Реферат

Объект исследования

Объектом исследования являются нейронные сети, в частности, решение задачи классификации фотографий растений по наличию или отсутствию заболеваний.

Цель работы

Целью проекта является обеспечение экспертов по заболеваниям растений инструментами предварительной автоматизированной диагностики. В частности, имеет место исследование влияния параметров и фильтров, примененных к исходным фотографиям растений на итоговое качество нейронной сети.

Методы исследования

В ходе работы были применены следующие методы работы: анализ литературы, классификация, синтез, формализация, наблюдение, сравнение

Результаты

Построена нейронная сеть, по фотографии определяющая вероятность наличия болезни у сельскохозяйственной культуры. Благодаря препроцессингу исходных изображений удалось уменьшить ошибку на тестовой выборке в > 5 раз.

Содержание

Введение

Актуальность

На больших плантациях становится затруднительным регулярно отслеживать состояние здоровья сельскохозяйственных культур. В то же время, не найденные своевременно болезни растений приводят к снижению урожайности. Развитие технологий позволяет переложить эту задачу на дронов или придумать другие механические решения, которые при минимальном вмешательстве человека смогут локализовать болезнь на ранних стадиях и помочь вовремя предотвратить её.

Постановка задачи

Необходимо разработать программное обеспечение, способное по фотографии определять наличие или отсутствие болезни у представленного на фото представителя сельскохозяйственной культуры. Для тренировки будут предоставлены фотографии одной культуры, содержащие одно и то же заболевание, заведомо контрастные по диагнозу.

Методы исследования

В ходе работы были и будут применены следующие методы исследования: анализ литературы, классификация, синтез, формализация, наблюдение, сравнение

Цель

Целью проекта является обеспечение экспертов по заболеваниям растений инструментами предварительной автоматизированной диагностики

Задачи

1. Преобразования исходных фотографий (формат HEIF, разрешение 3024 x 4032, глубина цвета 24) во внутреннее растровое представление, обеспечивающее дальнейшую эффективную обработку.
2. Уменьшения размерности обрабатываемых данных и обучение первого слоя нейросети на основе применения эвристик в виде настраиваемых цветовых фильтров.
3. Выбора и обучения дополнительных слоев нейросети с использованием представленных в репозитории проекта наборов фотографий (не менее 200, при этом фотографии, не имеющие отношения к растениям заданного вида, не предоставляются), сопровождаемых оценкой вероятности определенного диагноза.

4. Проведение эксперимента по влиянию применения фильтров к исходным фотографиям на качество нейросети.
5. Обмена данными (экспорта, импорта) и агрегации обученных нейросетей в одном или нескольких открытых форматов (HDF5 и др.).
6. Получения выводов нейросети в форме вычисленной вероятности определенного диагноза для поступающих на оценку фотографий.

Теоретическая значимость и практическая ценность

Задача определения болезней растений встает перед многими учеными современности. Болезни растений влияют как на экологическую обстановку, так и на урожайность. В одном случае речь идет про дикую растительность, в другом - про искусственно созданные фермы. Если на маленьких огородах для быстрого отслеживания состояния здоровья растений достаточно осматривать его своими силами, то в случае больших плантаций и диких лесов своевременное отслеживание болезней становится сложной задачей, которую проще и быстрее делать программно, нежели ручным трудом. Применение нейросетей для этой и похожих задач становится популярной областью развития, например, сейчас у человечества есть решения таких задач как:

- нахождение участков леса, поврежденных жуками-короедами [1]
- диагностика болезней популярного африканского овоща маниока по фотографии [2]

И многих других. Эти решения активно публикуются в СМИ и могут широко использоваться в смежных сферах.

