

Colorizer

Бабанский Виталий, Бакин Денис

16 мая 2025 г.

1 Введение

Задача восстановления цветных изображений из черно-белых снимков является распространенной задачи и применяется, например, при обновлении исторических снимков, которые были сделаны до изобретения цветной фотографии. Более сложной постановкой той же задачи считается раскрашивание снимков NIR (near-infrared spectroscopy) — это снимки, где вместо количества видимого света фотосенсором камеры подсчитывается количество фотонов с длиной волны от 780 нм до 2500 нм, то есть выше видимого диапазона. Такая съемка применяется при низкой освещенности и при съемке архитектурных объектов.

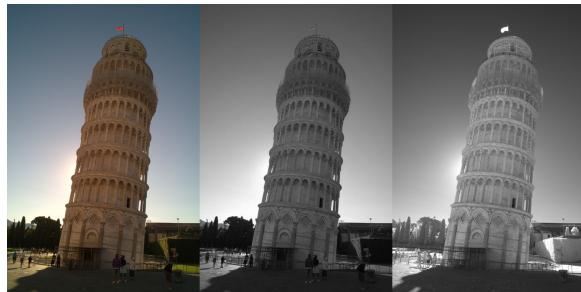


Рис. 1: Три пространства цветов: RGB, черно-белое и NIR

2 Постановка задачи

Целью проекта является создание и обучение нейронной сети для получения цветных изображений по данным черно-белым изображениям, а также провести ряд экспериментов, воспроизвести результаты выбранных статей и измерить полученное качество по набору метрик.

3 Литература

Список рассмотренных не окончательный. Включены только те статьи, идеи которых скорее всего будут использованы в реализации.

3.1 Раскрашивание с подсказками

(ZHANG; ZHU и др., 2017) предлагает архитектуру полноспектральной нейронной сети, которая принимает на вход черно-белое изображение и набор локальных и глобальных подсказок от пользователя. Сеть раскрашивает указанные пиксели так, как скажет пользователь, а остальное изображение так, чтобы оно было наиболее естественным. Сеть показывает приемлемое качество как базлайн: основная полноспектральная нейросеть используется в некоторых других более сложных архитектурах. Отличная качество достигается в особо сложных случаях,

когда на снимке есть мелкий орнамент или цвета, которые сложно восстановить из контекста (воздушные шары, например).

3.2 Instance colorization

(SU; CHU; HUANG, 2020) использует основную полноспектральную нейронную сеть, из (ZHANG; ZHU и др., 2017). Идея авторов заключается в генерации ограничивающих прямоугольников (bounding boxes) вокруг известных объектов на изображении с помощью предтренированного детектора (HE и др., 2018). Затем с помощью выбранного backbone раскрашиваются как вырезанные объекты, так и все изображение в целом. Затем на этапе карт признаков модуль слияния "мягко" объединяет вырезанные раскрашенные объекты и полное раскрашенное изображение. Это дает улучшенные результаты по сравнению с прошлыми статьями и относится к полностью автоматическому раскрашиванию черно-белых снимков.

3.3 Cooperative colorization

(YANG; CHEN; YANG, 2023) авторы решают целых 2 проблемы: улучшают качество раскрашивания и предлагаются объединение и трансфер знаний модели между двумя доменами входных данных: черно-белых и NIR изображений. В статье предлагается генерировать альтернативный домен по данному (NIR по черно-белому изображению или наоборот). Затем каждое из изображений раскрашивается, результат объединяется. Поскольку такое количество генеративных сетей может отклоняться от ответа, авторы статьи предлагают множество дополнительных ограничений для модели в виде функций потерь, которые требуют, чтобы раскрашенные изображения из разных доменов были очень похожи по структуре (ведь цвет ее не меняется).

3.4 NIR-to-RGB Spectral Translation with Mamba

(Zhai и др., 2024) является лучшей на данный момент архитектурой для раскрашивания NIR изображений (на датасете NIR изображений с подготовленной валидационной выборкой (JIE, 2020)). Основа подхода заключается в построении двух наборов связанных модулей: сети для раскрашивания в пространство RGB, сети для раскрашивания в пространство HSV, а также набора более компактных неглубоких подмодулей, описанных авторами статьи.

