Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Северо-Восточный федеральный университет имени М.К.Аммосова»

Институт математики и информатики

Кафедра информационных технологий

Выпускная квалификационная работа

02.03.02 «Фундаментальная информатика и информационные технологии»

Выполнил: студент гр. ФИИТ 16, ИМИ, СВФУ

Ефимов Максим Германович

Научный руководитель: Павлов Александр Викторович, доцент кафедры ИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка руководителя)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Якутск, 2020

Содержание

[Введение 3](#_Toc42528417)

[Глава 1. Теория 4](#_Toc42528418)

[Анализ существующих решений 4](#_Toc42528419)

[Algorithmia 4](#_Toc42528420)

[DeOldify 4](#_Toc42528421)

[MyHeritage 4](#_Toc42528422)

[Colourise.sg 4](#_Toc42528423)

[Сравнение результатов 4](#_Toc42528424)

[Вывод 5](#_Toc42528425)

[Основные понятия 5](#_Toc42528426)

[Глубокое обучение 5](#_Toc42528427)

[Свёрточные нейронные сети 6](#_Toc42528428)

[Generative adversarial network (GAN) 6](#_Toc42528429)

[Цветовое пространство LAB 7](#_Toc42528430)

[Feature Loss 7](#_Toc42528431)

[U-Net 8](#_Toc42528432)

[Глава 2. Разработка решения 9](#_Toc42528433)

[Архитектура модели 9](#_Toc42528434)

[Тренировочные данные 9](#_Toc42528435)

[Feature Loss 10](#_Toc42528436)

[Дискриминатор 11](#_Toc42528437)

[Генератор 12](#_Toc42528438)

[GAN 13](#_Toc42528439)

[Цикл тренировки 14](#_Toc42528440)

[Конвертация в CoreML 17](#_Toc42528441)

[Разработка приложения 18](#_Toc42528442)

[Настройка 18](#_Toc42528443)

[Функция обработки 18](#_Toc42528444)

[Обработка результата 19](#_Toc42528445)

[Список литературы 20](#_Toc42528446)

# Введение

**Актуальность:** Первая черно-белая фотография была сделана 1827 году [1], и с тех до распространения цветных камер были сняты миллионы черно-белых фотографий и видео. Также большинство фотографий и документов, таких как книги, газеты, манги и т. п. ради дешевизны по сей день печатаются в черно-белом варианте.

На данный момент черно-белые изображения в основном раскашиваются вручную с помощью программы Photoshop. Это занимает от нескольких часов до нескольких дней и требует специальных умений [2]. Широко известен опыт колоризации черно-белых кинофильмов, отдельные компании утверждают, что используют автоматизацию на основе нейронных сетей, позволяющая выделять области для раскрашивания разными цветами автоматически, что позволяет снизить расходы на колоризацию. Известно, что раскраска каждой минуты сериала 17 мгновений весны в 2009 году обошлась в 3000 долларов за минуту [3].

**Цель исследования:** Разработка приложения под iOS для раскраски черно-белых изображений методами машинного обучения.

**Объект исследования:** Методы машинного обучения.

**Предмет исследования:** Применение нейронных сетей для колоризации изображений.

**Задачи исследования:**

1. Изучить алгоритмы глубокого обучения
2. Сделать модель для раскраски черно-белых изображений
3. Сконвертировать модель в CoreML
4. На основе модели разработать приложение для iOS

# Глава 1. Теория

## Анализ существующих решений

### Algorithmia

Алгоритм разработанный Ричардом Чжаном в 2016 году. Модель представляет из себя feed-forward CNN и тренирована на более чем миллионе фотографий ImageNet. Реализовано на устаревшем на данный момент фреймворке caffe.