Задача нахождения факторов болезней растений специфична и зависит от многих факторов, например, сорта растения, выраженности болезни, ракурса, фона фотографии и так далее. Есть много научных трудов, оптимизирующих результат нейросети для конкретных видов растений или конкретных болезней. Однако, не все методы универсальны, поэтому для оптимизации локальной нейросети для отдельно взятой задачи, необходимо проводить самостоятельный эксперимент

Обзор и сравнительный анализ источников

Нейросети

Актуальность

Нейронные сети были придуманы с целью попытаться повторить процесс принятия решений человеком. В некоторой степени нейронные сети похожи на нервную систему живых существ - они состоят из нейронов и синапсов - связей между нейронами. Нейроны принимают информацию, обрабатывают её каким-то образом и подают на выход. Синапсы же передают информацию между

нейронами с некоторыми коэффициентами, которые показывают важной полученной информации в сравнении с остальными входами. При этом для удобства обычно считается, что нейроны и синапсы работают с числами в отрезке [0; 1]
Благодаря структуре, взятой из биологии, нейронные сети получили способность анализировать информацию подобно живым существам. Как ребенок, порезавшись о пару разных ножей, может далее классифицировать все ножи как острые объекты, обращаться с которыми нужно осторожно, так и нейросеть, получив несколько заданий и правильных ответов к ним, может вывести по аналогии ответы на новые задания такого же типа, которые видит впервые. В основном процесс обучения нейросети направлен на нахождение наиболее удачных коэффициентов в синапсах. Если первоначально коэффициенты ставятся случайные, то с каждой новой итерацией нейронная сеть переобучается, стараясь уменьшить ошибку на тренировочном сете.

Использование

Для нейросетей нашли множество применений, в основном это задачи классификации, т.е. распределение объектов по категориям. Примеры задач, с которыми сейчас успешно справляются нейронные сети:

- ИИ в автомобилях [3]

Многие компании стремятся к созданию автономных машин, способных передвигаться по дорогам без помощи водителя. Эта задача вызывает массу не только технических, но и этических вопросов. В том числе, автономный автомобиль должен уметь ориентироваться на дорогах, т.е. обладать компьютерным зрением - распознавать дорожные знаки и правила, различать ситуации, видеть другие автомобили и принимать решения на основе этих данных. Если полностью самостоятельный автомобиль пока остается на этапах экспериментов, то отдельные задачи, с которыми сталкиваются разработчики, внедряются в умные системы машин с водителями. Например, электромобили Тесла включают в себя систему распознавания дорожных знаков с целью определить максимально допустимую скорость и помочь водителю обрабатывать больше информации, чем он успевает.

- Управление кредитными рисками [4]

Прежде чем выдать кредит, банку необходимо по какому-то комплекту информации о клиенте (кредитная история, пол, возраст, заработная плата и другое) определить риск убытков. На основе сложных вычислений банк принимает решение - готов ли он принять на себя риски и выдать кредит этому человеку или стоит отказать. С этими вычислениями также может помочь обученная нейронная сеть, обладающая некоторой статистикой по прошлым выданным кредитам - на вход нейросети подается вся информация о человеке (описанные выше пол, возраст, кредитная история и прочее), а на выходе -

вернул человек кредит или нет. Обучившись на таких данных, построенная нейронная сеть может предсказывать вероятность того, что человек вернет кредит в срок, основываясь на точных измеримых данных.

- Создание картинок по описанию [5]

Одно из самых популярных среди широких масс применение нейросетей - создание новых объектов. Пользователь может описать в художественном стиле, что он хочет видеть на картинке, а нейросеть сгенерирует по этому описанию подходящее изображение. Сейчас эта сфера находится на стадии экспериментов и работает не идеально, однако в будущем сеть можно будет применять для генерации постеров, рекламных баннеров и так далее, упрощая механическую работу дизайнеров.

Математические модели

Как было сказано выше, нейронные сети представляют собой граф, состоящий из вершин-нейронов и ребер-синапсов. У каждого синапса есть коэффициент, обозначающий его “важность” в сравнении с другими выходами. Цель обучения нейронной сети - нахождение оптимальных коэффициентов синапсов, минимизирующих ошибку. По сути, все построение нейросети складывается в несколько шагов:

1. Построение графа. На этой стадии нужно выбрать конфигурацию нейронной сети, т.е. определить расположение нейронов и синапсов. Согласно топологии графов, по этому пункту выделяют несколько основных разновидностей нейронных сетей
    - Полносвязные - каждый нейрон соединен синапсами со всеми нейронами (в том числе с собой)
    - Слабосвязные - нейроны представляются узлами клетчатой решетки, а синапсы соединяют соседние клетки
    - Многослойные - несколько множеств нейронов, из каждого множества сигналы по синапсам передаются только в следующее множество. Такое множество называется слоем.

