Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ Фундаментальные науки

КАФЕДРА Прикладная математика

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Применение нейронных сетей для улучшения решения уравнения переноса

Студент ФН2-71Б (Группа) _____ (Подпись, дата) О. Д. Климов (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) М. П. Галанин (И. О. Фамилия)

2024 г.

Оглавление

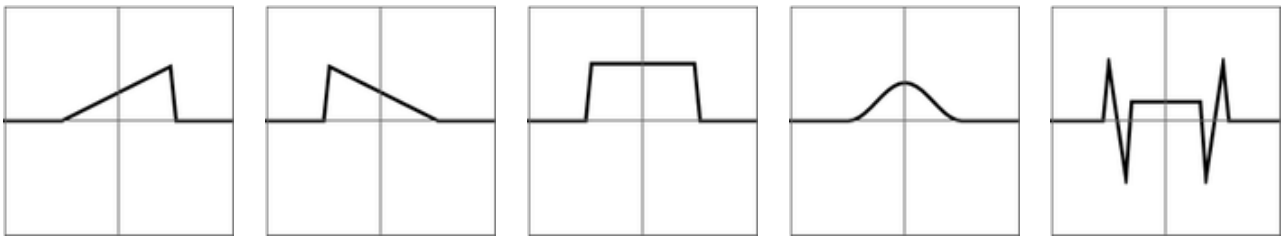
Введение

Технологиям нейронных сетей можно найти применение в совершенно разных прикладных задачах науки и техники. Например, в задачах улучшения или восстановления графика решений уравнения по неточно заданным данных о решениях. В силу нелинейности распознавания изображений нахождение точных алгоритмов для такой задачи испытывало ряд трудностей. Однако с развитием программирования и вычислительной техники стало возможным решать данную задачу методами нейронных сетей.

1. Постановка задачи

1.1. Формулировка

Необходимо для одномерного уравнения переноса реализовать и протестировать алгоритм улучшения решения на основе искусственных нейронных сетей. Для анализа работы программы использовать систему из 5 тестов (рис. 1): левый и правый треугольники, прямоугольник, косинус, зуб.



(a) Тест 1 (b) Тест 2 (c) Тест 3 (d) Тест 4 (e) Тест 5

Рис. 1. Система тестов

1.2. Пояснения

Рассмотрим задачу Коши для одномерного уравнения переноса следующего вида:

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

Приведем аналитическое решение. Уравнение можно записать в виде $u_t + au_x = 0$. Запишем и решим характеристическое уравнение:

$$\frac{dt}{1} = \frac{dx}{a} = \frac{du}{0} \quad \Longrightarrow \quad \begin{cases} u = C_1, \\ x = at + C_2 \end{cases} \quad \Longrightarrow \quad \begin{cases} C_1 = u, \\ C_2 = x - at \end{cases} \quad \Longrightarrow \quad u = \psi(x - at).$$

Применим граничные условия:

$$u(x,0) = u_0(x), \quad \Longrightarrow \quad \psi(x) = u_0(x), \quad \Longrightarrow \quad u = u_0(x - at).$$

Получим аналитическое решение уравнения $u = u_0(x - at)$. Решение заключается в сносе неизменного профиля по характеристикам.

Важнейшим свойством рассматриваемого решения будет являться сохранение начального профиля: если начальное решение представляет собой, например, профиль буквы «М», то оно будет сохранять его таким всегда.

Разностные схемы, аппроксимирующие такое уравнение, в той или иной степени искажают точное решение. Выделяют два эффекта потери формы — **диссипацию** и **дисперсию**.

Дисперсия возникает, когда различные гармонические компоненты волны (с различными частотами) распространяются с разными фазовыми скоростями. В результате форма волны искажается: волновой фронт «растягивается» или «смещается» . Так появляются ложные осцилляции (колебания) вокруг резких границ волнового фронта. [1, с.283]

Диссипация описывает процесс затухания волнового сигнала из-за рассеивания энергии. Она возникает при использовании схем с ненулевой вязкостью, из-за чего искажается амплитуда волны.

(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]

Задачей является исправление и приведение решения уравнения к менее искаженному виду. В силу особенностей выбранного алгоритма далее ограничимся слабым смещением и растяжением решения, таким, чтобы различить зрительно.

2. Технологии нейронных сетей

Для решения поставленной задачи обратимся к теории нейронных сетей. В следующих пунктах представлены общие идеи традиционных алгоритмов сетей. Также отдельно рассмотрен алгоритм сверточной нейронной сети.

2.1. Понятие нейронной сети

Нейронная сеть — это модель, основанная на принципе организации биологических нейронных сетей, т.е. лежащих в основе живых существ. Ее можно интерпретировать совершенно по-разному.

С математической точки зрения, нейронная сеть представляет собой математическую модель, которая соответствует некому процессу взаимодействия вычислительных блоков (нейронов). В нее входит многопараметрическая задача нелинейной оптимизации, появляющейся при установлении параметров сети. [2]

Также имеет место другой взгляд на математическую интерпретацию. Нейронную сеть можно рассматривать как нелинейное преобразование входного вектора, которое получается композицией всех функций сети(нейронов). Нелинейность получается за счет наличия нелинейных функций активации в каждом нейроне.

С точки зрения программирования, нейронная сеть представляет собой результат эволюции алгоритмов решения алгоритмически сложных задач с обработкой большого объема входных данных, а также возможность интеграции ускорителей вычислений (GPU, TPU). Именно здесь появляется связь нейронных сетей с машинным обучением.