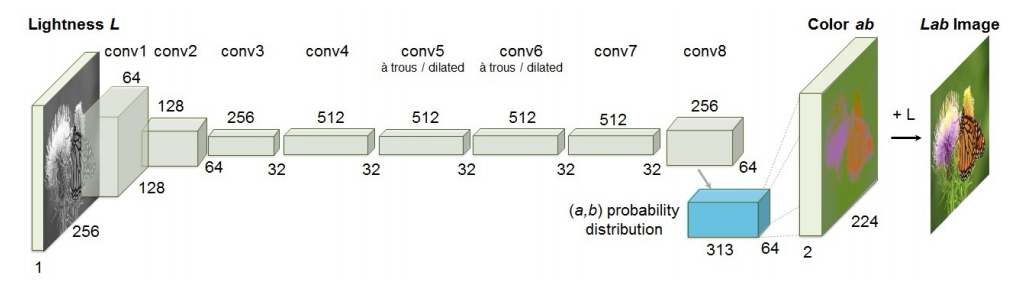
Модель использует цветовую модель CIELab. В качестве входа используется L канал, а в качестве выхода каналы a и b. Минус такого подхода в том что сложно использовать предобученные модели, потому что они, как правило, обучаются с использованием цветовой модель RGB. [4] 

Рисунок 1Архитектура модели Algorithmia

### DeOldify

Модель глубокого обучения с использованием архитектуры GAN. Генеративная модель – это предобученная на imagenet модель U-Net. Функция потерь состоит из двух частей. Первая это Perceptual Loss (Feature Loss) основанная на VGG16. Вторая это функция потерь дискриминационной модели. Реализована с использованием фреймворка Fast.AI.

Модель является весьма требовательным к ресурсам поэтому его нельзя запустить на телефоне без сильной оптимизации. Плюс к этому у фреймворка Fast.AI слабая поддержка конвертиции в CoreML. [5]

### MyHeritage

Платная версия DeOldify, которая по заверениям разработчиков работает лучше [6]. Можно бесплатно обработать до 10 фотографий, далее нужно оформить подписку за 10$ в год.

### Colourise.sg

Бесплатный сайт с закрытым исходным кодом вдохновленный DeOldify. Разработан во время хакатона технологическим отделом правительства Сингапура. [7]

### Сравнение результатов

Будем сравнивать качество существующих решений на нескольких фотографий с сайта Unsplash. Для этого возьмем несколько цветных фотографий, сделаем их черно белыми и обработаем через модели.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Оригинал | Черно-белая версия | Algorithmia | DeOldify | MyHeritage |
| Изображение выглядит как дорога, здание, внешний, улица  Автоматически созданное описание | Изображение выглядит как здание, внешний, дорога, улица  Автоматически созданное описание | Изображение выглядит как здание, дорога, улица, внешний  Автоматически созданное описание | Изображение выглядит как дорога, здание, внешний, улица  Автоматически созданное описание | Изображение выглядит как дорога, здание, внешний, улица  Автоматически созданное описание |
| Изображение выглядит как внутренний, женщина, передний, комната  Автоматически созданное описание | Изображение выглядит как здание, внутренний, фотография, сидит  Автоматически созданное описание | Изображение выглядит как мужчина, женщина, фотография, стоит  Автоматически созданное описание | Изображение выглядит как внутренний, мужчина, стоит, сидит  Автоматически созданное описание | Изображение выглядит как внутренний, мужчина, фотография, женщина  Автоматически созданное описание |
| Изображение выглядит как гора, внешний, трава, снег  Автоматически созданное описание | Изображение выглядит как гора, внешний, снег, природа  Автоматически созданное описание | Изображение выглядит как гора, трава, внешний, природа  Автоматически созданное описание | Изображение выглядит как гора, внешний, природа, фон  Автоматически созданное описание | Изображение выглядит как гора, внешний, трава, снег  Автоматически созданное описание |

### Вывод

Учитывая что MyHeritage и Colourise являются ПО с закрытым исходным кодом, а Algorithmia работает не очень хорошо, самым интересным для исследования является DeOldify. Учитывая сложности конвертации модели DeOldify было решено переписать его оптимизированную под телефоны версию с использование популярного фреймворка Keras, потому что у него хорошая поддержка конвертации в CoreML.

