# Реферат

Отчёт по учебной технологической практике содержит: 29 с., 21 рис., 6 источников

LINUX. UBUNTU. ДРАЙВЕРА ВИДЕОКАРТЫ. CUDA. NVIDIA CUDNN. NVIDIA CONTAINER TOOLKIT. DOCKER. ANACONDA. JUPYTER.

Целью учебной технологической практики является установка дистрибутива ubuntu, установка нужных драйверов для правильной работы видеокарты, anaconda, а также docker в дальнейшем для работы каждого студента со своим контейнером.

В результате выполнения, были установлены нужные драйвера, протестирована видеокарта, настроен контейнер Jupyter.

# Содержание

[Реферат 6](#_Toc171963196)

[Содержание 7](#_Toc171963197)

[Введение 8](#_Toc171963198)

[Настройка Ubuntu 22.04 LTS 9](#_Toc171963199)

[Установка драйвера видеокарты 10](#_Toc171963200)

[Установка Nvidia CUDA 11](#_Toc171963201)

[Установка Nvidia CUDnn 15](#_Toc171963202)

[Установка Nvidia Container Toolkit и Docker 18](#_Toc171963203)

[Установка Anaconda 21](#_Toc171963204)

[Заключение 23](#_Toc171963205)

[Список использованных источников 24](#_Toc171963206)

# Введение

Целью учебной технологической практики является установка дистрибутива ubuntu, установка нужных драйверов для правильной работы видеокарты, anaconda, а также docker в дальнейшем для работы каждого студента со своим контейнером.

Основные задачи учебной технологической практики:

1. Настройка Ubuntu 22.04 LTS;

2. Установка драйвера видеокарты;

3. Установка Nvidia CUDA;

4. Установка Nvidia CUDnn;

5. Установка Nvidia Container Toolkit и Docker;

6. Установка Anaconda

# Настройка Ubuntu 22.04 LTS

Для установки необходим установочный образ Ubuntu 22.04 LTS с официального сайта Ubuntu. Далее создаем загрузочный USB-накопитель. Подключаем загрузочный USB-накопитель к компьютеру, который хотим установить. Перезагружаем компьютер и видим, что загрузка идет с USB-накопителя. После загрузки с USB-накопителя видим экран установки Ubuntu. Выбираем язык установки и нажимаем "Установить Ubuntu". Следуем инструкциям на экране. После завершения установки извлекаем USB-накопитель и перезагружаем компьютер.

На рисунке 1 показано название дистрибутива, номер версии и кодовое имя установленной Ubuntu. Таким образом, убеждаемся, что все установлено корректно.

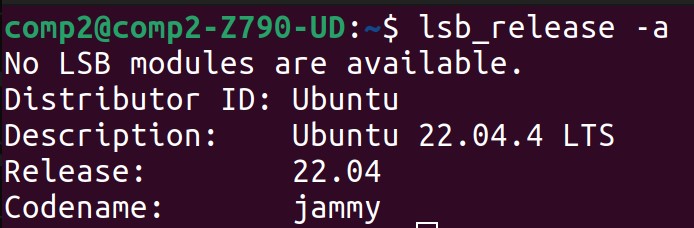


Рисунок 1 – Информация об установленной Ubuntu

# Установка драйвера видеокарты

Для начала, нужно проверить список рекомендуемых драйверов командой ubuntu-drivers devices. В ней будет предложен список доступных драйверов на устройства, в том числе и на видеокарту (рисунок 2). Выбираем нужную версию и устанавливаем.

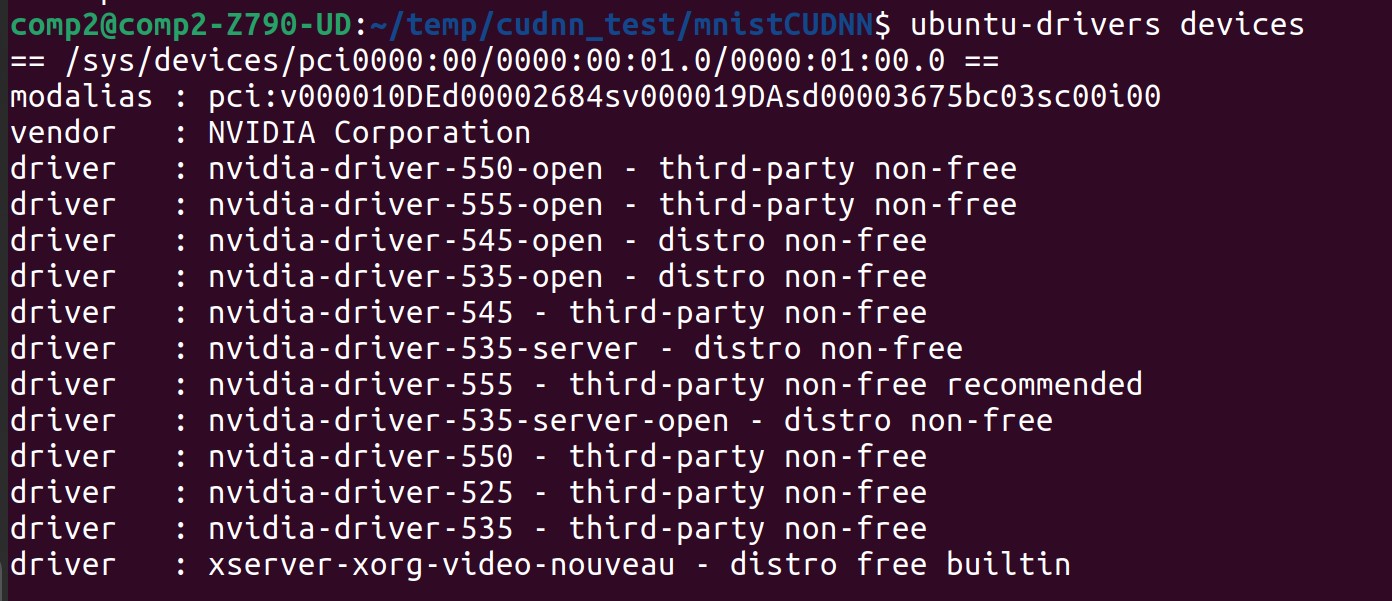


Рисунок 2 – Список версий драйверов

После установки перезагружаем компьютер и проверяем установленную версию командой nvidia-smi. Так как все установлено корректно, выбранная версия драйвера будет отражена в заголовке вывода в поле Driver version (рисунок 3).