Машинное обучение - это область компьютерных наук, к которой относят новую парадигму программирования, где на вход программы поступают данные и ответы, соответствующие этим данным, а результатом получается правила, которые можно применить к новым данным для получения оригинальных ответов. То есть в этой парадигме система обучается, в то время как в классическом программировании в программу вводятся правила и данные для обработки и получают ответ. Таким образом нейронные сети это один из подходов в рамках машинного обучения. [3, с.28]

Стоит отметит, что также отдельно выделяют область **глубокого обучения**, к которой относят нейронные сети с большим количеством слоев. В определенных случаях увеличение количества слоев сети приводит яркому развитию ее способностей к понимаю данных. [3, с.31]

Далее будем нейронную сеть будем рассматривать в виде приближенной математической модели, описывая включенные в нее алгоритмы.

2.2. Нейрон

Нейрон — единица обработки информации в нейронной сети. Он представляет из себя элемент, который вычисляет выходной сигнал из совокупности входных сигналов по определенному правилу. На блок-схеме рис. 3 показана модель нейрона, лежащего в основе искусственных нейронных сетей [4, с.28].

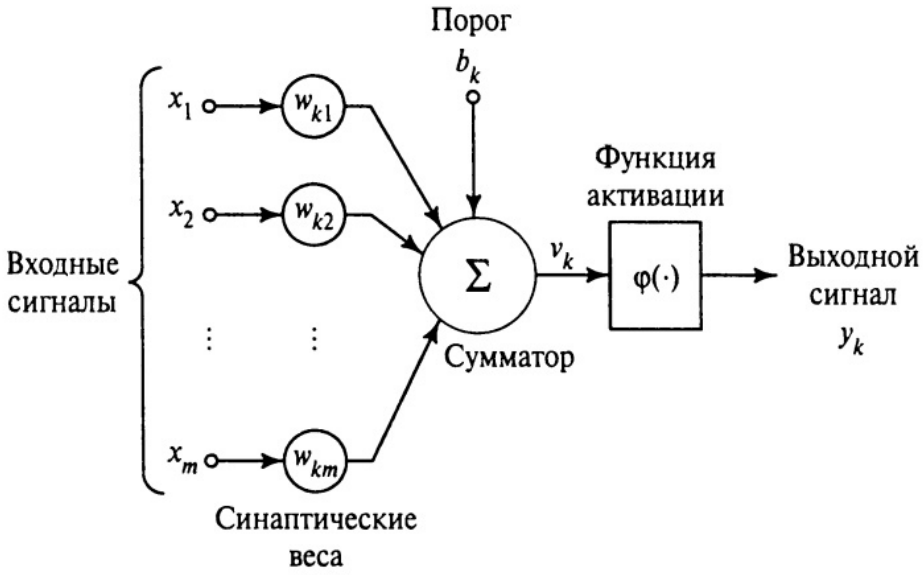


Рис. 3. Нелинейная модель k-го нейрона

В этой модели выделяют три основных элемента:

- **Набор связей**, каждый из которых характеризуется своим весом(силой). В частности, сигнал x_j на входе связи j, связанного с нейронном k, умножается на вес w_{kj}(первый индекс относится к рассматриваемому нейрону, а второй ко входному окончанию связи). Вес может иметь как положительные так и отрицательные значения.
- **Сумматор** складывает входные сигналы, взвешенные относительно соответствующих связей нейрона. Эту операцию можно описать как линейную комбинацию.
- **Функция активации** ограничивает амплитуду выходного сигнала нейрона. Обычно нормализованный диапазон амплитуд выхода нейрона лежит в интервале $[0, 1]$ или $[-1, 1]$.
- **Пороговый элемент** b_k, который отражает увеличение или уменьшение входного сигнала на функцию активации.

В математическом представлении функционирование нейрона k можно описать следующей парой соотношений

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

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

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

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

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. Хотя на первый взгляд кажется, что такая модель (3) отличается от изначальной (2), математически они эквивалентны.

Часто используют следующие функции активации

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

2.3. 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. Принято разделять три ключевых типа слоев: входной, скрытые и выходной слои.

Входной слой служит для приема исходных данных. Входной слой не выполняет вычислений и лишь передает данные на первый скрытый слой. Каждый нейрон этого слоя соответствует одному из признаков входного вектора

$$x = (x_1, x_2, ..., x_m). \tag{4}$$

Скрытые слои — это промежуточные слои между входным и выходным слоями. Они выполняют основную обработку информации и могут включать несколько уровней, каждый из которых преобразует данные через нелинейные функции активации.

Выходной слой производит итоговый результат работы сети. Количество нейронов в этом слое зависит исключительно от задачи. Как результат мы получаем

$$y = (y_1, y_2, ..., y_n). \tag{5}$$

Глубина сети определяется числом скрытых слоев, а **ширина** — количеством нейронов в каждом слое. Добавляя один или несколько скрытых слоев, мы можем выделить из входных данных закономерности более высокого порядка.

Архитектуру сети определяют связи между слоями. Существует множество вариантов расположить связи в сети, наиболее значимыми являются:

- **Полносвязная сеть** — каждый нейрон одного слоя соединен со всеми нейронами следующего.
- **Сверточные сети** — связи устанавливаются между локальными группами нейронов для анализа пространственных структур данных.
- **Рекуррентные сети** — связи могут иметь обратную направленность для учета временной зависимости.

2.4. Обучение

Обучением нейронной сети называется процесс настройки весов w_{ij} таким образом, чтобы сеть могла эффективно выполнять поставленную задачу. Обучение основано на минимизации ошибки между предсказанным выходом сети и целевыми значениями на обучающем наборе данных. Рассмотрим исключительно тип обучения с учителем, где нам известны целевые значения на входные данные. [5]

Сначала происходит инициализация параметров случайными значениями с использованием специальных методов. Производят прямое распространение — входные данные $x = (x_1, x_2, ..., x_m)$ передаются на вычисления в сеть. После выход последнего слоя y_{pred} сравнивается с целевым значением y_{true}.

