

Анализ работы различных видов сверточных нейронных сетей для видовой классификации растений

Зиганшина Динара, Минаева Мария

Май 2020

1 Введение

Задача видовой классификации является довольно сложной задачей из-за больших внутриклассовых различий, таких как внешние различия в окраске и размере, различные положения на фотографии и тд., и незначительных межклассовых отличий, например мимикии и виды с очень похожими окрасками.

В течение последний нескольких лет был достигнут существенный прогресс в решении подобного рода задач. Например, точность классификации на датасете видов птиц CUB200-2011 значительно увеличилась с 10% в 2011 году [1] до 84,6% в 2015 году [6]. На данный момент существует множество проектов и статей, которые занимаются классификацией растений, так как это довольно сложный и трудоёмкий процесс. Более того, подобного рода проекты могут быть использованы учёными ботаниками для классификации видов и отслеживания их ареала обитания, а также количественной оценки растений в популяции.

В нашем исследовании мы взяли уже готовый датасет растений [3], состоящий из 4242 изображений пяти наиболее популярных родов цветочных растений: ромашки, одуванчика, шиповника, тюльпана и подсолнуха,- в каждом из которых присутствуют изображения снятые с разных ракурсов, различные рисунки, фотографии с людьми, а также изображения, не относящиеся к цветам для создания небольшого шума. На этом датасете мы провели обучением нескольких широко используемых моделей сверточных нейронных сетей (CNN), таких как ResNet18, ResNet34, VGG16 и VGG19 по двум различным сценариям: обучение последнего слоя или смешанный сценарий обучения.

Дальнейший отчет построен следующим образом: мы обсудим теорию, связанную с используемыми моделями и сценариями трансферного обучения в разделе 2; далее мы рассмотрим полученные результаты, а также проведем сравнение моделей и сценариев обучения в разделе 3; в разделе 4 мы сделаем основные выводы из проделанной работы с кратким описанием неполучившихся моментов, возможным дальнейшим развитием и применением нашей работы.

2 Обзор используемых моделей свёрточных сетей и сценариев их обучения

2.1 Обзор глубоких сверточных нейронных сетей

- **VGG**

VGG16 — модель сверточной нейронной сети была предложена K. Simonyan и A. Zisserman из Оксфордского университета [2]. Модель достигает точности 92.7 % — топ-5 при тестировании на ImageNet при решении задачи распознавания объектов на изображении. Этот датасет состоит из более чем 14 миллионов изображений, принадлежащих к 1000 классам.

Архитектура VGG16 представлена на [рисунке 1](#).

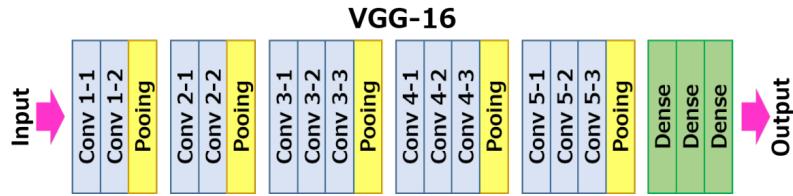


Рис. 1: Архитектура VGG16

На вход слоя conv1 подаются RGB изображения размера 224x224. Далее изображения проходят через стек сверточных слоев, в которых используются фильтры с очень маленьким рецептивным полем размера 3x3 (который является наименьшим размером для получения представления о том, где находится право/лево, верх/низ, центр изображения).

В одной из конфигураций используется сверточный фильтр размера 1x1, который может быть представлен как линейная трансформация входных каналов (с последующей нелинейностью). Сверточный шаг фиксируется на значении 1 пиксель. Пространственное дополнение (padding) входа сверточного слоя выбирается таким образом, чтобы пространственное разрешение сохранялось после свертки, то есть дополнение равно 1 для 3x3 сверточных слоев. Пространственный пулинг осуществляется при помощи пяти max-pooling слоев, которые следуют за одним из сверточных слоев (не все сверточные слои имеют последующие max-pooling). Операция max-pooling выполняется на окне размера 2x2 пикселей с шагом 2.

После стека сверточных слоев (который имеет разную глубину в разных архитектурах) идут три полно связанных слоя: первые два имеют по 4096 каналов, третий — 1000 каналов (так как в соревновании ILSVRC требуется классифицировать объекты по 1000 категориям; следовательно, классу соответствует один канал). Последним идет soft-max слой. Конфигурация полно связанных слоев одна и та же во всех нейросетях.

Все скрытые слои снабжены ReLU. Отметим также, что сети (за исключением одной) не содержат слоя нормализации (Local Response Normalisation), так как нормализация не улучшает результата на датасете ILSVRC, а ведет к увеличению потребления памяти и времени исполнения кода.

Конфигурации сверточных сетей представлены на [рисунке 2..](#) Каждая сеть соответствует своему имени (A-E). Все конфигурации имеют общую конструкцию, представленную в архитектуре, и различаются только глубиной: от 11 слоев с весами в сети A (8 сверточных и 3 полно связанных слоя) до 19 (16 сверточных и 3 полно связанных слоя). Ширина сверточных слоев (количество каналов) относительно небольшая: от 64 в первом слое до 512 в последнем с увеличением количества каналов в 2 раза после каждого max-pooling слоя.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-128	conv3-64 conv3-64	conv3-64 conv3-128	conv3-64 conv3-128
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Рис. 2: Архитектура VGG

Недостатки:

- медленная скорость обучения
- из-за глубины и количества полносвязных слоёв, VGG16 весит более 533 МБ. Это делает процесс развертывания VGG утомительной задачей

• Inception

Для извлечения информации с большой площади лучше всего подходят большие фильтры, и наоборот для маленьких объектов лучше использовать маленькие фильтры. Глубокие нейронные сети намного сложнее обучать: для них появляется проблема затухания градиента и таким сетям свойственно переобучаются. Чтобы решить первую проблему исследователи придумали Inception модуль, который применяет фильтры разного размера и затем склеивает полученные каналы. При этом извлекается как информация из больших объектов, так и из маленьких.

Реализацию модуля Inception можно сделать более эффективной, если сначала уменьшить количество каналов с помощью сверточного слоя 1×1 и лишь затем применить слой с фильтрами 5×5 . Сокращение вычислений происходит за счет того, что мы сначала уменьшаем размерность данных и лишь затем преобразовываем их. [4] Продвинутая реализация:

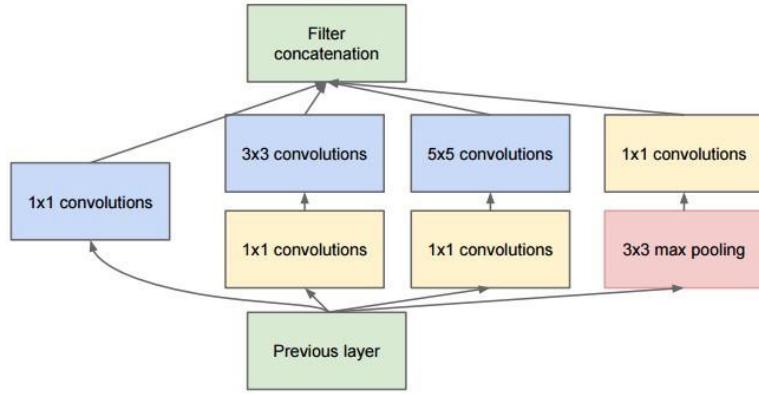


Рис. 3: Inception модуль

Сеть Inception состоит из корня (нескольких сверточных слоев) и Inception модулей идущих за ним. На [рисунке 4](#) оранжевым прямоугольником выделен корень, а фиолетовыми - вспомогательные классификаторы. Именно они помогают бороться со второй проблемой, которую мы упомянули ранее, а именно - переобучением. Теперь наша функция потерь - взвешенная сумма LogLoss на двух вспомогательных классификаторах и основном в конце нейронной сети, что уменьшает вероятность переобучения и фактически сводит её на нет.[\[4\]](#)

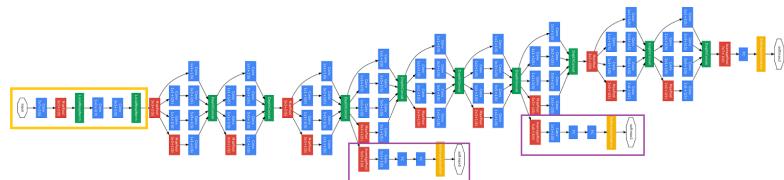


Рис. 4: Inception

В следующем поколении моделей Inception (Inception V2) авторы разложили слой 5×5 на два слоя 3×3 . Далее была использована техника Batch Normalization, позволяющая многократно увеличивать скорость обучения за счет нормализации распределения выходных слоев внутри сети. В модели Inception-v3 авторы предложили разложить фильтр $N \times N$ на два последовательных фильтра $1 \times N$ и $N \times 1$.[\[7\]](#) На [рисунке 5](#) представлена архитектура модели Inception-v3

