**«Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)»**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | [01.03.02 Прикладная математика и информатика](http://www.eltech.ru/ru/abiturientam/napravleniya-podgotovki/bakalavriat/01040062-prikladnaya-matematika-i-informatika) | |
| **Профиль** | Математическое обеспечение программно-информационных систем | |
| **Факультет** | КТИ | |
| **Кафедра** | МО ЭВМ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | А.А. Лисс |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

**Тема:** **РАЗРАБОТКА БЕНЧМАРКА МОДЕЛЕЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ 3D-РЕКОНСТРУКЦИИ ПРОТЯЖЕННЫХ ОБЪЕКТОВ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент |  |  | |  | В.А.Шангичев |
|  |  | *подпись* | |  |  |
| Руководитель | к.т.н., доцент |  | |  | А.А. Лисс |
|  |  | *подпись* | |  |  |
| Консультанты |  |  | |  | Р.П. Шестопалов |
|  |  | *подпись* | |  |  |
|  |  |  | |  | А.И. Маловский |
|  |  | *подпись* | |  |  |
|  | к.т.н. |  | |  | М.М. Заславский |
|  |  | *подпись* | |  |  |
|  |  | |

Санкт-Петербург

2024**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | Шангичев В.А. | | | |  | Группа | 0382 | |
| Тема работы: Разработка бенчмарка моделей искусственного интеллекта для 3D-реконструкции протяженных объектов | | | | | | | | |
| Место выполнения ВКР: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина) | | | | | | | | |
| Исходные данные (технические требования):  Любое ОС, обеспечивающие доступ к платформе Kaggle | | | | | | | | |
| Содержание ВКР:  Введение, Выбор моделей монокулярной оценки глубины, Выбор датасета, Выбор метрик, Оценка моделей, | | | | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал, Безопасность жизнедеятельности | | | | | | | | |
| Дополнительные разделы: Безопасность жизнедеятельности | | | | | | | | |
|  | | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | | |
| «\_02\_»\_\_апреля\_\_\_\_\_2024\_ г. | | | «14»\_\_июня\_\_\_2024\_ г. | | | | | |
|  | | |  | | | | | |
| Студент | |  | | В.А. Шангичев | | | |
| Руководитель к.т.н., доцент | |  | | А.А. Лисс | | | |
| Консультант | |  | | Р.П. Шестопалов | | | |

**КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ**

**ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_А.А. Лисс |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Шангичев В.А. |  | Группа | 0382 |
| Тема работы: Разработка бенчмарка моделей искусственного интеллекта для 3D-реконструкции протяженных объектов | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме работы | 02.04 – 14.04 |
| 2 | Подготовка набора данных | 14.04 – 22.04 |
| 3 | Реализация метрик | 22.04-28.04 |
| 4 | Оценка моделей | 28.04-05.05 |
| 5 | Безопасность жизнедеятельности | 05.05-07.05 |
| 6 | Оформление пояснительной записки | 07.05-01.06 |
| 7 | Оформление иллюстративного материала | 01.06 – 04.06 |
| 8 | Предзащита | 07.06.2024 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | |  | | В.А. Шангичев | |
| Руководитель к.т.н., доцент | |  | | А.А. Лисс | |
| Консультант | |  | | Р.П. Шестопалов | |
|  | | |  | |  | |
|  |

**РЕФЕРАТ**

Пояснительная записка 56 стр., 24 рис., 5 табл., 26 ист.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ, 3D-РЕКОНСТРУКЦИЯ, МОНОКУЛЯРНАЯ ОЦЕНКА ГЛУБИНЫ

**Объектом исследования** являются алгоритмы искусственного интеллекта монокулярной оценки глубины.

**Предметом исследования** является устойчивость к шумам алгоритмов искусственного интеллекта монокулярной оценки глубины.

**Цель работы:** провести сравнительный анализ алгоритмов искусственного интеллекта монокулярной оценки глубины на предмет устойчивости к шумам на изображениях, содержащих протяженные объекты.

В ходе выполнения настоящей выпускной квалификационной работы была измерена устойчивость к шумам 5 моделей искусственного интеллекта монокулярной оценки глубины. В качестве набора данных был использован датасет NYU-Depth V2. В ходе оценки каждой из моделей значения метрик были измерены для исходного набора изображений. Затем фиксировались изменения в значениях метрик на изображениях, подверженных шумам.

**ABSTRACT**

During the course of this graduating work, the noise resistance of 5 artificial intelligence models of monocular depth estimation was measured. The NYU-Depth V2 dataset was used as a data set. During the evaluation of each model, metric values were measured for the original set of images. Changes in metric values in images affected by noise were then recorded.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 8](#_Toc168259233)

[1 Выбор моделей монокулярной оценки глубины 10](#_Toc168259234)

[1.1 Zoedepth [7] 10](#_Toc168259235)

[1.2 LapDepth [11] 14](#_Toc168259236)

[1.3 PixelFormer [14] 19](#_Toc168259237)

[1.4 GLPN [17] 23](#_Toc168259238)

[1.5 Depth Anything [19] 26](#_Toc168259239)

[2 Описание условий проводимого эксперимента 28](#_Toc168259240)

[3 Выбор датасета 29](#_Toc168259241)

[4 Выбор метрик 30](#_Toc168259242)

[4.1 Математическое описание используемых метрик 30](#_Toc168259243)

[4.2 Программная реализация используемых метрик 31](#_Toc168259244)

[5 Выбор модели шума 33](#_Toc168259245)

[6 Оценка моделей на изображениях с разной интенсивностью шума 36](#_Toc168259246)

[6.1 Получение значений 36](#_Toc168259247)

[6.2 Анализ результатов эксперимента 41](#_Toc168259248)

[7 Безопасность жизнедеятельности 47](#_Toc168259249)

[7.1 Основные положения 47](#_Toc168259250)

[7.2 Анализ соответствия среды разработки установленным нормам. 49](#_Toc168259251)

[7.3 Выводы 51](#_Toc168259252)

[ЗАКЛЮЧЕНИЕ 53](#_Toc168259253)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 54](#_Toc168259254)

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

Алгоритм искусственного интеллекта - модель (здесь и далее термины "модель" и "алгоритм" используются взаимозаменяемо), способная выполнять задачу (или несколько задач), успешное решение которой может быть охарактеризовано, как уникальное проявление процесса человеческого мышления. Подобные задачи можно разделить на области, среди которых: "Компьютерное зрение", "Текстовые переводы", "Преобразование звука в текст" и многие другие.

Алгоритм машинного обучения - алгоритм искусственного интеллекта, для формирования численных параметров которого использовался набор данных. Процесс вычисления параметров на основе данных именуется обучением.

Алгоритм глубокого обучения - алгоритм машинного обучения, работа которого представлена обработкой входных данных посредством последовательного применения набора слоев. Каждый слой представляет собой комбинацию вычислений, нередко представимых в виде операций над тензорами (массив данных любой размерности). Результаты обработки входных данных слоями могут быть скомбинированы посредством сложения, соединения, умножения и других операций, вследствие чего модель глубокого обучения условно может быть представлена как граф, узлами которого являются слои. Распространенным подмножеством алгоритмов глубокого обучения являются модели, параметры которых оптимизируются при помощи метода градиентного спуска.

Пулинг – слой модели глубокого обучения, как правило применяемый в моделях, ориентированных на решение задач компьютерного зрения. Представляет собой скользящее окно, где из всех значений берется максимальное.

Сверточный слой – слой модели глубокого обучения, включающий в себя фильтр для каждого канала, ядро свёртки которого обрабатывает признаки предыдущего слоя по фрагментам.

## ВВЕДЕНИЕ

3D-реконструкция является процессом воссоздания формы реального объекта. Данный процесс находит применение в медицине, археологии, наблюдении Земли, робототехнике, проектировании дорожных покрытий. Существуют разные способы проведения данного процесса. Методы 3D-реконструкции можно подразделить на активные и пассивные.

Активные методы включают в себя построение модели объекта посредством физического с ним воздействия. Для этого используют специальные устройства: лидары, глубинометры, микроволновые датчики. Пассивные методы 3D-реконструкции не взаимодействуют с объектом напрямую и восстанавливают форму объекта посредством анализа его изображений. Так, алгоритмы фотограмметрии восстанавливают 3D-модель реального объекта с использованием нескольких изображений и информации о положениях и характеристиках камеры, с которой они были сделаны. Алгоритмы “структуры-из-движения” осуществляют 3D-реконструкцию без использования информации о положениях камер.

Алгоритмы монокулярной оценки глубины оценивают форму объекта с использованием лишь одного изображения. По сравнению с вышеперечисленными способами такой подход имеет ряд преимуществ: во-первых, алгоритмы монокулярной оценки глубины требуют лишь один снимок объекта, что позволяет проводить процесс 3D-реконструкции с использованием одной лишь фотокамеры. Потребность в анализе только одной фотографии делает алгоритмы монокулярной оценки глубины менее затратными в вычислительных ресурсах и позволяет формировать датасеты меньших размеров для обучения (в случае алгоритмов машинного обучения).

В ходе развития области монокулярной оценки глубины были созданы датасеты для оценки и сравнения существующих подходов. Датасет NYU-depth V2 [1] содержит снимки комнат Нью-Йорка и соответствующие карты глубины, полученные с использованием камеры Kinect. Датасет Kitti-Eigen Split [2] позволяет оценить алгоритмы монокулярной оценки глубины на изображениях городских улиц. Датасет CityScapes содержит изображения зданий. Все эти датасеты предоставляют обширные возможности для оценки эффективности подходов монокулярной оценки глубины.

Тем не менее процесс получения фотосъемки неидеален, и выходное изображение подвержено шумам. Изучение влияния шумов изображения объекта на результат работы алгоритмов монокулярной оценки глубины может помочь выбрать наиболее устойчивые к шумам подходы, что, в свою очередь, зачастую является существенным в вопросе применения того или иного подхода в реальных условиях. В данной работе будет использован датасет NYU-depth V2, содержащий протяженные комнаты. Фотографии, сделанные внутри помещения, могут содержать значительное количество шумов, в связи с отличительной неравномерностью освещения внутри апартаментов.

**Объектом исследования** являются алгоритмы искусственного интеллекта монокулярной оценки глубины.

**Предметом исследования** является устойчивость к шумам алгоритмов искусственного интеллекта монокулярной оценки глубины.

**Цель работы:** провести сравнительный анализ алгоритмов искусственного интеллекта монокулярной оценки глубины на предмет устойчивости к шумам на изображениях, содержащих протяженные объекты.

**Практическая ценность работы:** подобный анализ позволит выявить архитектуры, наиболее устойчивые к шумам.

Задачи:

Выбор моделей монокулярной оценки глубины для проведения сравнительного анализа.

Выбор датасета.

Выбор метрик.

Выбор модели шума.

Замер результатов работы алгоритмов на разных уровнях шума с последующим анализом.

## 1 Выбор моделей монокулярной оценки глубины

### 1.1 Zoedepth [7]

Основой данной модели является блок DPT [3] (*англ*. Dense prediction transformer). Данная архитектура является попыткой устранения недостатков классических моделей компьютерного зрения, использующих большое количество сверточных слоев с постепенным уменьшением размеров карт признаков.

Глубокие сверточные сети увеличивают рецептивное поле (размер окна исходного изображения, информацию о котором содержит отдельный вектор признаков) и позволяют переводить низкоуровневые элементы исходного изображения в абстрактные высокоуровневые признаки. Данная особенность не является препятствием в некоторых задачах, например, в классификации изображений. Однако, в данном процессе происходит потеря разрешения и детализации, что может помешать в таких задачах, как сегментация изображения, обнаружение объектов, оценка глубины.

