Санкт–Петербургский государственный университет

Пахомов Максим Юрьевич

Выпускная квалификационная работа

Дистилляция генеративно-состязательных моделей для эффективного улучшения качества изображения

Уровень образования: бакалавриат
Направление 02.03.02 «Фундаментальная информатика и информационные технологии»
ООП СВ.5003.2017 «Программирование и информационные технологии»

Научный руководитель:
доцент кафедры компьютерных технологий и систем,
к.ф.- м.н., Погожев С. В.

Рецензент:
ассистент кафедры математического моделирования
энергетических систем, Вольф Д. А.

Санкт-Петербург
2021 г.

Содержание

Термины и сокращения

В настоящей выпускной квалификационной работе применяются следующие термины с соответствующими определениями:

- Нейронная сеть – математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма.
- Задача Super-Resolution – это класс методов, которые повышают разрешение системы формирования изображения.
- GAN – алгоритм машинного обучения без учителя, построенный на комбинации двух нейронных сетей, одна из которых генерирует образцы, а другая старается отличить правильные образцы от неправильных.
- Сверточная нейронная сеть (CNN) – специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения.
- Perceptual loss – мера, которая показывает разницу промежуточных представлений двух изображений, взятых со слоев нейросети, ответственных за высокоуровневые признаки.
- Knowledge Distillation – метод, применяемый к нейросетевым моделям, позволяющий переносить часть знаний более крупной модели на маленькую модель.
- Peak Signal-to-Noise Ratio – инженерный термин, означающий соотношение между максимумом возможного значения сигнала и мощностью шума, искажающего значения сигнала.
- Structure Similarity – один из методов измерения схожести между двумя изображениями, основанный на учете структурного изменения информации.

Также в данной выпускной квалификационной работе используются следующие сокращения и обозначения:

- SR – Super Resolution,
- GAN – Generative Adversarial Networks,
- CNN – Convolutional Neural Networks,
- SSIM – Structure Similarity,
- PSNR – Peak Signal-to-Noise Ratio,
- BN – Batch Normalization,
- KD – Knowledge Distillation.

Введение

В наше время на смартфон делается огромное количество фотографий, также снимается много различных видео, при этом каждый пользователь хочет, чтобы все моменты были запечатлены как можно более качественно. При этом на качество изображения оказывает влияние огромное количество различных факторов, приводящих к зашумлению изображения. Поэтому сейчас большая востребованность у алгоритмов, которые могут увеличивать разрешение или качество изображения. Данная задача называется задачей Super-Resolution. Методы ее решения используются всеми современными компаниями, например: Google используют их в своем смартфоне Google Pixel, Apple в Iphone, Huawei, Samsung и многие другие. Часто она встречается именно в процессе обработки изображений при фотографировании на смартфон, именно поэтому возникает необходимость иметь быстрые алгоритмы для её решения, так как это всего лишь составляющая большой последовательности действий.

Постановка задачи

Целью данной работы является разработка фреймворка, который позволяет примененить метод дистилляции знаний к выбранной модели ESRGAN [1] с различными параметрами сжатия, а также найти компромисс между ускорением модели и потерями в ее качестве.

Для достижения цели были поставлены следующие задачи:

- выбор датасета для тренировки и валидации, полученных моделей;
- реализация модели для решения задачи SR;
- реализация фреймворка, который позволяет применять метод дистилляции данных для решения задачи SR с использованием модели ESRGAN;
- проведение экспериментов с различными функциями ошибок, коэффициентами уменьшения размера модели;
- формирование вывода о том какие комбинации гиперпараметров являются наиболее подходящими.

Результатом работы должен служить фреймворк, который позволит запустить процесс дистилляции знаний модели ESRGAN, получая на вход путь до весов большой модели, набор функций, которые составляют функцию потерь, коэффициенты при каждой из составляющих, а также коэффициент уменьшения размера модели, и формируя на выходе веса обученной модели, которая сжата в выбранное количество раз. Затем эти веса можно использовать для инициализации уменьшенной модели и использования ее для своих задач.

Обзор литературы

Задача SR существует уже давно [2], поэтому предложены различные нейросетевые подходы к ее решению. Например в [3] используется большое количество dense connections, суть которых заключается в переносе промежуточных представлений с одного слоя блока на все остальные, а также добавление так называемых пирамид Лапласа, где текущее скрытое представление картинки разделяется на три альтернативных пути с разными разрешениями, проходя через которые потом собирается снова и на выходе получается новое скрытое представление картинки. Эти пирамиды используются для так называемого Laplacian Attention (см. Рис. 1).

