МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Кафедра прикладной математики

КУРСОВАЯ РАБОТА

ПРИМЕНЕНИЕ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ЗАДАЧИ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

Работу выполнил _____ С.Н. Капсамун
(подпись, дата) (инициалы, фамилия)

Факультет компьютерных технологий и прикладной математики курс 3

Направление 09.03.03 Прикладная математика и информатика

Научный руководитель,
преподаватель _____ М.Х. Уртенов
(подпись, дата) (инициалы, фамилия)

Нормоконтролер,
к. ф. - м. н _____ Г.В. Калайдина
(подпись, дата) (инициалы, фамилия)

Краснодар 2018

РЕФЕРАТ

Курсовая работа 43с., 18 рис., 6 источника.

СВЕРТОЧНАЯ НЕЙРОНАЯ СЕТЬ, KERAS, TANSERFLOW,OPENCV, КЛАССИФИКАЦИЯ.

Цель работы: изучить сверточную нейронную сеть для классификации пшеницы, а так же исправить проблемы связанные с неточными результатами.

В курсовой работе были изложены основные проблемы, связанные с данной темой и рассмотрен метод для решения данной проблемы. Были изучены основные принципы работы

СОДЕРЖАНИЕ

ВВЕДЕНИЕ..... 5

1 История нейронной сети 7

2 Архитектура свёрточной нейронной сети 9

2.1 Свёрточный слой 13

2.1.1 Згорткових слой 13

2.1.2 Локальная соединенность 14

2.1.3 Пространственная организация..... 14

2.1.4 Совместное использование параметров 16

2.2 Методы Регуляризации 17

2.2.1 Понимание регуляризации и переобучения..... 17

2.2.2 Эмпирические 19

2.2.2.1 Исключение……19

2.2.2.2 Исключение соединений……..20

2.2.2.3 Стохастическая подвыборка……21

2.2.2.4 Искусственные данные…….21

2.3 Явные сети..... 22

2.3.1 Размер Сети 22

2.4 Метод Обучения Сети 23

2.4.1 Вычисление Ошибки 23

2.4.2 Вычисление Градиента 25

3 Структура Программы..... 27

3.1 Изучение программы..... 27

3.2 Обучение с учителем..... 31

3.3 Нейрон..... 33
3.4 Обучение и тестирование 36
ЗАКЛЮЧЕНИЕ 42
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43

ВВЕДЕНИЕ

Искусственные нейронные сети (ИНС) – математические модели, а также их программные или аппаратные реализации, построенные 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. Первой такой попыткой были нейронные сети Маккалока и Питтса. 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 локально 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. С 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. А с точки зрения искусственного интеллекта, ИНС является основой

философского течения коннективизма и 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 искаженных данных.

Способности нейронной сети к прогнозированию напрямую следуют из ее способности к обобщению и выделению скрытых зависимостей между входными и выходными данными. После обучения сеть способна предсказать будущее значение некой последовательности на основе нескольких предыдущих значений и/или каких – то существующих в настоящий момент факторов. Следует отметить, что прогнозирование возможно только тогда, когда предыдущие изменения действительно в какой-то степени предопределяют будущие. Например, прогнозирование котировок акций на основе котировок за прошлую неделю может оказаться успешным (а может и не оказаться), тогда как прогнозирование результатов завтрашней лотереи на основе данных за последние 50 лет почти наверняка не даст никаких результатов.

1 История нейронной сети

Разработка сверточных нейронных сетей последовала за открытием зрительных механизмов в живых организмах. Ранняя работа 1968 г. показала, что зрительная кора животных содержит сложные компоновки клеток, чувствительные к выявлению света в малых подобластей зрительного поля, которые перекрываются, названных рецептивными полями. Эта работа идентифицировала два основных типа клеток: простые клетки, реагирующие максимально особых контуроподибни образы в их рецептивных полях, и сложные клетки, имеющих большие рецептивные поля, и является локально инвариантными к точного положения образа. Эти клетки действуют как локальные фильтры над пространством входа. Неокогнитрон, предшественник сверточных сетей, был представлено в работе 1980 года. В1988 году их количество было разработано отдельно, с явными параллельными и способными к обучению свертками для сигналов, распространяющихся во времени. Их конструкцию была усовершенствована в 1998 года, обобщена в 2003 и того же года упрощена. Знаменитая сеть LeNet-5 может успешно классифицировать цифры, и применяется для распознавания цифр в чеках. Однако при более сложных задачах широта размаха и глубина сети растут, и становятся ограниченными вычислительными ресурсами и сдерживают производительность. В 1988 году был предложен отличную конструкцию ОНР для разложения одномерных сигналов электромиографии. 1989 года она была видоизменена для других схем на основе свертки.

С появлением эффективных вычислений на ГП стало возможным тренировать большие сети. 2006 несколько публикаций описали эффективные пути тренировки згорткових нейронных сетей с большим количеством слоев. 2011 года их было уточнено и реализовано на ГП , с впечатляющими результатами. 2012 Чирешан и др. значительно

усовершенствовали лучшую производительность в литературе для нескольких наборов изображений, включая базы данных MNIST, базой данных NORB, набором символов HWDB1.0 (китайские символы), набором данных CIFAR10 (набор из 60000 меченых изображений RGB 32 × 32) и набором данных ImageNet.

2 Архитектура свёрточной нейронной сети

В обычном перцептроне, который представляет собой полносвязную нейронную сеть, каждый нейрон связан со всеми нейронами предыдущего слоя, причем каждая связь имеет свой персональный весовой коэффициент. В свёрточной нейронной сети в операции свёртки используется лишь ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою (в самом начале – непосредственно по входному изображению), формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются общие веса – матрица весов, которую также называют набором весов или ядром свёртки. Она построена таким образом, что графически кодирует какой – либо один признак, например, наличие наклонной линии под определенным углом. Тогда следующий слой, получившийся в результате операции свёртки такой матрицей весов, показывает наличие данной наклонной линии в обрабатываемом слое и её координаты, формируя так называемую карту признаков. Естественно, в свёрточной нейронной сети набор весов не один, а целая гамма, кодирующая всевозможные линии и дуги под разными углами. При этом такие ядра свертки не закладываются исследователем заранее, а формируются самостоятельно путем обучения сети классическим методом распространения ошибки. Проход каждым набором весов формирует свой собственный экземпляр карты признаков, делая нейронную сеть многомерной (много независимых карт признаков на одном слое). Также следует отметить, что при переборе слоя матрицей весов её передвигают обычно не на полный шаг (размер этой матрицы), а на небольшое расстояние. Так, например, при размерности матрицы весов 5×5 её сдвигают на один или два нейрона (пикселя) вместо пяти, чтобы не «перешагнуть» искомый признак.

Операция субдискретизации (англ. subsampling, англ. pooling, также переводимая как «операция подвыборки» или операция объединения), выполняет уменьшение размерности сформированных карт признаков. В данной архитектуре сети считается, что информация о факте наличия искомого признака важнее точного знания его координат, поэтому из нескольких соседних нейронов карты признаков выбирается максимальный и принимается за один нейрон карты признаков уменьшенной размерности. Также иногда применяют операцию нахождения среднего между соседними нейронами. За счёт данной операции, помимо ускорения дальнейших вычислений, сеть становится более инвариантной к масштабу входного изображения.

Таким образом, повторяя друг за другом несколько слоёв свёртки и субдискретизации строится свёрточная нейронная сеть. Чередование слоёв 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. На выходе сети часто дополнительно устанавливают несколько слоев полносвязной нейронной сети (перцептрон), на вход которому подаются оконечные карты признаков.

Если на первом слое ядро свёртки проходит только по одному исходному изображению, то на внутренних слоях одно и то же ядро проходит параллельно по всем картам признаков этого слоя, а результат свертки суммируется, формируя (после прохождения функции активации) одну карту признаков следующего слоя, соответствующую этому ядру свертки.

Модель свёрточной сети (Рисунок 1), которую рассмотрим, состоит из трёх типов слоёв: свёрточные слои, субдискретизирующие (subsampling,подвыборка) слои и слои "обычной"нейронной сети –

перцептрона.