Для сравнения y_{pred} и y_{true} вводят понятие **функции потерь** L, которая измеряет ошибку сети. Ее выбор зависит от типа задачи, часто используют **среднеквадратическую ошибку (MSE)**:

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_{pred,i} - y_{true,i})^2, \tag{6}$$

Одним из фундаментальных и наиболее часто применимых подходов к обучению сети является **метод обратного распространения ошибки**. Этот метод позволяет вычислить градиенты функции потерь по каждому параметру сети, используя правило цепочки для передачи ошибки от выходного слоя к предыдущим слоям.

Алгоритм следующий:

1) Прямое распространение: вычисление выходных значений нейронов и функции потерь.
2) Обратное распространение: вычисление частных производных функции потерь по параметрам сети.
3) Обновление параметров с использованием метода оптимизации (например, градиентного спуска).

Согласно соотношению (3) на каждом нейроне k вычисляется взвешенная сумма входов $u_k = \sum_{j=0}^{m} w_{kj}x_j$ и его активация с использованием функции активации φ.

Для вычисления ошибки на выходном слое используется частная производная функции потерь L по выходному значению y_k

$$\delta_k = \frac{\partial L}{\partial y_k} \cdot \varphi'(u_k), \tag{7}$$

где $\varphi'(u_k)$ — производная функции активации.

Ошибка передается обратно через связи, корректируя веса на каждом слое:

$$\delta_j = \sum_{k=1}^{n} \delta_k w_{kj} \cdot \varphi'(u_j), \tag{8}$$

где n — количество нейронов следующего слоя.

Частные производные функции потерь по весам и смещениям вычисляются следующим образом

$$\frac{\partial L}{\partial w_{kj}} = \delta_k \cdot x_j, \tag{9}$$

Для обновления параметров используется метод градиентного спуска [6, с.23]

$$w_{kj} := w_{kj} - \eta \cdot \frac{\partial L}{\partial w_{kj}}, \tag{10}$$

где η — скорость обучения.

Таким образом, метод обратного распространения ошибки позволяет сети корректировать свои параметры, постепенно минимизируя ошибку на входных данных. После повторения описанных шагов для определенного набора входных данных сеть настраивает свои параметры, чтобы лучше обобщать на новых данных.

2.5. Применение алгоритма

Нейронные сети представляют собой мощный инструмент для решения широкого спектра задач, которые можно условно разделить на два основных класса: задачи *обучения с учителем* и *обучения без учителя* [2, с.17].

При обучении с учителем на вход подается набор тренировочных примеров, которые обычно называют обучающим или **тренировочным набором данных**, и задача состоит в том, чтобы продолжить уже известные ответы на новый опыт, выраженные в виде **тестового набора данных**. Предполагаем, что данные, доступные для обучения, будут чем-то похожи на данные, на которых потом придется применять обученную модель, иначе никакое обобщение не будет возможно.

Задачей классификации называется задача на определение объекта к одному из известных классов. Обычно для ее решения в сети используют функцию активации «сигмоида» или «softmax», а функцию потерь «кросс-энтропия».

Задачей регрессии называется задача на определение значения некоей функции, у которой может быть бесконечно много разных значений. Обычно для ее решения используют линейную функцию активации и квадратичную функцию потерь.

Обучение без учителя, в отличие от обучения с учителем, не требует размеченных данных. Основной акцент делается на выявление скрытых закономерностей, кластеризацию или понижение размерности. Этот подход особенно полезен для анализа больших объемов информации, где разметка невозможна или слишком трудоемка.

Таким образом, выбор метода обучения зависит от природы задачи и доступности исходных данных. Правильное определение типа задачи является ключом к успешному применению нейронных сетей и построению эффективной модели.

3. Сверточная нейронная сеть

С увеличением объема данных и усложнением задач, связанных с обработкой изображений, классические полносвязные нейронные сети стали демонстрировать ограничения в производительности и устойчивости. Для эффективного анализа данных с пространственной структурой, таких как изображения, был разработан особый тип нейронных сетей — **сверточные нейронные сети** (СНС). Основное отличие заключается в способности автоматически извлекать признаки различного уровня сложности, минимизируя число параметров модели и улучшая обобщающую способность. Далее будем рассматривать двумерные входные и выходные данные.

Главной особенностью такой сети является наличие слоев свертки и пуллинга.

3.1. Сверточные слои

Свертка — ключевая операция в структуре этих сетей, предназначенная для выделения локальных признаков. Для входной матрицы x и ядра свертки w с размером $h \times h$ операция свертки 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 исходного размера изображения добавляют нулевой **паддинг** — обрамление изображение нулями. Без паддинга размер выходной матрицы уменьшается на $(h-1)$ пикс. по каждому измерению. С его учетом, ширина и высота выходного изображения y соответственно рассчитывается по формулам

$$H_{out} = \frac{H_{in} - h + 2P}{S} + 1, \quad W_{out} = \frac{W_{in} - h + 2P}{S} + 1,$$

где P — размер паддинга, S — шаг свертки.

Также к СНС относят наличие так называемых слоев пуллинга или подвыборки, которые производят функцию уменьшения размерности при сохранении признаков. Цель пуллинга — сократить количество параметров, уменьшить вычислительные затраты и сделать сеть более устойчивой к небольшим смещениям. В основном его алгоритм заключается в делении входного изображения на подматрицы и заменой каждой на ее максимальное или среднее значение. После операции пуллинга размерность изменяется по той же формуле, как и у свертки с учетом, что в данном случае h — размерность подматриц.

3.2. Преимущества

Сверточные нейронные сети обладают рядом преимуществ:

- **Снижение количества параметров**: ядра свертки совместно используют свои веса, что уменьшает общее число параметров, повышая эффективность обучения.
- **Инвариантность к локальным преобразованиям**: благодаря свёрткам и пулингу сеть устойчива к сдвигам, поворотам и масштабированию входных данных.

Таким образом, сверточные сети являются мощным инструментом для решения задач, требующих анализа данных с пространственной структурой, предоставляя возможность автоматически извлекать признаки и минимизировать риск переобучения.

