Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

Кафедра автоматизированных и вычислительных систем

Направление подготовки 09.04.01 «Информатика и вычислительная техника»

Программа «Распределенные автоматизированные системы»

ОТЧЕТ

ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Тема: «Восстановление растровых изображений в условиях неполной входной

информации с использованием нейросетевого аппарата»

Выполнил ВМм-161 Д.А. Юров

Группа Подпись, дата Инициалы, фамилия

Принял В.Ф. Барабанов

Подпись, дата Инициалы, фамилия

Руководитель по дипломному

проекту В.Ф. Барабанов

Подпись, дата Инициалы, фамилия

Защищен Оценка

Воронеж 2018

Оглавление

[1 Назначение и область применения компонентов. 3](#_Toc504608124)

[2 Цели и задачи работы 3](#_Toc504608125)

[3 Теоретическая часть 4](#_Toc504608126)

[3.2 Технологические платформы разработки искусственных нейронных сетей 7](#_Toc504608127)

[3.3 Алгоритмы построения искусственных нейронных сетей 11](#_Toc504608128)

[3.3.1 Нейроная сеть Хопфилда 11](#_Toc504608129)

[3.3.2 Сверточные нейронные сети 14](#_Toc504608130)

[3.4 Базовая имплементация алгоритма 18](#_Toc504608131)

[Список литературы 20](#_Toc504608132)

[Список публикаций 21](#_Toc504608133)

# 1 Назначение и область применения компонентов.

Назначение – автоматизированное восстановление качества и устранение повреждений изображений. Под неполной входной информации подразумеваются растровые изображения, затронутые следующими эффектами: пикселизацией, артефактами сжатия, прямым повреждением данных, физические препятствиями при съемке (разводы, царапины, пыль на объективе и между линзами).

Область применения – ведение электронных архивов, коррекция изображений после оцифровки, восстановление поврежденных при сжатии или передаче изображений.

# 2 Цели и задачи работы

Целями данной работы являются:

* исследование и разработка алгоритмов восстановления изображений;
* проектирование, реализация и обучение нейронной сети;
* имплементация обученной нейронной сети в существующие решения в виде библиотеки либо разработка нативного приложения.

Для достижения поставленных целей необходимо:

* изучить предметную область;
* проанализировать существующие алгоритмы обработки изображений;
* адаптировать существующий либо разработать собственный алгоритм восстановления изображений;
* реализовать алгоритм в виде нейронной сети с последующим ее обучением в качестве библиотеки либо самостоятельного приложения;
* проанализировать полученные в ходе работы результаты.

# 3 Теоретическая часть

3.1 Нейронные сети и их классификация

Искусственная нейронная сеть (ИНС) - математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей - сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

ИНС представляют собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие по отдельности простые процессоры вместе способны выполнять довольно сложные задачи.

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п.; c математической точки зрения, обучение нейронных сетей - это многопараметрическая задача нелинейной оптимизации; с точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники; с точки зрения развития вычислительной техники и программирования, нейронная сеть - способ решения проблемы эффективного параллелизма; с точки зрения искусственного интеллекта, ИНС является основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения - одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искажённых данных.

Нейронные сети можно классифицировать по следующим критериям.

1. По характеру обучения их можно разделить на нейронные сети, использующие обучение с учителем и нейронные сети, использующие обучение без учителя. Обучение с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются обучающей парой. Обычно сеть обучается на некотором числе таких обучающих пар. Предъявляется выходной вектор, вычисляется выход сети и сравнивается с соответствующим целевым вектором. Далее веса изменяются в соответствии с алгоритмом, стремящимся минимизировать ошибку. Векторы обучающего множества предъявляются последовательно, вычисляются ошибки и веса подстраиваются для каждого вектора до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемого уровня.

