



**Московский государственный университет имени М.В. Ломоносова**

Факультет вычислительной математики и кибернетики

Кафедра интеллектуальных информационных технологий

Лаборатория компьютерной графики и мультимедиа

Гончаренко Дмитрий Александрович

---

**Алгоритм изменения времени суток на изображении**

---

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**Научный руководитель:**  
К.С. Зипа

Москва, 2019

## **Аннотация**

Перенос изображений – это класс задач компьютерного зрения, цель которого состоит в извлечении стилевых признаков из изображений (или множеств изображений) и последующем применении предобученной модели к базовому изображению для генерации новых изображений с заданными стилями с сохранением семантической информации. Данная сфера значительное продвинулась благодаря современным вычислительным возможностям, в частности, переносе обучения на графические процессоры, GPU. За последние несколько лет появилось немало исследовательских работ на тему междоменных<sup>1</sup> переносов изображений, стилей и колоризация. В данной работе рассматриваются современные подходы к переносу изображений, применимые к задаче изменения времени суток на изображении. Проводится описание нейросетевых моделей, обоснование и выбор метода и сравнительный анализ серии экспериментов обучения. Обученную модель и полученные результаты переноса можно использовать для аугментации данных в смежных задачах машинного обучения.

## **Abstract**

The algorithm of changing the time of the day on images is a subclass of Machine Learning problems of image translation. Image-2-image translation is one of the of computer vision problems, where the goal is to extract style features from image (or set of images) with subsequent applying the pt-model to the base image to generate new images with specified styles while preserving semantic information. This area has advanced significantly due to the modern computing capabilities, in particular the training transfer on GPUs. Over the past few years, many research papers have appeared on the subjects of crossdomain images translation, styles transfering and colorization. This research reveals modern approaches of image translation applicable to problem of changing the time of day on the image. A description of the neural network models, the rationale and choice of method for training, and a comparative quality analysis of a series of training experiments are carried out. The model and translation results can be applied to augment data in related machine learning problems.

---

<sup>1</sup> **Домен** – область, множество, содержащее в себе объекты одного типа.

# Содержание

<b>1 Введение</b>	<b>3</b>
<b>2 Постановка задачи</b>	<b>5</b>
2.1 Цель работы . . . . .	5
2.2 Решаемые задачи . . . . .	5
2.3 Формальная постановка задачи . . . . .	5
<b>3 Обзор существующих методов</b>	<b>6</b>
3.1 Генеративно состязательные сети (GAN) . . . . .	6
3.2 Автокодировщики (AE) . . . . .	6
3.3 Вариационные автокодировщики (VAE) . . . . .	7
3.4 Методы использующие обучение с учителем . . . . .	8
3.4.1 Условные состязательные сети (cGAN) . . . . .	8
3.5 Методы использующие обучение без учителя . . . . .	9
3.5.1 Цикловая согласованность (CC) . . . . .	10
3.5.2 Состязательные сети с общим скрытым пространством . . . . .	13
3.6 Выводы . . . . .	14
<b>4 Архитектура нейронной сети</b>	<b>15</b>
4.1 Домены изображений . . . . .	15
4.2 Вариационные автокодировщики . . . . .	15
4.3 Генеративно состязательные сети . . . . .	16
4.4 Сверточная сеть выделения признаков . . . . .	17
4.5 Нормализация . . . . .	18
<b>5 Проведенное исследование</b>	<b>19</b>
5.1 Собранная база изображений . . . . .	19
5.2 Эксперименты . . . . .	21
5.2.1 Эксперименты на базе изображений NEXET . . . . .	22
5.2.2 Эксперименты на расширенной базе NEXET . . . . .	23
5.2.3 Эксперименты на расширенной базе без ошибочных изображений . . . . .	24
5.2.4 Эксперименты на объединенной базе изображений NEXET и BDD100K	25
5.3 Экспериментальная оценка . . . . .	26
<b>6 Результаты</b>	<b>28</b>
6.1 Применение результатов работы в задачах компьютерного зрения . . . . .	28
<b>7 Программная реализация</b>	<b>30</b>
7.1 Инструментарий . . . . .	30
7.2 Программный код . . . . .	30
<b>8 Заключение</b>	<b>31</b>
<b>9 Список литературы</b>	<b>32</b>

# 1 Введение

Многие математические описания моделей машинного обучения появились еще в середине 20-го века. А в конце 50х годов уже начались первые попытки их практической реализации. Спустя почти полвека задачи машинного обучения все также не теряют своей актуальности. Рост числа работ, открытие исследовательских центров внутри компаний и при университетах показывает, что высокий интерес к данной сфере не ограничивается только наукой. За последние 10 лет появилось множество удобных инструментов для построения самых разнообразных моделей обучения, что оказало немалое влияние на развитие *науки о данных*<sup>1</sup>.

Стоит отметить, что не последнюю роль здесь сыграли многократно выросшие вычислительные возможности. Это позволило обучать *нейронные сети*<sup>2</sup> на персональных компьютерах за несколько минут, для чего ранее могли требоваться недели, а то и месяцы на специализированных вычислительных устройствах.

*Перенос обучения* является одной из центральных исследовательских задач современного машинного обучения. Данная область направлена на получение некоторой информации об объекте, сохранение и последующее применение этих знаний к другому объекту, связанному с первым. *Перенос изображений*, в свою очередь, является подклассом переноса обучения применяемым на изображениях. В настоящее время переносу изображений уделяется значительное внимание в исследовательских работах [1], а разнообразие сферы применения поистине впечатляющее. Так с помощью переноса изображений можно добиться объединения стилей двух изображений [2], что может использоваться художниками в создании интересных картин и композиций, колоризации<sup>3</sup> черно-белых фотографий [3], благодаря чему можно в автоматизированном режиме перекрашивать отфильтрованные монохромные киноленты и фотоснимки, объединение локальных признаков объектов и животных [4], что позволяет создавать несуществующие породы животных и по наброску генерировать текстурированное изображение, увеличение разрешения изображений [6], с помощью чего можно достигать точности, превышающей традиционные алгоритмы билинейной интерполяции.

Результаты задач переноса, в частности изменения времени суток на изображении, могут использовать для аугментации (размножения) данных для дальнейшего использования в других задачах машинного обучения. Помимо этого, предобученную модель может использовать для художественной автокоррекции фотографий.

В данной исследовательской работе рассматриваются и систематизируются современные подходы переноса изображений, применимые к задаче изменения времени суток на изображениях. Проводится обучение подходящего алгоритма на различных наборах тренировочных данных. Моя задача состоит в получении предобученной модели, умеющей из изображений дня синтезировать реалистичные изображения ночи, и наоборот из ночи научиться генерировать максимально правдоподобное изображение дня. Также я провожу сравнительный анализ скорости переноса и качества полученных результатов.

В качестве инструмента был выбран фреймворк PyTorch<sup>4</sup>, так как он разработан в виде

<sup>1</sup> **Наука о данных** (англ. *data science*) – совокупность процессов и методов направленных на извлечение информации из исследуемых данных.

<sup>2</sup> **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. Является мощным современным инструментом машинного обучения. Главная особенность – способность обучаться на предоставленных данных, называемых тренировочными.

<sup>3</sup> **Колоризация** (англ. *colorization*) – преобразование монохромных изображений в цветные.

<sup>4</sup> **PyTorch** – библиотека машинного обучения, основанная на Torch. Разработана исследовательской группой в Facebook.

библиотеки для языка Python – основного языка современного машинного обучения [21], и зарекомендовал себя мощным и гибким инструментом для исследования и обучения нейронных сетей.

## 2 Постановка задачи

### 2.1 Цель работы

Цель работы – получить модель, которая будет уметь переносить изображения между доменами дня и ночи и будет способна решить проблему изменения времени суток на изображениях.

### 2.2 Решаемые задачи

Проблему изменения времени суток на изображении можно разбить на следующие этапы:

- Исследование существующих методов переноса изображений и анализ решений родственных задач. На основе проведенного исследования, выбор модели, подходящей для решения поставленной задачи наилучшим образом
- Поиск применимых к проблеме тренировочных данных – изображений с различными временем суток. Полуавтоматическое исправление ошибок в базе изображений
- Множественное обучение выбранной модели с различными конфигурациями на наборах данных с помощью библиотеки PyTorch, анализ качества и скорости работы модели, выбор наилучших параметров.

### 2.3 Формальная постановка задачи

Формально алгоритм изменения времени суток на изображении можно сформулировать следующим образом:

#### Вход

На вход алгоритму поступают два изображения из различных доменов  $x_1 \in X_1$  и  $x_2 \in X_2$ , где

- $X_1$  – домен с набором изображений первого типа
- $X_2$  – домен с набором изображений второго типа

Не ограничивая общности, пусть  $X_1$  содержит изображения дня, а  $X_2$  соответствует ночи.

#### Выход

Необходимо построить *кросдоменные*<sup>1</sup> отображения  $f_{12}: X_1 \rightarrow X_2$  и  $f_{21}: X_2 \rightarrow X_1$ .

Реализовать функцию, принимающую на вход изображения из разных доменов  $x_1, x_2$  и возвращающую перенесенные изображения  $x'_2, x'_1$  соответственно:

$$x'_2, x'_1 = \text{CrossDomainTranslator}(x_1, x_2), \text{ где } x'_1 \in X_1, x'_2 \in X_2 \quad (1)$$

---

<sup>1</sup> Кросдоменный перенос (англ. *cross-domain transfer*) – перенос объектов между доменами

### 3 Обзор существующих методов

На сегодняшний день одними из наиболее популярных алгоритмов, на которых базируются решения многих задач переноса изображений, являются генеративно состязательные нейронные сети и вариационные автокодировщики. В этой главе я исследую подходы решения родственных методов и обосновываю выбор алгоритма для выполнения поставленной задачи.

#### 3.1 Генеративно состязательные сети (GAN)

**Генеративно состязательная нейронная сеть (GAN)** – комбинация двух нейронных сетей  $G$  (генератор) и  $D$  (дискриминатор). Генератор подбирает *латентные* (скрытые) параметры для генерации нового объекта. Дискриминатор пытается отличить оригинал от объекта, созданного генератором.

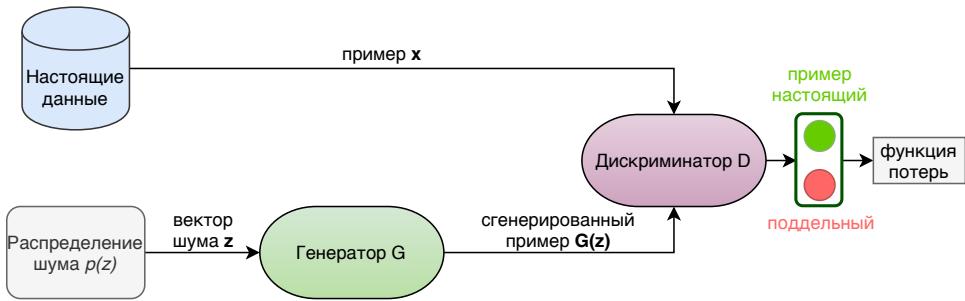


Рис. 1: Схема работы генеративно состязательной сети (GAN)

Формально генератор можно определить как отображение некоторого пространства скрытых параметров  $\mathcal{Z}$ , на котором задано априорное распределение  $p_z(z)$ , в пространство данных  $\mathcal{X}$ . Дискриминатор же будет производить отображение  $\mathcal{X}$  в отрезок  $[0, 1]$  – вероятность того, что пример настоящий. На рис. 1 представлена общая схема работы генеративной сети.

$$\begin{aligned} G: \mathcal{Z} &\rightarrow \mathcal{X} \\ D: \mathcal{X} &\rightarrow [0, 1] \end{aligned} \tag{2}$$

Основной целью генеративно состязательной сети является получение генератором распределения данных  $p_{gen}$ , не отличимого дискриминатором от исходного распределения  $p_{data}$ . То есть, по сути, заключается в решении задачи оптимизации [14]:

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(D, G), \text{ где} \tag{3}$$

$$\mathcal{L}_{\text{GAN}}(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

#### 3.2 Автокодировщики (AE)

**Автокодировщик (AE)** – комбинация двух нейронных сетей: кодировщика  $E$  и декодировщика  $D$ . Кодировщик получает и преобразует данные в сжатый код скрытого пространства. Декодировщик же старается из этого кода восстановить объект наиболее близко к исходному. Математически можно представить автокодировщик как отображения пространства

входных данных  $\mathcal{X}$  в латентное пространство  $\mathcal{Z}$  и обратно:

$$\begin{aligned} E: \mathcal{X} &\rightarrow \mathcal{Z} \\ D: \mathcal{Z} &\rightarrow \mathcal{X} \\ x \xrightarrow{E} z \xrightarrow{D} x', \text{ где} \end{aligned} \tag{4}$$

- $x \in \mathcal{X}$  – исходное изображение
- $z \in \mathcal{Z}$  – скрытый код
- $x' \in \mathcal{X}$  – восстановленное изображение

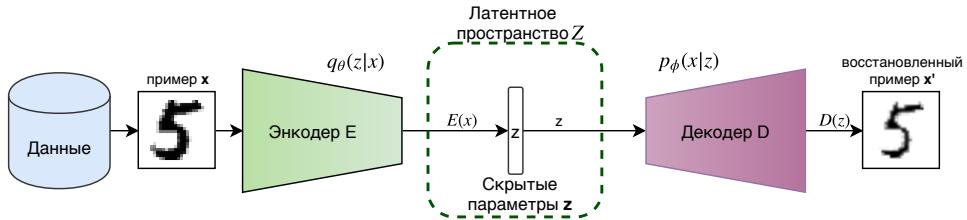


Рис. 2: Схема работы вариационных автокодировщиков (VAE)

Задача автокодировщика состоит в минимизации разницы между исходным изображением  $x$  и восстановленным  $x'$ . Для этого вводится функцию потерь  $\mathcal{L}_{\text{AE}}$ , характеризующая потерю при неправильном принятии решений:

$$\mathcal{L}_{\text{AE}}(x, x') = \|x - x'\|^2 \tag{5}$$

Основная проблема автокодировщиков заключается в том, что скрытое пространство может не быть непрерывным. Из-за чего не получается произвести интерполяцию, и как следствие, их область применения ограничивается. Этую проблему способны решить вариационные приближения.

