Geekbrains

**Использование технологии Transfer Learning для классификации изображений по типам клеток крови человека на примере нейросетей VGG19 и ResNet50**

Программа: Разработчик - аналитик

Специализация: Data Engineer.

Мишина Олеся Анатольевна

Москва

2024

**Содержание.**

Введение…………………………………………………………………………..1

1. Теоретические основы Transfer Learning ……………………………………5
   1. Основные типы Transfer Learning……………………………………5
   2. Основные категории Transfer Learning………………………………9
   3. Рекомендации по выбору и применению типов Transfer

Learning……………………………………………………………....16

* 1. Применение Transfer Learning в анализе медицинских

изображений…………………………………………………………19

1. Обзор используемых нейросетей…………………………………………...23
   1. Нейросеть VGG 19…………………………………………………...23
   2. Нейросеть ResNet50…………………………………………………25
2. Классификация типов клеток крови c использованием

transfer learning.................................................................................................28

* 1. Импорт и описание исследуемыхданных………………………….28
  2. Предварительная обработка данных…………………………….….29
  3. Классификация типов клеток крови сетью VGG19

методом извлечения признаков...…………………………………...34

* 1. Классификация типов клеток крови сетью VGG19

методом тонкой настройки…......…………………………………...44

* 1. Классификация типов клеток крови сетью ResNet50

методом извлечения признаков ...……………………………...…...49

* 1. Классификация типов клеток крови сетью ResNet50

методом тонкой настройки……...……………………………...…...52

Заключение………………………………………………………………………59

Список литературы……………………………………………………………...63

**Введение.**

Диагностика заболеваний крови часто приводит к необходимости идентифицировать и характеризовать образцы крови пациента. Поэтому вопрос создания методов автоматизированного обнаружения и классификации подтипов клеток крови имеет важное медицинское значение. В данной работе будет проанализирована возможность использования предобученных нейронных сетей для классификации медицинских изображений, а именно типов клеток крови, используя методы Transfer learning.

Обучение нейронной сети для классификации медицинских изображений является сложной и важной задачей в области медицинской визуализации. В настоящее время исследователи уже научились строить эффективные глубокие модели для решения большого спектра задач компьютерного зрения, обработки изображений и естественного языка. Однако обучение подобных моделей для достаточно точной классификации конкретного типа изображений требует значительного объема размеченных тренировочных данных, поскольку большинство задач решается с использованием методов обучения с учителем, а также - это длительный, трудоемкий и дорогостоящий процесс. На данный момент в связи с развитием технического оснащения рабочих мест врачей, увеличивается не только количество доступных данных, но и расширяется список медицинских сфер для применения компьютерного анализа изображений. И если мы научили определенную сеть хорошо классифицировать, например, типы клеток крови, а нам нужна сеть, которая классифицирует типы раковых клеток, то зачастую намного проще и быстрее дообучить уже готовую сеть, используемую нами ранее, нежели обучать новую с нуля. Этот принцип и лежит в основе метода Transfer learning.

Таким образом Transfer Learning - это метод машинного обучения, который предполагает использование уже обученной модели на решении одной задачи, для решения другой, похожей задачи. Первые статьи, посвященные трансферному обучению нейронных сетей, появились еще в конце 70х. С тех

пор теоретическая база методов transfer learning постоянно дополняется, обновляется и расширяется.

При анализе изображений окружающего нас мира использовать для классификации изображений, обнаружения объектов или их сегментации доступные предварительно обученные модели намного проще. Причина этого в том, что все RGB-изображения получены при помощи одного физического принципа - отражения света от реальных объектов - и имеют примерно одно распределение модальности. Сдвиг между различными наборами данных RGB не очень велик. Однако, природа медицинских изображений в большинстве случает отличается от обычного RBG изображения. Каждый медицинский прибор создает изображения, основанные на различных физических принципах. Следовательно, все медицинские изображения, как правило, имеют распределения модальности, отличные тех, на которых обучались сети, используемые для переобучения.

Для моей задачи я возьму две доступные глубокие нейронные сети, VGG19 и ResNet50, это будут исходная сети (их называют backbone или upstream). Обе они обучались на очень большом наборе данных ImageNet из нескольких миллионов изображений с фотографиями разных объектов: собак, кошек, деревьев, машин, животных, людей, и так далее. Этот набор данных содержит 1000 классов различных объектов. Обе сети принимают на входном слое изображение размера X на Y и возвращают на последнем слое в качестве результата распределение вероятности принадлежности входного изображения к одному из тысячи различных классов. Данные проходят от входов к выходам через большое количество слоев, каждый из которых имеет свою собственную функцию преобразования входного изображения, и в конце обучения мы находим веса для каждого слоя нейронной сети и сохраняем их. Я хочу использовать эти сети для задачи классификации объектов, не присутствующих в первом датасете (эта целевая задаче называется "таргетной" или downstream.), а именно, клеток крови человека, но данных слишком мало для полного обучения глубокой нейросети с нуля. Здесь нам и может помочь transfer learning.

Применение переноса обучения заключается в том, чтобы взять копию нашей нейронной сети, в которой сохраняются все слои вплоть до предпоследнего. Последний слой мы отрежем, и добавим к сети другую нужную нам конструкцию слоев, выходной слой которой будет иметь уже нужное нам количеством выходов для классификации (4 в моей работе). Я буду в качестве этой конструкции использовать простейший классификатор с минимальным количеством параметров. На выходе мы будем искать уже вероятностное распределение принадлежности данного конкретного изображения клетки крови к какому-то из классов клеток крови по максимуму этого распределения. Такой способ использования сети называется извлечением признаков (feature extraction). Однако, при переходе на новый набор данных (домен), нашей предобученной модели может понадобиться адаптация, которая называется доменной адаптацией. И что бы избежать эффекта переобучения исходной сети, я буду "замораживают" часть слоев, то есть запрещу им менять свои веса в процессе обучения, что так же позволит мне экономить время на вычислениях. В последствии я "разморожу" часть сети и на очень маленькой скорости ее дообучить (fine tuning – тонкая настройка). Таким образом, в условиях ограниченности количества входных данных, transfer learning самая подходящая техника для решения таргетной задачи.

В современной медицинской практике метод transfer learning широко применяется для анализа в задачах компьютерного зрения на любых медицинских изображениях: рентгеновcких снимках, снимках микроскопов, гистологический снимках, ЭЭГ, ЭКГ, изображениях, полученные с эндоскопической камеры, снимках глаза для диагностики и распознавания воспалений, новообразований, глазных инфекций.

1. **Теоретические основы Transfer learning**
   1. **Основные типы Transfer Learning**

Домен в машинном обучении представляет собой это набор объектов с общими свойствами. Для наглядности в качестве примера возьмем ML приложение для классификации дефектов программных модулей, которое обучается предсказывать, имеет ли программный модуль дефект или нет [1].

Любой домен D определяется пространством признаков X и распределением вероятности случайной величины x = {𝑥1, 𝑥2, … , 𝑥𝑛} ∈ Χ [3]. То есть можно записать 𝐷 = {Χ, 𝑃(x)}. Например, в нашем ML приложении каждая программная метрика берется как функция, то xi - это i-й вектор признаков (экземпляр), соответствующий i-му программному модулю, n - количество векторов признаков в X, а X - конкретная обучающая выборка. Соответственно два домена не совпадают 𝐷1 ≠ 𝐷2, если не совпадают либо их пространства признаков Χ1 ≠ Χ2 (гетерогенное трансферное обучение), либо распределения 𝑃(x1) ≠ 𝑃(x2).  Если  Χ1 = Χ2 то это однородное трансферное обучение. Возвращаясь к примеру с классификатором дефектов программных модулей, гетерогенное трансферное обучение - это случай, когда исходный программный модуль имеет другие показатели (features), чем целевой программный модуль. В случае совпадения показателей программного обеспечения исходного и целевого программных модулей получаем однородное трансфертное обучение. Примером не совпадения распределений является случай, когда исходная программа написана для системы пользовательского интерфейса, а целевая программа написана для алгоритма декодирования сигналов DSP.

Определим для данного домена D задачу 𝑇 = {Y, 𝑓(∙)}, где Y – пространство меток, которые выдаст алгоритм обучения, а 𝑓(∙) функция предсказания, которая представляет собой распределение условной вероятности того, что при заданном признаковом описании 𝑥 наблюдается определенная метка 𝑦: 𝑓(𝑥) = 𝑃(𝑦|𝑥). Получаем, что две задачи являются несовпадающими 𝑇1 ≠ 𝑇2, если различаются или пространства их меток Y1 ≠ Y2, либо распределения вероятностей 𝑃(𝑌1|x1) ≠ 𝑃(𝑌2|x2). На примере приложение для классификации дефектов программного модуля, 𝑌 представляет собой набор меток и в данном случае содержит true и false, yi принимает значение true или false, а f(x) - это классификатор, который предсказывает значение метки для программного модуля x. Примером не совпадения пространства меток является случай, когда исходный программный модуль имеет двоичное пространство меток (true и false), а целевой домен имеет пространство меток из другого количества уровней дефективности модулей. Примером несоответствия условного распределения является случай, когда конкретный программный модуль выдает разные результаты для одинаковых дефектов в исходном и целевом доменах.

Пусть у нас есть исходная задача 𝑇s в домене 𝐷s и целевая задача 𝑇t в домене 𝐷t. Классическое машинное обучение используют, когда совпадают и домены 𝐷t = 𝐷s и задачи 𝑇t = 𝑇𝑠. В то время как при переносе обучения не совпадают или домены 𝐷t ≠ 𝐷s, или задачи 𝑇t ≠ 𝑇𝑠, и цель состоит в повышении качества функции предсказания (классификатора) 𝑓t(∙) целевой задачи в домене 𝐷t с использованием знаний, полученных при обучении задачи 𝑇s в домене 𝐷s.

Теперь мы можем обозначить основные типы моделей Transfer Learning. Исходя из наличия или отсутствия меток в исходной и целевой областях мф можем выделить следующие типы: **Inductive transfer learning** (индуктивный перенос обучения), **Transductive transfer learning** (трансдуктивный перенос обучения), **Unsupervised** **transfer learning** (обучения без учителя). Основные их различия и области применения можно увидеть в таблице 1.

При индуктивном переносе обучения исходная задача и целевая не совпадают 𝑇t ≠ 𝑇s, а совпадение доменов не имеет значения. Основной целью индуктивного трансферного обучения является повышение эффективности обучения и производительности целевой задачи путем применения знаний,

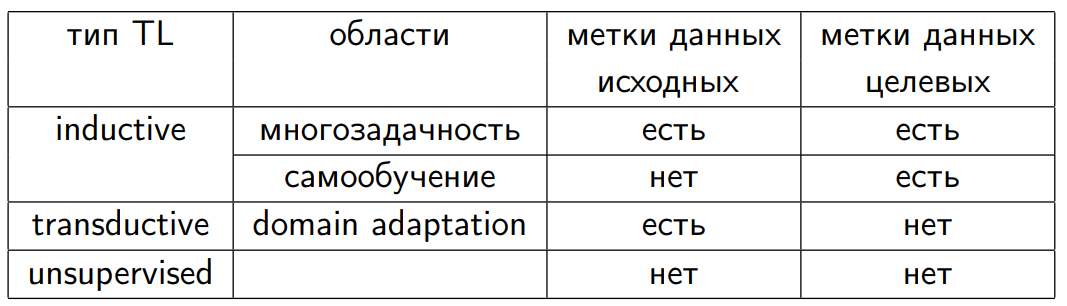


Рис 1. Сводная таблица типов Transfer Learning, [4]

извлеченных из связанной исходной задачи. Это достигается путем переноса

индуктивного смещения, представляющего собой набор предположений, которые используются для прогнозирования выходных данных на входных данных, с исходной задачи на целевую. Для создания корректной функции предсказания необходимы размеченные данные в рамках целевого домена. Индуктивный перенос обучения делят на два класса в зависимости от разметки исходных данных. При наличии разметки тренировочных данных в исходной задаче выделяют:

- **one-shot transfer learning или n-shot learning** (однократное обучение) самый часто встречающийся метод переноса обучения, как правило используется в ситуациях, когда получение больших объемов размеченных данных нецелесообразно или невозможно. Суть его в том, что модель обучается на небольшом объеме помеченных данных в исходной задаче и применяет эти знания к связанной целевой задаче. Например, автоматические сканеры паспортов, турникеты принимают данные с одной целью, подтвердить или отклонить сканируемые данные.

- **multitask learning** (многозадачное обучение), когда перенос может происходить между двумя или более учебными задачами, выполняемыми одновременно, для этого должна быть разметка тренировочных данных в исходной задаче. В качестве примера рассмотрим модель, обученную распознавать на изображениях как кошек, так и собак. В этом случае перед моделью стоят две задачи: распознавание кошек и распознавание собак. Модель изучает признаки, которые являются общими для обеих задач (например, обнаружение краев, распознавание текстур и т. д.), а также признаки, специфичные для каждой задачи (например, различение форм кошек и собак).

- **self-taught learning** (самообучение), в этом случае разметки входных данных нет. Например, в работе [3] авторы рассматривают в качестве примера задачу компьютерного зрения по классификации изображений слонов и носорогов.

При трансдуктивном переносе обучения (Transductive Transfer Learning) совпадают задачи 𝑇t = 𝑇s, а домены имеют сильное сходство, но не являются полностью одинаковыми. Исходный домен обычно содержит большой объем помеченных данных, а целевой домен содержит только ограниченное количество непомеченных данных. Для получения ft(·) можно адаптировать функцию fs (·) для использования в Tt на данных из Dt. Здесь тоже возможно два класса:

- **domain confusion** (путаница доменов), не совпадают пространства признаков исходного и целевого домена, Χs ≠ Χt, и модель при обучении не может различать исходную и целевую области, тем самым изучая признаки, которые являются инвариантными к предметной области. Например, мы имеем модель, обученную распознавать породы собак по изображениям (исходный домен). Теперь мы хотим использовать эту модель для распознавания пород собак по другому набору изображений, значительно отличающемуся от тех, что находятся в нашем исходном наборе данных (целевая область). Например, в нашем исходном наборе данных у нас есть только изображения пуделей, а в нашем целевом наборе данных у нас есть изображения различных пород собак.

- **domain adaptation** (адаптация домена), не совпадают распределения вероятности 𝑃(Χs) ≠ 𝑃(Χt) и модель обучается на исходном домене, а затем адаптируется для корректной классификации в другом, но связанном целевом домене. Например, рассмотрим модель, обученную распознавать собак на изображениях, сделанных в течение дня (исходный домен). Теперь мы хотим использовать эту модель для распознавания собак на изображениях, сделанных ночью (целевая область). Несмотря на то, что задача одна и та же (распознавание собаки), домены различны из-за изменения условий освещения.

Перенос обучения без учителя (Unsupervised Transfer) работает аналогично индуктивному трансферному обучению. Разница заключается в том, что здесь не предполагается наличие меток классов ни для целевого домена данных, ни для исходного. Задачи здесь не совпадают 𝑇t ≠ 𝑇s. Можно выделить следующие категории:

- **multitask learning** (многозадачное обучение)

- **zero-shot learning** (обучение с нуля), когда модель обучается на одном наборе классов, а затем тестируется на совершенно другом наборе классов, который она никогда раньше не видела. В качестве примера рассмотрим модель, которая обучена классифицировать животных средней полосы, присваивая им текстовые метки. А мы подаем на вход сети изображение зебры, и благодаря известному ей распределению классов, модель присваивает зебре правильную метку – например, полосатая лошадь. Звучит необычно, но метод появился в 2008 году и широко применяется в области компьютерного зрения.

* 1. **Основные категории Transfer Learning**

В зависимости от того, что именно мы ходим передать из исходных данных в целевой домен, можно выделить 4 категории (Рис. 2).

- **instance transfer,** или передача примеров, предполагает, что определенная часть данных из Ds может быть передана для обучения в Dt посредством переназначения их весов (TrAdaBoost, Sample Reweiting) или при помощи метода опорных векторов (SVM). Данную концепцию, например, использовали авторы [5] для анализа и мониторинга структурного состояния мостов. Метод особенно подходит для структурно схожих и часто повторяющихся типов мостов и других распространенных дорожных конструкций (эстакад). Авторы так же повторно взвешивают размеченные исходные данные в зависимости от их соответствия и сходства с целевыми данными, что помогает изучить новый мост с точки зрения (целевой домен). Полезным исходным данным присваиваются большие веса, и их можно объединить с целевыми данными на этапе обучения модели/классификатора.

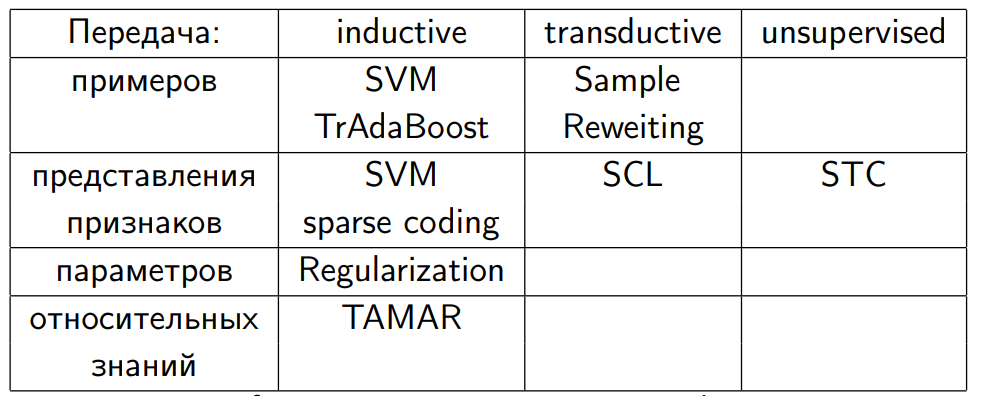


Рис.2. Соответствие типов Transfer Learning категориям передачи с указанием используемых методов обучения [4] .