Обучение без учителя является намного более правдоподобной моделью обучения с точки зрения биологических корней искусственных нейронных сетей. Развитая Кохоненом и многими другими, она не нуждается в целевом векторе для выходов и, следовательно, не требует сравнения с предопределенными идеальными ответами. Обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса сети так, чтобы получались согласованные выходные векторы, т. е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы. Процесс обучения, следовательно, выделяет статистические свойства обучающего множества и группирует сходные векторы в классы.

1. По подходу к настройке весов нейронные сети можно разделить на сети с фиксированными связями (весовые коэффициенты нейронной сети выбираются сразу, исходя из условий задачи) и сети с динамическими связями (для них в процессе обучения происходит настройка синаптических весов).
2. По типу входной информации нейронные сети делятся на аналоговые ( входная информация представлена в форме действительных чисел), двоичные (вся входная информация в таких сетях представляется в бинарном формате) и образные (вся входная информация относится к таким образам, как к знакам, символам, иероглифам и т.п.).
3. По типу применяемой модели ИНС делятся на сети прямого распространения, рекуррентные нейронные сети, радиально базисные функции и самоорганизующиеся карты (сети Кохонена) [1].

В сетях прямого распространения все связи направлены строго от входных нейронов к выходным. К таким сетям относятся, например: простейший персептрон (разработанный Розенблаттом) и многослойный персептрон.

В реккурентных ИНС сигнал с выходных нейронов или нейронов скрытого слоя частично передается обратно на входы нейронов входного слоя.

Радиально базисные функции – вид нейронной сети, имеющий скрытый слой из радиальных элементов и выходной слой из линейных элементов. Сети этого типа довольно компактны и быстро обучаются. Предложены в работах Broomhead and Lowe (1988) и Moody and Darkin (1989). Радиально базисная сеть обладает следующими особенностями: один скрытый слой, только нейроны скрытого слоя имеют нелинейную активационную функцию и синаптические веса входного и скрытого слоев равны единицы.

Самоорганизующиеся карты или Сети Кохонена – такой класс сетей, как правило, обучается без учителя и успешно применяется в задачах распознавания. Сети такого класса способны выявлять новизну во входных данных: если после обучения сеть встретится с набором данных, непохожим ни на один из известных образцов, то она не сможет классифицировать такой набор и тем самым выявит его новизну. Сеть Кохонена имеет всего два слоя: входной и выходной, составленный из радиальных элементов.

К основным задачам, решаемым при помощи нейронных сетей, относят:

* распознавание образов и классификация (перцептрон, сети адаптивного резонанса, сеть радиально-базисных функций);
* принятие решений и управление (перцептрон, сеть радиально-базисных функций);
* кластеризация;
* прогнозирование (перцептрон, сети адаптивного резонанса, сеть радиально-базисных функций);
* аппроксимация (перцептрон, сети адаптивного резонанса, сеть радиально-базисных функций);
* сжатие данных и ассоциативная память. Восстановление исходного набора данных из части информации — называется (авто)ассоциативной памятью. Ассоциативная память позволяет также восстанавливать исходный сигнал/образ из зашумленных/поврежденных входных данных (перцептрон, Нейронная сеть Хопфилда);
* анализ данных (Перцептрон, Самоорганизующаяся карта Кохонена, Нейронная сеть Кохонена);
* оптимизация (Самоорганизующаяся карта Кохонена, Нейронная сеть Кохонена).

# 3.2 Технологические платформы разработки искусственных нейронных сетей

В настоящее время существует большое количество библиотек и фреймворков с открытым исходным кодом, позволяющих реализовывать системы машинного обучения, снимающих необходимость низкоуровневой реализации алгоритмов. Различают библиотеки высокого уровня абстракции, такие как Keras, так и низкого уровня, например, Tensorflow и Theano.

TensorFlow — открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия. TensorFlow применяtтся для исследований и разработки продуктов Google. TensorFlow является продолжением закрытого проекта DistBelief. Изначально, TensorFlow была разработана командой Google Brain для внутреннего использования в Google, а потом (9 ноября 2015 года) была переведена в свободный доступ с открытой лицензией Apache 2.0.