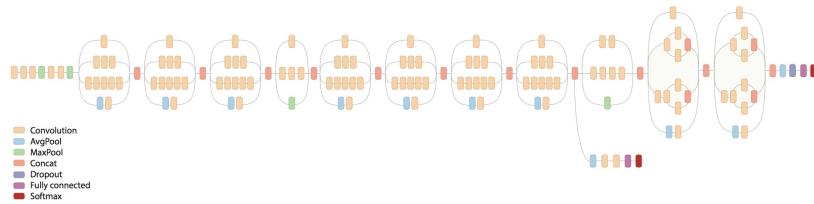


Рис. 5: Inception V3

• ResNet

Одной из основных проблем глубоких нейросетей является проблема исчезающего градиента. В двух словах, это техническая проблема, возникающая при использовании метода обратного распространения ошибки для алгоритма вычисления градиента. При работе с обратным распространением ошибки используется цепное правило. При этом, если градиент имеет малое значение в конце сети, то он может принять бесконечно малое значение к тому моменту, как он достигнет начала сети. Это может привести к проблемам совершенно различного свойства, включая невозможность обучения сети в принципе.

Для решения этой проблемы была предложена следующая идея — позволить сети изучать остаточное отображение (элемент, который следует добавить ко входным данным) вместо отображения как такового. Технически это выполняется с помощью обходного соединения, показанного на рисунке.

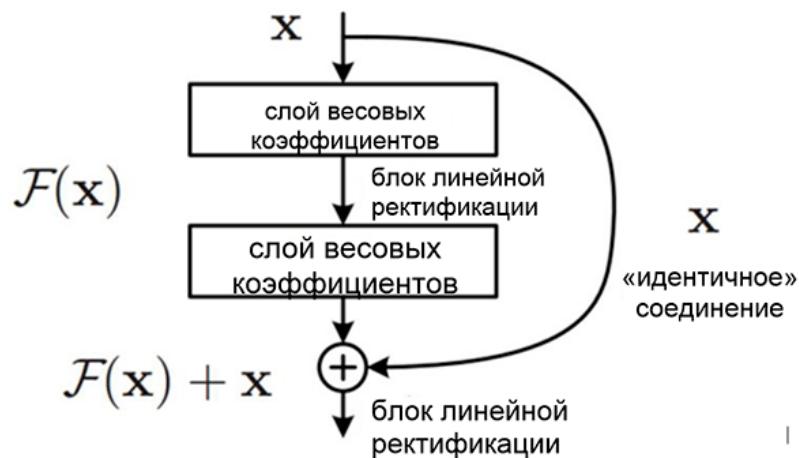


Рис. 6: Обходное соединение

Принципиальная схема остаточного блока: входные данные передаются по сокращенному соединению в обход преобразующих слоев и добавляются к полученному результату. Обратите внимание на то, что «идентичное» соединение не добавляет дополнительные параметры в сеть, поэтому ее структура не усложняется. Эта идея чрезвычайно проста, но в то же время исключительно эффективна. Она решает проблему исчезающего градиента, позволяя ему перемещаться без каких-либо изменений от верхних слоев к нижним посредством «идентичных» соединений. Благодаря этой идеи можно обучать очень глубокие, чрезвычайно глубокие сети. На [рисунке 7](#) представлена архитектура разных моделей нейронной сети ResNet.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112x112			7x7, 64, stride 2		
				3x3 max pool, stride 2		
conv2_x	56x56	$\left[\begin{array}{c} 3 \times 3, 64 \\ 3 \times 3, 64 \end{array} \right] \times 2$	$\left[\begin{array}{c} 3 \times 3, 64 \\ 3 \times 3, 64 \end{array} \right] \times 3$	$\left[\begin{array}{c} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{array} \right] \times 3$	$\left[\begin{array}{c} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{array} \right] \times 3$	$\left[\begin{array}{c} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{array} \right] \times 3$
conv3_x	28x28	$\left[\begin{array}{c} 3 \times 3, 128 \\ 3 \times 3, 128 \end{array} \right] \times 2$	$\left[\begin{array}{c} 3 \times 3, 128 \\ 3 \times 3, 128 \end{array} \right] \times 4$	$\left[\begin{array}{c} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{array} \right] \times 4$	$\left[\begin{array}{c} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{array} \right] \times 4$	$\left[\begin{array}{c} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{array} \right] \times 8$
conv4_x	14x14	$\left[\begin{array}{c} 3 \times 3, 256 \\ 3 \times 3, 256 \end{array} \right] \times 2$	$\left[\begin{array}{c} 3 \times 3, 256 \\ 3 \times 3, 256 \end{array} \right] \times 6$	$\left[\begin{array}{c} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{array} \right] \times 6$	$\left[\begin{array}{c} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{array} \right] \times 23$	$\left[\begin{array}{c} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{array} \right] \times 36$
conv5_x	7x7	$\left[\begin{array}{c} 3 \times 3, 512 \\ 3 \times 3, 512 \end{array} \right] \times 2$	$\left[\begin{array}{c} 3 \times 3, 512 \\ 3 \times 3, 512 \end{array} \right] \times 3$	$\left[\begin{array}{c} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{array} \right] \times 3$	$\left[\begin{array}{c} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{array} \right] \times 3$	$\left[\begin{array}{c} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{array} \right] \times 3$
	1x1			average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Рис. 7: Архитектура ResNet

2.2 Transfer Learning

Transfer Learning - это процесс дообучения на новых данных какой-либо нейросети, уже обученной до этого на других данных, обычно на каком-нибудь хорошем, большом (содержащем миллионы картинок) датасете, например, ImageNet (14 млн картинок). Идея трансферного обучения заключается в том, что знания, накопленные моделью, подготовленной для выполнения одной задачи - могут быть перенесены на другую модель, чтобы помочь в построении прогнозов для другой, схожей задачи. Существует два возможных сценария использования уже обученных нейросетей:

1. Feature Extractor (CNN как средство для извлечения признаков)

В этом случае используется уже обученная сеть, у которой убираются последние полно связанные (Fully-Connected) слои сети, отвечающие за классификацию, веса предыдущих слоёв замораживаются, остаются неизменными. Такая сеть выдаёт не метки классов, а то, что поступало на вход Fully-Connected слою. Далее запускают сеть на новом датасете, получают выходные данные сети, которые и являются признаками объектов. На этих признаках можно обучить какой-либо классификатор. Таким образом, получили сеть, хорошо работающую на новом датасете.

2. Fine Tuning(CNN, которую можно дообучить)

Берётся уже обученную сетку, убираются последние Fully-Connected слои сети, отвечающие за классификацию. Затем backpropagation распространяют ещё на сколько-то слоёв назад (размораживают веса в этих слоях), чтобы скорректировать их под новые данные. Можно распространить обучение и на всю сеть, однако часто первые слои всё же оставляют замороженными, поскольку они (как ожидается) извлекают более общие признаки из объектов. К тому же, обучать всю сеть дольше, чем несколько слоёв. Выбор количества слоёв, которые будут обучаться, зависит от того, какое качество классификации должно быть достигнуто. Теперь сеть выдаёт не метки классов, а то, что поступало на вход Fully-Connected (веса последних (или всех) слоёв были изменены под наши данные). На этих признаках можно обучить какой-либо классификатор. Таким образом, получили сеть, хорошоирующую на новом датасете.

В зависимости от количества и природы входных данных могут быть реализованы несколько стратегий Transfer Learning, а именно:

- На вход сети подается мало данных ($\leq 10k$), и они похожи на данные, на которых была обучена сеть до этого. Если данные сильно схожи с данными, на которых производилось предобучение, то можно попробовать использовать готовую модель. Если качество не достигает желаемого уровня, то стоит использовать CNN для извлечения признаков и обучить классификатор на этих данных (см. [пункт 1 стратегий обучения](#)). Так как данные похожи на те, на которых обучалась сеть, то высокогорные признаки, полученные с помощью последних слоёв сети, должны оказаться информативными. Если делать в этом случае Fine-Tuning, то сеть может переобучиться, поскольку данных мало.
- На вход сети подается мало данных ($\leq 10k$), и они не похожи на данные, на которых была обучена сеть до этого. Самый невыгодный случай. В данном случае нельзя ожидать от сети информативности выходов последних слоёв для новых данных. Следует также действовать в соответствии с [пунктом 1 стратегий](#)

обучения, но брать как признаки выходы более ранних слоёв, так как они соответствуют более общим признакам данных.