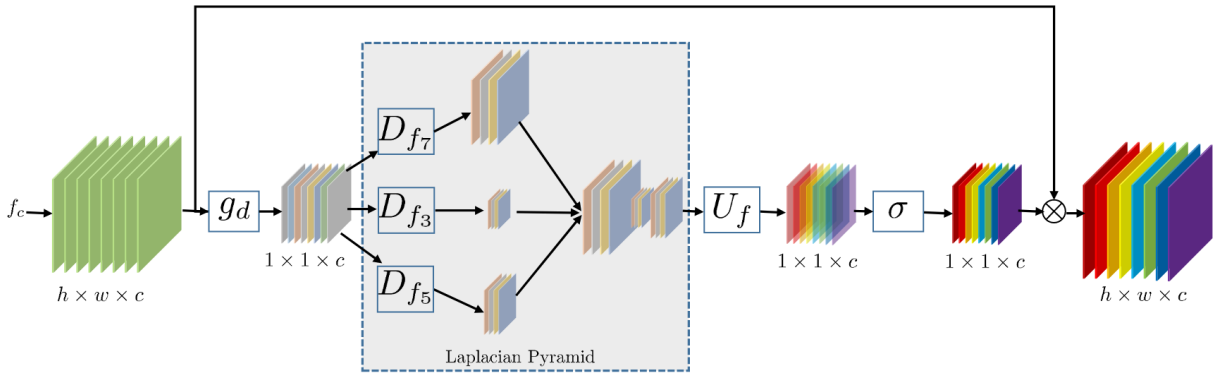

Рис. 1: Архитектура Laplacian Attention блока [3]

Другой подход используется в [4], где ключевая идея состоит в том, чтобы предложить собственный алгоритм уменьшения размерности изображения – Resampler и уже из полученного изображения низкого разрешения пытаться восстановить исходную картинку. Эта часть сети состоит из сверточных слоев и ее задача предсказать фильтры, которые будут применены к исходной картинке высокого разрешения, и смещения для этих фильтров. В качестве сети, которая будет улучшать качество изображения авторы взяли одну из лучших на тот момент моделей – Enhanced Deep Super-Resolution Network (далее EDSR) [5]. Таким образом, получился очень «сильный» алгоритм для сжатия-разжатия изображения, но с задачей увеличения изображений, которые были получены с помощью уменьшения исходного бикубическим алгоритмом, данный алгоритм справлялся хуже большинства. Архитектуру данной модели можно увидеть на Рис. 2.

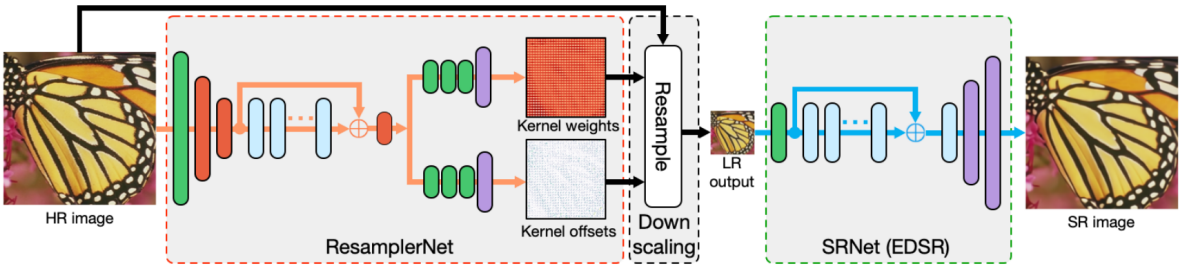

Рис. 2: Архитектура модели Content Adaptive Resampler (CAR) [4]

Подход описанный в [6] использует так называемый GAN фреймворк. В статье были впервые использованы такие понятия как Perceptual Loss и Adversarial Loss в контексте задачи SR. Данная статья положила начало использованию GAN для решения задачи SR.

В [7] описывается способ сопоставления промежуточных представлений изображения при использовании метода Knowledge Distillation. В статье представляется метод, который позволяет сделать одинаковыми по размерности промежуточные представления «студента» и промежуточные представления «учителя», что является основным вкладом данной статьи.

Глава 1. Обзор методов

1.1 Super Resolution со сверточными нейронными сетям

Одним из первых подходов для решения задачи Super Resolution, который актуален до сих пор, было применение сверточных нейронных сетей с простейшей функцией ошибки, которая обычно имеет вид

$$L(W) = \sum_{n=1}^{N} ||NN(x_i) - y_i||_1,$$

где $NN(\cdot)$ – функция, которая выступает как сверточная нейронная сеть, x_i – i-ое изображение в низком разрешении, полученное применением бикубического алгоритма к оригинальному изображению, y_i – i-ое оригинальное изображение.

Таким образом работает алгоритм, представленный в [3]. Прирост в качестве обычно был связан с новыми архитектурными решениями, а не засчет использования новых функций ошибок.

