Научное общество учащихся «Эврика»

МУНИЦИПАЛЬНОе АВТОНОМНОЕ ОБЩЕобразовательное учреждение «ЛИЦЕЙ № 38»

Советского района г. Нижнего Новгорода

“Применение нейронных сетей для распознавания изображений”

Выполнил:

Соколов Андрей Валерьевич,

Ученик 10 “Д” класса

Руководитель:

Кожин Александр Николаевич

учитель информатики

Нижний Новгород

2024

Оглавление

[1. Введение 3](#_Toc190004412)

[2. Целеполагание 5](#_Toc190004413)

[3. Основная часть 6](#_Toc190004414)

[3.1 Персептроны 6](#_Toc190004415)

[3.2 Сигмовидные нейроны 7](#_Toc190004416)

[3.3 Архитектура нейронных сетей 10](#_Toc190004417)

[3.4 Простая сеть для классификации рукописных цифр 12](#_Toc190004418)

[3.5 Алгоритм градиентного спуска 16](#_Toc190004419)

[3.6 Алгоритм обратного распространения 19](#_Toc190004420)

[3.7 Использования нейронной сети для разбора рукописных чисел 21](#_Toc190004421)

[3.8 Распознавание рукописных цифр с помощью Matlab 23](#_Toc190004422)

[3.8.1 Convolution2dLayer 23](#_Toc190004423)

[3.9 Исследование сети семантической сегментации с помощью Grad-CAM в Matlab 28](#_Toc190004424)

[4. Список Литературы 35](#_Toc190004425)

# Введение

Математическая модель нейронных сетей была известна довольно давно. Можно отметить несколько событий, которые позволили найти широкое применение данной модели во всех сферах человеческой жизни.

В 2009 году Раджат Райна, Ананд Мадхаван и Эндрю Ын опубликовали статью «Крупномасштабное глубокое обучение без учителя с использованием графических процессоров», где представили идею использования GPU для обучения больших нейронных сетей.

Примерно с этого момента времени нейронные сети начали активно развиваться, поскольку появились достаточно производительные электронные вычислители, такие как графические процессоры.

В 2011 году Юрген Шмидхубер, Дан Клаудиу Чирешан, Ули Майер и Джонатан Маши разработали первую сверточную нейронную сеть, которая превзошла людей в соревновании по распознаванию дорожных знаков в Германии. В этом же году Apple выпустила Siri – голосовую помощницу, которая может генерировать ответы и выполнять действия в ответ на голосовые запросы.

Таким образом, развитие нейронных сетей позволило использовать их в различных сферах человеческой жизни заменяя, человеческий труд. Количество таких сфер стремительно увеличивается по сей день.

В качестве примера, на основе которого будут рассмотрены принципы работы нейронных сетей, было выбрано распознавание рукописного текста.

Сосредоточившись на распознавании рукописного текста, легко понять, как работает нейронная сеть в целом, так как распознавать написанные от руки цифры непросто, но не настолько сложно, чтобы требовать чрезвычайно сложного решения или огромных вычислительных мощностей

По ходу дела будут разобраны многие ключевые идеи о нейронных сетях, включая два важных типа искусственных нейронов (персептрон и сигмовидный нейрон), а также стандартный алгоритм обучения нейронных сетей, известный как стохастический градиентный спуск.

# Целеполагание

Цель проекта:

* Реализация алгоритмов распознавания рукописных цифр и классификации изображений с использований нейронных сетей

Задачи проекта:

* Изучение структуры нейронной сети
* Изучение алгоритма градиентного спуска
* Изучение разбора рукописных цифр с использованием нейронных сетей на языке python
* Реализация алгоритма разбора рукописных цифр с использованием нейронных сетей на языке python
* Изучение разбора рукописных цифр с использованием нейронных сетей в среде Matlab
* Реализация алгоритма разбора рукописных цифр с использованием нейронных сетей на языке Matlab
* Изучение задачи классификации объектов изображения в среде Matlab
* Реализация алгоритма классификации объектов изображения в среде Matlab

# Основная часть

## Персептроны

Персептрон - это вид нейрона, который принимает несколько двоичных входных сигналов …, и выдает один двоичный выходной сигнал:

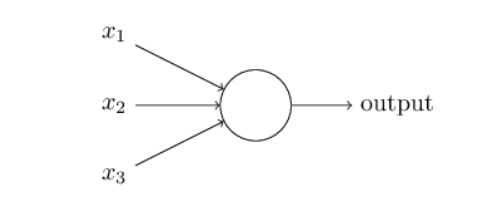


Рисунок 1

В приведенном примере персептрон имеет три входа: . В общем, у него может быть больше или меньше входов. Были введены веса, …- действительные числа, выражающие важность соответствующих входных данных для результата.

Выходной сигнал нейрона, равный 0 или 1, определяется тем, является ли взвешенная сумма меньше или больше некоторого порогового значения(threshold). Как и веса, пороговое значение — это действительное число, которое является параметром нейрона. Выражаясь более точными алгебраическими терминами:

Упростим неравенство, записав в виде произведения, , где и - векторы, весов и входных данных. Также заменим threshold на , где – это смещение персептрона противоположное пороговому значению. Используя все изменения можно переписать правило персептрона:

Смещение () — это показатель того, насколько легко заставить персептрон сработать или наоборот. Дальше больше не будет использоваться пороговое значение, вместо него везде будет смещение(b).

## Сигмовидные нейроны

Проблема персептронов в том, что небольшое изменение веса или смещения любого отдельного персептрона в сети иногда может привести к тому, что выходные данные этого персептрона полностью изменятся. Это изменение может привести к тому, что поведение остальной сети полностью изменится каким-то очень сложным образом. Это затрудняет понимание того, как постепенно изменять веса и смещения, чтобы сеть приближалась к желаемому поведению. Можно решить эту проблему, внедрив новый тип нейронов, называемый сигмовидным нейроном. Сигмовидные нейроны похожи на персептроны, но модифицированы таким образом, что небольшие изменения их веса и смещения вызывают лишь небольшое изменение их выходных данных. Это ключевой факт, который позволит сети сигмовидных нейронов обучаться.