- На вход сети подается много данных ($\geq 10k$), и они похожи на данные, на которых была обучена сеть до этого. В этом случае можно смело делать **Fine-Tuning**, если желаемое качество не было достигнуто на готовой модели, потому что вероятность переобучения на большом количестве входных данных существенно меньше. В данном случае имеет смысл попробовать разморозить веса последних нескольких слоёв. Количество слоёв, которое стоит подвергать разморозке зависит от конкретной задачи, доступных вычислительных мощностей и расчетного времени обучения.
- На вход сети подается много данных ($\geq 10k$), и они не похожи на данные, на которых была обучена сеть до этого. Общий подход к обучению остается таким же, как и в [предыдущем пункте](#). Разумно использовать стратегию **Fine-Tuning** для практических всех слоёв сети. В данном случае уместно полностью менять все параметры и гиперпараметры нейросети, оставляя неизменной только архитектуру сети. Однако часто веса предобученной сети оставляют в качестве инициализации для обучения на новых данных.

3 Результаты

3.1 Модель классификатора

Для решения поставленной задачи мы использовали следующий классификатор:

- Полносвязанный слой, принимающий на вход признаки от модели
- далее идет ReLU
- для моделей ResNet далее идет слой нормализации батчей
- далее идет слой Dropout, который в случайном порядке зануляет веса признаков с указанной вероятностью. Это делается для борьбы с переобучением
- затем снова идет полносвязанный слой
- далее слой ReLU
- далее для моделей ResNet идет слой нормализации батчей
- далее идет слой Dropout
- затем снова следует полносвязанный слой
- последним слоем является логарифмический слой SoftMax

3.2 Сравнение моделей

В данной работе были исследованы следующие модели:

- ResNet 18
- ResNet 50
- ResNet 152
- VGG 16
- VGG 19
- Inception V3

Для каждой из этих моделей было проведено обучение последнего слоя, т.е. готовая модель использовалась в качестве **Feature Extractor**, с количеством эпох равным 7, построены графики зависимости метрик оценки качества, таких как F1 score, Precision и Recall, а также функция потерь Loss для обучающей выборки и валидационной (см.[рисунок 8](#) для валидационной выборки и [дополнительный рисунок 1](#) для обучающей выборки). Для тестовой выборки были оценены средние значения метрик оценки качества (см. [таблицу 1](#)).

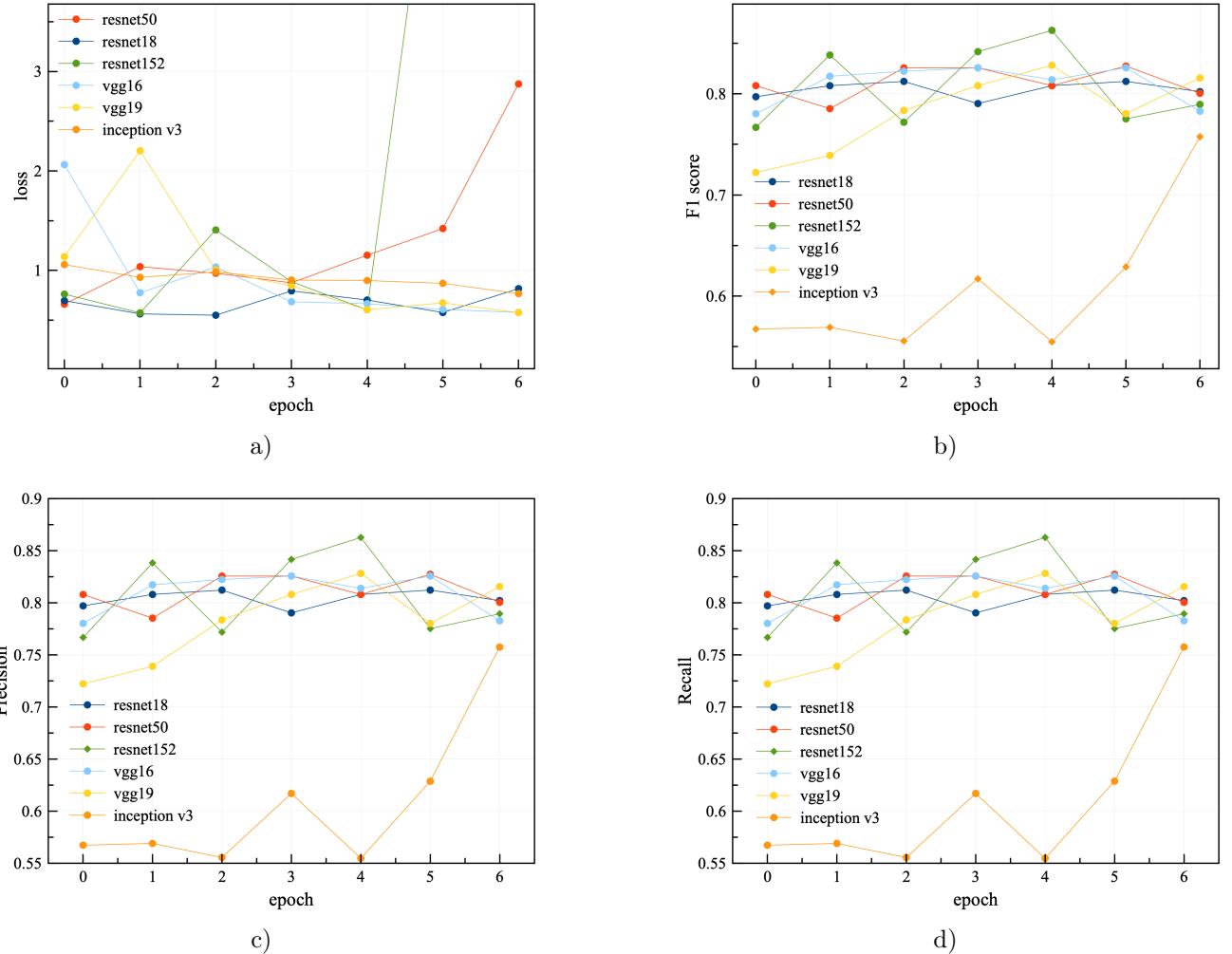


Рис. 8: Сравнение моделей на валидационной выборке: а) функция потерь Loss в зависимости от номера эпохи, б) метрика качества модели F1 score в зависимости от номера эпохи, в) метрика качества модели Precision в зависимости от номера эпохи, д) метрика качества модели Recall в зависимости от номера эпохи.

Таблица 1: Метрики качества для различных моделей на тестовой выборке.

Модель	Num. of correct answers
ResNet 18	82 %
ResNet 50	86 %
ResNet 152	88%
VGG 16	86 %
VGG 19	82 %
Inception v3	69 %

Исходя из полученных метрик качества можно сделать вывод, что модели семейства ResNet начинают переобучаться после 4-7 эпохи, что приводит к значительному росту метрики Loss. Самый значительный рост наблюдается для модели ResNet 152 после 5 эпохи, хотя у модели ResNet 18 переобучение начинается на последней седьмой эпохе и оно незначительно. Напротив, исходя из метрики Loss, модели семейства VGG и Inception не подвержены переобучению в течение всего процесса обучения. Из зависимостей метрик F1 score, Precision и Recall можно сделать вывод, что наиболее подходящей для нашей задачи моделью является модель ResNet 152 с пятью эпохами в обучении, так как именно пятой эпохе для этой модели соответствует максимальное значение данных метрик на валидационной выборке: F1 score = 0.863. Далее идет модель VGG 19 со значением метрик на валидационной выборке: F1 score = 0.828, затем следуют модели: ResNet 50 F1 score = 0.827. и VGG 16 F1 score = 0.826. Наименее пригодной из исследуемых моделей оказалась модель ResNet 18 с F1 score = 0.812 на валидационной выборке. Из зависимостей различных метрик от номера эпохи можно сделать вывод, что модель Inception v3 в рамках данной задачи не успела дообучиться и показать максимально возможный для нее результат. Мы предполагаем, что при большем количестве эпох эта модель показала бы результаты, сравнимые с результатами моделей ResNet 152 и VGG 19.