### 3.3 Вариационные автокодировщики (VAE)

**Вариационный автокодировщик (VAE)** отличается от автокодировщика непрерывностью латентного пространства и предположениями накладываемыми на распределение данных. Пусть выбор тренировочных данных  $x$  находится в некоторой условной зависимости  $p(x|z)$  от переменных скрытого пространства  $z$ . Кодировщик на вход получает пример  $x$  и выдаст вектор размерности  $2d$  распределения скрытых переменных  $z$ , состоящий из двух векторов размерности  $d$ : вектора стандартных отклонений  $\sigma$  и вектора средних значений  $\mu$ . То есть кодировщик представляет собой нейронную сеть с параметрами  $\theta$  и обучается аппроксимировать апостериорное распределение  $q_{\theta}(z|x)$ . Декодировщик тоже является нейронной сетью с параметрами  $\phi$  и старается из вектора размерности  $2d$  скрытых переменных  $z$  получить объект из вариационного распределения над  $x$ :  $p_{\phi}(x|z)$ . На рисунке 2 представлена верхнеуровневая схема модели вариационных автокодировщиков.

Задачей вариационных автокодировщиков является задача оптимизации (6). Необходимо так подобрать параметры  $\theta$  в  $q_{\theta}(z|x)$ , чтобы максимально точно приблизить  $p(z|x)$ :

$$\arg \min_{\theta} \mathbb{KL}(q_{\theta}(z|x) \| p(z|x)), \text{ где} \tag{6}$$

$$\mathbb{KL}(q_{\theta}(z|x) \| p(z|x)) = \mathbb{E}_q [\log q_{\theta}(z|x)] - \mathbb{E}_q [\log p(z|x)] + \log p(x)$$

$\text{KL}$  – расстояние Кульбака-Лейблера<sup>1</sup> – мера удаленности двух распределений.

Так или иначе, методы реализующие перенос изображений опираются на состязательные сети GAN, VAE или их модификации [4, 5, 7, 9, 10, 12, 13]. Рассмотрим следующие алгоритмы с позиции решения предложенной задачи изменения времени суток на изображении. Будем рассматривать, уже введенные в формальном определении, домены изображений  $X_1$  и  $X_2$ , соответствующие разному времени суток (глава 2.3).

Пусть изображения  $x_1, x_2 : x_1 \in X_1, x_2 \in X_2$

### 3.4 Методы использующие обучение с учителем

**Обучение с учителем** (англ. *supervised learning*) – способ машинного обучения, в ходе которого входные данные соотносятся с выходными до начала обучения. Обучение происходит на заготовленных парах объектов  $(x_1, x_2)$ , называемыми ”стимул-реакция”, которые находятся в доменах в некотором совместном распределении  $P_{X_1, X_2}(x_1, x_2)$ <sup>2</sup>. Задача метода – построить функцию внутренней зависимости между примерами, которая затем будет отображать входные изображения желаемым образом.

Успешными примерами применения этого способа для решения задач переноса изображения можно считать эти работы [5, 10].

#### 3.4.1 Условные состязательные сети (cGAN)

Метод представленный на международной конференции компьютерного зрения (CVPR) в 2017 году исследовательской группой из университета Беркли под именем pix2pix [5] использует для переноса изображений обучение с учителем и условные генеративно состязательные нейронные сети.

Условные GAN являются модификацией над стандартными GAN. От обычных генеративных сетей (3), cGAN отличаются добавочным вектором информации  $y$  для генератора  $G(z|y)$  и дискриминатора  $D(X|y)$ . Вектор  $y$  может содержать любую уточняющую информацию, например он может содержать метки объектов на изображении, в случае с pix2pix,  $y$  – это добавочное входное изображение, видимое генератору и дискриминатору.



Рис. 3: Схема работы условно генеративно состязательной сети (cGAN)  
 $x$  – настоящее изображение ночи, соответствующее  $y$

<sup>1</sup> **Расстояние Кульбака–Лейблера**  $\text{KL}(P\|Q)$  – мера разницы между двумя вероятностными распределениями  $P$  и  $Q$ . Служит для оценки качества приближения  $P$  с помощью  $Q$ . Выражает количество потерянной информации. В общем случае выражается формулой:  $\text{KL}(P\|Q) = \int \log \frac{dP}{dQ} dP$

<sup>2</sup> **Совместное распределение** – это распределение совместных исходов образованных из нескольких случайных величин.

Выше на рис. 3 приведена общая схема работы cGAN в модели pix2pix. В качестве функции потерь была взята совокупность двух функций потерь: генеративной сети (3) в случае с cGAN и  $\mathcal{L}_1$ . Сеть решает следующую задачу оптимизации:

$$\arg \min_G \max_D \left( \mathcal{L}_{\text{cGAN}}(D, G) + \lambda \mathcal{L}_1(G) \right), \text{ где} \quad (7)$$

$$\mathcal{L}_{\text{cGAN}}(D, G) = \mathbb{E}_x [\log D(x)] + \mathbb{E}_{y,z} [\log(1 - D(G(z, y)))] , \mathcal{L}_1(G) = \sum_{i=1}^n (x - G(z, y))^2$$

$\lambda$  – параметр, контролирующий вклад  $\mathcal{L}_1$ . При  $\lambda = 0$  получаются более четкие изображения, но подверженные артефактам, при больших значениях параметра, будут получаться менее четкие, размытые изображения, но с визуально уменьшенным количеством артефактов [5]. Модель обучалась на датасете<sup>1</sup> [28] парных фотографий (с одинаковых ракурсов) сделанных с уличных вебкамер в разрешении 256x256px в разное время суток. После проведения серии экспериментов стало ясно, что на данных, немного отличных от обучающей выборки, точность модели падает, появляются артефакты и размытости, поэтому данный метод в своей задаче использовать не получится.

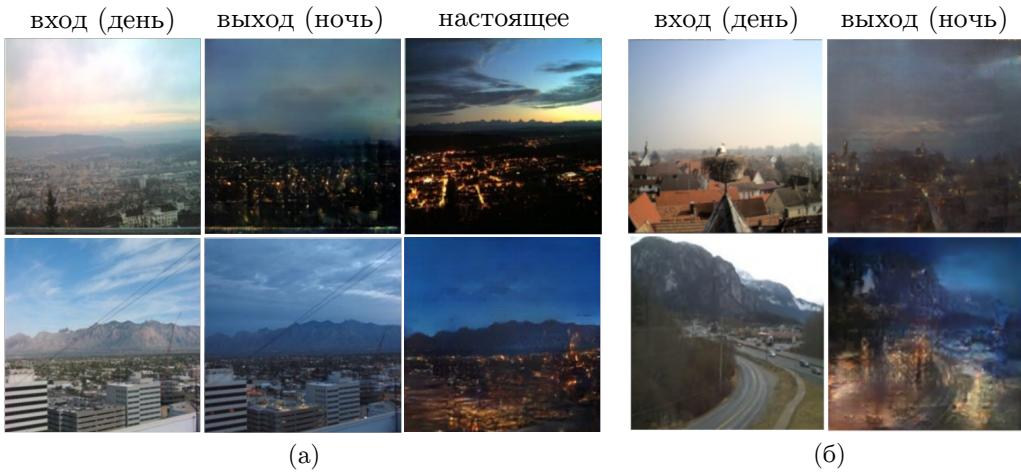


Рис. 4: Результаты работы pix2pix (день в ночь)

(а) успешная работа метода, (б) артефакты на нестандартных примерах

Результаты работы программы изображены на рис. 4. Как можно видеть 4 а), метод генерирует довольно неплохие результаты на примерах близких к обучающим, но сильно деградирует на нестандартных изображениях, отличных от тренировочной выборки 4 б).

Существенной проблемой методов, использующих обучение с учителем, является необходимость хранить объекты доменов по парам. Это сказывается на сложности поиска и сбора тренировочных данных для применения их на практике. С другой стороны, методы без учителя, лишены этого недостатка, и данные можно получать фактически "из воздуха".

### 3.5 Методы использующие обучение без учителя

**Обучение без учителя** (англ. *unsupervised learning*) – совокупность задач машинного обучения, решаемых на неразмеченных данных. В отличие от обучения с учителем, где тренировочные данные находятся в совместном распределении  $P_{X_1, X_2}(x_1, x_2)$ , в методах

---

<sup>1</sup> Датасет (англ. *dataset*) – то же, что и набор данных

без вмешательства учителя требуется найти априорное распределение, выбирая данные  $(x_1, x_2)$  из доменов с частным распределением  $P_{X_1}(x_1)$  и  $P_{X_2}(x_2)$ <sup>1</sup>.

Метод обучения без учителя значительно упрощает сбор тренировочных данных, так как для обучения не требуются парные объекты. К минусам можно отнести, что этот подход требует большего числа обучающих изображений для получения реалистичных результатов, чем методы с учителем. К наиболее интересным работам, использующим этот подход, можно отнести [4, 7, 9].

### 3.5.1 Цикловая согласованность (CC)

Введем новое понятие, которое будет использоваться в алгоритмах обучения без учителя. **Цикловая согласованность** (также **круговая**, англ. *cycle-consistency*) означает, что если изображение перенести между доменами и обратно, то должно получиться исходное изображение. Более формально: пусть заданы биективные отображения  $G : X \rightarrow Y$  и  $F : Y \rightarrow X$  где  $X, Y$  – различные домены, тогда если изображение  $x \in X$  отображается в  $G(x) = y \in Y$ , то изображение  $y$  отображается обратно в  $F(y) = x$ . Ниже на рис. 5 изображена схема цикловой согласованности:

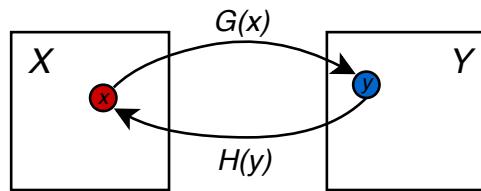


Рис. 5: Схема круговой согласованности для двух доменов  $X$  и  $Y$   
 $x$  – изображение домена  $X$ ,  $y$  – изображение домена  $Y$

Математически, можно сформулировать цикловую согласованность для случая двух доменов, формально введенного выше, в виде задачи минимизации следующей функции:

$$\mathcal{L}_{\text{CC}}(G, F) = \mathbb{E}_x [\|F(G(x)) - x\|] + \mathbb{E}_y [\|G(F(y)) - y\|] \quad (8)$$

Данный способ и его модификации используют многие методы в задачах обучения без учителя. Определенных успехов добились в этих работах [4, 7, 9]. Рассмотрим наиболее известные алгоритмы последних лет, в которых решаются задачи переноса изображения без учителя и которые можно применить к задаче дня и ночи.