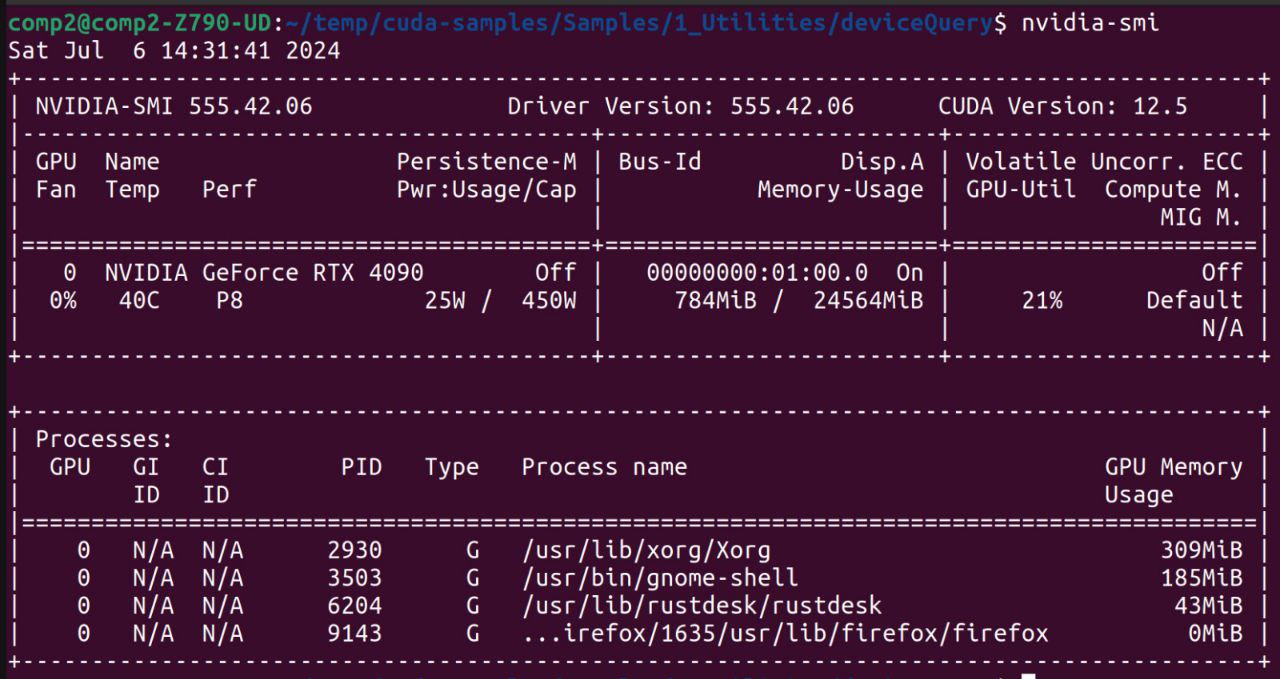


Рисунок 3 – Вывод версий драйвера и Cuda

# Установка Nvidia CUDA

Nvidia CUDA (Compute Unified Device Architecture) – это платформа и модель программирования, разработанная компанией Nvidia, которая позволяет использовать графические процессоры (GPU) для выполнения вычислений общего назначения. Изначально предназначенная для рендеринга графики, CUDA теперь значительно расширила возможности GPU, позволяя им выполнять сложные математические расчеты параллельно. Это значительно ускоряет вычислительные процессы, что делает CUDA незаменимой для задач, требующих больших вычислительных ресурсов.

CUDA используется для различных применений, включая машинное обучение, научные исследования, обработку изображений и видео, а также рендеринг графики. В машинном обучении CUDA помогает ускорять обучение и инференс нейронных сетей, что позволяет обрабатывать большие объемы данных быстрее и эффективнее. В научных исследованиях она способствует более быстрому выполнению сложных вычислений, необходимых для моделирования физических процессов и анализа больших данных. Обработка изображений и видео также выигрывает от CUDA, благодаря возможности быстрого выполнения преобразований и фильтрации.

Для работы с pytorch используется CUDA версии 12.1 из-за особенностей зависимости torch от версий CUDA и CUDNN и их зависимостями между собой. Установочный пакет идет с этим руководством, также идет официальный гайд от Nvidia по установке пакета. Все команды описаны в нем. С официального сайта pytorch можно проверить совместимость (рисунок 4).[5]

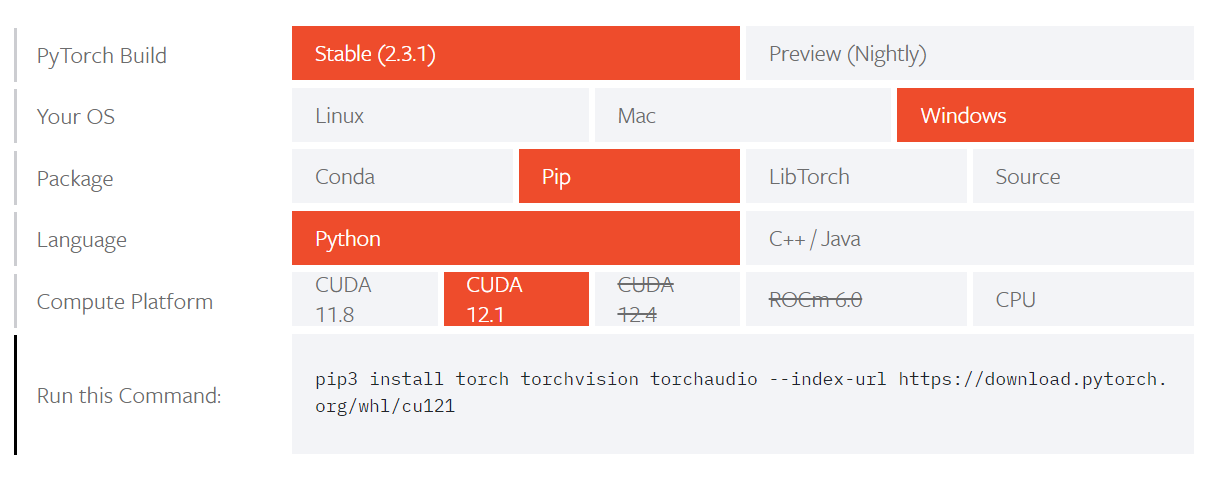


Рисунок 4 – Совместимость версий

С официального сайт Pytorch взяты тесты для проверки корректной работы видеокарты. С помощью псевдографического монитора nvtop проверяем работу видеокарты во время выполнения тестов. На рисунке 5 показан код теста для cuda. На рисунке 6 показаны результаты файла. Программа работает корректно, с помощью псевдографического монитора nvtop смотрим работу видеокарты.