3.3 Сравнение стратегий обучения

После определения наиболее подходящей для нашей задачи модели мы решили выбрать наиболее подходящую стратегию обучения. Была выбрана стратегия обучения последних пяти слоев нейронных сетей на нашем датасете и проведено сравнение со стратегией обучения последнего слоя для каждой модели.

На [рисунке 9](#) приведены зависимости метрик качества от номера эпохи для двух стратегий обучения модели ResNet 18, измеренных на валидационной выборке. Значения Loss для смешанной стратегии обучения оказались существенно выше, чем для обучения только последнего полностью связанного слоя, к тому же после некоторого уменьшения Loss снова начинает расти после 4 эпохи и уже к 7 эпохе значительно выше лучшего значения, что указывает на переобучение модели. Вдобавок, F1 score для смешанного сценария оказался слишком низким для того, чтобы можно было использовать обученную в таких условиях модель в качестве классификатора видов. Исходя из полученных данных, можно сделать вывод, что для модели ResNet 18 для текущего датасета смешанный сценарий обучения не даёт желаемых результатов и не может быть использован в каких-либо разумных целях.

На [рисунке 10](#) приведены зависимости метрик качества от номера эпохи для двух стратегий обучения модели ResNet 50, измеренных на валидационной выборке. Из значений Loss для смешанного сценария обучения видно, что смешанная модель не подвержена влиянию переобучения на всем времени обучения, тогда как стратегия обучения только последнего слоя привела к наличию переобучения начиная с 4ой эпохи. Однако, значения Loss для смешанной стратегии обучения все же остаются выше, чем значение Loss для сценария обучения только последнего слоя до появления переобучения, что может говорить о том, что количество объектов каждого класса в датасете недостаточно для применения подобной стратегии. Вдобавок, F1 score для смешанного сценария существенно ниже, чем для Feature Extractor сценария. Из графиков зависимости Loss и F1 score для двух сценариев обучения можно сделать вывод, что смешанный сценарий пригоден для обучения данной модели, однако датасет должен содержать не менее, чем 5000 объектов, а желательно 10000 объектов для достижения

приемлемого качества.

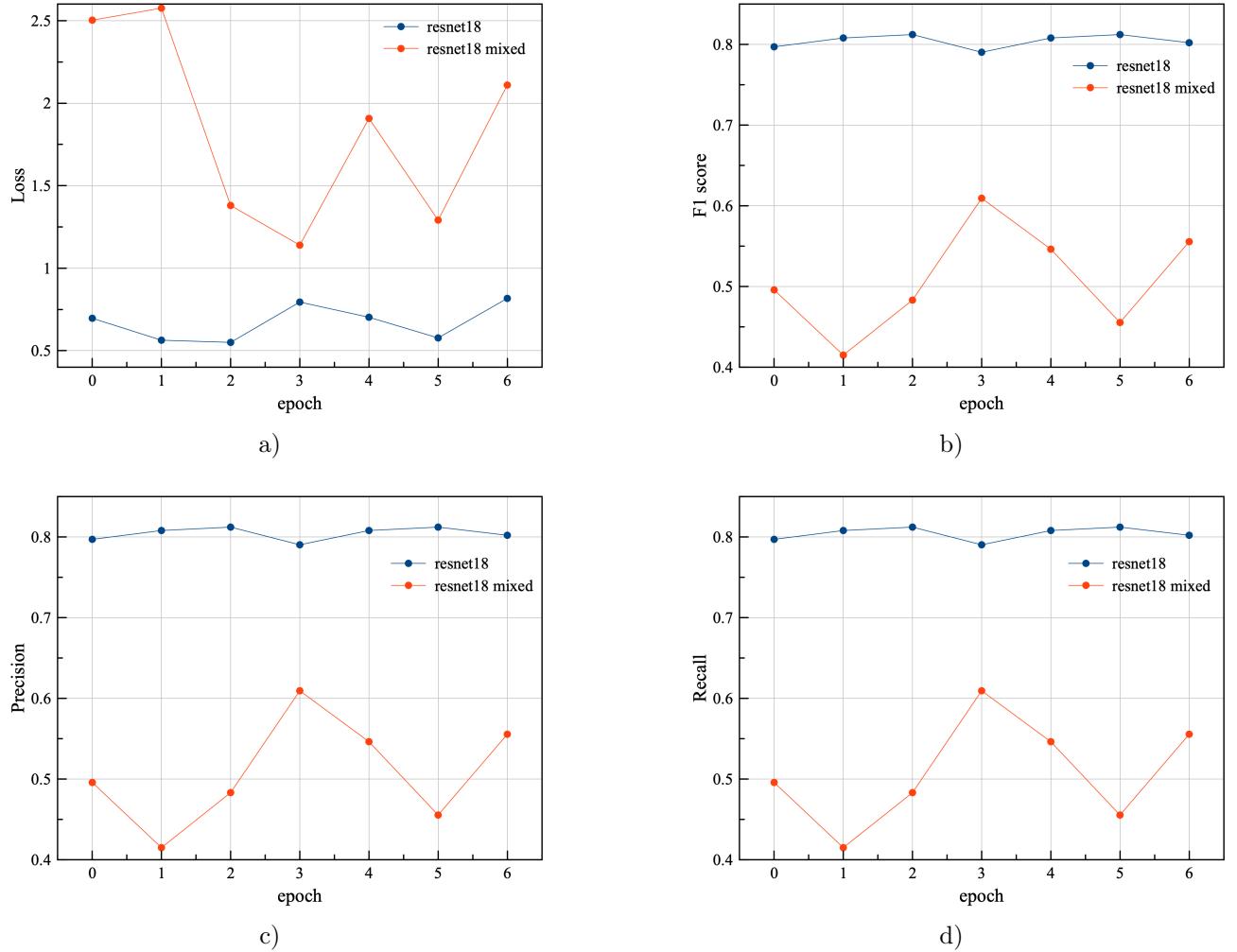


Рис. 9: Сравнение стратегий обучения для модели Resnet 18 на валидационной выборке: а) функция потерь Loss в зависимости от номера эпохи, б) метрика качества модели F1 score в зависимости от номера эпохи, в) метрика качества модели Precision в зависимости от номера эпохи, г) метрика качества модели Recall в зависимости от номера эпохи.

На [рисунке 11](#) приведены зависимости метрик качества от номера эпохи для двух стратегий обучения модели VGG 16, измеренных на валидационной выборке. Для данной модели значения Loss при смешанной стратегии обучения меньше, чем при обучении только последнего слоя до 4ой эпохи, после которой для смешанного сценария обучения наблюдается незначительное увеличение Loss, что скорее всего связано с слишком большим шагом при градиентном спуске и проскачиванием экстремума оптимизируемой функции NLLLoss. Более того, F1 score для смешанного сценария достигает сравнимых значений на последней эпохе с сценарием обучения последнего слоя, что свидетельствует о том, что при большем количестве эпох данная модель со смешанным сценарием обучения вполне может быть использована в качестве классификатора для наборов данных, схожих с нашим датасетом, и давать достаточно достоверные предсказания.