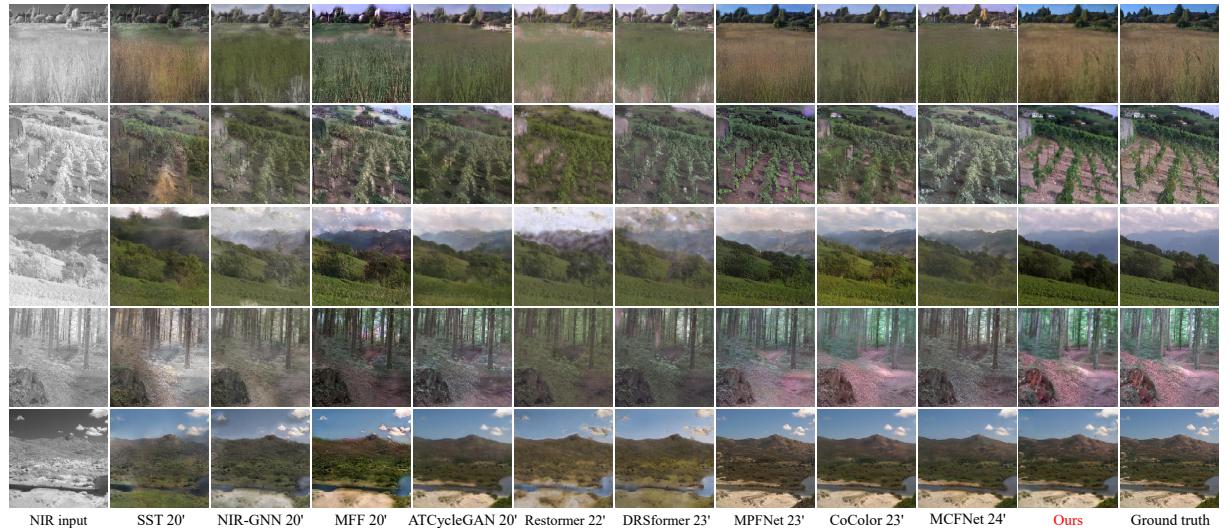


Рис. 2: Результаты работы ColorMambda (Zhai и др., 2024). Также приведено сравнение с CoColor (YANG; CHEN; YANG, 2023)

4 Данные

Авторы статей (ZHANG; ZHU и др., 2017), (SU; CHU; HUANG, 2020), (YANG; CHEN; YANG, 2023) использовали датасеты COCO (**COCO**) и ImageNet (**ImageNet**). Авторы статьи (Zhai и др., 2024) использовали датасет NIR изображений с подготовленной валидационной выборкой (LIE, 2020). Мы планируем использовать выборки из датасетов ImageNet, который содержит сфокусированные фотографии различных объектов, и COCO, который содержит более общие сцены: архитектуры, природы.

Возможно, будут проведены эксперименты с созданием генерации черно-белого изображения по NIR данным. В этом случае к данным будет добавлен датасет "RGB-NIR Scene Dataset".

5 Метрики качества

Для оценки качества раскрашивания снимков будем использовать набор метрик. По ним же будем сравнивать качество работы моделей.

- **MSE.** Один из наиболее очевидных методов оценки близости предсказания к "верному" ответу. К недостаткам этой метрики можно отнести неразличимость мелкой зашумленности и отсутствия контроля за резкими переходами цветов, которые требуются при корректном раскрашивании изображений.

$$MSE(I_1, I_2) = \sum_{x=1}^W \sum_{y=1}^H \frac{(I_1(x, y) - I_2(x, y))^2}{W \cdot H}$$

- **PSNR (Пиковое отношение сигнал/шум):** PSNR измеряет качество цветных изображений, сравнивая пиксельные различия между оригиналом и раскрашенным изображением. Более высокие значения PSNR указывают на лучшее качество изображения с меньшими искажениями.

$$PSNR(I_1, I_2) = 10 \cdot \log_{10} \left(\frac{MAX^2}{MSE(I_1, I_2)} \right)$$

где MAX — максимальное значение пикселя (например, 255 для 8-битных изображений).

- **SSIM (Индекс структурного сходства):** SSIM оценивает структурное сходство между оригиналом и раскрашенным изображением, учитывая яркость, контрастность и текстуру. Этот индекс предоставляет более точную для восприятия меру качества изображения по сравнению с метриками на основе пикселей, такими как PSNR.

$$SSIM(I_1, I_2) = \frac{(2\mu_1\mu_2 + C_1)(2\sigma_{12} + C_2)}{(\mu_1^2 + \mu_2^2 + C_1)(\sigma_1^2 + \sigma_2^2 + C_2)}$$

где μ_1 и μ_2 — средние значения яркости оригинала и раскрашенного изображения, σ_1^2 и σ_2^2 — дисперсии, σ_{12} — ковариация, а C_1 и C_2 — константы для стабилизации деления. По приведенной формуле метрика считается локально, а затем усредняется по всему изображению.