- **feature representation transfer** или передача представления признаков предполагает целью получить упорядоченное по какому-то определенному критерию представление для целевого домена Dt. Знания исходного домена, используемые для передачи, кодируются в определенное представление признаков и передаются в целевую задачу для ее улучшения. Примером этого подхода может служить работа [6] где авторы в своем приложении для классификации диффузных заболеваний легких используют двухступенчатую передачу признаков. Изначально модель последовательно обучается на огромном количестве естественных изображений, некоторых текстурных изображений и целевых изображений, и передает признаки в задачу классификации текстурных рентгеновских компьютерно-томографических изображений диффузных заболеваний легких. В этом эксперименте двухступенчатая передача признаков обеспечивает лучшую производительность по сравнению с обучением с нуля и обычной одноступенчатой передачей признаков.

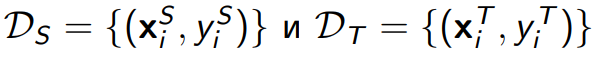
- **parameter transfer** или передача параметров, предполагает, что исходная задача Ts и целевая задача Tt имеют общие параметры моделей θ или априорные распределения параметров fs (·) и ft (·). Передаваемые данные кодируются так, чтобы оставить только общие параметры или признаки, и далее передаются в целевую задачу. Передача параметров широко используется для решения современных задач переобучения моделей. Например, авторы [8] в своей работы используют этот подход для анализа метаобучения моделей при передаче параметров на основе метода градиента (GBML).

**- transfer learning for relational domains** (relational-knowledge-transfer) или передача относительных знаний предполагает, что некоторое соотношение между данными в доменах Dt и Ds аналогичны и имеют какие- то пропорции. Эти пропорции и передаются в целевую задачу. Здесь часто используется алгоритм TAMAR - Transfer via Automatic Mapping and Revision.

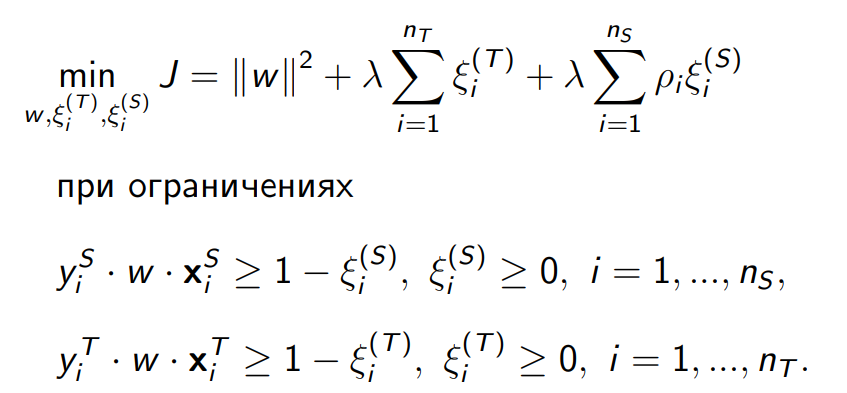
Рассмотрим каждую категорию подробнее в соответствии с типом transfer learning.

**Inductive transfer learning, передача примеров.** Мы используем данные из обоих доменов, но некоторые из исходных данных (xi , yi)∈Ds полезны для ft(·), а другие могут наоборот все испортить Необходимо выбрать только полезные исходные данные и выбросить остальные. Один из методов - назначить веса ρi для исходных данных, оставленных в примерах в соответствии с их значимостью для целевой функции распределения f t(·) и определить размеры этих весов исходя из эвристических методов.

Определим исходный и целевой домены:



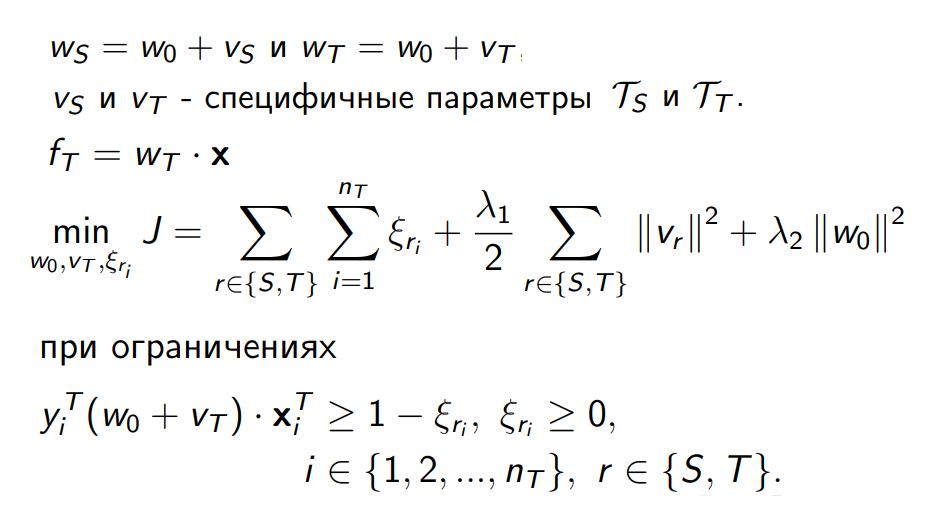
Теперь мы ищем минимум функции потерь J с учетом новых весов pi :



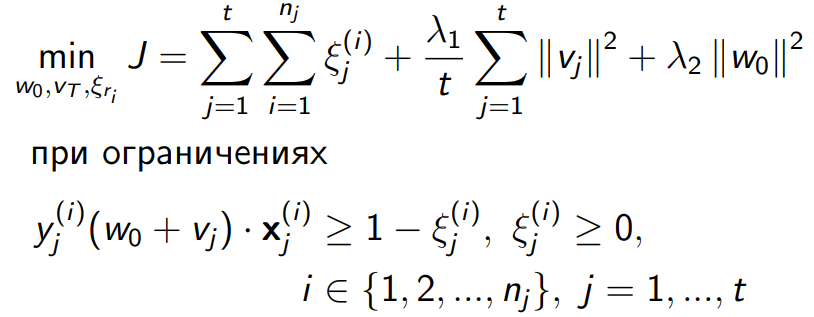


Как можно увидеть, разница со стандартной функцией потерь заключается только во втором члене равенства, зависящего от новых весов модели.

**Inductive transfer learning, передача параметров**. Пусть ws и wt веса модели для задач Ts и Tt соответственно, а w0 – общая часть весов моделей. Предположим, что веса wt и ws имеют нормальное распределение вероятностей, тогда wt и ws близки к некоторому среднему вектору параметров w0. Тогда wt и ws мы можем выразить через общий вес и отдельные веса:



При обобщении на многозадачную ситуацию с t источниками данных с параметрами wi = w0 + vi , где i = 1, ...,t, а w0 - общие параметры функцию потерь можно записать так:



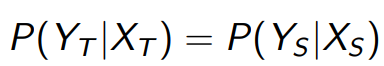
где



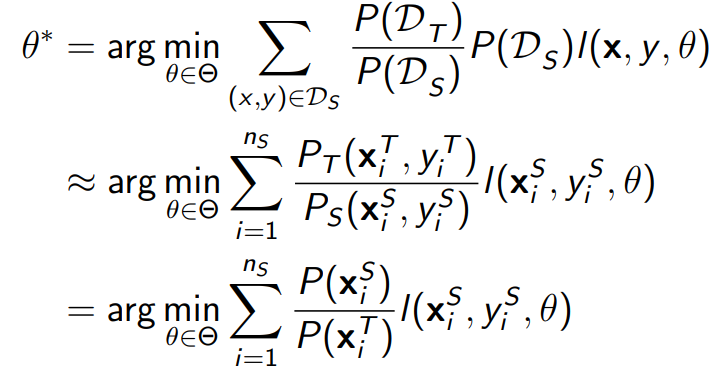
**Transductive transfer learning** предполагает **передачу примеров и представления признаков** иосуществляется аналогично индуктивной передаче параметров, но учитывая отсутствие меток в целевом домене. Рассмотри особенности трансдуктивного переноса обучения. Оптимальные параметры θ\* модели можно определить как минимизацию:



где *I*(x, y, *θ*) функция потерь и зависит от θ. Так как здесь целевые данные не размечены, необходимо обучать модель используя исходные данные Ds. Из равенства задач следует совпадения и пространства меток, и вероятностей

,

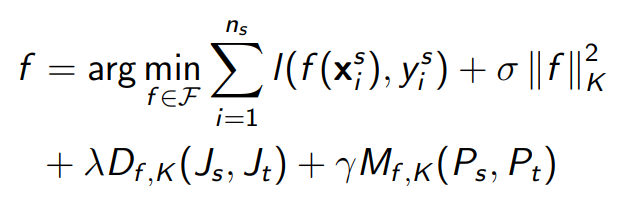
тогда отношение вероятностей распределения P(Ds) и P(Dt) определяется только отношением P(Xt) и P(Xs) и оптимальное состояние системы можно записать через исходные данные:



Для оптимизации состояния системы мы должны решить три проблемы: минимизировать функцию fs(·), минимизировать разницу распределениями

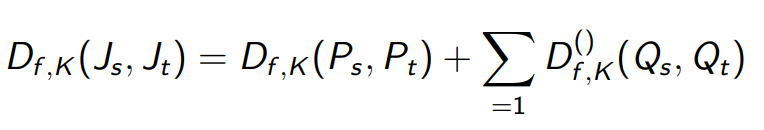
вероятностей Js и Jt (или вычисление ) и максимизировать согласованность предельных распределений Ps и Pt.

Для нахождения минимума функции fs(·):

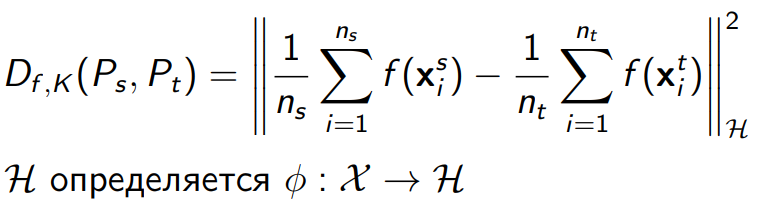
,

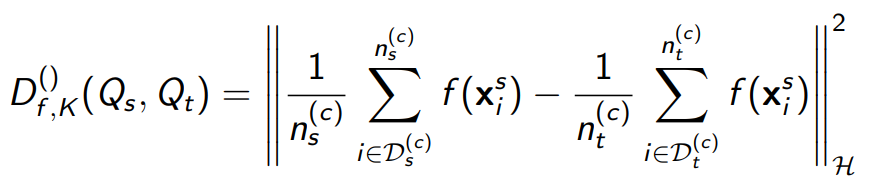
где K - ядро σ, λ, γ - положительные параметры регуляризации (ограничения на f ).

Найдем как:

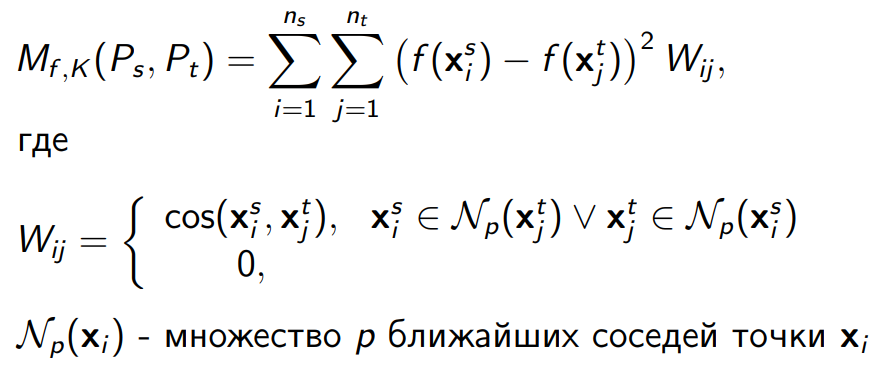
,

где



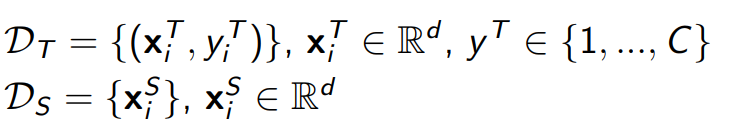


Максимизация согласованности предельных распределений Ps и Pt основана на использовании выборочной дисперсии:

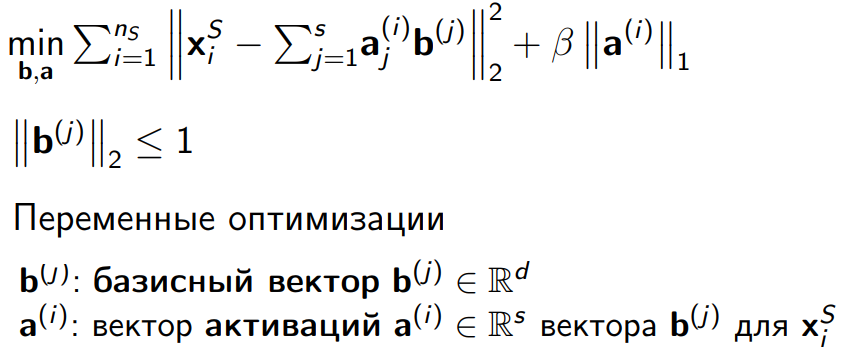


В итоге задача адаптации функции предсказания f(.) сводится к стандартной задаче квадратичного программирования [9].

**Unsupervised Transfer, передача представления признаков** без учителя. Основная идея заключается так же в использовании Ds для улучшения ft (·). Определим исходный и целевой домены [3]:



Нам нужно решить следующую задачу оптимизации на Ds, где 1-е слагаемое реконструирует xi источника через весовую линейную комбинацию базисных векторов b(j) с весами a(i), а 2-е слагаемое ограничивает веса a(i) единичной нормой - получаем разреженные веса:



Решая эту задачу, для каждой точки целевого пространства (xi , yi ), вычисляем разреженный вектор признаков aj(i), который и будет реконструированным представлением вектора x в целевой задаче. То есть тренировать модель мы будем на целевой задаче (a(xi ), yi ) вместо (xi , yi).

* 1. **Рекомендации по выбору и применению типов Transfer Learning**

Принятие решения о том, какие знания передавать, когда и как переносить, и какими методами является основной проблемой при переводе обучения. Эффективность трансферного обучения во многом зависит от принятия правильных решений в этих аспектах. Основные факторы для выбора типа переноса обучения это размер тренировочного набора данных целевой задачи и степень сходства данных для обучения моделей целевой и исходной задач, соответственно, получаем 4 варианта .

Если набор данных целевой задачи ограничен, но он похож на набор данных исходной задачи, то не имеет смысла выполнять на нем тонкую настройку весов глубокой модели вследствие возможности переобучения. Наиболее правильное решение здесь использовать начальную часть исходной сети для извлечения признаков и на этих признаках обучать простейший классификатор.

Если набор данных целевой задачи большой, и он похож на набор данных исходной задачи, то можно выполнить тонкую настройку весов сети, ожидая при этом повышение эффективности работы нейросети на целевой задаче.

Если набор данных целевой задачи небольшой, и он отличается от набора данных исходной задачи, то следует обучать простейший классификатор на признаках, извлеченных из наиболее ранних слоев исходной сети, так как более ранние слои сети содержат больше преобразований, описывающих особенности именно набора данных. При этом следует полностью обучить сеть, так как тренировочные наборы данных исходного и таргетного домена принципиально отличаются друг от друга.

Если набор данных целевой задачи большой, и он принципиально отличается от набора данных исходной задачи, то нужно обучить модель, структура которой аналогична модели исходной задачи. На практике часто оказывается, что начальная инициализация весов значениями, полученными в ходе решения исходной задачи, позволяет более точно выполнить настройку параметров сети для решения целевой задачи.

Теперь рассмотрим рекомендации по решению некоторых типов задач переноса обучения. Для этого нам понадобятся понятия извлечение признаков и тонкая настройка модели. Задача извлечения признаков состоит в использовании представлений, уже изученные предыдущей сетью, для извлечения значимых признаков из целевого набора данных. Нужно просто добавить новый классификатор, который будет обучаться с нуля, поверх предварительно обученной модели, заморозив предыдущие слои сети. Вам не нужно (повторно) обучать всю модель. Задача тонкой настройки (**fine-tuning,** файнтюнинг) предполагает разморозку несколько верхних слоев замороженной базовой модели и совместное обучение как и недавно добавленного классификатора, так и базовой модели. Это позволяет точно настроить представления функций более высокого порядка в базовой модели на конкретную целевую задачу.

**Задача доменной адаптации** как правило включает в себя следующие этапы:

**-** извлечение признаков,

**-** выравнивание домена: модель учится выравнивать или сопоставлять объекты из исходного домена с целевым доменом. Это помогает модели понять, что, несмотря на различия распределении модальности изображения, общие черты изображения остаются неизменными,

- тонкая настройка модели.

**Многозадачное обучение** как правило включает в себя следующие этапы:

- извлечение общих признаков: модель использует данные из всех задач для изучения важных признаков,

- обучение по конкретным задачам: модель изучает функции, специфичные для конкретной задачи, используя данные из каждой задачи. Эти функции используются для прогнозирования для каждой задачи,

- тонкая настройка: модель настраивается на основе данных из всех задач, что позволяет ей лучше адаптироваться к каждой задаче.

Процесс **одноразового обучения** обычно включает в себя следующие этапы:

- извлечение признаков,

- обучение сходству: модель изучает функцию сходства, которая измеряет, насколько похожи или различны два изображения. Эта функция используется для сравнения нового, невидимого изображения с одним примером из обучающей выборки,

- классификация: если сходство между новым изображением и примером ниже определенного порогового значения, модель классифицирует новое изображение как принадлежащее к тому же классу, что и пример.

При одноразовом обучении существует возможность ошибки переобучения относительно одного примера обучения.

**Обучение с нуля** обычно включает в себя следующие этапы:

**-** извлечение признаков.

- семантическое внедрение: модель изучает пространство семантического внедрения, в котором объекты одного класса находятся близко друг к другу, а объекты разных классов находятся далеко друг от друга. Это пространство встраивания обычно изучается с помощью вспомогательной информации, такой как метки классов или текстовые описания.

- классификация: модель классифицирует невидимые классы на основе их близости к видимым классам в пространстве внедрения.