1.2 Generative Adversarial Networks

GAN [8] – это класс методов, которые во время обучения используют две отдельные сети, а также так называемый Adversarial Loss. Одна из них называется генератором и ее суть в том, чтобы генерировать изображения похожие на оригинальные. Вторая называется дискриминатором и ее основная задача в том, чтобы уметь различать сгенерированное генератором изображение от реального изображения из датасета. Во время обучения они конкурируют между собой посредством Adversarial Loss:

$$L_D = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x))],$$
$$L_G = -\mathbb{E}_{x \sim p_{data}(x)}[\log(D_Y(G(x))],$$

где L_D – ошибка дискриминатора, L_G – ошибка генератора, D_Y – функция, которая выступает в качестве дискриминатора, G – функция, которая

выступает в качестве генератора, $p_{data}(y)$ – распределение исходных изображений, $p_{data}(x)$ – распределение шума из которого генерируется изображение, в нашем случае это распределение уменьшенных бикубическим алгоритмом исходных изображений.

Данный класс нейросетей обычно используется в задаче Super Resolution для генерации изображений с четкими границами.

1.3 Enhanced Super Resolution GAN

Далее будет произведено рассмотрение модели, на основе которой базируется данное исследование.

Архитектура Enhanced Super Resolution GAN (далее ESRGAN [1]) основана на GAN фреймворке. Она основана на предыдущей модели, которая также использовала GAN фреймворк – Super-Resolution GAN (далее SRGAN [6]). Авторы избавились от BN [9] слоев, что помогло уменьшить вычислительную сложность нейросети и избавиться от артефактов, которые могли появляться при подаче на сеть изображений с отличающейся статистикой от той, на которой она была обучена. Также они предложили новый вид блоков для этой задачи – Residual-in-Residual Dense Block (далее RRDB, архитектура представлена на Рис. 3).

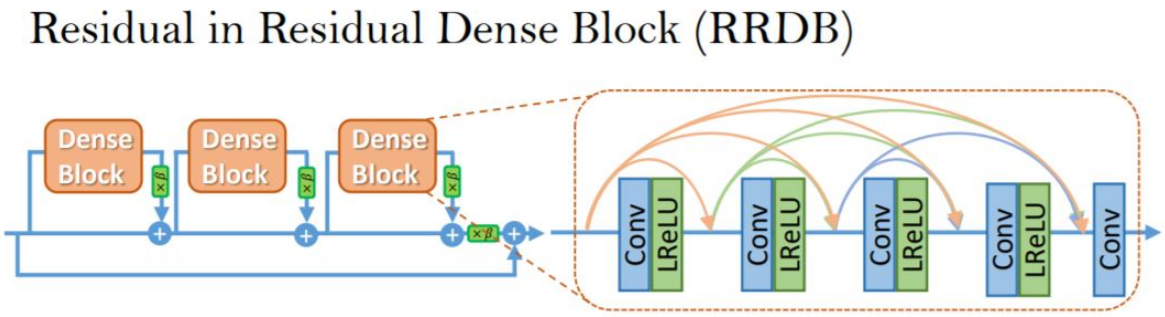

Рис. 3: Архитектура блока RRDB [1]

На Рис. 4 представлена архитектура всей модели. В свою очередь, архитектура дискриминатора остается такой же как и в оригинальном SRGAN, но меняется подход его использования, вместо того, чтобы предсказывать сгенерированное изображение или реальное, дискриминатор сравнивает два изображения между собой и предсказывает, какое из них более

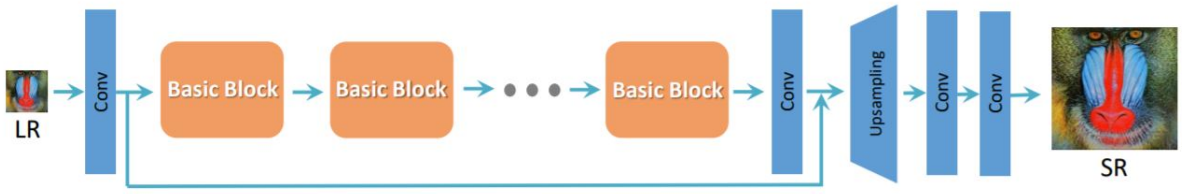

Рис. 4: Архитектура сети ESRGAN

вероятно является настоящим, благодаря этому дискриминатор получает градиенты и от реальных изображений, и от сгенерированных. Последнее важное изменение – взятие скрытых представлений с VGG [10] до активационного слоя, а не после него, для подсчета Perceptual Loss. Авторы утверждают, что на самом деле активируется лишь небольшое количество нейронов и, как следствие, влияние данной функции ошибки является не таким сильным, как хотелось бы.

Итоговая функция ошибки имеет вид:

$$L_G = L_{percep} + \lambda L_{G_{Ra}} + \nu L_1,$$

где L_{percep} – обсуждаемый выше perceptual loss, $L_{G_{Ra}}$ – ошибка генератора от дискриминатора, L_1 – ошибка сгенерированного изображения и исходного. Таким образом, с этими улучшениями на Set5 [11] датасете можно получить PSNR равный 32.73 и на Set14 [12] 28.99 соответственно, что является лучшим результатом среди GAN моделей на текущий момент.