Основой модели DPT является ViT (*англ.* Vision Transformer) [4] – архитектура, созданная на базе архитекруры трансформеров, разработанных с целью решения задач NLP, адаптированная под обработку изображений. Таким образом, каждое изображение сначала представляется в виде набора векторов-слов, после чего может быть передано на вход трансформеру. Сначала изображение разделяется на квадратные неперекрывающиеся куски. Затем каждый пиксель в каждом куске расценивается, как компонента вектора. Все вектора преобразуются посредством слоя линейной проекции.

Другим вариантом извлечения токенов является применение модели ResNet50 [5]. Токены извлекаются из признаков результатов работы модели. Также Vit добавляет специальный токен с обучаемыми параметрами, который призван собирать глобальные признаки изображения. Полученные токены многократно подаются на вход трансформеру. Таким образом, на каждой серии итераций работы трансформера формируются наборы выходных токенов. Каждый из выходных наборов декодируется в набор признаков с помощью операции *Reassemble (англ.* Собрать заново*):*

где обозначает отношение размеров выходной карты признаков к размерам исходного изображения, а – размерность выходных признаков.

Операция Read (*англ.* Чтение):

Где – количество токенов, не считая дополнительного при разделении изображения на квадраты размером . Данная операция позволяет обработать дополнительный токен. Дополнительный токен может как участвовать в формировании выходной карты признаков, так и быть полностью проигнорирован. Следующей операцией является операция соединения:

После этого выполняется операция передискретизации:

Данное преобразование выполняется с использованием сверточного слоя с размером окна 1х1 с последующим применением пошагового или транспонированного сверточного слоя с размером окна 3х3 в зависимости от знака неравенства между величинами и . Наконец, полученные признаки комбинируются посредством метода, взятого за основу в блоке смешения признаков модели RefineNet [6] (рисунок 1).

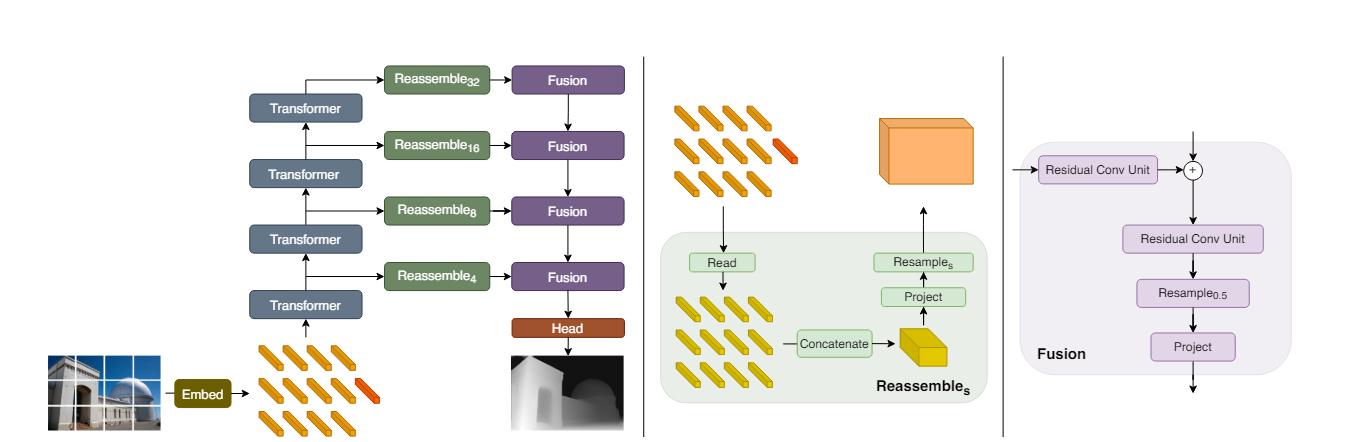
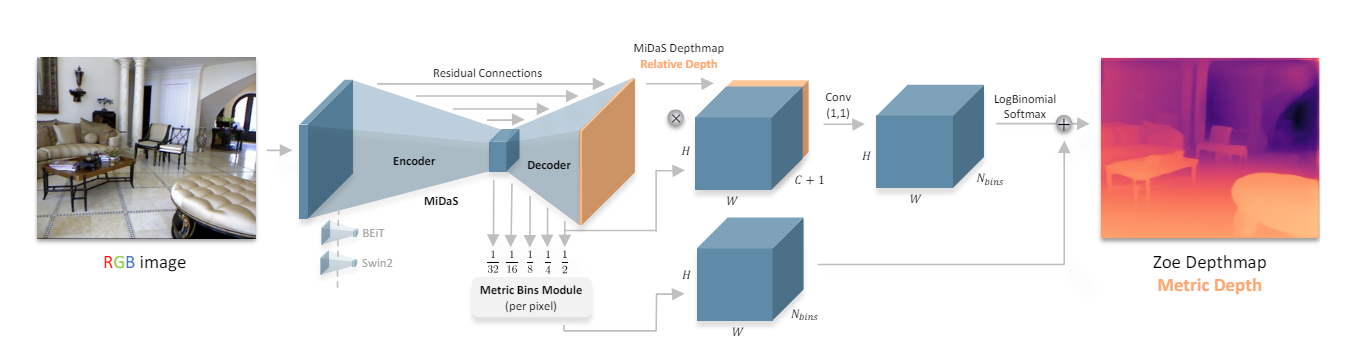


Рисунок 1 – Схема работы DPT

ZoeDepth использует вышеописанную архитектуру, однако энкодер заменен на более поздние модификации [8]. Одним из приемов получения карты глубины является предсказание карты относительной глубины (безразмерной) с последующим преобразованием в метрические величины (например, метры). Zoedepth использует данный подход. Тренировка блока предсказания относительной глубины происходит в соответствии с инструкциями, описанными в MiDas [9].

Тренировка блока с предсказанием относительной глубины позволяет расширить тренировочную выборку (разные датасеты имеют разную размерность карт глубины, однако обучение с применение предсказания относительной глубины позволяет нивелировать данное ограничение). Полученная карта относительной глубины впоследствии преобразовывается с использованием “Модуля метрических делений” (*англ. Metric bins module*). Полученные признаки обрабатываются и в результате из относительной карты глубины выводится карта глубины метрическая (Рисунок 2).

Рисунок 2 – Архитектура модели ZoeDepth

В основе модуля метрических делений лежит архитектура LocalBins [10]. Модуль метрических делений действует следующим образом (рисунок 3). Формируется выборка признаков в пяти разных уровнях декодера. После этого признаки конвертируются в вектора с помощью многослойного перцептрона. Выходные признаки агрегируются посредством сложения и участвуют в формировании выходной карты глубины.

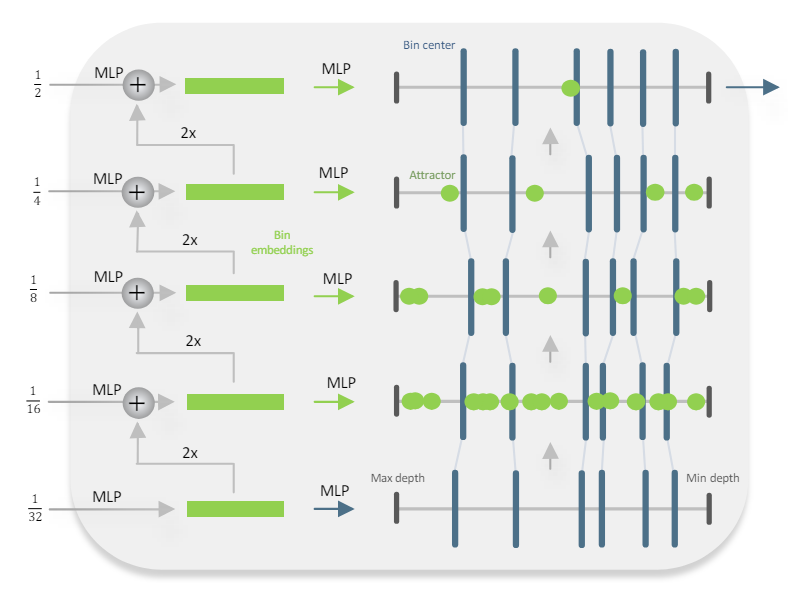
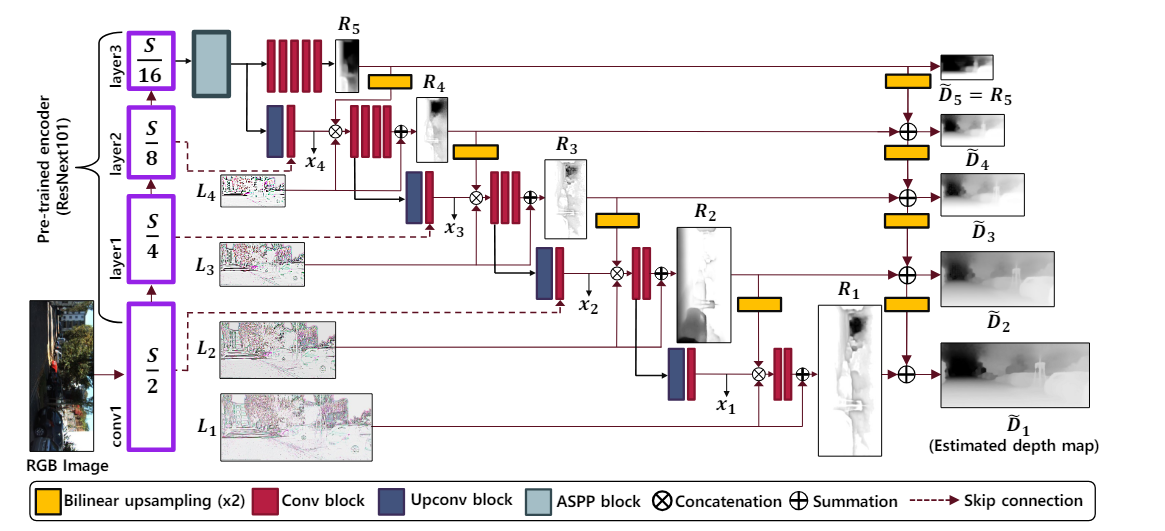


Рисунок 3 – Модуль метрических делений

### 1.2 LapDepth [11]

Архитектура модели изображена на рисунке 4.

Рисунок 4 – Архитектура модели LapDepth

В качестве энкодера используется архитектура ResNext101 [12]. Данная архитектура состоит из сверточных слоев, слоев макс-пулинга. Также в модели есть остаточные соединения (*англ. Residuals*). Остаточные соединения призваны способствовать устранению эффекту затухания градиента. Используются веса модели, формируемые в процессе обучения для задачи классификации изображений. В ходе решения данной задачи модель ResNext101 обучалась формировать ценные представления признаков, которые могут быть полезны при использовании ее в качестве энкодера в задаче монокулярной оценки глубины.

В основе данной архитектуры лежит механизм работы с изображениями, называемый пирамидой Лапласса. Пирамида Лапласса это линейно инвертируемое представление изображения, состоящее из набора последовательно уменьшаемых изображений и их разностей (рисунок 5).