2. Инициализация. На этом шаге коэффициентам синапсов выставляются случайные значения, т.к. у нейросети нет никакой информации о задаче, которую ей предстоит решать

3. Переобучение. Входные данные тестового сета пропускаются через нейросеть. По всем данным считается ошибка, учитывая полученное значение Y_{pred} и имеющееся верное значение Y. Ошибку можно считать множеством разных способов, основные из них:
    - Среднеквадратичная - считается среднее арифметическое квадратов ошибок. Часто используется для задач линейной оптимизации, когда нужно вывести линейную формулу, наиболее

точно приближающую данные точки. Ошибка показывает классическую дисперсию, то есть, средний квадрат расстояния от точки до предсказанной функции
Ошибка считается как сумма $(Y_{pred} - Y)^2$ по всем точкам

- BCE - Binary Cross Entropy, она эффективно используется в задачах классификации, когда ответ - не случайный Y, а вероятность в отрезке [0; 1].
Ошибка считается как сумма $Y \cdot log\, Y_{pred} + (1 - Y) \cdot log(1 - Y_{pred})$ по всем точкам

Ошибки в задачах выбора решения/диагноза (например, как в нашем случае, определения наличия болезни у культуры) принято делить на “ложно-положительные” и “ложно-отрицательные”. Если некоторое решение нейросети влечет за собой необходимость действий, этот класс решений принимают за основной, а оставшийся - за вторичный. Соответственно, ошибкой первого рода является вероятность принять первый класс за второй, а ошибкой второго рода - вероятность принять второй класс за первый. Разумеется, хочется разделять 2 рода ошибок, поэтому в процессе подсчета оба рода считаются отдельно. В нашем случае первым классом будет наличие болезни у растения, т.к. в этом случае необходимо действие - лечение культуры

4. Подстройка весов. На этой стадии в зависимости от полученной ошибки, пересчитываются веса синапсов. Для этого тоже придумано несколько различных алгоритмов, эффективных в разных задачах:
    - Градиентный спуск. В общем случае, градиентный спуск используется для нахождения минимума функции. Алгоритм каждый раз пытается идти в направлении наискорейшего спуска. Выбирается несколько случайных векторов одинаковой длины, прибавляются к текущему состоянию, а после для полученного вектора вычисляется ответ. С каким вектором ответ получился наименьшим, тот и прибавляем в итоге. Несложно представить, что задача минимизации ошибки тоже решается градиентным спуском. [6]
    - Метод отжига. Основная идея в том, чтобы спускаться в минимум, но на каждом ходу с некоторой вероятностью к текущему вектору добавлять случайный. Такая хитрость помогает не застрять в локальном минимуме, если мы хотим попасть в глобальный. [7]
    - Генетический алгоритм. Он, как и нейронные сети, взят из биологии. Его цель - вырастить наиболее успешного индивида (коэффициенты синапсов). На каждом шаге он имитирует скрещивание текущих кандидатов (например, среднее значение из

2 векторов коэффициентов), вымирание наихудших и позволяет некоторые мутации (аналогично прыжкам в сторону в методе отжига) [8]

Сверточные сети

Для обработки нейронной сетью многомерных объектов, используют сверточные сети [9]. В частности это касается изображений, которые представляют из себя трехмерный объект. 2 измерения - это положение в пространстве (таблица пикселей), третье - цвет, содержащий 3 числа R, G, B (если мы работаем в этой цветовой модели). Сверточная нейронная сеть содержит несколько чередующихся сверточных и прореживающих слоев, каждая вершина которых представляет собой трехмерный тензор.
Ребрами между сверточными слоями являются свертки - двумерные матрицы с весами. При переходе из тензора A по ребру с матрицей C, для каждого индекса тензора A к итогу прибавляется подматрица A того же размера, что и C, начинающаяся в фиксированном индексе. За счет того, что размер матрицы C зачастую выбирается больше единицы, итоговый тензор получается меньшего размера.
Для перехода в прореживающий слой, тензор делится на непересекающиеся подматрицы некоторого размера (обычно 2x2), в каждой подматрице выбирается максимальный элемент и он идет в итог. За счет такого прореживания размер тензора уменьшается в несколько раз, выделяя только главные выявленные характеристики.