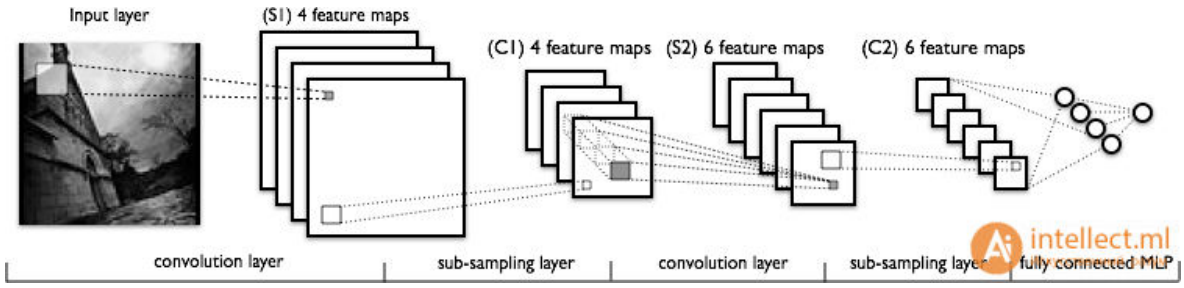

Рисунок 1 – Схема свёрточной сети

Первые два типа слоёв (convolutional,subsampling), чередуясь между собой, формируют входной вектор признаков для многослойного перцептрона. Сеть можно обучать с помощью градиентных методов. Своё название свёрточная сеть получила по названию операции – свёртка, она часто используется для обработки изображений и может быть описана следующей формулой:

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l] \cdot g[k,l]$$

Здесь f - исходная матрица изображения, g - ядро (матрица) свёртки.

Неформально эту операцию можно описать следующим образом - окном размера ядра g проходим с заданным шагом (обычно 1) всё изображение f, на каждом шаге поэлементно умножаем содержимое окна на ядро g, результат суммируется и записывается в матрицу результата.

При этом в зависимости от метода обработки краёв исходной матрицы результат может быть меньше исходного изображения (Рисунок - 2), (valid), такого же размера (Рисунок - 3) (same) или большего размера (Рисунок - 4) (full).

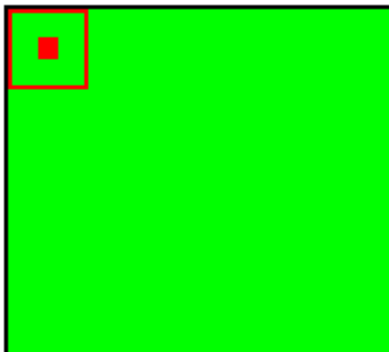

Рисунок 2 – Обработка краёв valid

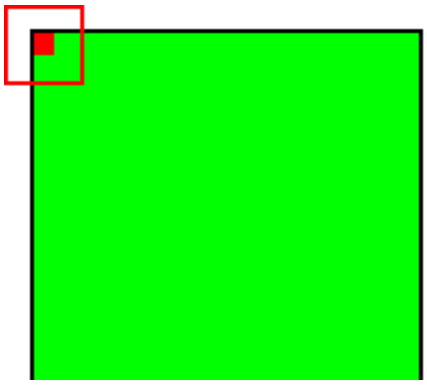

Рисунок 3 – Обработка краёв same

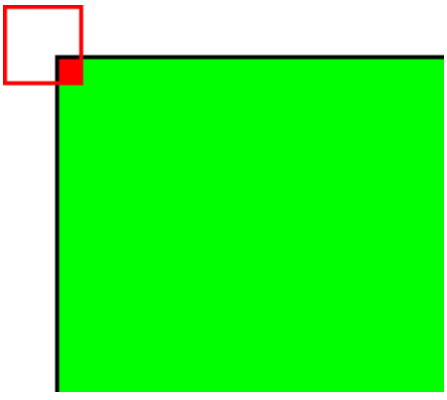

Рисунок 4 – Обработка краёв full

2.1 Свёрточный слой

В этом разделе мы рассмотрим устройство свёрточного слоя.

Свёрточный слой реализует идею т.н. локальных рецептивных полей, т.е. каждый выходной нейрон соединен только с определённой (небольшой) областью входной матрицы и таким образом моделирует некоторые особенности человеческого зрения.

В упрощённом виде этот слой можно описать формулой:

$$x^l = f(x^{l-1} * k^l + b^l)$$

Здесь x^l - выход слоя l, f() - функция активации, b - коэффициент сдвига, символом $*$ обозначена операция свёртки входа x с ядром k.

При этом за счёт краевых эффектов размер исходных матриц уменьшается по формуле:

$$x_j^l = f(\sum_i x_i^{l-1} * k_j^l + b_j^l)$$

Здесь x_j^l - карта признаков j (выход слоя l), f() - функция активации, b j - коэффициент сдвига для карты признаков j, k_j - ядро свёртки номер j, x_j^{l-1} - карты признаков предыдущего слоя.

2.1.1 Згорткових слой

Згорткових слой (англ. Convolutional layer) является основным строительным блоком ОНР. Параметры слоя состоят из набора фильтров для обучения (или ядер), которые имеют небольшое рецептивное поле, но простираются на всю глубину входного объема. В течение прямого прохода

каждый фильтр осуществляет свертку по ширине и высоте входного объема, вычисляя скалярное произведение данных фильтра и входа, и формируя 2 - мерную карту активации этого фильтра. В результате сеть учится, какие фильтры активируются, когда они видят определенный конкретный тип признаки в определенном пространственном положении при входе.

Составление активационных карт всех фильтров вдоль измерения глубины формирует полный исходный объем сверточного слоя. Таким образом, каждая запись в исходном объеме может также трактоваться как выход нейрона, который смотрит на небольшую область в входе, и делит параметры с нейронами той же активационной карты.

2.1.2 Локальная соединенность

При обработке входов высокой размерности, таких как изображения, нецелесообразно соединять нейроны со всеми нейронами предыдущего объема, поскольку такая архитектура сети не принимает во внимание пространственную структуру данных. Сверточные сети используют пространственно локальную корреляцию путем обеспечения схемы локальной зьеднаности между нейронами соседних слоев: каждый нейрон соединяется лишь с небольшой областью входящего объема. Весь объем этой зьеднаности
является гиперпараметром, что называется рецептивным полем нейрона.
Соединение являются локальными в пространстве (вдоль ширины и высоты), но всегда распространяются вдоль всей глубины входящего объема. Такая архитектура обеспечивает, чтобы обученные фильтры производили сильный отклик на пространственно локальных входных образов.

2.1.3 Пространственная организация

Размер выходного объема сверточного слоя контролируют три гиперпараметры: глубина , шаг и нулевое дополнения .

Глубина выходного объема контролирует количество нейронов слоя соединяются с одной и той же областью входящего объема. Все эти нейроны учиться активироваться для различных признаков входа. Например, если первый згорткових слой берет как вход сырое изображение, то различные нейроны вдоль измерения глубины могут активироваться в присутствии различных ориентированных контуров, или пятен цвета.

Шаг контролирует то, как столбики глубины распределяются по пространственными измерениями (шириной и высотой). Когда шагом является 1, то глубокие столбики нейронов размещаются в пространственных положениях на расстоянии всего 1 пространственной единицы друг от друга. Это ведет к сильному перекрытия рецептивных полей между столбиками, а также к большим выходных объемов. И наоборот, если применяются большие шаги, то рецептивные поля перекрываться меньше, и получаемый в результате выходной объем будет меньше пространственные размеры.

Иногда удобно дополнять вход нулями по краям входного объема. Размер этого нулевого дополнение является третьим гиперпараметром. Нулевое дополнение позволяет контролировать пространс твенный размер выходных объемов. В частности, иногда желательно точно сохранять пространственный размер входного объема.

Пространственный размер выходного объема может исчисляться как функция от размера входного объема W, размера рецептивного поля нейронов сверточного слоя F, шага, с которым они применяются, S и величины нулевого дополнения P, применяемой на краях. Формула для вычисления того, сколько нейронов «умещается» в заданного объема, задается формулой:

$$(W - F + 2P)/S + 1.$$

Если это число не является целым, то шаги установлено неправильно, и нейроны не может быть размещено вдоль входного объема симметричным образом. В общем, установление нулевого дополнения когда шагом является S=1 , обеспечивает, чтобы входной и выходной объемы имели одинаковый пространственный размер.

2.1.4 Совместное использование параметров

Схема совместного использования параметров применяется в згорткових слоях для того, чтобы контролировать количество свободных параметров. Она опирается на одно разумное предположение: если одна лоскутное признак является полезной для вычисления в определенном пространственном положении, то она также должна быть полезной для вычисления в другом положении. Иными словами, обозначая двухмерный срез по глубине как срез глубины , мы ограничиваем нейроны в каждом срезе глубины использованием одних и тех же веса и предубеждения.