Изобразим сигмовидные нейроны так же, как изображали персептроны:

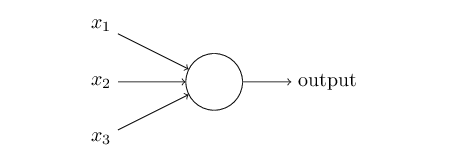


Рисунок 2

Как и у персептрона, у сигмовидного нейрона есть входы … . Но вместо того, чтобы принимать значение 0 или 1 , эти входные данные могут быть любые значениями от 0 до 1 . Так, например, 0,638… является допустимым входным сигналом для сигмовидного нейрона. Также, как и у персептрона, у сигмовидного нейрона есть веса для каждого входного сигнала, ... , и общее смещение, b . Но выходной сигнал не равен 0 или 1 . Вместо этого это он равен , где называется сигмовидной функцией, которая определяется как:

Чтобы представить все это более наглядно, выходные данные сигмовидного нейрона с входами ... , весами ... и смещением равны:

поведение сигмовидного нейрона также очень близко к поведению персептрона. Только когда небольшого размера, наблюдается значительное отклонение от модели персептрона. График функции :

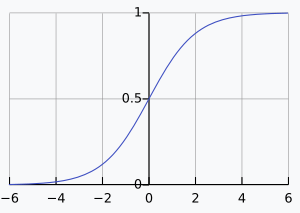


Рисунок 3

Используя фактическую -функцию, получим сглаженный персептрон. Действительно, решающим фактом является гладкость функции, а не ее детализированная форма. Плавность означает, что небольшие изменения в весах и в смещении приведут к небольшому изменению в выходных данных нейрона. Фактически можно представить математически формулой:

Где и обозначают частные производные выходных данных по отношению к и . Хотя приведенное выше выражение со всеми частными производными выглядит сложным, на самом деле оно говорит о чем-то очень простом: является линейной функцией изменений и в весах и смещениях. Такая линейность позволяет легко выбирать небольшие изменения в весах и смещениях для достижения любого желаемого небольшого изменения выходных данных. Таким образом, хотя сигмовидные нейроны обладают во многом тем же качественным поведением, что и персептроны, с их помощью гораздо проще понять, как изменение весов и смещений повлияет на выходные данные.

сигмовидными могут иметь в качестве выходных данных любое действительное число от 0 до 1 , поэтому такие значения, как 0,173… и 0,689… являются допустимыми выходными данными. Это может быть полезно, например, при использовании выходного значение для представления средней интенсивности пикселей в изображении, вводимом в нейронную сеть.

## Архитектура нейронных сетей

Предположим, у нас есть нейронная сеть:

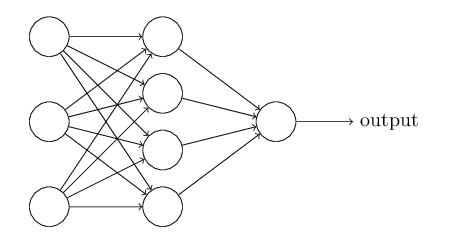


Рисунок 4

Как упоминалось ранее, самый левый слой в этой сети называется входным слоем, а нейроны внутри слоя называются входными нейронами. Самый правый или выходной слой содержит выходные нейроны или, как в данном случае, один выходной нейрон. Средний слой называется скрытым, поскольку нейроны в этом слое не являются ни входными, ни выходными данными. В сети, описанной выше, есть только один скрытый слой, но в некоторых сетях есть несколько скрытых слоев. Например, следующая четырехуровневая сеть имеет два скрытых слоя:

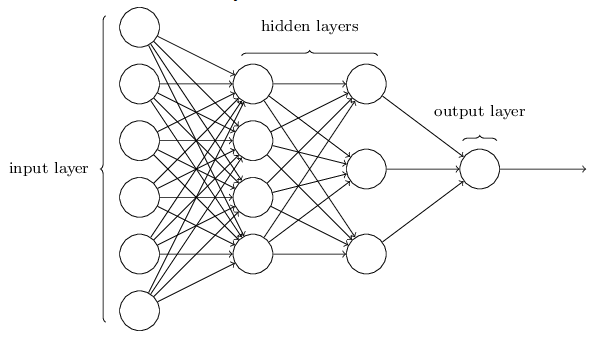


Рисунок 5

Такие многослойные сети иногда называют многослойными персептронами или MLP, несмотря на то, что они состоят из сигмовидных нейронов, а не из персептронов.

Структура входного и выходного уровней в сети часто проста. Например, предположим, что пытаемся определить, изображена ли на изображении, написанном от руки, цифра "9" или нет. Естественным способом проектирования сети является кодирование интенсивности пикселей изображения во входных нейронах. Если изображение представляет собой изображение в оттенках серого размером 64 на 64, то у нас будет 4096 = 64×64 входных нейрона с соответствующим масштабом интенсивности от 0 до 1. Выходной слой будет содержать только один нейрон с выходными значениями менее 0,5 что указывает на то, что "входное изображение не соответствует 9", и значениями, превышающими 0,5 указывает на то, что "входное изображение равно 9 ".

В то время как проектирование входных и выходных слоев нейронной сети часто является простым, то описать процесс проектирования скрытых слоев с помощью нескольких простых эмпирических правил невозможно . место этого исследователи нейронных сетей разработали множество эвристик проектирования для скрытых слоев, которые помогают людям добиться от своих сетей желаемого поведения. Например, такие эвристики могут быть использованы для определения соотношения количества скрытых слоев и времени, необходимого для обучения сети.

## Простая сеть для классификации рукописных цифр

Определившись с нейронными сетями, вернемся к распознаванию рукописного текста. Можно разделить задачу распознавания написанных от руки цифр на две подзадачи. Во-первых, нам нужен способ разбить изображение, содержащее много цифр, на последовательность отдельных изображений, каждое из которых содержит одну цифру. Например, хотелось бы расформировать изображение:

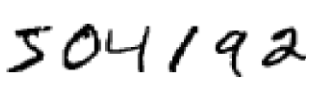


Рисунок 6

на шесть отдельных изображений:

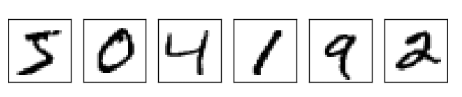


Рисунок 7

После того, как изображение было сегментировано, программе необходимо классифицировать каждую отдельную цифру. Так, например, нужно, чтобы наша программа распознала, что первая цифра выше, это 5.