## Основные понятия

### Глубокое обучение

Глубокое обучение — семейства методов машинного обучения, основанных на имитации работы человеческого мозга в процессе обработки данных и создания паттернов. Использует многослойную систему нелинейных фильтров для извлечения признаков с преобразованиями. Каждый последующий слой получает на входе выходные данные предыдущего слоя [8]

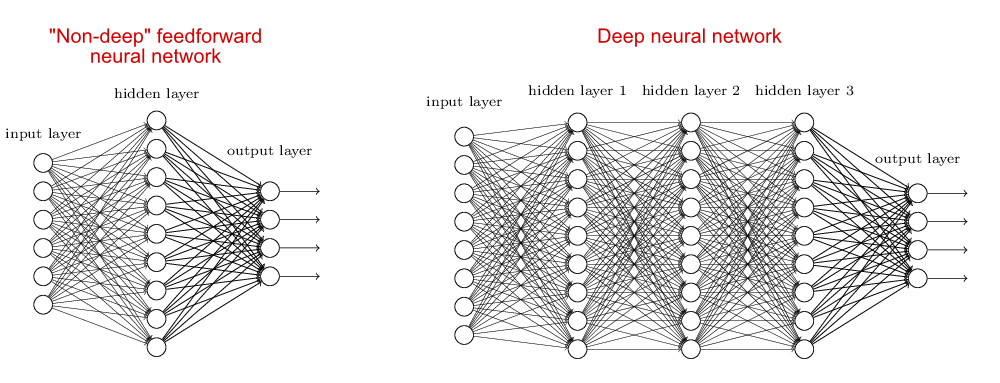


Рисунок 2Глубокие и неглубокие нейронные сети.

### Свёрточные нейронные сети

Сверточная нейронная сеть — специальная архитектура нейронных сетей, предложенная Яном Лекуном, изначально нацеленная на эффективное распознавание изображений.

Две основные концепции в свёрточных нейронных сетях:

* свёртка
* операция подвыборки (pooling, max pooling)

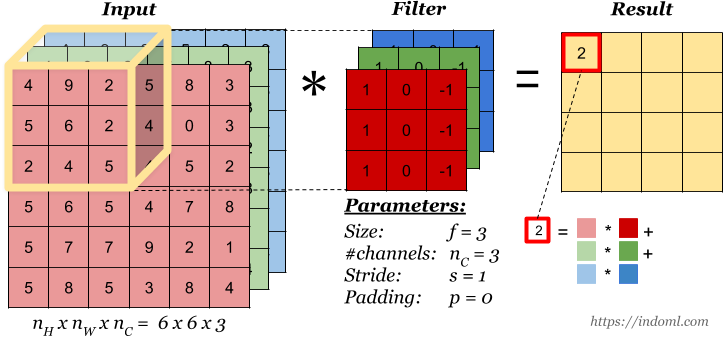
Свёртка — процесс применения фильтра («ядра») к изображению. 

Рисунок 3Пример свертки

Операция подвыборки по максимальному значению — процесс уменьшения размеров изображения через объединение группы пикселей в единое максимальное значение из этой группы.



Рисунок 4Пример пулинга по максимуму

### Generative adversarial network (GAN)

Генеративно-состязательная сеть – это класс моделей машинного обучения без учителя, в котором используется две нейронных сети, одна из которых генерирует новые объекты (*генеративная модель*), а другая старается отличить правильные объекты от неправильных (*дискриминативная модель*). Алгоритм был разработан Яном Гудфеллоу и исследователями из университета Монреаля в 2014 году [9].

#### Применение

GAN используется в генерации изображений, музыки, голоса и текстов, которые человеку сложно отличить от настоящих объектов окружающего мира. Благодаря этому алгоритм нашел применение в методике Deepfake синтеза новых изображений и видео на основе существующих [10].

В последнее время системы GANs стали использоваться для подготовки кадров фильмов или мультипликации. Кроме того, GAN может использоваться для улучшения качества нечётких или частично испорченных фотографий.

#### Как работает GAN

Дискриминативная модель – это стандартная сверточная сеть, которая классифицирует поданные на вход объекты. Генеративная модель – это обратная сверточная сеть, которая на основе случайного шума создает объект

Изображение выглядит как объект

Автоматически созданное описание

Рисунок 5 Архитектура GAN

### Цветовое пространство LAB

LAB – это цветовое пространство разработанное международной комиссией по освещению (CIE) в 1976 году. Оно кодирует цвет тремя числами, *L* – это яркость серого, *a* – зеленый-красный и *b* – синий-желтый оттенки.

