МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра программных систем

Дисциплина нейронные сети

**ОТЧЕТ**

по лабораторной работе №1

«Нейронные сети глубокого обучения»

Вариант 4

«Подбери себе друга»

Обучающийся группы 6131-020402D В.Д. Гижевская

Обучающийся группы 6131-020402D Д.А. Кремнёв

Руководитель А.Н. Жданова

Самара 2022

СОДЕРЖАНИЕ

[1 Описание и анализ предметной области 3](#_Toc117189859)

[1.1 Нейронная сеть 3](#_Toc117189860)

[1.2 Постановка задачи 5](#_Toc117189861)

[2 Проектирование системы 6](#_Toc117189862)

[2.1 Выбор и обоснование архитектуры нейронной сети 6](#_Toc117189863)

[2.2 Обучение нейронной сети 8](#_Toc117189864)

[2.3 Функция активации 11](#_Toc117189865)

[2.4 EfficientNet 13](#_Toc117189866)

[3 Описание вычислительных экспериментов 16](#_Toc117189867)

[3.1 Описание параметров нейронной сети 16](#_Toc117189868)

[3.2 Результаты экспериментов 18](#_Toc117189869)

[3.2.1 Основной эксперимент 18](#_Toc117189870)

[3.2.2 Эксперимент с уменьшенным количеством эпох 18](#_Toc117189871)

[3.2.3 Эксперимент с уменьшенным количеством слоёв 19](#_Toc117189872)

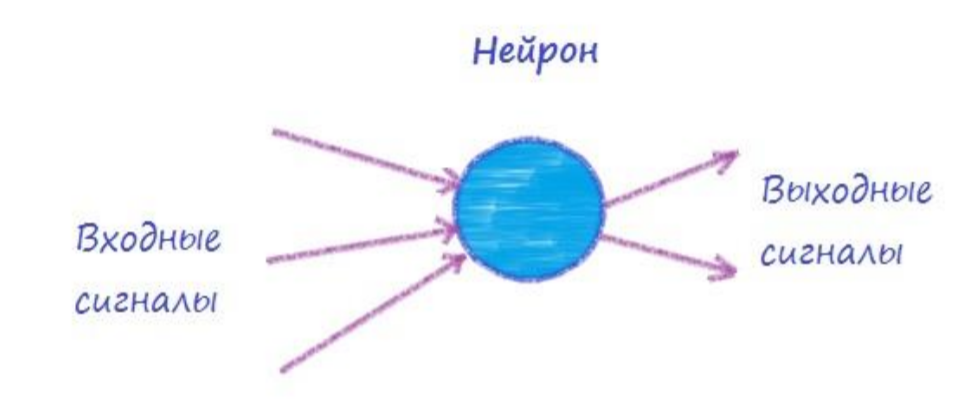
[4 Результат работы 20](#_Toc117189873)

[ЗАКЛЮЧЕНИЕ 21](#_Toc117189874)

1. Описание и анализ предметной области
   1. Нейронная сеть

Изучение и использование искусственных нейронных сетей, в принципе, началось уже достаточно давно – в начале 20 века, но по-настоящему широкую известность они получили несколько позже. Связано это, в первую очередь, с тем, что стали появляться продвинутые (для того времени) вычислительные устройства, мощности которых были достаточно велики для работы с искусственными нейронными сетями. По сути, на данный момент можно легко смоделировать нейронную сеть средней сложности на любом персональном компьютере.

Нейронная сеть представляет из себя совокупность нейронов, соединённых друг с другом определённым образом. Рассмотрим один нейрон, его модель представлена на рисунке 1.

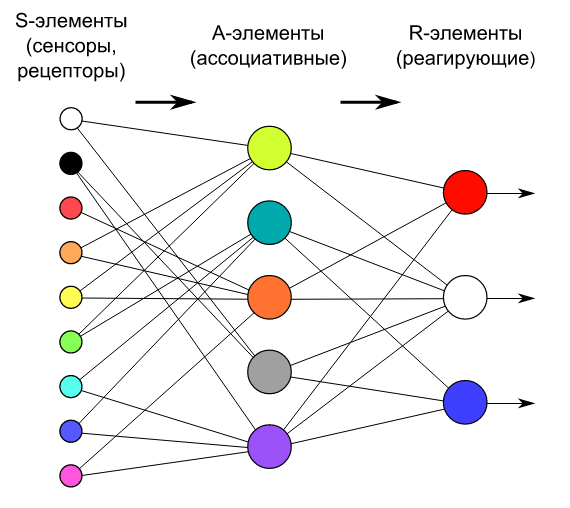
  
Рисунок 1 – Модель нейрона

Нейрон представляет из себя элемент, который вычисляет выходной сигнал (по определённому правилу) из совокупности входных сигналов. То есть основная последовательность действий одного нейрона такая:

* приём сигналов от предыдущих элементов сети;
* комбинирование входных сигналов;
* вычисление выходного сигнала;
* передача выходного сигнала следующим элементам нейронной сети.