Закрытая система машинного обучения DistBelief разрабатывалась Google Brain для внутренних проектов с 2011 года для работы с нейронными сетями глубинного обучения. Она стала использоваться во многих исследовательских и коммерческих проектах группы фирм холдинга Alphabet. После успеха DistBelief фирма Google решила вывести проект на новый уровень. Целью было упрощение и оптимизация кодов библиотеки, увеличение надёжности и удобства пользования. Новая библиотека получила название TensorFlow. В 2009 году команда под руководством Джеффри Хинтона реализовала метод обобщенного обратного распространения ошибки и другие улучшения, которые позволили существенно улучшить точность нейронных сетей, что привело, в частности, к снижением погрешности в распознавании речи на 25 %.[2] В 2013 году Джеффри Хинтон присоединился к проекту.

В то время как эталонная реализация работает на единичных устройствах, TensorFlow может работать на многих параллельных процессорах, как CPU, так и GPU, опираясь на архитектуру CUDA для поддержки вычислений общего назначения на графических процессорах). TensorFlow доступна для 64-разрядных Linux, macOS, Windows, и для мобильных вычислительных платформ, включая Android и iOS. Вычисления TensorFlow выражаются в виде потоков данных через граф состояний. Название TensorFlow происходит от операций с многомерными массивами данных, которые также называются «тензорами».

Theano - это библиотека, написанная на Python, которая позволяет определять, оптимизировать и оценивать математические выражения с многомерными массивами. Используя Theano, можно достичь скоростей, конкурирующих с реализацией на C с помощью ручного управления для проблем, связанных с большими объемами данных. Она также может превосходить C на много порядков, воспользовавшись ресурсами графических процессоров.

Theano объединяет аспекты системы компьютерной алгебры (CAS) с аспектами оптимизирующего компилятора. Она также может генерировать настраиваемый код на C для различных математических операций. Эта комбинация CAS с оптимизирующей компиляцией особенно полезна для задач, в которых сложные математические выражения неоднократно оцениваются, а скорость оценки является критической. В ситуациях, когда много разных выражений оцениваются после расчетов, Theano может свести к минимуму объем издержек компиляции/анализа.

Компилятор Theano применяет множество оптимизаций различной сложности к символическим выражениям. Эти оптимизации включают:

* использование графического процессора для вычислений;
* группировку констант;
* слияние подобных подграфов для избегания избыточных расчетов;
* арифметическое упрощение;
* вставку эффективных операций BLAS (например, GEMM) в различных контекстах;
* использование сглаживания памяти для сокращения объема вычислений;
* использование операций inplace везде, где это не мешает сглаживанию;
* циклическое слияние для элементарных подвыражений;
* усовершенствования численной устойчивости (например, log (1+ exp(x)) и log (Ʃi exp (x [i]))) [3].

Библиотека Theano была написана в лаборатории LISA для поддержки быстрой разработки эффективных алгоритмов машинного обучения.

Keras представляет собой высокоуровневый API нейронных сетей, написанный на Python и способный работать поверх TensorFlow, CNTK или Theano. Он был разработан с упором на возможность быстрого экспериментирования. Разработчики считают, что способность идти от идеи к результату с наименьшей возможной задержкой является ключом к проведению хороших исследований [4].

Отличительными чертами Keras являются:

* возможность легко и быстро создавать прототипы (благодаря удобству, модульности и расширяемости);
* поддержка как сверточных, так и рекуррентных сетей, а также их комбинации;
* доступно использование GPU (ядра CUDA в современных графических процессорах значительно повышают скорость обучения).

Keras совместим с Python 2.7-3.5.

Keras - это API, предназначенный для людей, а не для машин. Он ставит приоритетом взаимодействие с разработчиком. Keras следует методам снижения когнитивной нагрузки: он предлагает простые и простые API, он минимизирует количество действий пользователя, необходимых для случаев общего использования, и обеспечивает четкую и эффективную обратную связь с ошибкой пользователя.