При обучении с нуля может возникнуть ошибка смещения домена (domain Shift). Она происходит, когда распределение невидимых классов значительно отличается от видимых классов.

* 1. **Применение Transfer Learning в анализе медицинских изображений**

Наборов данных естественных изображений существует значительно больше, чем медицинских. Наиболее распространенным для переноса обучения является набор ImageNet с более чем миллионом изображений, которые представляют собой 1000 различных классов объектов. Поэтому модели, предварительно обученные на ImageNet, имеют много параметров на выходном слое. Однако, наборы данных медицинских изображений имеют пока не более 20 классов. В связи с этим возникает вопрос: насколько повторное использование функций распределения модальности изображений ImageNet полезно для анализа медицинских изображений?

В качестве примера рассмотрим работу [10] по переносу обучения с использованием классификатора на основе набора данных ImageNet для 2D-классификации медицинских изображений (например, КТ или изображений сетчатки). Для анализа используются два набора данных: срезы КТ грудной клетки размером 224x224 (измененного размера), которые используются для диагностики 5 различных патологий грудной клетки: ателектаз, кардиомегалия, консолидация, отек и плевральный выпот, и набор данных RETINA, который состоит из фотографий глазного дна сетчатки, которые представляют собой изображения задней части глаза.

Основной вывод авторов заключается в том, что трансферное обучение в первую очередь актуально при использовании больших исходных моделей. Модели меньшего размера не демонстрируют такого прироста производительности. Возможно, причина в том, что такие модели, как ResNet и InceptionNet при предварительном обучении используют весовые коэффициенты не на основе случайной инициализации. Кроме того, большие модели меньше меняются при тонкой настройке, особенно в самых нижних слоях.

Для решение проблемы инициализации весов авторы инициализировали весовые коэффициенты двух нижних слоев сети таким образом, что среднее значение и дисперсия матрицы весов вычислялись из предварительно обученных весовых коэффициентов на основе их нормального распределения. Этот расчет был выполнен для каждого слоя отдельно. В результате новая схема инициализации наследует масштабирование предварительно обученных весовых коэффициентов, но забывает сами веса. Веса остальной части сети инициализируется случайным образом и настраивается для выполнения задачи медицинской визуализации. Этот гибридный метод оказывает наибольшее влияние на сходимость. Подводя итог, можно сказать, что большинство наиболее значимых представлений объектов изучаются на двух нижних уровнях.

На Рис.3 показаны параметры модели для описанный в работе метод (Mean Var) в сравнении со случайной инициализацией весов и использование весов из ImageNet [10].

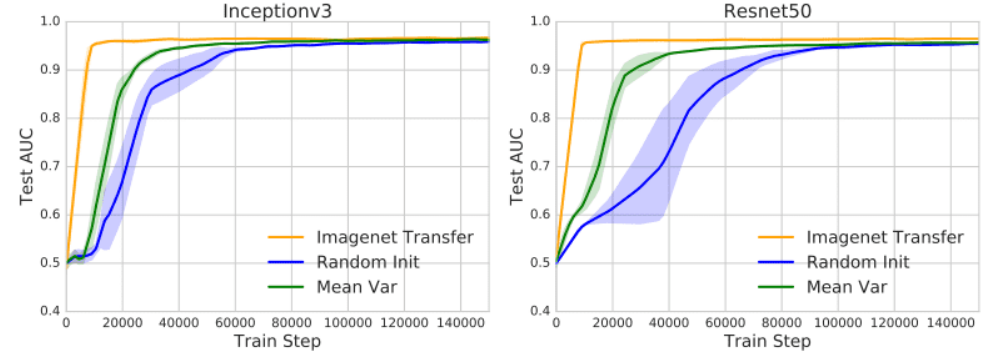


Рис.4 Показатели моделей в зависимости от метода инициализации весов.

В качестве примера анализа данных 3D-медицинской визуализации рассмотрим работу [11] по переносу обучения для 3D-сегментации легких и классификации легочных узелков. Так как перенос весов из ImageNet не так эффективен авторы собрали серию общедоступных наборов данных КТ и МРТ (большую часть данных можно найти на [Medical Image Decathlon](http://medicaldecathlon.com/index.html)) а так же другие данные из разных областей, модальностей, органов-мишеней, патологий и использовали общий набор данных для предварительной тренировки. Таки образом для работы вместо мультимодального набора данных использовалась только одна модальность общего набора. В качестве энкодера авторы используют семейство моделей 3D-ResNet. Декодер состоит из транспонированных сверток для повышения дискретизации объекта за счет изменения размерности карты сегментации. Для работы с несколькими наборами данных использовались разные декодеры. Данная архитектура называется Med3D.

Авторы сравнили предварительную тренировку модели медицинской визуализации с разными весовыми коэффициентами: Med3D, Train From Scratch (TFS, обучение с нуля),  [Kinetics](https://deepmind.com/research/open-source/kinetics) (из набора видеоданных для распознавания действий). Во всех случаях был предварительно обучено только кодировщик. Результаты эксперимента показаны на рис.4. Видно, что архитектура Med3D в независимости от сети дает лучший вариант как в задачах сегментации (seg), так и в задачах классификации легочных узелков. Так же видим, что чем больше слоев в нейросети, тем лучше показатели переноса обучения.

Рассмотрим пример полуконтролируемого трансферного обучения [12]. Под переносом обучения в данном случае понимается передача знаний от исходной модели к целевой.  Важной концепцией является псевдомаркировка, при которой обученная модель прогнозирует метки на основе неразмеченных данных. Сгенерированные метки (псевдометки) затем используются для дальнейшего обучения. Модель обучается на небольшом помеченном наборе исходных данных, чтобы потом присвоить псевдометки для большого неразмеченного набора данных. Таким образом целевая сеть обучается как на размеченных, так и на псевдоразмеченных данных, итеративно пытаясь

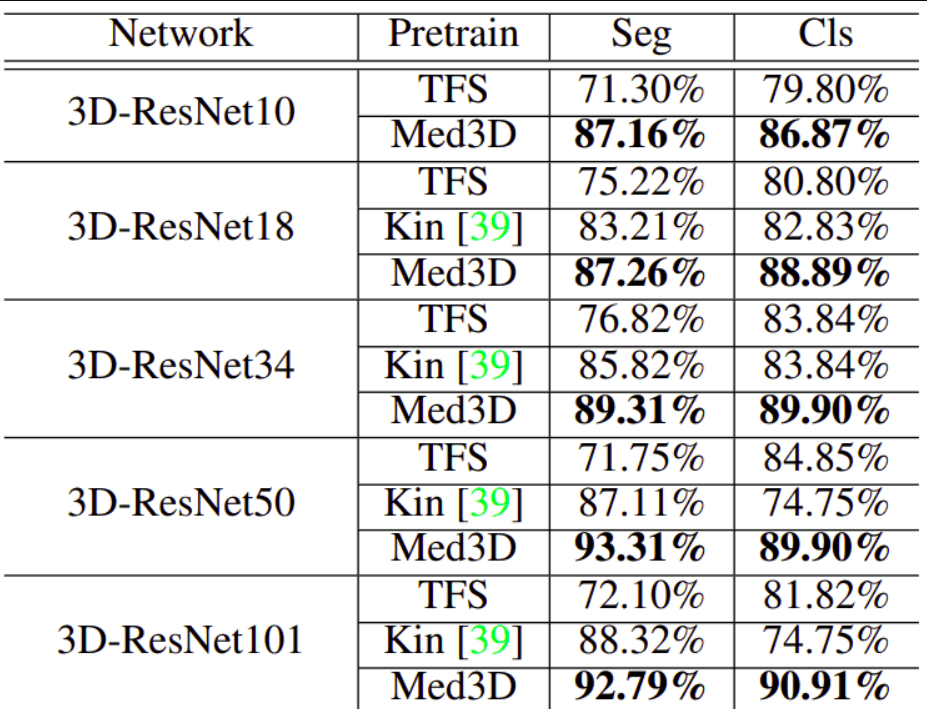


Рис.4.Параметры моделей в зависимости от типа предварительной тренировки [11].

улучшить псевдометки. Распространенной практикой является добавление шума к целевой модели для лучшей производительности во время обучения (любое дополнение данных: поворот, перемещение, обрезка). После обучения целевая сеть используется чтобы снова псевдопометить все неразмеченные данные. Производительность модели зависит от сходства между исходной и целевой областью. Чем больше сходство доменов, тем выше производительность, и лучшие результаты достигаются при предварительном обучении исходной сети в предметной области, близкой к целевой области.

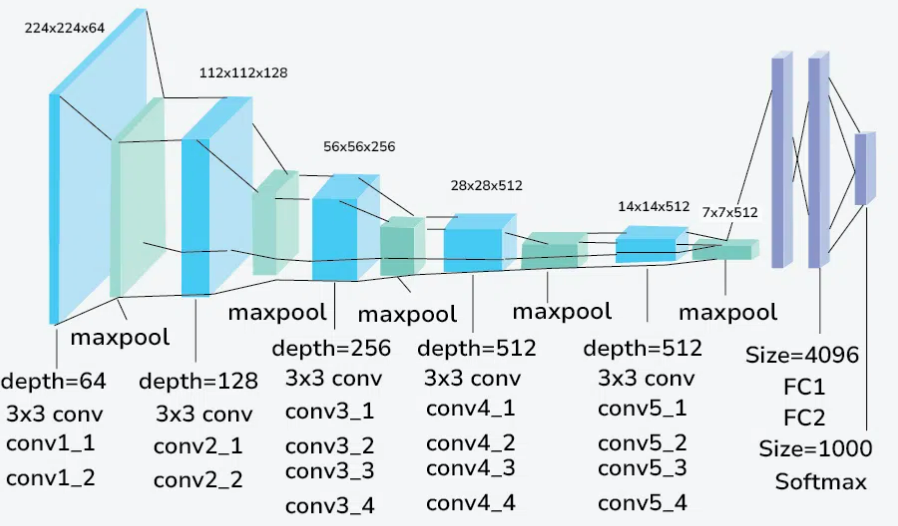
Авторы опробовали аналогичный подход на небольших цифровых изображениях гистологии тканей. Ткань окрашивается, чтобы подчеркнуть признаки, имеющие диагностическое значение. В конце обучения целевая сеть, как правило, превосходила исходную и становилась следующим учителем.

1. **Обзор используемых нейросетей.**
   1. **Нейросеть VGG19**

Основные параметры сети - размер: 549 MB, ошибка Top-1: 71.3%, ошибка Top-5: 90.0%, количество параметров модели: 143,667,240, общее количество слоев: 25.

Модели VGG были представлены Карен Симонян и Эндрю Зиссерманом в 2014 г. Основной целью было исследовать влияние увеличения глубины сверточной сети на крупномасштабные задачи распознавания изображений. По сути, Deep Convolutional Network VGG это углубление сети AlexNet которая появилась в 2012. И после появления VGG16 и VGG19 (с 16 и 19 весовыми слоями соответственно) были одними из наиболее заметных моделей на тот момент. Их дизайн характеризовался последовательным использованием небольших фильтров свертки 3 × 3 на всех уровнях, что упростило структуру сети и повысило производительность.

VGG19 это глубокая сверточная нейронная сеть с 19 весовыми уровнями, включающая 16 сверточных слоев и 3 полностью связанных слоя. Архитектура следует прямолинейному и повторяющемуся шаблону, что упрощает ее понимание и реализацию. Ключевыми компонентами архитектуры VGG19 являются: cверточные слои (сonvolutional layers): фильтры 3 × 3 с шагом 1 и отступом 1 для сохранения пространственного разрешения, функция активации (activation function) ReLU применяется после каждого сверточного слоя для придания нелинейности, объединяющие слои (pooling layers) используются для уменьшения пространственных размеров, три полносвязных слоя в конце сети для классификации и последний слой softmax для вывода вероятностей классов (Рис.7).

При проектировании архитектуры VGG19 следовали нескольким ключевым принципам: единообразные фильтры свертки 3 × 3 упрощают архитектуру, более глубокая архитектура позволяет изучать более сложные функции, функция активация ReLU вводит введение нелинейность и помогает в изучении сложных шаблонов, уменьшение пространственных размеров при сохранении важных функций, объединение изученных функции для классификации на выходе. ****Рис.7. Архитектура сети VGG19.

Для обучения сети VGG19 использовалось более чем 1 миллиона изображений из базы данных ImageNet. Для процесса transfer learning можно импортировать модель с обученными весами от ImageNet. В качестве входных данных для этой сети используются RGB-изображение фиксированного размера (224 \* 224), что означает, что матрица имеет форму (224,224,3), а на выходе — тензор признака формы (7,7,512). В последствии VGG19 проиграла сети GoogLeNet в задаче классификации, но победила в задаче локализации изображений. Архитектура VGG19 служит основой для многих современных нейронных сетей, однако, из недостатков сети можно отметить очень долгое обучение из-за проблемы затухания градиентов.

Сеть VGG19 широко используется в переносе обучения, так как поддерживает надежное извлечение признаков: глубина модели VGG19

позволяет ей улавливать сложные детали на изображениях. Чтобы повысить производительность и возможности обобщения VGG19, во время обучения

часто используются методы увеличения данных, такие как случайное кадрирование, горизонтальное переворачивание и дрожание цвета.

* 1. **Нейросеть ResNet50**

Основные параметры сети - размер: 98 MB, ошибка Top-1: 74.9%, ошибка Top-5: 92.5%, количество параметров модели: 25,636,712, общее количество слоев: 176. Несмотря на то, что сеть ResNet50 имеет меньше параметров, она показывает более высокие результаты, чем VGG19.

[Residual Networks](https://arxiv.org/pdf/1512.03385v1.pdf), сокращённо ResNet (остаточные сети) появились в конце 2015 года и победили в конкурсе Large Scale Visual Recognition Challenge (ILSVRC) в 2015 году в ImageNet 2015. Их изобретатели команда из азиатского подразделения Microsoft Research, которые смогли построить и успешно обучать сети очень большой глубины на тот момент. Сейчас ResNet представляет собой семейство сетей, которые содержат от 18 до 152 слоев. Я буду использовать сеть ResNet50, обученную на наборе данных ImageNet представляющую собой сверточную (CNN) нейронная сеть глубиной 50 слоев. Преимущества ResNet50: мощность и быстрота, относительная простота в использовании, легкая реализация. Самое главное, что она хорошо себя зарекомендовала в извлечении признаков объекта из изображений в широком спектре задач классификации.

В отличии от сетей с архитектурой, подобной VGG19, архитектурах ResNet используется остаточное обучение, которое включает в себя пропуск соединений или быстрых соединений, которые обходят один или несколько уровней. Эти пропускные соединения позволяют сети изучать остаточные отображения, упрощая обучение очень глубоких сетей. Остаточное обучение решает проблему исчезающих градиентов и позволяет обучать более глубокие архитектуры. Соответственно, основной отличительный элемент ResNet это residual-блок (остаточный блок) с shortcut-соединением, через которое данные могут при необходимости проходить без изменений (Рис.5). Res-блок представляет собой несколько свёрточных слоёв с активациями, которые преобразуют входной сигнал х в выходной f(x), а shortcut -соединение представляет собой тождественное преобразование x -> x. В результате такой конструкции Res-блока, входной сигнал может проходить через блок двумя путями, и если на некотором слое сеть уже достаточно хорошо аппроксимировала исходную функцию, то на дальнейших слоях оптимизатор может в Res-блоках делать веса близкими к нулю, и сигнал будет почти без изменений проходить по shortcut-соединению. В некотором смысле можно сказать, что остаточная сеть сама определяет свою глубину, что позволяет обойти проблему деградации сети при увеличении ее глубины.

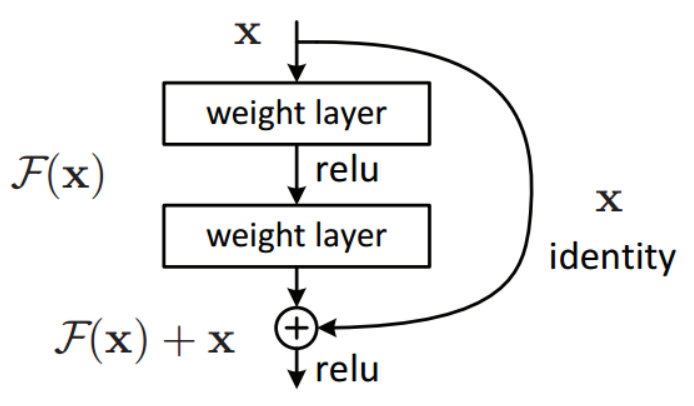


Рис. 5. Схема остаточного блока.

Самый главный вопрос, от решения которого будет зависеть производительность сети, заключается в строении Res-блока. На Рис.6 слева можно увидеть первоначально строение Res-блока, а справа обновленное, которое предложили в 2016 году те же авторы (Рис.6). Они предложили внести последнюю функцию активации (ReLU) в Res-блок и переместить слои нормализации (batch normalization) и активации перед свёрточными слоями. Такая конструкция препятствует затуханию градиента и позволяет сигналу без изменений протекать от одного Res-блока к другому, что позволило авторам успешно обучили ResNet с 1001 слоем.

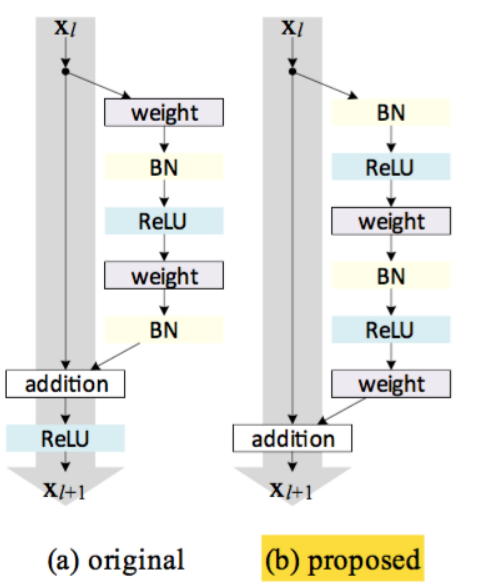


Рис.6 Строение Res-блока, новый справа.