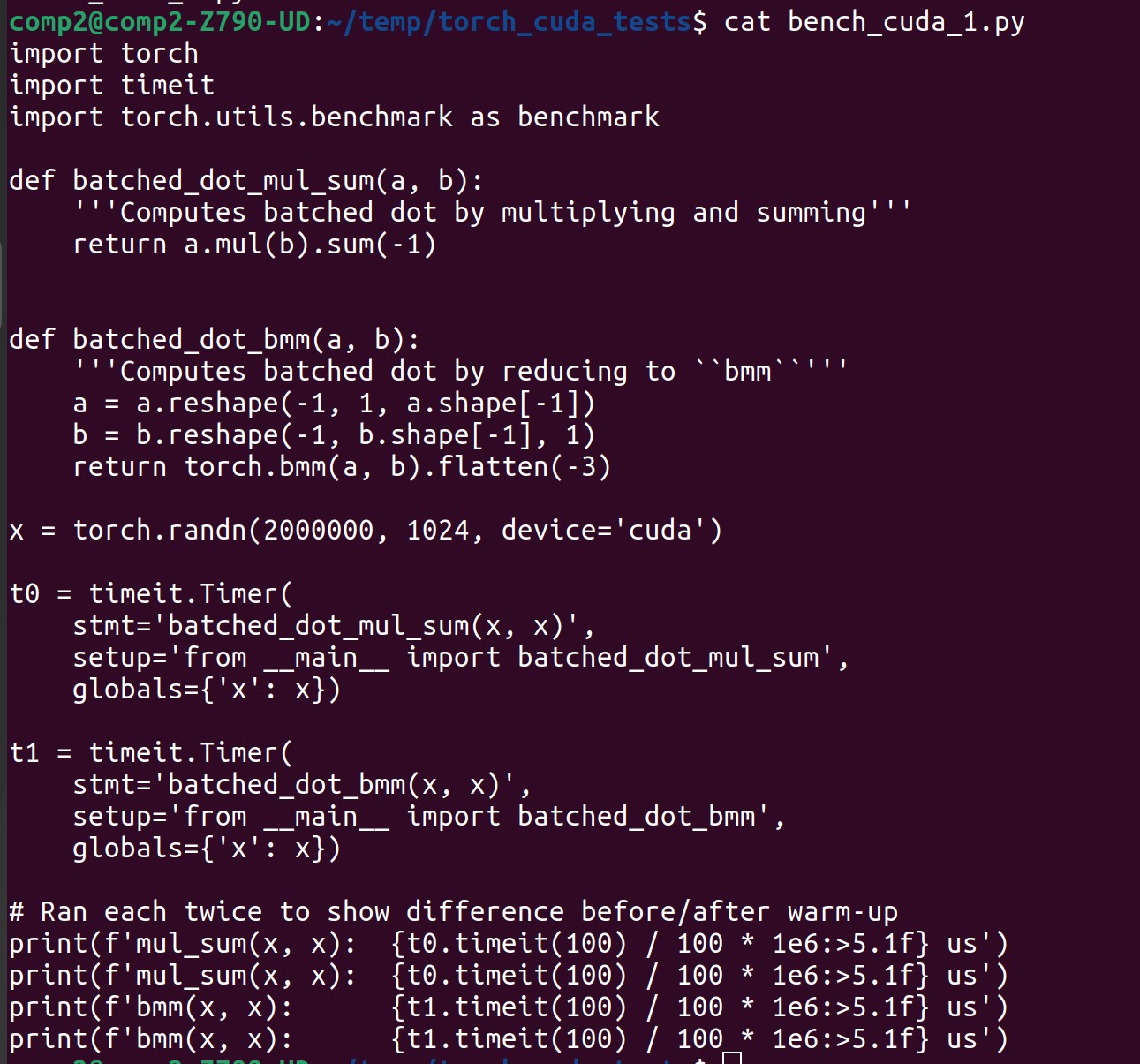


Рисунок 5 – Файл для проверки работы видеокарты

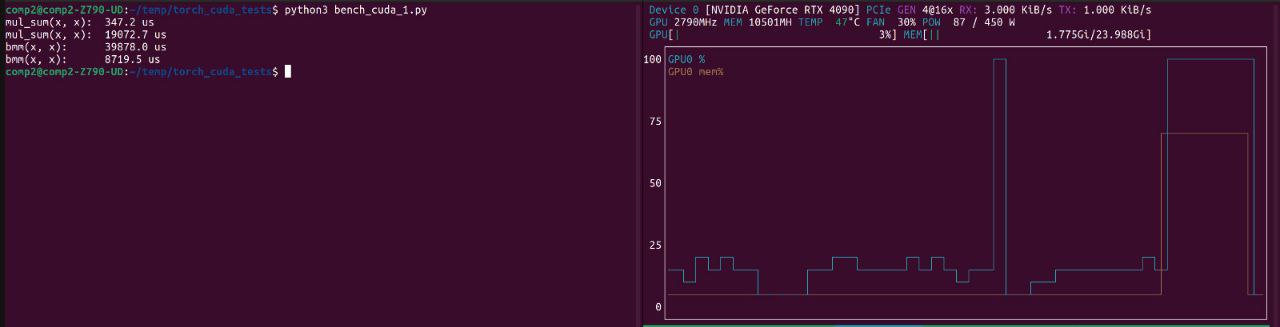


Рисунок 6 – Вывод тестовой программы

Тест DeviceQuery в CUDA samples представляет собой базовый утилитарный инструмент для анализа характеристик и возможностей графического процессора (GPU) на нашей системе. Данный тест помогает убедиться, что используемые функции и возможности CUDA соответствуют аппаратным возможностям установленного устройства.[1] Результаты тестов изображены на рисунке 7.