Между собой нейроны могут быть соединены абсолютно по-разному, это определяется структурой конкретной сети. Но суть работы нейронной сети остаётся всегда одной и той же. По совокупности поступающих на вход сети сигналов на выходе формируется выходной сигнал (или несколько выходных сигналов). То есть нейронную сеть упрощённо можно представить в виде чёрного ящика, у которого есть входы и выходы. А внутри этого ящика сидит огромное количество нейронов.

Персептрон (Perceptron) – простейший вид нейронных сетей. В основе лежит математическая модель восприятия информации мозгом, состоящая из сенсоров, ассоциативных и реагирующих элементов. Модель персептрона представлена на рисунке 2.

  
Рисунок 2 – Модель персептрона

Разные исследователи по-разному его определяют. В самом общем своём виде (как его описывал Розенблатт) он представляет систему из элементов трёх разных типов: сенсоров, ассоциативных элементов и реагирующих элементов.

Рассмотрим принцип работы персептрона. Первыми в работу включаются S-элементы. Они могут находиться либо в состоянии покоя (сигнал равен 0), либо в состоянии возбуждения (сигнал равен 1). Далее сигналы от S-элементов передаются A-элементам по так называемым S-A связям. Эти связи могут иметь веса, равные только -1, 0 или 1. Затем сигналы от сенсорных элементов, прошедших по S-A связям, попадают в A-элементы, которые ещё называют ассоциативными элементами. Стоит заметить, что одному A-элементу может соответствовать несколько S-элементов. Если сигналы, поступившие на A-элемент, в совокупности превышают некоторый его порог ​θ​, то этот A-элемент возбуждается и выдаёт сигнал, равный 1. В противном случае (сигнал от S-элементов не превысил порога A-элемента), генерируется нулевой сигнал.

Далее сигналы, которые произвели возбуждённые A-элементы, направляются к сумматору (R-элемент). Однако, чтобы добраться до R-элемента, они проходят по A-R связям, у которых тоже есть веса. Здесь они уже могут принимать любые значения (в отличие от S-A связей).

Далее. R-элемент складывает друг с другом взвешенные сигналы от A-элементов и, если превышен определённый порог, генерирует выходной сигнал, равный 1. Это означает, что в общем потоке информации от глаз мы распознали лицо человека. Если порог не превышен, то выход персептрона равен -1. То есть мы не выделили лицо из общего потока информации.

* 1. Постановка задачи

Пусть X – множество описаний объектов, Y – множество номеров (или наименований) классов. Существует неизвестная целевая зависимость – отображение , значения которой известны только на объектах конечной обучающей выборки . Требуется построить алгоритм , способный классифицировать произвольный объект .

Во время лабораторной работы необходимо разработать программу, с помощью которой можно распознать изображение и определить его к одному из заданных классов. Количество классов – 15.

1. Проектирование системы
   1. Выбор и обоснование архитектуры нейронной сети

Наилучшие результаты в области классификации изображений показала Convolutional Neural Network или сверточная нейронная сеть (СНС). Успех обусловлен возможностью учета двумерной топологии изображения.

СНС обеспечивают частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям. СНС объединяют три архитектурных идеи, для обеспечения инвариантности к изменению масштаба, повороту сдвигу и пространственным искажениям:

* локальные рецепторные поля (обеспечивают локальную двумерную связность нейронов);
* общие синаптические коэффициенты (обеспечивают детектирование некоторых черт в любом месте изображения и уменьшают общее число весовых коэффициентов);
* иерархическая организация с пространственными подвыборками.

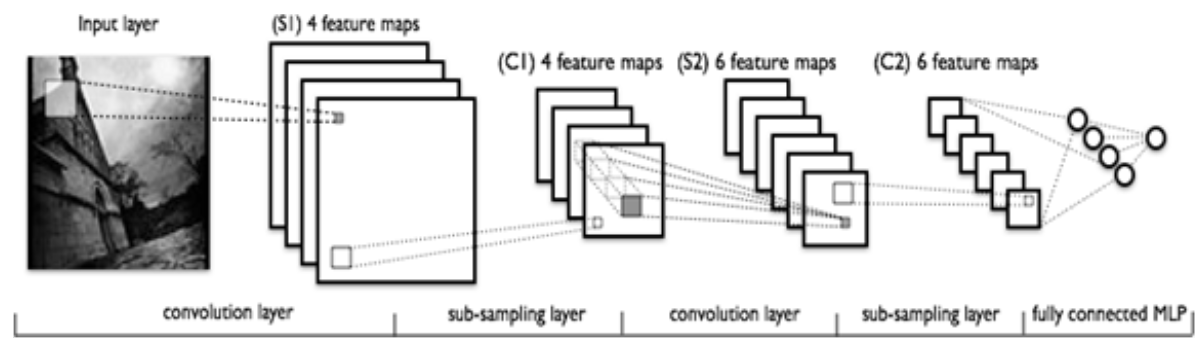
На данный момент сверточная нейронная сеть и ее модификации считаются лучшими по точности и скорости алгоритмами нахождения объектов на сцене. Начиная с 2012 года, нейросети занимают первые места на известном международном конкурсе по распознаванию образов ImageNet.