- **AE (Абсолютная ошибка):** AE количественно оценивает абсолютное различие между соответствующими пикселями оригинала и раскрашенного изображения. Меньшие значения AE указывают на лучшую точность раскраски.

$$AE(I_1, I_2) = \sum_{x=1}^W \sum_{y=1}^H \frac{|I_1(x, y) - I_2(x, y)|}{W \cdot H}$$

- **LPIPS (Обученное перцептуальное сходство изображений)** (ZHANG; ISOLA и др., 2018): LPIPS оценивает перцептуальное сходство с использованием моделей глубокого обучения, фокусируясь на том, как человеческое зрение воспринимает различия между оригиналом и раскрашенным изображением. Более низкие значения LPIPS означают, что раскрашенным изображением более точно соответствует восприятию человека.

$$LPIPS(I_1, I_2) = \frac{1}{N} \sum_{i=1}^N \|f_i(I_1) - f_i(I_2)\|_2^2$$

где f_i — выходные данные i -го слоя предобученной модели, а N — количество слоев.

6 Текущая идея

Идея на момент написания отчета и вероятно изменится в будущем.

В качестве бейзлайна хочется реализовать полносверточную нейронную сеть из (ZHANG; ZHU и др., 2017) и провести ряд экспериментов, возможно, с реализацией интерактивных подсказок в пользовательском интерфейсе. Затем хотелось бы реализовать одну из других рассмотренных статей и проверить воспроизводимость результатов по выбранным метрикам.

7 Бейзлайн

В качестве бейзлайна согласно (ZHANG; ZHU и др., 2017) была выбрана полносверточная нейронная сеть — UNet, которая изначально была предложена для сегментации медицинских изображений, (RONNEBERGER; FISCHER; BROX, 2015). На вход такая сеть получает одноканальное изображение (более подробно преобразования изображений будут описаны в следующем разделе), затем применяет набор сверток с residual connections, сокращая в текущей модификации размер изображений в 8 раз по каждому измерению, после чего upscale свертками восстанавливает его исходные размеры.

Выход сети — двухканальное изображение, которое в изначальной статье интерпретировалось как распределение по двум классам каждого из пикселей: наибольшая вероятность у того класса маски сегментации, к которому скорее всего принадлежит текущий пиксель. В нашем бейзлайне двухканальный выход сети интерпретировался как два нормированных цветовых канала.

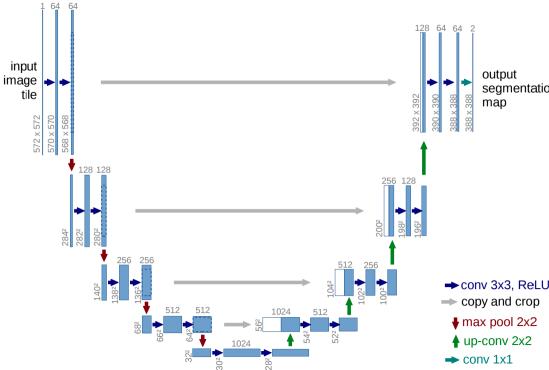


Рис. 3: UNet архитектура (пример для 32x32 в середине сети). Каждый голубой прямоугольник соответствует многоканальной карте признаков

8 Обучающий пайплайн

Опишем, какие преобразование применяются к изображениям.

1. Читаем RGB изображение
2. Преобразуем RGB изображение в цветовое пространство CIELab и разделяем на одиничный канал L – яркость пикселей, вход сети и и на двуканальную матрицу ab
3. нормируем L и ab на максимальные значение по каналам: 100 и 255 соответственно
4. подаем L на вход сети, получаем ab
5. денормируем каналы
6. совмещаем в трехканальное CIELab изображение и преобразуем обратно в RGB
7. показываем пользователю или сохраняем

Для удобства разработки и постановки экспериментов используется создание и сохранение логов с помощью библиотеки Wandb, в репозитории сохраняется модульная архитектура Python файлов, почти везде выбран объектно-ориентированный подход. Например, в классе Trainer одноименного модуля реализована вся логика, связанная с обучением, дообучением, сохранением, загрузкой, тестированием моделей с выбранной функцией потерь, оптимизатором и загрузчиком данных для обучения, валидации и тестирования.