Поскольку все нейроны в едином срезе делят между собой одну и ту же параметризации, то прямой проход в каждом срезе глубины сверточного (англ. CONV) слоя может быть вычислено как свертке весовых коэффициентов нейронов с входным объемом (отсюда и название: згорткових слой) . Таким образом, является обычным называть наборы весовых коэффициентов фильтром (или ядром), который сворачивается с входом. Результатом этой свертки является активационная карта, и набор активационных карт для каждого из различных фильтров состоит вместе вдоль измерения глубины для получения исходного объема. Совместное использование параметров способствует инвариантности архитектуры ОНР относительно сдвига.

Важно заметить, что допущение совместного использования параметров иногда может не иметь смысла. Особенно в том случае, коды входные изображения в ОНР имеют определенную особую центрированную структуру, в которой мы ожидаем обучение совершенно разных признаков в различных пространственных положениях. Одним из практических примеров является когда вход является лицами, было отцентрирован в картинке: мы можем ожидать, что учиться различных особых признаков глаз и волос в разных частях изображения. В таком случае является обычным смягчать схему совместного использования параметров, на фоне просто называть слой локально соединенным слоем.

2.2 Методы Регуляризации

2.2.1 Понимание регуляризации и переобучения

Суть переобучения (overfitting) - Если слишком интенсивно обучать модель, то в конечном счете получите весовые значения, чрезвычайно точно подходящие к обучающим данным, но, когда применим полученную модель к новым данным, точность прогноза окажется весьма скверной.

Переобучение проиллюстрировано двумя графиками на Рисунок - 5. На первом графике показана гипотетическая ситуация, где цель заключается в классификации двух типов элементов, обозначенных черными и серыми точками. Плавная светло-серая кривая представляет истинное разделение двух классов с черными точками над кривой и серыми точками под кривой. Заметим: из - за случайных ошибок в данных две из черных точек находятся под кривой, а две серых - над ней. При хорошем обучении (без переобучения) получились бы весовые значения, которые соответствуют плавной светло-серой кривой. Допустим, что были вставлены новые данные в (3, 7). Элемент данных окажется над кривой и будет правильно

спрогнозирован как относящийся к черному классу.

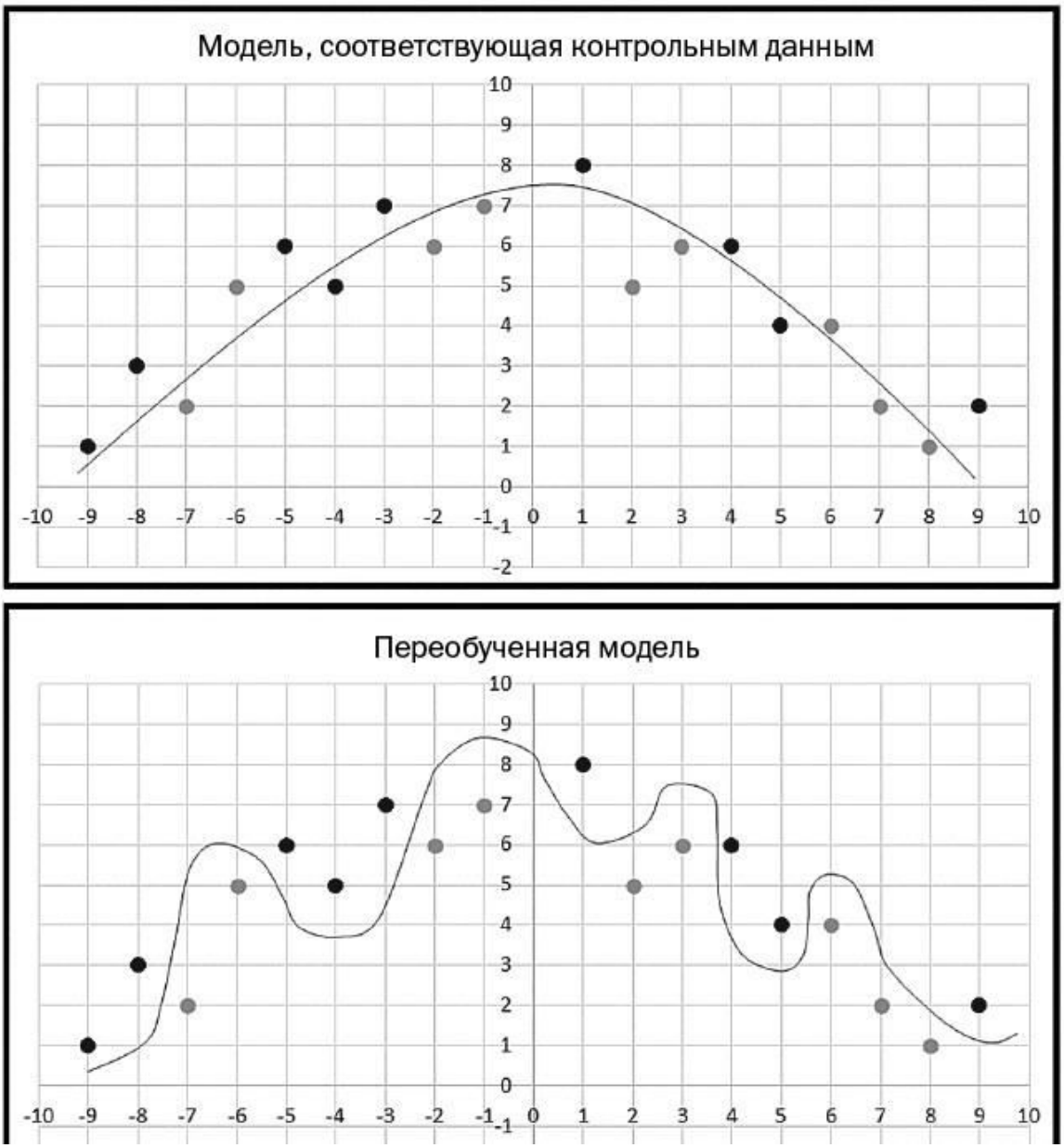

Рисунок – 5 Переобучение модели

На втором графике Рисунок - 5 мы видим те же точки, но другую кривую, которая является результатом переобучения. На этот раз все черные точки находятся над кривой, а все зеленые – под ней. Но кривая слишком сложна. Новый элемент данных в (3, 7) оказался бы под кривой и был бы неправильно спрогнозирован как серый класс.

Переобучение дает неплавные кривые прогнозирования, т. е. «нерегулярные». Такие плохие сложные кривые прогнозирования обычно характеризуются весовыми значениями, которые имеют очень большие или очень малые величины. Поэтому один из способов уменьшить степень переобучения состоит в том, чтобы не допускать очень малых или больших весовых значений для модели. В этом и заключается суть регуляризации.

При обучении ML-модели нужно использовать некую меру ошибки, чтобы определять хорошие весовые значения. Есть несколько способов измерения ошибок.

2.2.2 Эмпирические

2.2.2.1 Исключение

Поскольку полносвязный слой имеет больше всего параметров, он склонен к переобучению. Для предотвращения переобучению введено метод исключения (англ. Dropout). На каждом этапе тренировки отдельные узлы или «исключаются» из сети с вероятностью 1-P или остаются с вероятностью P, так что остается уменьшена сеть; входные и выходные ребра исключенных узлов также устраняются. На следующем этапе на данных тренируется уже только уменьшена сеть. После этого устранены узлы повторно вставляются в сети с их первичными весовыми коэффициентами.

На этапах тренировки вероятностью оставление (то есть, не исключение) скрытого узла обычно 0.5; для входных узлов вероятность оставления должна быть намного выше, интуитивно том, что при игнорировании входных узлов происходит непосредственная потеря информации.

Во время проверки после завершения тренировки мы в идеале хотели бы найти выборочное среднее всех возможных 2^n сетей с исключениями к

сожалению, это невыполнимой для больших значений .Тем не менее, мы можем найти приближения, используя полную сеть, в которой выход каждого узла взвешенно на коэффициент P, так что математическое ожидание значения выхода любого узла будет таким же, как и на этапах тренировки. Это самым большим вкладом метода исключения: хотя он эффективно порождает 2^n нейронных сетей, и таким образом делает сочетание моделей, при проверке проверять нужно только одну сеть.