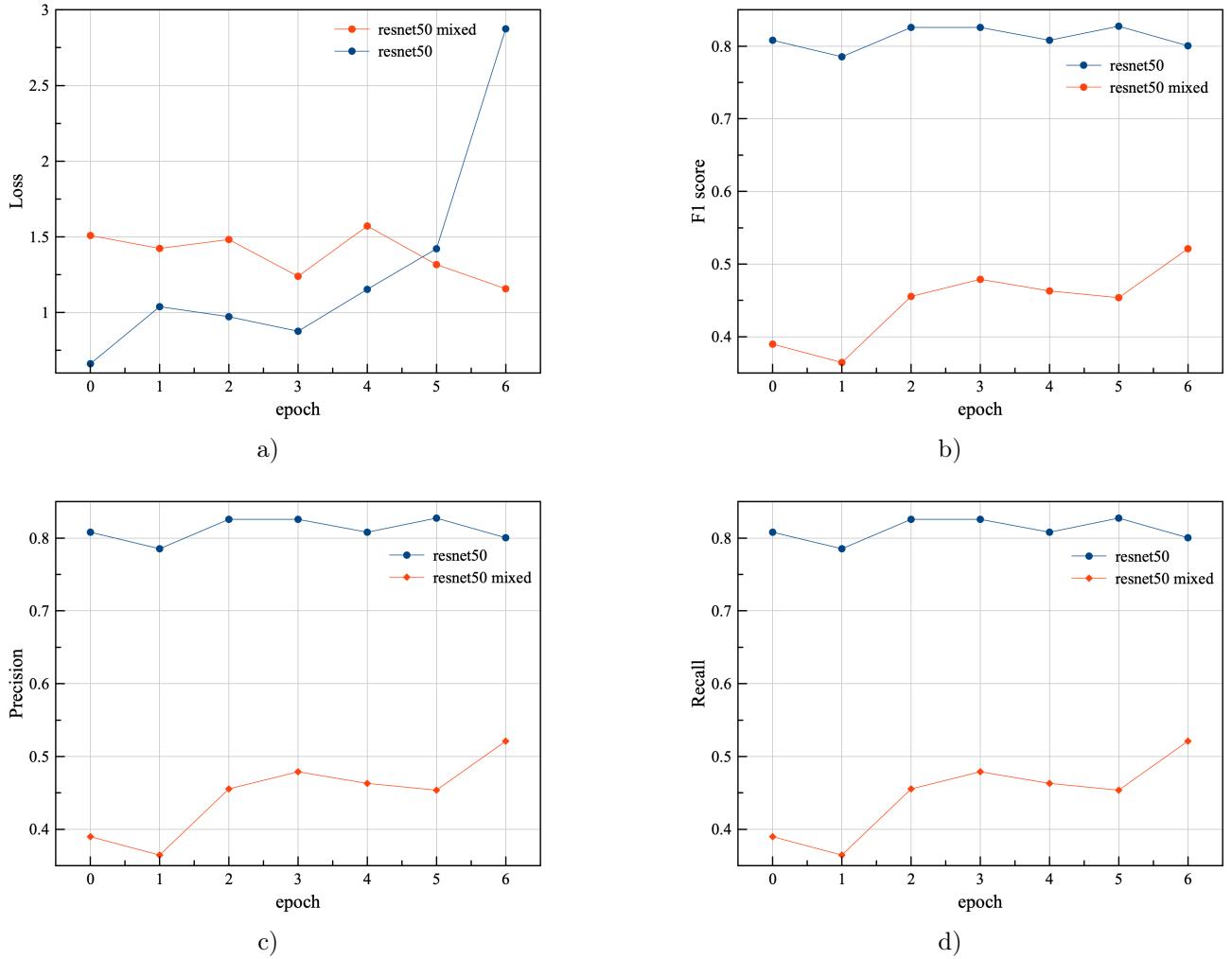
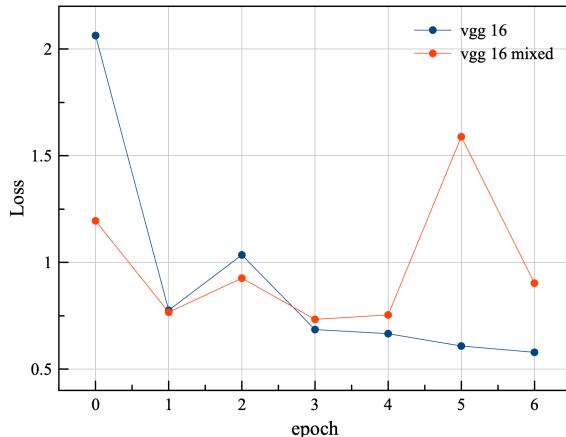
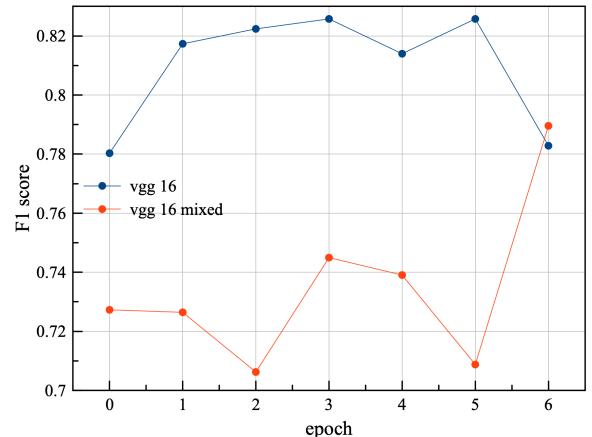


Рис. 10: Сравнение стратегий обучения для модели Resnet 50 на валидационной выборке: а) функция потерь Loss в зависимости от номера эпохи, б) метрика качества модели F1 score в зависимости от номера эпохи, в) метрика качества модели Precision в зависимости от номера эпохи, г) метрика качества модели Recall в зависимости от номера эпохи.

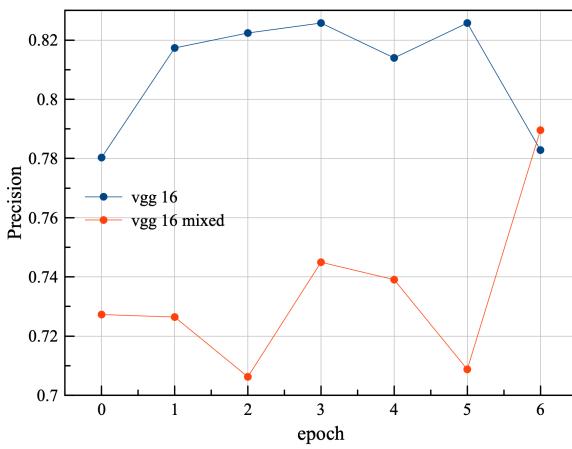
На [рисунке 12](#) приведены зависимости метрик качества от номера эпохи для двух стратегий обучения модели VGG 19, измеренных на валидационной выборке. Для этой модели метрика Loss для смешанной стратегии обучения немного меньше, чем для сценария обучения последнего слоя, однако хорошо заметна общая тенденция Loss к уменьшению с ростом номера эпохи. Более того, к концу обучения Loss для смешанной стратегии становится меньше 1, что является очень хорошим результатом в условиях нашего эксперимента. Вдобавок, F1 score для модели VGG 19 со смешанным сценарием обучения показывает сопоставимый со стратегией обучения последнего слоя рост, как и для стратегии обучения последнего слоя, и достигает весьма приемлемых значений к концу обучения $F1\ score = 0.795$, что несущественно меньше значения $F1\ score$, которое было достигнуто моделью ResNet 18 в [предыдущем разделе](#). Исходя из полученных для этой модели данных, смешанная и "несмешанная" стратегии обучения показывают хорошие результаты точности на тестовой выборке 82 % и 78% процентов соответственно, поэтому можно сделать вывод, что обе стратегии могут быть применены для решения данной задачи, но для смешанной модели обучения мы советуем взять количество эпох ≥ 10 .



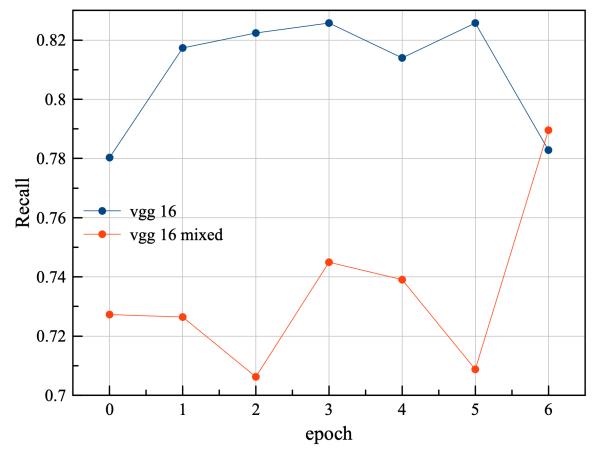
a)



b)



c)



d)

Рис. 11: Сравнение стратегий обучения для модели VGG 16 на валидационной выборке: а) функция потерь Loss в зависимости от номера эпохи, б) метрика качества модели F1 score в зависимости от номера эпохи, в) метрика качества модели Precision в зависимости от номера эпохи, д) метрика качества модели Recall в зависимости от номера эпохи.

На [рисунке 13](#) приведены зависимости метрик качества от номера эпохи для двух стратегий обучения модели ResNet 152, измеренных на валидационной выборке. Для этой модели метрика Loss для смешанной стратегии обучения претерпевает огромные скачки со 2 по 4 и с 5 по 7 эпохи, что скорее всего связано со слишком большим шагом при градиентном спуске и проскачиванием экстремума оптимизируемой функции NLLLoss. Вдобавок, модель ResNet 152 со смешанным сценарием обучения показывает весьма скромные результаты метрики F1 score, которая на всем процессе обучения не достигает значения 0.5 и начинает незначительно уменьшаться после 4 эпохи. Исходя из полученных данных, можно сделать вывод, что для модели ResNet 152 для текущего датасета смешанный сценарий обучения не даёт желаемых результатов и не может быть использован в качестве классификатора видов.