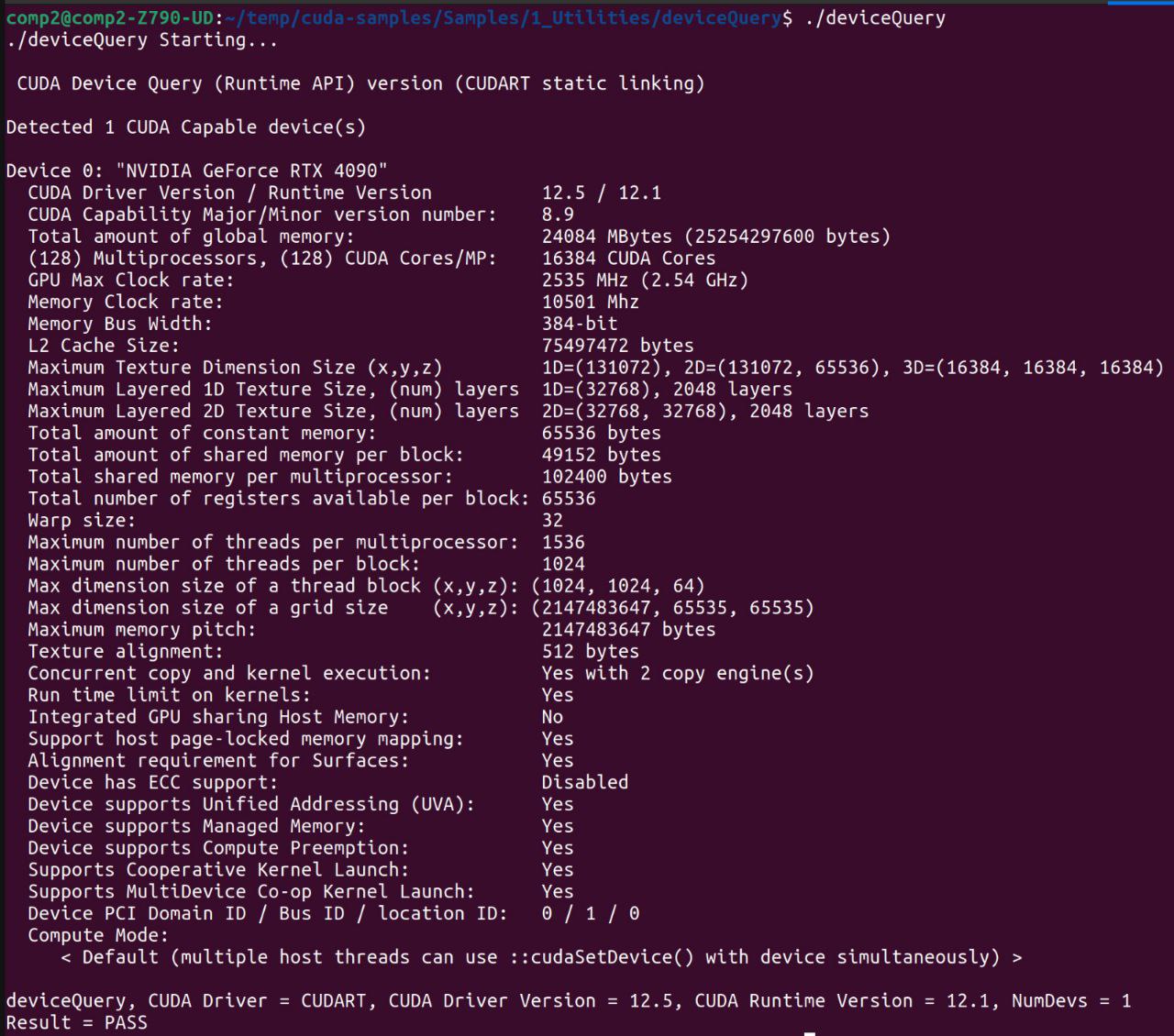


Рисунок 7 – Результаты тестов DeviceQuery

Основная задача bandwidthTest заключается в измерении скорости передачи данных между глобальной памятью GPU и другими участками памяти, такими как хостовая память (RAM компьютера). Результаты данного теста изображены на рисунке 8.[1]

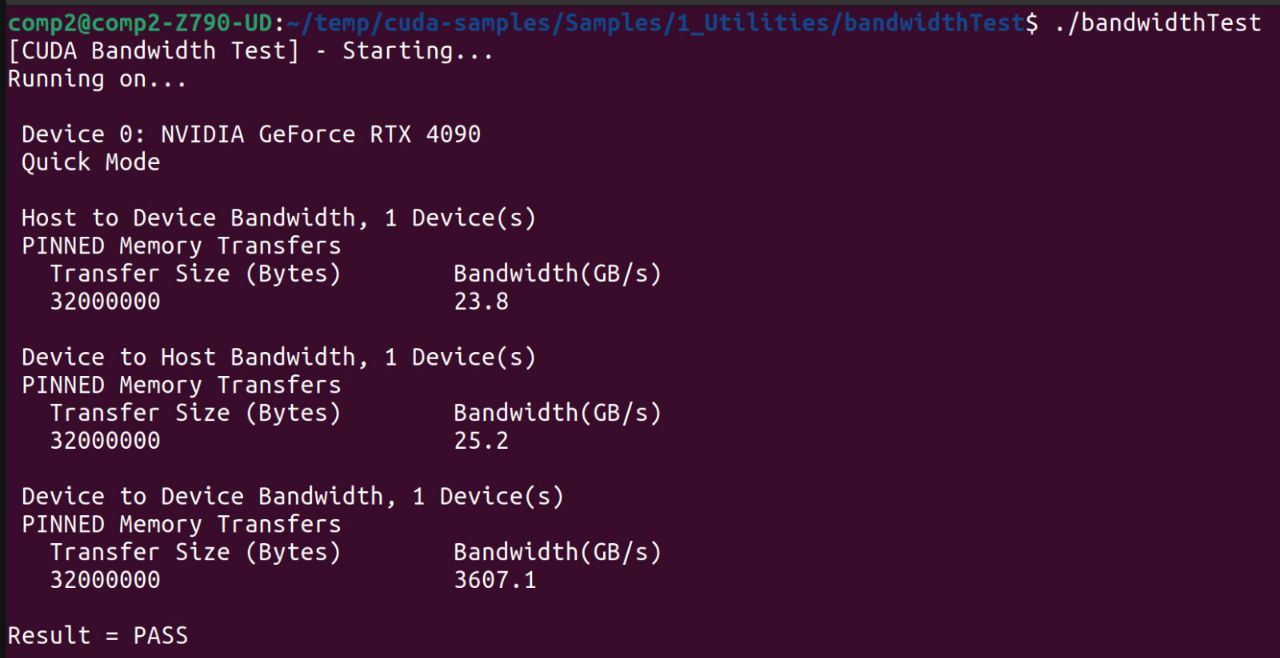


Рисунок 8 – Результаты теста bandwidthTest

# Установка Nvidia CUDnn

Nvidia cuDNN (CUDA Deep Neural Network library) – это высокопроизводительная библиотека для работы с глубокими нейронными сетями, разработанная компанией Nvidia. cuDNN предоставляет оптимизированные примитивы для глубокого обучения, такие как свертки, нормализация, активации и другие операции, часто используемые в нейронных сетях. Эти примитивы специально разработаны для эффективного использования вычислительной мощности GPU, что позволяет значительно ускорить обучение и инференс моделей глубокого обучения.

cuDNN используется в различных областях, где требуется быстрое и эффективное обучение нейронных сетей, таких как компьютерное зрение, обработка естественного языка и распознавание речи. Благодаря своей высокой производительности, cuDNN помогает исследователям и разработчикам создавать и обучать более сложные и точные модели, а также позволяет интегрировать глубокое обучение в приложения реального времени.

Также как и CUDA, CUDNN необходимо строго версии 9.2, совместимой с CUDA 12.1. В файлах отобран этот deb-пакет. Процесс установки описан в официальном руководстве.[2] На рисунке 9 показан фрагмент с официальной документации по тестированию CUDnn. Все тесты для видеокарты находятся в директории temp.