**Состязательные сети с цикловой согласованностью** Исследовательская группа при университете Беркли, предложившая метод переноса с учителем pix2pix [5] (глава 3.4.1), признала, что для многих задач переноса изображений парное обучение будет неприменимым. Авторы представили новый метод, основанный на генеративно-состязательных сетях с цикловой согласованностью и на обучении без учителя [4], написанный на языке Lua с использованием Torch. Новый метод, получивший название CycleGAN, быстро обрел свою популярность, появилось множество его модификаций и реализаций в том числе и с использованием PyTorch.

В основу метода лег все тот же pix2pix с его условными состязательными сетями, см. подробнее в главе 3.4.1, с тем отличием, что теперь тренировочная выборка была без заранее

<sup>1</sup> Частное распределение – это распределение вероятности компонент некоторого множества, без зависимости между компонентами.

сформированных пар. Также в работе используется круговая согласованность генеративной сети, то есть модель обучается, не только переносить объекты между доменами, но и восстанавливать исходные изображения обратным переносом.

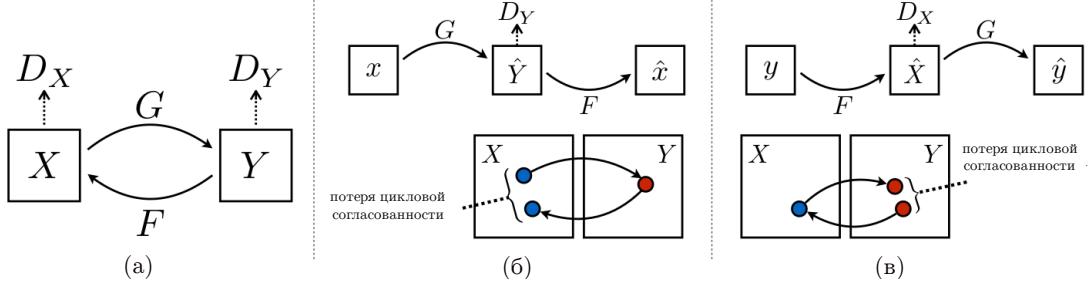


Рис. 6: Схема цикловой согласованности состязательных сетей CycleGAN

(а) модель состоит из двух отображений  $G : X \rightarrow Y$  и  $F : Y \rightarrow X$ , и двух состязательных дискриминаторов  $D_Y$  и  $D_X$ , для регулирования переноса используются (б) прямая цикловая согласованность:  $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$ , и (в) обратная:  $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

В модели используются две состязательные сети:

- дискриминатор  $D_X$ , отличающий изображения  $\{x\}, x \in X$  от перенесенных  $\{F(y)\}$
- дискриминатор  $D_Y$ , отличающий изображения  $\{y\}, y \in Y$  от перенесенных  $\{G(x)\}$

$D_X$  заставляет генератор  $G$  переносить изображения из  $X$  в изображения неотличимые от домена  $Y$ , аналогично  $D_Y$  поощряет  $F$  синтезировать изображения близкие к  $X$ . Авторы ввели функции потерь цикловой согласованности  $\mathcal{L}_{CC}$  (8) и две функции состязательных потерь  $\mathcal{L}_{GAN_X}$  и  $\mathcal{L}_{GAN_Y}$  (3). Получилась следующая задача оптимизации:

$$\arg \min_{G,F} \max_{D_X,D_Y} \mathcal{L}_{GAN_X}(D_X, F, Y, X) + \mathcal{L}_{GAN_Y}(D_Y, G, X, Y) + \lambda \mathcal{L}_{CC}(G, F) \quad (9)$$

Круговая согласованность генеративных сетей модели схематически изображена на рис. 6.

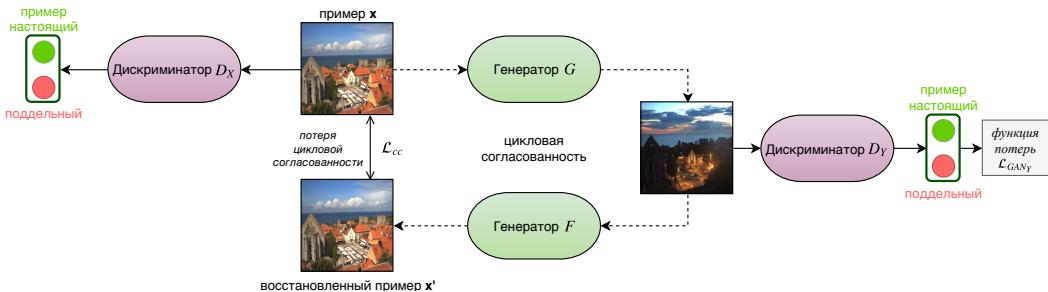


Рис. 7: Схема верхнеуровневой архитектуры CycleGAN (день в ночь)<sup>1</sup>

Выше, на рис. 7, представлена верхнеуровневая логика реализуемой системы без учителя для случая изменения времени суток со дня на ночь.

**Архитектура CycleGAN** представляет собой глубокую сверточную нейронную сеть, основанную на остаточной сети ResNet [16]. Сеть поддерживает два режима обучения в зависимости от разрешения изображений обучающей выборки. В случае 128x128px и ниже используется 6 остаточных блоков ResNet, а в случаях 256x256px и выше их количество увеличивается до 9. Архитектуру генератора можно разбить на три части: сверточную,

<sup>1</sup> для случая ночь в день, необходимо сменить местами генераторы и дискриминаторы, и по аналогии

трансформирующую и декодирующую. Сверточная часть состоит из трех сверточных слоев, уменьшающих исходное изображение в 4 раза, трансформирующая часть содержит 6 или 9 остаточных блоков, в зависимости от параметров сети, требующаяся для объединения и наложения признаков изображения, последняя часть состоит из двух обратных сверточных слоев, необходимы для возвращения матрицы изображения к исходной размерности, и одного выходного слоя, преобразующего матрицу в RGB изображение.

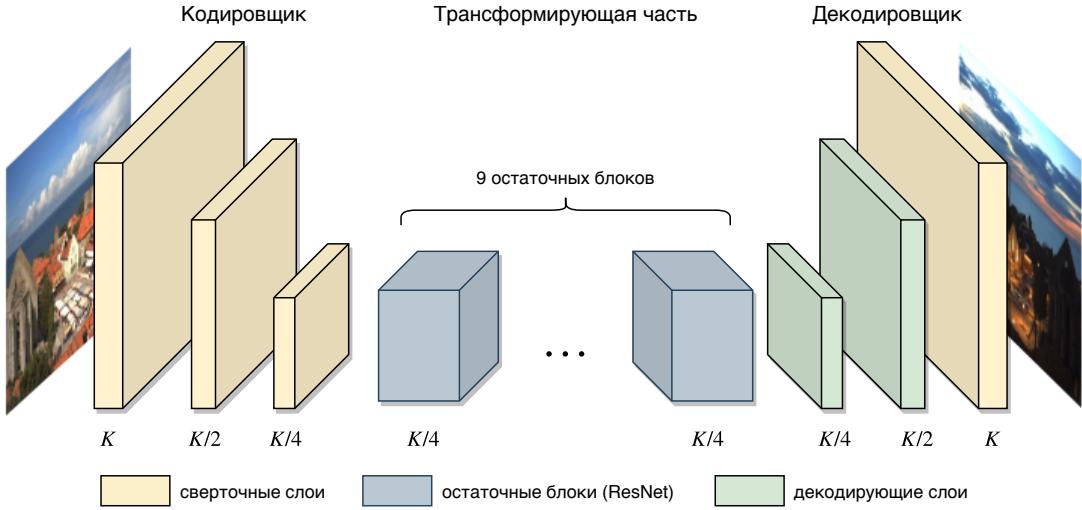


Рис. 8: Архитектура генератора CycleGAN  
Каждый слой завершается instance-нормализацией и слоем ReLU.  
 $K$  – исходный размер изображения

Благодаря сверточной структуре генератора, применить обученную сеть можно к изображению любого разрешения. На рис. 8 представлена схема сверточной сети генератора. В качестве архитектурного решения для дискриминатора был выбран тот же PatchGAN (см. подробнее на рис. 9), уже показавший неплохие результаты в pix2pix [5]. PatchGAN – это дискриминатор генеративно-состязательной сети, получающий на вход наложенные друг на друга части исходного изображения, "патчи". Результатом работы такого дискриминатора является матрица вероятностей, каждый элемент которой хранит вероятность того, что соответствующий "патч" настоящий.

Результаты работы CycleGAN превосходят по качеству все методы обучения без учителя, выходившие до этого и почти достигают по точности методы парного обучения. На сегодняшний день, эта работа показывает один из самых достойных результатов переноса изображений без учителя. При всех достоинствах алгоритма, задачи требующие существенных геометрических изменений на изображениях, например перенос кошки в собаку, зачастую терпят неудачу и требуют внесения ряда изменений [4].

У сети появилось и продолжает появляться множество конкурентов [7, 9, 10, 12]. Одним из первых успешных алгоритмов без учителя и прямым конкурентом CycleGAN, можно считать вот эту работу исследователей из NVIDIA [7]. Модель, получившая название UNIT<sup>1</sup>, основана на архитектуре схожей с [4], но выполняет ряд дополнительных предположений о распределении данных, а также включает в себя вариационные автокодировщики, глава 3.3, что позволяет разрешить проблемы искажений при геометрических преобразованиях у CycleGAN.

<sup>1</sup> UNIT (расп. англ. *UNsupervised Image-to-Image Translationns*)

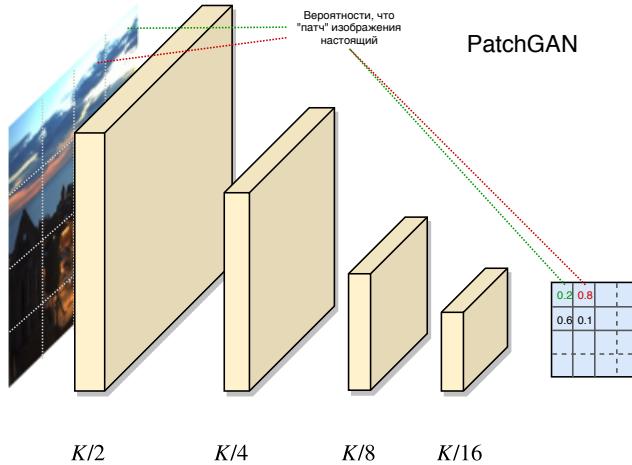


Рис. 9: Архитектура дискриминатора PatchGAN

(каждый слой завершается instance-нормализацией и слоем ReLU)

На вход подается разбитое изображение в масштабе 70x70px – на выходе имеем матрицу вероятностей, что каждая отдельная часть является настоящей

### Взаимосвязанные состязательные сети (coGAN)

Взаимосвязанные генеративно состязательные сети представляют собой надстройку над обычными состязательными сетями [13]. CoGAN состоят из объединения двух сетей GAN, каждая сеть относится к своему домену. Взаимосвязанные сети позволяют решать задачи поиска совместного распределения на непарных данных, что позволяет производить обучение на наборах данных без какой-либо заранее сформированной связи между объектами. Основной идеей coGAN является связь первых слоев генераторов  $G_1$  и  $G_2$ , отвечающими за декодирование верхнеуровневой семантики, и обмен весами между ними, аналогично связаны последние слои у дискриминаторов  $D_1$  и  $D_2$ , отвечающие за кодирование верхнеуровневой семантики. На данном подходе основываются генеративные сети в UNIT [7].

#### 3.5.2 Состязательные сети с общим скрытым пространством

Создатели UNIT в своей работе используют связку VAE-GAN, что позволяет добиться реалистичности при непарном переносе изображений. Состязательные сети в модели основаны на взаимосвязанных GAN (coGAN), с обменом параметрами весов между первыми слоями генераторов  $G_1$  и  $G_2$ , см. главу 3.5.1. По этой же логике связаны последние высокоуровневые слои кодировщиков  $E_1$  и  $E_2$ .