4. Применение сверточной сети для решения поставленной задачи

Теперь, используя всю изложенную теорию, можем найти ее применение в поставленной задаче. Для программной реализации всех алгоритмов и функций будем использовать комплексную библиотеку машинного обучения TensorFlow, которая находится в открытом доступе для любого разработчика.

4.1. Методика подготовки данных

Первым делом необходимо описать задачу, выполняемую алгоритмом сети. Так как глобальное задание требует улучшить уже имеющееся некачественное решение, то можно сделать вывод о том, что входными данными и выходным результатом будет изображение.

Для обучения сети необходимо разработать и создать специальные наборы данных. Также нужно иметь и тестовый набор, для оценки результатов. Эти данные должны состоять из пар изображений: некачественного графика и идеального.

Наборы данных будем создать с помощью математического пакета компьютерной алгебры Wolfram Mathematica. Все изображения будем делать с одними свойствами: размер $128 \cdot 128$ пикс., область графика $[-1, 1] \times [-1, 1]$. Создавать будем дискретные графики, соединенные по точкам, используя функцию $ListLinePlot$.

Шум будем добавлять методом умножения координаты абсцисс на случайное числовое значение в диапазоне $[0.75, 1.25]$. Все это можно сделать, используя функцию $Randint$[75, 125], деленную на длину ее диапазона.

Используем следующие начальные условия, соответствующие тестам задачи:

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

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

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

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

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

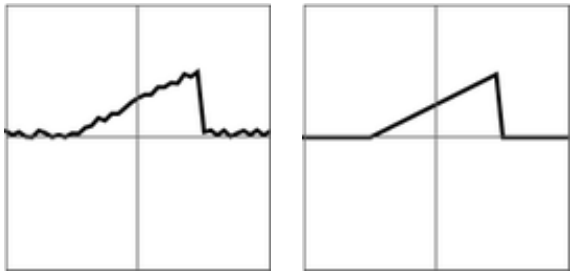
где l_1, l_2, l_{11}, l_{22} — различные параметры размера фигуры.

Таким образом создадим 2 набора данных, состоящих из тренировочной и тестовой выборок. Каждый из наборов содержит изображения, основанные на тестах 1-3 с добавленным шумом.

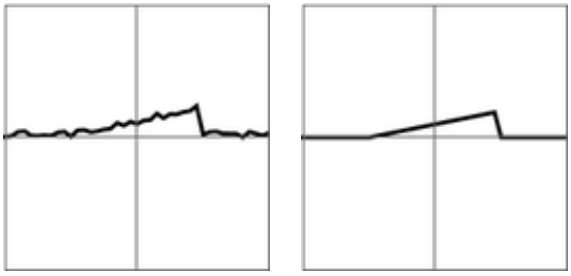
Набор №1 характеризуется тем, что фигуры имеют одни и те же параметры начального условия, соответствующего тесту. Он содержит около 500 изображений.

Набор №2 представляет более общий случай, где параметры начального условия могут немного изменяться, тем самым меняя высоту или ширину фигуры. Он содержит около 2000 изображений.

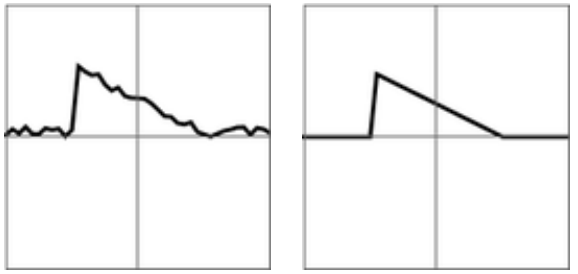
На рис. 1 показаны некоторые пары изображений из созданных наборов данных. Первое изображение (шумное) подается на вход, второе является его идеальной версией и целевым результатом.



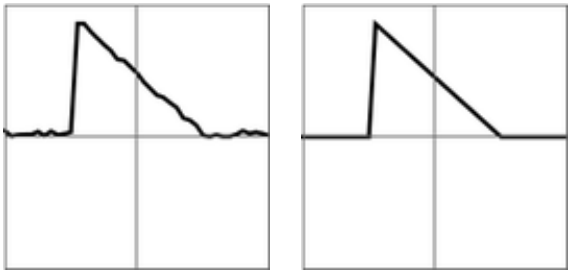
(a) Тест 1 из набора №1



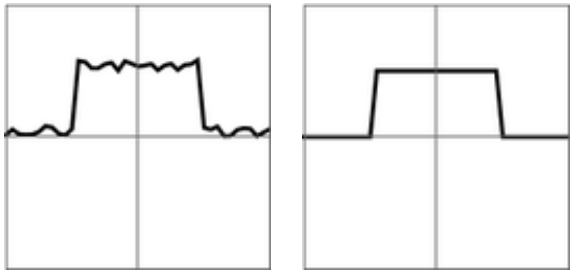
(d) Тест 1 из набора №2



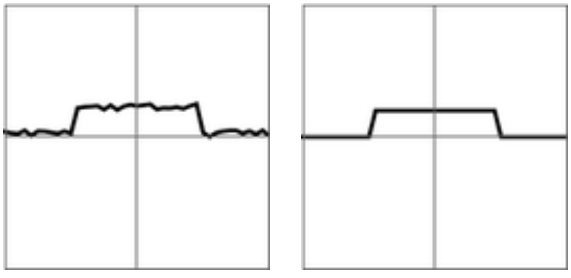
(b) Тест 2 из набора №1



(e) Тест 2 из набора №2



(c) Тест 3 из набора №1



(f) Тест 3 из набора №2

Рис. 4. Примеры пар изображений из наборов данных

4.2. Архитектура модели

Для решения задачи очистки зашумленных изображений была выбрана сверточная нейронная сеть (CNN), способная эффективно извлекать пространственные закономерности из изображений. Такие нейронные сети хорошо зарекомендовали себя в задачах обработки изображений благодаря их способности автоматически извлекать иерархические признаки с помощью сверток. В данном случае сеть обучается на парах «зашумленное изображение — чистое изображение», чтобы научиться удалять шум.