1.4 Knowledge Distillation

Дистилляция знаний (Knowledge Distillation [13]) – метод, позволяющий передавать знания от большой и тяжелой модели, которая обычно называется «учителем», к легковесной и быстрой модели, которую называют «студентом». Далее будут приведены основные идеи, используемые этим методом.

Основная идея дистилляции знаний заключается в том, чтобы научить студента возвращать похожие, в идеале такие же, скрытые представления, при подаче одного и того же входного изображения. Обычно

это скрытое представление берется с самого последнего слоя нейронной сети. Общая картина данного концепта представлена на Рис. 5.

Рис. 5: Идея метода Дистилляция знаний

Однако при попытке сранивать скрытые представления может возникнуть такая проблема как различные размерности скрытых представлений, это проблема до сих пор является открытой и нет какого-то решения, которое работает лучше всех, сам подход сопоставления промежуточных скрытых представлений называется Feature Matching, и его обобщенную схему можно увидеть на Рис. 6.

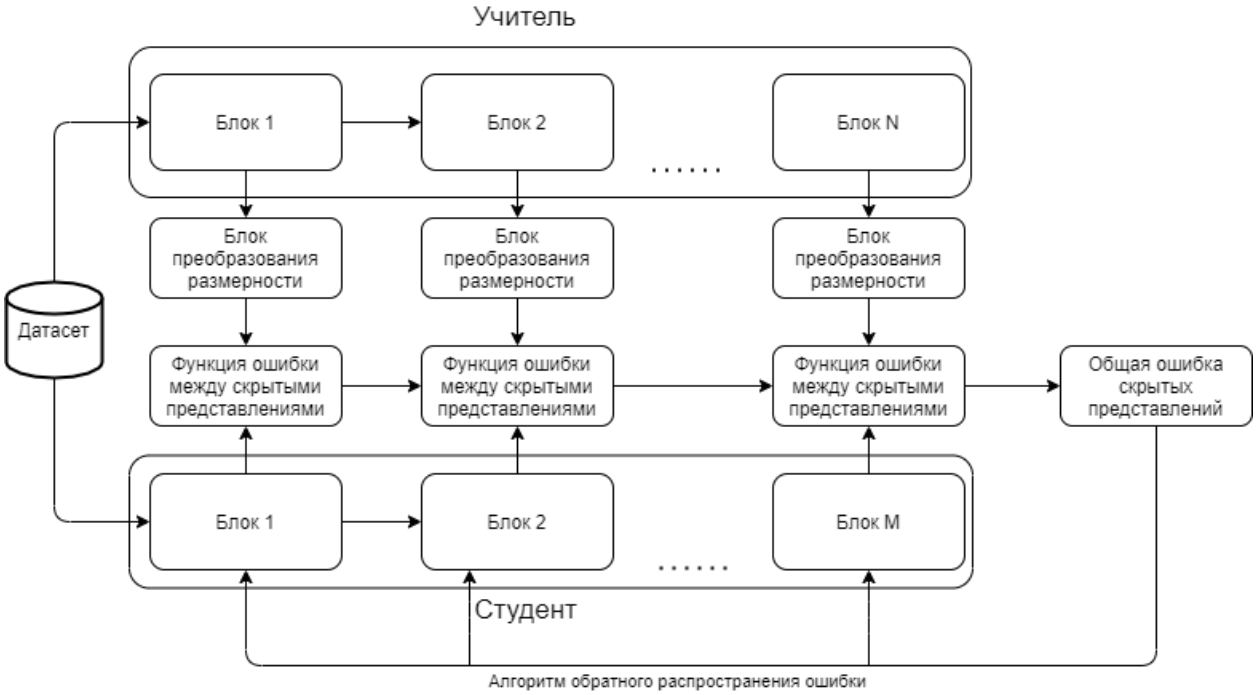

Рис. 6: Схема метода Feature Matching

Его суть заключается в том, чтобы сравнивать также и промежуточные скрытые представления наравне со скрытыми представлениями с последнего слоя, при этом используется блок преобразования размерности скрытого представления модели учителя, чтобы получить скрытые представления того же размера, что и у студента.

1.5 Основные проблемы GAN подхода

У тяжелых нейросетевых моделей, основанных на GAN фреймворке, есть несколько проблем, с которыми данная работа призвана бороться:

- размер модели,
- скорость модели,
- время тренировки,
- стабильность тренировки **??**.

Проблема с размером GAN моделей заключается в том, что данные модели должны обладать сильной обобщающей способностью, вследствие чего эти модели получаются большие по размеру, а маленькие модели обучаются тяжело, либо не обучаются совсем.

Следствием размера модели является ее скорость. И так как в основном GAN модели достаточно большие, то соответственно они относительно медленные.

Процесс тренировки GAN модели может занимать долгое время, так как в ней участвуют две нейросети: генератор и дискриминатор, которые оптимизируются друг под друга, поэтому этот процесс занимает дольше времени чем тренировка одной модели с обычной функцией ошибки.