Препроцессинг

Актуальность

Основная доля успеха нейронных систем в классификации изображений лежит в нахождении повторяющихся шаблонов. Например, если обучать нейронную сеть отличать человеческое лицо от кошачьего, она будет пытаться фильтрами найти наиболее весомые структуры, которые есть на человеческих фотографиях и нет на кошачьих. Если говорить грубо и не углубляясь в детали, нейронная сеть будет пытаться найти человеческий нос, круглые зрачки и прочие характеристики, указывающие на принадлежность к человеку.
Разумеется, если на вход такой нейросети подать грязные фотографии с лишним шумом или с лишними объектами, обучить её будет сложнее, по той причине, что кроме нахождения лица человека/кошки, ей нужно будет понять, а что вообще такое лицо и почему стул на переднем фоне картинки - это не то же самое. [10]
Аналогичное происходит и при классификации заболеваний растений. Ставя перед собой цель автоматизировать процесс своевременного отслеживания болезней, нужно принимать во внимание, что в производственных буднях сделать идеальные фотографии растения на постоянном фоне без

отвлекающих предметов представляется невозможным. Из этого положения есть 2 выхода:

- обучить нейронную сеть “находить” растение на фотографии ([11], [12]). Для этого требуется много ресурсов и обучение отдельной, более сложной по структуре нейросети, которая будет находить повторяющиеся паттерны растения и, например, указывать его локацию, чтобы после можно было удалить лишние куски или другими способами избавиться от не консистентного фона
- провести препроцессинг ([13]), в ходе которого обработать исходные данные, пытаясь выделить интересующие детали. Это гораздо менее энергозатратный способ написать устойчивую к фону нейросеть, к тому же, если в какой-то момент появится запрос с новым окружением, нам не нужно будет переобучать нейросеть, что, возможно, понадобится сделать в предыдущем варианте

Популярные методы

Человечество умеет эффективно решать задачу нахождения болезней у растений с некоторым препроцессингом, выделяющим наиболее важные детали для соответствующей задачи. Проблема заключается в том, что пока не существует решения, идеально подходящего под все культуры. Поэтому при решении новой задачи приходится искать важные свойства самостоятельно. В некоторых ранее построенных нейронных сетях, решающих ту же задачу, но для других культур, наиболее эффективными идеями для обработки исходных изображений без использования дополнительных нейросетей оказываются:

- распознавание цветов с помощью RGB [11]
- распознавание цветов с помощью HSV и HSL форматов [13]
- изменение контрастности изображения

Представление цвета

RGB

Наибольшую популярность кодирования цвета в изображениях приобрела модель RGB, которая представляет из себя 3 целых числа от 0 до 255, обозначающих интенсивность красного, зеленого и синего цвета, соответственно. Это удобная система для кодирования цветов компьютером, за счет большого количества цветов, работы с целыми числами и сравнительно небольшой памяти. [14]

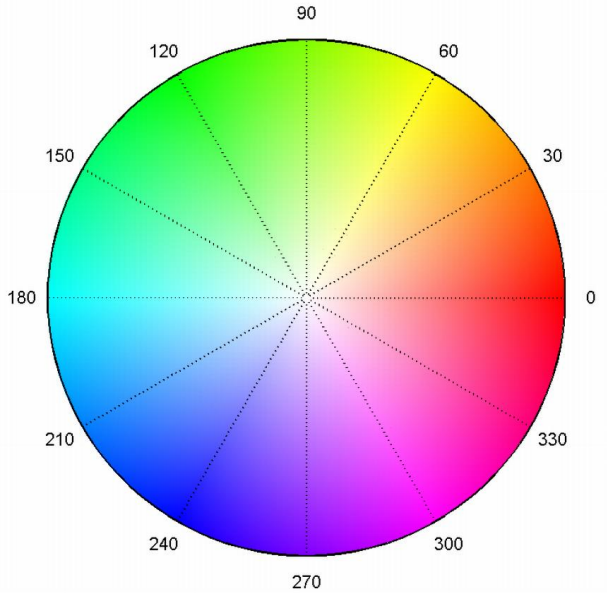
HSV, HSL

С точки зрения глаза человека какой-то особой логики в RGB нет. Нейронные сети же стремятся повторить работу человеческого мозга, поэтому в более интуитивной для человека системе кодирования цвета может быть смысл. HSV, HSL и HSI довольно похожие модели для представления цвета [15]. Все они ориентируются на одну и ту же шкалу оттенков (рис.1). Шкала, по сути,