Избегая обучение всех узлов на всех тренировочных данных, исключения снижает переобучение нейронных сетей. Этот метод также значительно улучшает скорость обучения. Это делает сочетание моделей практичным, даже для глубинных нейронных сетей. Похоже, что эта методика ослабляет сложные, тесно подогнаны взаимодействия между узлами, ведя их к обучению надежных признаков, лучше обобщаются на новые данные. Было показано, что исключение улучшает производительность нейронных сетей в задачах в видении, распознавании речи, классификации документов и в вычислительной биологии.

2.2.2.2 Исключение Соединений

Исключение соединений (англ. DropConnect) является обобщением исключения (англ. Dropout), в котором каждое соединение, а не каждый выходной узел, может быть исключен из вероятностью 1-P. Таким образом, каждый узел получает вход со случайной подмножества узлов предыдущего слоя.

Исключение соединений подобен исключения тем, что оно вводит в модели динамическую разреженность, но отличается тем, что вероятность есть на весовых коэффициентах, а не на выходных векторах слоя. Иными словами, полносвязный слой с исключением соединений становится разреженно соединенным слоем, в котором соединение избираются случайно

во время этапа тренировки.

2.2.2.3 Стохастическая Подвыборка

Главным недостатком исключения является то, что оно не имеет таких же преимуществ для згорткових слоев, где нейроны не является полносвязным.

В стохастической пидвибирци (англ. Stochastic pooling) обычные дете рминированы действия подвыборки заменяются стохастической процедурой, в которой активация в пределах каждой области подвыборки выбирается случайно, в соответствии с полиномиального распределения , заданного показателям в пределах области подвыборки. Этот подход является свободным от гиперпараметрив, и может сочетаться с другими подходами к регуляризации, такими как исключение, и наращивание данных.

Альтернативным взглядом на стохастической подвыборку является то, что она является равнозначной стандартной максимизацийний пидвибирци, но со многими копиями входного изображения, каждая из которых имеет небольшие локальные деформации. Это подобным явных эластичных деформаций входных изображений, которые обеспечивают отличную производительность в MNIST. Применение стохастической подвыборки в многослойной модели дает экспоненциальное число деформаций, поскольку выборы в высших слоях зависят от выборов в низших.

2.2.2.4 Искусственные данные

Поскольку степень переобучения модели определяется как ее мощности, так и количеством получаемого ею тренировки, обеспечение згорткових сети дополнительными тренировочными примерами может снизить переобучение. Поскольку эти сети обычно уже натренированно

всеми имеющимся данным, одним из подходов является или порождать новые примеры с нуля (если это возможно), или будоражить имеющиеся тренировочные образцы для создания новых. Например, входные изображение может быть асимметрично обрезаемая на несколько процентов для создания новых примеров с таким же маркером, как и первичный.

2.3 Явные сети

2.3.1 Размер сети

Самым простым способом предотвратить переобучению сети просто ограничить количество скрытых узлов и свободных параметров (соединений) в ней. Это непосредственно ограничивает предсказательную мощность сети, снижая сложность функции, которую она выполняет в данных, и поэтому ограничивает размер переобучение. Это равнозначно «нулевой норме».

2.3.2 Ослабление веса

Простым видом добавленного регуляризатора является ослабление веса (англ. Weight decay), которое просто добавляет к погрешности каждого узла дополнительную погрешность, пропорциональную сумме весовых коэффициентов (норма L1) или квадратном ступенчатые (норма L2) вектора весовых коэффициентов. Уровень приемлемой сложности модели может быть снижена увеличением постоянной пропорциональности, тем самым увеличивает штраф за большие векторы весовых коэффициентов.

L2 -регуляризация (англ. L2 regularization) является, возможно, самым распространенным видом регуляризации. Она может быть реализовано штрафованием квадратного степени всех параметров непосредственно в цели. L2 -регуляризация имеет интуитивную интерпретацию в сильном штрафовании пиковых весовых векторов, и отдаче преимущества

рассеянным весовым векторам. В связи с многократными взаимодействиями между весами и входами, это имеет привлекательную свойство поощрения сети использовать все ее входы понемногу, вместо сильного использования только некоторых из них.

L1 - регуляризация (англ. L1 regularization) является другим относительно распространенным видом регуляризации. L1 -регуляризацию возможно сочетать с L2 -регуляризацией (это называется эластично-сетевым регуляризации, англ. Elastic net regularization). L1 -регуляризация имеет такую увлекательную свойство , что она ведет весовые векторы до вступления разреженности течение оптимизации. Иными словами, нейроны с L1 -регуляризациею заканчивают использованием только разреженной подмножества их важнейших входов, и становятся почти инвариантными относительно зашумленных входов.

Другим видом регуляризации является навязывание абсолютной верхней границы величине весового вектора для каждого нейрона, и применение метода проекционного наискорейшего спуска для обеспечения этого ограничения. На практике это соответствует выполнению уточнения параметров как обычно, а потом обеспечению ограничения зажатием весового вектора $\vec{w}$ каждого нейрона, чтобы он удовлетворял $\|\vec{w}\|_2 < c$.

2.4 Метод обучения сети

2.4.1 Вычисление ошибки

Для выходного (MLP) слоя ошибка рассчитывается формулой:

$$\delta = (T - Y) \cdot f'(u)$$

Здесь T - ожидаемый (учебный) выход , Y - реальный выход, f'(u) -

производная функции активации по её аргументу.

Для скрытых слоёв MLP ошибка имеет следующий вид:

$$\partial^{l-1} = (W^l)^T \cdot \delta^l \cdot f'(u^{l-1})$$

Здесь δ^l - ошибка слоя l, f′(u^l) - производная функции активации, u^l - состояние (не активированное) нейронов слоя l, W^l - матрица весовых коэффициентов слоя l.

Ошибка на выходе свёрточного слоя формируется путём простого увеличения размера матриц ошибки следующего за ним субдискретизирующего слоя, описывается формулой:

$$\delta^{l-1} = upsample(\delta^l) \cdot f'(u^{l-1})$$

Здесь δ^l - ошибка слоя l, f′(u^l) - производная функции активации, u^l - состояние (не активированное) нейронов слоя l, upsample() - операция увеличения размера матриц.

Ошибка на выходе субдискретизирующего слоя рассчитывается путём выполнения «обратной свёртки» карт признаков следующего за ним свёрточного слоя, т.е. над каждой картой признаков выполняется свёртка с соответствующим ”перевернутым” ядром, при этом за счёт краевых эффектов размер исходных матриц увеличивается. Далее над получившимися картами вычисляются несколько частичных сумм по числу ядер свертки, в соответствии с матрицей смежности субдискретизирующего и свёрточного слоёв, описывается формулой:

$$\delta^{l-1} = f'(u^{l-1}) \cdot \sum \delta^l * rot180(k)$$

Здесь δ^l - ошибка слоя l, f′(u^l) - производная функции активации, u^l - состояние (не активированное) нейронов слоя l, k - ядра свёртки.

2.4.2 Вычисление градиента

В этом разделе описана процедура вычисления градиента функции потери сети, т.е. направления максимального роста функции потери. Обучение сводиться к её минимизации в пространстве параметров (весов) сети.

Градиент для ядра свёртки можно посчитать, как свёртку матрицы входа свёрточного слоя с "перевёрнутой" матрицей ошибки для выбранного ядра, описывается формулой:

$$\Delta k_j^l = rot180(x^{l-1} * rot180(\partial_j^l))$$

Здесь ∂^l - ошибка слоя l, x^l - вход слоя l, k - ядра свёртки.

Градиент для сдвига для свёрточного слоя вычисляется как сумма значений соответствующей матрицы ошибки:

$$\Delta b_j^l = \sum \partial_j^l$$

Градиент для коэффициентов субдискретизирующего слоя вычисляется формулой:

$$\Delta a_j^l = \partial_j^l \cdot subsample(x^{l-1})$$

Здесь x^l - выход слоя l, ∂^l - ошибка слоя l, subsample() - операция выборки локальных максимальных значений.

Градиент для коэффициента сдвига для субдискретизирующего слоя вычисляется, как сумма значений соответствующей матрицы ошибки:

$$\Delta b_j^l = \sum \partial_j^l$$

Здесь ∂^l - ошибка слоя l

Градиент для весов MLP вычисляется следующим образом:

$$\Delta W^l = (\partial^l)^T \cdot x^{l-1}$$

Здесь ∂^l - ошибка слоя l, x^l - вход слоя l, W^l - матрица весовых коэффициентов слоя l.