Как было упомянуто выше, в процессе тренировки участвуют две нейросети. Они оптимизируются друг под друга, чтобы минимизировать свои функции ошибок. Из-за этого если одна из нейросетей начнет «портиться» в каком-то смысле, то от этого пострадает и вторая. Также возможна ситуация, когда генератор найдет способ постоянно обманывать дискриминатор, при этом производя изображения плохого качества. Таким образом,

можно увидеть, что процесс тренировки GAN модели испытывает проблемы со стабильностью.

Глава 2. Алгоритм решения задачи

2.1 Датасеты

В качестве тренировочного датасета был выбран DIV2K [14], он содержит большое количество изображений в высоком разрешении, а именно:

- 800 тренировочных изображений в высоком разрешении с соответствующими уменьшенными версиями: в 2, в 3 и в 4 раза;
- 100 валидационных изображений в высоком разрешении с соответствующими уменьшенными версиями;
- 100 тестовых изображений в высоком разрешении с соответствующими уменьшенными версиями.

Как итог, в нем содержится 1000 изображений в качестве 2K.

В качестве валидационного датасета был выбран Set14 [12], так как он содержит разнообразные по своей природе изображения, что поможет проверить обобщающую способность натренированной нейросети. Он состоит из 14 изображений, в их числе: зебра, ваза с цветами, бабочка, лицо в профиль, бабуин и так далее. Если мы хотим провалидировать предложенный нами метод увеличения разрешение изображения, то используем изображения из датасета в качестве эталонных, оттуда же возьмем изображения уменьшенные в выбранное количество раз с помощью бикубического метода, а затем попробуем восстановить с помощью предложенного нами метода. После этого сможем посчитать одну из метрик, оценивающих качества метода. Кратность увеличения разрешения была взята равной 4, так как это уже более показательное увеличение, чем в 2 раза, но при этом не такое экстремальное как в 8 раз. На Рис. 7 приведен пример изображения из данного датасета.

Рис. 7: Бабочка из датасета Set14

2.2 Метрики

В качестве метрик были выбраны две классические метрики:

- Peak Signal-to-Noise Ratio (PSNR),
- Structure Similarity (SSIM).

PSNR – пиковое отношение сигнала к шуму, показывает соотношение между максимумом значения сигнала и мощностью шума, который этот сигнал искажает. Обычно эту величину считают в логарифмической шкале из-за широкого диапазона значений:

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right),$$

где MAX_I – максимальное значение, которое могут принимать пиксели изображения, обычно это число равно 255, MSE – средний квадрат разности между пикселями исходного и зашумленного изображения.

SSIM – метод измерения похожести изображений, который основывается на идее, что соседние пиксели имеют сильную взаимосвязь, отсюда следует, что окна пикселей могут нести структурную информацию об объектах на изображении или сцене на изображении:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

где μ_x – среднее x, μ_y – среднее y, σ_x^2 – дисперсия x, σ_y^2 – дисперсия y, $\sigma_x y$ – ковариация x и y, c_1 и c_2 – переменные, зависящие от максимального значения пикселей изображения.

2.3 Предложенное решение проблемы

В 1.5 упомянуты проблемы, которые возникают при обучении больших моделей, основанных на GAN фреймворке. Со всеми этими проблемами может помочь метод дистилляции знаний. Далее рассмотрим как дистилляция знаний помогает в решении возникающих проблем, и опишем само решение, которое этот метод использует.

- Размер модели – метод дистилляции знаний был придуман для того, чтобы передавать знания большой модели на маленькую, поэтому уменьшение размеры модели вытекает из самого метода.

- Скорость модели – не всегда, но очень часто за уменьшением модели следует и ее ускорение, с учетом выбранной архитектуры ESRGAN, ускорение следует из уменьшения модели.

- Время тренировки – обычно тренировать нейросетевые модели с нуля может стать достаточно затратным по времени процессом, особенно когда речь заходит о тренировке моделей, основанных на GAN фреймворке, менее затратно натренировать маленькую модель, которой помогает учиться умная и большая, чем тренировать такую же совсем с нуля.

- Стабильность тренировки – как уже было ранее упомянуто, тренировка, основанная на GAN фреймворке, бывает нестабильной, что может привести к нужде перезапуска тренировки, смены гиперпараметров и так далее. В данной ситуации метод дистилляции данных помогает тем, что это прямолинейный подход, который не требует использования GAN фреймворка для тренировки маленькой модели.

Предложенное решение заключается в том, чтобы взять модель для решения задачи SR – ESRGAN и применить данный метод к ней, при этом

позволяя контролировать то, во сколько раз модель будет уменьшена и какие функции потерь с какими коэффициентами будут использоваться при обучении уменьшенной версии модели.

Далее будет описаны шаги, которые позволяют применить метод дистилляции знаний к нейросетевой модели ESRGAN.