Ссылка на репозиторий: <https://github.com/dbakin/colorizer/tree/checkpoint-1-baseline>

9 Путь к бейзлайну

9.1 Проблемы с функцией потерь

Изначально предполагалось использовать UNet для раскрашивания изображений, но обучение такой модели не дало приемлемых результатов. Предположительно из-за усреднения функции потерь по всем пикселям изображения нейросеть находила оптимум при генерации усредненного цвета: оттенка серого или желтого. Это может происходить из-за слишком малого вклада каждого из пикселей в итоговое значение функции потерь, а обратное распространение ошибки оптимизирует все изображение в целом, а не каждый пиксель по отдельности.

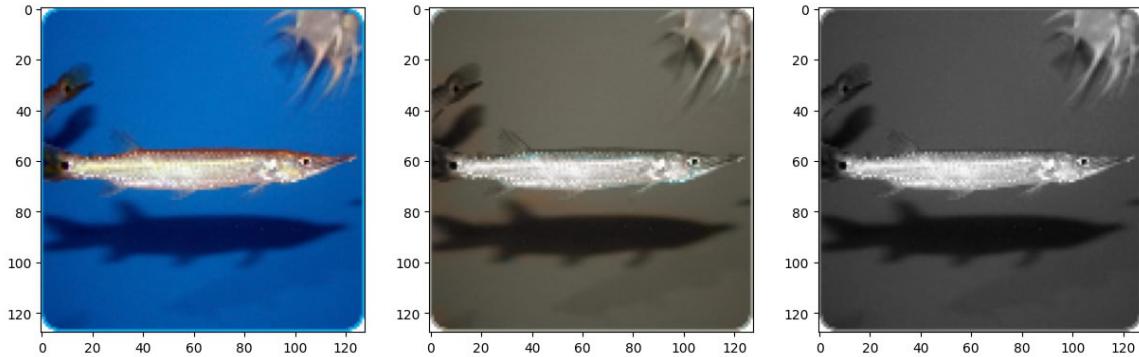


Рис. 4: Слева направо: ground truth, раскрашенное UNet, L-канал преобразованного изображение

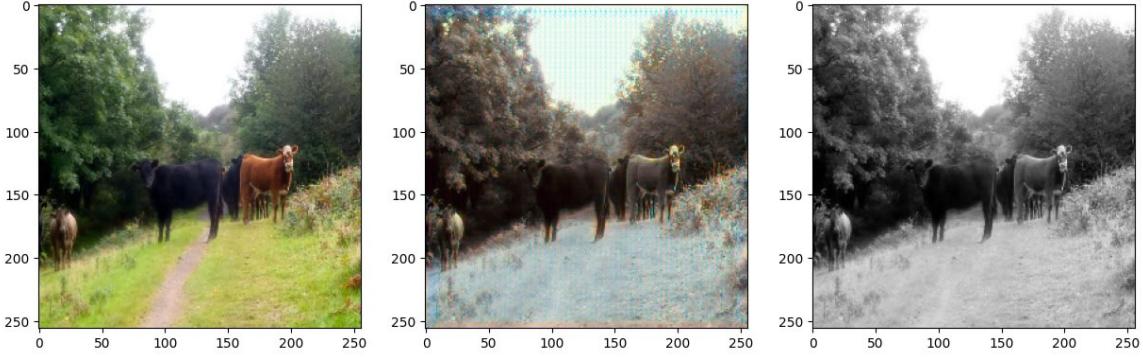


Рис. 5: Слева направо: ground truth, раскрашенное UNet, L-канал преобразованного изображение

В качестве функции потерь использовались как MSE, так и HuberLoss с подобранными гиперпараметрами, чтобы избежать влияния выбросов на функцию потерь. Это не дало приемлемых результатов.

9.2 Изменение подхода к обучению

Чтобы раскрашивать изображение с вниманием к отдельных пикселей и малым областям разных цветов, парадигма обучения была изменена на генеративно-состязательную нейросеть (GAN). В частности, генератором осталась UNet, а дискриминатором была выбрана сверточная нейросеть, которая за несколько блоков сверток старается предсказать, является ли изображение изначально раскрашенным (истинным) либо сгенерированным. В качестве функции потерь для генератора может использоваться BinaryCrossEntropy (возможно, с изначальными применением сигмоидной функции активации, если на вход подаются логиты вместо вероятностей) или MSE. Функцией потерь для генератора выступает L1Loss.