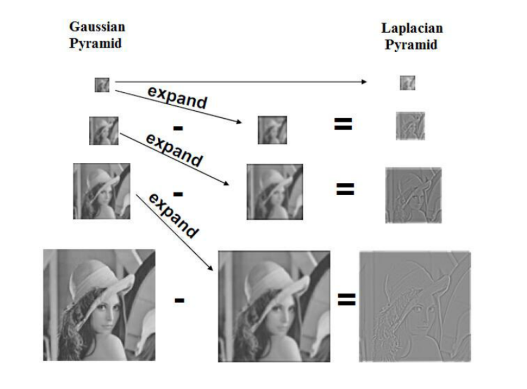


Рисунок 5 – Схема пирамиды Лапласса

Пусть – оператор уменьшения размерности, который преобразовывает изображение размером таким образом, что имеет размеры . Также введем оперцию , которая увеличивает размеры изображения в два раза. Таким образом будет являться изображением с размерами . Сначала выполняется построение пирамиды Гаусса: где и является результатом применения отображения раз к изображению . – количество уровней в пирамиде. Коэффициенты на каждом уровне пирамиды Лапласса формируются посредством взятия разницы между соседними уровнями пирамиды Гаусса с применением функции к уровню, имеющему меньшую размерность, чтобы размеры были совместимы:

Таким образом, каждый из уровней представляет структуру изображения, представленного в различных размерах. Последним уровнем пирамиды Лапласса является не разность, а низкочастотный остаток, равный последнему уровню пирамиды Гаусса, т. е. . Переход от коэффициентов пирамиды Лапласса осуществляется посредством применения обратной подстановки:

Вернемся к изначальной архитектуре. В качестве функции снижения размерности используется модель ResNext101. В качестве функции увеличения размерности была использована билинейная интерполяция.

Билинейная интерполяция активно используется в области компьютерного зрения, так как является инструментом, позволяющим преобразовать любое изображение (или карты признаков) с размерами в изображение с размерами . Первым этапом изменения размеров является составление равномерной выборки точек на поверхности изображения в соответствии с желаемыми размерами итогового изображения (рисунок 6).

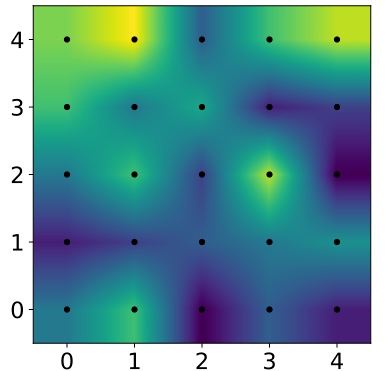


Рисунок 6 – Взятие точек с поверхности исходного изображения

Так как размеры выходного изображения прозвольны, координаты выбранных точек могут иметь дробные значения. Билинейная интерполяция позволяет присвоить каждой из выбранных точек значение на основе значений ближайших пикселов.

Пусть имеется точка с координатами . Предположим, мы знаем значения в четырех соседних точках: (рисунок 7).

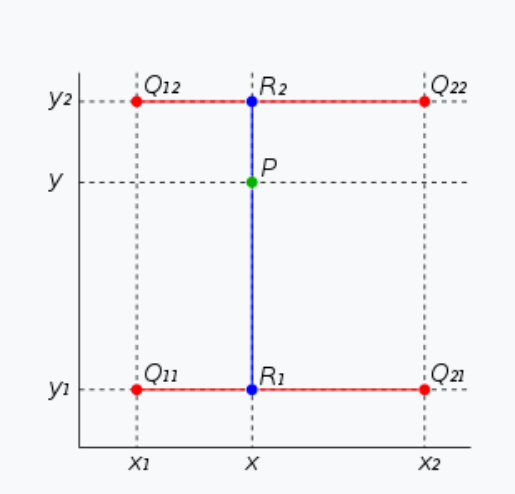


Рисунок 7 – Билинейная интерполяция

Сначала выполняется линейная интерполяция по оси х:

Полученные интерполированные значения используются для интерполяции в направлении оси y:

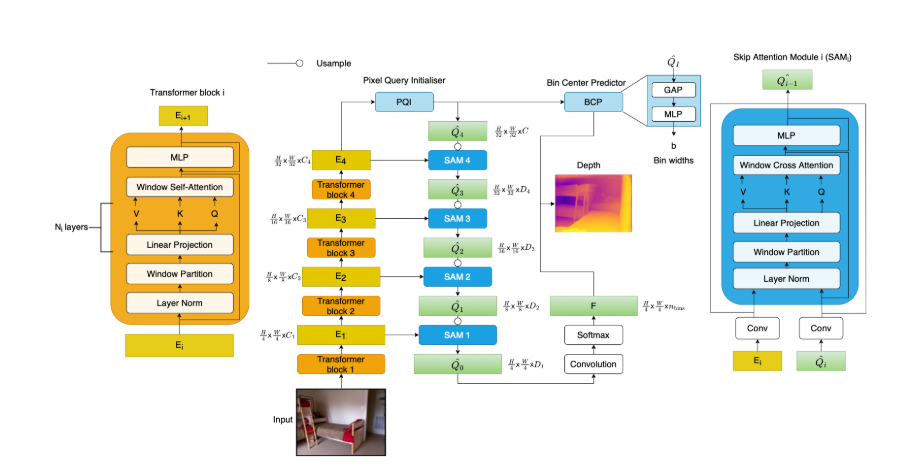
Данный подход легко обобщается на многомерные признаки, достаточно применить билинейную интерполяцию для каждой из компонент.

Верхний уровень пирамиды Лапласса дополняется блоком ASPP [13].

Следует отметить, что данная модель, в силу предсказания метрических карт глубины напрямую, лишена возможности тренировки на разных датасетах, что несомненно может в результате привести к низкой устойчивости к шумам в силу меньшего размера тренировочных выборок.

### 1.3 PixelFormer [14]

Данная архитектура представлена на рисунке 8.

Рисунок 8 – Модель PixelFormer

Входное изображение передается на вход трансформеру Swin [15]. Данный трансформер использует множество слоев для извлечения карт признаков в разрешениях по отношению к исходному изображению. Карта признаков с размерами от размера исходного изображения передается на вход блоку PQI. Данный модуль аггрегирует всю информацию в изображении с использованием глобального пулинга с разным разрешением, формируемого посредством усреднения. Получая на вход признаки, размером где и – исходные размеры изображения, а – глубина карты признаков. Модуль PQI использует блок пирамидального пространственного объединения PSP [16]. Полученные карты признаков затем увеличиваются до исходных ширины и высоты и объединяются. После этого применяется сверточный слой. Выходные признаки имеют размерность , где С = 512.

Полученные представления смешиваются с признаками энкодера с помощью блока SAM. Данный блок работает следующим образом. Для признаков, полученных на этапе работы блока PQI и соответствующих признаков энкодера применяется сверточный слой с размером окна 3 на 3 с одинаковой глубиной выходных признаков. После применения операции свертки формируется матрица запросов из , а также матрицы ключей и матрицы значений , полученных из признаков энкодера в ходе применения весов , реализованных с помощью многослойного перцептрона. Затем, делятся на окна размерами . В реализации модели используется . Пусть – значения соответствующих матриц, соответствующих окну . Тогда выход вычисляется по следующей формуле:

– смещение, представляющее относительную позицию. Представлено обучаемой матрицей с размерами , представляющую относительную позицию вектора, соответствующего каждому запросу и каждой ключевой паре. Функция внимания высчитывается для каждого окна, после чего операция перегруппировки расставляет окна в их соответствующую локацию в .

Для более детального извлечения информации используется головных блоков и функция внимания применяется к каждому из них. После этого векторы глубины аггрегируются с использованием многослойного перцептрона. Таким образом, признаки и обрабатываются следующим образом:

В ходе реализации модели были использованы , где – количество каналов в ядре свертке, которая применяется в начале обработки перед механизмом внимания на уровне . Количество головных блоков где – количество блоков внимания, использованных в модуле SAM на уровне .

В финальной части модели используется блок BCP. Данный блок предсказывает центры специальных контейнеров, которые дискретизируют глубину изображения в фиксированный набор интервалов. Блок состоит из слоя глобального среднего пулинга. После данного слоя следует многослойный перцептрон. Данный блок предсказывает набор интервалов размерностью . В реализации модели данное число рано 256. Таким образом, центры интервалов для входного изображения вычисляются следующим образом:

Работа декодера модели начинается с получения на вход признаков модели , полученных из модуля PQI. Размеры увеличиваются в два раза с применением модели Pixel Shuffle и подаются на вход модулю SAM вместе с соответствующими признаками энкодера :

Для применяется операция свертки для получения финального вектора глубины размером . После этого применяется функция активации Softmax для получения распределения вероятностей для каждого интервала:

Итоговая карта глубины вычисляется по формуле:

Где – значение глубины, предсказанное для пикселя , – центр интервала , – количество интервалов, – вероятность центра интервала для пикселя .

### 1.4 GLPN [17]

Общая архитектура модели представлена на рисунке 9.

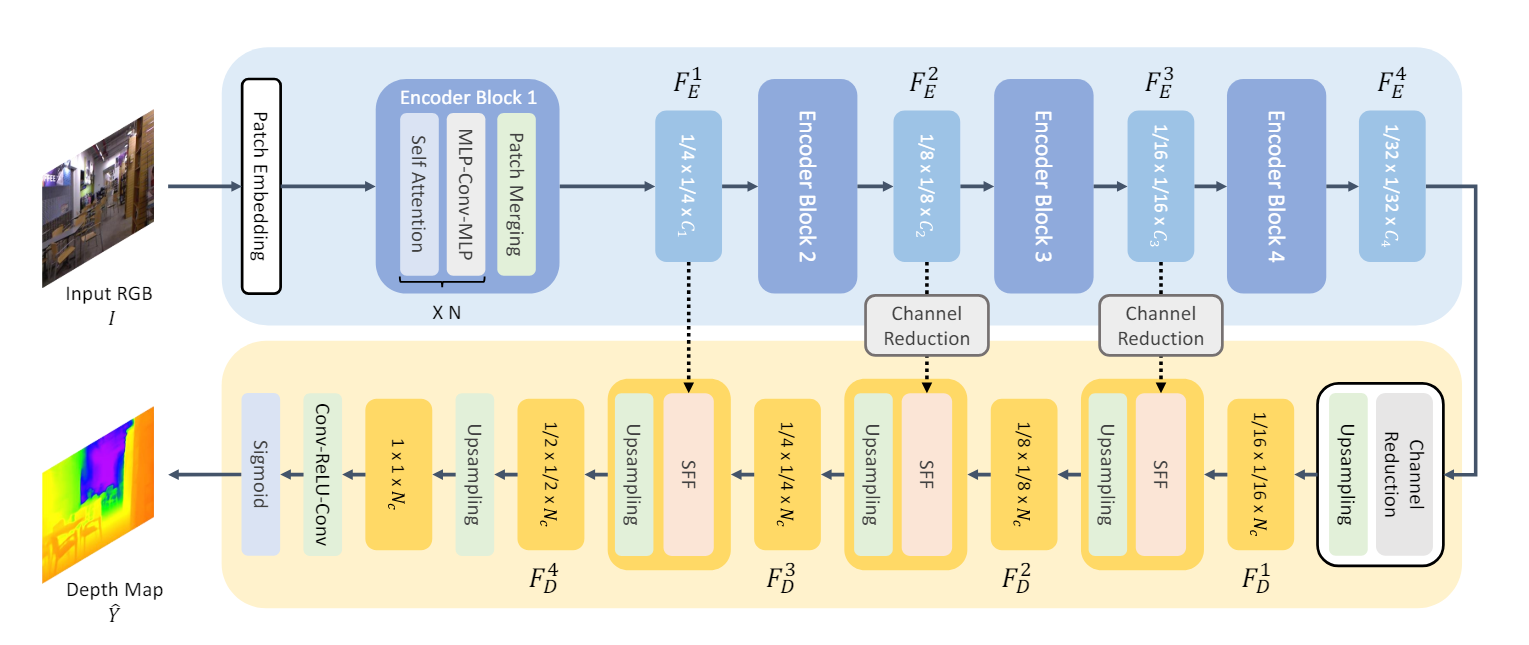


Рисунок 9 – Архитектура модели GLPN

В качестве энкодера для извлечения богатой глобальной информации из изображения используется иерархический трансформер. Для этого изображение векторизуется в последовательность кусков с использованием сверточного слоя с размером окна 3 на 3. После этого вектора передаются на вход блоку трансформера. В данной архитектуре используется четыре блока трансформера, генерирующие генерирующие карты признаков с размерами от размеров исходного изображения с размерностью .