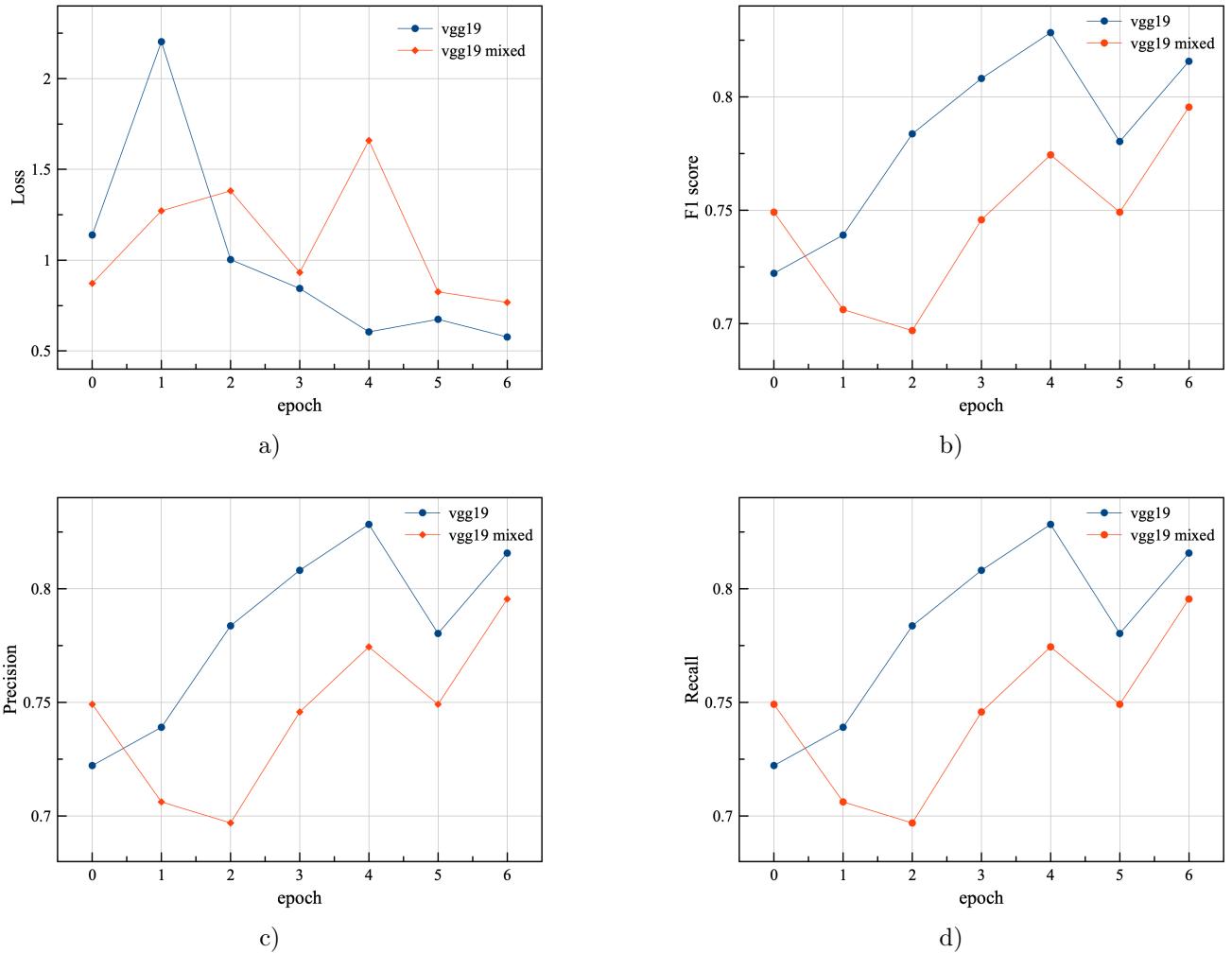


Рис. 12: Сравнение стратегий обучения для модели VGG 19 на валидационной выборке: а) функция потерь Loss в зависимости от номера эпохи, б) метрика качества модели F1 score в зависимости от номера эпохи, в) метрика качества модели Precision в зависимости от номера эпохи, д) метрика качества модели Recall в зависимости от номера эпохи.

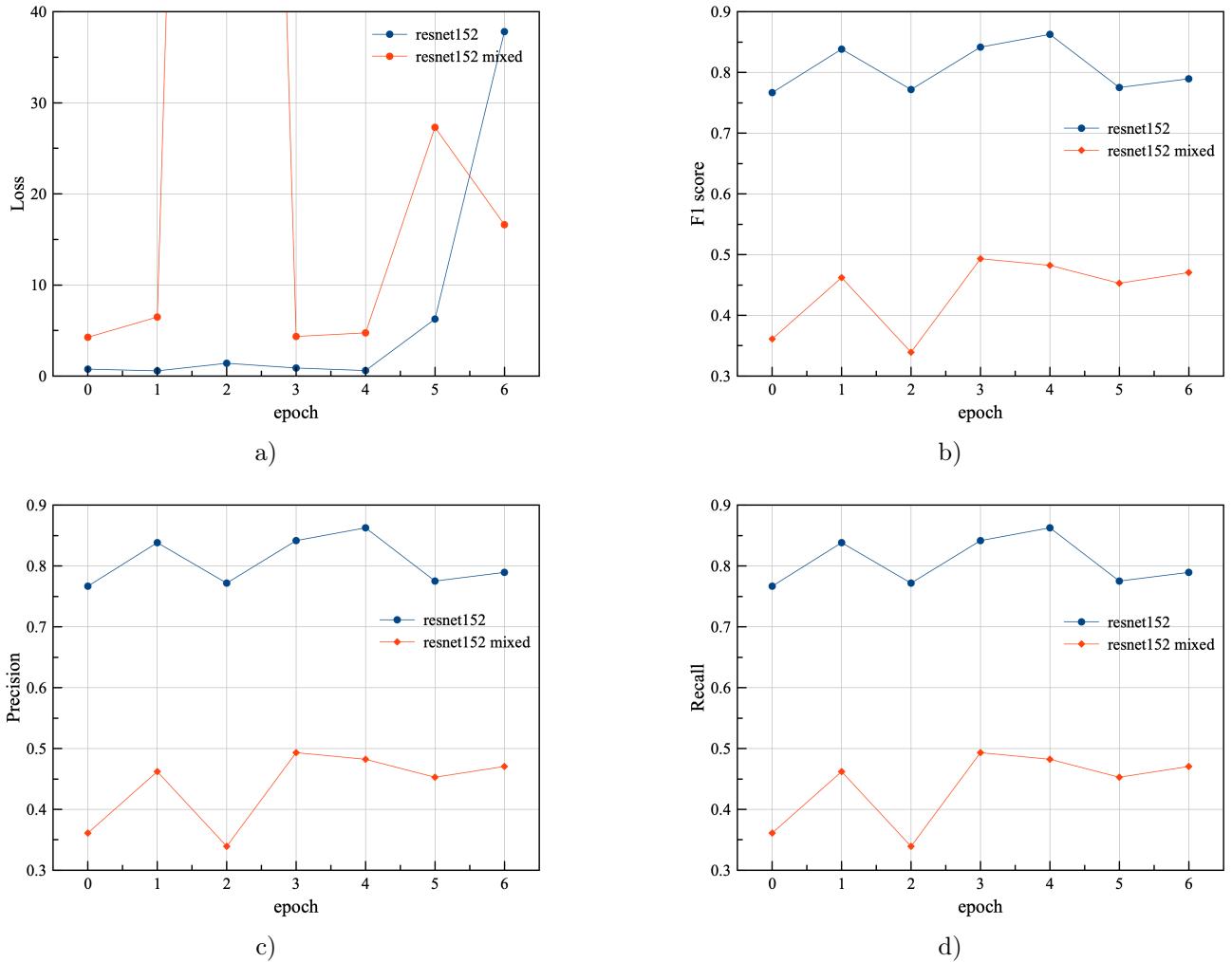


Рис. 13: Сравнение стратегий обучения для модели Resnet 152 на валидационной выборке: а) функция потерь Loss в зависимости от номера эпохи, б) метрика качества модели F1 score в зависимости от номера эпохи, в) метрика качества модели Precision в зависимости от номера эпохи, г) метрика качества модели Recall в зависимости от номера эпохи.