Авторы вводят понятие общего скрытого пространства  $\mathcal{Z}$ , как гипотетического пространства скрытых переменных  $\{z\}$ , являющегося общим для каждого из доменов изображений. Формализовать эту теорию можно математически. Пусть у двух доменов изображений  $X_1$  и  $X_2$ , между которыми планируется перенос, задано общее скрытое пространство  $\mathcal{Z}$ . Пусть для каждого  $x_1 \in X_1, x_2 \in X_2$  существует общий латентный код  $z$  в пространстве  $\mathcal{Z}$ . Тогда существуют такие функции, что выполняются следующие соотношения:

$$\begin{aligned} E_1(x_1) &= E_2(x_2) = z \\ G_1(z) &= x_1 \\ G_2(z) &= x_2 \end{aligned} \tag{10}$$

А весь процесс переноса изображений между доменами  $X_1$  и  $X_2$ , можно представить в виде композиции функции  $F_{1 \rightarrow 2}(x_1) = G_2(E_1(x_1)) = x_2$ , и обратно  $F_{2 \rightarrow 1}(x_2) = G_1(E_2(x_2)) = x_1$ .

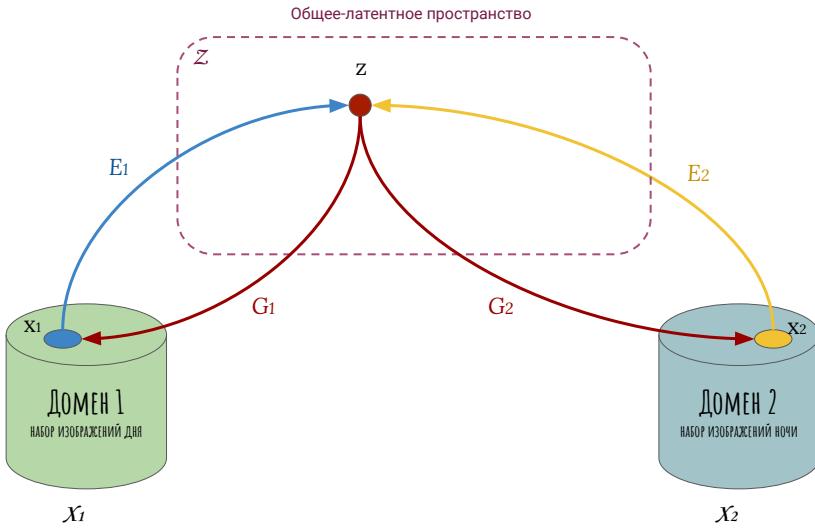


Рис. 10: Схема работы VAE-GAN состязательных сетей с общим скрытым пространством  
 $z$  – код в скрытом пространстве  $Z$ ,  $x_1$  – пример дня из домена  $X_1$ ,  $x_2$  – пример ночи из домена  $X_2$ ,  $E_1$  – кодировщик сети для домена  $X_1$ ,  $E_2$  – кодировщик сети для домена  $X_2$ ,  $G_1$  – генератор, синтезирующий изображения дня из скрытого представления,  $G_2$  – генератор, создающий изображения ночи из скрытого представления.

Схематически работа связанных VAE-GAN состязательных сетей с общим латентным пространством для двух доменов изображена на рис. 10.  
 Можно заметить, что предположение об общем скрытом пространстве одновременно подразумевает цикловую согласованность, глава 3.5.1.

### 3.6 Выводы

Подводя итог, после изучения различных методов по переносу изображений, стало ясно, что для задачи изменения времени суток на изображении лучше подойдут алгоритмы обучения без учителя.

После исследования литературы, я определил, что наиболее удачными являются методы основанные на GAN с круговой согласованностью [4, CycleGAN] и [7, UNIT]. Они используют схожие архитектуры и получают близкие результаты [11], но UNIT обладает более сложной структурой из 6 связанных сверточных нейросетей: вариационных автокодировщиков и взаимосвязанных состязательных сетей, что в перспективе дает более гибкую настройку и пространство для маневров и расширений, поэтому в моей работе основной моделью выступает именно UNIT, но также проводится обучение и сравнение с CycleGAN.

Исходя из изучения методов без учителя, для реализации поставленной задачи потребуются собрать значительную базу изображений различных времен суток. Авторы просмотренных мной работ не предоставляют доступы к внутренним датасетам лабораторий для обучения, поэтому под сбор данных я отвел отдельную подзадачу и подробнее о ее результатах расскажу в разделе 5.1.

## 4 Архитектура нейронной сети

В качестве архитектурного решения в работе используется сверточная нейронная сеть, состоящая из двух взаимосвязанных генеративно состязательных сетей (coGAN, глава 3.5.1) и двух связных вариационных автокодировщиков (VAE, глава 3.3). Данный подход был предложен в статье [7] и основан на предположении об общем скрытом пространстве переменных двух доменов, которое было описано в разделе 3.5.2.

### 4.1 Домены изображений

Модель подразумевает обучение без учителя на непарных данных, то есть на наборах классифицированных изображениях, не имеющих заранее установленных связей и меток. В архитектуре заложено использование двух доменов, поэтому для решения задачи необходимо заранее распределить тренировочные данные между ними. Подробнее сбор и классификация данных будут описаны в разделе 5.1. Сейчас остановимся на архитектурном решении загрузки и манипуляции изображениями в сети.

Для двух различных доменов  $X_1$  и  $X_2$  настроены загрузчики данных, выполняющие загрузку, преобразование и отправку изображений в модель. Для аугментации данных<sup>1</sup> в загрузчике используются композиция трех видов преобразований входных изображений:

1. Изменение размера (resize) – уменьшение или увеличение разрешения изображения
2. Обрезка (crop) – обрезание входного изображения в произвольном месте
3. Отражение (flip) – отражение входящих изображений по вертикали

Далее преобразованные данныечитывают кодировщики  $E_1$  и  $E_2$ .

### 4.2 Вариационные автокодировщики

В сети используются вариационные автокодировщики (VAE), их описание уже было проведено в разделе 3.3. В архитектуре модели используется два таких автокодировщика для каждого из доменов. VAE состоят из связанных кодировщиков  $E_1$  и  $E_2$ , и двух связных генераторов  $G_1$  и  $G_2$ . Первому домену соответствует автокодировщик  $\{E_1, G_1\}$ , второму –  $\{E_2, G_2\}$ . Каждый кодировщик получает на вход очередное изображения своего домена, сворачивает его в общий скрытый код  $z$ . Последние высокоуровневые слои  $E_1$  и  $E_2$  связаны и обмениваются параметрами весов, что дает прирост в точности [13]. Декодировщики, они же генераторы, переносят изображение между доменами  $X_1$  и  $X_2$  (11) и восстанавливают исходное изображение из скрытого представления (12):

$$\begin{aligned} G_1(z_2 \sim q_2(z_2|x_2)) &= \tilde{x}_2^{2 \rightarrow 1} \\ G_2(z_1 \sim q_1(z_1|x_1)) &= \tilde{x}_1^{1 \rightarrow 2} \end{aligned} \tag{11}$$

$$\begin{aligned} G_1(z_1 \sim q_1(z_1|x_1)) &= \tilde{x}_1^{1 \rightarrow 1} \\ G_2(z_2 \sim q_2(z_2|x_2)) &= \tilde{x}_2^{2 \rightarrow 2} \end{aligned} \tag{12}$$

Здесь  $z_1 \sim q_1(z_1|x_1)$  произвольное возмущение от скрытого представления  $z$  с распределением Гаусса и единичной дисперсией  $q_1(z_1|x_1) \equiv \mathcal{N}(z_1|E_1(x_1), I)$ , аналогично  $z_2$  получается из  $z$  с распределением  $q_2(z_2|x_2) \equiv \mathcal{N}(z_2|E_2(x_2), I)$ .

<sup>1</sup> **Аугментация данных** (англ. *augmentation*) – увеличение общего числа данных, посредством преобразований над исходными данными

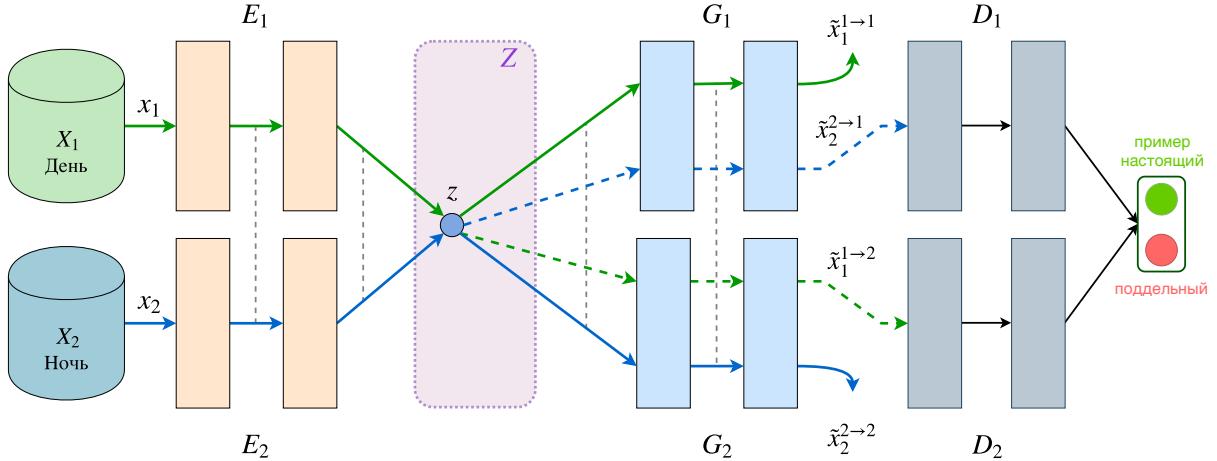


Рис. 11: Схема используемой в работе сверточной нейронной сети

Кодировщики  $E_1, E_2$  преобразуют изображения в скрытый представление  $z$ , генераторы  $G_1, G_2$  восстанавливают изображения  $\tilde{x}_1^{1 \rightarrow 1}, \tilde{x}_2^{2 \rightarrow 2}$  (сплошные стрелки) и переносят изображения в противоположные домены  $\tilde{x}_1^{1 \rightarrow 2}, \tilde{x}_2^{2 \rightarrow 1}$  из скрытого представления (пунктирные стрелки), дискриминаторы  $D_1, D_2$  отвечают за реалистичность и отличают настоящие изображения ( $x_1, x_2$ ) от сгенерированных ( $\tilde{x}_2^{2 \rightarrow 1}, \tilde{x}_1^{1 \rightarrow 2}$ ). Поскольку восстановление изображений контролируется сетью, GAN достаточно применять к перенесенным изображениям. Пунктирными серыми линиями отражены связи между последними слоями кодировщиков и первыми слоями генераторов, отвечающими за извлечение верхнеуровневой информации.

Полная схема работы нейронной сети, используемой в работе, представлена на рис. 11. Целью обучения VAE является минимизация следующих функций потерь (13):

$$\begin{aligned} \mathcal{L}_{\text{VAE}_1}(E_1, G_1) &= \lambda_1 \mathbb{KL}(q_1(z_1|x_1) \| p_\eta(z)) - \lambda_2 \mathbb{E}_{z_1 \sim q_1(z_1|x_1)} [\log p_{G_1}(x_1|z_1)] \\ \mathcal{L}_{\text{VAE}_2}(E_2, G_2) &= \lambda_1 \mathbb{KL}(q_2(z_2|x_2) \| p_\eta(z)) - \lambda_2 \mathbb{E}_{z_2 \sim q_2(z_2|x_2)} [\log p_{G_2}(x_2|z_2)], \text{ где} \end{aligned} \quad (13)$$

параметры  $\lambda_1$  и  $\lambda_2$  контролируют вклад соответствующих членов.

### 4.3 Генеративно состязательные сети

Помимо вариационных автокодировщиков в модели используются связанная пара генеративно-состязательных сетей GAN. Первый GAN состоит из генератора  $G_1$  и дискриминатора  $D_1$ , вторая состязательная сеть – из  $G_2$  и  $D_2$ . Генераторы уже были описаны в предыдущем подпункте. Дискриминаторы на вход получают настоящее изображение из соответствующего домена и перенесенное генератором из другого, и возвращают вероятности того, что изображения настоящие.

Функции потерь состязательных сетей можно записать в виде:

$$\begin{aligned} \mathcal{L}_{\text{GAN}_1}(E_2, G_1, D_1) &= \lambda_0 \mathbb{E}_{x_1 \sim P_{\mathcal{X}_1}} [\log D_1(x_1)] + \lambda_0 \mathbb{E}_{z_2 \sim q_2(z_2|x_2)} [\log(1 - D_1(G_1(z_2)))] \\ \mathcal{L}_{\text{GAN}_2}(E_1, G_2, D_2) &= \lambda_0 \mathbb{E}_{x_2 \sim P_{\mathcal{X}_2}} [\log D_2(x_2)] + \lambda_0 \mathbb{E}_{z_1 \sim q_1(z_1|x_1)} [\log(1 - D_2(G_2(z_1)))] \text{, где} \end{aligned} \quad (14)$$

гиперпараметр  $\lambda_0$  отвечает за вклад соответствующих членов.