Для распознавания отдельных цифр будем использовать трехслойную нейронную сеть:

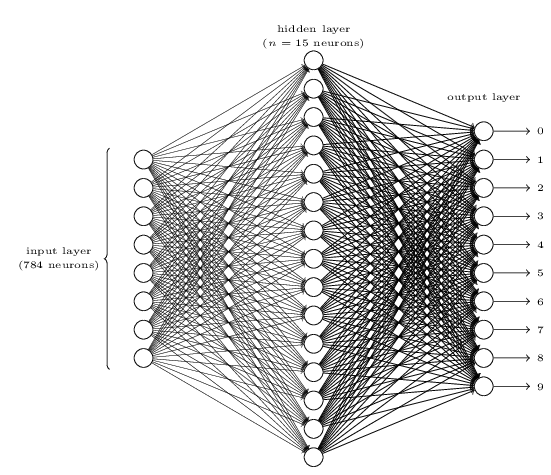


Рисунок 8

Входной уровень сети содержит нейроны, считывающие значения входных пикселей. Обучающие данные для сети будут состоять из множества отсканированных рукописных цифр размером 28 на 28 пикселей, и поэтому входной слой содержит 784=28×28 нейроны.

Входные пиксели представлены в оттенках серого: со значением 0 представляет белый цвет, значение 1.0 представляет черный, а промежуточные значения представляют постепенно темнеющие оттенки серого. Второй слой сети является скрытым слоем. Обозначим количество нейронов в скрытом слое на и будем экспериментировать с разными значениями для . Приведенный пример иллюстрирует небольшой скрытый слой, содержащий =15 нейроны.

Выходной слой сети содержит 10 нейронов. Если сработает первый нейрон, т.е. выдаст значение ≈ 1 , то это будет означать, что сеть считает что на картинке цифра равной 0 . Если сработает второй нейрон, это будет означать, что сеть считает, что на картинке цифра 1 . И так далее.

Давайте сосредоточимся на первом выходном нейроне, который пытается определить, присутствует ли на изображении цифра 0 . Он делает это, взвешивая данные, полученные из скрытого слоя нейронов. Предположим, что первый нейрон в скрытом слое определяет, присутствует ли изображение, подобное приведенному ниже:

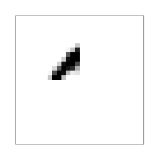


Рисунок 9

Он может это сделать, сильно увеличив вес входных пикселей, которые перекрываются с изображением, и лишь слегка увеличив вес других входных данных. Аналогичным образом, давайте предположим, что второй, третий и четвертый нейроны в скрытом слое определяют, присутствуют ли следующие изображения или нет:

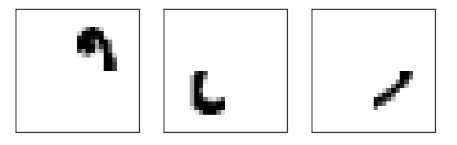


Рисунок 10

Эти четыре изображения вместе составляют изображение 0, которое было предоставлено в строке цифр, показанной ранее:

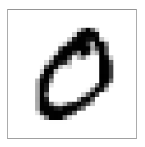


Рисунок 11

Итак, если активируются все четыре из этих скрытых нейронов, то можно заключить, что цифра равна 0. Конечно, это не единственное доказательство, которое используется, чтобы сделать вывод о том, что на изображение был 0 - возможно получить 0 многими другими способами (например, путем перевода приведенных выше изображений или незначительных искажений). Но, с уверенностью можно сказать, что, по крайней мере, в этом случае входные данные были равны 0

Теперь, после всего сказанного ничто не говорит о том, что трехслойная нейронная сеть должна работать так, как было описано выше, со скрытыми нейронами, обнаруживающими простые формы компонентов. Возможно, умный алгоритм обучения найдет такое распределение весов, которое позволит нам использовать разное количество нейронов. Но как эвристический способ мышления, он работает довольно хорошо и может сэкономить много времени при разработке хороших архитектур нейронных сетей.

## Алгоритм градиентного спуска

Теперь, когда у нас есть дизайн нейронной сети, она должна научиться распознавать числа. В первую очередь, нам нужен набор данных для обучения. В качестве такого набора данных можем взять MNIST (Modified National Institute of Standards and Technology). Набор данных MNIST представляет из себя 2 части: 1 часть содержит 60 тысяч изображений, используемых как данные для обучения. Изображения представляют из себя рукописные цифры, написанные 250 людьми. Все изображения размером 28x28 пикселей, цвет каждого пикселя – оттенок серого. Второй набор данных – это 10 тысяч изображений, используемых для проверки.

Обозначим данные для обучения как вектор x, размерность вектора 28x28 = 784. Каждый элемент вектора – это число соответствующее оттенку серого. Вектор результата нейронной сети обозначим как = . – вектор, состоящий из 10 элементов. Например, если входное изображение соответствует цифре 6, то вектор - это желанный выход с сети, где T – это операция транспонирования, превращающая вектор строки в обычный вектор (столбец).

Чтобы оценить насколько близко выходные данные сети приближаются к *y(x)*,определим функцию оценки

– обозначает веса нейронной сети

– обозначает смещение

– обозначает количество входов

– вектор выходов нейронной сети при условии, что - вход нейронной сети

- обозначает длину вектора

Функция – квадратичная функция оценки, иногда она носит название СКО (Средняя квадратичная ошибка)

Можно заметить что функция – неотрицательна. Функция близка к нулю, когда Значение близко к значению для всех тренировочных векторов . Для такого случая можно сказать, что мы нашли оптимальное значение для весов и смещений в нейронной сети. Другими словами работа алгоритма сводится к нахождению таких и при которых значение функции оценки минимально. Для этого используется алгоритм известный как градиентный спуск.