циклическая и устроена так, что близкие для глаза человека цвета находятся рядом (1 и 359 тоже находятся рядом из-за цикличности). Собственно, во всех 3 системах H - это Hue (оттенок) и принимает вещественное значение от 0 до 360

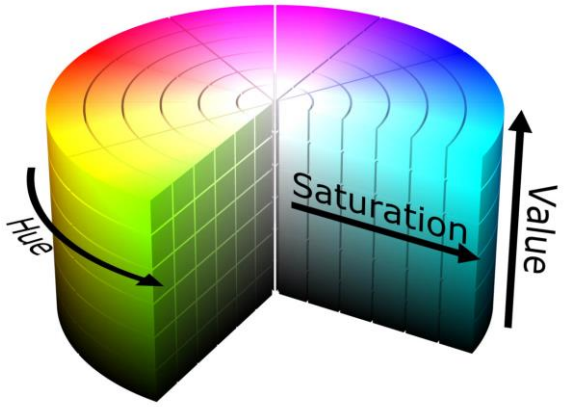
(рис.1)

Второй параметр S = Saturation тоже совпадает у всех 3 систем и означает насыщенность и чистоту цвета. На рисунке (рис.2), по сути, H - это угол наклона вектора, а S - его длина. S принимает вещественное значение в [0; 1]

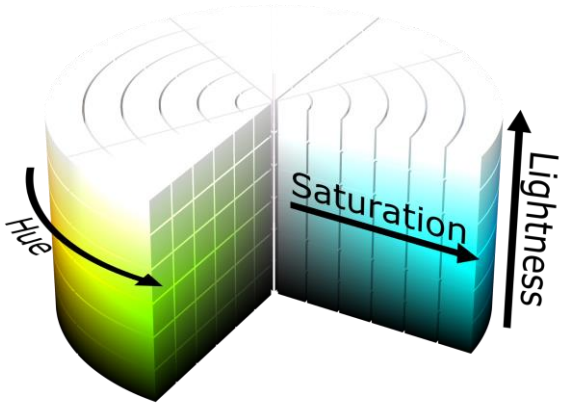


(рис.2)

Последнее же число различается. V - это Value, яркость цвета, а L - Lightness, светлота цвета. На рисунках (рис.3, рис.4) можно заметить разницу между ними. По сути, последний параметр - это высота цилиндра, в котором мы берем точку, третья координата вектора. Эта величина тоже принимает вещественное значение в [0; 1]



(рис.3)



(рис.4)

Описание выбранных методов, алгоритмов, моделей, методик

Обучение нейронной сети

Для классификации изображений была выбрана сверточная нейронная сеть. Обучение нейросети при прочих равных оказалось наиболее эффективным с использованием Python 3 и библиотеки PyTorch в качестве удобного инструмента для машинного обучения.

Препроцессинг

Было решено провести эксперимент со следующими фильтрами предобработки исходных изображений перед подачей на обучение нейронной сети:

- повышение контрастности изображения в модели RGB на основе всех пикселей или только ближней области каждого пикселя
- регулирование цвета на основе зеленой составляющей RGB модели
- перевод в HSV, HSL представление
- регулирование цвета на основе близости к зеленому цвету в HSV, HSL

Эксперименты по препроцессингу было решено проводить так же в Python 3 с помощью PyTorch на упрощенной нейронной сети.

Описание вычислительного эксперимента

Эксперимент проводился и документировался с помощью ipynb-документа в среде Anaconda.

Нейросеть

С помощью библиотеки PyTorch была написана функция, принимающая набор изображений и обучающая на них сверточную нейронную сеть. Обучение нейронной сети происходило на примерно 350 изображениях (включая и тренировочные и тестировочные данные) и длилось 30 эпох (полных прогонов по всем изображениям). После каждой эпохи подсчитывалась функция ошибки на тренировочных и тестовых данных и записывалась в отдельные массивы для последующего анализа данных.