Архитектура сети состоит из нескольких сверточных слоев и механизмов декодирования для восстановления изображения. Рассмотрим основные компоненты сети:

- **Входной слой:** Принимает изображения размером 128×128 с одним каналом (градации серого).
- **Сверточные слои:** Используются 4 последовательные свертки с ядром размером 3×3 и шагом 1. Каждая свертка сопровождается функцией активации `ReLU`, что позволяет сети эффективно обучаться нелинейным зависимостям.
- **Уменьшение размерности (MaxPooling):** После нескольких сверточных слоев применяется слой пулинга с окном 2×2 и шагом 2 для уменьшения размерности карты признаков и сокращения вычислительных затрат.
- **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 сверточный слой с функцией активации `sigmoid`, который возвращает выходное изображение с диапазоном значений $[0, 1]$.

Каждый из этих компонентов обеспечивает извлечение и восстановление признаков изображения, что делает сеть способной эффективно устранять шум.

Для подготовки данных к обработке алгоритмом и преобразованием их в единый объект используется функция `create_denoising_dataset()`, которая:

- Загружает изображения из указанных папок.
- Преобразует изображения в формат 128×128.
- Нормализует данные в диапазон $[0, 1]$.

Для оптимизации используется обыкновенный алгоритм градиентного спуска с параметром скорости обучения $\eta = 0.01$, который обеспечивает быструю и стабильную сходимость. Функция потерь — среднеквадратичная ошибка. Количество эпох обучения было подобрано эмпирически с учетом темпов изменения значения функции потерь в зависимости от эпох.

Как следствие оптимальным для набора №1 оптимальное количество эпох обучения составило около 100, а для набора №2 выбрано 70.

Таким образом, сеть минимизирует разницу между предсказанным и целевым изображением, что позволяет эффективно удалять шум из изображений.

Листинг 1. Модель нейронной сети с использованием TensorFlow

```
model = Sequential([

    Conv2D(64, kernel_size=3, padding="same", activation="relu", input_shape=(128, 128,
        1)),
    BatchNormalization(),
    MaxPooling2D(pool_size=2),

    Conv2D(128, kernel_size=3, padding="same", activation="relu"),
    BatchNormalization(),
    MaxPooling2D(pool_size=2),

    Conv2D(128, kernel_size=3, padding="same", activation="relu"),
    BatchNormalization(),
    UpSampling2D(size=2),

    Conv2D(64, kernel_size=3, padding="same", activation="relu"),
    BatchNormalization(),
    UpSampling2D(size=2),

    Conv2D(1, kernel_size=3, padding="same", activation="sigmoid")
])

model.compile(optimizer=SGD(learning_rate=0.01), loss="mse", metrics=["mae"])
model.summary()
```

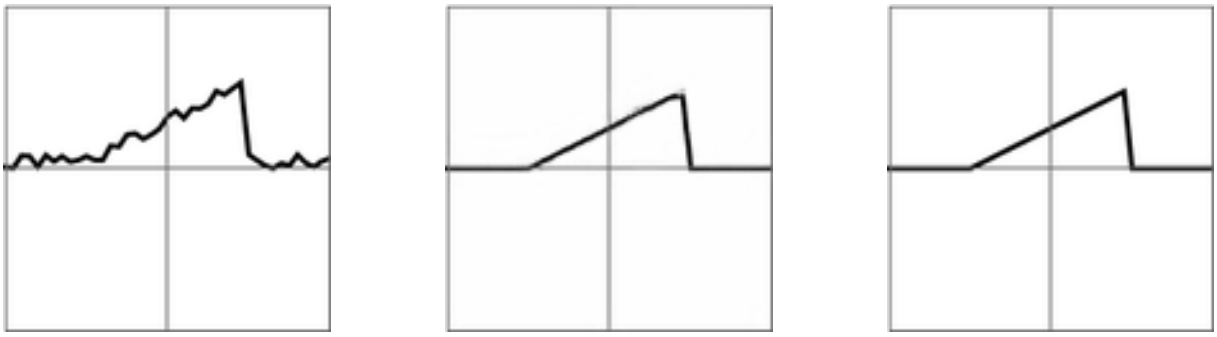
5. Анализ результатов

Для оценки работы алгоритма будем использовать значение функции потерь для различных тестовых примеров. Оценивать будем работу сети как на тестах 1-3, так и на тестах 4-5 из исходного задания.

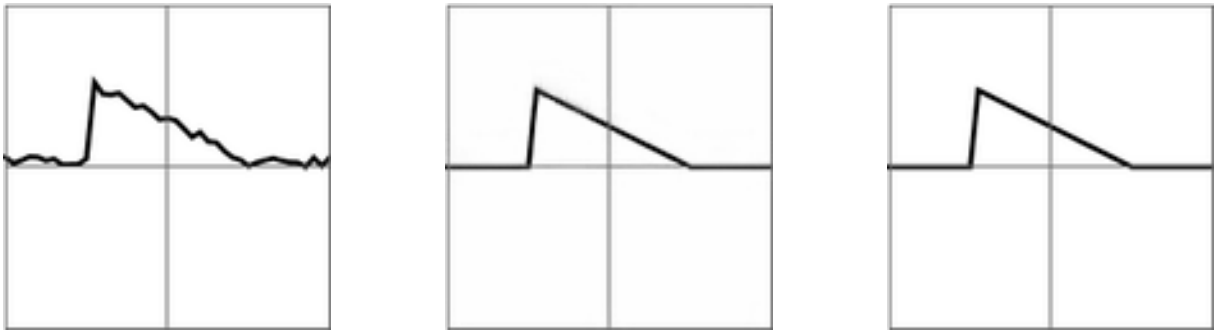
Введем величины $\varkappa = L_{in} - L_{out}$ и $\psi = L_{in}/L_{out}$, где L_{in} — значение функции потерь на входных данных, L_{out} — ее значение на результате сети. С помощью этих характеристик будем измерять величину улучшения решения алгоритмом.