Именно поэтому в данной работе использовалась сверточная нейронная сеть, обученная по алгоритму обратного распространения ошибки.

СНС состоит из разных видов слоев: сверточные (convolutional) слои, субдискретизирующие (subsampling, подвыборка) слои и слои «обычной» нейронной сети – персептрона, в соответствии с рисунком 3.

Первые два типа слоев (convolutional, subsampling), чередуясь между собой, формируют входной вектор признаков для многослойного персептрона.

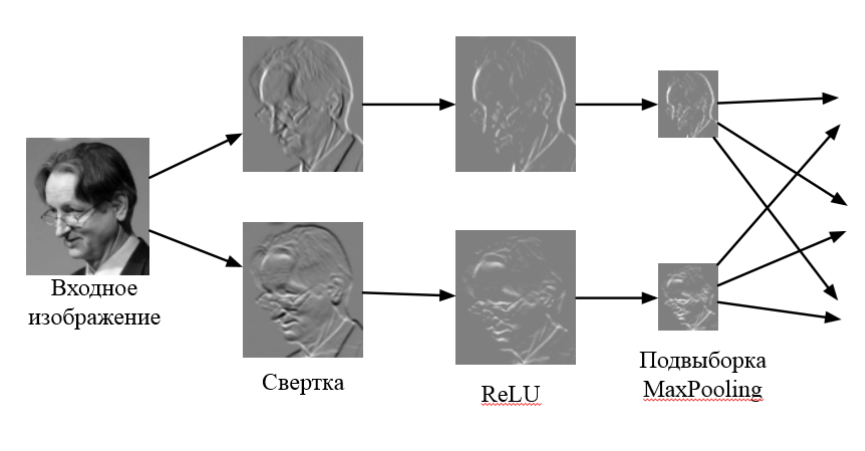
Свое название сверточная сеть получила по названию операции – свертка.

  
Рисунок 3 – Топология СНС

Сверточные сети являются удачной серединой между биологически правдоподобными сетями и обычным многослойным персептроном. На сегодняшний день лучшие результаты в распознавании изображений получают с их помощью. В среднем точность распознавания таких сетей превосходит обычные ИНС на 10-15%.

Основной причиной успеха СНС стало концепция общих весов. Несмотря на большой размер, эти сети имеют небольшое количество настраиваемых параметров по сравнению с их предком – неокогнитроном. Имеются варианты СНС (Tiled Convolutional Neural Network), похожие на неокогнитрон, в таких сетях происходит, частичный отказ от связанных весов, но алгоритм обучения остается тем же и основывается на обратном распространении ошибки.

На рисунке 4 продемонстрирована визуализация свертки и подвыборки.

  
Рисунок 4 – Визуализация свёртки и подвыборки

* 1. Обучение нейронной сети

Цель обучения состоит в подборе таких значений весов и для двух слоёв сети, чтобы при заданном входном векторе получить на выходе значения сигналов , которые с требуемой точностью будут совпадать с ожидаемыми значениями для Если рассматривать единичный сигнал порогового элемента как один из компонентов входного вектора , то веса пороговых элементов можно добавить в векторы весов соответствующих нейронов обоих слоёв.

При таком подходе выходной сигнал -го нейрона скрытого слоя удается описать функцией:

(1)

в которой индекс 0 соответствует сигналу и весам пороговых элементов, причём , . В выходном слое -ый нейрон вырабатывает выходной сигнал, определяемый как:

(2)

Из формулы (2) следует, что на значение выходного сигнала влияют веса обоих слоёв, тогда как сигналы, вырабатываемые в скрытом слое, не зависят от весов выходного слоя.

Для того чтобы сеть можно было применять в дальнейшем, её прежде надо обучить на полученных ранее данных, для которых известны и значения входных параметров, и правильные ответы на них. Это обучение состоит в подборе весов межнейронных связей, обеспечивающих наибольшую близость ответов сети к известным правильным ответам.