Под моделью в Keras понимается последовательность или график автономных, полностью настраиваемых модулей, которые могут быть соединены вместе с минимальными ограничениями. В частности, нейронные слои, функции затрат, оптимизаторы, схемы инициализации, функции активации, схемы регуляризации - это автономные модули, которые можно комбинировать для создания новых моделей.

Новые модули просто добавлять (как новые классы и функции), а существующие модули предоставляют множество примеров. Модели описаны в коде Python, который компактен, легче отлаживается и обеспечивает простоту расширяемости.

# 3.3 Алгоритмы построения искусственных нейронных сетей

# 3.3.1 Нейроная сеть Хопфилда

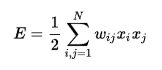
Нейронная сеть Хопфилда (англ. Hopfield network) — полносвязная нейронная сеть с симметричной матрицей связей. В процессе работы динамика таких сетей сходится (конвергирует) к одному из положений равновесия. Эти положения равновесия определяются заранее в процессе обучения, они являются локальными минимумами функционала, называемого энергией сети (в простейшем случае — локальными минимумами отрицательно определённой квадратичной формы на n-мерном кубе). Такая сеть может быть использована как автоассоциативная память, как фильтр, а также для решения некоторых задач оптимизации.

В отличие от многих нейронных сетей, работающих до получения ответа через определённое количество тактов, сети Хопфилда работают до достижения равновесия, когда следующее состояние сети в точности равно предыдущему: начальное состояние является входным образом, а при равновесии получают выходной образ. Ее вариацией является нейронная сеть Хемминга [5].

Нейронная сеть Хопфилда устроена так, что её отклик на запомненные эталонных «образов» составляют сами эти образы, а если образ немного исказить и подать на вход, он будет восстановлен и в виде отклика будет получен оригинальный образ. Таким образом, сеть Хопфилда осуществляет коррекцию ошибок и помех. Сеть Хопфилда однослойная и состоит из искусственных нейронов. Каждый нейрон системы может принимать на входе и на выходе одно из двух состояний (что аналогично выходу нейрона с пороговой функцией активации):



Из-за их биполярной природы нейронные сети Хопфилда иногда называют спинами. Каждый нейрон связан со всеми остальными нейронами. Взаимодействие нейронов сети описывается выражением:

,

где де w_{ij} элемент матрицы взаимодействий W, которая состоит из весовых коэффициентов связей между нейронами. В эту матрицу в процессе обучения записывается М «образов» — N-мерных бинарных векторов: S_m=(s_{m1},s_{m2},...,s_{mN})

В сети Хопфилда матрица связей является симметричной w_{ij}=w_{ji}, а диагональные элементы матрицы полагаются равными нулю (w_{ii}=0), что исключает эффект воздействия нейрона на самого себя и является необходимым для сети Хопфилда, но не достаточным условием, устойчивости в процессе работы сети. Достаточным является асинхронный режим работы сети. Подобные свойства определяют тесную связь с реальными физическими веществами называемыми [спиновыми стеклами](http://ru.cybernetics.wikia.com/wiki/%D0%A1%D0%BF%D0%B8%D0%BD%D0%BE%D0%B2%D0%BE%D0%B5_%D1%81%D1%82%D0%B5%D0%BA%D0%BB%D0%BE?redlink=1&veaction=edit&flow=create-page-article-redlink).

Алгоритм обучения сети Хопфилда имеет существенные отличия в сравнении с такими классическими алгоритмами обучения перцептронов как [метод коррекции ошибки](http://ru.cybernetics.wikia.com/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BA%D0%BE%D1%80%D1%80%D0%B5%D0%BA%D1%86%D0%B8%D0%B8_%D0%BE%D1%88%D0%B8%D0%B1%D0%BA%D0%B8) или [метод обратного распространения ошибки](http://ru.cybernetics.wikia.com/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BE%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B3%D0%BE_%D1%80%D0%B0%D1%81%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F_%D0%BE%D1%88%D0%B8%D0%B1%D0%BA%D0%B8). Отличие заключается в том, что вместо последовательного приближения к нужному состоянию с вычислением ошибок, все коэффициенты матрицы рассчитываются по одной формуле, за один цикл, после чего сеть сразу готова к работе. Вычисление коэффициентов базируется на следующем правиле: для всех запомненных образов X_i матрица связи должна удовлетворять уравнению

X_i^T=WX_i,

поскольку именно при этом условии состояния сети X_i будут устойчивы - попав в такое состояние, сеть в нем и останется.