Энкодер преобразовывает входное изображение в признаки с размерностью . Для получения карты глубины необходимо привести признаки к размерам . Сначала размерность признаков понижается до с помощью сверточного слоя с размером окна 1 на 1. После этого размерность повышается до с применением билинейной интерполяции. После этого применяются два сверточных слоя с функцией активации сигмоида для предсказания карты глубины размерностью . После этого карта глубины умножается на максимальное значение глубины для приведения размерности к метрам.

Для интеграции локальных и глобальных признаков используется модуль SFF (рисунок 10).

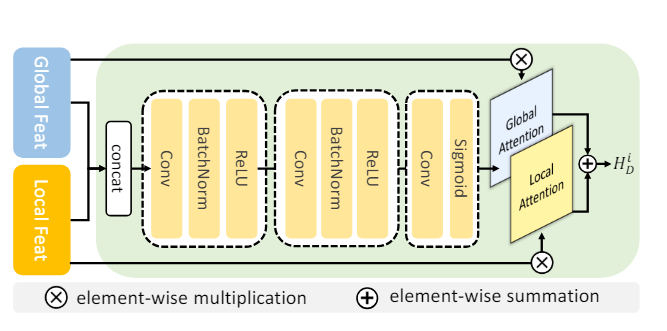


Рисунок 10 – Строение модуля SFF

Для уравнения размерности карт признаков и , признаки энкодера приводятся к глубине при помощи сверточного слоя. Затем признаки объединяются по глубине и подаются на вход двум сверточным слоям с размером окна 3 на 3, идущими последовательно со слоями нормализации пакета и функцией активации ReLU. После этого применяется еще один сверточный слой с последующим применением функции активации сигмоиды. Выход формирует двух канальную карту активации, где каждый локальный и глобальный признак умножается на соответствующий канал для увеличения значений активаций значимых локаций. После поэлементного суммирования данные признаки формируют представления .

В ходе обучения модели применяется метод расширения данных, именуемый “Vertical CutDepth”. Данный подход базируется на алгоритме расширения данных для монокулярной оценки глубины CutDepth [18]. Пусть имеется входное изображение и карта глубины . W и H являются шириной и длиной исходного изображения соответственно, а и – количеством каналов в исходном изображении и карте глубины. Расширенное изображение получается из изображения по формуле:

где матрица из единиц и нулей, обозначающая область, где значения исходного изображения будут заменены на значения карты глубины. Верхний левый угол и размеры прямоугольной области рассчитываются следующим образом:

где берутся из равномерного распределения . Переменная p является гиперпараметром, задающим максимальные размеры окна (рисунок 11).

В ходе обучения модели GLPN используется вертикальное окно, координаты которого высчитываются по формуле:

где и также берутся из равномерного распределения Переменная является гиперпараметром.

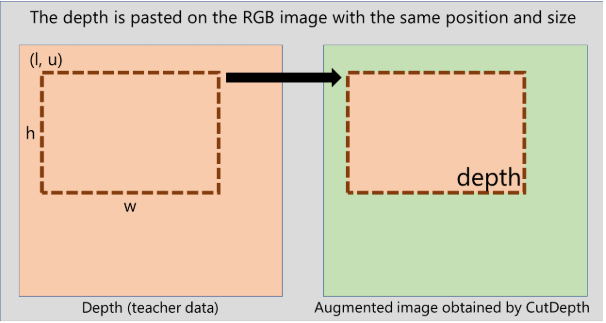


Рисунок 11 – Метод расширения данных CutDepth

### 1.5 Depth Anything [19]

Данная архитектура монокулярной оценики глубины использует в процессе обучения как размеченные, так и неразмеченные изображения. Пусть – набор размеченных изображений (имеющих соответствующую карту глубины), а – набор неразмеченных изображений. Сначала происходит обучение модели на размеченных изображениях . Затем, используется для назначения карт глубины изображениям из набора данных . После этого осуществляется тренировка итоговой модели с обучением на размеченном наборе данных и наборе данных с картами глубины, полученными в ходе работы модели .

Процесс обучения на размеченном наборе данных напоминает тренировку MiDaS. Каждое значение глубины трансформируется в обратное и затем нормализуется таким образом, чтобы значения были между нулем и единицей. Подобные преобразования позволяют использовать разные наборы данных для обучения модели. Для обучения используется следующая функция потерь:

где и предсказанная карта глубины и истинная карта глубины соответственно.

После тренировки модели , для каждого из изображений из неразмеченного набора данных предсказывается глубина:

После этого набор данных используется для тренировки итоговой модели . В ходе обучения модели S применялись приемы расширения данных . В качестве расширения набора данных использовалось размытие по Гауссу, случайные изменения яркости, контраста, насыщенности изображений. Также использовалась методика расширения набора данных CutMix [20]. Осуществляется она следующим образом: выбираются два случайных изображения - и . Расширенное изображение высчитывается по формуле:

где – бинарная маска, где единичные значения содержатся внутри случайного прямоугольника. Функция потерь формируется следующим образом:

Для извлечения признаков используется энкодер DINOv2 [21]. В качестве декодера используется архитектура DPT. Общая архитектура модели представлена на рисунке 12.

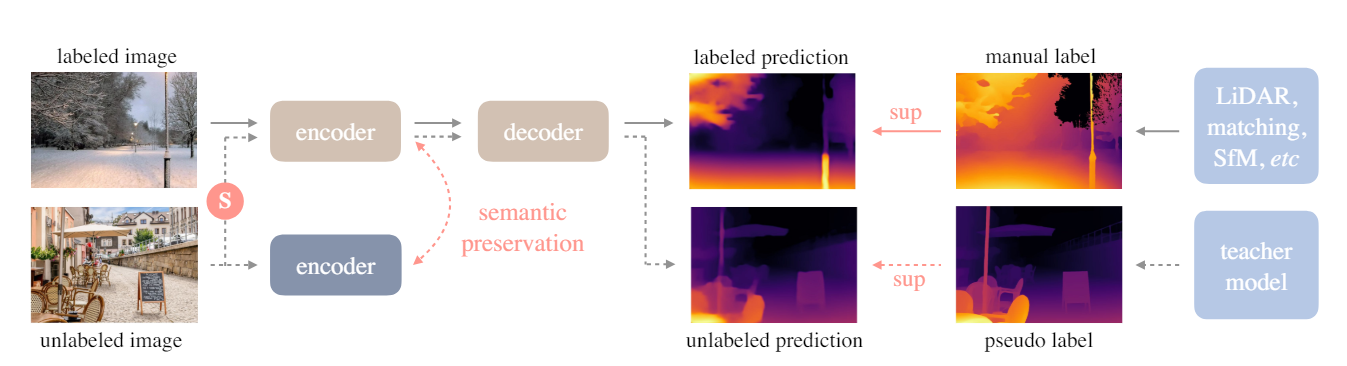


Рисунок 12 – Архитектура модели Depth Anything

## 2 Описание условий проводимого эксперимента

Модели ZoeDepth, PixelFormer, GLPN, DepthAnything были протестированы с использованием интерактивного блокнота, предоставляемого платформой Kaggle. В качестве графического ускорителя была использована бесплатно предоставляемая платформой видеокарта NVIDIA Tesla P100.

Модель LapDepth была протестирована на платформе Google Colab с бесплатно предоставляемым графическим ускорителем NVIDIA Tesla T4.

## 3 Выбор датасета

В качестве набора данных для проведения измерения метрик моделей был выбран набор данных NYU Depth V2, содержащий фотоснимки квартир Нью-Йорка. Данный датасет содержит пары изображение-карта глубины. Карты глубины были получены с использованием камеры Microsoft Kinect. В ходе измерения метрик использовалась версия данных, представленная на платформе Kaggle [23]. Версия платформы содержит изображения, представленные в формате png, в то время как модели изначально тестировались на изображениях jpg. Из-за изменения информации, содержащейся в изображениях, сопутствующего конвертации типов, значения метрик могут несколько отличаться от приведенных в оригинальных статьях. Изображения имеют размеры с 480 пикселями в высоту и 640 в ширину.

Для замеров метрик использовалась тестовая выборка датасета, содержащая 654 пар изображений и карт глубины. Карта глубины, соответствующая изображению с названием rgb\_<number>.png расположена в той же директории, что и изображение, и имеет название depth\_<number>.png. Примеры изображений и соответствующих им карт глубины представлены на рисунке 13.

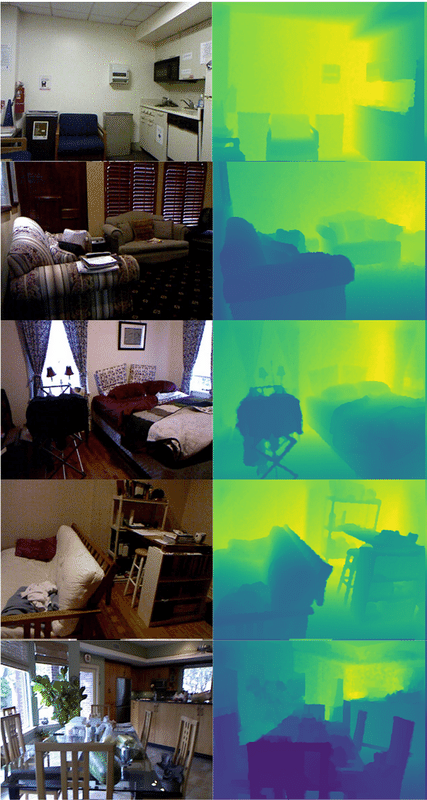


Рисунок 13 – Примеры пар изображений и карт глубины

## 4 Выбор метрик

### 4.1 Математическое описание используемых метрик

Часть используемых в работе метрик была взята из работы [22].

Пусть для изображения с глубинной картой была предсказана глубинная карта . Для оценки схожести предсказанной глубинной карты и истинной используются следующие метрики:

**Абсолютная разница (mae).**

Данная метрика позволяет судить об отклонении значений предсказанной карты глубины от истинных. Тем не менее, в силу поэлементного сравнения, данная метрика не позволяет установить несоответствия в текстуре сравниваемых карт глубины.

**Перепись (census).**

Определим трансформацию . Каждый бит в позиции является флагом, обозначающим, имеет ли соседний пиксель меньшее значение, чем значение обрабатываемого пикселя :

Метрика определяется следующим образом:

где – расстояние Хэмминга, т. е. количество компонент векторов, в которых их значения различаются.

В случае метрик, описанных выше, вычисляются значения метрики для каждой координаты после чего полученные значения усредняются по количеству пикселей в карте глубины.

**Среднеквадратичное отклонение (RMSE).**

Также вычисляется метрика среднеквадратичного отклонения (корень из дисперсии):

**Среднее относительное отклонение (MRE).**

**Дельта-отклонения ().**

Данная метрика позволяет выявить процент пикселей предсказанной карты глубины, значения которых в относительном соотношении отличаются от истинной карты глубины не более установленного порога :

В ходе измерения метрик использовались общепринятые значения порогов: .