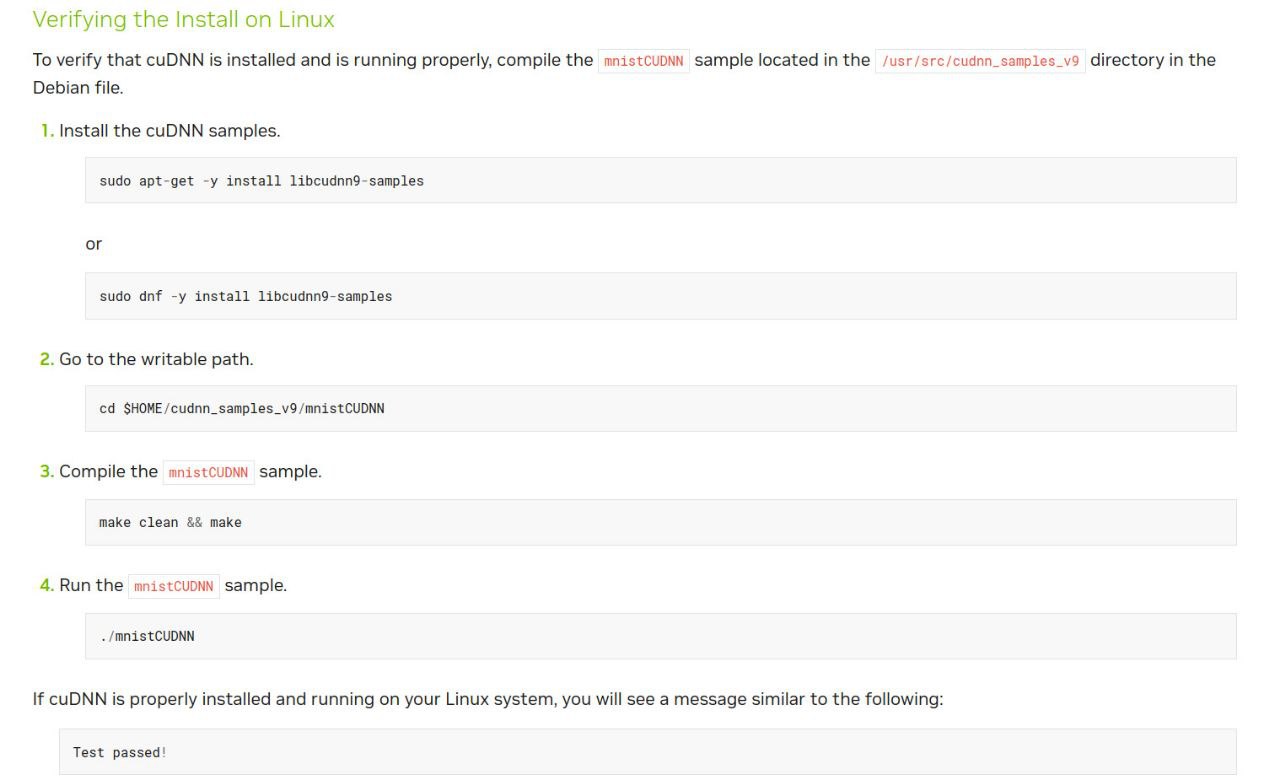


Рисунок 9 – Команды для тестирования CUDnn

Тест mnistCUDNN относится к бенчмарку или тесту производительности, который использует библиотеку CUDA Deep Neural Network (cuDNN) для обучения и тестирования модели нейронной сети на датасете MNIST. [2]

На рисунках 10 и 11 показаны результаты тестов mnistCUDNN.

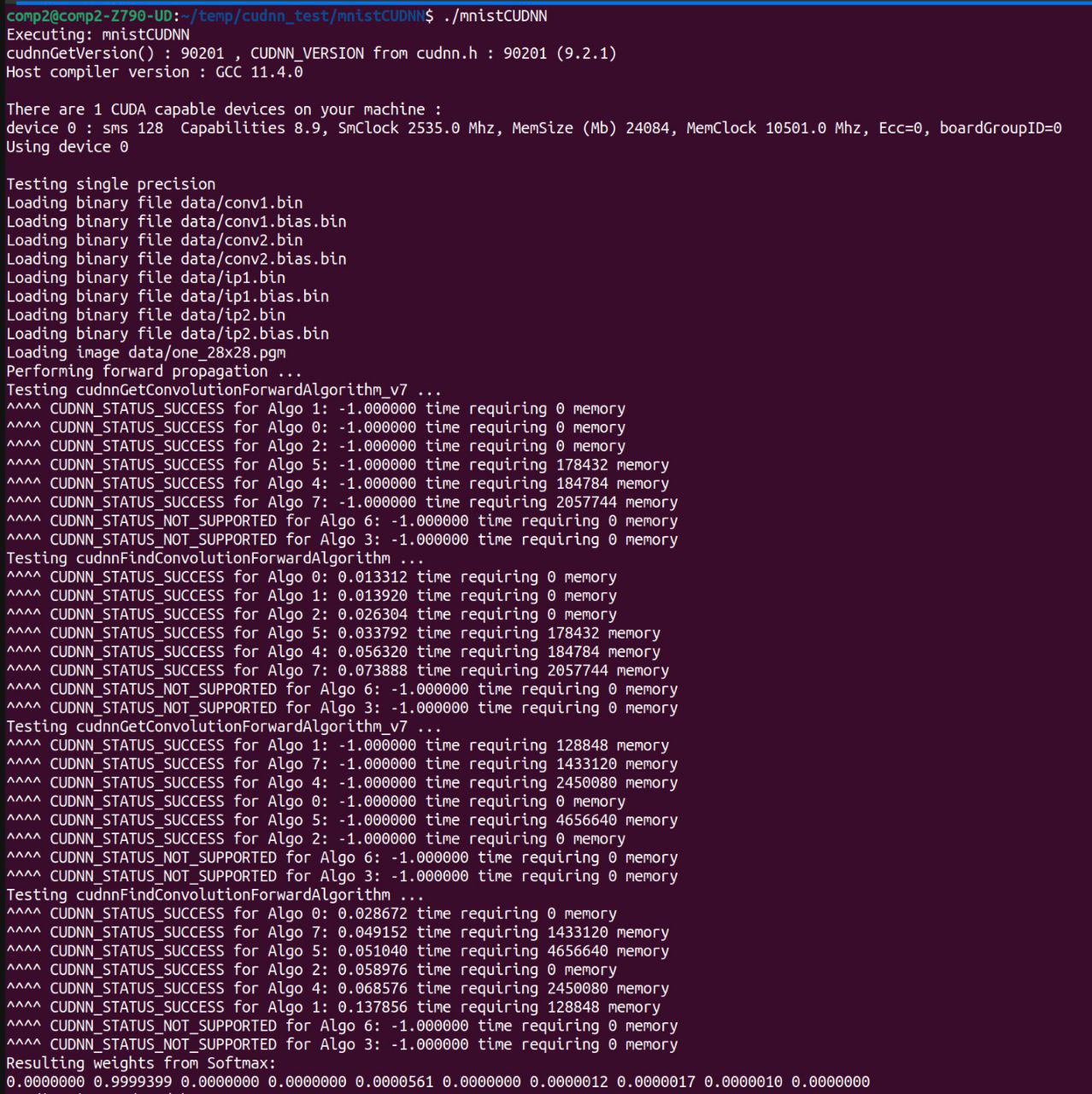


Рисунок 10 – Результаты тестов mnistCUDNN

mnistCUDNN – это измерение производительности и скорости обучения нейронной сети на GPU с использованием cuDNN. Этот тест позволяет оценить эффективность алгоритмов обучения и инференса на конкретной аппаратной конфигурации с поддержкой CUDA.

Тест mnistCUDNN является важным инструментом для оценки производительности глубокого обучения на GPU и помогает оптимизировать разработку и развертывание нейронных сетей, учитывая аппаратные особенности и возможности, предоставляемые CUDA и cuDNN.