Алгоритм обратного распространения ошибки определяет стратегию подбора весов многослойной сети с применением градиентных методов оптимизации. В настоящее время считается одним из наиболее эффективных алгоритмов обучения многослойной сети. При обучении ставится задача минимизации целевой функции, формируемой, как правило, в виде квадратичной суммы разностей между фактическими и ожидаемыми значениями выходных сигналов, которая для Р обучающих выборок определяется по формуле:

(3)

В случае единичной обучающей выборки целевая функция имеет вид:

(4)

Уточнение весов может проводиться после предъявления каждой обучающей выборки (так называемый режим «онлайн»), при этом используется целевая функция вида (4), либо однократно после предъявления всех обучающих выборок (режим «оффлайн»), при этом используется целевая функция вида (3). В последующем изложении используется целевая функция вида (4).

Для упрощения можно считать, что цель обучения состоит в таком определении значений весов нейронов каждого слоя сети, чтобы при заданном входном векторе получить на выходе значения сигналов , совпадающие с требуемой точностью с ожидаемыми значениями при .

Обучение сети с использованием алгоритма обратного распространения ошибки проводится в несколько этапов.

На первом из них предъявляется обучающая выборка и рассчитываются значения сигналов соответствующих нейронов сети. При заданном векторе определяются вначале значения выходных сигналов скрытого слоя, а затем значения выходного слоя. Для расчета применяются формулы (1) и (2). После получения значений выходных сигналов становится возможным рассчитать фактическое значение целевой функции ошибки .

На втором этапе минимизируется значение этой функции.

Так как целевая функция непрерывна, то наиболее эффективными методами обучения оказываются градиентные алгоритмы, согласно которым уточнение вектора весов (обучение) производится по формуле:

(5)

(6)

– коэффициент обучения, а – направление в многомерном пространстве . В алгоритме обратного распространения ошибки определяется как частная производная , взятая со знаком минус.

Обучение многослойной сети с применением градиентных методов требует определения вектора градиента относительно весов всех слоёв сети, что необходимо для правильного выбора направления . Эта задача имеет очевидное решение только для весов выходного слоя. Для других слоёв используется алгоритм обратного распространения ошибки, который определяется следующим образом.

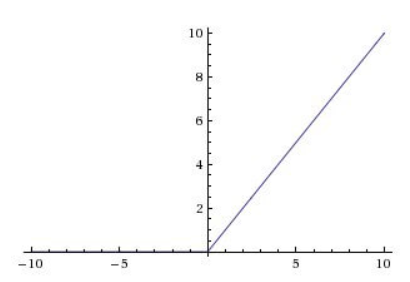
1. Подать на вход сети вектор и рассчитать значения выходных сигналов нейронов скрытых слоев и выходного слоя, а также соответствующие функций активации каждого слоя ( – количество слоёв).
2. Создать сеть обратного распространения ошибок путём изменения направления передачи сигналов, замены функций активации их производными и подачи на бывший выход сети в качестве входного сигнала разности между фактическими и ожидаемыми значениями.
3. Уточнить веса по формулам (5) и (6) на основе результатов, полученных в п.1 и п.2 для исходной сети и для сети обратного распространения ошибки.
4. Пункты 1, 2, 3 повторить для всех обучающих выборок, вплоть до выполнения условия остановки: норма градиента станет меньше заданного значения , характеризующего точность обучения.
   1. Функция активации

Функция активации определяет выходное значение нейрона в зависимости от результата взвешенной суммы входов и порогового значения.

Значение выходного сигнала может быть любым в диапазоне от до . В действительности нейрон не знает границу, после которой следует активация. Для этой цели надо добавлять активационную функцию. Она проверяет произведенное нейроном значение выходного сигнала на предмет того, должны ли внешние связи рассматривать этот нейрон как активированный, или его можно игнорировать.

В данной работе использовалась активационная функция ReLu.

Из определения следует, что ReLu возвращает значение х, если х положительно, и 0 в противном случае. Схема работы приведена на рисунке 5.

  
Рисунок 5 – Схема работы ReLu

ReLu нелинейна по своей природе, как и её комбинация ReLu. Область допустимых значений ReLu – [0,inf), то есть активация может “взорваться”.

Отдельным преимуществом данной функции является разряженности активации. Relu позволяет сделать некоторые нейроны не активированными, чтобы сделать активацию разряженной и эффективной.

Представим сеть со случайно инициализированными весами (или нормализированными), в которой примерно 50% активаций равны 0 из-за характеристик ReLu (возвращает 0 для отрицательных значений х). В такой сети включается меньшее количество нейронов (разреженная активация), а сама сеть становится легче.