Все предобученные модели семейства ResNet ожидают, что входные изображения будут нормализованы одинаковым образом, то есть в пакеты 3-канальных изображений RGB размерности `(3 x H x W)`, где высота и ширина изображения равна минимум 224. Значения пикселей должны быть переведены в диапазон, `[0, 1]` а затем нормализованы с помощью `mean = [0.485, 0.456, 0.406]` и `std = [0.229, 0.224, 0.225]`. Я буду использовать в работе обе сети (VGG19 и

ResNet50) для определения их возможностей в процессе классификации изображений клеток крови по типам и определения точности этой классификации.

1. **Классификация типов клеток крови c использованием**

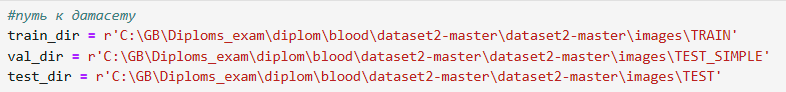
**методов transfer learning**

**3.1. Импорт и описание исследуемых данных.**

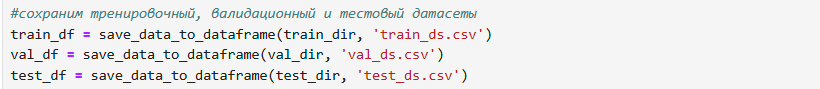
Для поставленной задачи я взяла датасет Blood Cells Images с сайта kaggle.com: <https://www.kaggle.com/datasets/paultimothymooney/blood-cells>

Я использовала набор изображений из папки dataset2-master, который был уже дополнен (аугментирован). Он содержит 12 500 изображений клеток крови в формате JPEG следующих типов клеток: эозинофилы, лимфоциты, моноциты и нейтрофилы. Набор изначально разделен на три выборки: папка TRAIN – выборка для тренировки модели, папку SIMPLE\_TRAIN я попробую использовать для валидации данных, а папка TEST – данные для тестирования модели. В папке тренировочного наборы существует около 3000 изображений для каждого из 4 различных типов ячеек, сгруппированных в 4 разных папки (в соответствии с типом ячейки). Данный датасет обладаем следующими преимуществами для исследования: он имеет достаточный объем выборки, все изображения одного размера и размечены по типам клеток, в изображениях практически отсутствуют дефекты и посторонние шумы.

Загрузим данные в Jupiter notebook, предварительно скачав их на локальный компьютер:



Для более наглядного представления данных сохраним в датафреймы пандас информацию с именем и путем до файла и его класс. Класс я присваиваю исходя из названия папки с конкретным изображением и записываю в колонку ‘labels’.

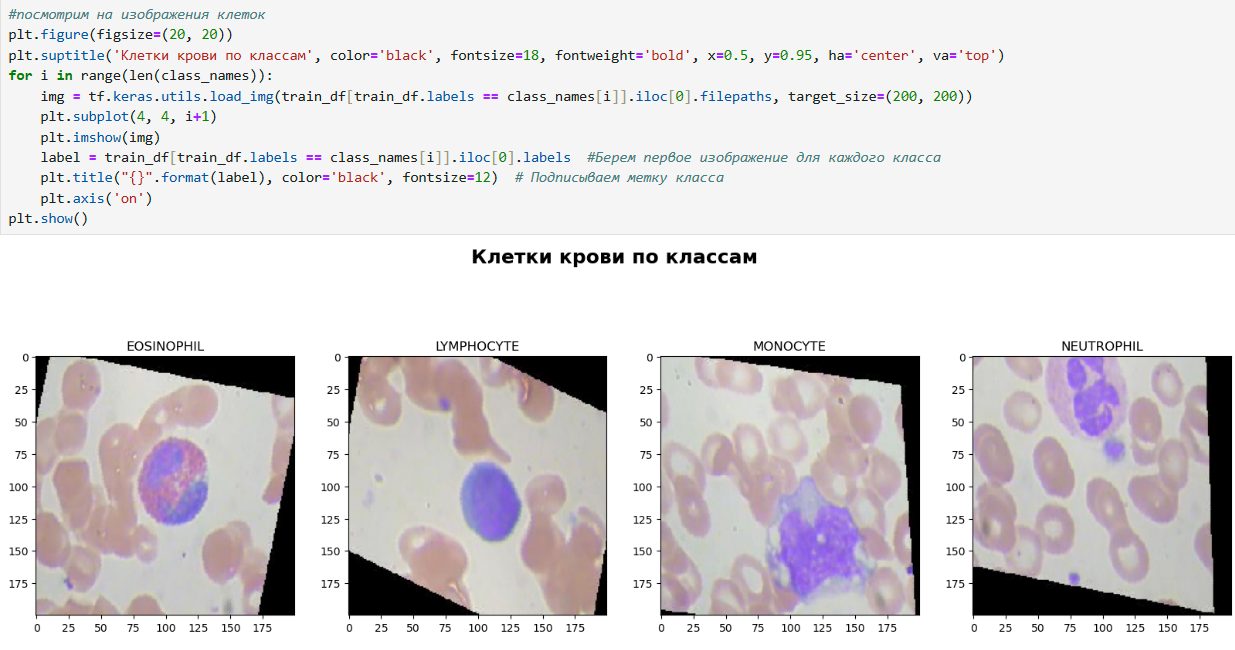


Функция записи данных в датафрейм с последующим сохранение в файл имеет

вид:



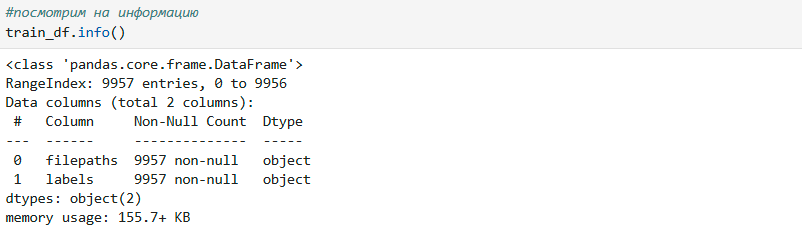
Посмотрим изображения клеток разных классов:



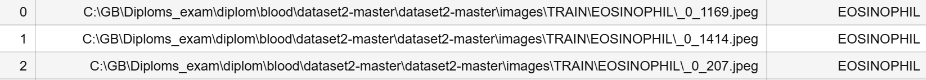
**3.2. Предварительная обработка данных**

Посмотрим первые 5 записей в датасете с тренировочной выборкой и информацию о нем:

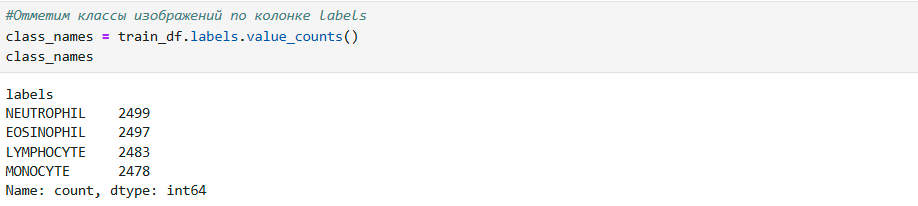




Тренировочный датасет имеет 9957 записей, что соответствует количеству файлов в папке с тренировочными данными. Вот так выглядят данные в сохраненном файле:



Проверим классы колонки labels тренировочного набора данных:

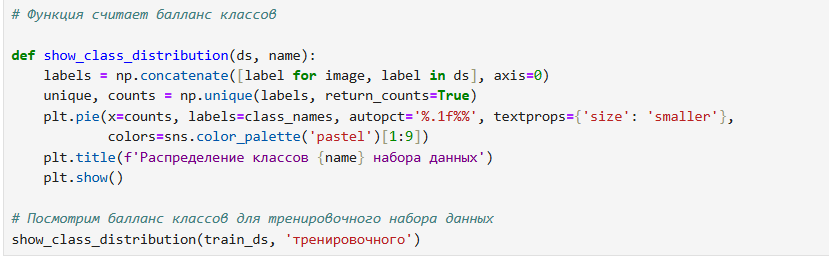


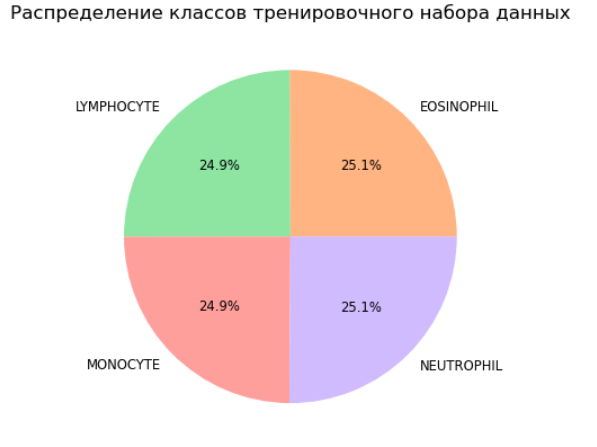
Для работы с моделями я буду использовать фреймворк Tensorflow. При помощи модуля image\_dataset\_from\_directory я загружу тренировочный, валидационный и тестового датасет в виде наборов изображений и меток, которые генерируются исходя из структуры каталогов, то есть будут соответствовать названию папки. Размер батча данных (пакета) я возьму равным 32, размеры изображений 224\*224. Так же проверим метки классов тренировочного датасета.



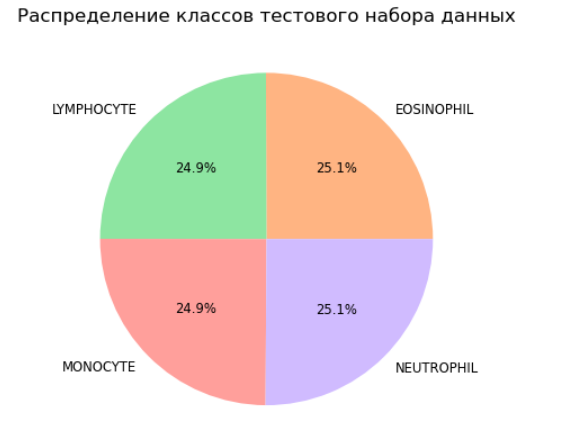


Классы определены правильно, размеры данных 312 батчей тренировочных данных, 78 батчей тестовых и 3 батча валидационных. При помощи функции show\_class\_distribution посмотрим баланс классов во всех наборах.



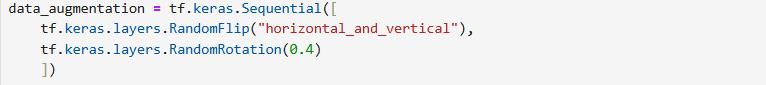


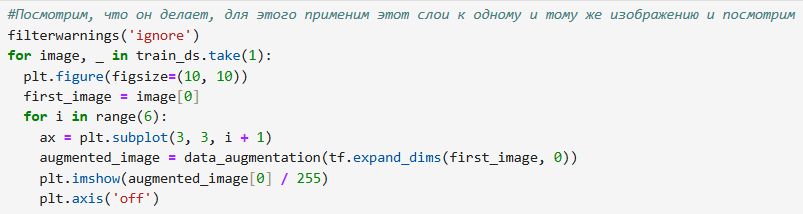




Из диаграмм видно, что тренировочный и тестовый наборы данных имеют сбалансированные классы, в валидационном наборе количество нейтрофилов значительно больше, чем других клеток, например в 11 раз больше, чем моноцитов. Пока оставим валидационный набор таким, в последствие заменим его на сбалансированный набор, и посмотрим, как это отразится на результатах обучения моделей.

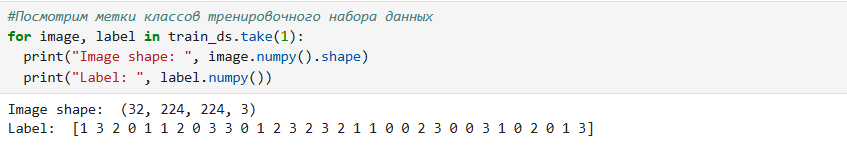
Наш набор данных уже является дополненным (аугментированным), то есть имеет копии изображении, полученные методом геометрических преобразований – поворота, смещения и т.п. Посмотрим, как работает аугментация на примере слоя аугментации (в модели его использовать не будем).







Посмотрим метки классов тренировочного набора данных:



Видим, что они перемешались и могут принимать 4 значения от 0 до 3 соответственно количеству типов клеток крови для распознавания. Первую сеть, которую я буду использовать, это VGG19. Проведем предварительную обработку входных изображений для нее при помощи модуля tensorflow.keras. applications.vgg19.preprocess\_input:



В результате изображения преобразуются из RGB в BGR, затем каждый цветовой канал центрируются по нулю по отношению к набору данных ImageNet, без масштабирования. Форматы наборов данных правильные.

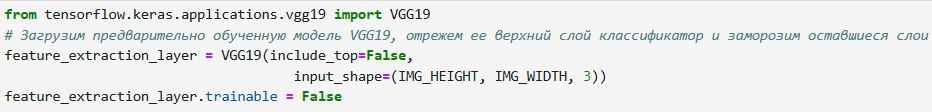
* 1. **Классификация типов клеток крови моделью VGG19**

**методом извлечения признаков.**

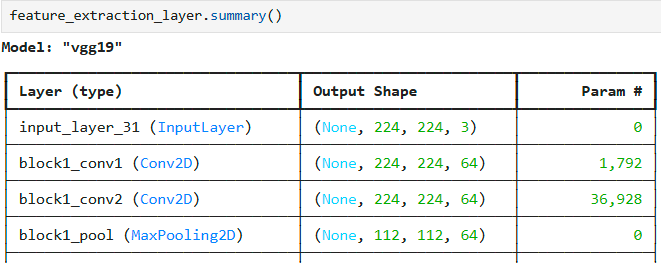
Для использования нейросети VGG 19 ее нужно импортировать из модуля tensorflow.keras.applications.vgg19. Для извлечения признаков изображений при помощи готовой нейросети исследуемый набор данных нужно пропустить через сеть, а полученные выходные данные этой базовой модели передать дальше в

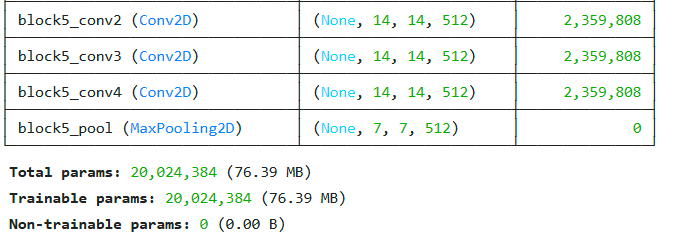
качестве входных данных. При этом не нужно повторно обучать базовую модель,

она уже содержит карты признаков (эмбеддинги) изученных ранее изображений, что позволяет извлекать значимые признаки из набора данных изображений клеток крови.



Я отбрасываю слои классификатора модели VGG19 при помощи параметра include\_top=False, и замораживаю все предыдущие слои сети, установив параметр слоем сети feature\_extraction\_layer.trainable = False. Посмотрим, что мы имеем на входе и выходе feature\_extraction\_layer (все слои можно посмотреть в файле с кодом, ссылка внизу работы):

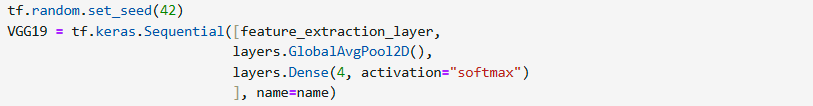




Всего модель имеет 20 024 384 параметров, но ни один из них обучаться не будет. Добавим к базовой модели новый классификатор, который будет обучаться с

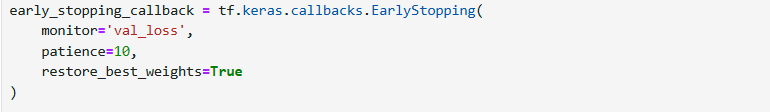
нуля, используя функционал кeras.layers. Добавим к замороженным слоям слой

GlobalAvgPool2D, который применяет операцию объединения глобальных средних значений (Global Average Pooling - GAP) к пространственным измерениям изображения. На выходе мы получаем 512   нейронов, которые мы присоединяем к выходному полносвязному (Dense) с 4 нейронами по числу классов, которые он должен предсказывать. В нем будет использоваться функция активации softmax и на выходе слой будет давать предсказание класса клетки от 0 до 1, где 1 это стопроцентная вероятность.

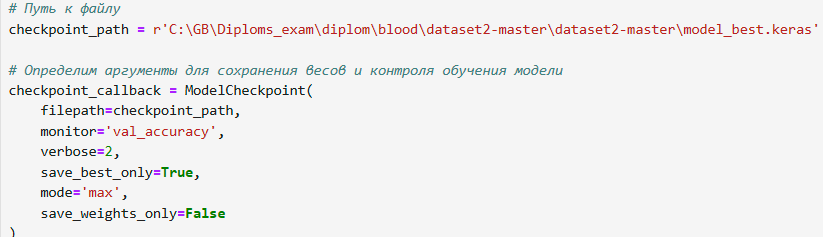




Таким образом у полученной модели есть 2052 тренируемых параметра и 4106 оптимизируемых параметра. Перед запуском обучения модели определим условия остановки обучения, если отслеживаемая метрика (val\_loss – функция потерь на валидационном наборе данных ) перестанет уменьшаться:



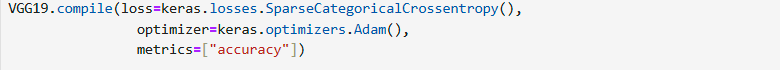
Будем сохранять контрольные точки модели во время обучения используя метод обратного вызова контрольной точки:



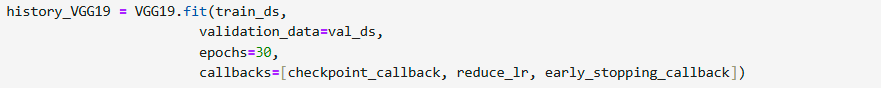
Определим уменьшение скорости обучения, когда метрика val\_accuracy – точность модели на валидационном наборе - данных перестанет увеличиваться:

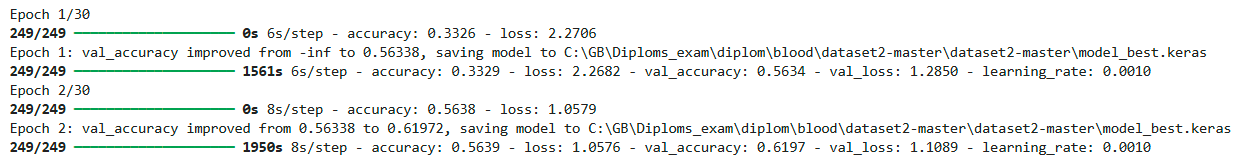


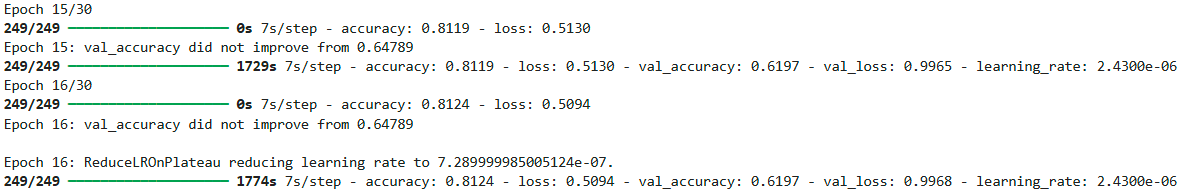
Теперь я могу скомпилировать модель, при этом нужно указать параметры: потери (Loss function) я буду измерять методом кросс-энтропии, оптимизатор, который определяет способ корректировки весов, я возьму Adam, метрикой, которую я буду отслеживать при обучении будет точность (аccuracy). Она вычисляется как отношение числа правильно классифицированных объектов к общему числу объектов и особенно полезна, когда классы в данных равномерно распределены, то есть каждый класс представлен примерно в одинаковом количестве.