В первую очередь, нужно было реализовать возможность одновременного нахождения в памяти как модели учителя, так и модели студента. Более того это должно быть реализовано так, чтобы для модели учителя не считались градиенты, когда через нее проходит изображение, которое используется при тренировке студента. Данная функциональность поддерживается встроенными средствами фреймворка PyTorch, он позволяет отключать вычисление градиентов у модели при ее запуске с помощью применения данной конструкции ко всем весам модели «weights.requires_grad = False», где weights - одна из матриц весов нейронной сети. Также имеется возможность запускать некоторые куски кода без подсчета градиентов совсем, это осуществляется с помощью конструкции «with torch.no_grad():», после этого следует код, который должен быть исполнен без подсчета градиентов. Программную реализацию можно будет увидеть дальше, в псевдокоде тренировки «студента».

Следующим шагом была реализация последовательности действий, которая позволяет подать на вход пути до датасетов DIV2K и Set14, конфигурационные параметры такие как: размер квадрата, который вырезается из исходного изображения, размер тренировочного батча, размер валидационного батча. Данный пайплайн должен возвращать итератор по упомянутым выше датасетам, удовлятворяющий конфигурационным параметрам. Это было реализовано с использованием встроенных классов для обработки данных, а именно: «torch.utils.data.dataset.Dataset», который является абстрактным классом, который позволяет реализовать логику считывания данных, подсчета их количества и взятия отдельного элемента датасета по его индексу, а также «torch.utils.dataloader.DataLoader», который позволяет на вход получается объект типа «torch.utils.data.dataset.Dataset», размер батча, параметр отвечающий за то, чтобы перемешивать данные после каждого прохода по всему датасету или нет, а на выходе возвращает

итератор по этим данным.

Также была необходимость в конфигурационном файле тренировки и возможности его чтения моделью. В нем описано количество RRDB блоков, которые используются в ESRGAN, количество каналов во входном, выходном изображении, то во сколько раз уменьшить модель, а также названия используемых функций ошибки с коэффициентами при них. Данная функциональность была реализована с использованием YAML файла, в котором были записаны все конфигурационные параметры. Затем этот файл читается с помощью Python кода и передается в класс, ответственный за чтение и подготовку данных, а также в класс ответственный за инициализацию моделей. Далее уже на программном уровне заложено то, как эти классы должны распоряжаться полученным словарем из конфигурационных параметров.

Реализация метода дистилляции знаний к этой модели заключается в том, что нужно создать фреймворк, который позволяет управлять количеством RRDB блоков в сети и возможностью сопоставлять RRDB блоки учителя и RRDB блоки студента для подсчета ошибки сопоставления промежуточных представлений, когда у учителя и студента их разное количество. На вход подается конфигурационный файл тренировки, в зависимости от параметра nbShrink, который отвечает за уменьшение модели создается модель-студент, где количество RRDB блоков в nbShrink раз меньше, таким образом достигается выигрыш в производительности. Для обучения полученной модели-студента нужно уметь сопоставлять промежуточные представления модели-учителя и модели-студента, так как в них разное количество блоков. В данной работе это достигается за счёт того, что каждый блок модели-студента сопоставляется nbShrink-овому блоку из модели-учителя, то есть первый блок студента сопоставляется с первым блоком учителя, последний блок студента с последним блоком учителя, а затем i-ый блок студента сопоставляется $n * i$-ому блоку учителя, где n – значение параметра nbShrink.

Для подсчета ошибки сопоставления сжатых представлений сама модель ESRGAN была реализована так, чтобы на выходе было не только изображение в увеличенном разрешении, но и все сжатые представления

изображения, которые выходят из RRDB блоков.

Ниже представлена общая схема реализованного фреймворка (см. Рис. 8)