### 4.2 Программная реализация используемых метрик

Для подсчета метрик использовалась библиотека языка Python Numpy. Данная библиотека реализует функции для работы с массивами данных. Операции, реализованные в Numpy, выполняются параллельно. Данное обстоятельство позволит значительно увеличить скорость подсчета значений метрик.

Для подсчета дельта-метрики производилось вычисление двух массивов данных. Первый содержит отношение значений предсказанной карты глубины к значениям истинной карты глубины, а второй содержит отношение значений истинной карты глубины к значениям предсказанной карты глубины. После этого полученные массивы объединяются по добавленной в конец оси, по которой выполняется операция максимума. После этого вычисляется доля значений в массиве, которая не превышает заданный порог .

Для вычисления метрики градиента используется функция fftconvolve модуля scipy. Данная функция позволяет применить операцию свертки с использованием двумерного окна. Данная функция применяется для вычисления значений градиентов по осям и для предсказанной карты глубины и для истинной карты глубины. После этого вычисляются суммы абсолютных отклонений с применением функций библиотеки numpy sum и abs.

Для вычисления метрики переписи и ранка была создана вспомогательная фунция neighborhood. Функция принимает на вход массив данных и размер квадратного окна . Во избежание выхода за границу карты глубины, массив данных дополняется нулями с краев таким образом, чтобы для всех значений исходного массива существовали значения внутри вычисляемого окна. После дополнения нулями границ карты глубины нулями функция формирует трехмерный массив данных, содержащий по вертикали пиксель и все соседние с ним значения. Таким образом, выходные данные имеют размер , где – размер пакета.

После этого результат работы функции neighborhood передается на вход функции one\_hot, возвращающей результат трансформации . После этого полученное бинарное представление используется для вычисления метрики ранка с помощью функции rank\_metric и для вычисления метрики переписи при помощи функции census\_metric.

Для вычисления метрик среднеквадратичной ошибки, среднего абсолютного отклонения и средней относительной ошибки используются функции rmse, mae и mre соответственно. Вычисления, выполняемые поэлементно, суммируются и усредняются по количеству пикселей в карте глубины.

## 5 Выбор модели шума

Для моделирования шума было решено использовать модель, описанную в [24]. Общее устройство модели изображено на рисунке 14.

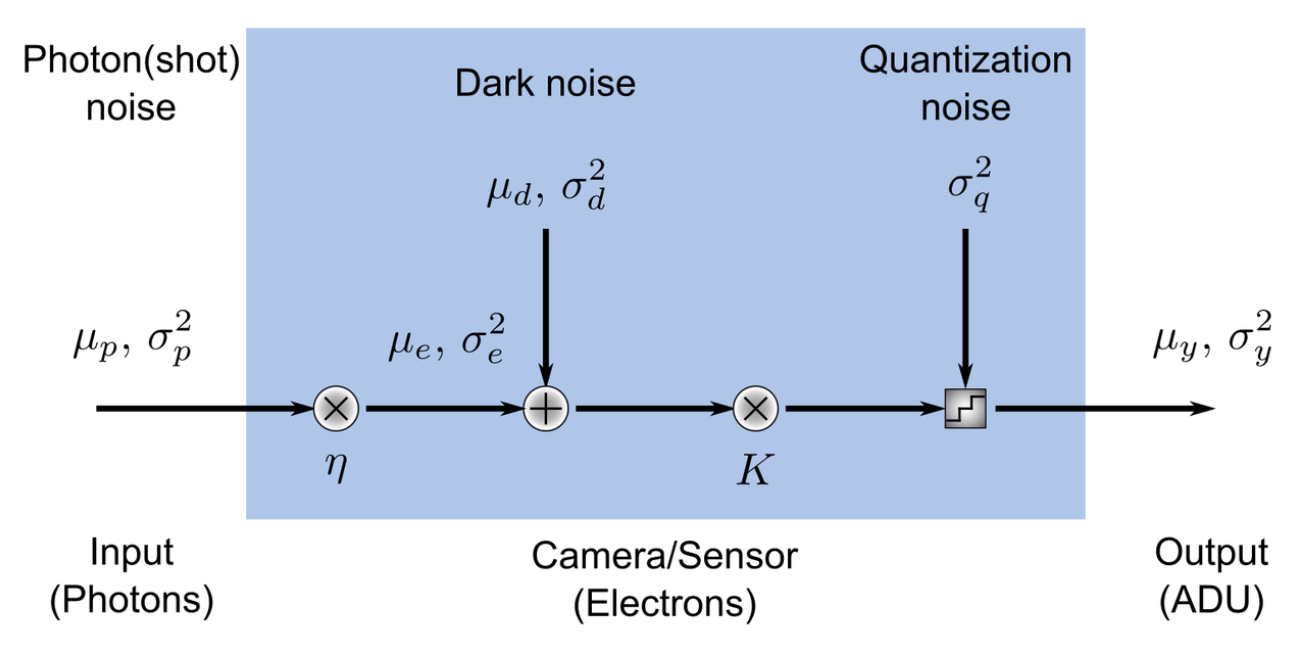


Рисунок 14 – Устройство модели шумов камеры

В данной модели на вход подается количество фотонов, полученное каждым пикселем. Выходом модели является ADU – аналого-цифровые единицы, т. е. значения пикселей камеры. В ходе выполнения данной трансформации различные источники шума добавляются к входному сигналу. В данной модели имеется ряд допущений:

1. Количество фотонов, принятое пикселем, зависит от произведения значения облучения и времени экспозиции
2. Сенсор линеен, т. е. цифровой сигнал увеличивается линейно по отношению к получаемому на вход количеству фотонов
3. Все источники шума постоянны во времени и пространстве
4. Только полная квантовая эффективность камеры зависит от длины волны падающего излучения
5. Только темновой поток зависит от температуры

Сенсор, удовлетворяющий допущениям, описанным выше, называется идеальным датчиком изображения.

Модель зависит от пяти параметров:

1. Полная квантовая эффективность
2. Дисперсия шума
3. Поток фотонов
4. Чувствительность камеры К
5. Разрядность камеры .

Согласно реализации, процесс получения изображения можно описать формулой:

Полученные значения обрезаются в интервал между нулем и .

Алгоритм добавления шумов в изображение следующий:

1. Получение среднего количества фотонов из :
2. Добавление шума по формулам выше

В ходе процесса добавления шума параметрам были назначены следующие значения:

1. Дисперсия шума принимала значения

Пример зашумленного изображения можно видеть на рисунке 15.

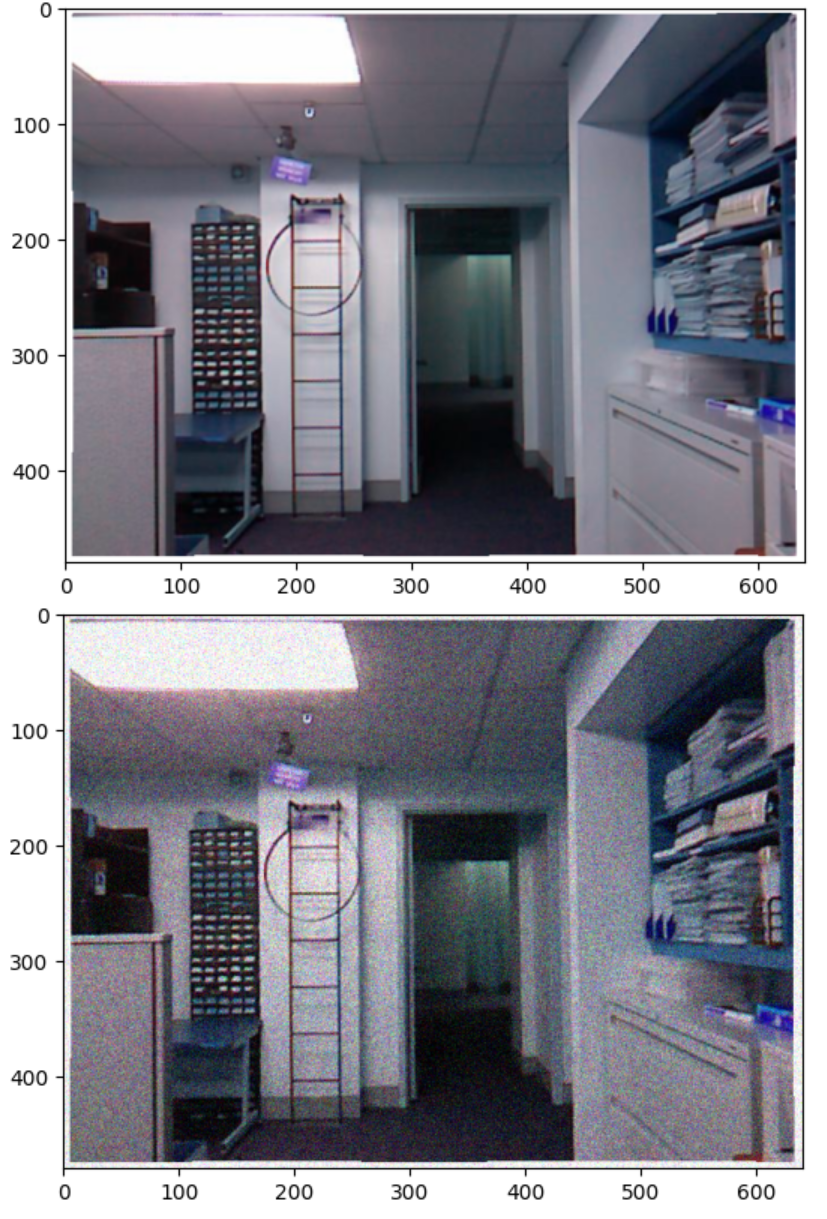


Рисунок 15 – Исходное изображение (сверху) и зашумленное (снизу)

## 6 Оценка моделей на изображениях с разной интенсивностью шума

Каждая модель вычислялась на изображениях из набора данных NYU v2 depth с постепенно увеличивающейся долей шума. Измерения для модели LapDepth производились до тех пор, пока значение метрики относительной ошибки не поднимались 0.215. Такую точность на датасете NYU показывал первый алгоритм глубокого обучения монокулярной оценки глубины [25]. Для остальных моделей измерения проводились до тех пор, пока значения всех метрик, не считая GR и CEN, не ухудшались по сравнению с последними значениями модели LapDepth.

### 6.1 Получение значений

Результаты оценки модели ZoeDepth представлены в таблице 1. Значения были округлены.