LAB был разработан после теории противоположных цветов, согласно которой два цвета не могут быть зелеными и красными в одно время или желтыми и синими в одно время.

Он удобен тем, что в черно-белых изображения присутствует лишь компонента *L*, то есть в модели компонента *L* является входом, по которому мы предсказываем *a* и *b*. [11]

### Feature Loss

Feature Loss – эта техника при которой вместо попиксельного сравнения картинок сравниваются высокоуровневые характеристики картинок. Для этого используется предобученная модель классификации из которой извлекается слой характеристик и генеративная модель старается восстановить эти характеристики. Характеристики представлены числами, минимизируется разница между ними, обычно используется среднеквадратическая ошибка. Предобученная модель во время обучения замораживается. [12]

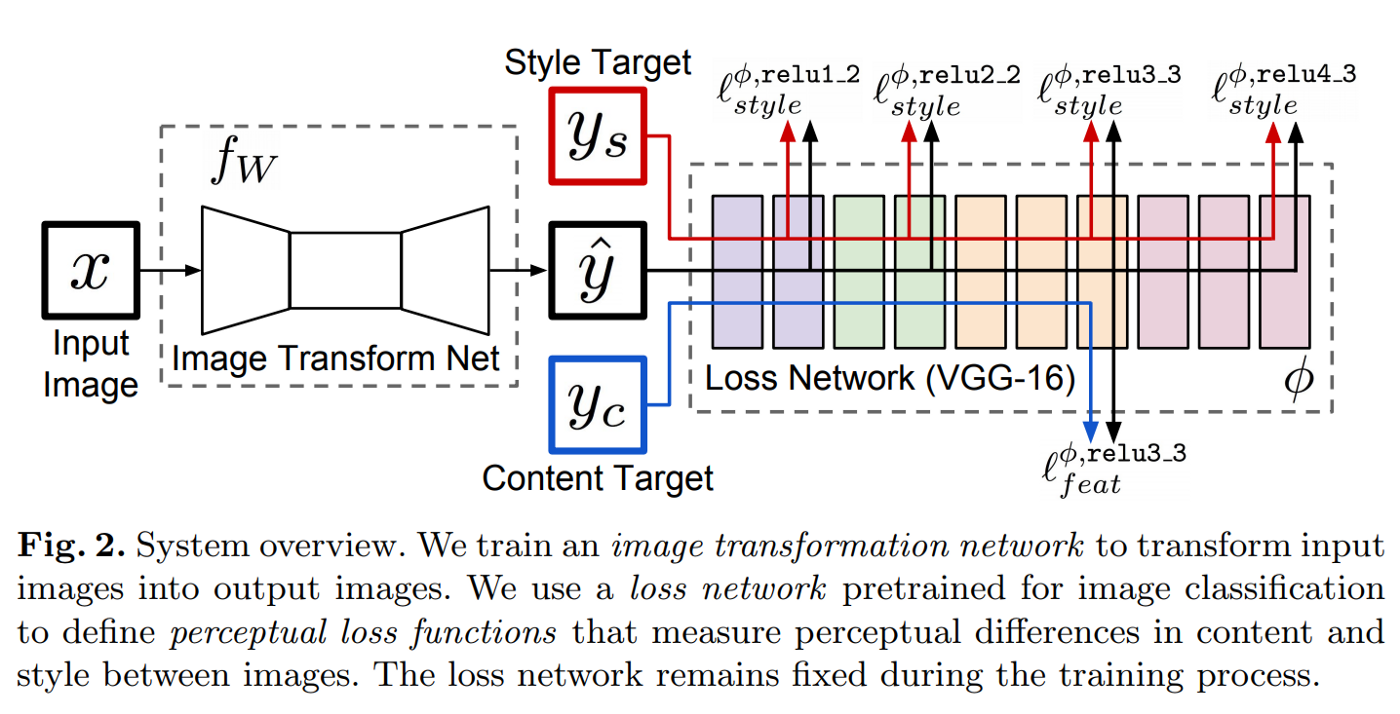


Рисунок 6Feature Loss

### U-Net

U-Net — это свёрточная нейронная сеть, которая была создана в 2015 году для сегментации биомедицинских изображений в отделении Computer Science Фрайбургского университета.