3 Структура программы

3.1 Изучение программы

Альтернативным взглядом на стохастической подвыборку было принято взять за основу среду в качестве инструмента программирования Python, как среду для изучения нейронной сети.

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Данная среда программирования была выбрана по нескольким критериям:

- Одним из первых факторов среды Python это легкость программирования. Чтобы научиться писать на Python не нужно тратить колоссальное время для обучения, если сравнивать с остальными языками программирования. Python имеет предельно простой синтаксис, на нем легко писать, его легко читать
- Количество справочной литературы: книг, сайтов, платных и бесплатных курсов, готовых шаблонов и исходников. На всех стадиях самообразования и работы не нужно испытывать недостатка в современных и актуальных информационных и развлекательных ресурсах, посвящённых языку Python.
- Всеобщая доступность Python - множество доступных сред разработки, сервисов и фреймворков. Легко найти качественный и бесплатный продукт для работы дома, в офисе и в дороге.

Развитие Python позволяет создавать сложный код, не затрачивая на это

много временных и «строчных» ресурсов. Можно использовать динамическую типизацию для упрощения кода и встроенные функции языка, чтобы избавиться от написания шаблонных кодов.

В качестве библиотеки для машинного обучения была взята открытая программная библиотека TensorFlow. Надстройку для фреймворка TensorFlow осуществляет открытая нейросетевая библиотека Keras.

TensorFlow – это нейронная сеть, которая учится решать задачи путем позитивного усиления и обрабатывает данные на различных уровнях (узлах), что помогает находить верный результат.

Преимущества фреймворка TensorFlow:

- TensorFlow поддерживает множество языков программирования, что делает возможным использования данного инструментария, даже не имея большого опыта в других средах программирования.

- Большое количеств руководств для изучения и тренировочных материалов, TensorFlow, обладает написанными туториалом, его легко понять и использовать на практике.

- Сверточные нейронные сети используются для распознавания изображений, рекомендательных систем, а также обработки естественного языка (NLP). CNN состоят из набора различных слоев, преобразующих входные данные в оценки для зависимой переменной с заранее известными классами. В результате анализа, легкость в построении моделей используя TensorFlow, временная верстка CNN в Torch выделяют данный фреймворк среди остальных.

- Рекуррентные нейронные сети (RNN) используются для распознавания речи, прогнозирования временных рядов, захвата изображений и решения других задач, в которых требуется обработка последовательной информации. Tensorflow имеет некоторый набор материалов по RNN, а TFLearn и Keras включают в себя большое количество примеров RNN, использующих TensorFlow.

- TensorFlow имеет легкий в использовании, модульный пользовательский интерфейс, создавая интуитивно понятную среду для разработки.

- TensorFlow показывает наилучший результаты при тестировании производительности сверточных нейронных сетей.

Keras является высокоуровневой нейронной сетью API, написанный на Python и может работать поверх TensorFlow, CNTK или Теано . Он был разработан с упором на возможность быстрого экспериментирования.

Используя библиотеку глубокого обучения Keras, которая:

- Позволяет легко и быстро создавать прототипы (благодаря удобству, модульности и расширяемости).

- Поддерживает как сверточные сети, так и повторяющиеся сети, а также комбинации этих двух.

- Легко работает на процессоре и графическом процессоре.

Преимущества библиотеки Keras:

- Удобство для пользователя. Keras следует наилучшим методам снижения когнитивной нагрузки: он предлагает последовательные и простые API, он минимизирует количество действий пользователя, необходимых для случаев общего использования, и обеспечивает четкую и эффективную обратную связь с ошибкой пользователя.

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

- Легкая растяжимость. Новые модули просто добавлять (как новые классы и функции), а существующие модули предоставляют множество примеров. Чтобы иметь возможность легко создавать новые

модули, вы можете полностью выразить свою выразительность, что делает Keras подходящим для передовых исследований.

- Работа с Python. Нет отдельных файлов конфигурации моделей в декларативном формате. Модели описаны в коде Python, который компактен, легче отлаживается и обеспечивает простоту расширяемости.

Перед тем как как обучить нейронную сеть, надо сначала разобраться во - первых, что она будет делать, а во - вторых каким методом ее надо обучить.

Программа должна классифицировать листья пшеницы. Иными словами она должна отличать здоровые листья пшеницы, от заболевших листьев.

Рассмотрим (Рисунок - 6) и (Рисунок - 7). Посмотрев на две фотографии, которые находятся ниже, можно найти различия. В (Рисунок - 6) листья имеют насыщенный зеленый цвет, а посмотрев на (Рисунок - 7) можно увидеть желтые оттенки цвета.

Рисунок – 6 Листья здоровой пшеницы

Рисунок – 7 Листья больной пшеницы

Чтобы обучить нейронную сеть отличать здоровые листья от заболевших, надо выбрать метод обучения нейронной сети.

3.2 Обучение с учителем

Пусть есть множество объектов, каждый из которых принадлежит одному из k пронумерованных (0,1,2,...k-1) классов. На этом обучающем множестве, предоставленным "учителем" (обычно человеком), система обучается. Затем, для неизвестных системе объектов (тестовом множестве), она проводит их классификацию, т.е. сообщает к какому классу принадлежит данный объект. В такой постановке - это задача распознавания образов после обучения с учителем.

Число признаков n называется размерностью пространства признаков.

Пусть признаки лежат в диапазоне [0...1]. Тогда любой объект представим точкой внутри единичного n - мерного куба в пространстве признаков.

Распознающую систему представим в виде чёрного ящика. У этого ящика есть n входов, на которые подаются значения (Рисунок – 8) признаков $x=\{x_1,x_2,...,x_n\}$ и k выходов $y=\{y_1,...,y_k\}$ (по числу классов). Значение выходов также будем считать вещественным числами из диапазона [0...1]. Система считается правильно обученной, если при подаче на входы признаков, соответствующих I - тому классу, значение I - того выхода равно 1, а всех остальных 0. На практике, такого результата добиться трудно и все выходы оказываются отличными от нуля. Тогда считается что номер выхода с максимальным значением и есть номер класса, а близость этого значения к единице говорит о "степени уверенности" системы.

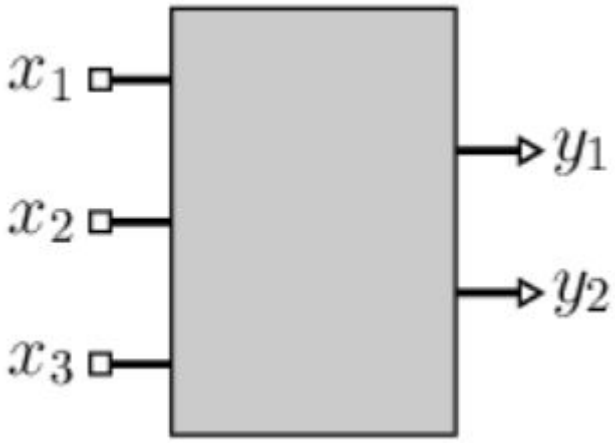

Рисунок – 8 – Черный ящик

Когда есть только два класса, ящик может иметь один выход. Если он равен 0, то это один класс, а если 1 - то другой. При нечётком распознавании вводятся пороги уверенности. Например, если значение выхода лежит в диапазоне y=[0 ... 0.3] - это первый класс, если y=[0.7 ... 1] - второй, а при y=(0.3 ... 0.7) система "отказывается принимать решение".

Ящик с одним выходом может также аппроксимировать

функцию $y=f(x_1,...,x_n)$, значения у которой непрерывны и обычно также нормируются на единицу, т.е. y=[0 ... 1]. В этом случае решается задача регрессии.

Таким образом принцип метода работы с учителем заключается в предоставлении нейронной сети наборов положительных и отрицательных изображений пшеницы на основе которых нейронная сеть выявляет отличительные особенности тех и других образов.

Для понимания разберем пример, при каких критериях нейрону подбираются параметры.