Таблица 1 – Значения метрик модели ZoeDepth

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | RMSE | MAE | MRE | CEN |  |  |  |
| 0 | 0.340 | 0.236 | 0.087 | 6.065 | 0.931 | 0.988 | 0.997 |
| 1 | 0.379 | 0.268 | 0.094 | 6.116 | 0.914 | 0.985 | 0.996 |
| 2 | 0.387 | 0.276 | 0.096 | 6.138 | 0.910 | 0.984 | 0.996 |
| 3 | 0.398 | 0.285 | 0.100 | 6.166 | 0.904 | 0.983 | 0.996 |
| 4 | 0.411 | 0.296 | 0.103 | 6.195 | 0.896 | 0.981 | 0.996 |
| 5 | 0.425 | 0.308 | 0.107 | 6.228 | 0.887 | 0.98 | 0.996 |
| 6 | 0.439 | 0.321 | 0.111 | 6.264 | 0.877 | 0.979 | 0.995 |
| 7 | 0.454 | 0.334 | 0.115 | 6.304 | 0.866 | 0.977 | 0.995 |
| 8 | 0.47 | 0.347 | 0.119 | 6.343 | 0.855 | 0.975 | 0.994 |
| 9 | 0.486 | 0.361 | 0.124 | 6.385 | 0.842 | 0.973 | 0.994 |
| 10 | 0.503 | 0.375 | 0.129 | 6.433 | 0.829 | 0.969 | 0.993 |
| 11 | 0.521 | 0.391 | 0.134 | 6.483 | 0.813 | 0.965 | 0.993 |
| 12 | 0.54 | 0.407 | 0.139 | 6.537 | 0.797 | 0.961 | 0.992 |
| 13 | 0.56 | 0.424 | 0.145 | 6.594 | 0.78 | 0.957 | 0.991 |
| 14 | 0.58 | 0.441 | 0.151 | 6.653 | 0.761 | 0.953 | 0.99 |
| 15 | 0.601 | 0.46 | 0.157 | 6.714 | 0.741 | 0.947 | 0.989 |
| 16 | 0.624 | 0.48 | 0.163 | 6.779 | 0.72 | 0.941 | 0.987 |
| 17 | 0.648 | 0.501 | 0.17 | 6.849 | 0.698 | 0.933 | 0.985 |
| 18 | 0.673 | 0.523 | 0.177 | 6.921 | 0.676 | 0.924 | 0.984 |
| 19 | 0.7 | 0.545 | 0.184 | 6.997 | 0.652 | 0.914 | 0.981 |
| 20 | 0.727 | 0.569 | 0.192 | 7.075 | 0.629 | 0.903 | 0.978 |
| 21 | 0.756 | 0.593 | 0.199 | 7.156 | 0.605 | 0.892 | 0.975 |
| 22 | 0.784; | 0.617 | 0.207 | 7.241 | 0.581 | 0.879 | 0.972 |
| 23 | 0.814 | 0.642 | 0.215 | 7.327 | 0.557 | 0.866 | 0.967 |
| 24 | 0.843 | 0.667 | 0.222 | 7.414 | 0.533 | 0.854 | 0.962 |
| 25 | 0.872 | 0.692 | 0.23 | 7.503 | 0.51 | 0.839 | 0.957 |
| 26 | 0.9 | 0.716 | 0.237 | 7.59 | 0.489 | 0.824 | 0.951 |
| 27 | 0.928 | 0.74 | 0.244 | 7.674 | 0.47 | 0.809 | 0.946 |
| 28 | 0.955 | 0.763 | 0.25 | 7.758 | 0.454 | 0.795 | 0.94 |
| 29 | 0.981 | 0.784 | 0.256 | 7.842 | 0.44 | 0.782 | 0.932 |
| 30 | 1.006 | 0.805 | 0.262 | 7.928 | 0.427 | 0.769 | 0.926 |

Результаты оценки модели PixelFormer представлены в таблице 2. Значения были округлены.

Таблица 2 – Значения метрик модели PixelFormer

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | RMSE | MAE | MRE | CEN |  |  |  |
| 0 | 0.403 | 0.285 | 0.106 | 7.690 | 0.892 | 0.981 | 0.995 |
| 1 | 0.463 | 0.335 | 0.118 | 8.205 | 0.853 | 0.972 | 0.993 |
| 2 | 0.473 | 0.343 | 0.121 | 8.252 | 0.846 | 0.969 | 0.992 |
| 3 | 0.488 | 0.355 | 0.124 | 8.305 | 0.835 | 0.966 | 0.992 |
| 4 | 0.504 | 0.368 | 0.129 | 8.361 | 0.823 | 0.961 | 0.991 |
| 5 | 0.523 | 0.383 | 0.133 | 8.423 | 0.810 | 0.956 | 0.989 |
| 6 | 0.544 | 0.399 | 0.139 | 8.496 | 0.795 | 0.951 | 0.987 |
| 7 | 0.566 | 0.417 | 0.145 | 8.586 | 0.779 | 0.945 | 0.985 |
| 8 | 0.591 | 0.438 | 0.152 | 8.698 | 0.760 | 0.939 | 0.982 |
| 9 | 0.619 | 0.46 | 0.159 | 8.835 | 0.739 | 0.931 | 0.979 |
| 10 | 0.652 | 0.487 | 0.168 | 8.995 | 0.714 | 0.921 | 0.976 |
| 11 | 0.69 | 0.517 | 0.179 | 9.177 | 0.687 | 0.91 | 0.971 |
| 12 | 0.733 | 0.552 | 0.19 | 9.373 | 0.657 | 0.895 | 0.966 |
| 13 | 0.776 | 0.588 | 0.201 | 9.58 | 0.626 | 0.88 | 0.96 |
| 14 | 0.818 | 0.624 | 0.213 | 9.792 | 0.596 | 0.864 | 0.954 |
| 15 | 0.858 | 0.658 | 0.223 | 9.997 | 0.568 | 0.848 | 0.948 |
| 16 | 0.894 | 0.689 | 0.234 | 10.195 | 0.544 | 0.835 | 0.942 |
| 17 | 0.926 | 0.716 | 0.243 | 10.386 | 0.523 | 0.822 | 0.937 |
| 18 | 0.955 | 0.741 | 0.252 | 10.565 | 0.502 | 0.808 | 0.933 |
| 19 | 0.982 | 0.764 | 0.26 | 10.727 | 0.485 | 0.794 | 0.928 |

Результаты оценки модели LapDepth представлены в таблице 3. Значения были округлены.

Таблица 3 – Значения метрик модели LapDepth

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | RMSE | MAE | MRE | CEN |  |  |  |
| 0 | 0.436 | 0.305 | 0.113 | 7.019 | 0.875 | 0.976 | 0.993 |
| 1 | 0.599 | 0.435 | 0.149 | 7.973 | 0.767 | 0.94 | 0.983 |
| 2 | 0.641 | 0.469 | 0.159 | 8.21 | 0.736 | 0.929 | 0.979 |
| 3 | 0.712 | 0.529 | 0.177 | 8.555 | 0.681 | 0.906 | 0.969 |
| 4 | 0.814 | 0.616 | 0.202 | 9.009 | 0.602 | 0.868 | 0.954 |
| 5 | 0.941 | 0.728 | 0.234 | 9.538 | 0.508 | 0.807 | 0.929 |

Результаты оценки модели GLPN представлены в таблице 4. Значения были округлены.

Таблица 4 – Значения метрик модели GLPN

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | RMSE | MAE | MRE | CEN |  |  |  |
| 0 | 0.481 | 0.321 | 0.123 | 7.754 | 0.867 | 0.971 | 0.991 |
| 1 | 0.528 | 0.360 | 0.138 | 8.505 | 0.830 | 0.962 | 0.989 |
| 2 | 0.538 | 0.368 | 0.141 | 8.578 | 0.823 | 0.959 | 0.988 |
| 3 | 0.554 | 0.381 | 0.146 | 8.568 | 0.810 | 0.953 | 0.986 |
| 4 | 0.572 | 0.395 | 0.150 | 8.743 | 0.795 | 0.948 | 0.986 |
| 5 | 0.592 | 0.411 | 0.155 | 8.832 | 0.782 | 0.944 | 0.983 |
| 6 | 0.605 | 0.423 | 0.160 | 8.916 | 0.771 | 0.938 | 0.982 |
| 7 | 0.628 | 0.440 | 0.166 | 8.996 | 0.757 | 0.929 | 0.978 |
| 8 | 0.645 | 0.455 | 0.171 | 9.069 | 0.743 | 0.922 | 0.976 |
| 9 | 0.663 | 0.471 | 0.177 | 9.14 | 0.728 | 0.914 | 0.974 |
| 10 | 0.680 | 0.484 | 0.181 | 9.224 | 0.718 | 0.911 | 0.971 |
| 11 | 0.699 | 0.5 | 0.187 | 9.299 | 0.701 | 0.903 | 0.968 |
| 12 | 0.719 | 0.516 | 0.192 | 9.372 | 0.687 | 0.894 | 0.964 |
| 13 | 0.735 | 0.531 | 0.198 | 9.422 | 0.676 | 0.887 | 0.961 |
| 14 | 0.762 | 0.552 | 0.204 | 9.513 | 0.664 | 0.878 | 0.956 |
| 15 | 0.776 | 0.565 | 0.21 | 9.554 | 0.646 | 0.872 | 0.954 |
| 16 | 0.785 | 0.573 | 0.211 | 9.62 | 0.639 | 0.868 | 0.953 |
| 17 | 0.807 | 0.592 | 0.218 | 9.681 | 0.624 | 0.856 | 0.948 |
| 18 | 0.827 | 0.61 | 0.224 | 9.748 | 0.608 | 0.847 | 0.943 |
| 19 | 0.845 | 0.625 | 0.229 | 9.804 | 0.598 | 0.84 | 0.94 |
| 20 | 0.867 | 0.643 | 0.234 | 9.869 | 0.584 | 0.828 | 0.938 |
| 21 | 0.882 | 0.657 | 0.239 | 9.924 | 0.57 | 0.821 | 0.932 |
| 22 | 0.897 | 0.671 | 0.243 | 9.971 | 0.56 | 0.814 | 0.932 |
| 23 | 0.921 | 0.692 | 0.25 | 10.055 | 0.547 | 0.8 | 0.923 |
| 24 | 0.935 | 0.704 | 0.251 | 10.104 | 0.536 | 0.795 | 0.92 |
| 25 | 0.953 | 0.718 | 0.256 | 10.148 | 0.529 | 0.785 | 0.913 |
| 26 | 0.971 | 0.736 | 0.261 | 10.212 | 0.514 | 0.777 | 0.91 |
| 27 | 0.988 | 0.751 | 0.265 | 10.258 | 0.503 | 0.77 | 0.905 |

Результаты оценки модели представлены в таблице 4. Значения были округлены.