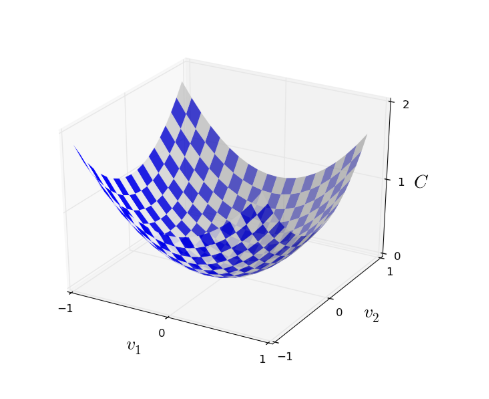


Рисунок 12

Предположим, что мы минимизируем функцию . Она может быть реальной функцией множества переменных . На рисунке для упрощения изображена функция двух переменных. Наша цель – найти глобальный минимум функции .

Один из методов решения подобной задачи – вычисление минимума аналитически. Можно вычислить производные для нахождения минимума функции. Данный метод можно было бы использовать, если функция зависит от небольшого числа аргументов. Но в реальных нейронных сетях количество аргументов исчисляется миллиардами. Данный метод практически невозможен ввиду чрезмерно большой вычислительной сложности.

Вместо этого можно менять аргументы функции и вычислять значение, на которое изменилось значение функции

Выберем и такими, что значение отрицательно. Это подразумевает, что в направлении и находится минимум функции . Обозначим вектор . Также определим градиент вектора , как

Можно записать следующее выражение:

Данное выражение связывает изменение вектора с через вектор градиента . И что более важно, оно позволяет нам определить, как выбрать , чтобы было отрицательным (ошибка уменьшается). Можно записать

где параметр известный как скорость обучения. Подставив выражение 4 в выражение 3, получим

Поскольку это гарантирует, что , если мы выбираем в соответствии с уравнением (10). Выражение (10) используется для вычисление . Допустим текущее значение аргумента равно . Новое значение аргумента

Используя формулу (12) уменьшаем пока не получим глобальный минимум функции.

Алгоритм, позволяющий уменьшить время обучения, называется стохастический градиентный спуск (*stochastic gradient descent - SGD*). Основная идея алгоритма – оценка градиента , используя не все тренировочные данные, а лишь небольшой набор - *mini-batch*, выбранный случайным образом из всего набора. В результате работы SGD производится выбор mini-batch и обновление параметров для каждой итерации. Выбор mini-batch и обновление параметров производится до тех пор, пока мы не используем все тренировочные данные – конец эпохи (epoch).

## Алгоритм обратного распространения

Алгоритм обратного распространения – это быстрый алгоритм вычисления градиентов, который будет использован для вычисления

Цель алгоритма обратного распространения в вычислении частных производных и .

- выражение ошибки на слое .

– вектор компоненты, которого представлены в виде частных производных .

.

- как быстро изменяется функция активации.

– выражение ошибки (ошибки на слое ) через (ошибка на слое ).

- матрица весов для ()-го слоя.

– ошибка на слое .

- выражение для скорости изменения функции оценки в зависимости от любого смещения в сети.

– выражение для скорости изменения функции оценки по отношению к любому весу в сети.

Алгоритм обратного распространения для одного обучающего примера:

1. На вход подаем вектор – обучающие данные и выполняем следующие действия:
2. Вычисляем и для слоев
3. Вычисляем вектор ошибки на последнем слое
4. Вычисляем для каждого
5. Задаем градиент функции оценки формулами и

На практике в основном учитывают много обучающих примеров, которые называются mini-batch и только потом изменяют веса и смещения за счет накопившейся ошибки. Тогда алгоритм для m обучающих примеров будет выглядеть следующем образом:

1. На вход подаем m обучающих примеров (mini-batch).
   1. Для каждого обучающего примера, вектором которого являются обучающие данные выполняем следующие действия:
   2. Вычисляем и для слоев
   3. Вычисляем вектор ошибки на последнем слое
   4. Вычисляем для каждого .
2. Обновляем веса в соответствии с правилом .
3. Обновляем смещения в соответствии с правилом

## Использования нейронной сети для разбора рукописных чисел

Используем готовую нейронную сеть, которую можно скачать по ссылке <https://github.com/mnielsen/neural-networks-and-deep-learning.git>. Данная модель написана на языке python. В качестве тренировочных данных используется MNIST, которая содержит 50.000 разных картинок с цифрами. C помощью этой модели исследуем как работает данная сеть.

import network

import mnist\_loader

# training\_data - данные для обучения сети, validation\_data - данные для проверки сети, test\_data - данные для проверки работоспособности

training\_data, validation\_data, test\_data = mnist\_loader.load\_data\_wrapper()

training\_data = list(training\_data)

# 1 значение(784) - количество входных нейронор на первом слое

# 2 значение(30) - количество нейронов в скрытом слое

# 3 значение(10) - количество нейронов на выходе сети

net = network.Network([784, 30, 10])

# 1 значение(training\_data) - данные для обучения сети

# 2 значение(30) - количество эпох

# 3 значение(10) - размер мини-набора

# 4 значение(3.0) - скорость обучения

# 5 значение(test\_data=test\_data) - данные для проверки работоспособности

net.SGD(training\_data, 30, 10, 3.0, test\_data=test\_data)

Результат обучения нейронной сети изображен ниже. На первой итерации из 10000 проверочных данных правильно было определено 8152. Видно, что с увеличением количеством эпох количество правильных результатов увеличивается. После 30 эпох видно, что лучший результат который может дать нейронная сеть, порядка 9500 правильных результатов из 10000.