3.3 Нейрон

При обучении сети необходим критерий, в соответствии с которым подбираются параметры нейронов. Обычно для этого служит квадрат отклонения выходов сети от их целевых значений. Так, для двух классов и одного выхода, ошибкой Error сети считаем по формуле:

$$Error^2 = (1/N) \sum (y—y_c)^2$$

Где y - фактический выход, а y_c - его правильное значение, равное 0 для одного класса и 1 - для второго. Сумма ведётся по N обучающим примерам. Эту среднеквадратичную ошибку по всем обучающим объектам стараются сделать минимальной.

Рассмотрим 2 - мерное пространство признаков x_1,x_2 и два класса 0 и 1. На рисунке - 9 ниже объекты одного класса представлены синими кружочками, а объекты второго класса - красными крестиками. Справа от пространства признаков приведена сеть [2,1] из одного нейрона. За ней, на сине-красном квадратике, нарисована карта значений выхода нейрона при

тех или иных входах (x_1,x_2 пробегают значения от 0 до 1 с шагом 0.01). Если y=0 - то это синий цвет, если y=1 - то красный, а белый цвет соответствует значению y=0.5:

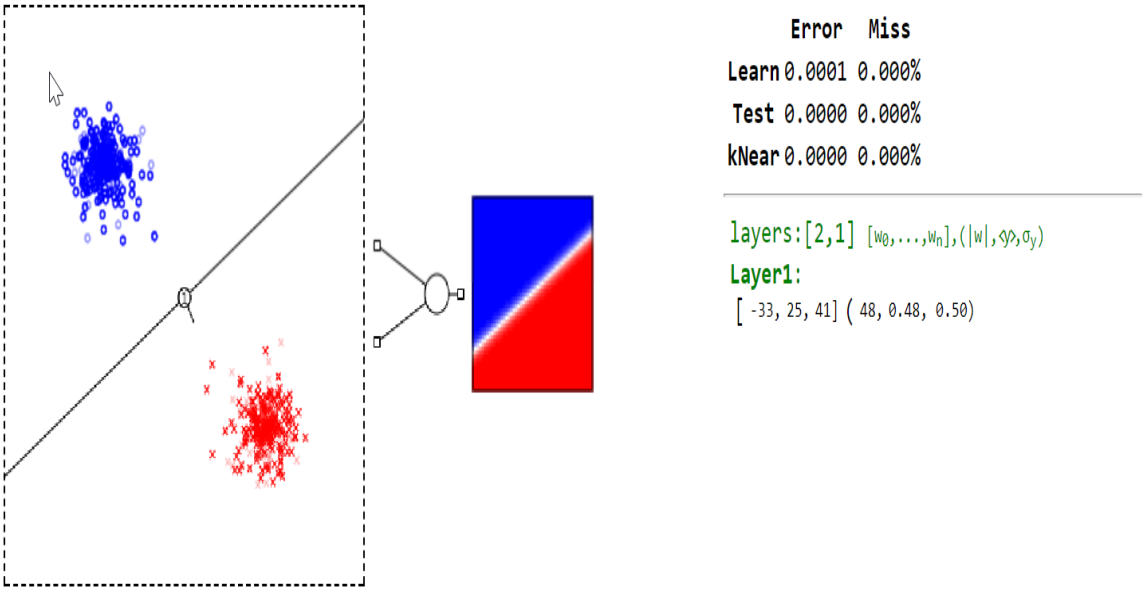

Рисунок 9 – Карта значений

Справа от рисунка - 9 под чертой, в квадратных скобках даны параметры нейрона: [w_0,w_1,w_2]. В круглых скобках приведена длина |w| вектора нормали ω и среднее значение выхода <y>. Ось x_1 пространства признаков направлена вправо, а ось x_2 - вниз. Поэтому вектор ω с положительными компонентами {w_1, w_2} направлен по диагонали вниз (он нарисован рядом с кружочком на прямой, содержащим номер нейрона 1).

Над чертой в таблице приведена среднеквадратичная ошибка Error такой сети. При этом строка Learn означает обучающую последовательность объектов, а Test - проверочную, которая не участвовала в обучении (тестовые объекты на графике пространства признаков изображены полупрозрачными). Колонка Miss содержит процент неправильно распознанных сетью объектов (отнесённых не к своему классу). Последняя строка kNear означает ошибку и процент ошибок в

методе 10 ближайших соседей (он будет описан позднее).

В этом примере разброс признаков объектов каждого класса невелик. Классы легко разделяются гиперплоскостью (линией в 2-мерии). Сеть стремиться минимизировать ошибку к целевым значениям 0 или 1 на выходе, поэтому модуль вектора |w|=48 принимает сравнительно большое значение. В результате, даже недалеко расположенные от плоскости объекты (в обычном евклидовом смысле) получают большое по модулю значение d. Соответственно y=S(d) = 0 или 1. Такой нейрон мы будем называть уверенным. Чем больше |w|, тем более уверен в себе нейрон. На его карте выхода тонкая белая линия (область неуверенности) отделяет насыщенный синий цвет (один класс) от насыщенного красного цвета (второй класс).

Несколько иная ситуация во втором примере (Рисунок – 10), где существует широкая область перекрытия объектов различных классов. Теперь нейрон не столь уверен в себе и длина вектора |w|=24 в 2 раза меньше:

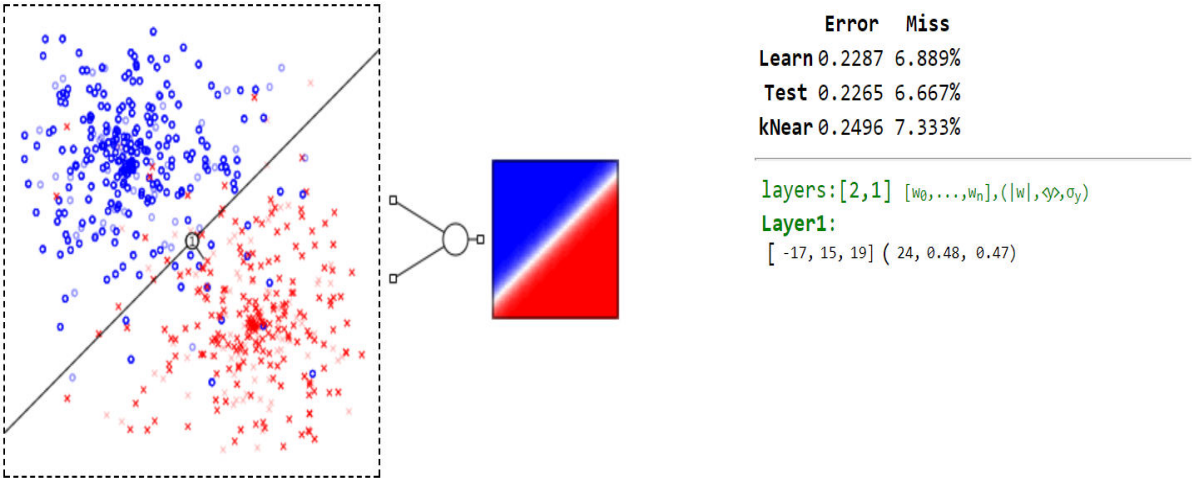

Рисунок 10 – Карта значений различный классов

Приведём функции (Рисунок – 11) деформации расстояния (сигмоид) при длине вектора нормали, равной 1,2,5,10,100:

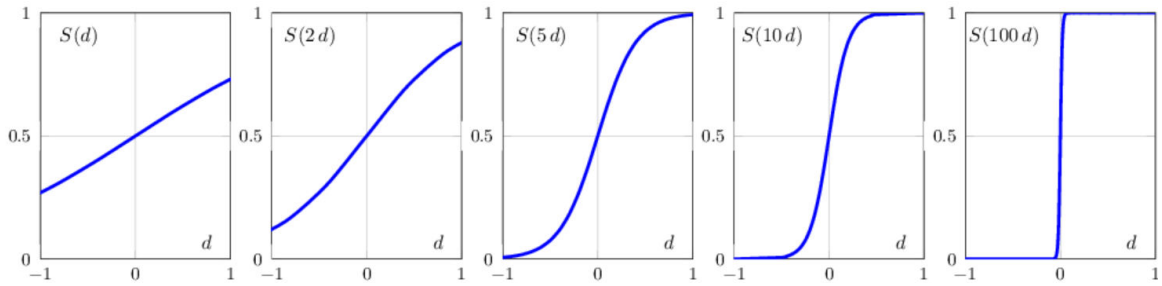

Рисунок 11 – Функции деформации состояний

Так как входы нейрона нормированы на единицу, максимальное расстояние от точки x с координатами $\{x_1,...,x_n\}$ в n-мерном кубе (его диагональ) равна корню из n. В 2-мерном пространстве признаков d_{max}=1.4. Если плоскость проходит через середину куба d_{max}~0.5.