Таблица 5 – Значения метрик модели Depth Anything

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | RMSE | MAE | MRE | CEN |  |  |  |
| 0 | 0.280 | 0.187 | 0.069 | 5.854 | 0.964 | 0.992 | 0.998 |
| 1 | 0.288 | 0.192 | 0.069 | 5.909 | 0.957 | 0.991 | 0.997 |
| 2 | 0.291 | 0.194 | 0.070 | 5.929 | 0.956 | 0.991 | 0.997 |
| 3 | 0.296 | 0.198 | 0.071 | 5.954 | 0.955 | 0.990 | 0.997 |
| 4 | 0.300 | 0.202 | 0.073 | 5.978 | 0.953 | 0.990 | 0.997 |
| 5 | 0.306 | 0.206 | 0.074 | 6.004 | 0.950 | 0.990 | 0.997 |
| 6 | 0.311 | 0.211 | 0.076 | 6.032 | 0.947 | 0.990 | 0.997 |
| 7 | 0.318 | 0.216 | 0.078 | 6.063 | 0.944 | 0.990 | 0.997 |
| 8 | 0.326 | 0.222 | 0.080 | 6.099 | 0.940 | 0.989 | 0.997 |
| 9 | 0.335 | 0.229 | 0.083 | 6.139 | 0.935 | 0.989 | 0.997 |
| 10 | 0.345 | 0.237 | 0.086 | 6.182 | 0.930 | 0.988 | 0.997 |
| 11 | 0.356 | 0.246 | 0.089 | 6.23 | 0.924 | 0.986 | 0.997 |
| 12 | 0.368 | 0.255 | 0.093 | 6.282 | 0.917 | 0.985 | 0.996 |
| 13 | 0.382 | 0.266 | 0.097 | 6.339 | 0.906 | 0.983 | 0.996 |
| 14 | 0.396 | 0.278 | 0.102 | 6.4 | 0.896 | 0.981 | 0.996 |
| 15 | 0.413 | 0.292 | 0.107 | 6.468 | 0.883 | 0.978 | 0.995 |
| 16 | 0.432 | 0.309 | 0.114 | 6.541 | 0.868 | 0.974 | 0.994 |
| 17 | 0.453 | 0.327 | 0.121 | 6.625 | 0.85 | 0.97 | 0.993 |
| 18 | 0.475 | 0.346 | 0.128 | 6.711 | 0.833 | 0.966 | 0.992 |
| 19 | 0.5 | 0.367 | 0.136 | 6.803 | 0.814 | 0.961 | 0.991 |
| 20 | 0.526 | 0.389 | 0.145 | 6.899 | 0.793 | 0.954 | 0.99 |
| 21 | 0.554 | 0.413 | 0.154 | 7.002 | 0.77 | 0.948 | 0.988 |
| 22 | 0.583 | 0.439 | 0.164 | 7.108 | 0.744 | 0.939 | 0.985 |
| 23 | 0.614 | 0.465 | 0.173 | 7.214 | 0.718 | 0.928 | 0.982 |
| 24 | 0.646 | 0.494 | 0.184 | 7.326 | 0.691 | 0.917 | 0.979 |
| 25 | 0.68 | 0.524 | 0.194 | 7.447 | 0.663 | 0.903 | 0.977 |
| 26 | 0.716 | 0.554 | 0.205 | 7.575 | 0.635 | 0.888 | 0.972 |
| 27 | 0.753 | 0.586 | 0.216 | 7.705 | 0.604 | 0.874 | 0.966 |
| 28 | 0.792 | 0.62 | 0.227 | 7.833 | 0.571 | 0.859 | 0.96 |
| 29 | 0.831 | 0.654 | 0.237 | 7.959 | 0.542 | 0.84 | 0.953 |
| 30 | 0.871 | 0.688 | 0.247 | 8.084 | 0.514 | 0.82 | 0.945 |
| 31 | 0.913 | 0.724 | 0.257 | 8.206 | 0.486 | 0.8 | 0.936 |
| 32 | 0.956 | 0.761 | 0.267 | 8.327 | 0.456 | 0.779 | 0.927 |

### 6.2 Анализ результатов эксперимента

После получения значений метрик были построены графики. На рисунке 16 можно видеть зависимость среднеквадратичной ошибки от уровня шума для каждой из моделей. Как можно видеть, модель LapDepth с методологией пирамиды Лапласа оказывается слабоустойчивой к шумам. При минимальном повышении уровня шума (в рамках проводимого исследования) значение метрики для данного алгоритма возрастает до 0.599 и уже на 5-й итерации превышает 0.94.

Модели GLPN и PixelFormer демонстрируют более пологий тренд увеличения среднеквадратичной ошибки. Несмотря на ощутимую разницу в значениях RMSE на незашумленных данных, после 11-й итерации модель GLPN начинает демонстрировать лучшую устойчивость по среднеквадратической ошибке по сравнению с PixelFormer.

Самые устойчивыми по метрике среднеквадратичной ошибке оказываются модели ZoeDepth и DepthAnything. Значения метрики для LapDepth на 5-й итерации достигаются моделями на 31-й и 33-ей итерации соответственно. На всех итерациях DepthAnything демонстрирует более низкие значения ошибки, величиной около 0.2.

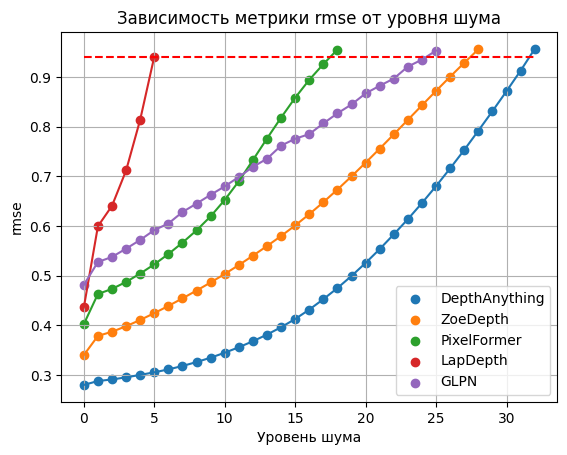


Рисунок 16 – Зависимость среднеквадратичной ошибки от уровня шума

Графики роста абсолютной ошибки (Рисунок 17) ожидаемо напоминают вышеизложенные тенденции роста метрики среднеквадратичной ошибки.

Модель LapDepth хуже всего справляется с увеличением шума, демонстрируя значение абсолютной ошибки более, чем 0.72 метра уже на пятой итерации. Модель GLPN после 11-й итерации демонстрирует уровень ошибки ниже, чем у PixelFormer. Модели DepthAnything и ZoeDepth имеют самый низкий уровень абсолютной разницы.

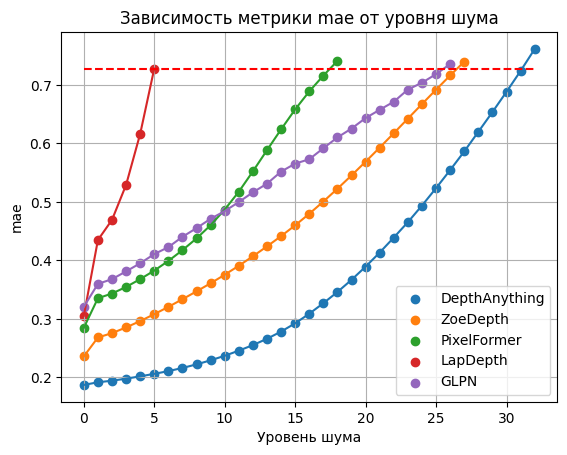


Рисунок 17 – Зависимость абсолютной ошибки от уровня шума

На рисунке 18 представлены графики изменения метрики дельта с пороговым значением 1.25. В отличие от предыдущих метрик, порог худшего значение (измеренная метрика для модели LapDepth на итерации 5) достигается моделью ZoeDepth раньше, чем моделью GLPN (26-я и 17-я итерация соответственно).

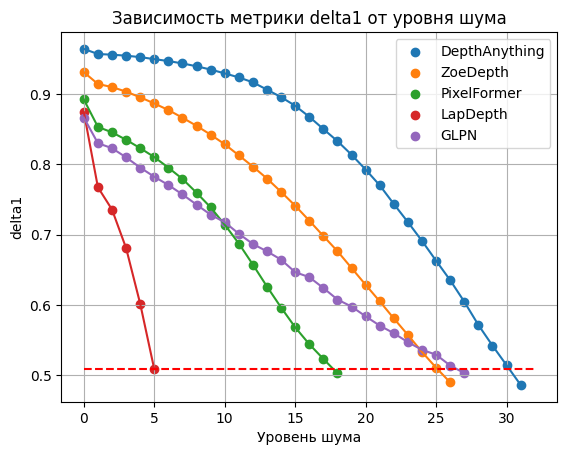


Рисунок 18 – Зависимость метрики дельта с пороговым значением 1.25 от уровня шума

Тем не менее, при измерении метрики со значением модель ZoeDepth вновь демонстрирует ощутимое превосходство на всех уровнях шума со средним отрывом порядка 0.075 (Рисунок 19).

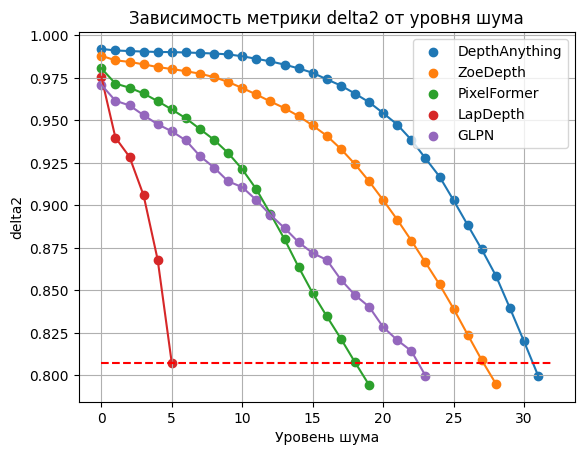


Рисунок 19 – Зависимость метрики дельта со значением

На рисунке 20 приведены значения метрики Census (для моделей ZoeDepth и DepthAnything приводятся значения для 30 уровней шума). Как можно видеть, изменения пространственной метрики также повторяют вышеизложенные тенденции.

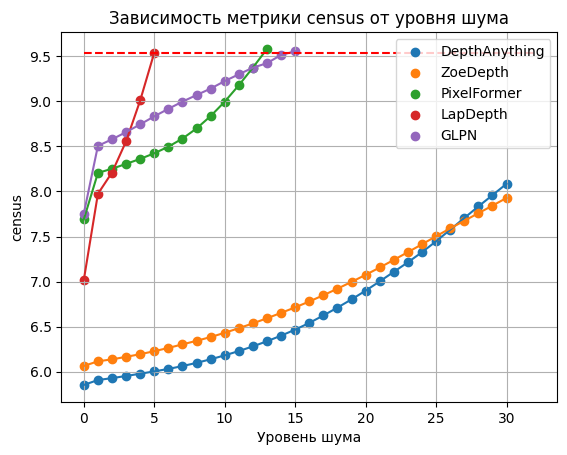


Рисунок 20 – Изменение метрики census

## 7 Безопасность жизнедеятельности

### 7.1 Основные положения

В ходе выполнения данной выпускной квалификационной работы было проведено взаимодействие с множеством реализаций алгоритмов. В качестве среды разработки была выбрана платформа Kaggle, являющаяся системой организации конкурсов по исследованию данных, а также социальной сетью специалистов по обработке данных и машинному обучению.

Работа среды осуществляется на публичной веб-платформе с возможностью публикации наборов данных, исследованием и созданием моделей, взаимодействия с другими специалистами по данным и инженерами по машинному обучению. Также возможна организация конкурсов по исследованию данных и участие в них.

Важной частью такой многофункциональной и популярной платформы является ее эргономичность. Так как платформа многофункциональна, пользователи работают с ней для решения широкого спектра задач в течении внушительного времени. Удобный и безопасный с точки зрения установленных норм интерфейс может помочь пользователям длительное время использовать технологические возможности платформы с высокой эффективностью без вреда для здоровья, в то время как неудобный интерфейс, не соответствующий нормам безопасности, ограничивает способность пользователя действовать эффективно и может повлечь нанесение вреда его здоровью.

В данной работе использовался интерфейс, названный “блокнотом”. Подобная реализация имеется на платформе Kaggle. Данный интерфейс позволяет разделять код на ячейки, с возможностью запуска каждой ячейки отдельно. При этом область имен для всех ячеек общая в рамках одного блокнота. Кроме того, в своих блокнотах Kaggle позволяет подгружать и использовать наборы данных, веса моделей, графические ускорители. В таблице 6 приведены характеристики данного инструмента.