Таким образом, архитектура сети состоит из двух кодировщиков, двух генераторов и двух состязательных дискриминаторов. Модель решает сразу несколько подзадач и обучает одновременно несколько подсетей. Ключевой целью является оптимизация минимаксной задачи:

$$\min_{E_1, E_2, G_1, G_2} \max_{D_1, D_2} \mathcal{L}_{\text{VAE}_1}(E_1, G_1) + \mathcal{L}_{\text{GAN}_1}(E_2, G_1, D_1) + \mathcal{L}_{\text{CC}_1}(E_1, G_1, E_2, G_2) + \mathcal{L}_{\text{VAE}_2}(E_2, G_2) + \mathcal{L}_{\text{GAN}_2}(E_1, G_2, D_2) + \mathcal{L}_{\text{CC}_2}(E_2, G_2, E_1, G_1), \text{ где} \quad (15)$$

функции потерь  $\mathcal{L}_{\text{VAE}} = (13)$  и  $\mathcal{L}_{\text{GAN}} = (14)$ , а потеря круговой согласованности  $\mathcal{L}_{\text{CC}}$  получается из функции:

$$\begin{aligned} \mathcal{L}_{\text{CC}_1}(E_1, G_1, E_2, G_2) &= \lambda_3 \mathbb{KL}(q_1(z_1|x_1) \| p_\eta(z)) + \lambda_3 \mathbb{KL}(q_2(z_2|x_1^{1 \rightarrow 2}) \| p_\eta(z)) - \\ &\quad \lambda_4 \mathbb{E}_{z_2 \sim q_2(z_2|x_1^{1 \rightarrow 2})} [\log p_{G_1}(x_1|z_2)] \\ \mathcal{L}_{\text{CC}_2}(E_2, G_2, E_1, G_1) &= \lambda_3 \mathbb{KL}(q_2(z_2|x_2) \| p_\eta(z)) + \lambda_3 \mathbb{KL}(q_1(z_1|x_2^{2 \rightarrow 1}) \| p_\eta(z)) - \\ &\quad \lambda_4 \mathbb{E}_{z_1 \sim q_1(z_1|x_2^{2 \rightarrow 1})} [\log p_{G_2}(x_2|z_1)]. \end{aligned} \quad (16)$$

гиперпараметры  $\lambda_3$  и  $\lambda_4$  управляют вкладом соответствующих компонент.

Сети	$E_1, G_1$	$E_1, G_2$	$G_1, D_1$	$E_1, G_1, D_1$	$G_1, G_2, D_1, D_2$
Роли	VAE для $X_1$	Перенос $X_1 \rightarrow X_2$	GAN для $X_1$	VAE-GAN	CoGAN [13]

Таблица 1: Роли подсетей в используемой модели

В таблице 1 представлены роли подсетей используемой модели.

#### 4.4 Сверточная сеть выделения признаков

Для выделения признаков на изображениях, в моей работе используется глубокая сверточная сеть VGG16 [17], рис. 12.

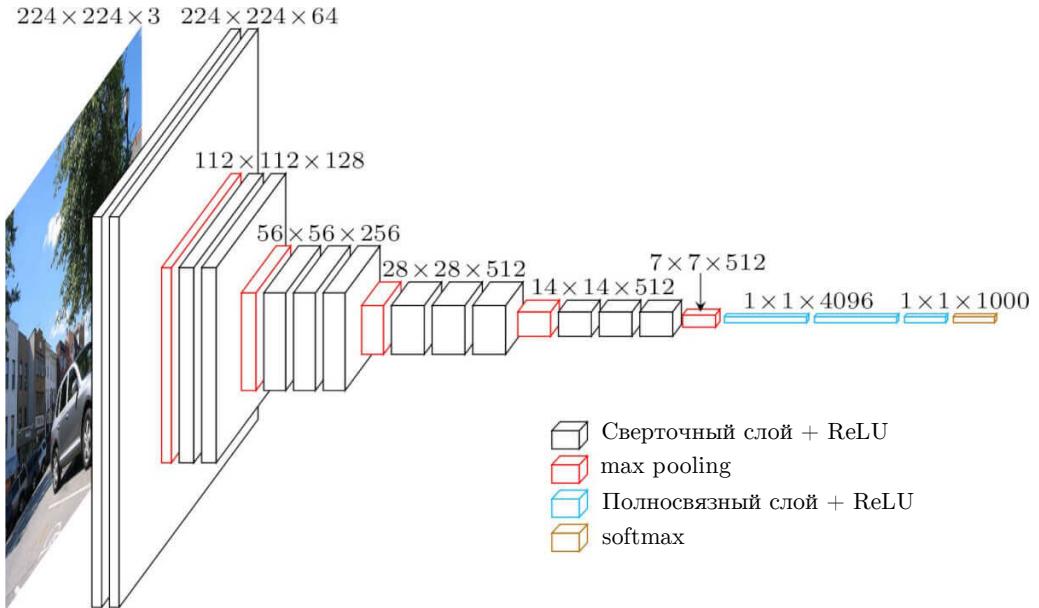


Рис. 12: Модель глубокой сверточной сети VGG16

VGG16 – это сеть нового поколения, построенная как усовершенствованная версия AlexNet [15]. VGG сопоставимая по точности с ResNet, но в обучении без учителя обладает большей стабильностью [9]. В процессе экспериментов проводится оценка качества работы модели как с использованием сверточной сети VGG16, так и без нее.

## 4.5 Нормализация

В процессе обучения были использованы различные стратегии повышения качества выходных изображений. Одним из подходов было применения слоев нормализации в генеративно-состязательных сетях. Всего было реализованы три типа нормализации: батч-нормализация [18], слой-нормализация [19] и instance-нормализация [20].

Общий вид вычисления нормализации можно записать следующим образом:

$$\hat{x} = \frac{x - \text{mean}(\bar{x})}{\text{std}(\bar{x})}, \text{ где} \quad (17)$$

mean – среднее значение, std – дисперсия.

Использование нормализаций позволяет существенно ускорить обучение нейронной сети. Одним из самых популярных видов нормировки является батч-нормализация, представленная в 2015 году в этой работе [18]. Авторы предлагают считать среднее значение от суммы всех тензоров в батче, после чего подсчитывать дисперсию и для каждого примера вычислять (17). Чем больше размер батча, тем эффективнее будет работа нормализации.

Другой подход применяют в этой работе [19], где предлагают вычислять слой-нормализацию вдоль входа, что дает независимое значение для каждого примера и позволяет использовать ее в модели даже при единичном размере батча.

Авторы instance-нормализации [20], продолжают идею слой-нормализаций, и вычисление происходит по каждому каналу каждого входного примера. Идеи каждой из нормализаций изображены на мнемонической схеме 13.

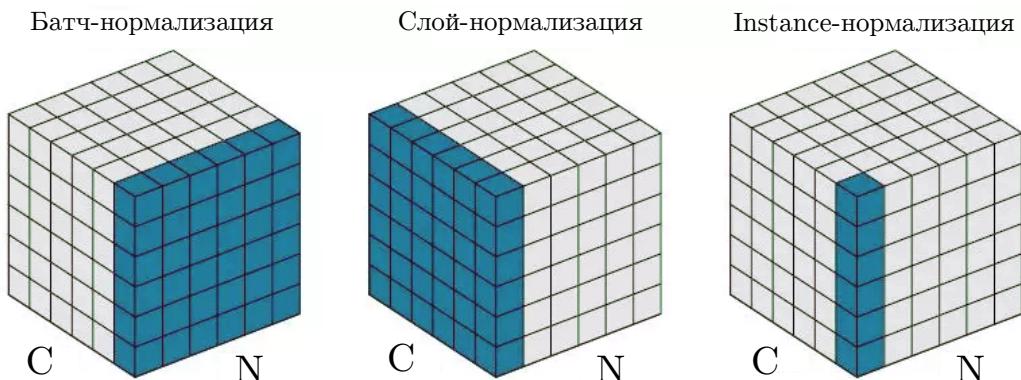


Рис. 13: Виды нормализации используемые в обучении

C – число каналов, N – размер батча

## 5 Проведенное исследование

### 5.1 Собранная база изображений

В науке о данных одну из ключевых ролей играет непосредственный выбор данных, на которых будет обучаться сеть, и которые будут использоваться для валидации качества модели. Перенос изображений не является исключением. Обучение без учителя глубоких сверточных сетей требуют большого объема тренировочных данных. Количество изображений, для получения достойных результатов, может варьироваться от нескольких сотен до десятков тысяч и больше. Во многих разобранных мной статьях, авторы, помимо описания алгоритма и демонстрации результатов, дают возможность провести самостоятельное обучение на предложенных датасетах или воспользоваться предобученными моделями. К сожалению, авторами не было предоставлено публичных наборов данных или предобученных моделей на тему изменения времени суток. Поэтому в своем исследовании, я выделил сбор требуемых данных и оценку их презентативности в отдельную подзадачу.

#### Возникшие трудности

После исследования области, оказалось, что основная проблема, заключается не в поиске данных, а в их разметке. Многие датасеты поставляются или с бесполезной для решения моей задачи разметкой, или в принципе без нее. Из всех найденных мной датасетов, которые теоретически возможно применить к поставленной задаче, самыми крупными и в наилучшем качестве, были наборы фотографий с автомобильных видеорегистраторов. Наиболее подходящие датасеты, которые я обнаружил:

1. NEXET [25] – набор из 50тыс. разнообразных дорожных размеченных изображений дня, ночи и сумерек. Данные расположены в одной директории, метки хранятся в .csv файле.
2. BDD100K [26] – набор из 100тыс. разнообразных дорожных изображений, изображения разбиты на тестовую и тренировочную выборки в соотношении 2:7, размечены только тренировочные данные, метки хранятся в .json файле.
3. Alderley DayNight [27] – набор из 15тыс. пар дорожных изображений, предоставлены в виде двух доменов: дня и ночи.
4. Night2Day Paired [28] – набор из 17тыс. пар изображений с веб-камер в низком разрешении, предоставлены соединенными.

Из датасетов выше: [27, 28] содержат недостаточное число изображений в низком разрешении для непарного переноса обучения. Alderley представляет собой кадры снятые только в одном месте – в пригороде Олдерли, Квинсленд. Night2Day содержат всего 101 уникальную пару и применяются в этой работе обучения с учителем [5], рис. 4. Такие данные больше подходят для парного обучения с учителем, в своей же работе я использую методы без учителя, поэтому мой выбор пал на два наиболее качественных датасета: NEXET и BDD100K. Датасеты содержат большое число изображений в высоком разрешении 720p с различной погодой и отснятых в разных городах, и отлично подходят для обучения выбранной нейронной сети.

#### Классификация изображений

Набор данных NEXET поставляется с файлом меток train.csv с тремя типами меток (день, ночь, сумерки), но как оказалось многие из них не соответствуют действительности, см. рис 14.

Изображения размечались в рамках соревнования [25], поэтому в наборе присутствуют спорные данные, неприменимые в обучении, так например целая метка "Сумерки" не представляет ценности, так как на ней могут быть как и яркие изображения, так и темные. Также никак не выделяются фотографии снятые в подземных парковках, туннелях и гаражах. Датасет BDD100K имеет более гибкие метки: ночь, день, рассвет/сумерки и неопределенность, в местах где нет однозначного ответа.