3.4 Обучение и тестирование

Перед тем как запустить модель нейронной сети, настроим ее:

```
classifier.fit_generator(
    training_set,
    steps_per_epoch=5,
    epochs=5,
    validation_data=test_set,
    validation_steps=30,
    callbacks=callbacks_list,
    verbose=1
)
```

Метод fit обучает модель. Он принимает на вход обучающую выборку вместе с меткой - training_set, steps_per_epoch используется для генерации всего набора данных один раз, вызывая время генератора, steps_per_epoch, где epochs задано количество тренировок модели

по всему набору данных . Validation_steps - общее количество шагов (партии выборок) для проверки перед остановкой.

Запустим нейронную сеть для тренировки (Рисунок – 12).

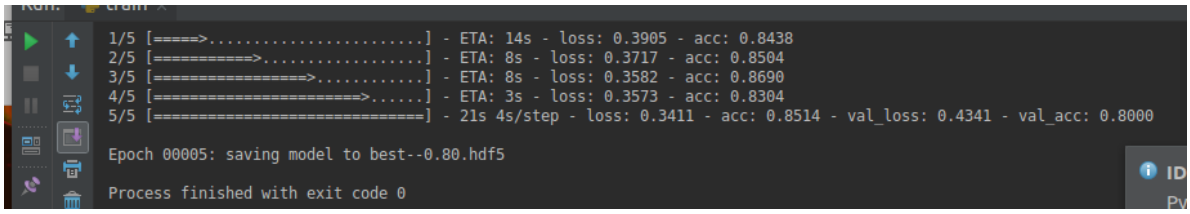

Рисунок 12 – Запуск нейронной сети

Здесь loss - функция ошибки, acc - точность на обучающей выборке, val_loss и val_acc - на тестовой выборке.

Видим, что точность на тестовой выборке получилась val_acc: 0.80, то есть, 80%.

Тренировочная модель закончила обучение нейронной сети. Теперь запустим модель (Рисунок – 13), которая должна отличить здоровые листья от больных.

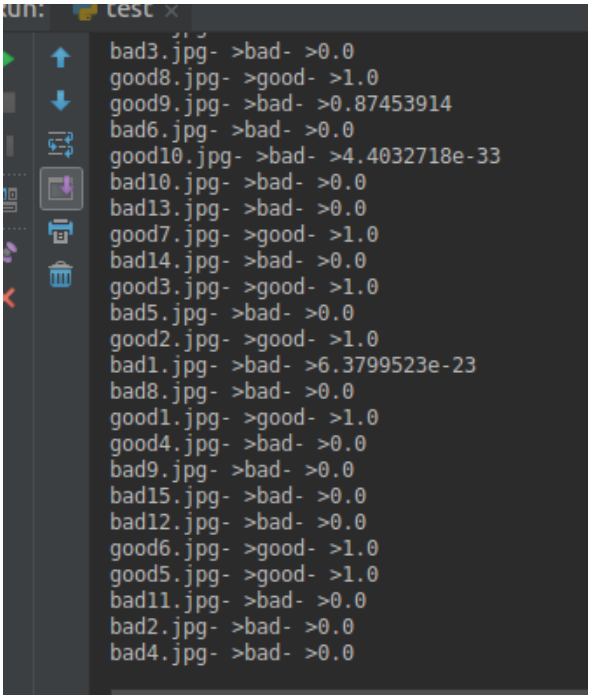

Рисунок 13 – Запуск модели

Посмотрев на результаты, можно заметить, что нейронная сеть работает корректно. Стоит отметить, что слово «good» выводит, когда лист не болеет, а «bad», когда он больной. Программа выводит наименование листа «good №» если не болеет и «bad №», где № это номер листа. Соответственно если результат «good № - >good» это хороший результат, а если «good № - >bad», то нейронная сеть ошиблась.

Посмотрев на вывод результатов, заметим, что нейронная сеть ошиблась три раза. Посмотрим на строчку «good 4 - >bad» и посмотрим на эту картинку (Рисунок – 14).

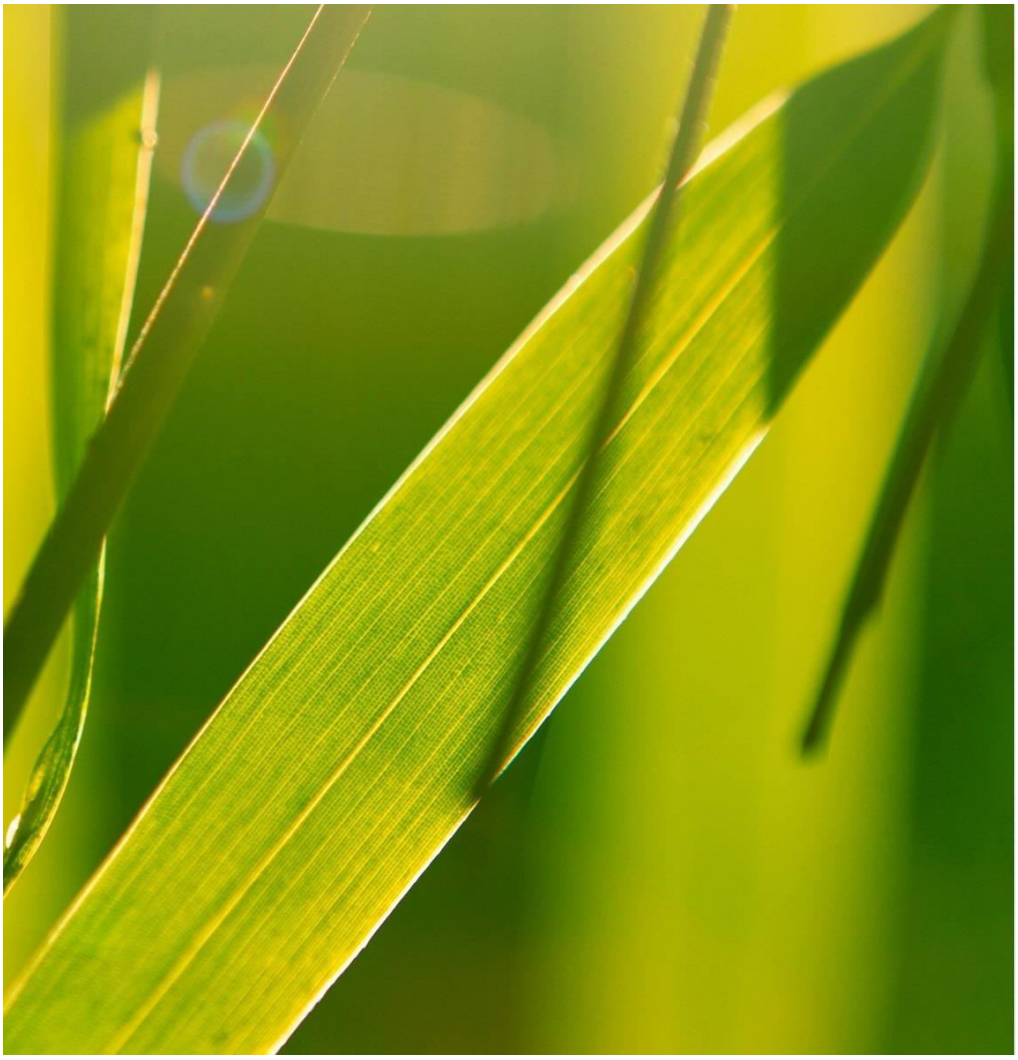

Рисунок 14 – Лист здоровой пшеницы

Рассмотрев лист и проанализировав, делаем вывод, что нейронная сеть ошиблась.

Причина по которой нейронная сеть ошиблась с конечном результатом это попадание луча солнца, а так же его отражение, которое дает желтые оттенки цвета. Соответственно оттенки желтого цвета, нейронная сеть приняла как за болезнь. Пример отражения показан на рисунке – 15.