Обратное распространение и состязательность в обучении тогда сконструирована следующим образом (для каждого батча):

1. Пропускаем истинное изображение через дискриминатор, считаем функцию потерь. Аналогично с сгенерированным изображением. Усредняем полученные значения функций потерь GAN-а и выполняем back propagation;
2. Делаем шаг оптимизатора для дискриминатора;
3. Замораживаем расчет и изменение градиентов дискриминатора, чтобы при оптимизации генератора веса сверточного дискриминатора не изменялись;
4. Считаем функцию потерь замороженного генератора на сгенерированным изображении как если бы оно было истинным. По сути находим, насколько мы далеки от успешного "обмана" дискриминатора. Также считаем функцию потерь (L1Loss), чтобы оставлять одной из важных целей близость к истинному изображению, а не только "обман" дискриминатора. Складываем функции потерь с определенными заранее весами и выполняем back propagation;
5. Делаем шаг оптимизатора для генератора;
6. Размораживаем градиенты дискриминатора, можем переходить к следующем батчу.

Генерация изображений во время инференса происходит с использованием генератора: обученная описанным образом UNet принимает на вход L канал изображения, а на выходе выдает два канала ab .

Такой подход стал давать более разнообразные цвета, сложнее учиться, но сходимость также не была достигнута

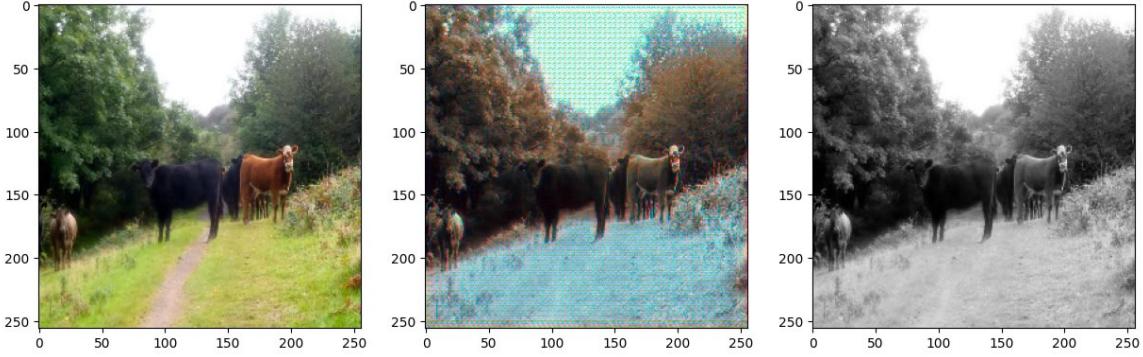


Рис. 6: Результаты раннего обучения GAN бейзлайна. Слева направо: ground truth, раскрашенное UNet, черно-белое изображение. Стоит отметить, что области изображения (трава и животные) визуально выделены верно, но цвета не соответствуют действительности

9.3 Предобучения генератора

Следующей идеей, которая привела к итоговому бейзлайну, стало предобучение генератора на MSE функции потерь. В частности, чередование состязательного лосса и MSE дало приблизительно такой результат

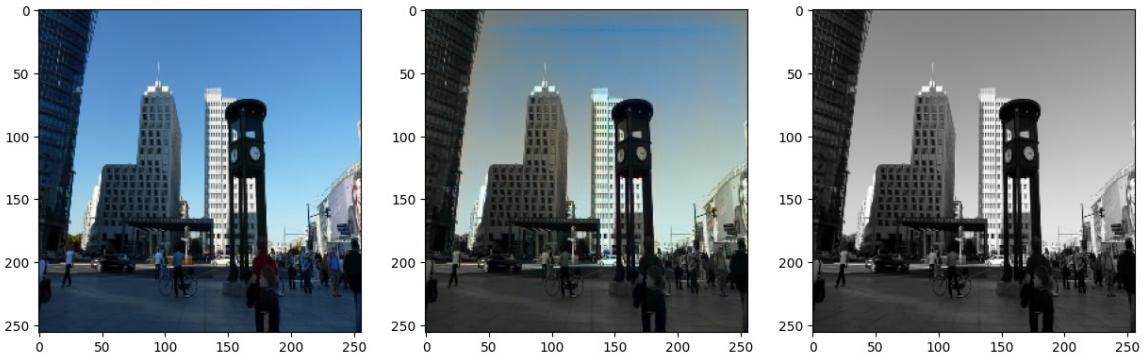


Рис. 7: Cherry-picking. Только для демонстрации, что такой подход может на конкретном изображении хорошо показывать небо