Таблица 6 - характеристики используемого ПО

|  |  |
| --- | --- |
| Программное обеспечение | Kaggle notebook |
| Вид | Класс |
| Функция ПО | Разработка программ |
| Режим эксплуатации | Интерактивная обработка (пользователь ПО вводит текст или команды, которые обрабатываются ПО) |
| Прикладная область информационной системы | Написание программ для аналитики данных |
| Масштаб ПО | Средний |
| Представление данных | Объектное (для командной операции) и форматированный файл (для командной операции, текста ввода-вывода и хранения данных) |
| Критичность ПО | Удобно для индивидуального использования |
| Класс пользователя | Начинающий (для начала работа с ПО достаточно базовых навыков программирования на языке Python) |
| Требуемые рабочие характеристики | Емкость – высокая (все данные хранятся удаленно);  Время отклика – быстрое;  Производительность – высокая (имеется возможность подключения графических процессоров) |
| Стабильность ПО | С контролируемым изменением (обычная модернизация и новые выпуски) |
| Требование защиты | Защита от несанкционированного доступа – средняя;  Контрольный след – средняя;  Защита программ и данных – сильная |
| Требование надежности | Завершенность: высокая;  отказоустойчивость: высокая;  восстанавливаемость: высокая |
| Вычислительная система и среда | Любая система с возможностью использования браузеров, корректно взаимодействующих с платформой Kaggle |
| Требования к вычислительным ресурсам | Требования к вычислительным ресурсам определяются браузером, посредством которого осуществляется доступ к ПО, т. к. все вычисления проводятся в облаке |
| Готовность программного продукта | Готовый некоммерческий |
| Использование программных данных | Для множества пользователей |
| Исходный язык | Python 3.10.13 |

### 7.2 Анализ соответствия среды разработки установленным нормам.

В ГОСТ Р ИСО 9241-110-2016 [25] выделяются следующие основные принципы организации диалога человек-система:

1. Приемлемость организации диалога для выполнения производственного задания.
2. Информативность.
3. Соответствие ожиданиям пользователей.
4. Пригодность для обучения.
5. Контролируемость.
6. Устойчивость к ошибкам.
7. Адаптируемость к индивидуальным особенностям пользователя.

В используемом в ходе данного исследования блокноте Kaggle выполнены все вышеперечисленные требования:

1. Блокнот Kaggle соответствует требованию приемлемости организации диалога для выполнения производственного задания. В среде существует возможность сохранения разных состояний блокнота, создании ссылок на GitHub, удобной публикации блокнота для других пользователей.
2. Блокнот Kaggle также соответствует требованию информативности. Каждая кнопка, ссылка имеет описание на английском языке, точно определяющее последствия взаимодействия с тем или иным средством управления (Рисунок 21).

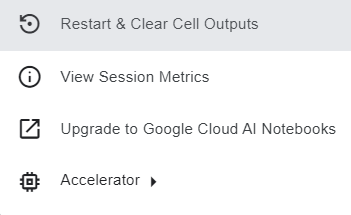


Рисунок 21 – Примеры подписей к кнопкам

1. Блокнот Kaggle соответствует принципам ожиданий пользователей. Данный интерфейс имеет много сходств с популярным интерфейсом блокнота Jupyter и в точности реализует его основные функции, такие как запуск кода ячейки и возможность использования специальных команд блокнота в ячейках с помощью синтаксиса, привычного пользователям блокнота Jupyter (Рисунок 22).



Рисунок 22 – Пример ячейки блокнота Kaggle со специальной командой

1. Блокнот Kaggle соответствует требованиям пригодности к обучению. Kaggle предоставляет достаточно подробную документацию по использованию своих блокнотов. Также на платформе имеется множество блокнотов других пользователей, что позволяет увидеть примеры использования различных возможностей, предоставляемых интерфейсом (Рисунок 23)

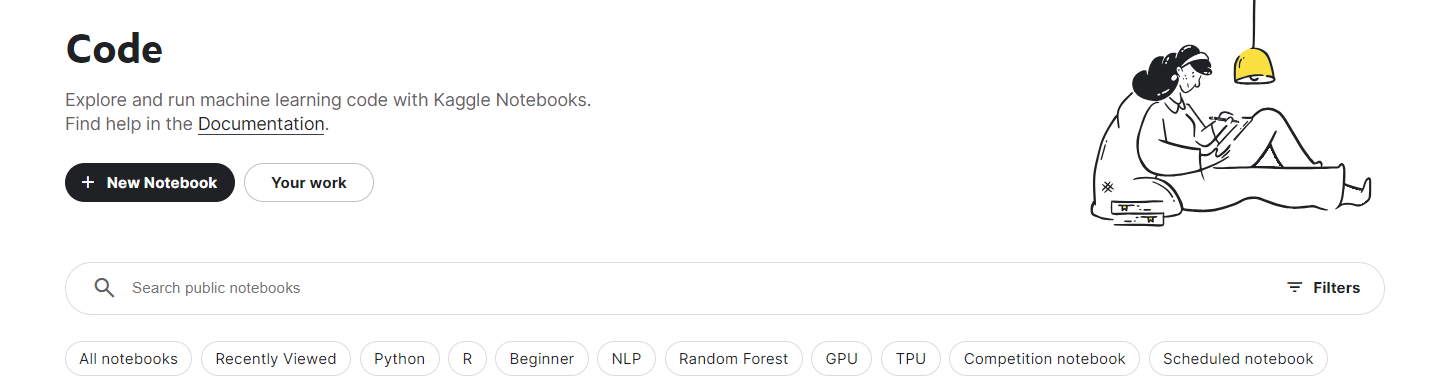


Рисунок 23 – Возможности поиска примеров блокнотов, созданных пользователями

1. Блокнот Kaggle соответствует требованиям контролируемости, т. к. позволяет отменять и повторять выполненные действия, использовать для выполнения команд как мышь, так и горячие клавиши.
2. Блокнот Kaggle соответствует принципу устойчивости к ошибкам. В случае возникновения ситуаций, в ходе которых была нарушена привычная работа приложения (к примеру, невозможность сохранения новой версии блокнота, поскольку он был отредактирован в нескольких сессиях одновременно), пользователь получает предупреждения (Рисунок 24).

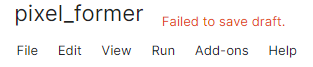


Рисунок 24 – Предупреждение об ошибке

1. Блокнот Kaggle соответствует принципу адаптируемости к индивидуальным особенностям пользователя, поскольку имеет настройки, позволяющие пользователя выбрать цветовую тему для элементов интерфейса, размеры шрифта.

### 7.3 Выводы

В ходе разработки был использован инструмент Kaggle notebook. Характеристики данного ПО были проанализированы в соответствии с ГОСТ Р ИСО/МЭК ТО 12182-2002 “Информационная технология” (ИТ). Классификация программных средств” [26]. Данное ПО отличается низким порогом вхождения для пользователя и низкими требованиями к вычислительным ресурсам – для использования ПО достаточно работающего браузера, который будет передавать информацию между клиентом и сервером, обрабатывающим вычисления.

ПО было проанализировано на соответствие основным принципам, указанным в ГОСТ Р ИСО 9241-110-2016. В ходе анализа было выяснено, что используемое в ходе разработки ПО соответствует принципам приемлемости организации диалога для выполнения производственного задания, информативности, соответствия ожиданиям пользователей, пригодности для обучения, контролируемости, устойчивости к ошибкам, адаптируемости к индивидуальным особенностям пользователя.

Таким образом, Kaggle notebook является эргономичным инструментом, подходящим для эффективной разработки программ в области аналитики данных.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной лабораторной работы были изучены существующие алгоритмы монокулярной оценки глубины. Среди них были отобраны 5 моделей. Все из них были созданы с помощью технологий глубокого обучения – способ организации модели в композицию вычислений (слоев) с последующей оптимизацией параметров методом градиентного спуска на выбранном наборе данных.

После этого был выбран набор данных, содержащий протяженные объекты – NYU Depth v2 и подключен на платформе Kaggle. В ходе работы была использована тестовая выборка, содержащая 654 пары изображений и соответствующих карт глубины.

В ходе работы были отобраны метрики. Для бенчмарка были использованы метрики абсолютной ошибки, среднеквадратической ошибки, относительной ошибки, метрика census, отражающая пространственную ошибку, а также дельта-метрики с тремя уровнями порога.

Затем была выбрана модель шума и применена к изображениям с протяженными объектами с разным уровнем зашумления.

Наконец, были измерены метрики моделей на изображениях с протяженными объектами при выбранных уровнях шума. Было выяснено, что модель LapDepth наиболее неустойчива к способности преодолевать дефекты исходных изображений протяженных объектов среди рассмотренных алгоритмов. PixelFormer и GLPN по итогам проведенного исследования зарекомендовали себя как решения, более устойчивые к шумам, при этом PixelFormer более устойчив при низких уровнях шума, в то время как GLPN – при более высоких.

Самую высокую устойчивость продемонстрировали модели DepthAnything и ZoeDepth, веса которых формировались в ходе обучения на большом количестве наборов данных с помощью предсказания относительной глубины и последующей настройки блока предсказания метрической глубины.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. NYU Depth Dataset V2 [Электронный ресурс]. URL: https://cs.nyu.edu/~fergus/datasets/nyu\_depth\_v2.html (дата обращения: 03.05.24).
2. Kitti Eigen split benchmark [Электронный ресурс]. URL: https://paperswithcode.com/sota/monocular-depth-estimation-on-kitti-eigen (дата обращения: 03.05.24)
3. Rene Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 12179–12188, October 2021
4. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In ICLR, 2021.
5. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016
6. Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D. Reid. RefineNet: Multi-path refinement networks for highresolution semantic segmentation. In CVPR, 2017.
7. Shariq Farooq et. al., ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth. arXiv preprint arXiv: 2302.12288, 2023.
8. Hangbo Bao, Li Dong, and Furu Wei. Beit: BERT pretraining of image transformers. CoRR, abs/2106.08254, 2021
9. Rene Ranftl, Katrin Lasinger, David Hafner, Konrad ´ Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2020
10. Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Localbins: Improving depth estimation by learning local distributions. In European Conference on Computer Vision, pages 480–496. Springer, 2022.
11. M. Song et. al. Monocular Depth Estimation Using Laplacian Pyramid-Based Depth Residuals. In IEEE Transactions on Circuits and Systems for Video Technology, pages 4381-4393. November, 2021.
12. S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He., “Aggregated residual transformations for deep neural networks,” in Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit., Jul. 2017, pp. 5987–5995.
13. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 4, pp. 834–848, Apr. 2018.
14. Ashutosh Agarwal, Chetan Arora ZoeDepth: Attention Attention Everywhere: Monocular Depth Prediction with Skip Attention. arXiv preprint arXiv: 2210.09071, 2022.
15. Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 10012–10022, October 2021
16. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, Computer Vision – ECCV 2014, pages 346–361, Cham, 2014. Springer International Publishing
17. Doyeon Kim et. al. ZoeDepth: Global-Local Path Networks for Monocular Depth Estimation with Vertical CutDepth arXiv preprint arXiv: 2201.07436, 2022.
18. Yasunori Ishii, Takayoshi Yamashita CutDepth: Edge-aware Data Augmentation in Depth Estimation arXiv preprint arXiv: 2107.07684, 2021.
19. Lihe Yang et. al. Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data arXiv preprint arXiv: 2401.10891, 2024
20. Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In ICCV, 2019
21. Maxime Oquab, Timothee Darcet, Th ´ eo Moutakanni, Huy Vo, ´ Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. TMLR, 2023
22. Krzysztof Wegner, Olgierd Stankiewicz Similarity measures for depth estimation IEEE 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2009
23. Kaggle NYU Depth Dataset V2 [Электронный ресурс]. URL: https://www.kaggle.com/datasets/awsaf49/nyuv2-official-split-dataset (дата обращения: 03.05.24).
24. Modeling noise for image simulations [Электронный ресурс]. URL: https://kmdouglass.github.io/posts/modeling-noise-for-image-simulations/ (дата обращения: 03.05.24).
25. ГОСТ Р ИСО 9241-151-2016 Эргономика взаимодействия человек-система. Часть 161. Элементы графического пользовательского интерфейса. М.: Стандартинформ, 2016.
26. ГОСТ Р ИСО/МЭК ТО 12182-2002 “Информационная технология” (ИТ). Классификация программных средств”