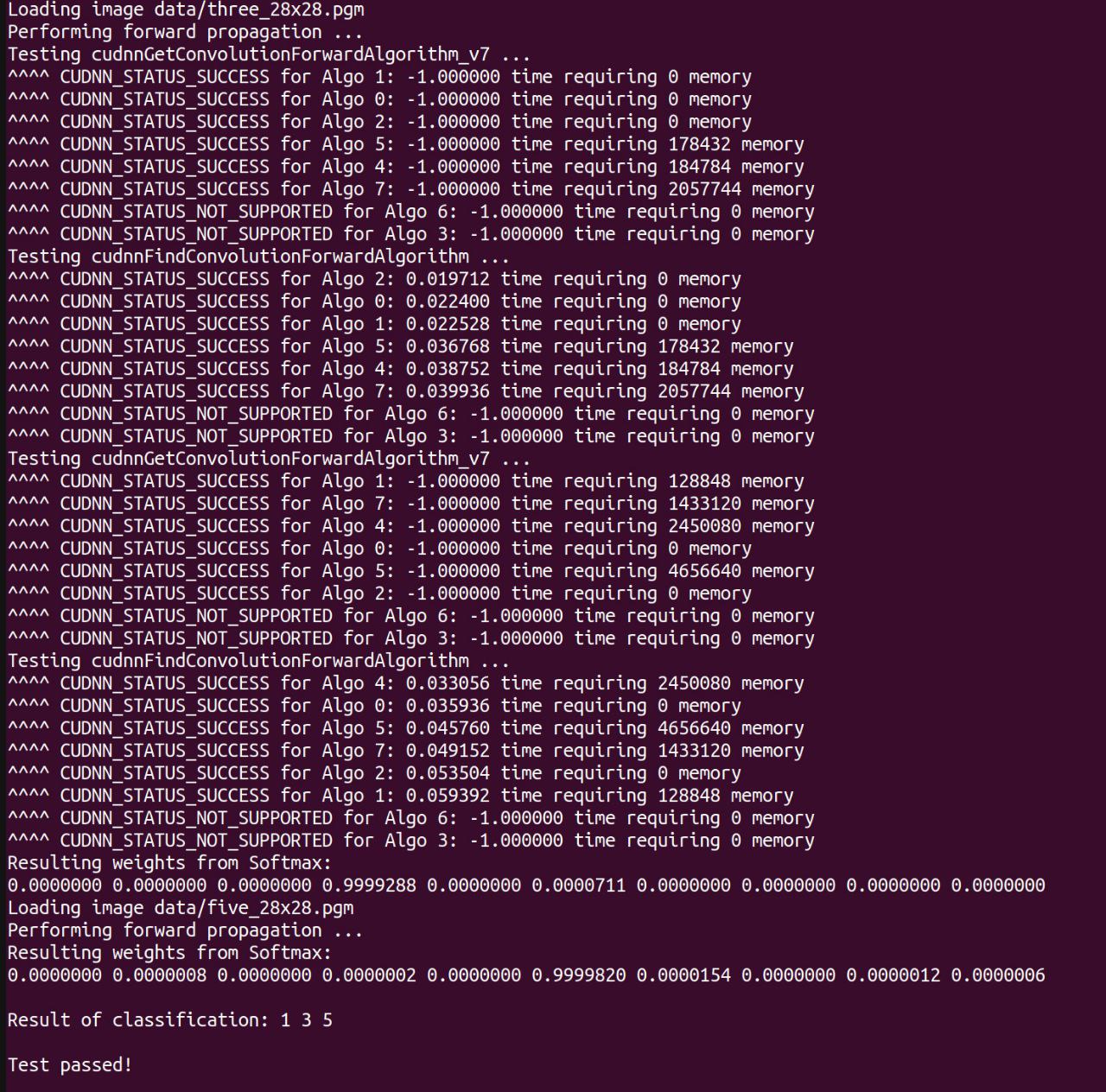


Рисунок 11 – Результаты тестов mnistCUDNN

# Установка Nvidia Container Toolkit и Docker

Docker – это открытая платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры представляют собой стандартизированный способ упаковки приложений и всех их зависимостей, включая библиотеки и другие ресурсы, в единое целое. Это обеспечивает изоляцию и портабельность, позволяя приложениям работать на любой совместимой с Docker системе, независимо от окружения разработки или развертывания.

Основные концепции Docker включают образы и контейнеры. Образы представляют собой шаблоны для создания контейнеров, включающие все необходимые компоненты приложения и его настроек. Контейнеры — это запущенные экземпляры этих образов, которые работают в изолированной среде, где они могут выполнять свои функции без вмешательства в другие контейнеры или хост-систему.

Docker предоставляет простой и эффективный способ управления приложениями, улучшая разработку и развертывание благодаря своей легковесной виртуализации и гибкости настройки окружений. Это делает Docker популярным инструментом для современной разработки программного обеспечения, особенно в контексте микросервисной архитектуры и непрерывной поставки приложений.

Устанавливаем Docker с помощью официальной документации.[3]

На рисунке 12 показана версия установленного Docker.

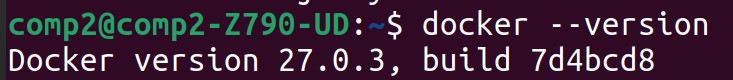


Рисунок 12 – Версия Docker

Устанавливаем образы для проверки корректной работы. Список образов изображен на рисунке 13.

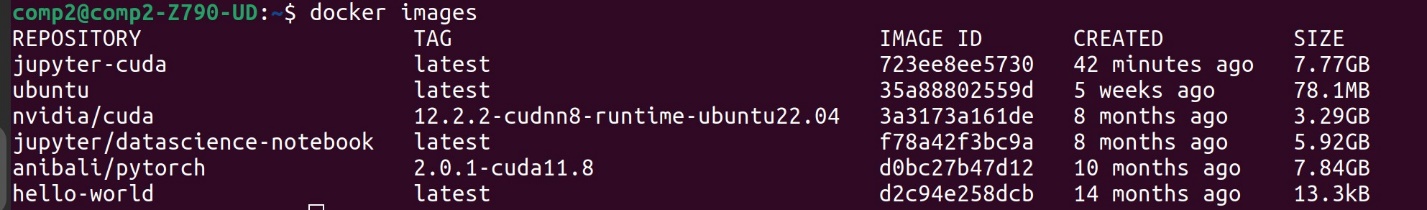


Рисунок 13 – Список образов

Для корректной работы Jupyter. Пишем свой образ. Dockerfile изображен на рисунке 14.[3]



Рисунок 14 – Dockerfile

Jupyter – это интерактивная вычислительная среда, которая позволяет пользователям создавать и обмениваться документами, содержащими живой код, уравнения, визуализации и текстовые пояснения. Она поддерживает более 40 языков программирования, включая Python, R и Julia, что делает её универсальным инструментом для научных вычислений, анализа данных и машинного обучения. С Jupyter пользователи могут выполнять код в реальном времени, видеть результаты мгновенно и интерактивно исследовать данные.

Основное предназначение Jupyter заключается в упрощении процесса исследования данных и создания воспроизводимых исследовательских отчетов. Она широко используется в образовании, науке и бизнесе для разработки и презентации данных и моделей. Благодаря возможности объединять код, визуализации и текст в одном документе, Jupyter упрощает совместную работу и обмен знаниями между специалистами и командами, что способствует более глубокому пониманию и эффективному решению задач.

Собираем данный образ, монтируем его к выбранной папке на компьютере для дальнейшего сохранения файлов и запускаем свой контейнер в интерактивном режиме. На рисунке 15 изображен запуск контейнера собранного образа.

Переходим по адресу в терминале на локальный адрес Jupyter.[4] Создаем файл example. Проверяем корректность работы cuda и cudnn для дальнейшего корректного обучения. Данные процедуры изображены на рисунке 16.