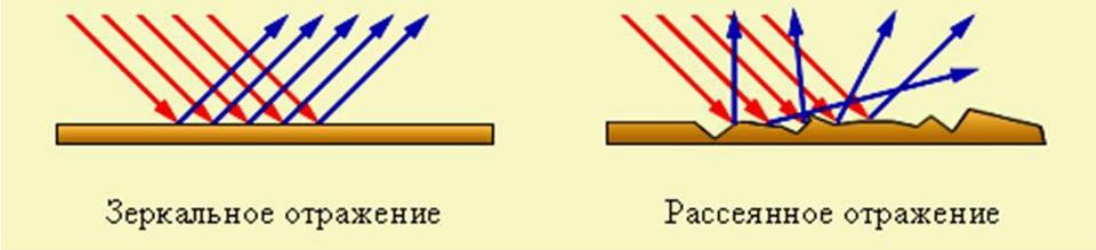

Рисунок 15 – Отражение световых лучей

Чтобы улучшить нейронную сеть - лучше распознавать, изменим входные данные нейронной сети. Для этого используем библиотеку «Opencv» и напишем алгоритм, который будет изменять градиент цвета, удаляя не нужные объекты на картинке.

Opencv - библиотека компьютерного зрения с открытым исходным кодом) - библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом.

Пример изменений рисунка библиотекой «Opencv» можно рассмотреть на Рисунке -16.

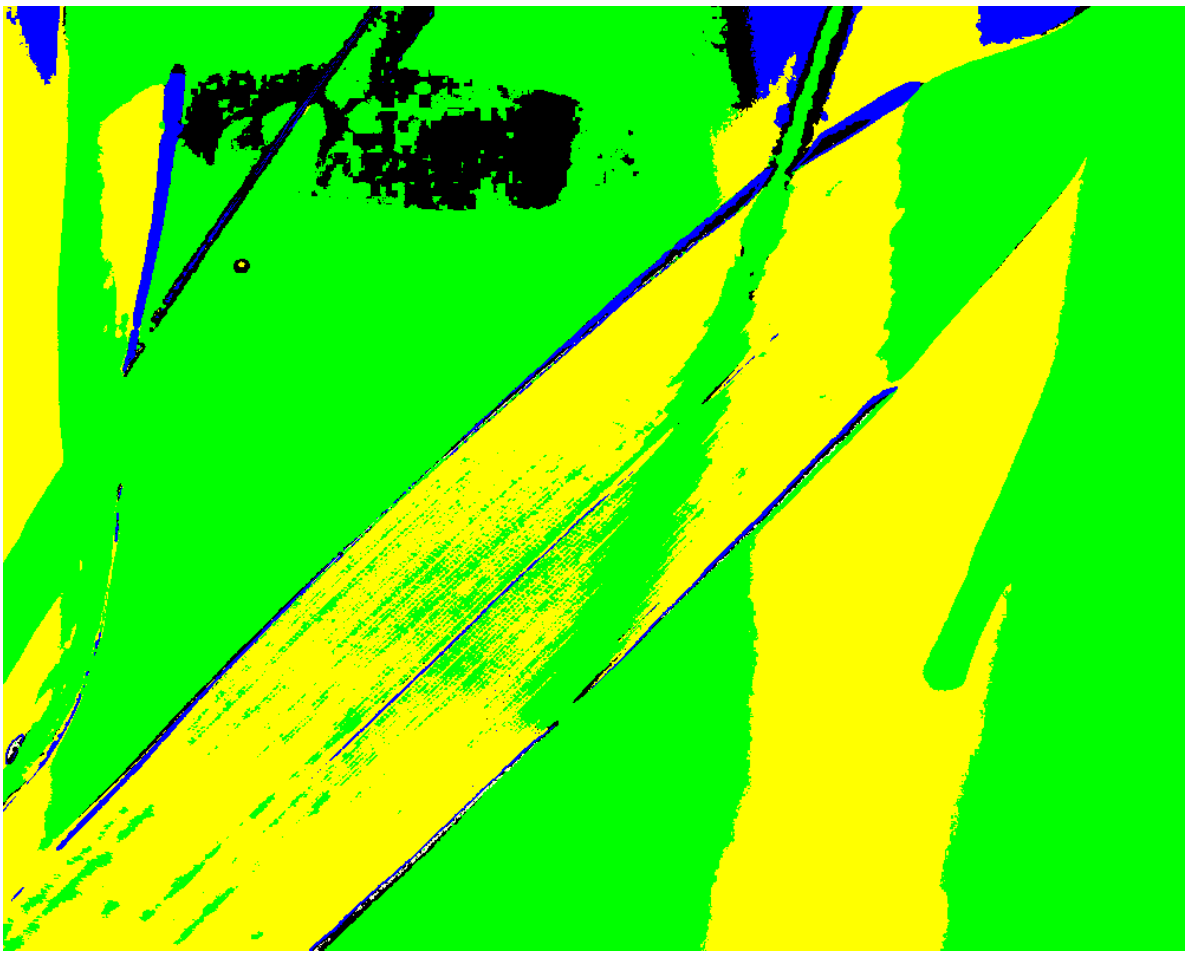

Рисунок 16 – После обработки изображения

Программа работает таким образом:

– Зеленый цвет, повышает свою контрастность до максимального значения.
– Ненужные объекты закрашиваются в черный цвет.
– Болезнь закрашивается синим цветом.
– Желтым закрашивается блики от солнца.

Запустим тренировочную модель (Рисунок – 17) с новыми входными данными, которые подготовили ранее, работая с библиотекой Opencv.

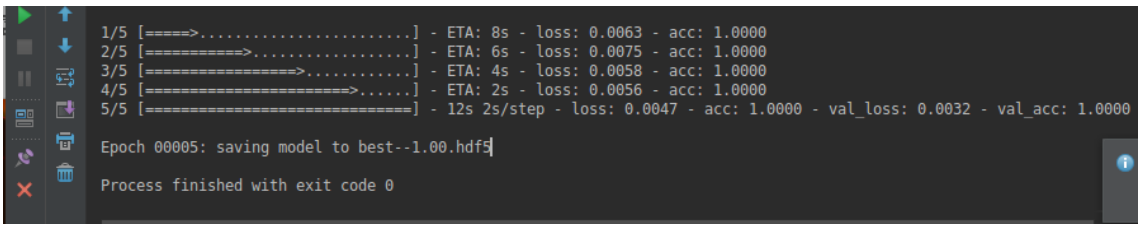

Рисунок 17 – Тренировочная модель

По тестовой выборке заметим отличный результат, целых 100%.

Тренировочная модель закончила обучение нейронной сети. Теперь запустим модель (Рисунок – 18), которая должна отличить здоровые листья от больных.

Рисунок 18 – Результаты модели

Проверим результаты, которые выдала модель. В тренировочной модели не выявлены ошибки. Делаем вывод, что модель работает лучше по сравнению с прошлой.

ЗАКЛЮЧЕНИЕ

За время выполнения курсовой работы были рассмотрены и изучены основные технологии и принципы разработки нейронной сети для классификации пшеницы. Также был рассмотрен инструмент для обработки изображения.

Для разработки программного продукта было изучено такое средство разработки нейронной сети как .TensorFlow и Keras. Были рассмотрены различные алгоритмы и структуры, используемые при работе с картинками. В целях оптимизации работы системы была изучена библиотека Opencv позволившая достичь высокой производительности разработанной системы.

На основе полученных знаний была изучена нейронная сеть, классифицирующая листья пшеницы . Данная программа была протестирована в среде Python с 25 выборками, 10 из которых являются примером здоровой пшеницы, а оставшиеся 15 – больной желтой ржавчиной.

В будущем планируется улучшить нейронную сеть, для повышения точности и увеличения скорости обработки изображений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Сверточная нейронная сеть URL: https://intellect.ml/svertochnaya-nejronnaya-set-convolutional-neural-network-cnn-6013 (дата обращения 01.05.2018)

2 Применение нейронных сетей для задач классификации URL: https://basegroup.ru/community/articles/classification (дата обращения 08.05.2018)

3 Что такое TansorFlow URL: https://www.computerworld.ru/articles/Chto-takoe-TensorFlow-i-kak-eto-ispolzuetsya (дата обращения 15.05.2018)

4 Keras: Deep Learning for humans URL: https://github.com/keras-team/keras

5 Deep Learning: Сравнение фреймворков для символьного глубокого обучения URL: https://habr.com/company/microsoft/blog/313318/ (дата обращения 27.05.2018)

6 Нейронные сети: Основы URL: http://synset.com/ai/ru/nn/NeuralNet_01_Intro.html (дата обращения 01.06.2018)