Постановка экспериментов

Для каждого эксперимента выписывалась функция обработки одного изображения. С помощью этой функции обрабатывались все данные, после чего обработанные изображения подавались на вход функции, обучающей нейросеть и получали функцию ошибки на каждой эпохе от 1 до 30.

Было решено провести эксперименты со следующей обработкой фотографий:

1. Модель RGB. Повышение контрастности каждого пикселя на основе среднего значения всего изображения, а именно:
    - Подсчет среднего значения R, G, B по всему изображению
    - Для каждого пикселя к r, g и b прибавляется разница между им и средним значением, умноженная на некоторый коэффициент
    - Проверяются коэффициенты 0.5 и 1
2. Модель RGB. Изменение зеленой составляющей каждого пикселя на основе всего изображения
    - Подсчет среднего значения G по всему изображению
    - К g прибавляется (g - G) * K
    - Проверяются коэффициенты K в множестве {0.5, 1}
3. Модель RGB. Изменение зеленой составляющей каждого пикселя на основе величины этой составляющей
    - Если G пикселя > max(R, B), то не изменяем пиксель, иначе заменяем G на 0
4. Перевод в HSV формат
    - RGB каждого пикселя переводится в HSV модель
    - Каждое значение HSV модели нормируется в целые значения [0; 255] и вставляется вместо RGB-пикселя
5. Перевод в HSL формат
    - RGB каждого пикселя переводится в HSL модель
    - Каждое значение HSL модели нормируется в целые значения [0; 255] и вставляется вместо RGB-пикселя
6. Регулирование яркости цвета на основе близости к зеленому цвету в HSV и HSL модели
    - Каждый пиксель переводится в HSV формат

- Если |H - 120| > D и яркость (V) > некоторого значения L, то все RGB-значения пикселя умножаются на K < 1
- Так как вариантов очень много, планируется перебирать отдельно L, отдельно K, отдельно D и при фиксированных остальных параметрах смотреть на изменения значения функции ошибок.

Результаты

Неуспешные

Эксперимент 1. С коэффициентом 0.5 получил результат, похожий на исходный (без препроцессинга изображений), с коэффициентом 1 в 2 раза большая ошибка, нежели без фильтров. Для более подробных значений см. приложение 1

Эксперимент 2. Запускался с коэффициентами 0.5 и 1, на обоих сильно переобучился. Результаты тренировочной выборки примерно такие же, как и без фильтров, результаты тестовой выборки примерно в 3.5 раза хуже. Для более подробных значений см. приложение 2

Эксперименты 4, 5. Перевод в HSV и HSL формат сам по себе ничего не дал. Ошибка на переводе в HSV и HSL модели абсолютно одинакова, видимо, последний параметр модели (V или L) не влияет на принятие решения. Нейронная сеть переобучилась. Ошибка на тренировочных данных в 4 раза меньше, чем у изображений без фильтров, а ошибка на тестовой выборке в 10 раз больше, чем на тесте изображений без фильтров. Для числовых значений см. приложение 3

Успешные

Эксперимент 3. К удивлению, простая обработка, которая обнуляет G составляющую пикселя RGB, если G <= max(R, B), показала отличные результаты. Ошибка на тренировочной выборке оказалась на 28% меньше, ошибка на тестовой выборке уменьшилась в 3.2 раза. См. приложение 4 для численных значений.

Эксперимент 6. Оптимальные значения были достигнуты при игнорировании |H - 120| <= 30 и V >= 1.5 и делении каждой составляющей остальных пикселей на коэффициент 1.5. При таких параметрах ошибка на тренировочных данных уменьшилась на 38%, а ошибка на тестовых данных уменьшилась в 5.3 раза, значительно обогнав результаты эксперимента 3. Для численных результатов каждой стадии эксперимента см. приложения 5, 6, 7

Заключение

В ходе проекта была достигнута основная цель - обучение нейронной сети, предсказывающей вероятность наличия заболевания сельскохозяйственной культуры по фотографии. Так же была написана серверная и клиентская части, позволяющие легко обращаться к нейросети - загружать фотографию и получать результат.
Экспериментально подтвердилось влияние препроцессинга исходных данных на ошибку нейронной сети, для конкретной задачи найден препроцессинг изображений, уменьшающих ошибку в 5 раз.