Некоторые авторы относят сеть Хопфилда к [обучению без учителя](http://ru.cybernetics.wikia.com/wiki/%D0%9E%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B5%D0%B7_%D1%83%D1%87%D0%B8%D1%82%D0%B5%D0%BB%D1%8F). Но это неверно, т.к. обучение без учителя предполагает отсутствие информации о том, к каким классам нужно относить стимулы. Для сети Хопфилда без этой информации нельзя настроить весовые коэффициенты, поэтому здесь можно говорить лишь о том, что такую сеть можно отнести к классу оптимизирующих сетей (фильтров). Отличительной особенностью фильтров является то, что матрица весовых коэффициентов настраивается детерминированным алгоритмом раз и навсегда, и затем весовые коэффициенты больше не изменяются. Это может быть удобно для физического воплощения такого устройства, т.к. на схемотехническом уровне реализовать устройство с переменными весовыми коэффициентами на порядок сложнее. Примером фильтра без обратных связей может служить [алгоритм CC4](http://ru.cybernetics.wikia.com/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C_CC4?redlink=1&action=edit&flow=create-page-article-redlink) (Cornel classification), автором которого является S.Kak.

В сети Хопфилда есть обратные связи и из-за этого нужно решать проблему устойчивости. Веса между нейронами в сети Хопфилда могут рассматриваться в виде матрицы взаимодействий W. В работе [[6]](http://ru.cybernetics.wikia.com/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C_%D0%A5%D0%BE%D0%BF%D1%84%D0%B8%D0%BB%D0%B4%D0%B0#cite_note-AA-0) показано, что сеть с обратными связями является устойчивой, если ее матрица симметрична и имеет нули на главной диагонали. Имеется много устойчивых систем, например, все сети прямого распространения, а так же современные рекуррентные сети Джордана и Элмана, для которых не обязательно выполнять условие на симметрию. Но это происходит вследствие того, что на обратные связи наложены другие ограничения. В случае сети Хопфилда условие симметричности является необходимым, но не достаточным, в том смысле, что на достижение устойчивого состояния влияет еще и режим работы сети. Ниже будет показано, что только асинхронный режим работы сети гарантирует достижение устойчивого состояния сети, в синхронном случае возможно бесконечное переключение между двумя разными состояниями (такая ситуация называется динамическим [аттрактором](http://ru.cybernetics.wikia.com/wiki/%D0%90%D1%82%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BE%D1%80), в то время как устойчивое состояние принято называть статическим аттрактором).

Запоминаемые векторы должны иметь бинарный вид. После этого происходит расчет весовых коэффициентов по следующей формуле :

w_{ij}=\frac{1}{N}\sum_{d=1..m}X_{id}X_{jd},

где N - размерность векторов, m – число запоминаемых выходных векторов; d – номер запоминаемого выходного вектора; X_{ij} – i-я компонента запоминаемого выходного j-го вектора.

Это выражение может стать более ясным, если заметить, что весовая матрица W может быть найдена вычислением внешнего произведения каждого запоминаемого вектора с самим собой и суммированием матриц, полученных таким образом. Это может быть записано в виде

W=\frac{1}{N}\sum_{i}X_{i}^{T}X_{i},

где X_i – i-й запоминаемый вектор-строка.

Расчет этих весовых коэффициентов и называется обучением сети.