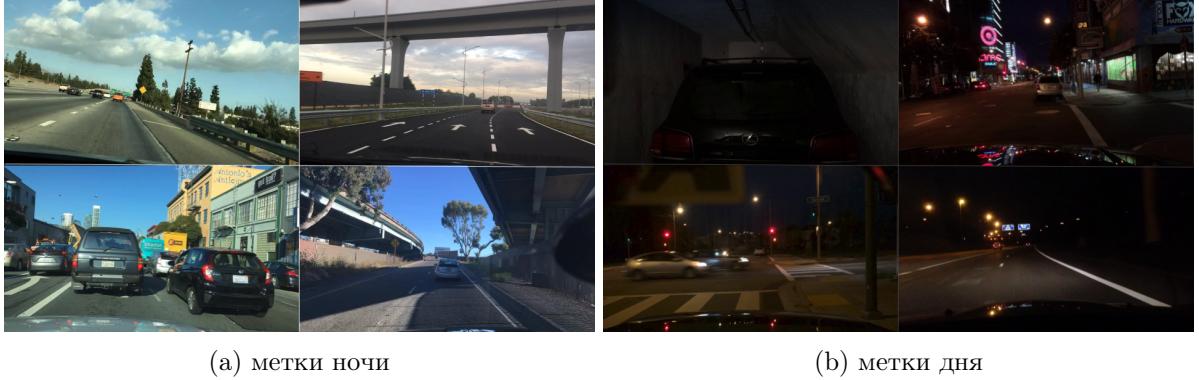


Рис. 14: Пример ошибочного сопоставления меток NEXET

(а) – примеры изображения с выставленными метками ночи, (б) – примеры изображения с выставленными метками дня, помимо явно ошибочных изображений, также присутствуют спорные (верхнее левое, (б)), например, изображения на подземных парковках и туннелях

Для исправляющей разметки датасетов был написан Python скрипт `detect_day_night.py`, определяющий время суток по яркости изображения. Математически формулу вычисляемую скриптом можно записать следующим образом:

$$\gamma = \frac{\sum_{i=1}^n \sum_{j=1}^m \sum_{s=1}^k c_{ijs}}{m * n * k}, \quad c_{ijs} \in I_{n \times m \times k} \text{ – тензор изображения} \quad (18)$$

Результат применения его к одному из датасетов ниже:

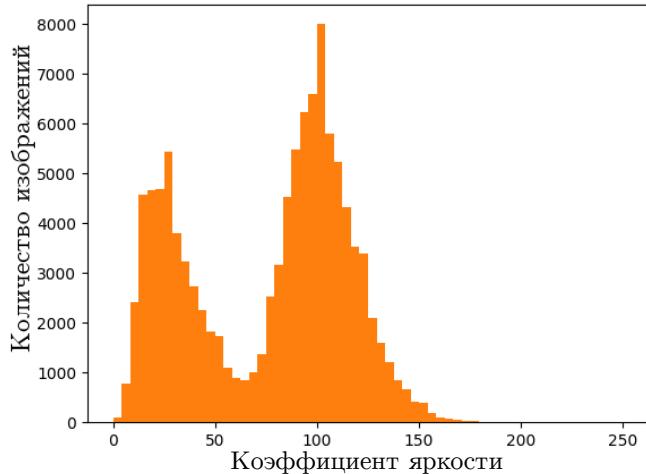


Рис. 15: Гистограмма распределения яркости изображений в датасете NEXET  
возвышение слева – коэффициент яркости меньше, темнее, ночь; возвышение справа – день

Экспериментальным путем было установлено, что данные находящиеся по коэффициенту яркости в диапазоне от  $60 < \gamma \leq 68$  с большей вероятностью относятся к спорным данным. Скрипт проводит сравнение с официальной разметкой, где выявляет несовпадения, после чего явно ошибочные данные отделяются от тренировочных.

## Распределение собранных данных на домены

В своей работе я использовал несколько различных стратегий разбиения данных. Всего в сети используется 4 домена: два тренировочных и два тестовых. Обучение проходило на изображениях из датасета NEXET [25], с распределением тренировочных и тестовых примеров в соотношении 9:1 и 1:1. Проводилась искусственная аугментация данных: отражения, обрезания и изменения размера, о чем уже упоминалось в разделе 4.1. Помимо этого, обучение проходило и на размеченных данных, без отделения от ошибочных и на совокупности с датасетом BDD100K [26], затем выполнялось сравнение полученных результатов.

## 5.2 Эксперименты

Обучение глубокой сверточной нейронной сети на базе изображений высокого разрешения требует внушительных вычислительных ресурсов. Так как тренировка проводилась на графических картах, основная нагрузка ложилась на ядра CUDA и видеопамять GPU.

Одно полное обучение может занимать от двух до 12 дней, в зависимости от параметров сети. Свои эксперименты я в основном производил на видеокартах с 12GB видеопамяти. Для датасета с изображениями в разрешении 256x256 пикселей требовалось 4700MB видеопамяти, а для изображений 512x288 пикселей – более 8.5GB. Ниже приведены график зависимости скорости обучения от используемой видеокарты, рис. 16, и таблица связи используемой памяти от разрешения изображений, табл. 2.

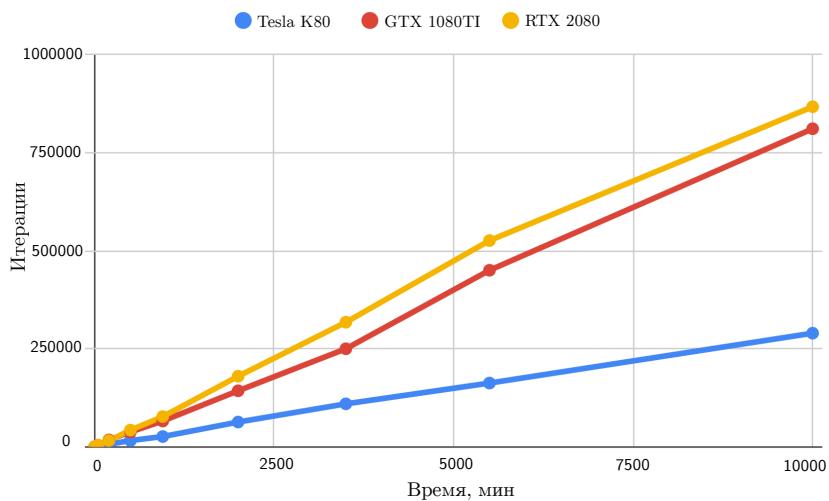


Рис. 16: Зависимость скорости обучения от GPU

В процессе обучения использовалось до 7 графических карт (3 различных). Самой быстрой себя показала графическая карта RTX 2080, близкие результаты у GTX 1080TI, поэтому основная часть обучения проводилась на них. Tesla K80 предоставляется на сервере Google Colab [29], и при большем количестве графических ядер CUDA, показывает худший результат. Скорее всего, это происходит из-за того, что мощность серверной карты урезается

Разрешение изображений	Видеопамять, VRAM	С использованием VGG16
64x64 px	1630 MB	1720 MB
128x128 px	2014 MB	2156 MB
256x256 px	4384 MB	4752 MB
512x288 px	7678 MB	8550 MB

Таблица 2: Зависимость используемой памяти от разрешения и VGG16

искусственно, так как она находится в распределенном доступе с другими пользователями. Из графика 16 также следует, что скорость обучения сети не меняется на всем протяжении обучения.

### Проведенные эксперименты

В общей сложности было проведено более 15 полноценных обучений с различными параметрами нейронной сети и на разнообразных наборах данных. Эксперименты можно разделить на 4 основные группы:

- Эксперименты на базе изображений NEXET [25]
- Эксперименты на базе NEXET с аугментацией данных<sup>1</sup> (расширенной)
- Эксперименты на расширенной базе NEXET с отделением от ошибочных изображений
- Эксперименты на объединенной базе NEXET + BDD100K [26]

Одним из наиболее значимых параметров, влияющих на результаты сети, является параметр слоя нормализации. В модели всего было реализовано 4 различных типа нормализации. В таблице 3 приведены различные конфигурации сети используемые при обучении.

Параметры обучения	Используемые нормализации				Модели	
	Нет	Batch [18]	Layer [19]	Instance [20]	Нет	VGG16
NEXET	✓	—	—	—	✓	—
NEXET+аугментация	✓	—	—	—	✓	✓
NEXET+скрипт <sup>2</sup>	✓	✓	✓	✓	✓	✓
NEXET+BDD100K	✓	—	✓	✓	—	✓

Таблица 3: Параметры нейронной сети в ходе экспериментов

#### 5.2.1 Эксперименты на базе изображений NEXET

Датасет NEXET [25] содержит 50 тыс. изображений с метками дня, ночи и сумерек. Изначально была произведена попытка обучить сеть с разделением на тестовые и тренировочные домены в соотношении 1:1. Сумерки использовались вместе с доменом ночи, табл. 4, эксп. 1.

Полученные результаты были далеки от идеала: преобразование ночи в день не получилось в принципе, на дне в ночь было множество артефактов. Тогда была предпринята попытка

<sup>1</sup> Аугментация – искусственное расширение датасета: отражения, обрезания, изменения размера, см. подробнее раздел 4.1

<sup>2</sup> `detect_day_night.py` – скрипт для определения времени суток и отделения спорных данных от тренировочных, см. разделы 5.1 и 5.1

	Эксперимент 1		Эксперимент 2	
Домены \ Время	День	Ночь + Сумерки	День	Ночь
Test	12451	12548	4981	4654
Train	12452	12549	19922	18578

Таблица 4: Разбиение датасетов для обучения на NEXET

увеличить тренировочную выборку за счет уменьшения тестовой, было выбрано соотношение 8 к 2. Также из обучения были убраны изображения сумерек (1874 шт.), поскольку их, по факту, невозможно было отнести ни к одному домену, так как под этой меткой могли скрываться как и дневные, так и ночные изображения, табл. 4, эксп. 2.

Перенос ночи в день все также не удался, день в ночь стал немного лучше, но артефакты все еще присутствовали.

### 5.2.2 Эксперименты на расширенной базе NEXET

Так как существует положительная зависимость между качеством обучения модели и количеством тренировочных данных, следующей предпринятой попыткой было искусственное расширение датасета, или аугментация. О способах аугментации, уже было сказано в разделе 4.1. Тренировочные и тестовые данные брались в отношении 9:1, чтобы максимизировать выборку для обучения, оставив достаточное количество изображений для валидации и оценки.

Качество результатов заметно выросло, артефакты почти исчезли, но присутствовала проблема в переносе цвета и выделении признаков у разных объектов. Так, например, деревья подсвечивались как фонари или дома, рис. 17.

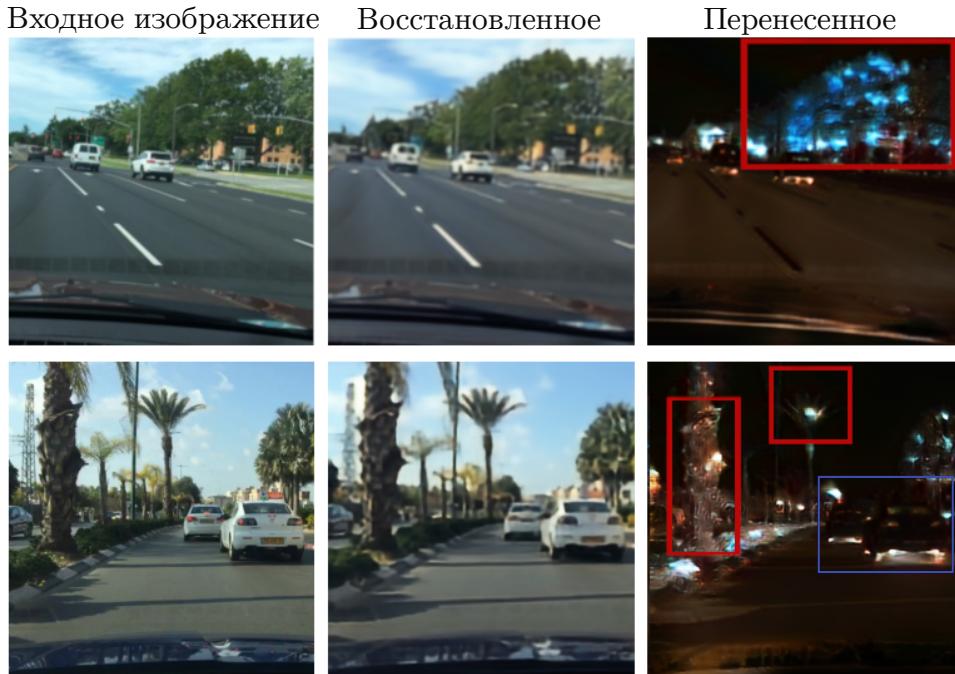


Рис. 17: Пример ошибочного выделения признаков  
**красный** – ошибка в выделении признаков, подсветка деревьев  
**синий** – ошибка в переносе цвета, затемнение машин

На этом этапе перенос ночи в день все еще был неуспешным. На перенесенных изображениях присутствовало большое число артефактов, а цвета были сильно искажены. Альтернативные стратегии разбиения данных на домены: другое соотношение тренировочных и тестовых примеров, изменение разрешения изображений, перемешивание изображений – также не увенчались успехом.

### 5.2.3 Эксперименты на расширенной базе без ошибочных изображений

В процессе экспериментов у некоторых изображений были обнаружены несовпадения меток, рис. 14. Для решения проблемы с разметкой данных была написана программа помогающая классифицировать изображения, подробнее см. раздел 5.1.