Список использованных источников

[1] В ЛЭТИ научили нейросеть выявлять участки леса, пораженные жуками-вредителями. URL: https://naked-science.ru/article/column/v-leti-nauchili-nejroset-vyyavlyat-uchastki-lesa
[2] Нейросеть распознает болезнь растения по фотографии. URL: https://zen.yandex.ru/media/code/neiroset-raspoznaet-bolezn-rasteniia-po-fotografii-5d6ced406d29c100ad3b5cf6
[3] Искусственный интеллект вашего автомобиля. URL: https://gb.ru/posts/ai_for_autos
[4] Скоринг с применением нейронных сетей. URL: https://vc.ru/finance/341642-skoring-c-primeneniem-neyronnyh-setey-ml-ii
[5] ruDALL-E: генерируем изображения по текстовому описанию, или Самый большой вычислительный проект в России. URL: https://habr.com/ru/company/sberbank/blog/586926/
[6] Градиентный спуск по косточкам. URL: https://habr.com/ru/post/467185/
[7] Введение в оптимизацию. Имитация отжига. URL: https://habr.com/ru/post/209610/
[8] Генетический алгоритм. Просто о сложном. URL: https://habr.com/ru/post/128704/
[9] Сверточные нейронные сети. URL: https://neerc.ifmo.ru/wiki/index.php?title=Сверточные_нейронные_сети
[10] Усы, лапы и хвост: как нейронная сеть распознает котиков и другие объекты. URL: https://habr.com/ru/company/binarydistrict/blog/354524/
[11] C. S. Hlaing and S. M. M. Zaw, “Tomato plant diseases classification using statistical texture feature and color feature,” in *Proc. IEEE/ACIS 17th International Conference on Computer and Information Science*, Singapore, 2018. URL: https://ieeexplore.ieee.org/abstract/document/8466483
[12] J. S. H. Al-bayati and B. B. Üstündağ, “Evolutionary feature optimization for plant leaf disease detection by deep neural networks,” *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 12–23, 2020. URL:

https://www.researchgate.net/profile/Burak-Uestuendag/publication/338784417_Evolutionary_Feature_Optimization_for_Plant_Leaf_Disease_Detection_by_Deep_Neural_Networks/links/5ec3d1ab92851c11a8745ecb/Evolutionary-Feature-Optimization-for-Plant-Leaf-Disease-Detection-by-Deep-Neural-Networks.pdf
[13] R. Pydipati, T. F. Burks, and W. S. Lee, “Identification of citrus disease using color texture features and discriminant analysis,” *Computers and Electronics in Agriculture*, vol. 52, no. 1–2, pp. 49–59, 2006. URL: https://www.sciencedirect.com/science/article/abs/pii/S0168169906000287
[14] RGB color model. URL: https://en.wikipedia.org/wiki/RGB_color_model
[15] HSL and HSV. URL: https://en.wikipedia.org/wiki/HSL_and_HSV

Приложения

Приложение 1

Сравнение результатов эксперимента 1 с результатами без фильтров

		epochs = 10	**epochs = 20**	**epochs = 30**
без фильтров	**train**	0.0076355943	0.0008941761	0.0004296261
	test	0.0247114046	0.0118135703	0.0082125369
coef = 0.5	**train**	0.0089575944	0.0009746532	0.0004031936
	test	0.0295293262	0.0124976370	0.0082233751
coef = 1	**train**	0.0109894069	0.0009589178	0.0003019577
	test	0.0381878632	0.0213689233	0.0167181262

Приложение 2

Сравнение результатов эксперимента 2 с результатами без фильтров

		epochs = 10	**epochs = 20**	**epochs = 30**
без фильтров	**train**	0.0076355943	0.0008941761	0.0004296261
	test	0.0247114046	0.0118135703	0.0082125369
coef = 0.5	**train**	0.0135962814	0.0015037021	0.0005679324

	test	0.0365074307	0.0267675443	0.0283282639
coef = 1	**train**	0.0147983264	0.0018894396	0.0004961923
	test	0.0366500103	0.0277036881	0.0271459699