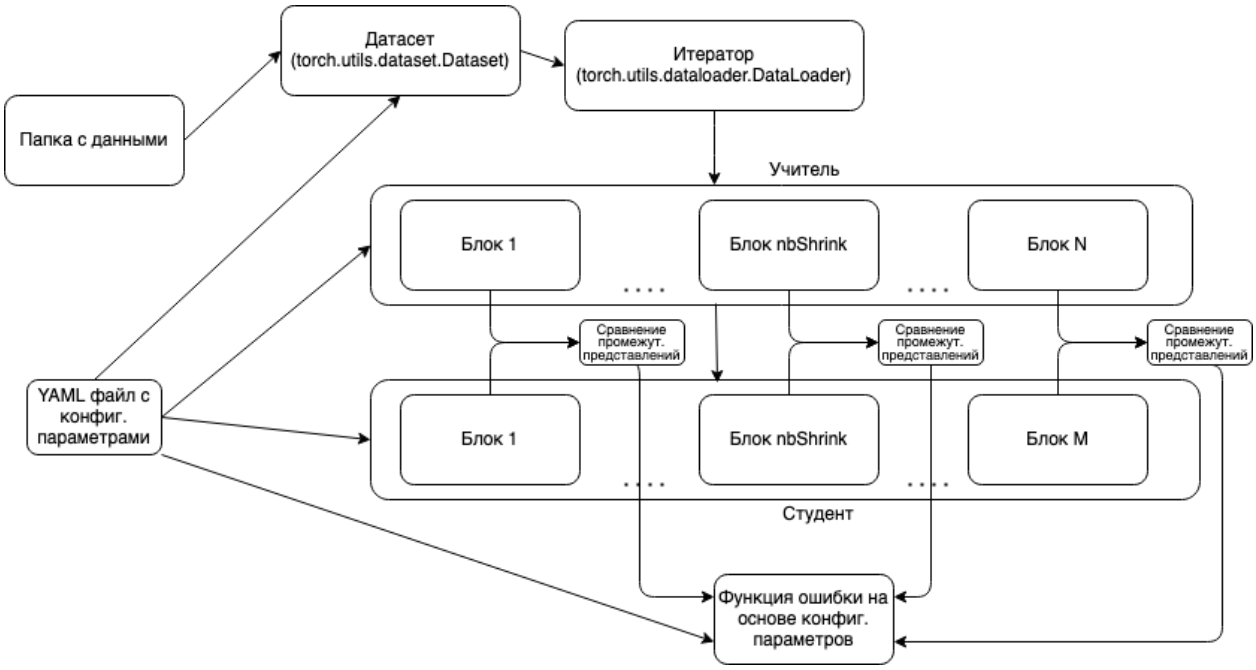

Рис. 8: Общая схема реализованного фреймворка

Далее Алгоритм 1 представляет собой псевдокод для тренировки «студента»:

Algorithm 1: Алгоритм дистилляции знаний

while *idx < len(train_dataloader)* **do**
    optimizer.zero_grad();
    batch, ground_truth = train_dataloader[idx];
    pred_student = student(batch);
    with torch.no_grad():
    | pred_teacher = teacher(batch);
    loss = loss_function(pred_student, pred_teacher,
     ground_truth);
    loss.calc_grad();
    optimizer.step();
end

Описанные выше решения были реализованы с использованием фреймворка для глубокого обучения PyTorch в данном репозитории: `https://`

github.com/AverageName/KD_ESRGAN.

2.4 Используемая функция потерь

Функция потерь для обучения данной модели состоит из следующих составляющих:

- L_1 ошибка между предсказанием студента и предсказанием учителя;
- L_{FM} ошибка сопоставления промежуточных представлений;
- L_1 ошибка между предсказанием студента и изображением в высоком разрешении.
- Также может использоваться L_{LPIPS} составляющая, которая отвечает за то, чтобы предсказания студента и учителя были похожи по смыслу, это делается засчет того, что увеличенные изображения попадают на вход нейросети VGG, затем с определенного ее слоя берутся скрытые представления и сравниваются между собой.

В конечном итоге формула функции потери имеет вид:

$$L = L_{1_{GT}} + L_{1_T} + \lambda L_{FM} + \nu L_{LPIPS},$$

$$L_{FM} = \sum_{n=1}^{M} \frac{1}{H_n W_n C_n} (Out_{student_n} - Out_{teacher_n})^2,$$

где $L_{1_{GT}}$ – L_1 ошибка между предсказанием студента и оригинальным изображением в высоком разрешении, L_{1_T} – L_1 ошибка между предсказанием студента и предсказанием учителя, L_{FM} – ошибка сопоставления промежуточных представлений учителя и студента, L_{LPIPS} – ошибка, отвечающая за соответствие изображений по смыслу.

Глава 3. Эксперименты и результаты

Для тренировки моделей-студентов был использован датасет DIV2K, состоящий из 800 изображений в качестве 2K, и веса, предоставленные авторами статьи [1]. Для валидации полученных результатов был использован датасет Set14 [12].

Таблица 1: Количество параметров в модели и метрики на датасете Set14

Фактор уменьшения	Количество параметров	Метрика PSNR	Метрика SSIM
Оригинал	16.7M	28.72	0.7906
x2	8.1M	28.25	0.7810
x4	3.7M	28.0	0.7729

В Таблице 1 представлены проверенные комбинации уменьшения модели и соответствующее данным моделям качество. Данные метрики считались как среднее значение по валидационной выборке, в данном случае датасет Set14. Здесь были использованы параметры $\nu = 0$ и $\lambda = 10$.

Результаты тренировки модели с параметрами $\nu = 1$ и $\lambda = 10$ можно посмотреть в Таблице 2.

Таблица 2: Количество параметров в модели и метрики на датасете Set14 с использование составляющей L_{LPIPS}

Фактор уменьшения	Количество параметров	Метрика PSNR	Метрика SSIM
Оригинал	16.7M	28.72	0.7906
x2	8.1M	27.7	0.7804
x4	3.7M	27.5	0.7778

Также рассмотрим размер моделей и их ускорения в результате ди-

стилляции знаний (см. Таблица 3 и Таблица 4).