Epoch 0: 8152 / 10000, took 6.69 seconds

Epoch 1: 9235 / 10000, took 6.70 seconds

Epoch 2: 9348 / 10000, took 6.67 seconds

Epoch 3: 9299 / 10000, took 6.76 seconds

Epoch 4: 9393 / 10000, took 6.66 seconds

Epoch 5: 9442 / 10000, took 6.70 seconds

Epoch 6: 9466 / 10000, took 6.72 seconds

Epoch 7: 9458 / 10000, took 6.71 seconds

Epoch 8: 9474 / 10000, took 6.76 seconds

Epoch 9: 9432 / 10000, took 6.70 seconds

Epoch 10: 9490 / 10000, took 6.70 seconds

Epoch 11: 9510 / 10000, took 6.72 seconds

Epoch 12: 9485 / 10000, took 6.72 seconds

Epoch 13: 9499 / 10000, took 6.70 seconds

Epoch 14: 9512 / 10000, took 6.72 seconds

Epoch 15: 9488 / 10000, took 6.70 seconds

Epoch 16: 9495 / 10000, took 6.71 seconds

Epoch 17: 9493 / 10000, took 6.71 seconds

Epoch 18: 9505 / 10000, took 6.70 seconds

Epoch 19: 9517 / 10000, took 6.70 seconds

Epoch 20: 9482 / 10000, took 6.68 seconds

Epoch 21: 9510 / 10000, took 6.70 seconds

Epoch 22: 9517 / 10000, took 6.71 seconds

Epoch 23: 9533 / 10000, took 6.71 seconds

Epoch 24: 9505 / 10000, took 6.72 seconds

Epoch 25: 9516 / 10000, took 6.70 seconds

Epoch 26: 9495 / 10000, took 6.74 seconds

Epoch 27: 9524 / 10000, took 6.73 seconds

Epoch 28: 9535 / 10000, took 6.73 seconds

Epoch 29: 9521 / 10000, took 6.72 seconds

## Распознавание рукописных цифр с помощью Matlab

Создадим простую нейронную сеть для выполнения задачи классификации, используя Deep Network Designer app.

Структура данной нейронной сети показана ниже:

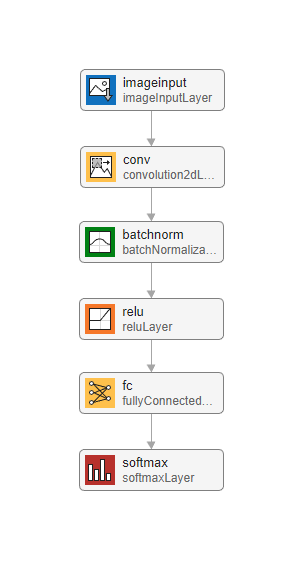


Рисунок 13

Imageinput – Слой ввода изображений, который содержит двумерные изображения и загружает их в нейронную сеть, применяя нормализацию данных.

## 3.8.1 Convolution2dLayer

Convolution2dLayer использует фильтры для работы. Фильтр – двухмерный массив с коэффициентами (весами), которые в начальные момент задаются случайно. Предполагается, что после обучения нейронной сети фильтр будет выделять какую-либо особенность в изображение, что упростит распознавание. Коэффициенты фильтров могут принимать любые значения.

Convolution2dLayer перемещает фильтры по изображению горизонтально и вертикально, суммируя перемноженное значение пикселей (от 0 до 255 по цветовой модели RGB) на веса фильтров.

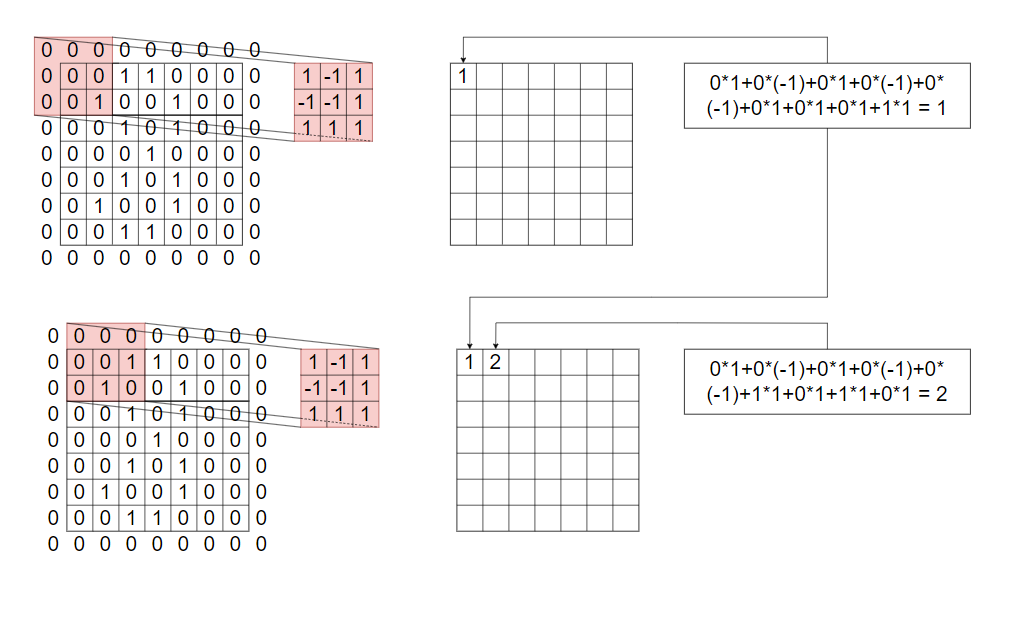


Рисунок 14

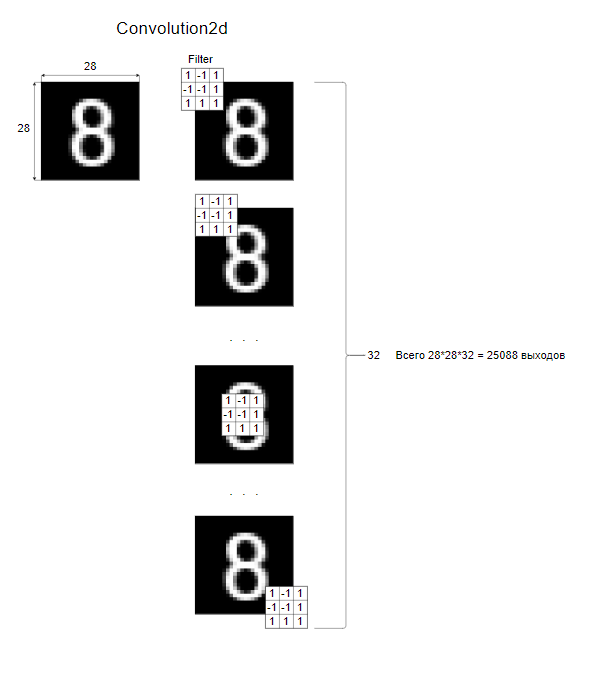


Рисунок 15

BatchNormalizationLayer – предназначен для улучшения скорости, производительности и стабильности обучения нейронных сетей.