Приложение 3

Сравнение результатов экспериментов 4, 5 (перевод в HSV, HSL) с результатами без фильтров

		epochs = 10	**epochs = 20**	**epochs = 30**
без фильтров	**train**	0.0076355943	0.0008941761	0.0004296261
	test	0.0247114046	0.0118135703	0.0082125369
HSV, HSL (эксп. 4, 5)	**train**	0.0078065769	0.0008782308	0.0001560205
	test	0.0941572281	0.0876864347	0.0877809693

Приложение 4

Сравнение результатов эксперимента 3 с результатами без фильтров

		epochs = 10	**epochs = 20**	**epochs = 30**
без фильтров	**train**	0.0076355943	0.0008941761	0.0004296261
	test	0.0247114046	0.0118135703	0.0082125369
эксп. 3	**train**	0.0089035915	0.001178426	0.0003134392
	test	0.0190888093	0.0048775009	0.0025646883

Приложение 5

Результаты перебора значения light в эксперименте 6 с фиксированными значениями coef = 2, dist = 80

		epochs = 10	epochs = 20	epochs = 30
light = 0	**train**	0.0104871416	0.0013262631	0.0002525698
	test	0.0380776602	0.0214672322	0.0170587953
light = 0.1	**train**	0.0121696042	0.0020342957	0.0003880123
	test	0.0580647945	0.0465202212	0.0440202744
light = 0.2	**train**	0.0099735862	0.0009990728	0.0004993440
	test	0.0256409628	0.0146111548	0.0132718218
light = 0.25	**train**	0.0095980351	0.0012933044	0.0003596272
	test	0.0173870881	0.0069401174	0.0056666192
light = 0.3	**train**	0.0095156783	0.0013505236	0.0004309756
	test	0.0183256968	0.0071170715	0.0043310716
light = 0.35	**train**	0.008518452	0.0011780944	0.0005540729
	test	0.0208574571	0.0084203145	0.005571338
light = 0.4	**train**	0.0080554777	0.0008165952	0.0004207914
	test	0.0277716645	0.0143988477	0.0118436026
light = 0.45	**train**	0.0101334774	0.0009211206	0.0005708266
	test	0.0281808159	0.0241189921	0.0237464506

Приложение 6

Результаты перебора значения coef в эксперименте 6 с фиксированными значениями light = 0.3, dist = 80

		epochs = 10	epochs = 20	epochs = 30

coef = 1.5	**train**	0.007440842	0.0010205758	0.0003814549
	test	0.0171875253	0.0065295724	0.0039513752
coef = 2	**train**	0.0095156783	0.0013505236	0.0004309756
	test	0.0183256968	0.0071170715	0.0043310716
coef = 2.5	**train**	0.0108879641	0.0015985534	0.0004929188
	test	0.0235796344	0.0115583322	0.0076568966

Приложение 7

Результаты перебора значения dist в эксперименте 6 с фиксированными значениями light = 0.3, coef = 1.5

		epochs = 10	**epochs = 20**	**epochs = 30**
dist = 20	**train**	0.0071339350	0.0009962593	0.0003820759
	test	0.0181440093	0.009356913	0.0058395636
dist = 30	**train**	0.0057860315	0.0007692942	0.0002640577
	test	0.0065623415	0.0028630567	0.0015428441
dist = 40	**train**	0.0070022399	0.0006308352	0.0003172219
	test	0.0116633598	0.007156636	0.0048605525
dist = 50	**train**	0.0059166337	0.0009934988	0.0004130901
	test	0.0130029060	0.0058494499	0.0049840415
dist = 60	**train**	0.0058259896	0.0009357335	0.0003640965
	test	0.010345887	0.0036409715	0.0023683305
dist = 70	**train**	0.0069321699	0.0010047112	0.0003606536

	test	0.0105480685	0.003889723	0.0025882972
dist = 80	**train**	0.007440842	0.0010205758	0.0003814549
	test	0.0171875253	0.0065295724	0.0039513752
dist = 90	**train**	0.0070281179	0.0010651766	0.0003811898
	test	0.0147707057	0.0047427529	0.0027968777