Теперь можно запустить процесс обучения модели командой model.fit, указав набор данных для тренировки, валидации, количество эпох обучения, а так же указав определенные раньше обратные вызовы (callbacks):

Посмотрим начало и конец процесса обучения полученной модели (полностью можно посмотреть его в рабочем ноутбуке):





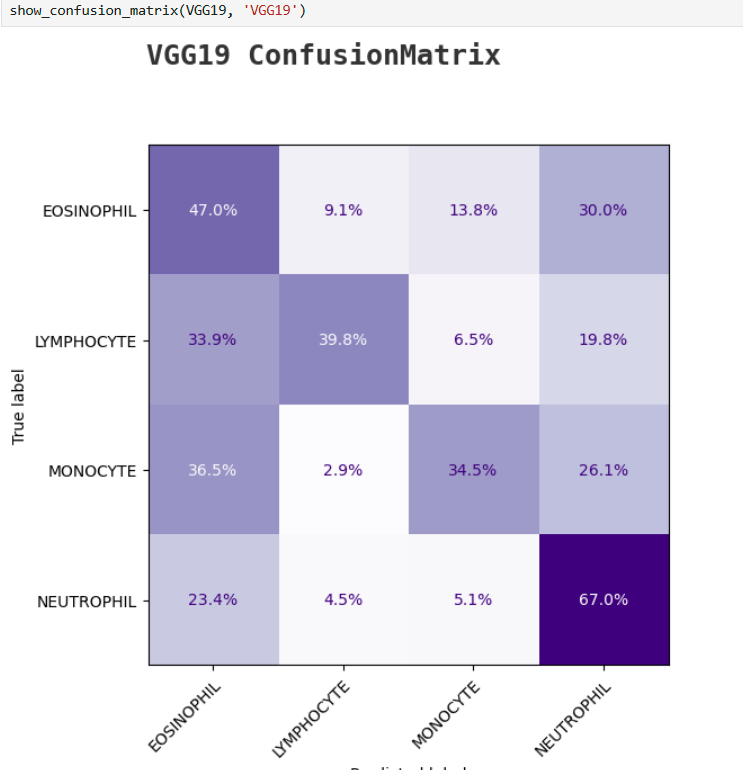
После 16 эпох обучения сработал автоматический выход из обучения, так как функция потерь для валидационных данных перестала значимо уменьшаться. Посмотрим графики точности и функции потерь для тренировочного и валидационного набора данных на графики. Функция для отображения графиков имеет следующий вид:



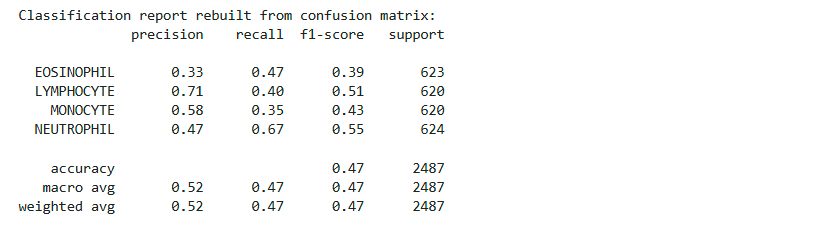
По графикам метрик видно, что метрики точность (accuracy) и функция потерь (loss) для валидационного набора данных, достигнув значений 0,81 и 0,99 соотвественно, практически не изменяется. Скорее всего, это последствия несбалансированности классов валидационного датасета.



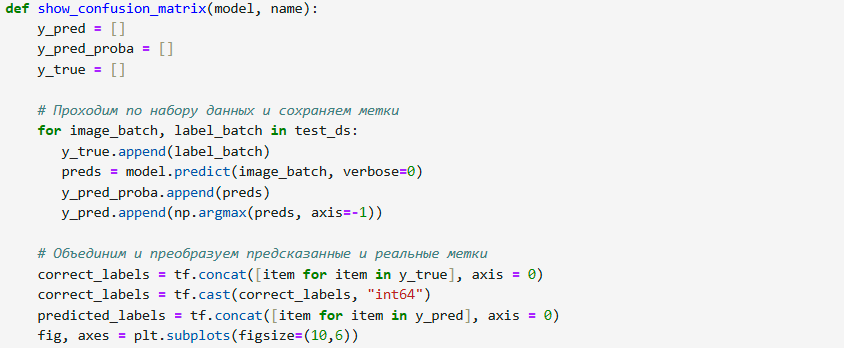
Для анализа посчитаем матрицу ошибок сети на тестовом наборе данных:



Матрица ошибок показывает количество правильных и ошибочных предсказаний для каждого класса. Из матрицы ошибок можно заключить, что лучше всего модель узнает нейтрофилы, что и ожидаемо, так как их в валидационном наборе данных 61%. Так же напечатаем классификационный отчет (сlassification\_report) по итогам предсказания на тестовом датасете:

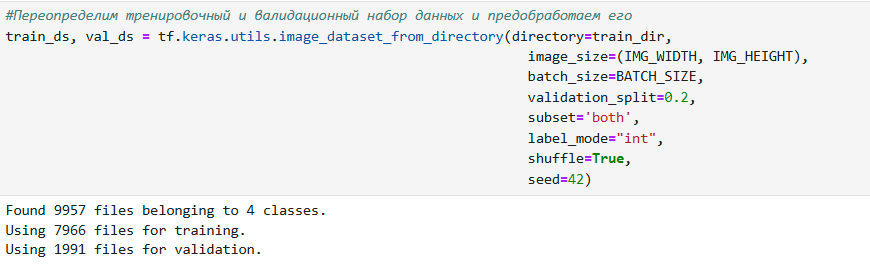


Где точность (precision) можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, полнота (recall) показывает процент правильных положительных прогнозов по отношению к общему количеству фактических положительных результатов, оценка f1 показывает средневзвешенное гармоническое значение точности и полноты, чем ближе этот параметр к 1, тем лучше модель. Поддержка (support) показывает сколько и каких классов было в датасете. Из результатов отчета видно, что точность классификации модели равна 47%, что конечно же, очень мало. Для построения матрицы ошибок я и распечатки отчета я использовала функцию:

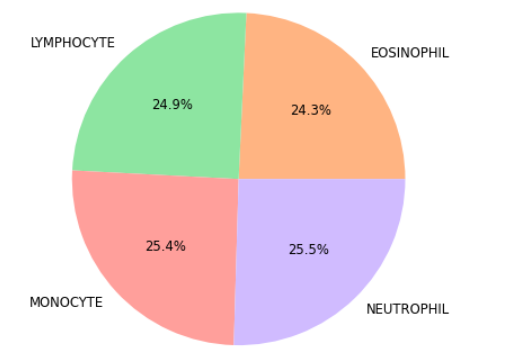
****

****

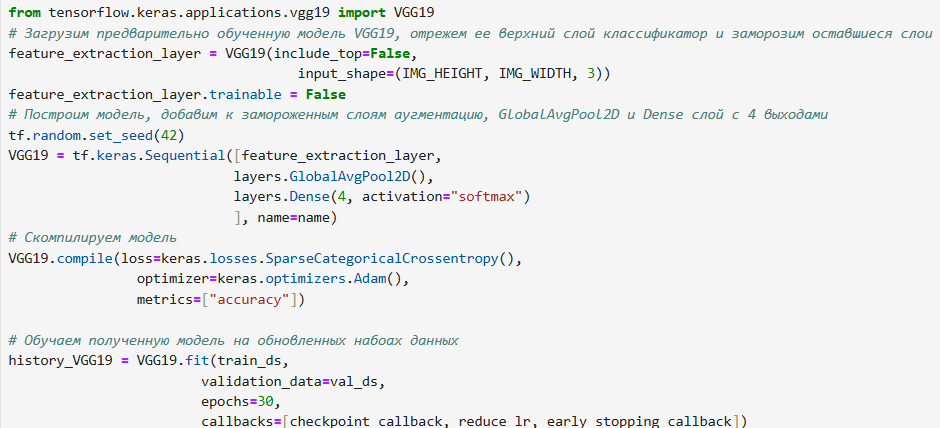
Попробуем увеличить точность модели за счет переопределения валидационного датасета в качестве 20% от тренировочного, чтобы соблюсти баланс классов клеток.



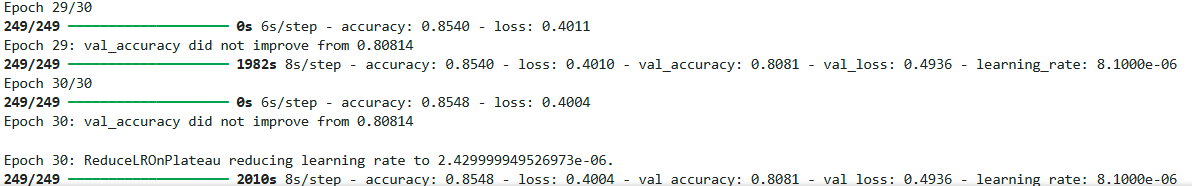
Распределение классов в новом валидационном наборе данных:

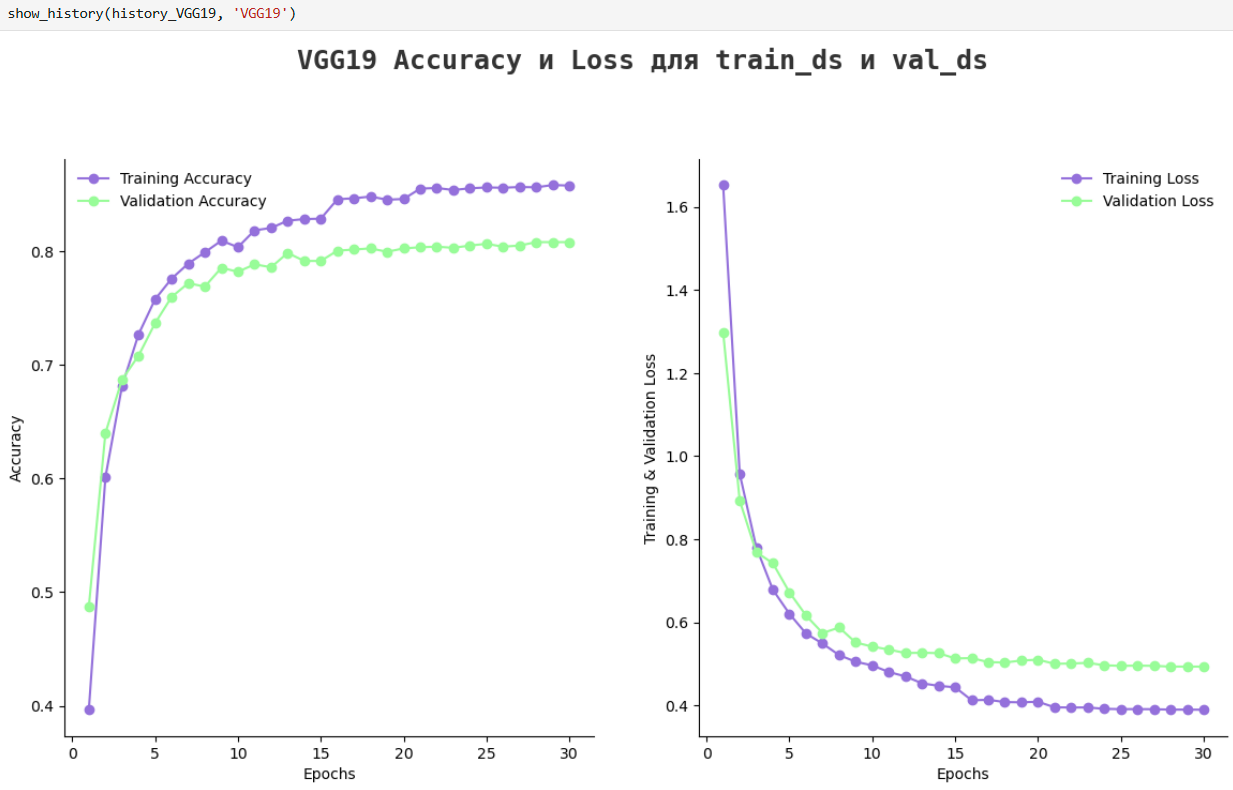


Видим, что теперь классы сбалансированы. Заново загружаем сеть и собираем ее

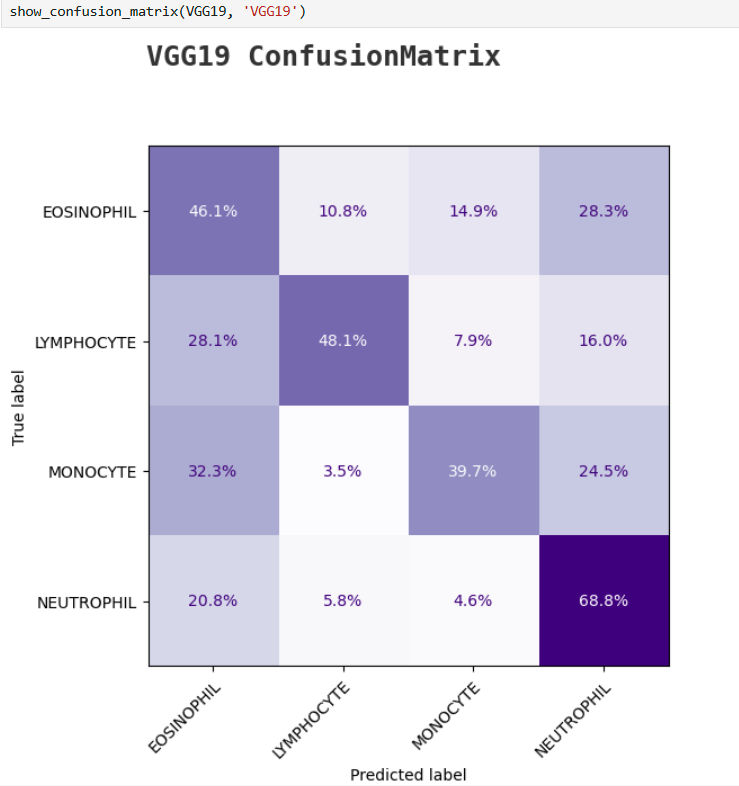


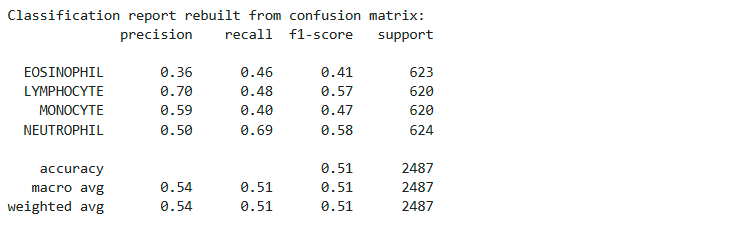
Обучение продолжилось до 30 эпохи с постепенным снижением скорости обучения, улучшение полученных метрик видно на графиках:



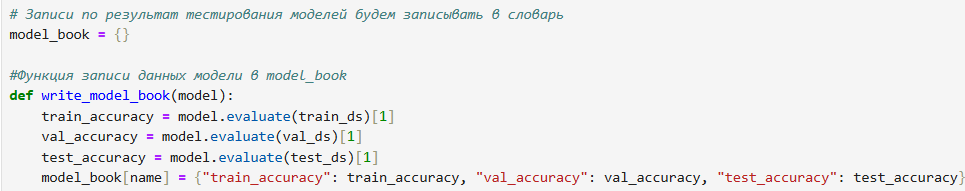


Матрица ошибок:





Теперь точность модели колеблется от 0,36 до 0,7, то есть средняя точность модели улучшилась. Создадим словарь для записи результатов тестирования моделей и будем туда записывать точность модели на всех наборах данных при помощи функции write\_model\_book:



Запишем результат обучения модели на базе замороженной сети VGG19 в словарь:



По результатам видим, что точность модели на всех наборах данных выросла, так

на тестовом датасете она увеличилась с 0,47 до 0,51 (почти 9%) за счет сбалансированности классов валидационного датасета.

* 1. **Классификация типов клеток крови моделью VGG19**

**методом тонкой настройки.**

Метод тонкой настройки предполагает полную разморозку или разморозку несколько верхних слоев базовой модели и дальнейшее совместное обучение и базовой модели и добавленных к ней слоев. Это позволяет нейронной базовой сети лучше определять характерные особенности изображений для задачи классификации изображений по типам клеток крови.