Из-за того, что часть ReLu представляет из себя горизонтальную линию (для отрицательных значений X), градиент на этой части равен 0. Из-за равенства нулю градиента, веса не будут корректироваться во время спуска. Это означает, что пребывающие в таком состоянии нейроны не будут реагировать на изменения в ошибке/входных данных (просто потому, что градиент равен нулю, ничего не будет меняться). Такое явление называется проблемой умирающего ReLu (Dying ReLu problem). Из-за этой проблемы некоторые нейроны просто выключатся и не будут отвечать, делая значительную часть нейросети пассивной. Однако существуют вариации ReLu, которые помогают эту проблему избежать. Например, имеет смысл заменить горизонтальную часть функции на линейную. Если выражение для линейной функции задается выражением y = 0.01x для области x < 0, линия слегка отклоняется от горизонтального положения. Существует и другие способы избежать нулевого градиента. Основная идея – сделать градиент неравным нулю и постепенно восстанавливать его во время тренировки.

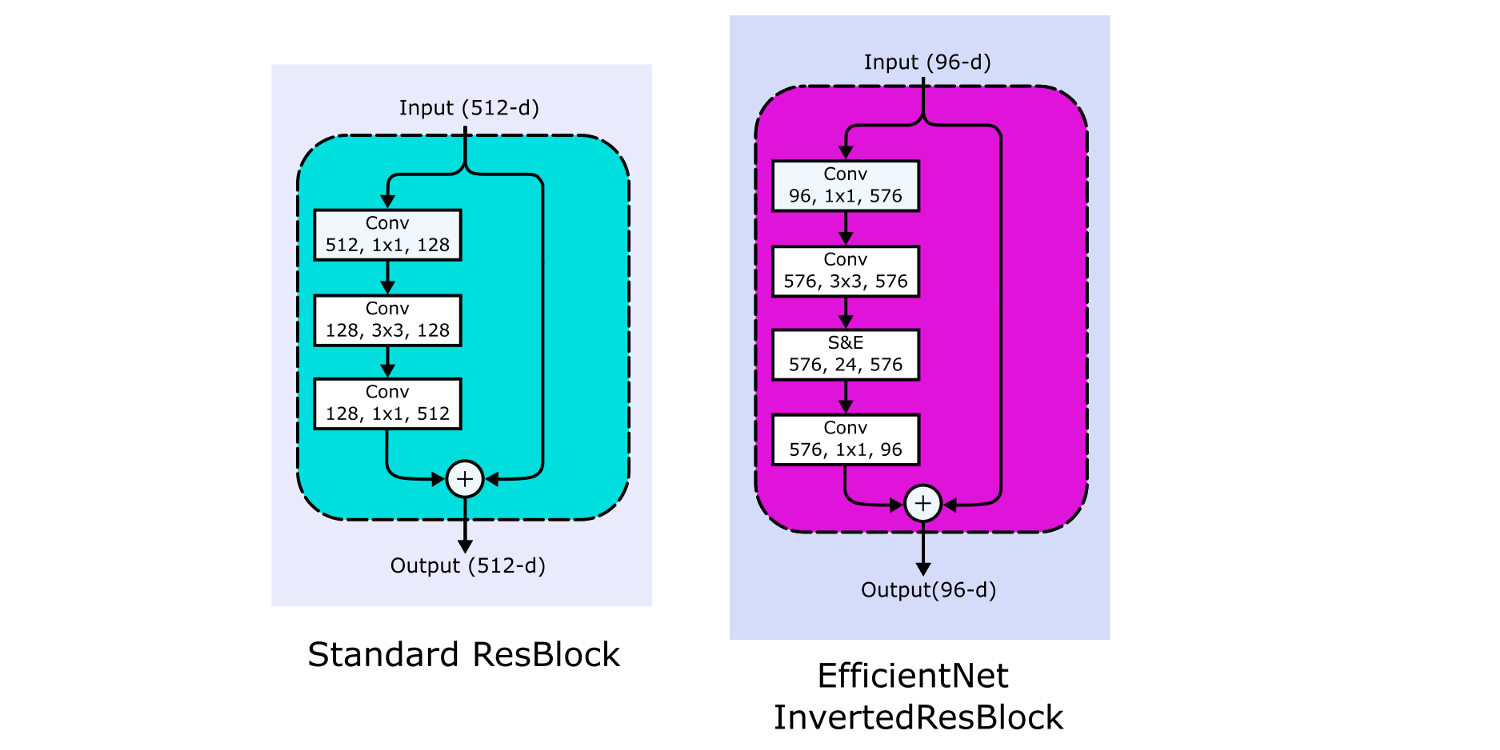
Softmax обычно применяется к выходному слою задач множественной классификации. Он может гарантировать, что сумма всех выходных нейронов равна 1, а значение интервала [0,1], соответствующего каждому выходу, является вероятностью выхода, а наибольшая вероятность используется в приложении. Результатом является окончательный прогноз. Форма функции Softmax следующая:

Функция преобразует вектор z размерности K в вектор той же размерности, где каждая координата полученного вектора представлена вещественным числом в интервале [0,1] и сумма координат равна 1. Координаты трактуются как вероятности того, что объект принадлежит классу i.

