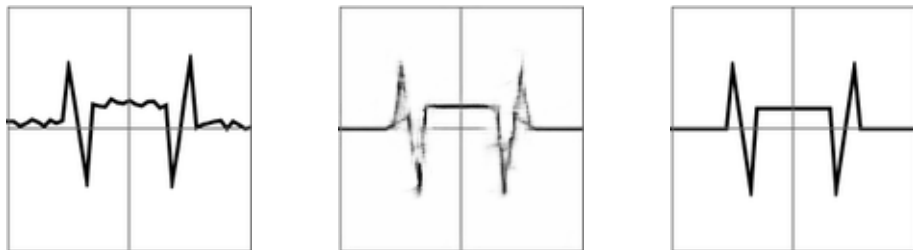
Результаты для набора №1



(с) Тест 3 для набора №1: $L_{in} = 0.0195$, $L_{out} = 0.0001$, $\varkappa = 0.0194$, $\psi = 220.01$



(d) Тест 4 для набора №1: $L_{in} = 0.0218$, $L_{out} = 0.0043$, $\varkappa = 0.017$, $\psi = 5.06$



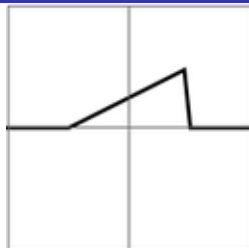
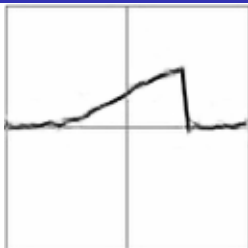
(е) Тест 5 для набора №1: $L_{in} = 0.0185$, $L_{out} = 0.0108$, $\kappa = 0.0076$, $\psi = 1.71$

Рис. 6: Результаты для набора №1 («входное-выходное-целевое» изобр.)

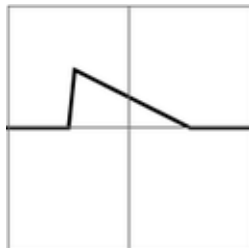
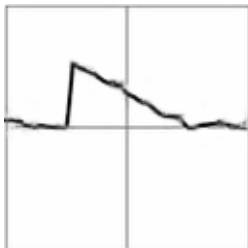
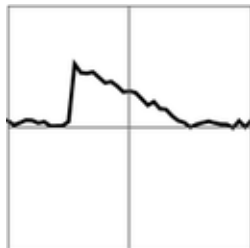
Таблица 1: Результаты тестирования алгоритма для набора данных №1

Тип теста	L_{in}	L_{out}	κ	ψ
Тест 1	0.0198	0.0003	0.0194	60.07
Тест 2	0.0205	0.0001	0.0204	254.75
Тест 3	0.0195	0.0001	0.0194	220.01
Тест 4	0.0218	0.0043	0.0170	5.06
Тест 5	0.0185	0.0108	0.0076	1.71

Результаты для набора №2

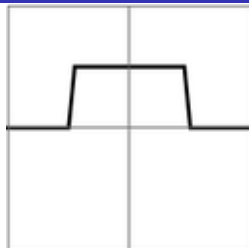
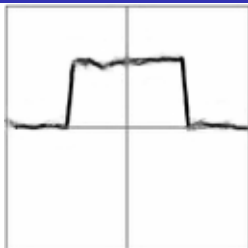
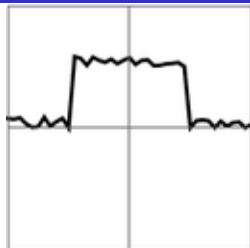


(a) Тест 1 для набора №1: $L_{in} = 0.0198$, $L_{out} = 0.0107$, $\varkappa = 0.0091$, $\psi = 1.84$

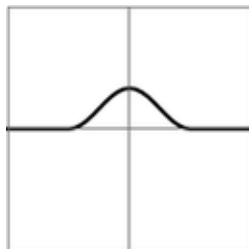
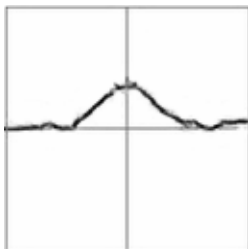
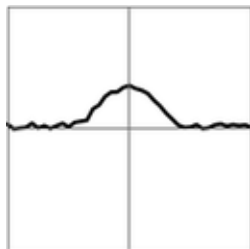