В основе Batch Normalization лежит решение проблемы «внутреннего ковариационного сдвига» (Internal Covariate Shift). Этот термин описывает явление, при котором распределение входных данных каждого слоя нейронной сети меняется в процессе обучения, из-за чего сети становится сложнее обучать. Это происходит из-за того, что параметры предыдущих слоев изменяются во время обучения, влияя на данные текущего слоя.

Batch Normalization решает эту проблему, нормализуя выход каждого слоя. Нормализация заключается в преобразовании входных данных каждого слоя таким образом, чтобы среднее значение было приближено к нулю, а стандартное отклонение — к единице. Это делает сеть менее чувствительной к масштабу входных данных и улучшает общую стабильность процесса обучения.

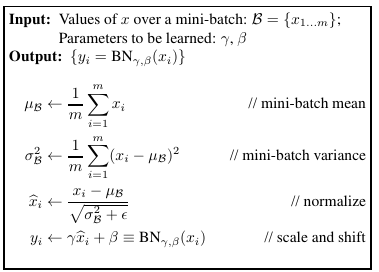


Рисунок 16

ReluLayer – выполняет пороговую операцию для каждого элемента входных данных, при которой любое значение, меньшее нуля, устанавливается равным нулю. Эта операция эквивалентна:

FullyConnectedLayer – умножает входы на матрицу весов и суммирует вектор смещения

SoftmaxLayer – функция, которая превращает наборы данных в вероятности, составляющие полную группу. В глубоком обучении термин "логитовый слой" обычно используется для последнего слоя нейронной сети (Neural Network) задач классификации, которая преобразует необработанные значения прогноза в виде действительных чисел в диапазоне от [-∞, +∞]. Логиты – это необработанные результаты, полученные на последнем уровне нейронной сети до того, как произойдет активация.Softmax превращает логиты в вероятности, получая экспоненту каждого значения, а затем подвергая нормализации (Normalization) каждое , то есть разделяя на их сумму, чтобы сумма всех экспонент равнялась единице.

logits = [2.0, 1.0, 0.1]

exps = [np.exp(i) for i in logits] vb

sum\_of\_exps = sum(exps)

softmax = [j/sum\_of\_exps for j in exps]

Результат работы модели для 10 эпох:

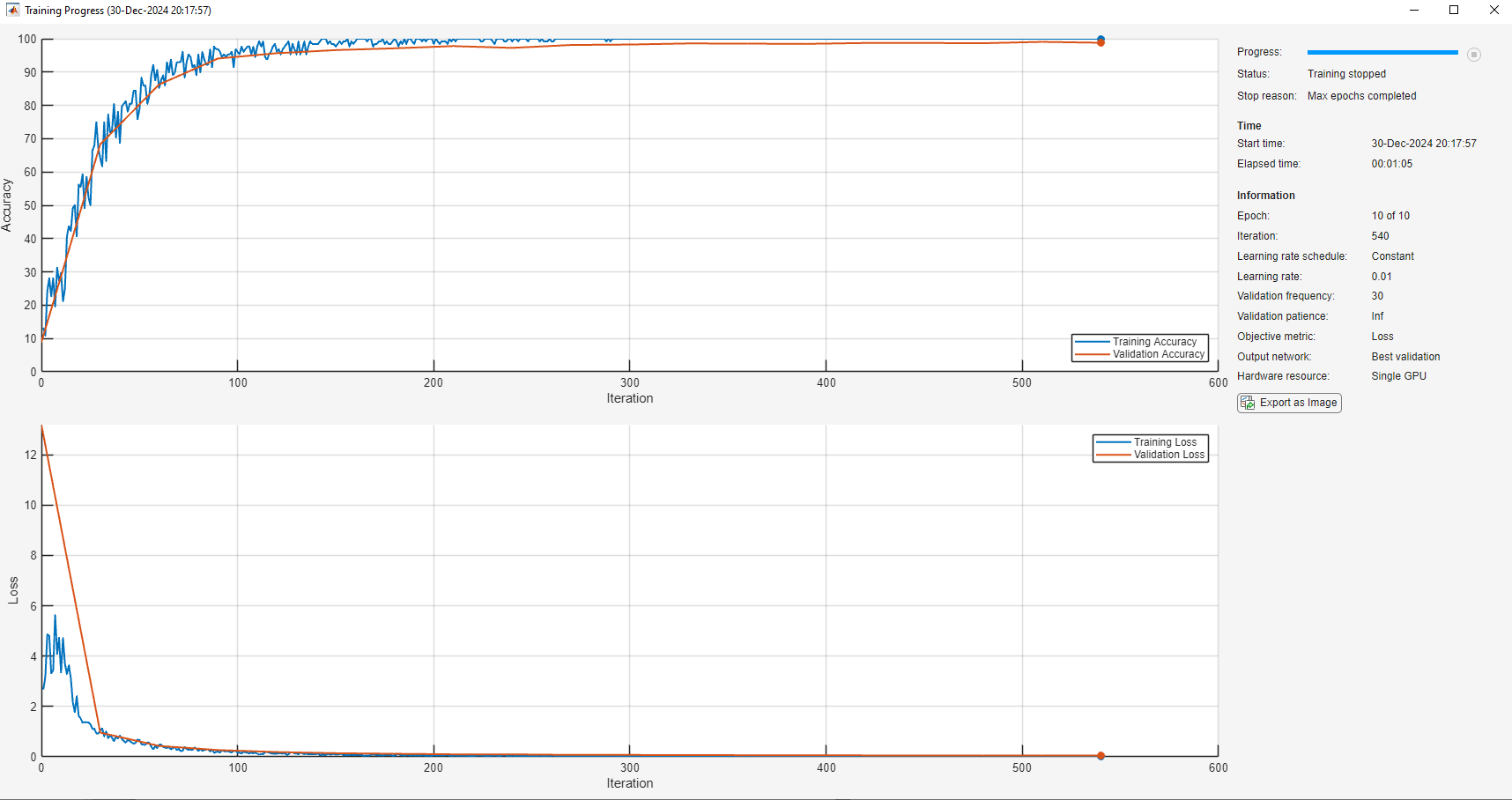


Рисунок 17

## Исследование сети семантической сегментации с помощью Grad-CAM в Matlab

Рассмотрим, как работает предварительно обученная сеть семантической сегментации с помощью Grad-CAM. Сеть семантической сегментации классифицирует каждый пиксель изображения, в результате чего изображение сегментируется по классу. Grad-CAM - метод визуализации с глубоким обучением, который позволяет увидеть, какие области изображения важны для принятия решения при классификации пикселей.