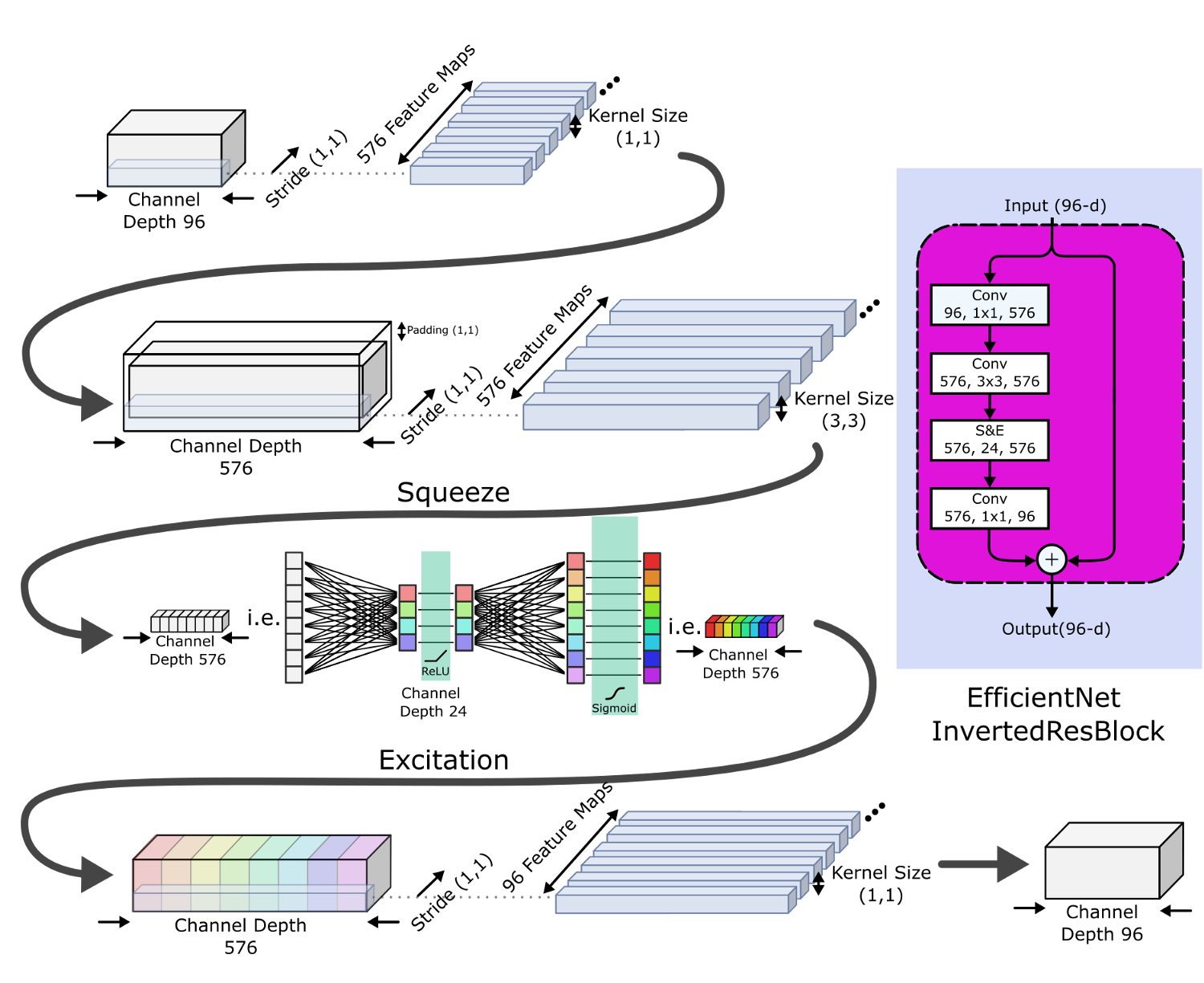
* 1. EfficientNet

Базовая нейронная сеть, EfficientNet-B0, была разработана авторами, искавшими архитектуру сети, оптимизирующую как точность, так и требуемое количество вычислений. Основной строительный блок EfficientNet-B0 – это MBConv (мобильная обращенная свертка с узким местом), к которой добавлено сжатие-и-стимуляция.

В традиционном ResBlock'е количество каналов следует образцу широкое-узкое-широкое (также называемое "узким местом"), параллельно с обходной связью идентичности. Вход имеет большое количество каналов, сжимаемых сверткой 1\*1. Затем количество опять увеличивается сверткой 1\*1, чтобы можно было прибавить данные по связи идентичности.