(b) Тест 2 для набора №1: $L_{in} = 0.0205$, $L_{out} = 0.0118$, $\varkappa = 0.0087$, $\psi = 1.73$

Результаты для набора №2



(с) Тест 3 для набора №1: $L_{in} = 0.0195$, $L_{out} = 0.0109$, $\varkappa = 0.0086$, $\psi = 1.78$



(d) Тест 4 для набора №2: $L_{in} = 0.0218$, $L_{out} = 0.0105$, $\varkappa = 0.0113$, $\psi = 2.08$



(d) Тест 5 для набора №2: $L_{in} = 0.0185$, $L_{out} = 0.0130$, $\kappa = 0.0055$, $\psi = 1.41$

Рис. 9: Результаты для набора №2 («входное-выходное-целевое» изобр.)

Таблица 2: Результаты тестирования алгоритма для набора данных №2

Номер теста	L_{in}	L_{out}	κ	ψ
Тест 1	0.0198	0.0107	0.0091	1.84
Тест 2	0.0205	0.0118	0.0087	1.73
Тест 3	0.0195	0.0109	0.0086	1.78
Тест 4	0.0218	0.0105	0.0113	2.08
Тест 5	0.0185	0.0130	0.0055	1.41








Перспективы задачи

- Дальнейшее исследование оптимизации модели
- Обучать модели на реальных данных, основанных на численных решениях определенных разностных схем
- Рассмотреть другие архитектуры моделей или, например, подход обучения без учителя

Итог

- Рассмотрены технологии нейронных сетей
- Разработан алгоритм улучшения решения уравнения переноса
- Созданы 2 набора данных
- Реализована программа и протестирована на системе тестов исходной задачи с примерами результатов

Список литературы

-  Галанин М.П., Савенков Е.Б. Методы численного анализа математических моделей. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2018. — 592 с. — ISBN 978-5-7038-4796-1
-  Хайкин С. Нейронные сети: полный курс., пер. с англ., 2-е изд. — М.: Издательский дом «Вильямс», 2006. — 1104 с. — ISBN 5-8459-0890-6.
-  Rashid T. Make Your Own Neural Network. — CreateSpace Independent Publishing Platform, 1st edition, 2016. — SAND96-0583.
-  Николенко С., Кадурын А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2022. — 476 с. — ISBN 978-5-4461-1537-2.
-  Шолле Ф. Глубокое обучение на Python. — СПб.: Питер, 2018. — 400 с. — ISBN 978-5-4461-0770-4.
-  Гафаров Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учебное пособие. — Казань: Изд-во Казан. ун-та, 2018. — 121 с.
-  Машинное обучение и TensorFlow. — СПб.: Питер, 2019. — 336 с. — ISBN 978-5-4461-0826-8.

Приложение 1: Начальные условия для системы тестов

$$(1a) : u_0(x) = \frac{x - l_1}{l_2 - l_1} \quad \text{— левый треугольник,}$$

$$(1b) : u_0(x) = \frac{l_2 - x}{l_2 - l_1} \quad \text{— правый треугольник,}$$

$$(1c) : u_0(x) = \frac{2}{3} \quad \text{— прямоугольник}$$

$$(1d) : u_0(x) = \frac{1}{3} \left(1 - \cos\left(\frac{2\pi(x - l_1)}{l_2 - l_1}\right) \right) \quad \text{— косинус}$$

$$(1e) : u_0(x) = \begin{cases} -\frac{2}{3}(l_{11} - l_1)(x - l_1) + 1, & l_1 \leq x < l_{11}, \\ \frac{1}{3}, & l_{11} \leq x \leq l_{22}, \\ \frac{2}{3}(l_2 - l_{22})(x - l_2) + 1, & l_{22} < x \leq l_2 \end{cases} \quad \text{— зуб,}$$