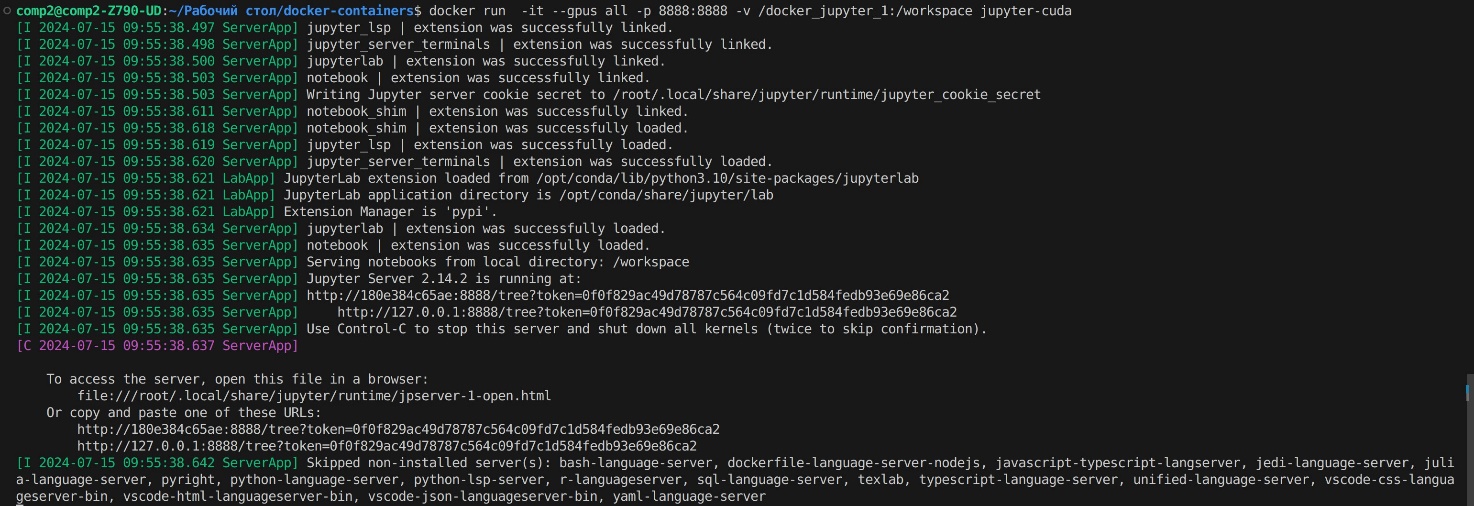


Рисунок 15 – Запуск контейнера jupyter-cuda

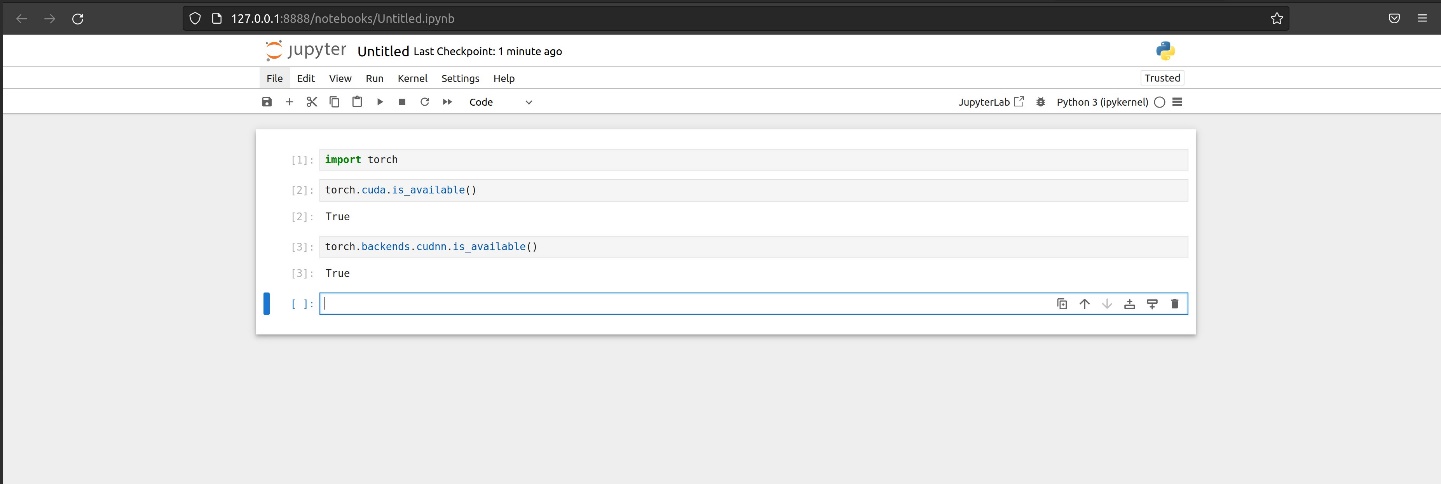


Рисунок 16 – Jupyter docker образ

Останавливаем контейнер и снова запускаем для проверки корректности монтирования образа. На рисунке 17 мы видим, что наш файл сохранился, значит все было выполнено корректно.

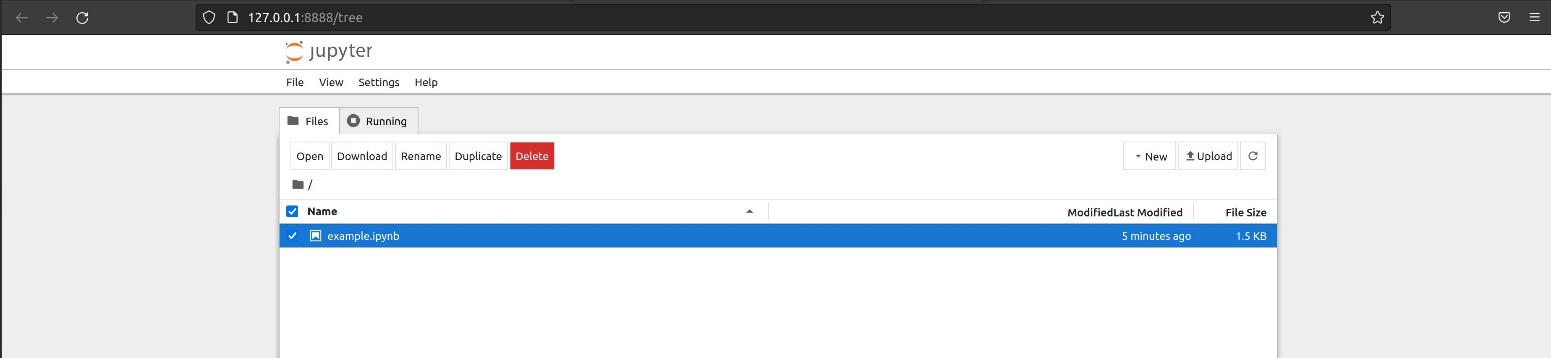


Рисунок 76 – Проверка корректности монтирования образа

На рисунке 18 изображены все контейнеры: запущенные и остановленные.