  
Рисунок 6 – Сравнение стандартного ResBlock'а с InvertedResBlock EfficientNet

Инвертированный остаточный блок (Inverted Residual Block), как и предполагает его название, использует подход узкий/широкий/узкий. Плоский вход расширяется сверткой 1\*1 перед применением глубинной свертки 3\*3, сильно сокращающей количество параметров. Затем снова используется свертка 1\*1 для сокращения количества каналов, чтобы можно было прибавить данные по связи идентичности.

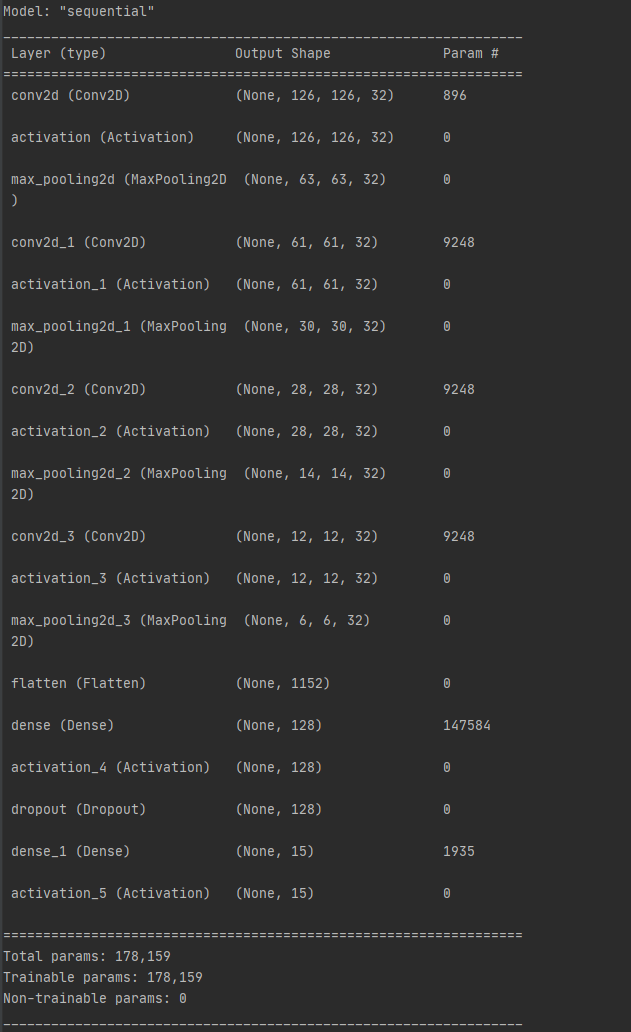
  
Рисунок 7 – Визуализация трансформации уровня в инвертированном остаточном блоке EfficientNet

Все сверточные нейронные сети (CNN) имеют три измерения: ширину, глубину и разрешение. Глубина означает количество слоев, ширина – количество каналов (например, три канала для RGB), а разрешение – количество пикселей в изображении.

Базовая нейронная сеть EfficientNet-B0 использовалась для сетевого поиска оптимальных параметров масштабирования с фиксированным комбинированным коэффициентом phi, равным единице, и лучшими параметрами оказались aplha=1.2, beta=1.1 и gamma=1.15. Затем, фиксируя оптимальные значения aplha, beta и gamma, исследователи масштабировали количество имеющихся ресурсов phi для создания моделей большего размера от EfficientNet-B1 до EfficientNet-B7.

1. Описание вычислительных экспериментов
   1. Описание параметров нейронной сети

На рисунке 8 показаны основные параметры нейронной сети в данной работе.

  
Рисунок 8 – Параметры нейронной сети

В данной работе используется набор данных Stanford Dogs Dataset, который содержит 2 714 изображений в 15 категориях. На изображениях показаны разные породы собак.

Здесь 80% (2177) изображений используются для обучения сети и 20% (537) изображений для оценки того, насколько точно сеть научилась классифицировать изображения. Каждое изображение представлено как 128х128 пикселей.

Данные должны быть предварительно обработаны перед обучением сети. Прежде чем передавать их в модель нейронной сети, необходимо масштабировать эти значения в диапазоне от 0 до 1. Для этого делим значения на 255.

Прежде чем модель будет готова к обучению, ей нужно еще несколько настроек. Они добавляются на этапе компиляции модели: функция потерь, оптимизатор и метрики.

Функция потерь – измеряет, насколько точна модель во время обучения. Необходимо минимизировать эту функцию, чтобы "направить" модель в правильном направлении. В данной работе использовалась функция потерь библиотеки Keras categorical\_crossentropy (категориальная перекрестная энтропия), формула которой:

Оптимизатор – именно так модель обновляется на основе данных, которые она видит, и ее функции потерь. В данной работе в качестве оптимизатора был выбран Adam.