Основным требованием для возможности тонкой настройки является уже обученный классификатор (выходной слой новой нейросети), что я уже сделала для VGG19. И еще особенностью метода тонкой настройки является очень маленькая скорость обучения что бы не переучить модель, а постепенно адаптировать предварительно выработанные в прошлом обучении веса.

Эмпирически выяснено, что эффективность тонкой настройки зависит от того, насколько сильно отличаются наборы данных. Чем больше разница между наборами исходных изображений для предварительного обучения и целевым набором, тем лучше работает настройка. Самый лучший вариант если предобученная сеть не видела объекты целевого набора, как и в моем варианте. Однако, чем больше разница между наборами данных, тем большее количество

слоев базовой модели нужно разморозить. Посмотрим, насколько улучшатся показатели модели на практике.

Сначала необходимо разрешить обучение на уровне сети, для этого поле trainable для feature\_extraction\_layer устанавливается в значение True.

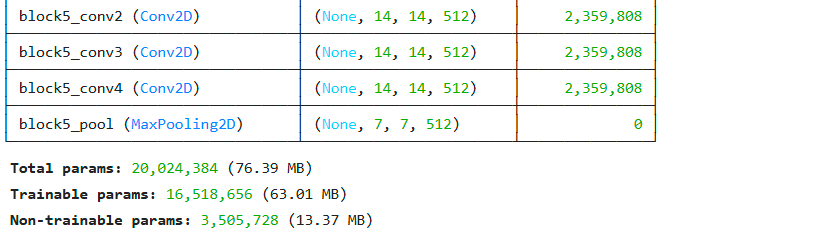


Всего у нас 22 слоя, так как в наборе данных ImageNet нет изображений клеток крови и сеть их не видела, то разморозка большего количества слоев дает больший прирост точности модели. Я остановилась на разморозке слоев 4 и 5 блока базовой модели VGG19, то есть всего я разморозила 10 верхних слоев базовой модели. Поэтому эту модель я назвала VGG19\_unfrozen10. Для заморозки нижних слоев модели будем проходить по ним и замораживать слои от начала до 12 слоя включительно, устанавливая поле trainable в значение False.

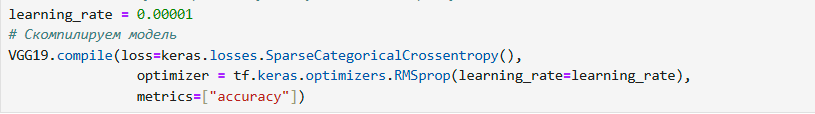


Посмотрим сколько параметров базовой модели обучаются теперь (не привожу все слои, только начало и конец):

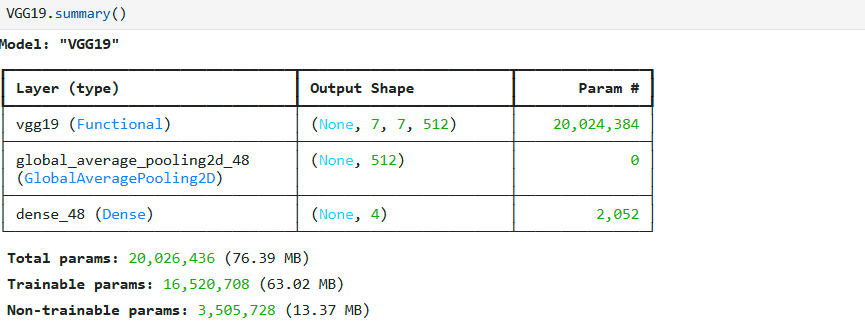




Общее количество обучаемых параметров базовой модели 16 518 656. Теперь нужно следует перекомпилировать целевую модель, чтобы, чтобы изменения вступили в силу, и снизить скорость обучения до 0,00001.



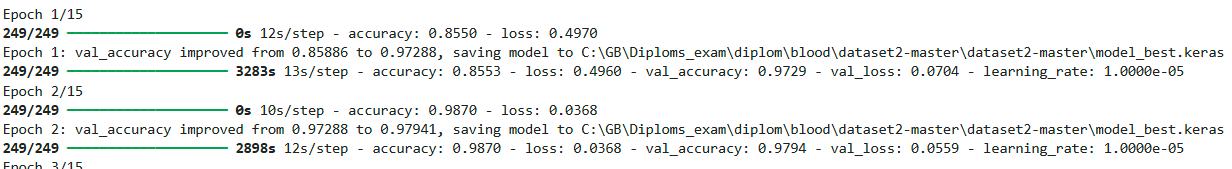
Обновленная модель с частично размороженными слоями выглядит так:

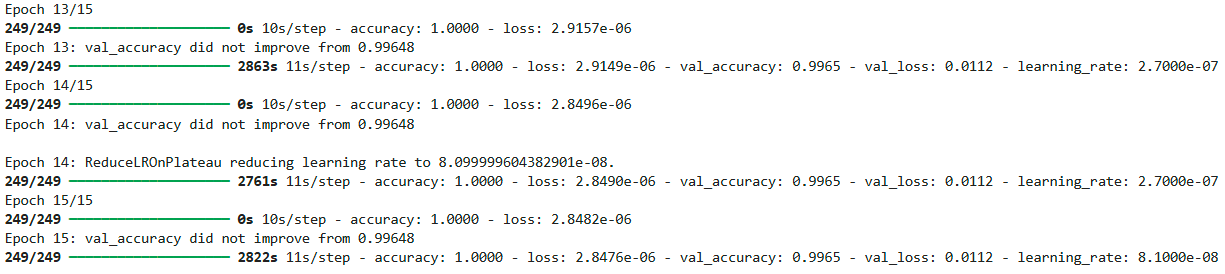


В результате у модели 16 520 708 настраиваемых параметров, что в 8000 раз больше, чем в замороженной модели. Посмотрим, какой это даст нам результат, запустив обучение обновленной модели.

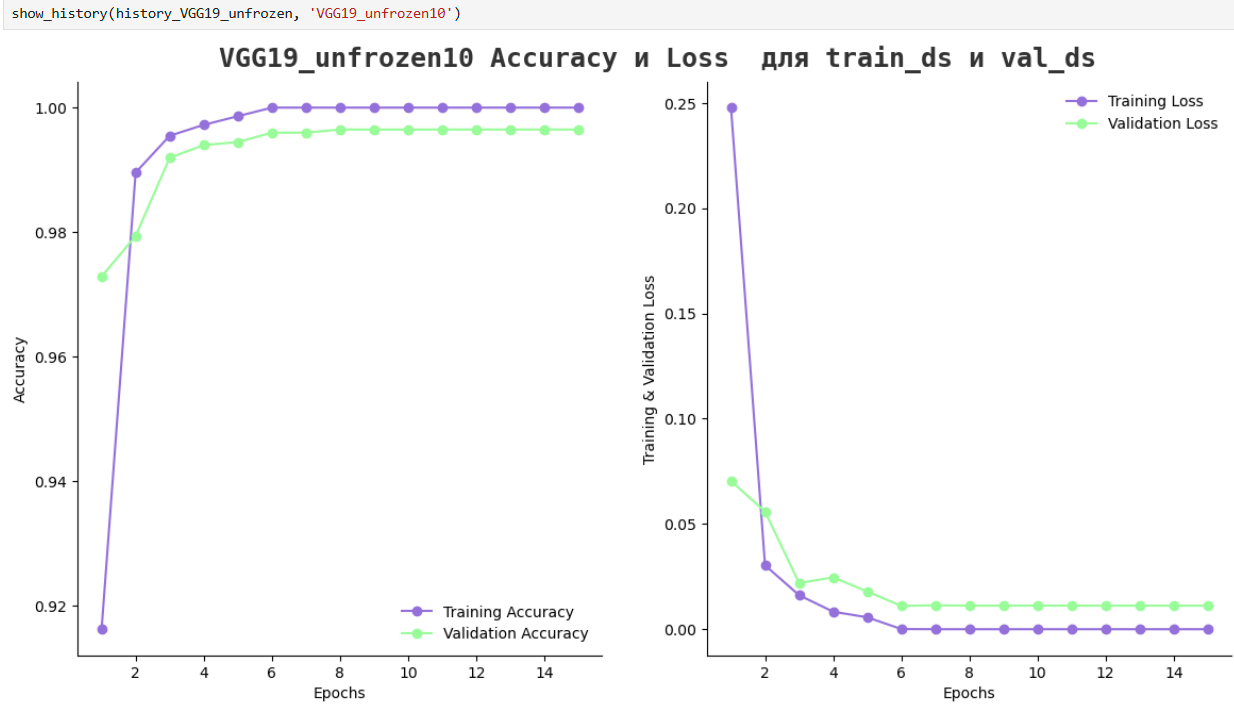


Обучение длилось 15 эпох, причем метрики в конце первой эпохи соответствуют метрикам последней эпохи обучения незамороженной модели. Скорость обучения автоматически постепенно уменьшалась:

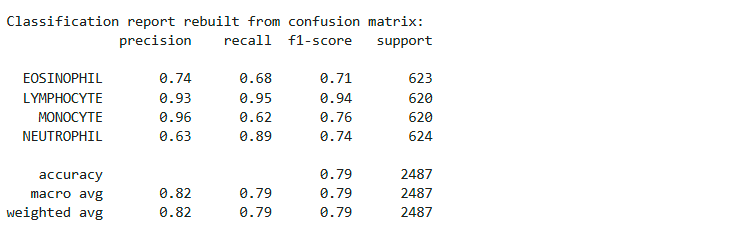




Видно, что метрики значительно улучшились, это хорошо видно на графиках, причем заметно, что метрики точности после 8 эпохи уже не увеличивались, а метрики потерь изменялись очень незначительно:

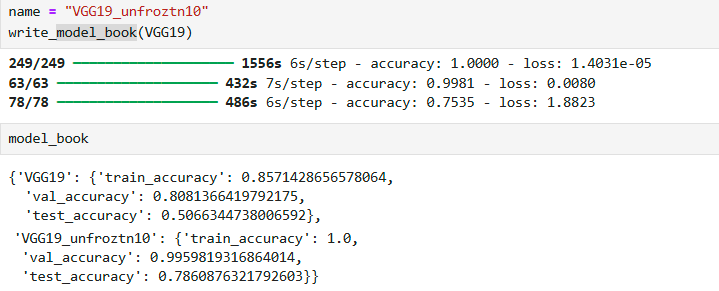


Посмотрим классификационный отчет и матрицу ошибок для настроенной сети VGG19:





Можно сделать вывод, что точность модели теперь составляет от 0,62 для моноцитов до 0,95 для лимфоцитов. Сохраним метрики для всех наборов данных в словарь:



Средняя точность модели на тестовом наборе данных за счет тонкой настройки выросла до 0,79. Для замороженной сети она составляла 0,51, то есть на 55%.

Сохраним веса модели:



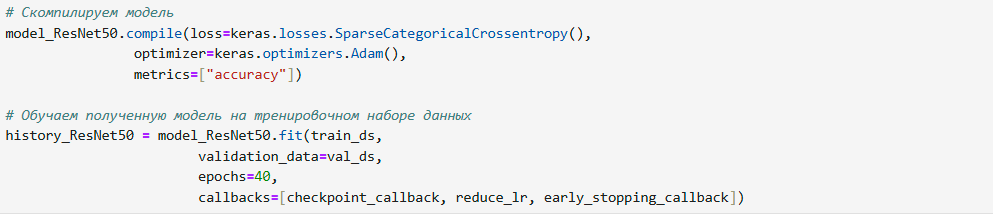
* 1. **Классификация типов клеток крови моделью ResNet50**

**методом извлечения признаков.**

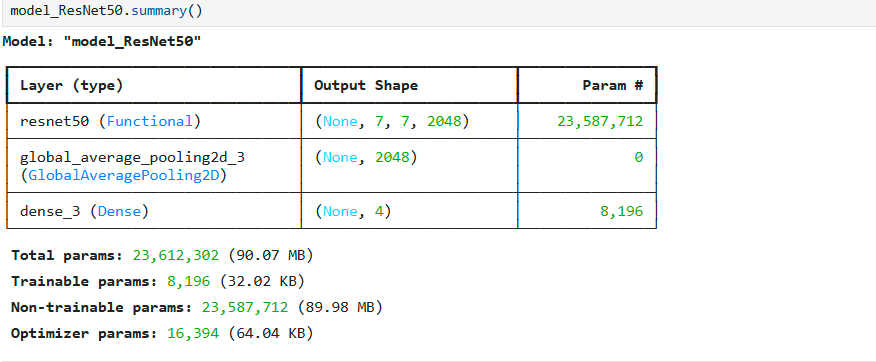
Вторая сеть, которую я выбрала для исследования, ResNet50. Так как она более глубокая, то я ожидаю прирост точности модели. Принцип сборки и использования будет аналогичным принципу для замороженной VGG19. Загружаем модель из tensorflow.keras.applications.resnet50, называем ее model\_ResNet50, отбрасываем классификатор, замораживаем все слои и собираем новый целевой классификатор, аналогичный прошлой модели:

****

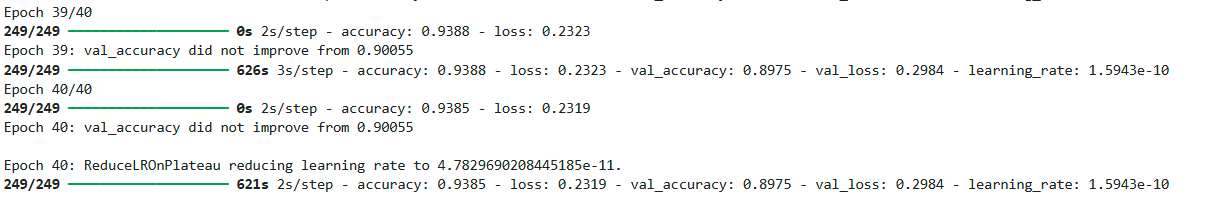
Как было указано выше, модель ResNet50 содержит 50 слоев, я не стала приводить в этой работе полную архитектуру слоев сети, ее можно посмотреть в рабочем ноутбуке. Скомпилируем и запустим обучение модели на базе ResNet50.



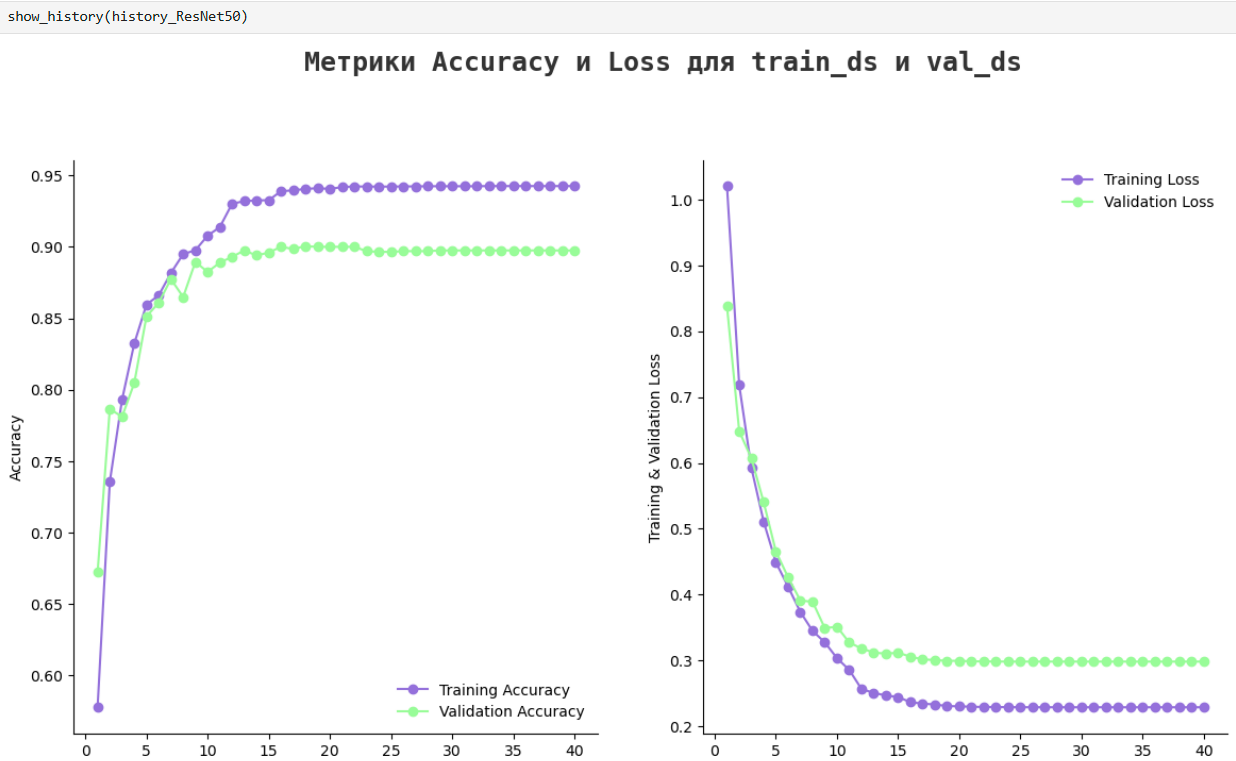
Посмотрим на количество параметров нашей новой собранной модели на основе ResNet50:

****

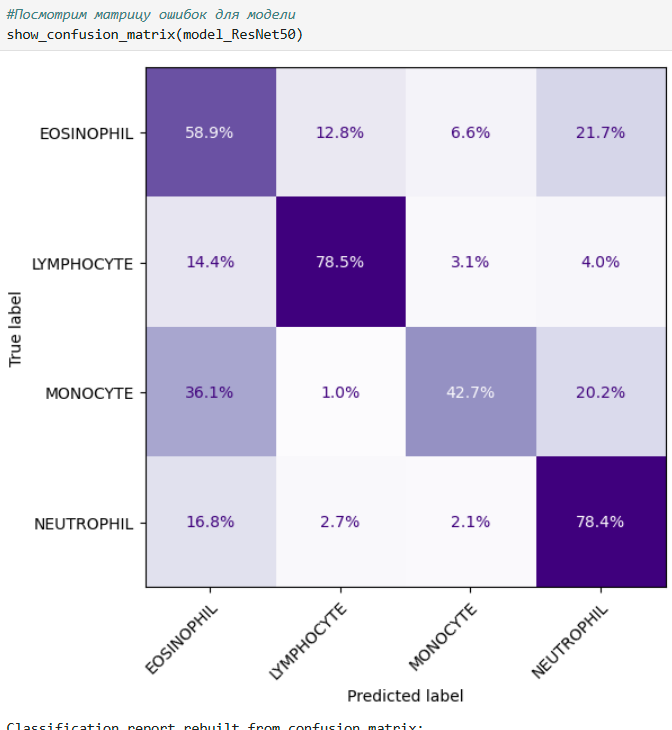
Полученная модель имеет 8 196 тренируемых параметров (почти в 4 раза больше, чем модель на основе сети VGG19), и 16 394 оптимизируемых параметра (тоже почти в 4 раза больше). Посмотрим, насколько эта модель будет точнее после обучения, чем предыдущая. Модель обучалась 40 эпох:

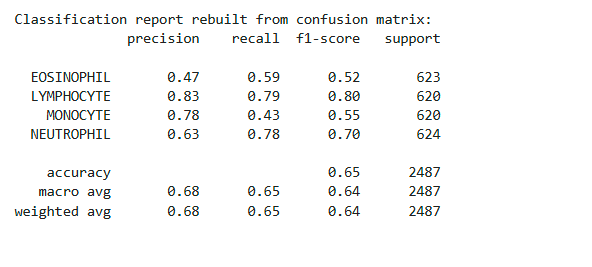


История изменения метрик точности и функции потерь можно увидеть на графике:

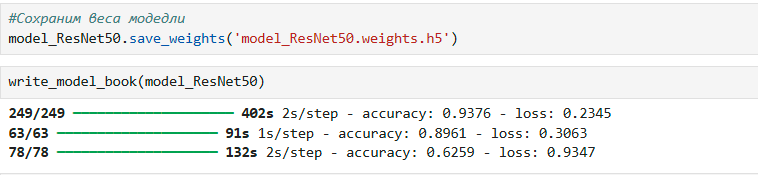


Посмотрим матрицу ошибок для сети:





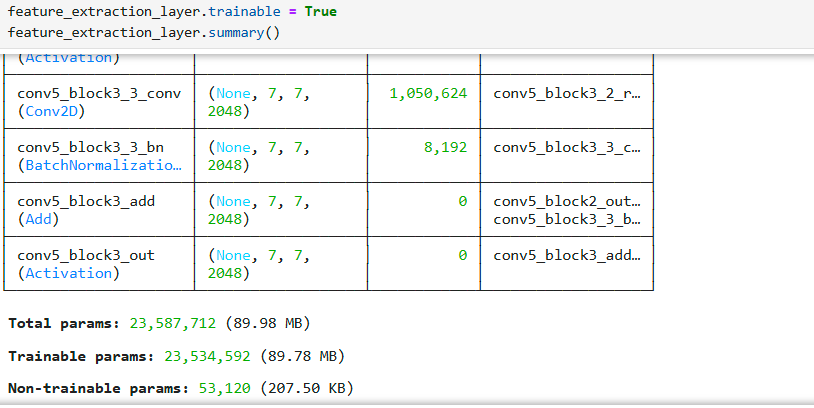
Из матрицы ошибок и классификационного отчета видно, что точность предсказания модели на основе ResNet50 колеблется от 0,47 для эозинофилов до 0,83 для лимфоцитов и в среднем составляет 0,65 на тестовом наборе данных. Этот показатель выше, чем точность модели на основе замороженной VGG19 на 27%. Сохраним веса модели в файл и запишем в базу моделей ее показатели.



* 1. **Классификация типов клеток крови моделью ResNet50**

**методом тонкой настройки.**

Попробуем увеличить точность модели за счет тонкой настройки. Принцип настройки будет такой же, как и для нейросети VGG19. Сначала разрешим обучение всем слоям указав параметр trainable=True и посмотрим, сколько параметров доступно для тренировки:

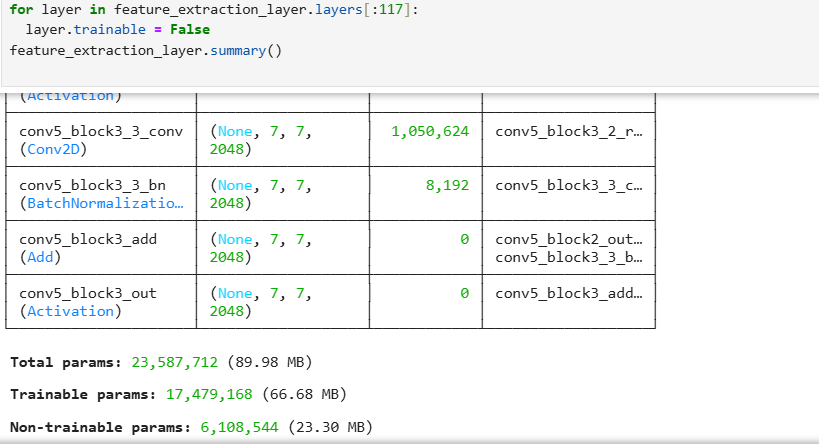


Все слои базовой модели разморозились, посчитаем сколько всего слоев в архитектуре базовой модели:



Как и для VGG19 исходный набор данных, на котором обучалась ResNet50 сильно отличается от целевого, поэтому разморозим треть слоев модели сверху, чтобы количество тренируемых параметров было равно примерно такому же

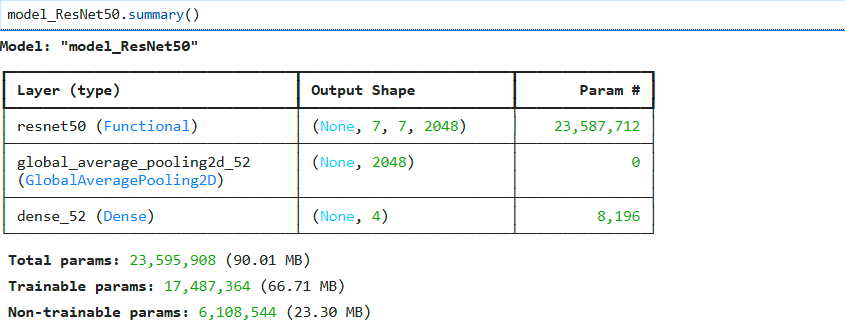
числу, как и при тонкой настройке сети на основе VGG19.



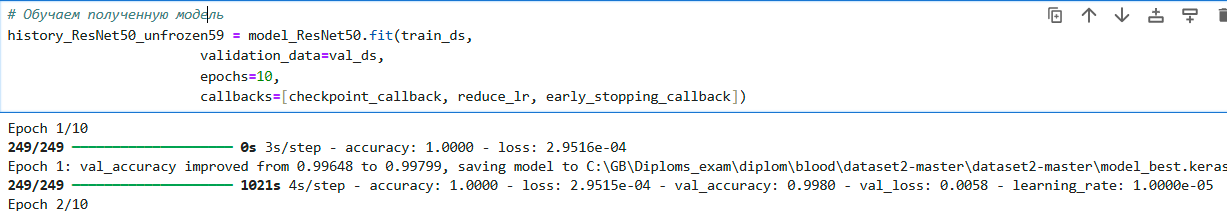
Так как я разморозила 59 слоев, то назвала модель ResNet50\_unfrozen59.

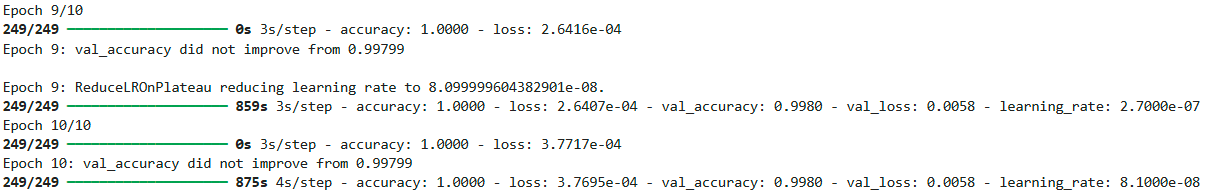
Скомпилируем модель заново, уменьшив скорость обучения, и посмотрим на ее параметры:



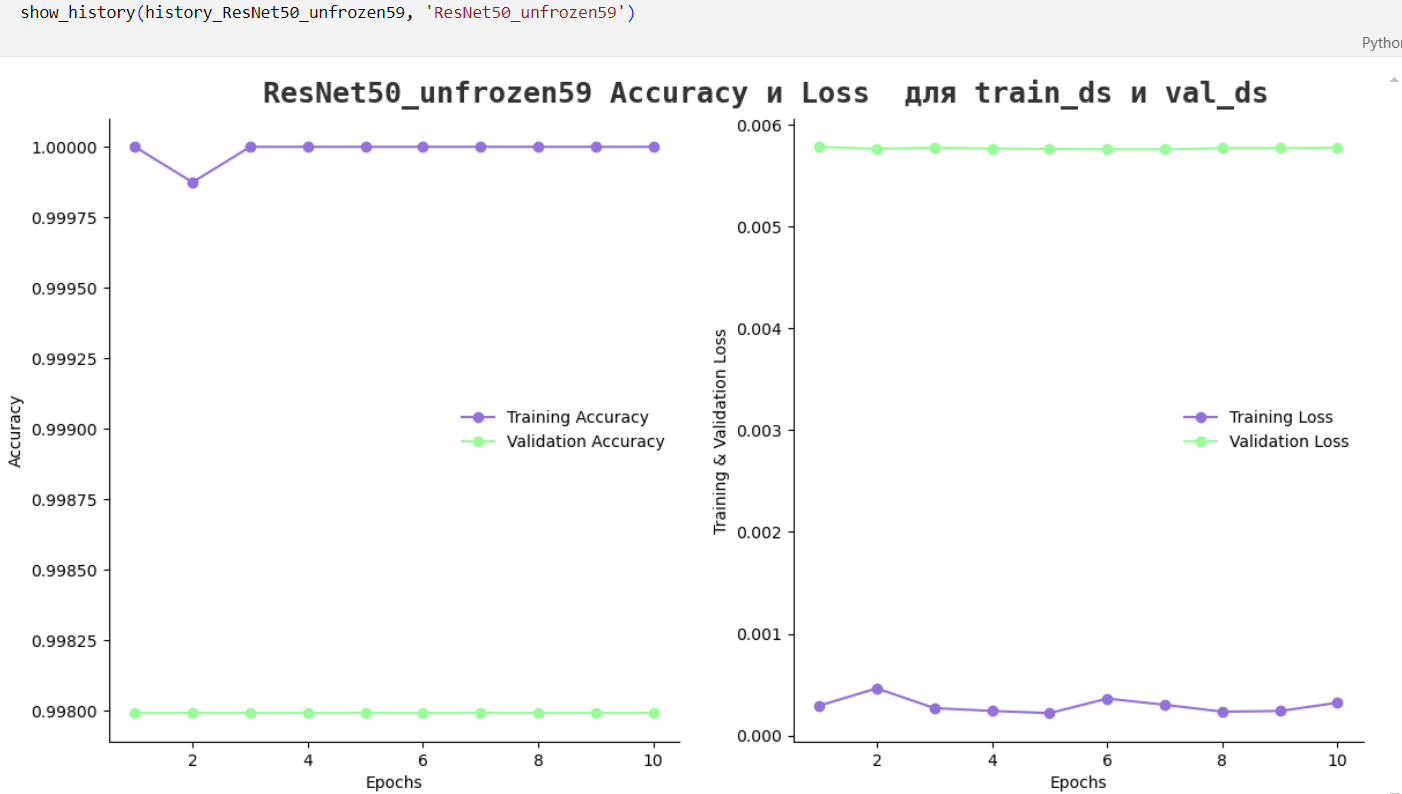


В получившейся модели тренироваться будут 17 487 364 параметра, что сравнимо с аналогичной настройкой сети VGG19. Так как для VGG19 было достаточно 8 эпох для тонкой настройки, а ResNet50 должна обучаться быстрее, запустим ее обучение на 10 эпох.

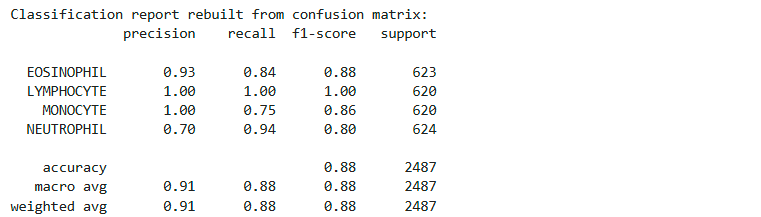


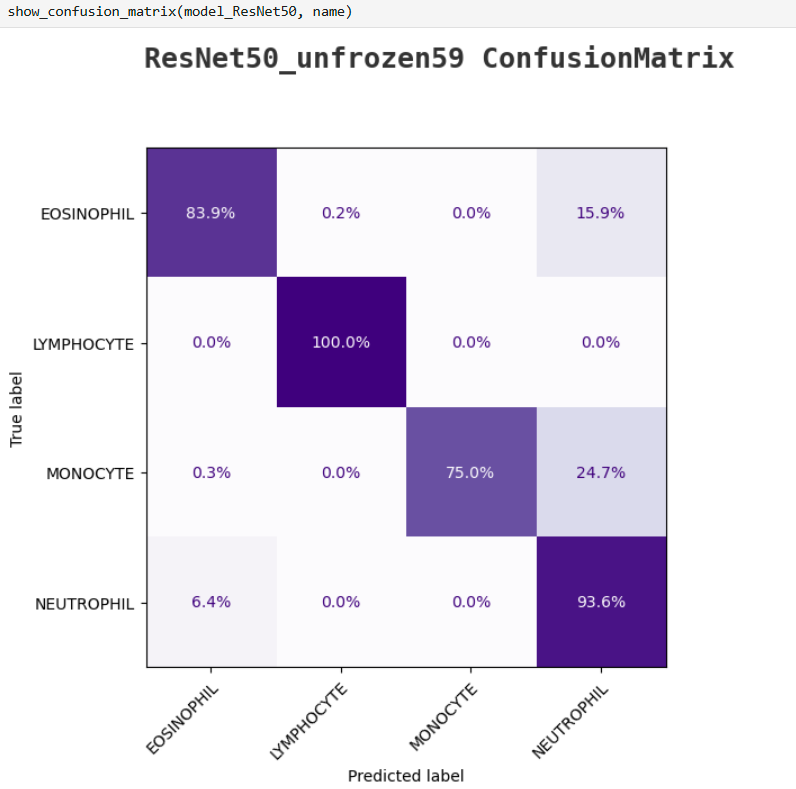


Улучшение метрик произошло в основном на первых эпохах тонкой настройки, скорость стала снижаться с 4 эпохи, сами эпохи длились недолго (в три раза короче чем у аналогичной модели на базе VGG19), и графики метрик представляют собой практически прямые линии, что говорит о том, что дальше обучать эту модель смысла не имеет.

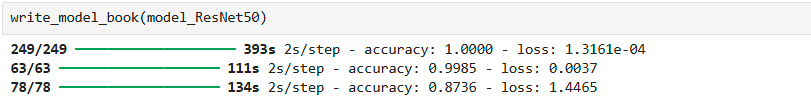


Посмотрим матрицу ошибок и отчет:





Видим, что точность модели после тонкой настройки колеблется от 75% для моноцитов до 100% у лимфоцитов. Точность модели на тестовом наборе данных составила 88% против 65% в модели без тонкой настройки, то есть увеличилась примерно на 35%. Запишем ее параметры в словарь с характеристиками моделей и сохраним веса.