После программной обработки датасета, оказалось, что более 5000 изображений имели несовпадающие метки. Отделив ошибочные данные и перегруппировав домены, обучение проводилось еще раз.

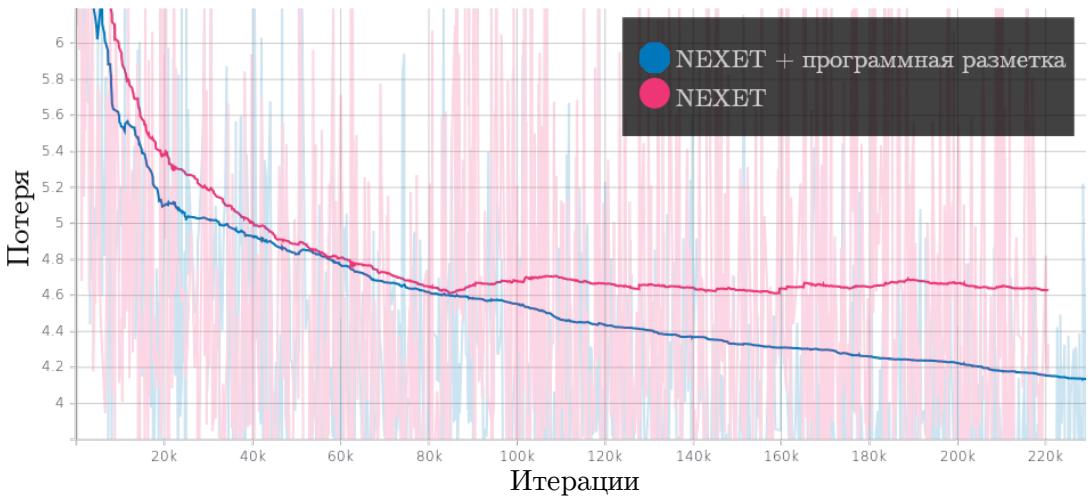


Рис. 18: Графики функция потерь NEXET после программной обработки

Как видно из графиков, на рис. 18, начиная с 85 тыс. итераций потеря модели на официальной разметке NEXET начинают колебаться около значения 4.7, когда как у датасета с программным удалением ошибочных данных обучение продолжается, а функция потерь убывает. На рисунке 19 приведены примеры работы алгоритма.

Дополнительно проводились эксперименты с добавлением слоев нормализации в GAN, см. раздел 4.5, что теоретически может разрешить проблему с искаженными цветами и засветами. Всего было использовано три различных подхода, табл. 3.

Батч-нормализация зависит от числа примеров одновременно участвующих в обучении [18]. Это накладывает ограничение на размера батча снизу. После сравнения результатов работы модели с нормализацией и без нее – визуальных улучшений обнаружено не было, поэтому дальнейшее обучение проводилось с размером минибатча равному одному, без батч-нормализации.

Также использовались альтернативные стратегии: слой-нормализация (layer) и instance-нормализация, у которых нет зависимости от размера батча [19, 20], что дает им преимущество над нормализацией выше в случае единичного размера минибатча.

Визуально результаты получались довольно близкие. Цвета передавались заметно точнее после применения к модели сверточной сети VGG16, подробнее в разделе 4.4. В совокупности со слой-нормализацией удалось затемнить яркие контуры и минимизировать

засветы. На рисунке 20 представлен пример работы сети, с различными типами конфигураций. Пример выбран случайно из тестовой выборки.



Рис. 19: Визуальное сравнение результатов модели с программной обработкой слева-направо: вход, без обработки NEXET, с программной обработкой NEXET

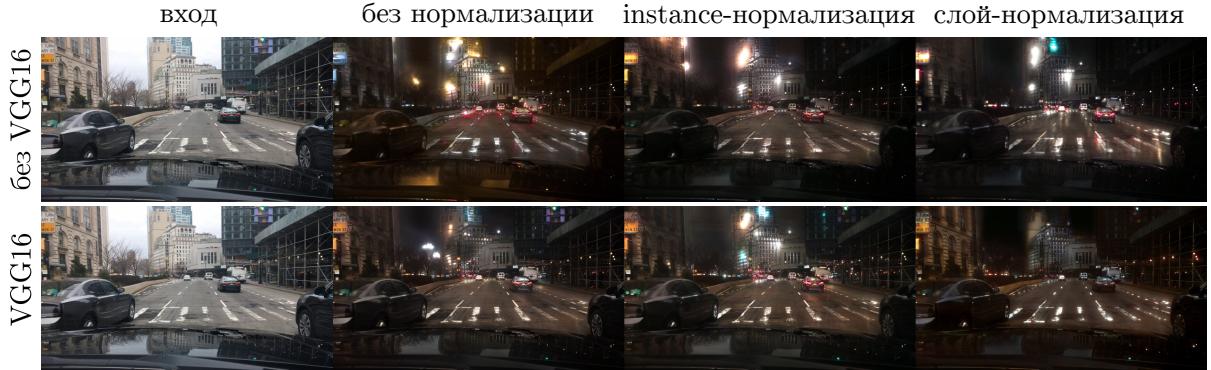


Рис. 20: Сравнение результатов модели с нормализацией и VGG16

#### 5.2.4 Эксперименты на объединенной базе изображений NEXET и BDD100K

Последняя группа экспериментов ставилась на наиболее крупной выборке, состоящей из двух датасетов: NEXET [25] и BDD100K [26]. Помимо всего прочего, к данным применялась аугментация и программное отделение ошибочных примеров, о которых уже было сказано в предыдущих разделах. Всего в обучении участвовало 140тыс. изображений. В общей сложности было поставлено 3 различных типа экспериментов с параметрами нормализации и VGG16 моделью, табл. 3.

На графике 21 изображены функции потерь для генератора и дискриминатора на протяжении всего обучения. В начале обучения потери генераторов компенсировались потерями

дискриминаторов, что затрудняло однозначно выделить победителя. Но в перспективе в районе 800 тыс. итераций, дискриминаторы показывали одинаковый результат, а генератор у BDD+NEXET показывал лучший.

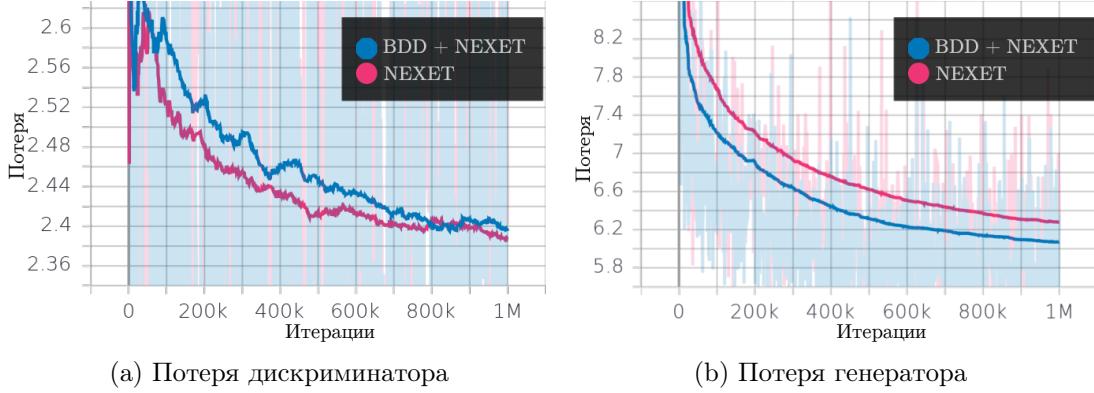


Рис. 21: Графики функция потерь при обучении на NEXET и на NEXET+BDD

После экспериментального сравнения с помощью метрики структурного сходства SSIM, было получено, что значение индекса в переносе из дня в ночь выше почти на 18%, в переносе ночи в день – на 7%. Это дает основания полагать, что эксперимент на расширенной базе совокупности двух датасетов можно считать успешным.

### 5.3 Экспериментальная оценка

В обучении без учителя на непарных данных оценить качество модели является непростой задачей. Для оценки качества переноса были взяты две основные метрики SSIM (Structural Similarity, [24]) и PSNR (Peak Signal-to-Noise Ratio). PSNR построена на вычислении среднего квадратического отклонения, и применить ее в обучении без учителя между противоположными доменами не представляется возможным, но при этом, она подходит для определения качества обучения по точности реконструкции модели. Метрика структурного сходства, напротив, позволяет сравнивать переносимое изображение с исходным, так как она учитывает структурное изменение информации. В табл. 5 приведена экспериментальная оценка результатов работы программы. Также была проведена оценка работы модели по точности выделения объектов на изображении, и о ее результатах я расскажу в разделе 6.1.

Метрика	Перенос: A → B		Перенос: B → A		Реконструкция	
	CycleGAN	Модель	CycleGAN	Модель	CycleGAN	Модель
SSIM	0.297	0.381	0.390	0.405	0.754	0.954
PSNR	-	-	-	-	20.3	33.7

Таблица 5: Экспериментальная оценка работы модели

A – домен дня, B – домен ночи.

### Оценка скорости работы предобученной модели

После продолжительного обучения результатом работы нейронной сети является предобученная модель, содержащая эмпирические параметры и веса сети, достаточные для переноса изображений между доменами. Для итоговой модели была вычислена скорость рабо-

ты, в зависимости от разрешения изображений и использования глубокой сверточной сети VGG16. По результатам тестирования был построен график зависимости 22.

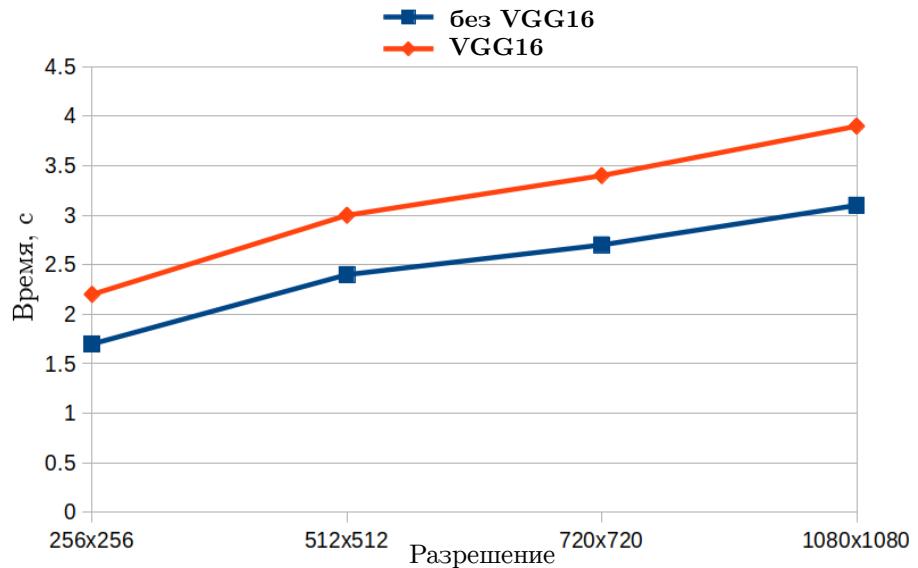


Рис. 22: Зависимость времени работы модели от разрешения и VGG16  
(перенос между доменами одного изображения)