Как только веса заданы, сеть может быть использована для получения запомненного выходного вектора по данному входному вектору, который может быть частично неправильным или неполным. Для этого выходам сети сначала придают значения этого начального вектора. Затем сеть последовательно меняет свои состояния согласно формуле:

X_{i+1}^T=F(WX_i),

где F - активационная функция, X_i и X_{i+1} - текущее и следующее состояния сети, до тех пор, пока состояния X_i и X_{i+1} не совпадут (или, в случае синхронного режима работы, не совпадут состояния X_{i-1}с X_{i+1} и одновременно X_{i-2} с X_i). Именно этот процесс называется конвергенцией сети. Полученное устойчивое состояние X_i (статический аттрактор), или, возможно, в синхронном случае пара {X_i, X_{i+1}} (динамический аттрактор), является ответом сети на данный входной образ.

# 3.3.2 Сверточные нейронные сети

Идея сверточных нейронных сетей заключается в чередовании сверточных слоев (C-layers), субдискретизирующих слоев (S-layers) и наличии полносвязных (F-layers) слоев на выходе. На рисунке 1 представлен пример такой архитектуры.

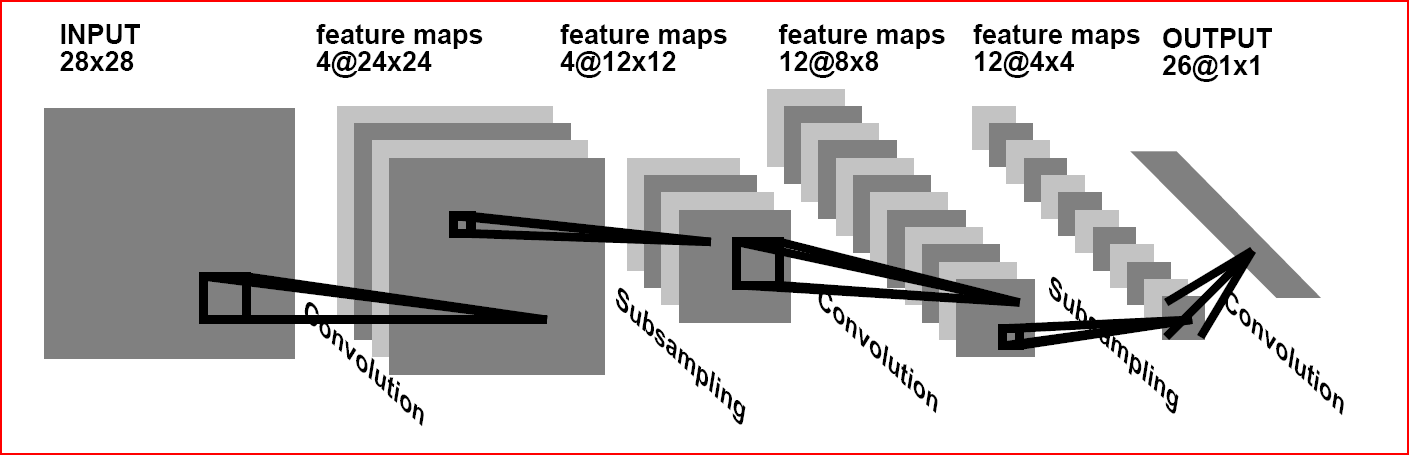


Рисунок 1 – Архитектура сверточных сетей

Такая архитектура заключает в себе 3 основных парадигмы: локальное восприятие, разделяемые веса, субдискретизация.

Локальное восприятие подразумевает, что на вход одного нейрона подается не все изображение (или выходы предыдущего слоя), а лишь некоторая его область. Такой подход позволил сохранять топологию изображения от слоя к слою.

Концепция разделяемых весов предполагает, что для большого количества связей используется очень небольшой набор весов. Т.е. если у нас имеется на входе изображение размерами 32х32 пикселя, то каждый из нейронов следующего слоя примет на вход только небольшой участок этого изображения размером, к примеру, 5х5, причем каждый из фрагментов будет обработан одним и тем же набором. Важно понимать, что самих наборов весов может быть много, но каждый из них будет применен ко всему изображению. Такие наборы часто называют ядрами (kernels). Даже для 10 ядер размером 5х5 для входного изображения размерами 32х32 количество связей окажется равным примерно 256000.