Таблица 2: Метрики качества для различных моделей и стратегий на тестовой выборке.

Модель	Correct answers not in mixed	Correct answers in mixed
ResNet 18	82 %	57 %
ResNet 50	86 %	42 %
ResNet 152	88%	50 %
VGG 16	86 %	79 %
VGG 19	82 %	78 %
Inception v3	69 %	-

В данном эксперименте модель Inception v3 была исключена, так как она показала слишком низкую скорость обучения при стратегии обучения последнего слоя, и на 7 эпохах не смогла достичь максимального для нее качества. Так как остальные модели при смешанной стратегии обучения показали, что для достижения их максимального результата нужно более, чем 7 эпох, то мы ожидаем, что для Inception v3 7 эпох окажется тем более недостаточно.

Из полученных значений процентного соотношения правильно определенных видов можно сделать вывод, что только модели семейства VGG пригодны для обучения по смешанному сценарию для нашего датасета. Кроме того, мы считаем, что смешанный сценарий плохо себя показал в нашей задаче, так как датасет является достаточно маленьким (4232 изображения), и скорее всего он схож с датасетом ImageNet, на котором были предобучены модели. В приложениях можно найти графики зависимости всех метрик качества для обучения смешанных моделей (см. [дополнительный рисунок 2](#) и [дополнительный рисунок 3](#)).

3.4 Обсуждение полученных результатов

В нашей работе были проведены исследования различных моделей при классификации родов растений на датасете[3]. Мы получили следующие основные результаты:

- модели семейства ResNet показывают хорошие значения метрики качества F1 score, однако при стратегии обучения последнего слоя они подвержены переобучению
- модели семейства VGG также показали хорошие значения метрики качества F1 score, но, в отличие от ResNet, они не оказались подвержены переобучению в течение всего процесса обучения. К сожалению, недостатком этих моделей оказался большой их вес и длительное время обучения, по сравнению с моделями ResNet и Inception v3
- модель Inception v3 при использовании стратегии обучения последнего слоя показала нулевую переобучаемость, однако, скорость обучения оказалась слишком низкая, и данной модели не хватило 7 эпох для обучения, как остальным моделям, поэтому мы предполагаем, что при количестве эпох ≥ 10 данная модель может хорошо себя показать, так как ее архитектура заточена под борьбу с переобучением
- стратегия смешанного обучения для нашего датасета показала себя не лучшим образом: значения метрики F1 score почти для всех моделей, кроме VGG, существенно меньше, чем для "не смешанной" стратегии обучения; скорее всего это вызвано тем, что наш датасет слишком маленький для применения такого рода стратегии обучения

4 Выводы

Наше исследование основывалось на статье [5], в которой описывалось исследование различных моделей для классификации видов бабочек и мотыльков. В нашей работе мы постарались охватить больше моделей и рассмотреть несколько стратегий обучения, однако используемый нами датасет [3] не очень большой и не содержит большое количество видов, в связи с нехваткой вычислительной мощностей и слишком долгим временем обучения (более 20 часов) на имеющихся мощностях. Для нашей задачи мы выявили, что датасеты сравнимые по размеру с нашим не пригодны для смешанной стратегии обучения свёрточных нейронных сетей. Однако стратегия обучения только последнего слоя хорошо себя зарекомендовала для таких целей.

Неясным моментом осталось то, почему модели Inception v3 не хватило 7 эпох для обучения и каким были бы результаты при ее полном обучении. Кроме того, неясной осталась и смешанная стратегия обучения для этой модели, а также возможные результаты для неё.

Наша работа может быть улучшена следующим образом: может быть взят более большой датасет, содержащий большее количество классов и имеющий большее практическое применение; также хорошим upgrade нашей работы будет перенос всех вычислений на GPU, так как это сильно ускорит обучение и позволит обучать большее количество эпох; в зависимости от датасета могут быть скорректированы стратегии обучения моделей.

После произведения ряда улучшений, описанных выше, наша работа может быть применена для подбора оптимальной модели и стратегии обучения для конкретного датасета, что может быть полезно в ряде случаев, таких как построение пользовательских классификаторов различных видов животных, растений, грибов и тп.

Список литературы

- [1] P. Welinder P. Perona C. Wah, S. Branson and S. Belongie. *The Caltech-UCSD Birds-200-2011 Dataset*. Technical report, 2011.
- [2] Andrew Zisserman Karen Simonyan. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv preprint arXiv:1409.1556, 2014.
- [3] Alexander Mamaev. *Flowers Recognition*. Kaggle, 2018.
- [4] Bharath Raj. *A Simple Guide to the Versions of the Inception Network*. Towards Data Science, 2018.
- [5] Qi Chang Hui Qu Pengxiang Wu Jingru Yi. *Fine-Grained Butterfly and Moth Classification Using Deep Convolutional Neural Networks*. Semantic Scholar, 2017.
- [6] Y. Zhang Z. Xu, S. Huang and D. Tao. *Augmenting strong supervision using web data for fine-grained categorization*. In IEEE International Conference on Computer Vision, ICCV, pages 2524–2532, 2015.
- [7] Сикорский О.С. *Обзор свёрточных нейронных сетей для задачи классификации изображений*. МГТУ им. Баумана, 2017.

5 Приложения

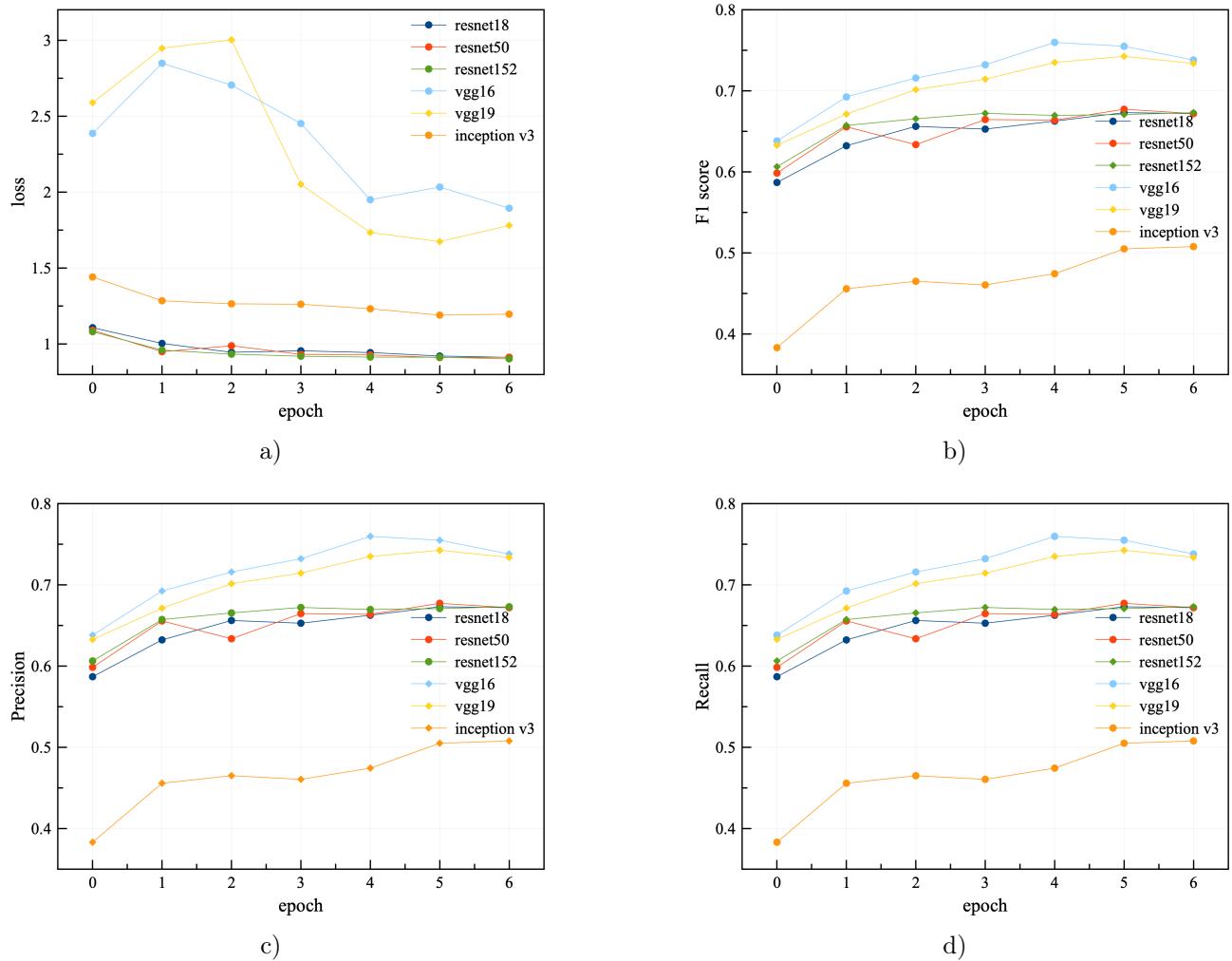
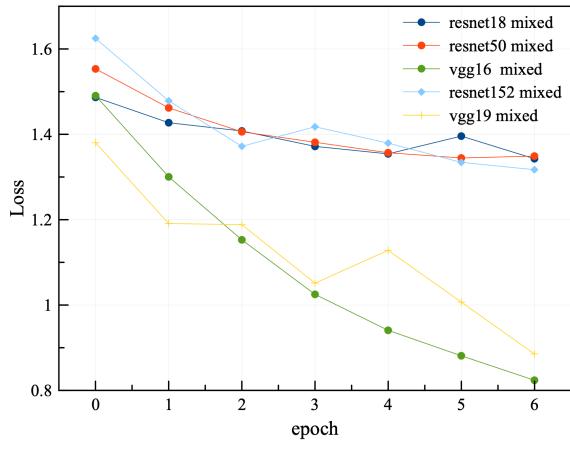
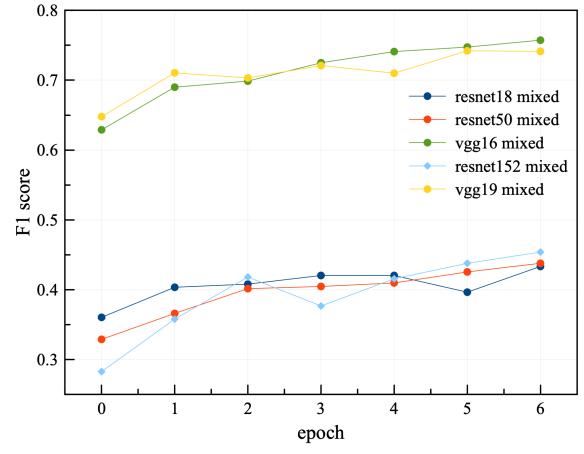