## 6 Результаты

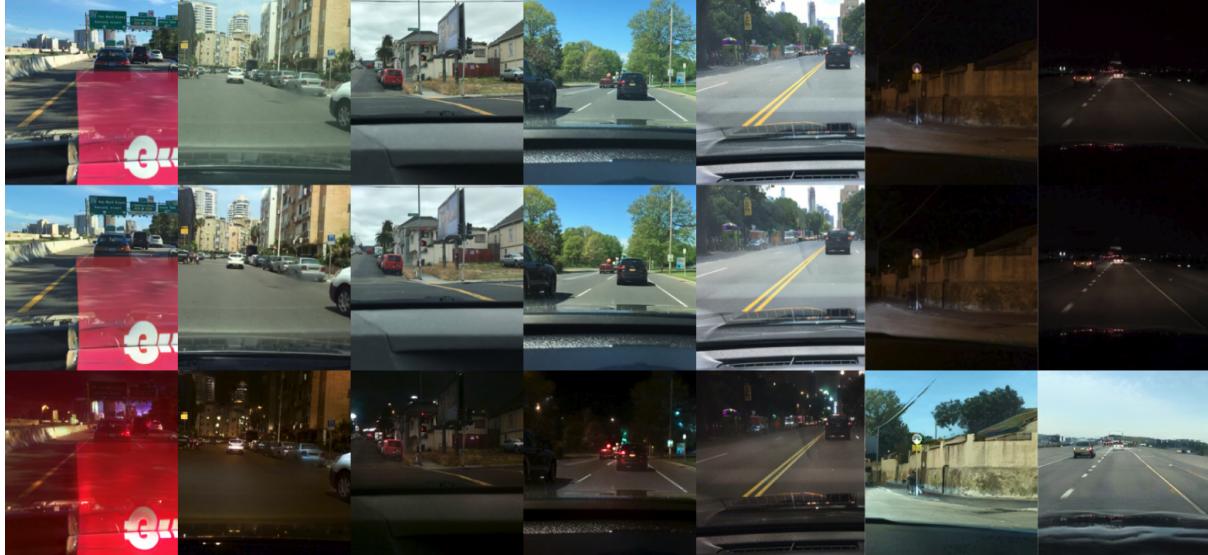
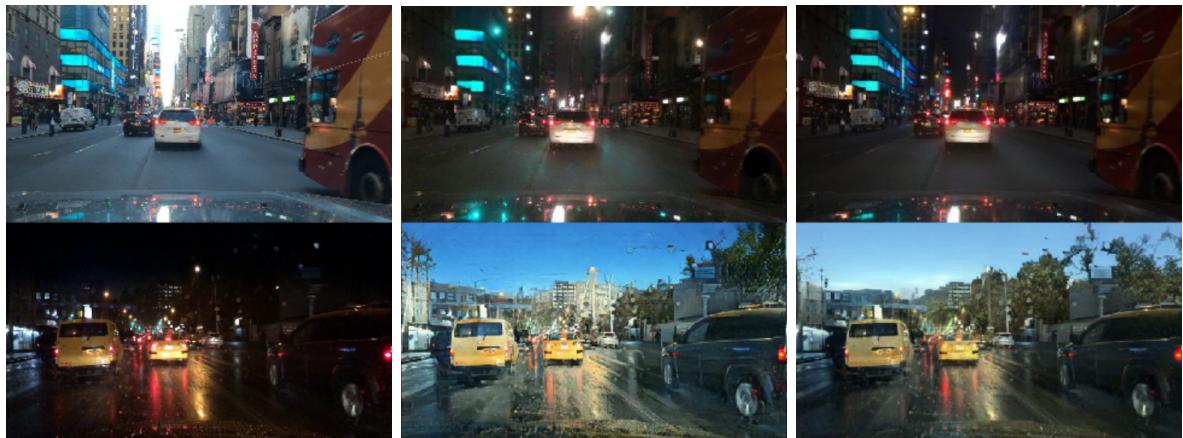


Рис. 23: Результаты работы метода  
сверху-вниз: вход, реконструкция, перенос между доменами

### 6.1 Применение результатов работы в задачах компьютерного зрения

Помимо генерации реалистичных изображений (рис. 23) построенная нейросетевая модель подходит для аугментации данных в смежных задачах компьютерного зрения, тем самым повышая общее качество обучения и результаты работы сетей. В частности, модель может повысить предельную точность выделения объектов на изображениях.

Совместно с магистерской диссертацией [23] было проведено исследование эффективности применения полученной мной модели на примере аугментации данных для выделения автомобилей на изображениях. Сравнение проводилось с нейросетевой моделью CycleGAN [4]. Результаты работы каждой из моделей представлены на рис. 24.



(a) Входное изображение      (b) CycleGAN [4]      (c) Применение модели

Рис. 24: Визуальное сравнение работы методов

После обучения детектора (подробнее о его работе можно узнать тут [23]) на предоставленной модели, был получен результирующий AP (Average Precision) выше, чем на реальных данных, что означает, общее повышение качества обучения, а значит подтверждает заявленную эффективность применения модели.

В сравнении с конкурентным алгоритмом CycleGAN [4], основанном на схожей архитектуре, был отмечен больший рост точности детекции автомобилей на изображениях дня. На аугментации с помощью CycleGAN детектор получил более высокие значения на изображениях ночи. Полные результаты проведенного сравнения были взяты из исследования [23] и представлены в таблице 6.

<b>RetinaNet</b>	<b>День</b>	<b>Ночь</b>	<b>Все</b>
Настоящие данные	0.8664	0.8406	0.8535
CycleGAN	0.8701 (+0.42%)	<b>0.8571 (+1.96%)</b>	<b>0.8636 (+1.18%)</b>
Модель	<b>0.8749 (+0.98%)</b>	0.8512 (+1.26%)	0.8631 (+1.12%)

Таблица 6: Результаты полученные детектором RetinaNet [23]

## Выход

Таким образом, в совместном исследовании с [23] подтверждается, что в работе получена конкурентоспособная модель изменения времени суток на изображении. Был отмечен прирост точности по сравнению с альтернативными методами аугментации на дневных изображениях, откуда можно сделать вывод, что перенос из ночи в день был выполнен успешнее у моей модели. Однозначно выделить победителя не удалось, так как детектор получает лучшие результаты на изображениях ночи у CycleGAN [4].

В любом случае, CycleGAN и UNIT, на котором основана моя модель, имеют схожую архитектуру, о чем уже было рассказано в разделах 3.5.1, 3.5.2 и отмечается в работе [11], откуда и следует близость их результатов. В совокупности с полученной экспериментальной оценкой и метрикой сходства SSIM, табл. 5, можно утверждать что теоретически оба метода имеют достаточную точность для выполнения реалистичного переноса изображений, в частности подходят для задачи изменения времени суток в равной степени.

На практике для генерации наиболее реалистичных изображений больше подойдет модель обученная на объединенном датасете BDD100K+NEXET, которая была получена уже после совместного эксперимента и показала существенный прирост в точности, см. раздел. 5.2.4.

## 7 Программная реализация

### 7.1 Инструментарий

В работе использованы различные средства для проведения вычислений и обучения. Большая часть экспериментов была поставлена на графических картах лаборатории: NVIDIA GTX 1080 Ti, но также вычисления проводились на сервере Google Colaboratory с помощью Tesla K80 [29] и на видеокарте NVIDIA RTX 2080. В качестве языка реализации был выбран Python 3.6. Все тесты проводились в виртуальных Docker контейнерах с предустановленной операционной системой семейства Linux, Ubuntu 18.04. Экспериментальная оценка была выполнена в интерактивной оболочке Jupyter Notebook, часть анализа работы алгоритма проводилась с помощью TensorBoard – инструмента визуализации этапов обучения нейронных сетей.

### 7.2 Программный код

Всего было написано 7 программ на языке Python, выполняющие тестирование, подготовку данных для обучения и экспериментальную оценку результатов, общей сложностью более 800 строк кода. И более 250 строчек было написано для реализации обучения и модификации используемой нейронной сети. Репозиторий с исходный кодом [22].

### Библиотеки

Основными используемыми библиотеками были:

- Pytorch – весь каркас обучения, библиотека для работы с нейросетью
- OpenCV, Pillow, imageio – библиотеки для работы с изображениями
- Pandas, Numpy – для работы с большими объемами данных и вычислениями
- Matplotlib, Scikit-image – библиотеки для экспериментальной оценки результатов

## 8 Заключение

В рамках работы были решены следующие задачи:

- Проведен обзор существующих методов переноса изображений и анализ решений родственных задач. На основе проведенного исследования и обзора литературы, выбрана модель, подходящая для решения поставленной задачи наилучшим образом
- Собраны применимые к проблеме тренировочные датасеты – изображения с различными временем суток [25, 26]. Проведена проверка и устранение ошибочных данных из датасетов
- Проведена серия экспериментов переноса изображений, получена экспериментальная оценка качества и скорости работы модели, найдены оптимальные конфигурации нейронной сети

Таким образом, поставленная цель получения предобученной модели, способной изменить время суток входного изображения выполнена. Полученный алгоритм достигает достаточной точности на объединенной базе изображений по метрикам структурного сходства. Это позволяет успешно использовать модель в смежных задачах машинного обучения для аугментации данных [23].

## 9 Список литературы

- [1] **Supervised and Unsupervised Image Translation** – A collection of image to image papers with code, 2019.
- [2] **J. Johnson, A. Alahi, L. Fei-Fei.** *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. Department of Computer Science, Stanford University, March 2016.
- [3] **R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A.S. Lin, T. Yu, A.A. Efros.** *Real-Time User-Guided Image Colorization with Learned Deep Priors*. University of California, Berkeley, May 2017.
- [4] **J.-Y. Zhu, T. Park, P. Isola, A.A. Efros.** *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. Berkeley AI Research (BAIR) laboratory, UC Berkeley, March 2017.
- [5] **P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros.** *Image-to-image translation with conditional adversarial networks*. Berkeley AI Research (BAIR) laboratory, UC Berkeley, November 2016.
- [6] **Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, L. Lin.** *Unsupervised Image Super-Resolution using Cycle-in-Cycle Generative Adversarial Networks*. Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Graduate School at Shenzhen, Department of Automation, Tsinghua University, September 2018.
- [7] **M.-Y. Liu, T. Breuel, J. Kautz.** *Unsupervised Image-to-Image Translation Networks*. NVIDIA Corporation, March 2017.
- [8] **L. Ma, X. Jia, S. Georgoulis, T. Tuytelaars, L.V. Gool.** *Exemplar Guided Unsupervised Image-to-Image Translation*. Berkeley AI Research, UC Berkeley, Adobe Research, March 2019.
- [9] **X. Huang, M.-Y. Liu, S. Belongie, J. Kautz.** *Multimodal Unsupervised Image-to-Image Translation*. Cornell University, NVIDIA Corporation, April 2018.
- [10] **J.-Y. Zhu, R. Zhang, D. Pathak, A.A. Efros.** *Toward Multimodal Image-to-Image Translation*. Berkeley AI Research, UC Berkeley, Adobe Research, November 2017.
- [11] **P. Welander, S. Karlsson, A. Eklund.** *Generative Adversarial Networks for Image-to-Image Translation on Multi-Contrast MR Images - A Comparison of CycleGAN and UNIT*. Linkoping University, Linköping, Sweden, June 2018.
- [12] **H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, M.-H. Yang.** *Diverse Image-to-Image Translation via Disentangled Representations*. European Conference on Computer Vision, August 2018.
- [13] **M.-Y. Liu, O. Tuzel.** *Coupled generative adversarial networks*. Mitsubishi Electric Research Labs, (MERL), 2016.
- [14] **С. Николенко, А. Кадурин, Е. Архангельская.** *Глубокое обучение*. СПб.: Питер, 2018.
- [15] **A. Krizhevsky, I. Sutskever, G.E. Hinton.** *ImageNet Classification with Deep Convolutional Neural Networks*. University of Toronto, 2012

- [16] **K. He, X. Zhang, S. Ren, J. Sun.** *Deep Residual Learning for Image Recognition*. Microsoft Research, December 2015
- [17] **K. Simonyan, A. Zisserman.** *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Visual Geometry Group, Department of Engineering Science, University of Oxford, April 2015
- [18] **S. Ioffe, C. Szegedy.** *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. Google Inc., March 2015
- [19] **J.L. Ba, J.R. Kiros, G.E. Hinton.** *Layer Normalization*. University of Toronto and Google Inc., July 2016
- [20] **D. Ulyanov, A. Vedaldi, V. Lempitsky.** *Instance Normalization: The Missing Ingredient for Fast Stylization*. Computer Vision Group, Skoltech & Yandex and Visual Geometry Group, University of Oxford, November 2017
- [21] **The State of the Octoverse: machine learning – GitHub: Report**, January 2019.
- [22] **The Repository of Day2Night Research – source code**, May 2019.
- [23] **Д.И. Савинкина.** *Преобразование изображений для обучения детекторов объектов*. дис. маг. фак. ВМиК, Московский Государственный Университет им. Ломоносова Москва, 2019.
- [24] **Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli.** *Image quality assessment: from error visibility to structural similarity*. IEEE Transactions on Image Processing, 2004
- [25] **NEXET Dataset**, the Nexar Challenge II, Road Images, 2017
- [26] **BDD100K: A Large-scale Diverse Driving Video Database**, Berkeley AI Research (BAIR) laboratory, UC Berkeley, 2018
- [27] **M. Milford, G. Wyeth – Alderley Day/Night Dataset**, IEEE International Conference on Robotics and Automation, St Paul, United States, 2012
- [28] **Paired WebCam Night2Day Dataset**, UC Berkley, pix2pix and BycicleGAN projects, 2018
- [29] **Google Colaboratory**, Google Research Server, Tesla K80