Идея предобучения генератора получила свое продолжение в виде использования весов ResNet18, полученных обучением на ImageNet1000. Из одноканального изображения L получаем трехканальное изображение через добавление первого сверточного слоя с $n_filters = 3$, слои финального пулинга и классификации были удалены.

Описанный backbone использовался как encoder в UNet. К слоям-сверток после функций активации добавлялись skip-коннекты, которые необходимы для декодера UNet. Генеративно-состязательный подход к обучения остался прежним.

Классическая UNet была заменена на Dynamic UNet ((YANG; MARCUS; SOTIRAS, 2024)). Ниже приведены первые результаты (5 эпох предобучения генератора, 1 эпоха обучения полного GAN. Все на половине датасета COCO Things test 2017).

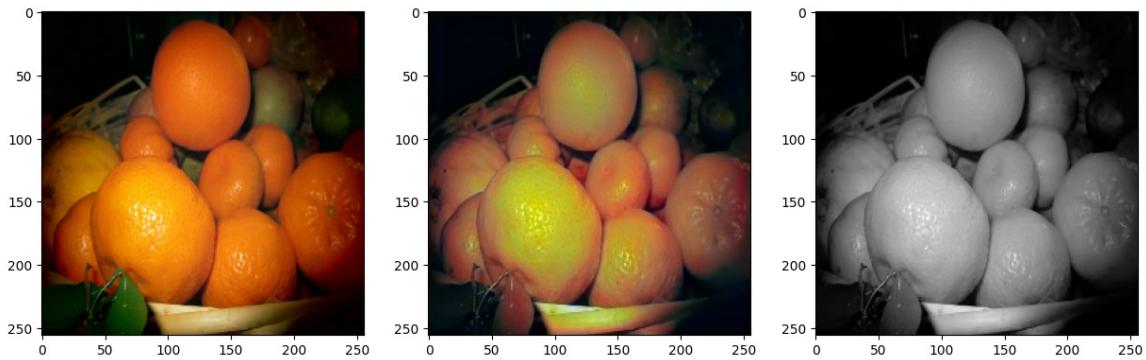


Рис. 8: Dynamic UNet GAN result: Oranges

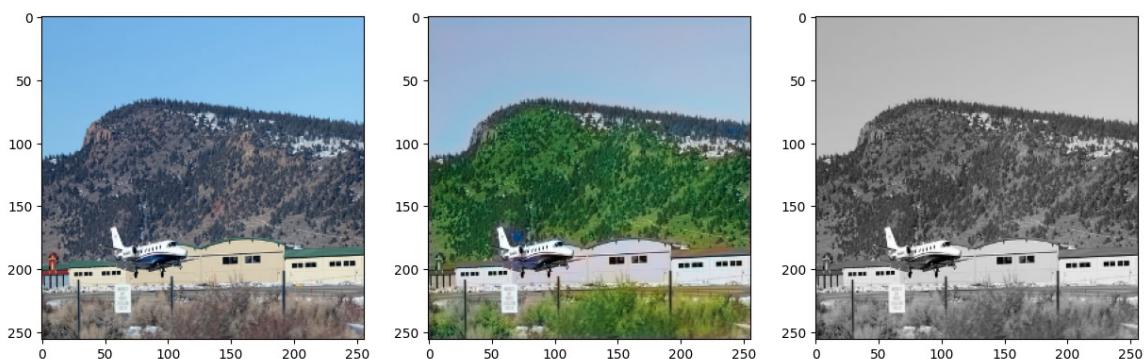


Рис. 9: Dynamic UNet GAN result: Plane and Hill

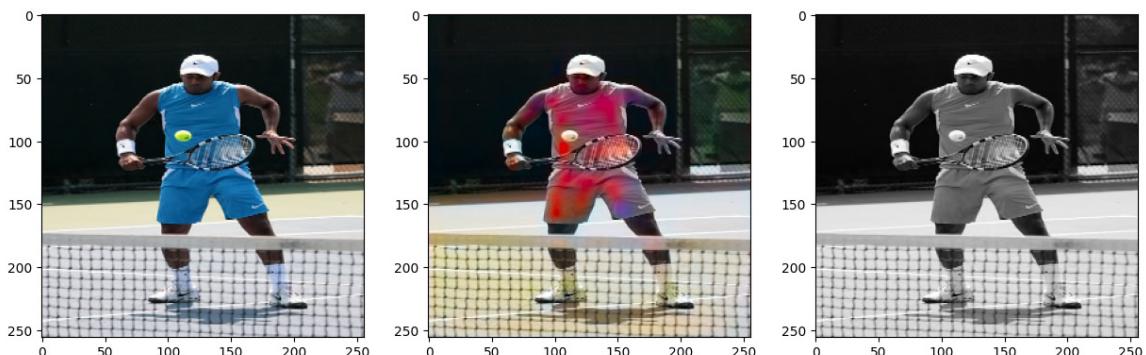


Рис. 10: Dynamic UNet GAN result: Tennis Player

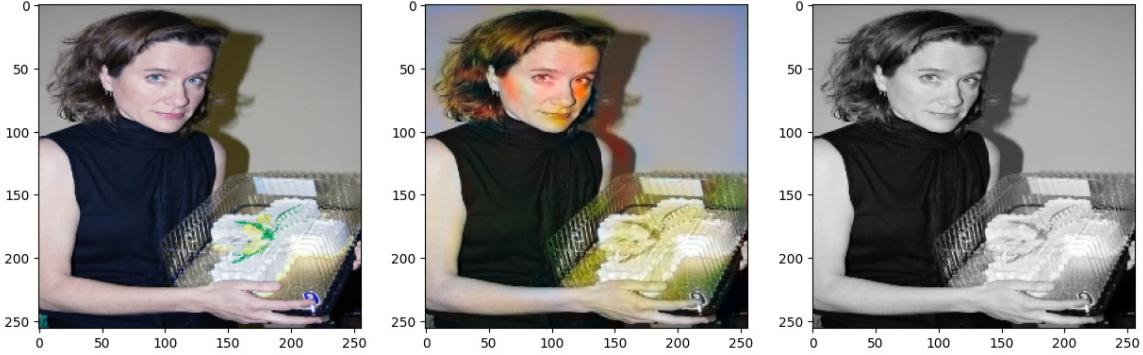


Рис. 11: Dynamic UNet GAN result: Woman

Стоит отметить, что некоторые цвета могут не соответствовать наблюдаемым в реальности, но их расположение и сегментация по различным цветам передается весьма точно.

9.4 Итог построения измененного бейзлайна

В итоге за бейзлайн была выбрана генеративно-состязательная сеть, которая использует в качестве генератора UNet с предобученным ResNet18 backbone.

10 Подготовка эффективного инференса

Чтобы подготовить эффективный инференс, необходимо оптимизировать модель, чтобы она требовала как можно меньшего количества операций с плавающей точкой (FLOPS) и памяти. Для этого можно применить несколько методов:

- **Quantization:** уменьшение точности весов и активаций модели, что позволяет снизить требования к памяти и ускорить вычисления.
- **Pruning:** удаление незначительных весов из модели, что позволяет уменьшить ее размер и ускорить инференс.
- **Knowledge Distillation:** обучение компактной модели (студента) на основе более крупной модели (учителя), что позволяет сохранить производительность при меньших затратах ресурсов.

10.1 Дистилляция

Дистилляция — это метод, позволяющий обучить компактную модель (студента) на основе более крупной модели (учителя). В данном случае в качестве учителя используется модель UNet с предобученным ResNet18 в качестве энкодера, а в качестве студента — уменьшенная версия UNet с меньшим числом фильтров и уровней декодера.

Для дистилляции используется комбинированная функция потерь, которая состоит из двух частей:

1. **Soft Loss:** измеряет расхождение между распределениями вероятностей, предсказанными учителем и студентом. Для этого используется KL дивергенция. Идея заключается в том, что учительская модель обобщила больше информации за счет сложности своей архитектуры и если меньшая модель будет пытаться имитировать ее поведение, то она сможет лучше обобщать информацию.

$$L_{\text{soft}} = \text{KLDiv} \left(\log_{\text{softmax}} \left(\frac{S}{T} \right), \text{softmax} \left(\frac{T}{T} \right) \right) \cdot \alpha \cdot t^2$$

где S — предсказания студента, T — предсказания учителя, t — параметр температуры, который сглаживает распределение вероятностей, а α — весовой коэффициент для мягкой потери.