В качестве примера используется данное изображение:



Рисунок 18

Результат семантической сегментации:

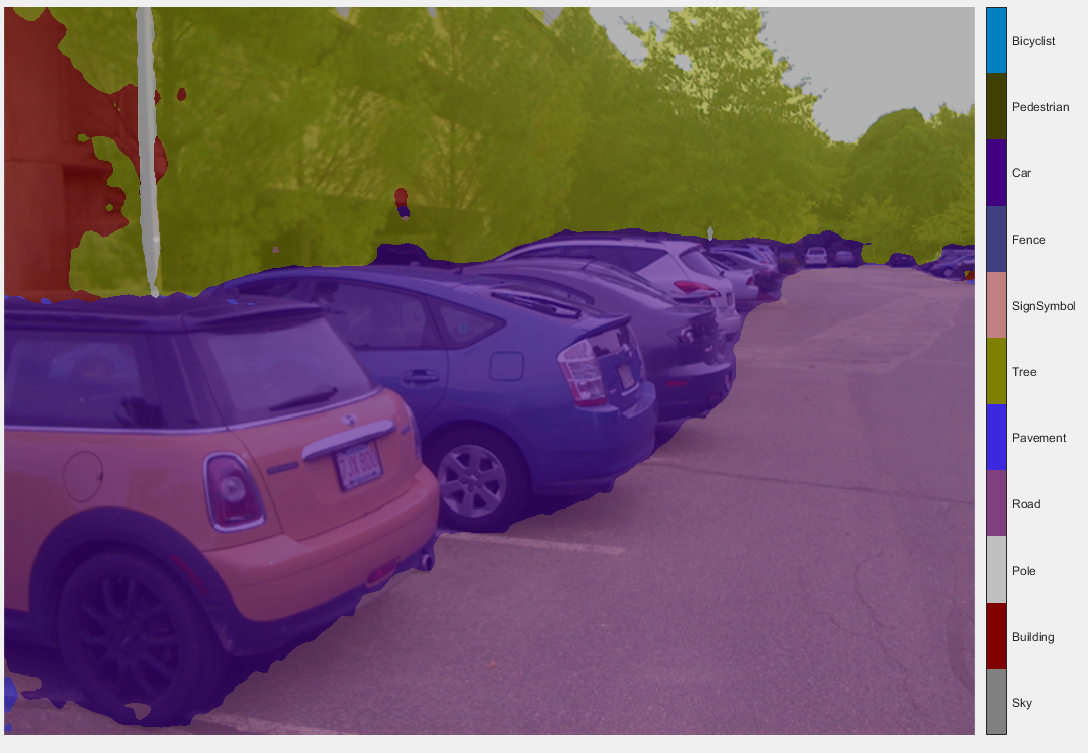


Рисунок 19

Видно, что нейронная сеть неправильно классифицирует участки дороги вблизи нижней части автомобилей из-за плохого разрешения между шиной и границей дороги, а также из-за теней автомобилей, размывающих их границу с дорогой.

Также можно рассмотреть отображение Grad-CAM данного изображения для двух классов: дорога и автомобиль. Функция Grad-CAM суммирует пространственные измерения выходного слоя для указанных классов, чтобы получить скалярное значение. Затем это скалярное значение дифференцируется по каждому объекту в выходном слое.

Другими словами данный слой позволяет вынести решение принадлежности части изображения к одному из классов, чем краснее, тем более вероятно.

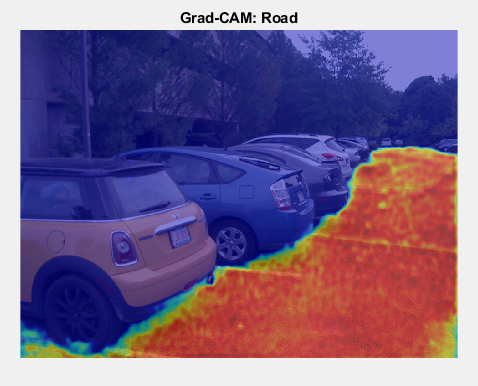
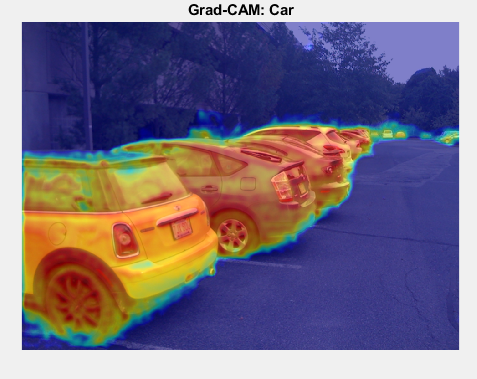
 

Рисунок 20

Отображения Grad-CAM и отображение семантической сегментации показывают схожее выделение. Отображение Grad-CAM для класса дорог показывает довольно равномерное распределение важности для принятия решения о классификации в сети. Для класса автомобилей, как и следовало ожидать, наиболее важна центральная масса каждого автомобиля.

Можно также использовать Grad-CAM для исследования промежуточных слоев сети для двух классов. Вычисление отображения Grad-CAM для слоев, расположенных последовательно глубже в сети:

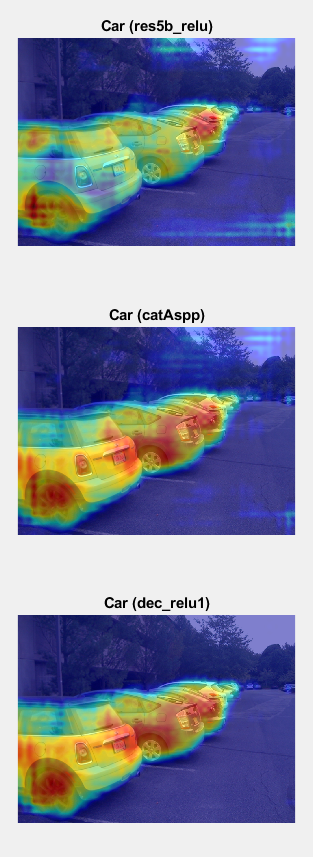
 

Рисунок 21

Слой res5b\_relu находится ближе к середине сети, тогда как dec\_relu1 - ближе к концу сети.