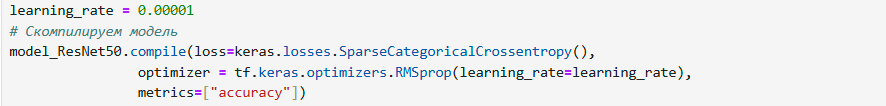


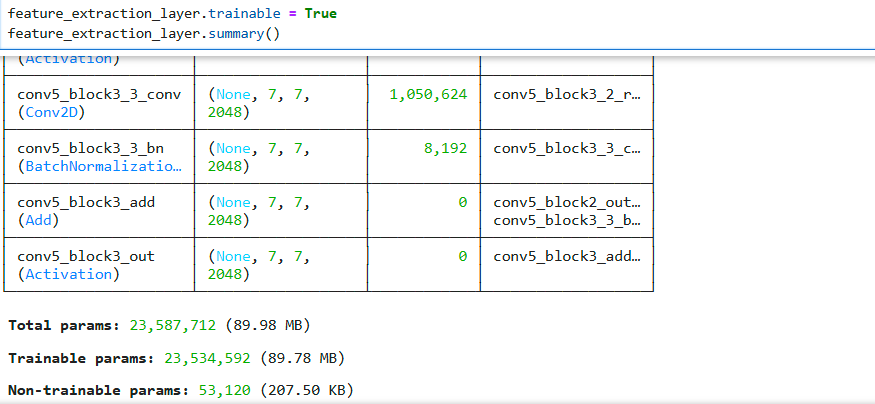
Теперь я попробую полностью разморозить все базовую сеть ResNet50 и посмотрю на результат. Чтобы вернуться к весам модели до начала тонкой настройки, мне пришлось заново собрать базовую сеть и загрузить в нее веса замороженной модели ResNet50:



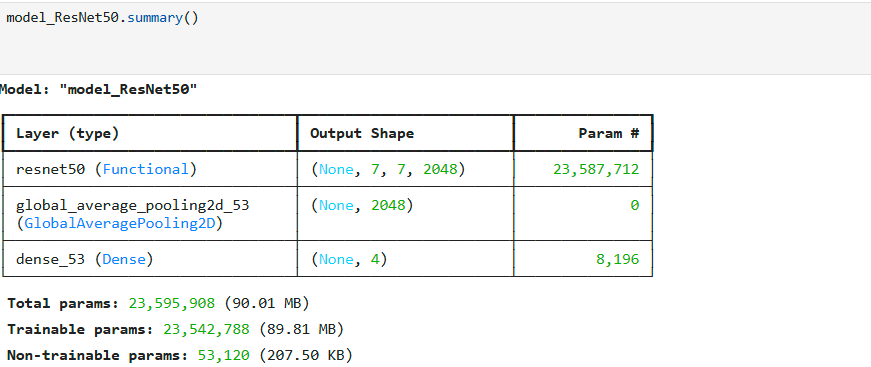
Теперь разрешаем тренироваться всем слоям, проверяю количество параметров у базовой модели – тренируемых теперь 23 534 592. Заново

компилируем модель:





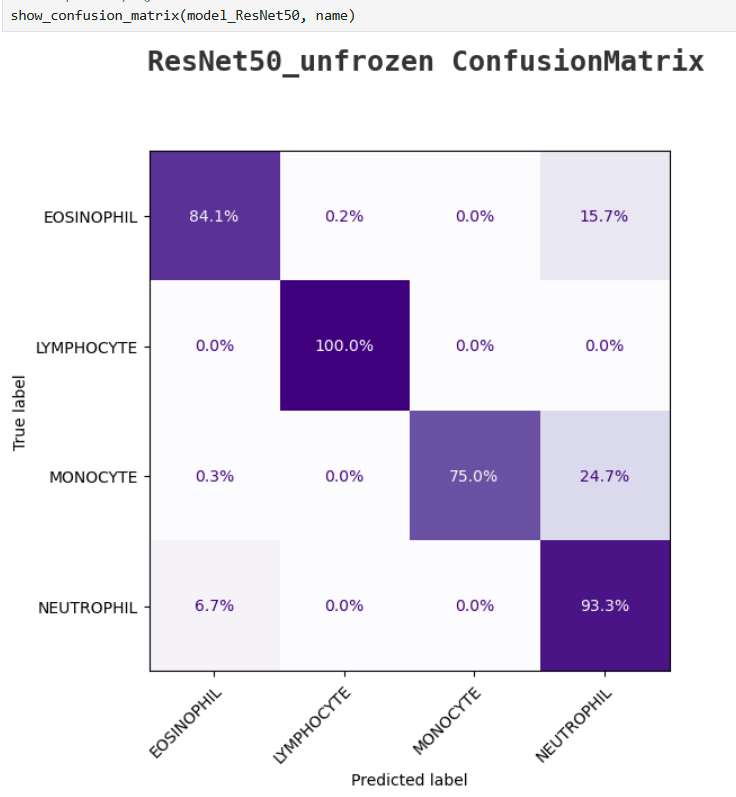
После компиляции модель имеет 23 542 788 обучаемых параметра и выглядит так:

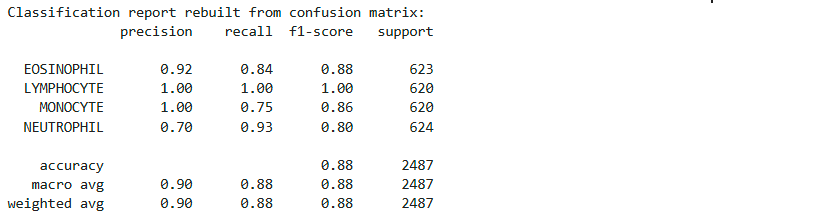


По опыту предыдущей тонкой настройки запустим обучение на 7 эпох, название модели запишем как ResNet50\_unfrozen:

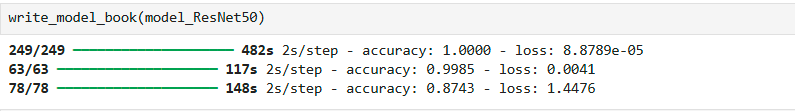


Как и в случае неполной разморозки метрики в течение обучения менялись незначительно, графики метрик так же очень похожи, поэтому приводить записи об обучении и сами графики не буду. Посмотрим матрицу ошибок и отчет.



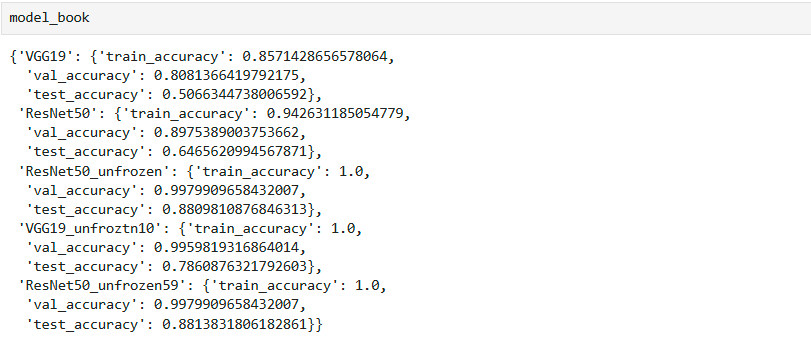


Видим, что точность на тренировочном наборе данных не изменилась не смотря на увеличение тренируемы параметров. Также как и в прошлых вариантах сетей лучше всего нейросеть распознает лимфоциты и нейтрофилы, хуже всего моноциты. Сохраним веса полученной модели в файл и запишем результаты обучения модели в наш словарь с характеристиками моделей.



****

Посмотрим получившие характеристики моделей:



**Заключение.**

В данной работе я хотела проверить возможность осуществления переноса обучения на базе доступных предобученных глубоких нейросетей с возможностью тренировать минимальное количество параметров для распознавания и классификации клеток крови человека по 4 группам, а так же методы повышения точности их работы. Для исследования в качестве базовых сетей мною были выбраны нейросети VGG19 и ResNet50. Обучение было проведено для сетей методами извлечения признаков (замороженные базовые сети) и методом тонкой настройки (часть слоев базовой сети разморожены). Этапы работы:

VGG19 с замороженными слоями, имя в работе VGG19, количество обучаемых параметров 2 052 (параметры выходного слоя классификатора),

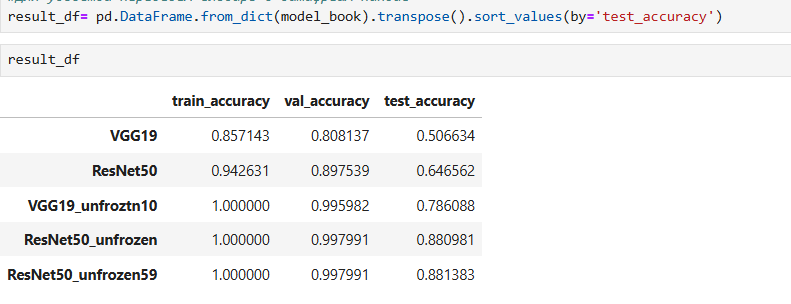
VGG19 с 10 размороженными слоями, имя в работе VGG19\_unfrozen10, общее количество обучаемых параметров модели и классификатора 16 520 708;

ResNet50 с замороженными слоями, имя в работе ResNet50, количество обучаемых параметров 8 196 (параметры выходного слоя классификатора),

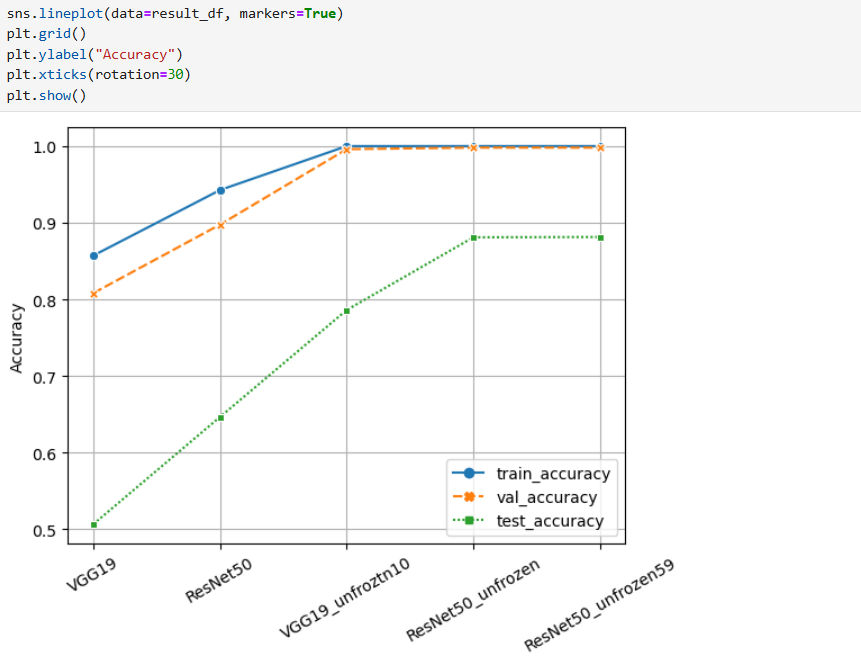
ResNet50 с 59 размороженными слоями, имя в работе ResNet50\_unfrozen59, общее количество обучаемых параметров модели и классификатора 17 487 364;

ResNet50 с полностью размороженными слоями, имя в работе ResNet50\_unfrozen, общее количество обучаемых параметров модели и классификатора 23 542 788.

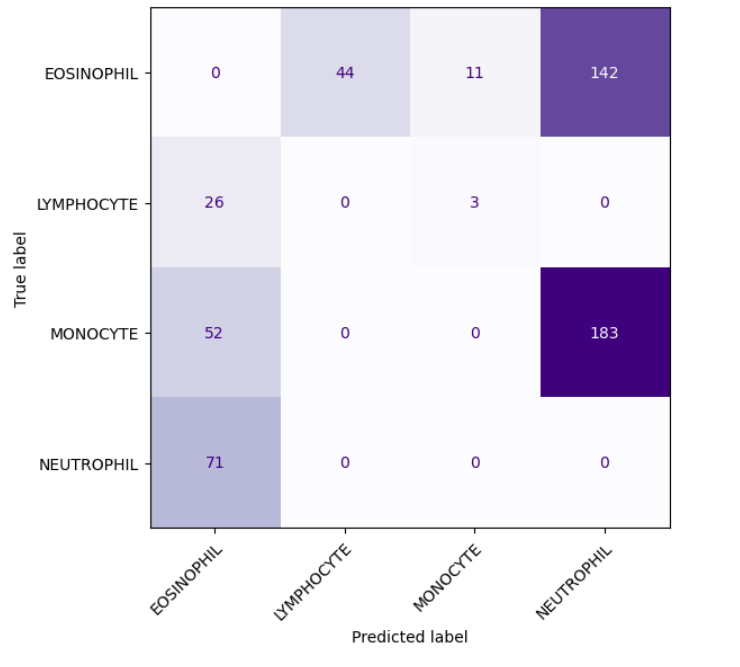
По результатам обучения я записывала точность моделей словарь характеристик моделей (model\_book) для всех трех наборов данных: тренировочный, валидационный и тренировочный. Для удобства переведем словарь в датафрейм пандас:



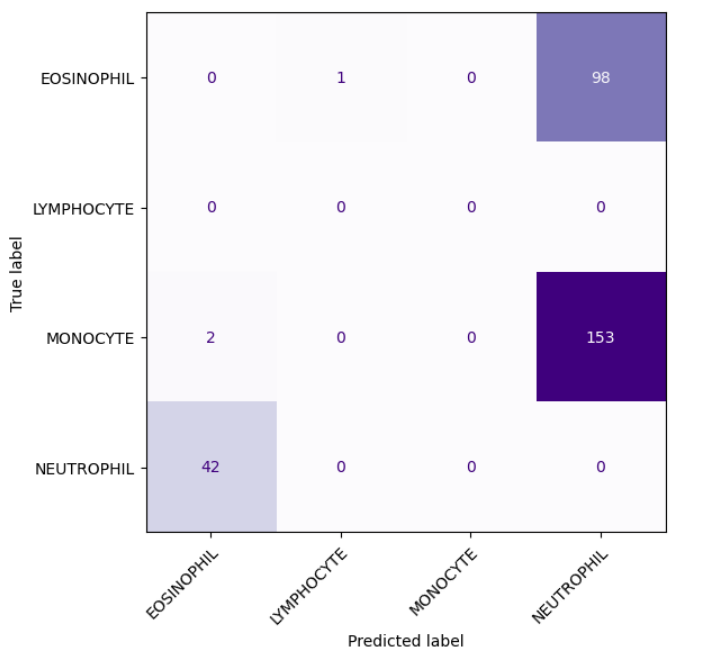
Построим графики изменения характеристик моделей:



Посмотрим топ три ошибок, которые делают обученные сети после тонкой настройки. Для модели VGG19 матрица ошибочных предсказаний выглядит так:



Для ResNet50:



Видим, что топ три ошибок: моноциты принимаются сетями за нейтрофилы, эозинофилы за нейтрофилы, и нейтрофилы за эозинофилы.

В процессе работы я пришла к следующим выводам:

1. При тренировке сети на основе замороженной VGG19 замена валидационного набора аналогичным, но со сбалансированными клетками крови дала прирост точности на 9%. Для обучения важно соблюдение баланса классов

в наборе данных.

2. Чем глубже базовая модель, тем точнее она дает предсказания, если в сети

решена проблема затухания градиента. Так точность сети ResNеt50 на всех наборах данных выше точности сети VGG19. Для валидационного и тренировочного набора разница примерно 9-10%, для тестового 27%.

3. Тонкая настройка модели значительно улучшает характеристики результирующих сетей. Так для всех трех настроенных моделей метрики точности равны 1 для тренировочного набора, и стремятся к 1 для валидационного набора. Точность моделей на тестовом наборе данных увеличилась для VGG19 в случае тонкой настройки на 55%, для ResNet50 на 35%.

4. Разморозка всех слоев базовой модели ResNet50 не дала прироста точности относительно модели с третью размороженных слоев, что подтверждает, что нижние слои отвечают за более общие признаки изображений, и чем выше слой, тем более специфичное изображение от помогает распознать.

5. Большой прирост точности между замороженными моделями и моделями точной настройки подтверждает выводы предыдущих работ, что эффективность тонкой настройки зависит от того, насколько сильно отличаются наборы данных и наиболее эффективна в случае, если новый набор данных содержит объекты, которых не было в наборе данных для предварительного обучения.

6. Обе сети делают одинаковые ошибки, возможно это связано с некачественной разметкой плохо определяемых клеток крови (моноцитов).

7. Максимальную точность дала сеть ResNet50\_unfrozen59 с третью размороженных слоев, но на тестовом наборе данных она составила 88%. Этого пока недостаточно для точной классификации типов клеток крови, которую можно использовать для диагностики заболеваний крови. Целью моего второго года обучения будет развитие этой модели за счет дополнительных слов для повышения ее точности.

**Список литературы.**

1. Karl Weiss, Taghi M. Khoshgoftaar and DingDing Wang. A survey of transfer learning. *J Big Data* **3**, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6

2. Pan, S. and Yang, Q. (2010) A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22, 1345-1359.

3. Raina, Rajat & Little, A. & Lee, Honglak & Packer, Ben & Ng, Andrew. (2007). Self-taught learning: Transfer learning from unlabeled data. Proceedings of the Twenty-fourth International Conference on Machine Learning. 227. 10.1145/1273496.1273592.

4. Машинное обучение (Machine Learning) Передача знаний и адаптация данных (Transfer Learning and Domain Adaptation), Уткин Л.В. Санкт-Петербургский политехнический университет Петра Великого.

5. Omori Yano, Marcus & Figueiredo, Eloi & da Silva, Samuel & Cury, Alexandre. (2023). Foundations and applicability of transfer learning for structural health monitoring of bridges. Mechanical Systems and Signal Processing. 204. 110766. 10.1016/j.ymssp.2023.110766.

6. Feature Representation Analysis of Deep Convolutional Neural Network using Two-stage Feature Transfer -An Application for Diffuse Lung Disease Classification, [Aiga Suzuki](https://arxiv.org/search/cs?searchtype=author&query=Suzuki,+A), [Hidenori Sakanashi](https://arxiv.org/search/cs?searchtype=author&query=Sakanashi,+H), [Shoji Kido](https://arxiv.org/search/cs?searchtype=author&query=Kido,+S), [Hayaru Shouno](https://arxiv.org/search/cs?searchtype=author&query=Shouno,+H), 2018, <https://doi.org/10.48550/arXiv.1810.06282>

7. Перенос обучения глубоких нейронных сетей, Кустикова В.

8. Jeffrey Li, Mikhail Khodak, Sebastian Caldas, DIFFERENTIALLY PRIVATE META-LEARNING, Published as a conference paper at ICLR 2020

9. M.Long, J.Wang, G.Ding, S.J.Pan, P.S.Yu Adaptation Regularization: A general Framework for Transfer Learning. IEEE Trans. on Knowledge and Data Eng., vol. 26(5), pp. 1076-1089, 2014

10. Raghu, M., Zhang, C., Kleinberg, J., & Bengio, S. (2019). [Transfusion: Understanding transfer learning for medical imaging](https://arxiv.org/abs/1902.07208). In Advances in neural information processing systems (pp. 3347-3357)

11. Chen, S., Ma, K., & Zheng, Y. (2019). [Med3d: Transfer learning for 3d medical image analysis](https://arxiv.org/abs/1904.00625). arXiv preprint arXiv:1904.00625.

12. Shaw, S., Pajak, M., Lisowska, A., Tsaftaris, S. A., & O'Neil, A. Q. (2020). [Teacher-Student chain for efficient semi-supervised histology image classification](https://arxiv.org/abs/2003.08797). arXiv preprint arXiv:2003.08797.

13. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Identity Mappings in Deep Residual Networks, arXiv:1603.05027v2 [cs.CV] 12 Apr 2016.

Приложения:

1. Источник данных: [https://www.kaggle.com/datasets/paultimothymooney/blood-cells](file:///C:\GB\Diploms_exam\diplom\kaggle.co%20https:\www.kaggle.com\datasets\paultimothymooney\blood-cells)

2. Рабочий ноутбук [blood-cells-classification/Blood\_cells\_classification/Blood\_cells\_classification.ipynb at main · OlesyaMishina/blood-cells-classification (github.com)](https://github.com/OlesyaMishina/blood-cells-classification/blob/main/Blood_cells_classification/Blood_cells_classification.ipynb)