#### Архитектура сети

Сеть содержит сжимающий путь (слева) и расширяющий путь (справа), поэтому архитектура похожа на букву U, что и отражено в названии. На каждом шаге мы удваиваем количество каналов признаков.

Сжимающий путь похож на типичную свёрточную сеть, он содержит два подряд свёрточных слоя 3x3, после которых идет слой ReLU и пулинг с функцией максимума 2×2 с шагом 2.

Каждый шаг расширяющего пути содержит слой, обратный пулингу, который расширяет карту признаков, после которого следует свертка 2x2, которая уменьшает количество каналов признаков. После идет конкатенация с соответствующим образом обрезанной картой признаков из сжимающего пути и две свертки 3x3, после каждой из которой идет ReLU. Обрезка нужна из-за того, что мы теряем пограничные пиксели в каждой свёртке. На последнем слое свертка 1x1 используется для приведения каждого 64-компонентного вектора признаков до требуемого количества классов.

Всего сеть имеет 23 свёрточных слоя. [13]

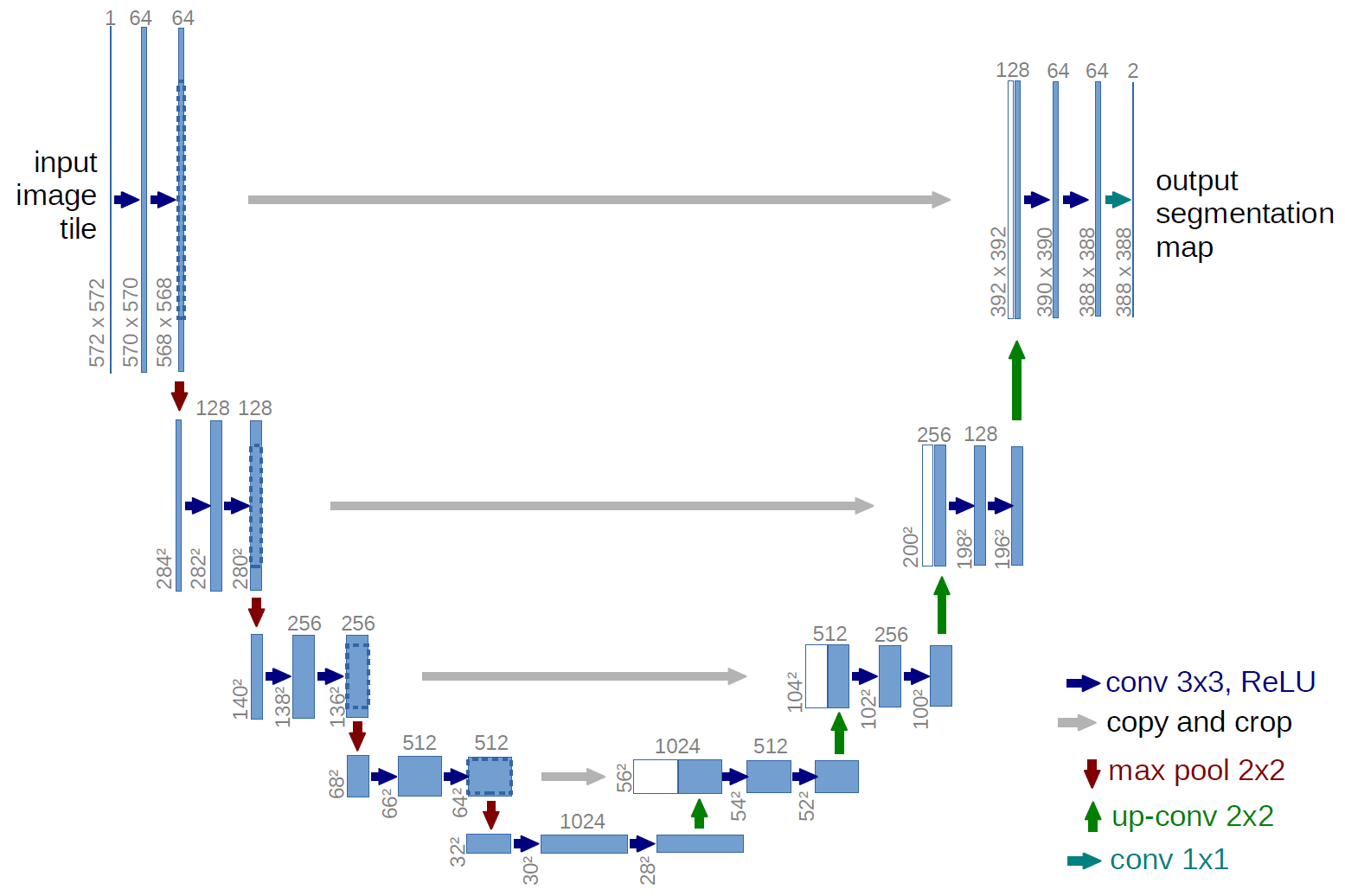


Рисунок 7Архитектура U-Net

# Глава 2. Разработка решения

## Архитектура модели

Модель является глубокой нейронной сетью GAN где генеративная модель это U-Net. Функция потерь это комбинация Feature Loss и потери дискриминатора.

### Тренировочные данные

Тренировочные данные можно получить очень просто – любое цветное изображение может быть преобразовать в черно-белое и можно составить из них пару. Все фотографии нормализуются в диапазоне от -1 до 1.



Figure 1Класс Загрузки данных

### Feature Loss

В качестве модели для feature loss используется модель VGG19 предобученная на датасете imagenet. В качестве output-а используется не вероятности классов, а из него извлекаются слой характеристик. Модель во время обучения GAN не обучается. В качестве функции потерь используется среднеквадратичная ошибка.



Figure 2Модель feature loss

### Дискриминатор

Дискриминатор получает на вход картинку и старается угадать класс картинки: настоящая или сгенерированная. Функция активации на выходном слое сигмоида и указывает на вероятность класса где 0 это сгенерированная картинка 1 настоящая. Функция потерь среднеквадратичная ошибка.



Figure 3Дискриминатор

### Генератор

Генератор это стандартный unet. Выходных слоев 3 соотвественно на каждый канал RGB. Функция активации на выходе тангенс потому что мы нормализовали наши данные в диапазоне от -1 до 1.