Приложение 2: Преобразование порога для модели нейрона

Использование порога b_k обеспечивает эффект аффинного преобразования выхода линейного сумматора u_k . В частности, в зависимости от того, какое значение принимает порог, положительное или отрицательное, можно двигать значения выхода нейрона. Обозначим $w_{k0} = b_k$ и преобразуем модель к следующему виду

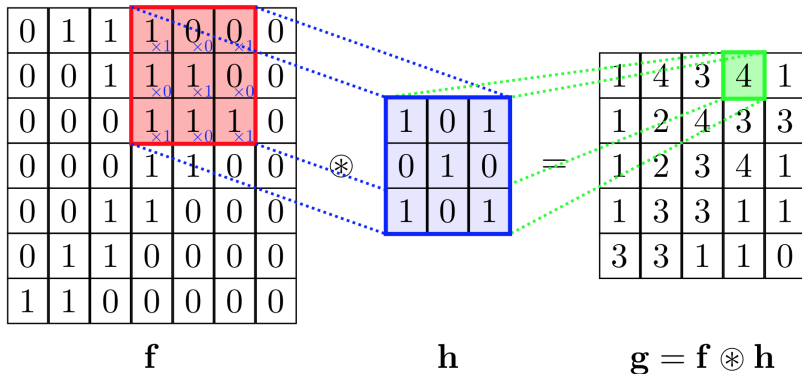
$$y_k = \varphi(v_k), \quad v_k = \sum_{j=0}^m w_{kj} x_j \quad (3)$$

Таким образом, в выражении (3) добавился новый синапс. Его входной сигнал и вес соответственно равны

$$x_0 = +1, \quad w_{k0} = b_k.$$

Это позволило трансформировать модель нейрона к виду, при котором добавляется новый входной сигнал фиксированной величины $+1$, а также появляется новый синаптический вес, равный пороговому значению b_k .

Приложение 3: Визуализация сверточной нейронной сети



$$y_{ij} = \sum_{m=0}^{h-1} \sum_{n=0}^{h-1} x_{i+m, j+n} \cdot w_{mn} \quad \text{— Свертка}$$

$$x_{i+m, j+n} = \sum_{m'=0}^{h-1} \sum_{n'=0}^{h-1} y_{i+m', j+n'} \cdot w_{m'n'} \quad \text{— Транспонированная свертка}$$