Таблица 3: Размер модели в мегабайтах (Мб)

Фактор уменьшения	размер в мегабайтах (Мб)
Оригинал	65 Мб
x2	31 Мб
x4	15 Мб

Таблица 4: Скорость полученных моделей на изображениях различного размера в миллисекундах

Фактор уменьшения	(96,96)	(128,128)	(196,196)	(224,224)	(256,256)
Оригинал	99.1	103	193	220	200
x2	48.5	49.8	97	111	106
x4	22.4	24.5	48.9	56.9	53.6

Ниже можно увидеть примеры работы моделей из Таблицы 1, а именно той модели, что уменьшена в 2 раза (см. Рис. 9).

Рис. 9: Результат работы моделей: уменьшенная в 2 раза модель, оригинальная ESRGAN модель, оригинальное изображение в высоком разрешении из датасета Set14

Далее приведен пример работы модели, которая в 4 раза меньше оригинальной (см Рис. 10).

Рис. 10: Результат работы моделей: уменьшенная в 4 раза модель, оригинальная ESRGAN модель, оригинальное изображение в высоком разрешении из датасета Set14

Изображения выше были получены с параметром $\lambda = 10$ и $\nu = 0$ при помощью фреймворка для глубокого обучения PyTorch, а модели были натренированы с помощью одной видеокарты NVIDIA Tesla V100.

Выводы

Таким образом, можно сделать некоторые выводы на основе проведенных экспериментов:

- При дистилляции знаний LPIPS функция ошибки пагубно влияет на качество получаемой модели, по крайней мере для весов, которые были выложены авторами статьи.
- Основное падение в качестве происходит при уменьшении размера модели в 2 раза, далее оно уже снижается не так значительно, это можно заметить по Таблице 1 и Таблице 2.
- С точки зрения человеческого глаза достигается компромисс между ускорением модели и падением качества, то есть прирост эффективности модели стоит того, чтобы потерять немного в визуальной составляющей.

Заключение

В ходе научно-исследовательской работы был реализован фреймворк, позволяющий управлять выбором составляющих функций потерь и коэффициентов при них, а также тем, во сколько раз произойдет уменьшение модели.

Были проведены эксперименты по применению метода дистилляции знаний к нейросетевой модели для решения задачи SR – ESRGAN. В результате было получено 2 конкурентоспособных модели: одна из них при размере в 2 раза меньше имеет PSNR равный 28.25 на Set14, когда оригинальная модель имеет 28.72, но визуально их отличить достаточно сложно. Вторая имеет размер в 4 раза меньше при PSNR равном 28.0. Данные эксперименты были проведены с параметрами $\lambda = 10$ и $\nu = 0$. Они были подобраны экспериментальным путем и считаются оптимальными.

Более того, метод дистилляции знаний универсален, поэтому его можно применить и к другим нейросетевым решениям задачи SR. Что касается этого решения, то еще есть много вещей с которыми можно проэкспериментировать и добавить: попробовать другие составляющие функции ошибки, коэффициенты при них, способы сопоставления промежуточных представлений модели-студента и модели-учителя.

Список литературы

[1] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, «ESRGAN: Enhanced super-resolution generative adversarial networks», 2018.

[2] M. Irani, S. Peleg, «Super resolution from image sequences», 1990.

[3] Saeed Anwar, Nick Barnes, «Densely Residual Laplacian Super-Resolution», 2019.

[4] Wanjie Sun, Zhenzhong Chen, «Learned Image Downscaling for Upscaling using Content Adaptive Resampler», 2019.

[5] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, Kyoung Mu Lee, «Enhanced Deep Residual Networks for Single Image Super-Resolution», 2017.

[6] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, «Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network», 2017.

[7] Zibin He, Tao Dai, Jian Lu, Yong Jian, Shu-Tao Xia, «Fakd:Feature-Affinity Based Knowledge Distillation for Efficient Image Super-Resolution», IEEE International Conference on Image Processing, 2020.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, BingXu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio, «Generative adversarial nets», 2014.

[9] Sergey Ioffe and Christian Szegedy, «Batch normalization:Accelerating deep network training by reducing internal covariate shift». In Proceedings of the 32nd International Conference on International Conference on Machine Learning, 2015, pp. 448–456.

[10] Karen Simonyan, Andrew Zisserman, «Very Deep Convolutional Networks for Large-Scale Image Recognition», 2015.

[11] Bevilacqua, «Low-complexity single-image super-resolution based on nonnegative neighbor embedding», 2012.

[12] Zeyde, R., Elad, M., Protter, M. «On single image scale-up using sparse-representations». International Conference on Curves and Surfaces, 2010, Springer.

[13] Geoffrey Hinton and Oriol Vinyals and Jeff Dean, «Distilling the Knowledge in a Neural Network», 2015.

[14] Agustsson, E., Timofte, R. Ntire «Dataset and study». Challenge on single image super-resolution, CVPRW, 2017.

[15] Николенко С.И., Кадурин А.А. «Глубокое обучение. Погружение в мир нейронных сетей».