def unet(pretrained\_weights = None,input\_size = (256, 256, 3)):

  inputs = Input(input\_size)

  conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(inputs)

  conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv1)

  pool1 = MaxPooling2D(pool\_size=(2, 2))(conv1)

  conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(pool1)

  conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv2)

  pool2 = MaxPooling2D(pool\_size=(2, 2))(conv2)

  conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(pool2)

  conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv3)

  pool3 = MaxPooling2D(pool\_size=(2, 2))(conv3)

  conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(pool3)

  conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv4)

  drop4 = Dropout(0.5)(conv4)

  pool4 = MaxPooling2D(pool\_size=(2, 2))(drop4)

  conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(pool4)

  conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv5)

  drop5 = Dropout(0.5)(conv5)

  up6 = Conv2D(512, 2, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(UpSampling2D(size = (2,2))(drop5))

  merge6 = concatenate([drop4,up6], axis = 3)

  conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(merge6)

  conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv6)

  up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(UpSampling2D(size = (2,2))(conv6))

  merge7 = concatenate([conv3,up7], axis = 3)

  conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(merge7)

  conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv7)

  up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(UpSampling2D(size = (2,2))(conv7))

  merge8 = concatenate([conv2,up8], axis = 3)

  conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(merge8)

  conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv8)

  up9 = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(UpSampling2D(size = (2,2))(conv8))

  merge9 = concatenate([conv1,up9], axis = 3)

  conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(merge9)

  conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv9)

  conv9 = Conv2D(3, 3, activation = 'relu', padding = 'same', kernel\_initializer = 'he\_normal')(conv9)

  conv10 = Conv2D(3, 1, activation = 'tanh')(conv9)

  model = Model(input = inputs, output = conv10)