Приложение 4 ч1: Программа модели сети

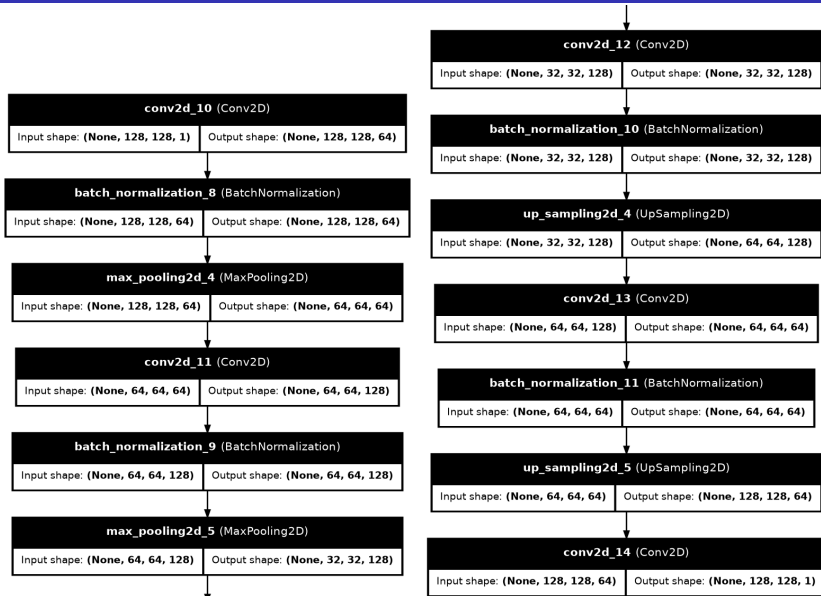


Рис. 10: Диаграмма модели

Приложение 4 ч2: Программа модели сети

Листинг 1: Модель нейронной сети с использованием TF

```
1 model = Sequential([
2
3     Conv2D(64, kernel_size=3, padding="same", activation="relu",
4         input_shape=(128, 128, 1)),
5     BatchNormalization(),
6     MaxPooling2D(pool_size=2),
7
8     Conv2D(128, kernel_size=3, padding="same", activation="relu"),
9     BatchNormalization(),
10    MaxPooling2D(pool_size=2),
11
12    Conv2D(128, kernel_size=3, padding="same", activation="relu"),
13    BatchNormalization(),
14    UpSampling2D(size=2),
15
16    Conv2D(64, kernel_size=3, padding="same", activation="relu"),
17    BatchNormalization(),
18    UpSampling2D(size=2),
19
20    Conv2D(1, kernel_size=3, padding="same", activation="sigmoid")
21 ])
22
23 model.compile(optimizer=SGD(learning_rate=0.01), loss="mse",
24     metrics=["mae"])
25 model.summary()
```

Приложение 4 ч3: Программа модели сети

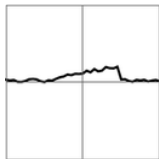
Особенности реализации

- Модель представляет собой сверточный автокодировщик
- Модель имеет 296.217 параметров
- Набор предварительно перемешивается с целью избежать корреляций между последовательными данными
- В процессе обучения в каждой эпохе вычисляется наименьшее значение потерь на валидационной выборке и, в случае увеличения ее значения, сохраняется предыдущая модель
- Использована опция предзагрузки данных, где для следующей эпохи обучения данные параллельно загружаются, пока обрабатывается текущая эпоха, причем учитывая специфику системы и загрузку процессора.
- Для набора №1 оптимальным выбрано 100 эпох обучения, для набора №2 около 70

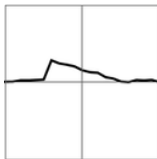
Приложение 5 ч.1: Доп. результаты, набор №1

Таблица 3: Результаты модели, обученной на наборе №1, при тестировании на валидационном наборе №2 (не входил в обучающую выборку)

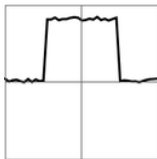
Номер теста	L_{in}	L_{out}	\varkappa	ψ
Тест 1	0.0113	0.0063	0.0050	1.79
Тест 2	0.0093	0.0071	0.0022	1.31
Тест 3	0.0104	0.0096	0.0008	1.08
Тест 4	0.0129	0.0052	0.0076	2.46
Тест 5	0.0095	0.0050	0.0045	1.90



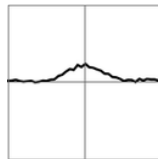
(a) Тест 1



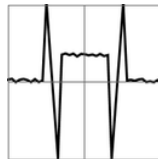
(b) Тест 2



(c) Тест 3



(d) Тест 4



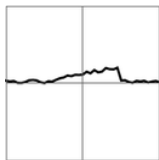
(e) Тест 5

Рис. 11: Тесты из набора №2, используемые в тестировании

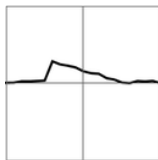
Приложение 6 ч.1: Доп. результаты, набор №1

Таблица 4: Результаты модели, обученной на наборе №2, при тестировании на валидационном наборе №2 (не входил в обучающую выборку)

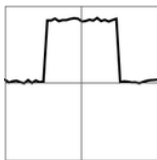
Номер теста	L_{in}	L_{out}	\varkappa	ψ
Тест 1	0.0113	0.0007	0.0106	17.19
Тест 2	0.0093	0.0005	0.0088	20.16
Тест 3	0.0104	0.0002	0.0102	50.89
Тест 4	0.0129	0.0019	0.0109	6.62
Тест 5	0.0095	0.0050	0.0045	1.90



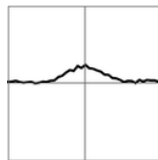
(a) Тест 1



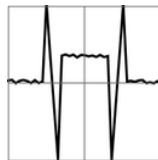
(b) Тест 2



(c) Тест 3



(d) Тест 4



(e) Тест 5

Рис. 12: Тесты из набора №2, используемые в тестировании