5.1. Набор №1 с фигурами одного размера

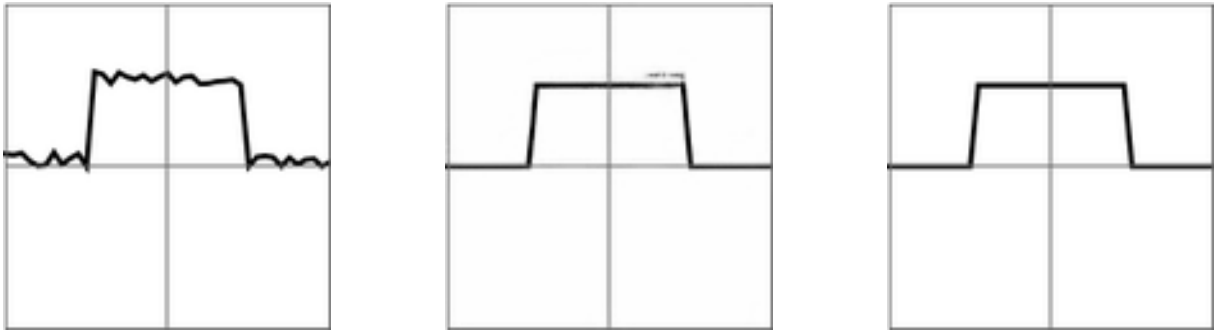
Первый набор данных характеризуется неизменностью размеров фигур, входящих в него. Приведем результаты работы сети на этом наборе.



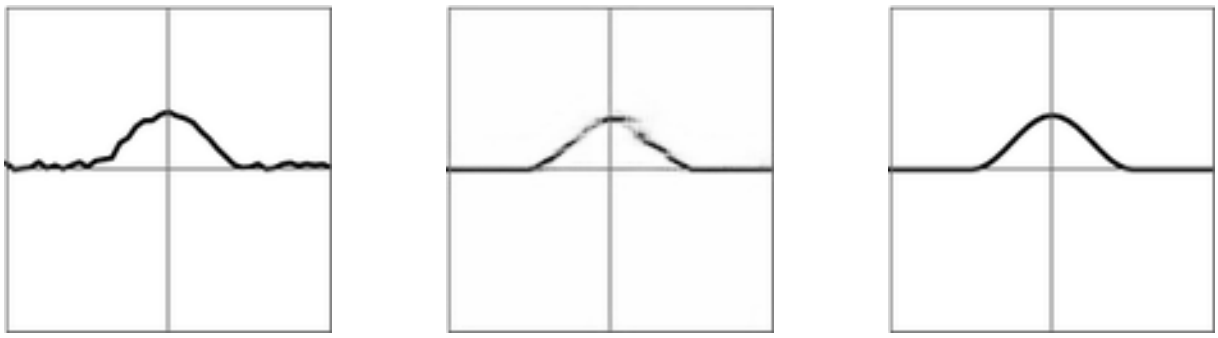
(a) Тест 1 для набора №1: $L_{in} = 0.0198, L_{out} = 0.0003, \varkappa = 0.0194, \psi = 60.07$



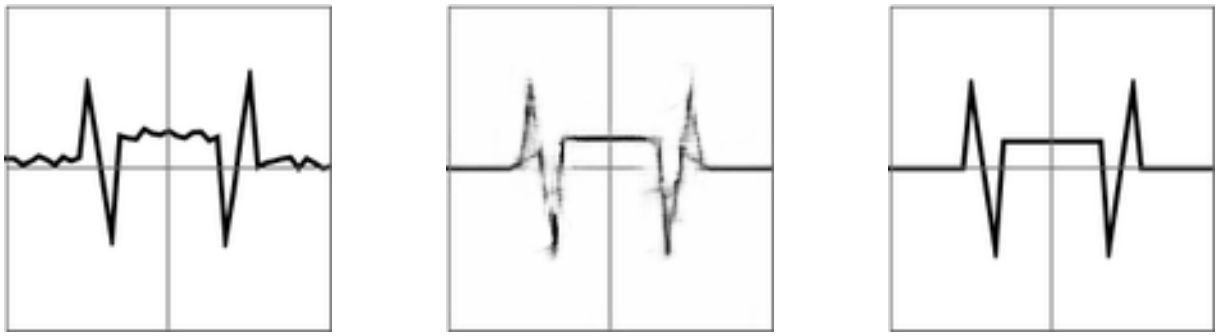
(b) Тест 2 для набора №1: $L_{in} = 0.0205, L_{out} = 0.0001, \varkappa = 0.0204, \psi = 254.75$



(c) 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: $L_{in} = 0.0218, L_{out} = 0.0043, \varkappa = 0.017, \psi = 5.06$



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

Рис. 6. Результаты тестирования алгоритма для набора данных №1
в виде «входное-выходное-идеальное» изображение

По результатам можно сделать вывод об успешной работе всего алгоритма. Набор данных для тестирования не входит в обучающую выборку. Тесты 1-3, из которых составлялась эта выборка, показали наивысший результат, в то время как незнакомые алгоритму тесты 4 и 5 имеют наименьшее улучшение.