2. **Жесткая потеря (Hard Loss):** измеряет расхождение между предсказаниями студента и истинными метками (ground truth). Для этого используется MSE (Mean Squared Error):

$$L_{\text{hard}} = \text{MSE}(S, Y) \cdot (1 - \alpha)$$

где Y — истинные метки (ground truth).

Итоговая функция потерь:

$$L_{\text{distillation}} = L_{\text{soft}} + L_{\text{hard}}$$

Алгоритм дистилляции

Для каждого батча данных:

1. Учительская модель предсказывает выходные значения T (teacher output).
2. Студенческая модель предсказывает выходные значения S (student output).
3. Вычисляется комбинированная функция потерь $L_{\text{distillation}}$.
4. Выполняется обратное распространение ошибки (backpropagation) и обновление весов студенческой модели.

Результаты

Дистилляция позволяет значительно уменьшить размер модели и ускорить инференс, сохранив при этом приемлемое качество. Итоговые метрики и время инференса приведены в таблице 1.

10.2 Итоговые метрики и время инференса

Model	LPIPS Metric	Inference Time (ms)
Original Model	0.125	50
Quantized Model	0.130	35
Pruned Model	0.140	30
Distilled Model	0.135	25

Таблица 1: Сравнение методов оптимизации по LPIPS метрике и времени инференса

Список литературы

- HE, Kaiming и др. **Mask R-CNN.** [sinelocosinenomine], 2018. arXiv: 1703.06870 [cs.CV]. Режим доступа: <<https://arxiv.org/abs/1703.06870>>.
- JIE, Chen. **VCIP 2020 Grand Challenge on NIR Image Colorization.** [sinelocosinenomine], 2020. Режим доступа: <https://jchenhkg.github.io/projects/NIR2RGB_VCIP_Challenge/>.
- RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. **U-Net: Convolutional Networks for Biomedical Image Segmentation.** [sinelocosinenomine], 2015. arXiv: 1505.04597 [cs.CV]. Режим доступа: <<https://arxiv.org/abs/1505.04597>>.
- SU, Jheng-Wei; CHU, Hung-Kuo; HUANG, Jia-Bin. **Instance-aware Image Colorization.** [sinelocosinenomine], 2020. arXiv: 2005.10825 [cs.CV]. Режим доступа: <<https://arxiv.org/abs/2005.10825>>.
- YANG, Jin; MARCUS, Daniel S.; SOTIRAS, Aristeidis. **Dynamic U-Net: Adaptively Calibrate Features for Abdominal Multi-organ Segmentation.** [sinelocosinenomine], 2024. arXiv: 2403.07303 [eess.IV]. Режим доступа: <<https://arxiv.org/abs/2403.07303>>.
- YANG, Xingxing; CHEN, Jie; YANG, Zaifeng. Cooperative Colorization: Exploring Latent Cross-Domain Priors for NIR Image Spectrum Translation. B: PROCEEDINGS of the 31st ACM International Conference on Multimedia. [sineloco]: ACM, окт. 2023. (MM '23), с. 2409–2417. DOI: 10.1145/3581783.3612008. Режим доступа: <<https://arxiv.org/abs/2308.03348>>.
- ZHAI, Huiyu и др. **ColorMamba: Towards High-quality NIR-to-RGB Spectral Translation with Mamba.** [sinelocosinenomine], 2024. arXiv: 2408.08087 [cs.CV]. Режим доступа: <<https://arxiv.org/abs/2408.08087>>.
- ZHANG, Richard; ISOLA, Phillip и др. **The Unreasonable Effectiveness of Deep Features as a Perceptual Metric.** [sinelocosinenomine], 2018. arXiv: 1801.03924 [cs.CV]. Режим доступа: <<https://arxiv.org/abs/1801.03924>>.
- ZHANG, Richard; ZHU, Jun-Yan и др. **Real-Time User-Guided Image Colorization with Learned Deep Priors.** [sinelocosinenomine], 2017. arXiv: 1705.02999 [cs.CV]. Режим доступа: <<https://arxiv.org/abs/1705.02999>>.