Искусственно введенное ограничение на веса улучшает обобщающие свойства сети (generalization), что в итоге позитивно сказывается на способности сети находить инварианты в изображении и реагировать главным образом на них, не обращая внимания на прочий шум. Данную систему можно упрощенно считать двухмерным фильтром. Фильтр представляет собой матрицу коэффициентов, обычно заданную вручную. Эта матрица применяется к изображению с помощью математической операции, называемой сверткой. Суть этой операции в том, что каждый фрагмент изображения умножается на матрицу (ядро) свертки поэлементно и результат суммируется и записывается в аналогичную позицию выходного изображения. Основное свойство таких фильтров заключается в том, что значение их выхода тем больше чем больше фрагмент изображения похож на сам фильтр. Таким образом изображение свернутое с неким ядром даст нам другое изображение, каждый пиксел которого будет означать степень похожести фрагмента изображения на фильтр. Иными словами это будет карта признаков.

Каждый фрагмент изображения поэлементно умножается на небольшую матрицу весов (ядро), результат суммируется. Эта сумма является пикселом выходного изображения, которое называется картой признаков. Взвешенная сумма входов также пропускается через функцию активации (как в любой другой нейросети). Следует сказать, что в идеале не разные фрагменты проходят последовательно через ядро, а параллельно все изображение проходит через идентичные ядра. Кроме того, количество ядер (наборов весов) определяется разработчиком и зависит от того какое количество признаков необходимо выделить. Еще одна особенность сверточного слоя в том, что он немного уменьшает изображение за счет краевых эффектов.

Суть субдискретизации и S-слоев заключается в уменьшении пространственной размерности изображения. Т.е. входное изображение грубо (усреднением) уменьшается в заданное количество раз. Чаще всего в 2 раза, хотя может быть и не равномерное изменение, например, 2 по вертикали и 3 по горизонтали. Субдискретизация нужна для обеспечения инвариантности к масштабу.

Чередование слоев позволяет составлять карты признаков из карт признаков, что на практике означает способность распознавания сложных иерархий признаков.

Обычно после прохождения нескольких слоев карта признаков вырождается в вектор или даже скаляр, но таких карт признаков становится сотни. В таком виде они подаются на один-два слоя полносвязной сети. Выходной слой такой сети может иметь различные функции активации. В простейшем случае это может быть тангенциальная функция, также успешно используются радиальные базисные функции.

Для данного типа сети также важен выбор функции активации нейронов и метода оптимизации. Для обработки изображений оптимальными считаются сигмоидальные функции, а в качестве метода оптимизации – метод ADADELTA.

Сигмоидальная функция или сигмоид - монотонно возрастающая всюду дифференцируемая S-образная нелинейная функция с насыщением. Сигмоид позволяет усиливать слабые сигналы и не насыщаться от сильных сигналов.

Слабые сигналы нуждаются в большом сетевом усилении, чтобы дать пригодный к использованию выходной сигнал. Однако усилительные каскады с большими коэффициентами усиления могут привести к насыщению выхода шумами усилителей, которые присутствуют в любой физически реализованной сети. Сильные входные сигналы в свою очередь также будут приводить к насыщению усилительных каскадов, исключая возможность полезного использования выхода. Каким образом одна и та же сеть может обрабатывать как слабые, так и сильные сигналы?

Примером сигмоидальной функции является логистическая функция:

 (2)

где e-x – параметр наклона сигмоидальной функции активации. Изменяя этот параметр, можно построить функции с различной крутизной.

В свою очередь, метод Adadelta позволяет:

* отказаться от ручного выбора параметра обучения;
* обеспечить нечувствительность к значениям параметров обучения;
* производить обучение с различной скоростью по каждому из параметров;
* уменьшить количество итераций при градиентном спуске;
* снизить зависимость обучения от начальных условий;
* возможность использовать один алгоритм для глобального и локального поиска.