Можно заметить, что отображения Grad-CAM похоже на отображение семантической сегментации, когда используется слой сети, стоящий ближе к концу. Более ранние слои имеют небольшой размер поля восприятия и позволяют изучать небольшие, низкоуровневые объекты по сравнению со слоями в конце сети. Ранние слои сети отображают более абстрактные результаты и, как правило, фокусируются на более низкоуровневых частях изображения, таких как края, с меньшей осведомленностью о семантических классах. Например, для более ранних отображений можно заметить, что небо выделено как для автомобилей, так и для дорог. Это говорит о том, что более ранние слои фокусируются на областях изображения, которые связаны с классом, но не обязательно принадлежат ему.

Можно также рассмотреть пример, где нет машин, но есть пешеход:



Рисунок 22

Результат семантической сегментации:

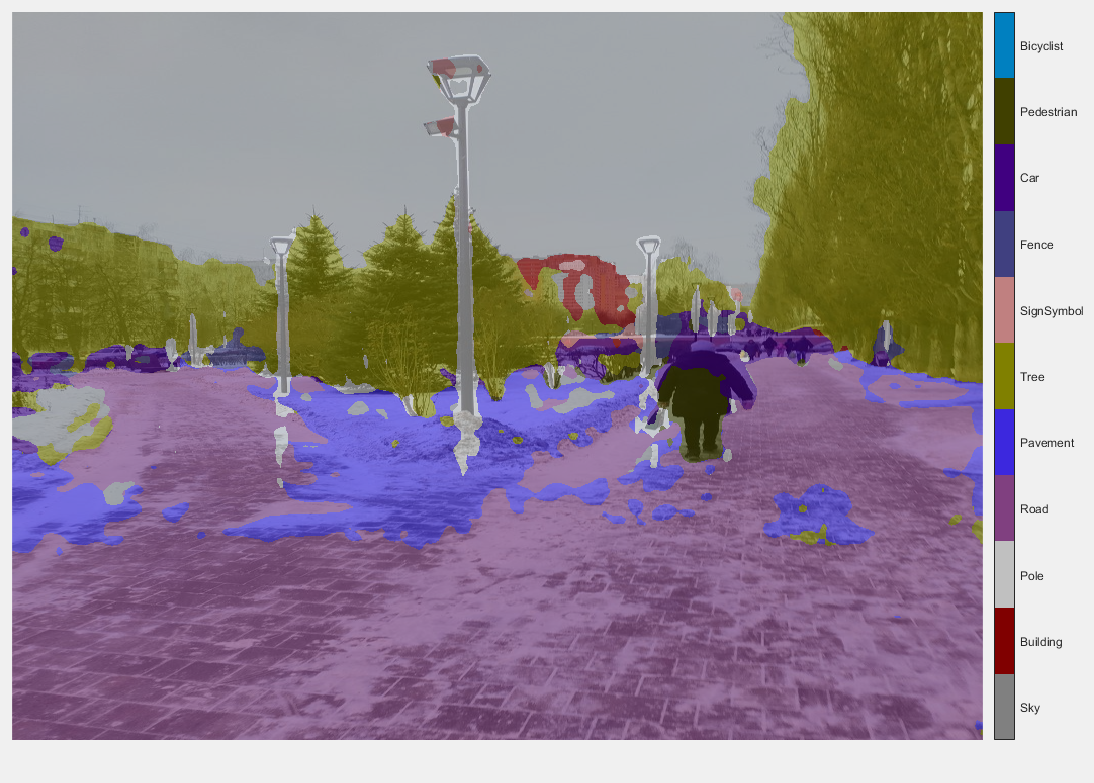


Рисунок 23

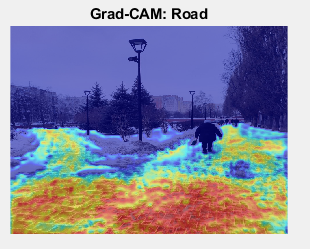
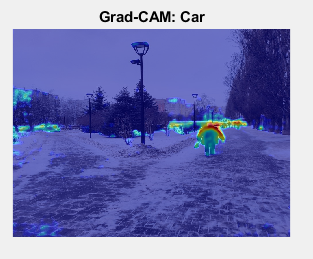
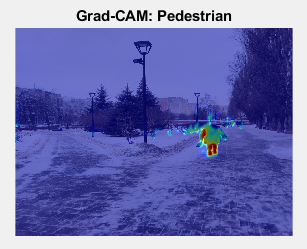
Отображение Grad-CAM для трех классов: дорога, автомобиль, пешеход:   

Рисунок 24

Интересно заметить распределения вероятностей для класса автомобиль (car) и пешеход (pedestrian). Видно, что для класса автомобиль вероятной областью выделена часть пешехода рук и плеч, которая образуют кривую, похожую на силуэта автомобиля. В то время как ноги определяются с достаточной вероятностью, как элемент пешехода.

# Заключение

В результате выполнения проекта были изучены принципы работы нейронных сетей, математическое описание применяемых алгоритмов и их реализация, разобрана работа нейронных сетей на примере распознавания рукописных цифр и изучена сеть семантической сегментации с помощью Grad-CAM в среде Matlab.

# Список Литературы

1. URL: <https://neuralnetworksanddeeplearning.com/index.html>, Neural Networks and Deep Learning (Последняя дата обращение: 17.01.2025).
2. URL: <https://www.mathworks.com/help/deeplearning/gs/create-simple-image-classification-network-using-deep-network-designer.html>, MathWorks Help Center (Последняя дата обращение: 23.01.2025).
3. URL: <https://www.mathworks.com/help/vision/ug/explore-semantic-segmentation-network-using-grad-cam.html>, MathWorks Help Center (Последняя дата обращение: 27.01.2025).
4. “Глубокое обучение (Николенко)”, книга, Николенко С., Кадурин А., Архангельская Е, Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия «Библиотека программиста»).
5. URL: <https://youtu.be/pj9-rr1wDhM?si=1pRaU7tl0-JcQaG7>, Convolutional Neural Networks Explained (CNN Visualized) (Последняя дата обращение: 22.01.2025).