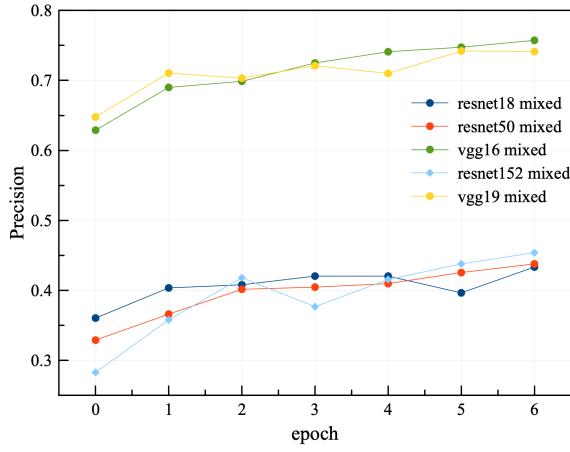
Рис. 14: Сравнение моделей на обучающей выборке: а) функция потерь Loss в зависимости от номера эпохи, б) метрика качества модели F1 score в зависимости от номера эпохи, в) метрика качества модели Precision в зависимости от номера эпохи, г) метрика качества модели Recall в зависимости от номера эпохи.



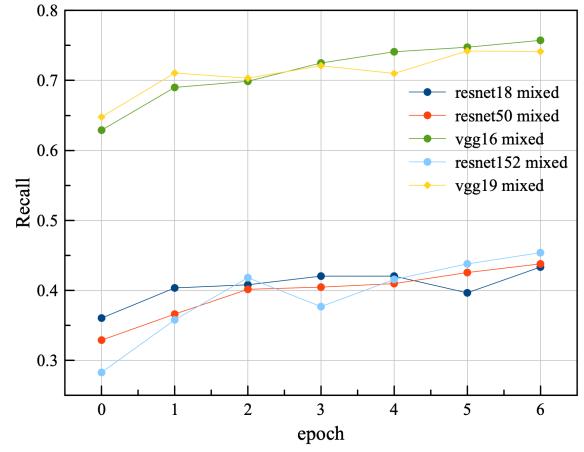
a)



b)

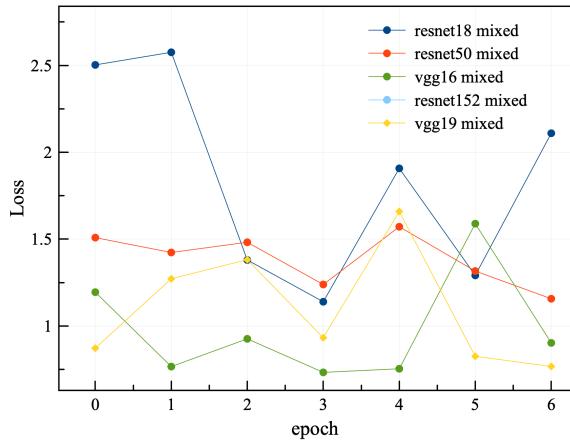


c)

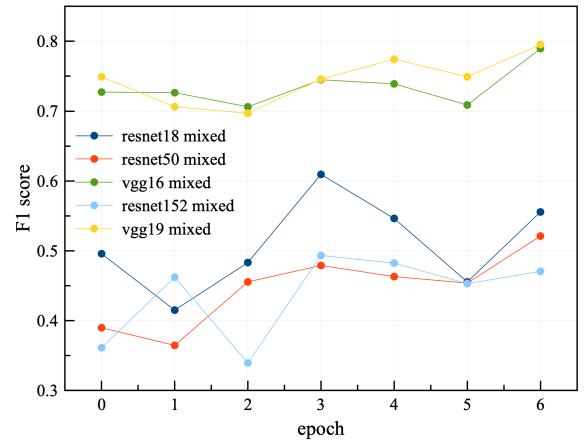


d)

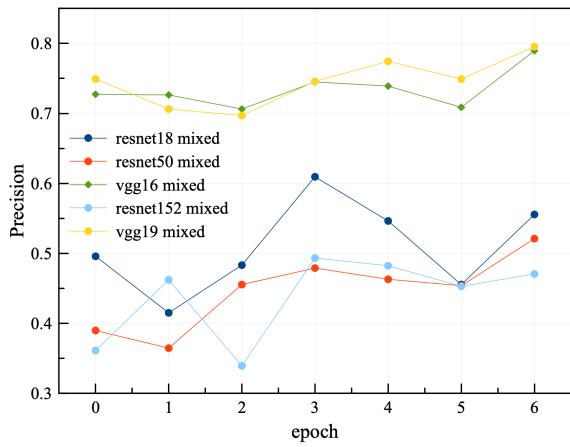
Рис. 15: Сравнение моделей на обучающей выборке для смешанной стратегии обучения: а) функция потерь Loss в зависимости от номера эпохи, б) метрика качества модели F1 score в зависимости от номера эпохи, в) метрика качества модели Precision в зависимости от номера эпохи, д) метрика качества модели Recall в зависимости от номера эпохи.



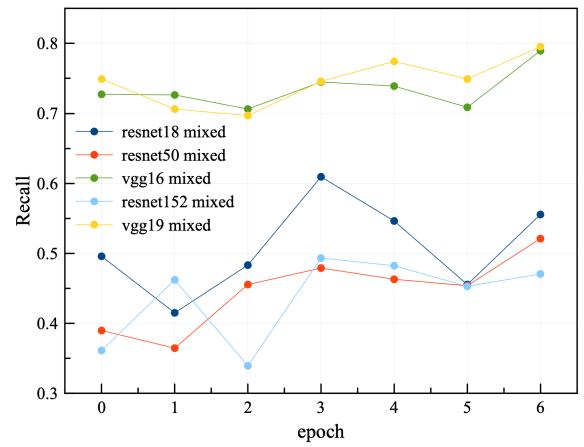
a)



b)



c)



d)

Рис. 16: Сравнение моделей на валидационной выборке для смешанной стратегии обучения: а) функция потерь Loss в зависимости от номера эпохи, б) метрика качества модели F1 score в зависимости от номера эпохи, в) метрика качества модели Precision в зависимости от номера эпохи, г) метрика качества модели Recall в зависимости от номера эпохи.