Алгоритм имеет следующий вид. Сначала рассчитывается экспоненциальное среднее градиента:

https://www.science-education.ru/i/2015/1/15266/image017.png,

https://www.science-education.ru/i/2015/1/15266/image018.png

где https://www.science-education.ru/i/2015/1/15266/image019.png – коэффициент затухания во времени, https://www.science-education.ru/i/2015/1/15266/image020.png– параметр для улучшения свойств знаменателя. В работе Цейлера указывается, что работа алгоритма не зависит существенным образом от выбора этих параметров.

После чего рассчитывается текущее приращение

https://www.science-education.ru/i/2015/1/15266/image021.png

И затем рассчитывается экспоненциальное среднее приращений

https://www.science-education.ru/i/2015/1/15266/image022.png

Выбор параметра n происходит в автоматическом режиме в зависимости от хода процесса обучения, и при решении задачи многомерной оптимизации процесс обучения управляется по всем размерностям в отдельности.

# 3.4 Базовая имплементация алгоритма

Спроектированная сеть состоит из чередующихся слоев двух типов (всего 9 слоев – сверточных двухмерных (Conv2D и UpSampling2D) и субдискретизирующих двухмерных (MaxPooling2D), так как целью является обработка двухмерных изображений). Conv2D – слой, создающий ядро свертки, которое свернуто со слоем ввода для создания тензора выходов (под тензором в TensorFlow понимается абстрактный вектор или матрица, в которой все элементы имеют одинаковый тип), а UpSampling2D – вспомогательный слой для масштабирования изображения с целью повышения разрешения. Структура сети представлена на рисунке 2.

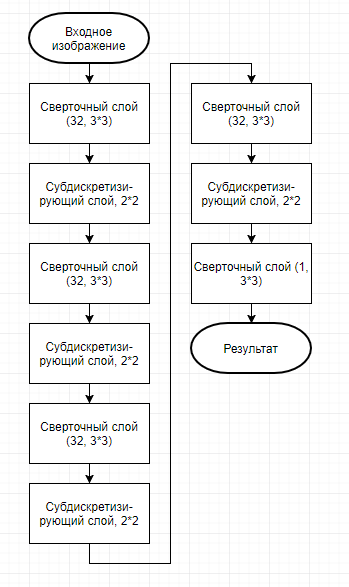


Рисунок 2 – Базовый вариант структуры сверточной сети.

# Список литературы

1 Классификация нейронных сетей [Электронный ресурс] Режим доступа: <http://www.aiportal.ru/articles/neural-networks/classification.html>

2 We're making «real progress»' on artificial intelligence [Электронный ресурс] Режим доступа: <https://www.csmonitor.com/Technology/2015/0914/Google-chairman-We-re-making-real-progress-on-artificial-intelligence>

3 Theano at a glance [Электронный ресурс] Режим доступа: <http://www.deeplearning.net/software/theano/introduction.html>

4 Keras documentation [Электронный ресурс] Режим доступа: <http://keras.io>

5 Нейронная сеть Хопфилда [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/Нейронная_сеть_Хопфилда>

6 Sebastian Raschka Python Machine Learning / Sebastian Raschka - Birmingham, UK – Packt Publishing, 2015, 425 c.

# Список публикаций

Юров Д.А., Кремер О.Б. Разработка геоинформационной системы о высших учебных заведениях для мобильных устройства -  Болгария, Варна: ЦЕНТЪР ЗА НАУЧНИ ИЗСЛЕДВАНИЯ И ИНФОРМАЦИЯ "ПАРАДИГМА", 2016. - с. 94-105, УДК: 004.42

Юров Д.А., Кремер О.Б. Геоинформационная система «ВУЗЫ ВОРОНЕЖА» для операционной системы Android - Воронеж: Студент и наука №2 Воронежский ГТУ, 2017. - c. 144-147, УДК 681.3.069

Юров Д.А., Кремер О.Б. Организация эффективного поиска информации в ходе разработки ГИС для мобильных устройств -

Воронеж: Научная книга, 2016. - с. 244-249, УДК: 004.42