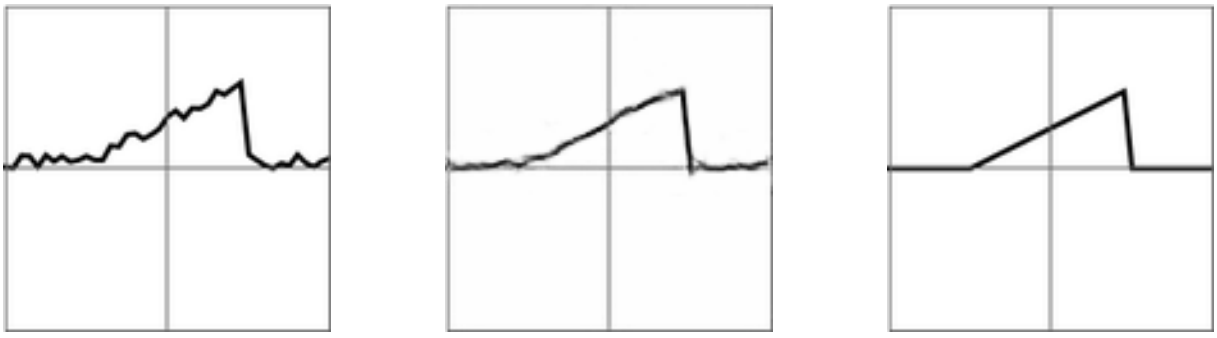
Также эти же результаты приведены в следующей таблице

Таблица 1. Результаты тестирования алгоритма для набора данных №1

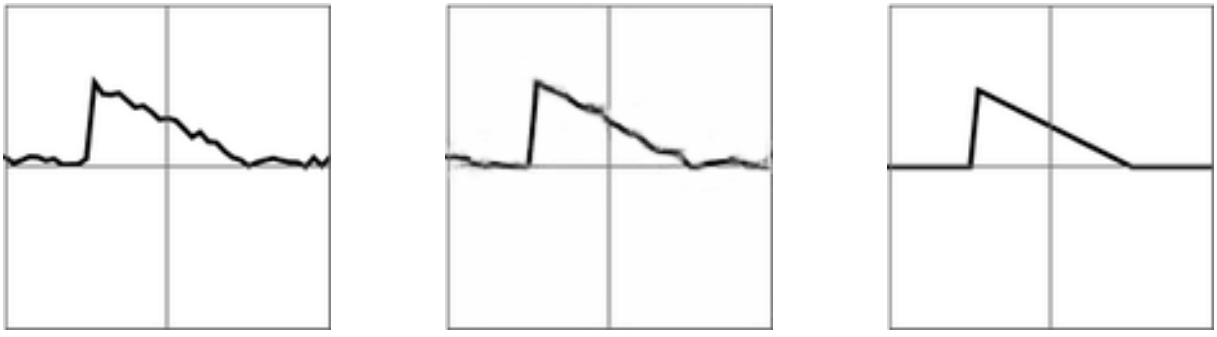
Тип теста	$\mathbf{L_{in}}$	$\mathbf{L_{out}}$	$\varkappa$	ψ
Тест 1	0.0198	0.0003	0.0194	60.07
Тест 2	0.0205	0.0001	0.0204	254.75
Тест 3	0.0195	0.0001	0.0194	220.01
Тест 4	0.0218	0.0043	0.0170	5.06
Тест 5	0.0185	0.0108	0.0076	1.71

5.2. Набор №2 с фигурами вариативного размера

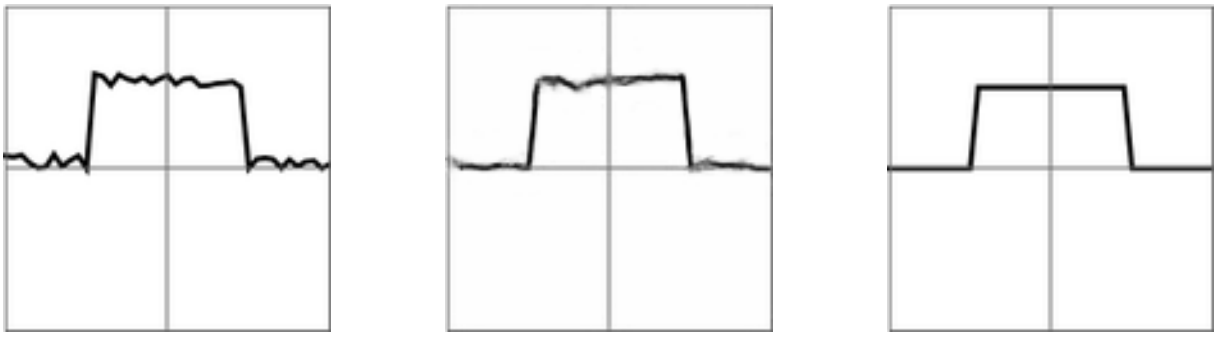
Второй набор данных характеризуется тем, что размеры фигур могут незначительно меняться из-за изменения параметров начальных условий. Приведем результаты работы сети на этом наборе.



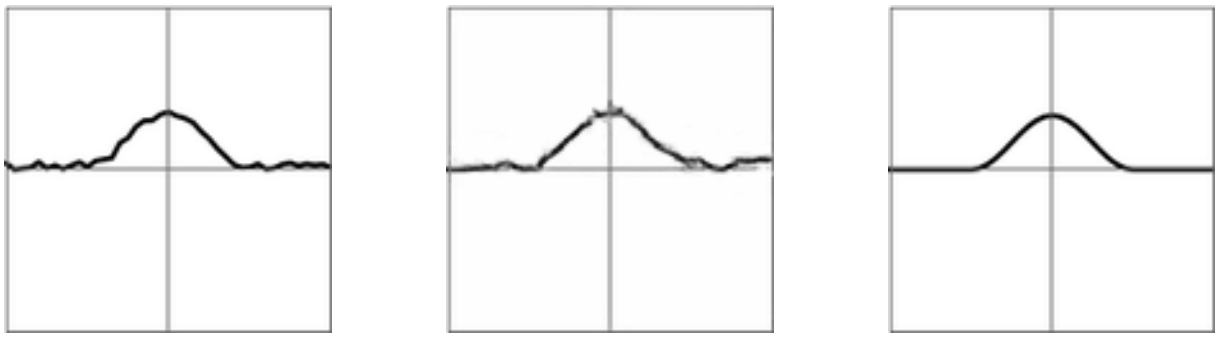
(a) Тест 1 для набора №1: $L_{in} = 0.0198, L_{out} = 0.0107, \varkappa = 0.0091, \psi = 1.84$



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



(c) 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: $L_{in} = 0.0185, L_{out} = 0.0130, \varkappa = 0.0055, \psi = 1.41$

Рис. 8. Результаты тестирования алгоритма для набора данных №2
в виде «входное-выходное-идеальное» изображение

По результатам второго набора можно сделать вывод о том, что в случае увеличенного диапазона размеров фигур алгоритм работает хуже.