  #model.summary()

  if(pretrained\_weights):

    model.load\_weights(pretrained\_weights)

  return model

generator = unet(input\_size=lr\_shape)

generator.compile(optimizer = Adam(lr = 1e-4), loss = 'mse', metrics = ['accuracy'])

generator.summary()

### GAN

GAN состоит из трех моделей: генератора, дискриминатора и VGG для feature loss. Сначала черно-белая картинка подается на вход генератору, а результат генератора имеет два пути:

* В первом случае она подается на вход дискриминатору который предсказывает класс картинки. Функция потерь здесь бинарная кросс энтропия.
* Второй путь это feature loss, из результата извлекаются характеристики. Функция потерь здесь среднеквадратичная ошибка.

Притом вес у функции потери feature loss больше, то есть модель будет учитывать его сильнее. Дискриминатор во время обучения GAN не обучается.



Figure 4GAN

### Цикл тренировки

Первым делом загружаются пары черно-белых и цветных фотографий. Потом генератор генерирует цветную версию из черно-белой версии. Оригинальная цветная и сгенерированная цветная подаются на обучение дискриминатору. Далее обучается GAN, ему на вход подается черно-белые фотографии. У GAN два выхода, первый это выход дискриминатора, второй это характеристики. Первый выход сравнивается с единицами, потому что генератору нужно генерировать правдоподобные картинки, второй выход сравнивается с характеристиками оригинальной цветной фотографии.



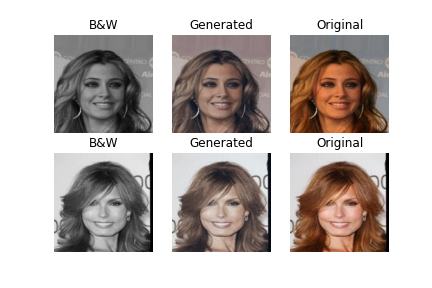
Примеры

Рисунок 8Пример 1

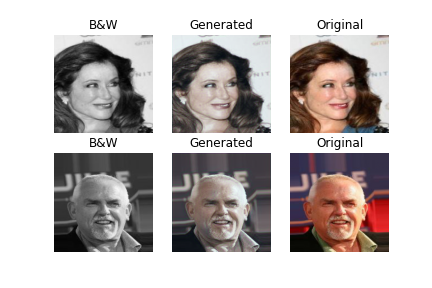


Рисунок 9Пример 2



Рисунок 10Пример 3

## Конвертация в CoreML

Запускать модель на телефоне вместо обработки картинок на сервере имеет много преимуществ. Основные из них:

* Бесплатные вычисления. Аренда сервера, интернет трафик стоят денег, это особенно критично для бесплатных приложений.
* Нет потенциальных проблем с безопасностью. При хранение и пересылки пользовательских данных всегда есть вероятность кражи, утери. При обработке данных на телефоне у Вас гарантированно нет такой проблемы.
* Неограниченная масштабируемость. При большом количестве пользователей сервер может не справиться с нагрузкой.

CoreML – это фреймворк разработанный компанией Apple для интеграции моделей машинного обучение в телефоны операционной системы iOS.

Core ML Community Tools – это библиотека от сообщества для конвертации, редактирование и валидации моделей машинного обучения таких как TensorFlow, Keras, Caffe, scikit-learn и других.



Рисунок 11Конвертация в CoreML

## Разработка приложения

Приложение написано на языке Swift в среде разработке Xcode. В приложение можно загрузить любое черно-белое изображение из библиотеки и получить его раскрашенную с помощью модели версию, которую потом можно сохранить. Для запуска модели и предобработки изображений используется фреймворк Vision.

### Настройка

CoreML модель которая была получена в результате конвертация перетаскивается внутрь проекта Xcode. Классы необходимые для его работы генерируются автоматически CoreML.