Adam – ​​это алгоритм оптимизации, который можно использовать вместо классической процедуры стохастического градиентного спуска для итеративного обновления весов сети на основе обучающих данных. Он использует среднее значение вторых моментов градиентов (нецентрированная дисперсия). В частности, алгоритм вычисляет экспоненциальную скользящую среднюю градиента и квадрата градиента, а параметры и управляют скоростью затухания этих скользящих средних. Начальное значение скользящих средних и значений и , близких к 1,0 (рекомендуется), приводит к смещению оценок моментов в сторону нуля. Это смещение преодолевается сначала вычислением смещенных оценок, а затем вычислением оценок с поправкой на смещение.

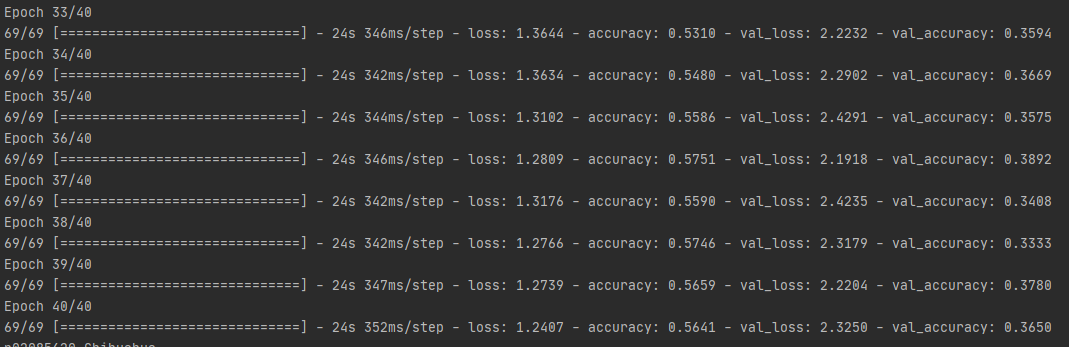
Метрики – используются для мониторинга этапов обучения и тестирования. В работе используется точность (accuracy), доля правильно классифицированных изображений.

* 1. Результаты экспериментов
     1. Основной эксперимент

В основном эксперименте обучение проходило 40 эпох. Изображения разбились в следующем соотношении: 80% для обучения, 20% для тестирования.

В обучении нейронная есть достигла точности 56% и значения функции потерь 1,2. В тестировании точность достигла 36%, а функция потерь – 2,3.

Результаты представлены на рисунке 9.

  
Рисунок 9 – Результаты основного эксперимента

* + 1. Эксперимент с уменьшенным количеством эпох

Этот эксперимент был проведён с количеством эпох равным 25.

В обучении нейронная есть достигла точности 40% и значения функции потерь 1,8. В тестировании точность достигла 28%, а функция потерь – 2,1.

Результаты представлены на рисунке 10.

  
Рисунок 10 – Результаты эксперимента

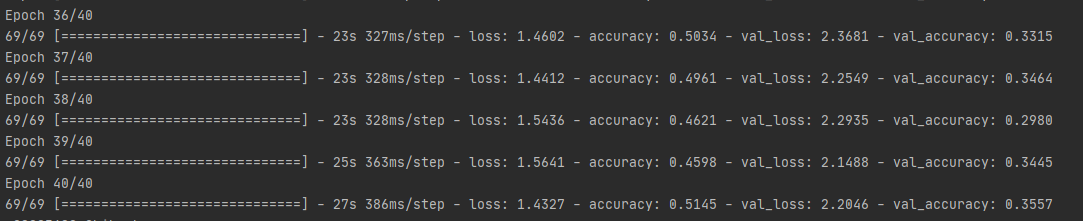
Можно заметить, что при уменьшении количества эпох точность ухудшается.

* + 1. Эксперимент с уменьшенным количеством слоёв

Этот эксперимент был проведён с количеством слоёв равным 2.

В обучении нейронная есть достигла точности 51% и значения функции потерь 1,4. В тестировании точность достигла 35%, а функция потерь – 2,2.

Результаты представлены на рисунке 11.

  
Рисунок 11 – Результаты эксперимента

Можно заметить, что при уменьшении количества слоёв точность ухудшается.

1. Результат работы

После проведения экспериментов было принято решение использовать готовую свёрточную сеть EfficientNet, описанную в пункте 2.4.

Результаты работы можно увидеть на рисунке 12.

  
Рисунок 12 – Результаты работы программы

Из рисунка становится понятно, что EfficientNet имеет высокую точность предсказания.

ЗАКЛЮЧЕНИЕ

В процессе выполнения лабораторной работы была разработана программа, позволяющая определить, к какому из заданных классов относится изображение.