Также эти же результаты приведены в следующей таблице

Таблица 2. Результаты тестирования алгоритма для набора данных №2

Номер теста	$\mathbf{L_{in}}$	$\mathbf{L_{out}}$	$\varkappa$	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

На основе приведенных результатов можно сделать общий вывод о высокой эффективности работы разработанной нейронной сети. Значение функции потерь на выходе модели (L_{out}) значительно ниже, чем на входе (L_{in}) во всех тестах, что указывает на успешное подавление шума и корректную обработку численного решения. Величина $\varkappa = L_{in} - L_{out}$ положительна во всех тестах, демонстрируя улучшение качества данных после обработки сетью. Например, для теста 2 величина $\varkappa$ составляет 0.0204, что свидетельствует о существенном снижении ошибки.

Стабильность работы модели подтверждается тем, что значения $\varkappa$ находятся в пределах одинакового порядка ($\approx 0.018-0.020$) для всех тестов, что говорит о надежности сети при обработке различных входных данных. Однако стоит отметить, что в тестах 4 и 5 наблюдаются более высокие значения L_{out} (0.0037 и 0.0118 соответственно). Это происходит из-за того, что эти фигуры не были включены в обучающую выборку.

Таким образом, можно сделать вывод, что модель эффективно справляется с задачей подавления шума на численных решениях уравнения переноса, что подтверждается значительным снижением функции потерь и стабильностью результатов на различных тестах.

6. Актуальность и перспективы задачи

Задача устранения шума с численных решений уравнения переноса с использованием нейронных сетей является актуальной в контексте широкого применения численных методов в моделировании физических процессов. Уравнение переноса описывает разнообразные явления, такие как распространение тепла, движение жидкостей и газов, а также перенос загрязняющих веществ в атмосфере и гидросфере. Однако численные методы при использовании приближенных схем часто сопровождаются возникновением численного шума, артефактов и дисперсии, что снижает точность моделирования и усложняет дальнейший анализ.

Использование сверточных нейронных сетей для устранения шума обладает рядом преимуществ:

- **Автоматическое устранение артефактов**: Нейронные сети могут обучаться выделять полезные сигналы и игнорировать шум без ручной настройки параметров фильтрации.
- **Высокая гибкость**: Сеть может адаптироваться к различным типам шума и условиям моделирования, что делает её универсальным инструментом для работы с численными данными. Автоматическое улучшение качества численного решения уменьшает потребность в дополнительных фильтрационных процедурах, что ускоряет процесс анализа.

Таким образом, разработка и внедрение методов устранения шума на основе нейронных сетей открывает новые перспективы для численного моделирования сложных процессов, позволяя достигать более высокой точности и надежности результатов.

Заключение

В данной работе рассмотрена задача устранения шума с численных решений уравнения переноса с помощью нейронных сетей. Применение современных методов машинного обучения, в частности сверточных нейронных сетей, позволило разработать эффективный подход к улучшению точности численных решений.

В ходе исследования реализована архитектура сверточной нейронной сети, способная обрабатывать дискретные численные данные, подавляя шум и восстанавливая исходную структуру решения. Подробно рассмотрены ключевые этапы построения модели, включая выбор слоев, функции активации, методы оптимизации и обучение сети на синтетических данных.

Анализ полученных результатов продемонстрировал, что предложенная модель способна эффективно устранять шум и значительно улучшать качество численных решений уравнения переноса. Это подтверждает перспективность использования нейросетевых подходов в задачах численного моделирования.

Применение разработанного метода может найти широкий отклик в различных научных и инженерных областях, где требуется высокоточная обработка численных данных. Перспективы дальнейших исследований включают оптимизацию архитектуры сети, адаптацию модели для различных типов уравнений и развитие методов реального времени для обработки больших объемов данных.

Список использованных источников

1. 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
2. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2022. — 476 с. — (Библиотека программиста). — Библиогр.: с. 451–476. — ISBN 978-5-4461-1537-2.
3. Шолле Ф. Глубокое обучение на Python. — СПб.: Питер, 2018. — 400 с. — (Библиотека программиста). — ISBN 978-5-4461-0770-4.
4. Хайкин С. Нейронные сети: полный курс.:,пер. с англ., 2-е изд. — М.: Издательский дом «Вильямс», 2006. — 1104 с. — ISBN 5-8459-0890-6.
5. Rashid T. Make Your Own Neural Network. — CreateSpace Independent Publishing Platform, 1st edition, 2016. — SAND96-0583.
6. Гафаров Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учебное пособие. — Казань: Изд-во Казан. ун-та, 2018. — 121 с.
7. Машинное обучение и TensorFlow. — СПб.: Питер, 2019. — 336 с. — (Библиотека программиста). — ISBN 978-5-4461-0826-8.
8. Вьюгин В.В. Элементы математической теории машинного обучения: учебное пособие для вузов. — М.: МФТИ-ИППИ РАН, 2010. — 231 с. — Библиогр.: с. 229–231. — ISBN 978-5-7417-0339-7.
9. Вьюгин В.В. Математические основы машинного обучения и прогнозирования. — М., 2018. — 484 с. — ISBN 978-5-4439-2014-6