Рисунок 12 Инициализация модели

### Функция обработки

Чтобы настроить запрос Vision с использованием модели, создайте экземпляр класса VNCoreMLRequest. В его обработчике завершения вызывается функция обработки результата.

Объекту класса VNImageRequestHandler передается изображение и вызывается метод perform с ранее созданным объектом request. Этот метод выполняется синхронно, поэтому оно запускается в фоновом потоке чтобы приложение не блокировалось во время обработки.



### Обработка результата

Обработчик завершения запроса Vision указывает, был ли запрос успешным или привел к ошибке. В случае успеха результат содержит свойство типа CVPixelBuffer, который может быть сконвертирован в UIImage.



# Список литературы

|  |  |
| --- | --- |
| [1] | Wikipedia, «Хронология фотографии,» [В Интернете]. Available: https://en.wikipedia.org/wiki/Timeline\_of\_photography\_technology. [Дата обращения: 1 2020]. |
| [2] | Vox, «Как одержимые художники раскрашивают старые фотографии,» 2018. [В Интернете]. Available: https://www.youtube.com/watch?v=vubuBrcAwtY&feature=emb\_logo. |
| [3] | Время электроники, «Как раскрашивали «Семнадцать мгновений весны»,» 2009. [В Интернете]. Available: http://www.russianelectronics.ru/leader-r/news/russianmarket/doc/43526/. [Дата обращения: 2020]. |
| [4] | R. Zhang, «Colorful Image Colorization,» 2016. [В Интернете]. Available: https://arxiv.org/pdf/1603.08511.pdf. |
| [5] | J. Antic, «Github,» [В Интернете]. Available: https://github.com/jantic/DeOldify. [Дата обращения: 15 май 2020]. |
| [6] | MyHeritage, «Придайте цвет Вашей семейной истории,» [В Интернете]. Available: https://www.myheritage.com/incolor?lang=RU. [Дата обращения: 22 март 2020]. |
| [7] | P. Lim, «Bringing black and white photos to life using Colourise.sg — a deep learning colouriser trained with old Singaporean photos,» 3 Февраль 2019. [В Интернете]. Available: https://blog.data.gov.sg/bringing-black-and-white-photos-to-life-using-colourise-sg-435ae5cc5036. [Дата обращения: 22 Март 2020]. |
| [8] | Университет ИТМО, «Глубокое обучение,» [В Интернете]. Available: http://neerc.ifmo.ru/wiki/index.php?title=%D0%93%D0%BB%D1%83%D0%B1%D0%BE%D0%BA%D0%BE%D0%B5\_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5. [Дата обращения: 15 май 2020]. |
| [9] | J. Romero, «A Beginner's Guide to GANs,» [В Интернете]. Available: https://pathmind.com/wiki/generative-adversarial-network-gan. [Дата обращения: 2020]. |
| [10] | «Deepfake,» [В Интернете]. Available: https://en.wikipedia.org/wiki/Deepfake. [Дата обращения: 2020]. |
| [11] | M. Rovai, «Colorizing Old B&W Photos and Videos With the Help of AI,» 22 3 2019. [В Интернете]. Available: https://towardsdatascience.com/colorizing-old-b-w-photos-and-videos-with-the-help-of-ai-76ba086f15ec. [Дата обращения: 24 1 2020]. |
| [12] | A. A. a. L. F.-F. Justin Johnson, «Perceptual Losses for Real-Time Style Transfer,» [В Интернете]. Available: https://cs.stanford.edu/people/jcjohns/papers/eccv16/JohnsonECCV16.pdf. [Дата обращения: 15 Май 2020]. |
| [13] | P. F. a. T. B. Olaf Ronneberger, «U-Net: Convolutional Networks for Biomedical,» [В Интернете]. Available: https://arxiv.org/pdf/1505.04597.pdf. [Дата обращения: 15 Май 2020]. |
| [14] | J. Antic, «DeOldify,» 2018. [В Интернете]. Available: https://habr.com/ru/post/428818/. [Дата обращения: 23 01 2020]. |