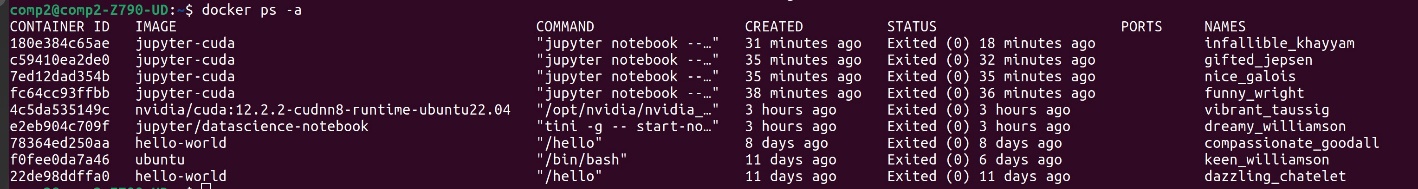


Рисунок 18 – Список всех Docker контейнеров

Nvidia Container Toolkit представляет собой набор инструментов, разработанных Nvidia, для интеграции возможностей работы с GPU в контейнерные среды, такие как Docker. Он позволяет контейнерам, запущенным в таких средах, использовать графические процессоры (GPU) для вычислений, что особенно полезно для задач, требующих высокой производительности, таких как машинное обучение, глубокое обучение и обработка больших данных. С помощью Nvidia Container Toolkit можно легко и эффективно переносить и запускать сложные вычислительные задачи в контейнерах, обеспечивая доступ к аппаратному ускорению на базе GPU.

Этот инструмент необходим для разработчиков и исследователей, работающих с высокопроизводительными вычислительными задачами, позволяя им оптимизировать и ускорить выполнение своих приложений. Nvidia Container Toolkit обеспечивает простую и гибкую настройку контейнеров с поддержкой GPU, что позволяет максимально эффективно использовать вычислительные ресурсы и улучшать производительность приложений. Это делает его незаменимым инструментом в экосистеме разработки и развертывания приложений с использованием GPU.

Далее устанавливаем Nvidia Container Toolkit при помощи следующих команд:

distribution=$(. /etc/os-release;echo $ID$VERSION\_ID)

curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -

curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list

sudo apt-get update && sudo apt-get install -y nvidia-container-toolkit

sudo systemctl restart docker

# Установка Anaconda

Anaconda – это дистрибуция Python и R, предназначенная для научных вычислений, анализа данных и машинного обучения. Она включает в себя множество пакетов и библиотек, необходимых для этих целей, а также инструменты для управления зависимостями и окружениями. Anaconda значительно упрощает установку и настройку среды для работы с данными, предоставляя пользователям всё необходимое в одном комплекте.

Основное предназначение Anaconda заключается в создании удобной и функциональной среды для работы с данными и разработке моделей машинного обучения. С её помощью можно легко управлять различными версиями библиотек и инструментов, что особенно полезно при работе над несколькими проектами. Anaconda также включает в себя Anaconda Navigator, графическую оболочку, которая позволяет пользователям управлять пакетами и окружениями через удобный интерфейс, что делает процесс работы с данными более интуитивным и доступным.

Устанавливаем Anaconda. Чтобы запустить графическую оболочку и работать в своем окружении нужно сначала перейти в директорию, где установлена anaconda3. Прописываем в терминале conda activate и anaconda-navigator. [6] Нам откроется графическая оболочка anaconda, данный пункт изображен на рисунке 19.

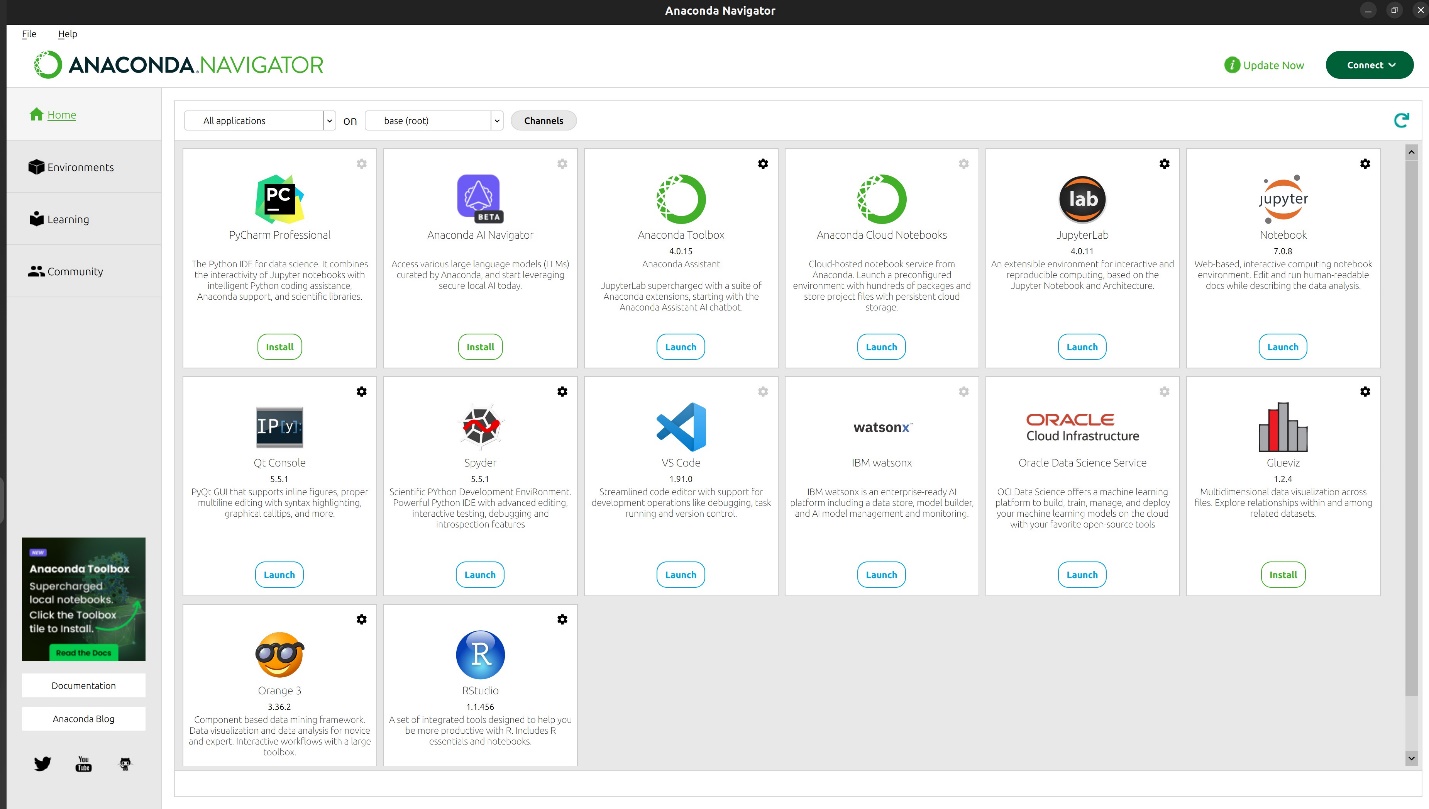


Рисунок 19 – Anaconda-navigator

Открываем anaconda-navigator из-под пользователя root. Можно создать свою среду для изолированной работы. На рисунке 20 изображены список сред.

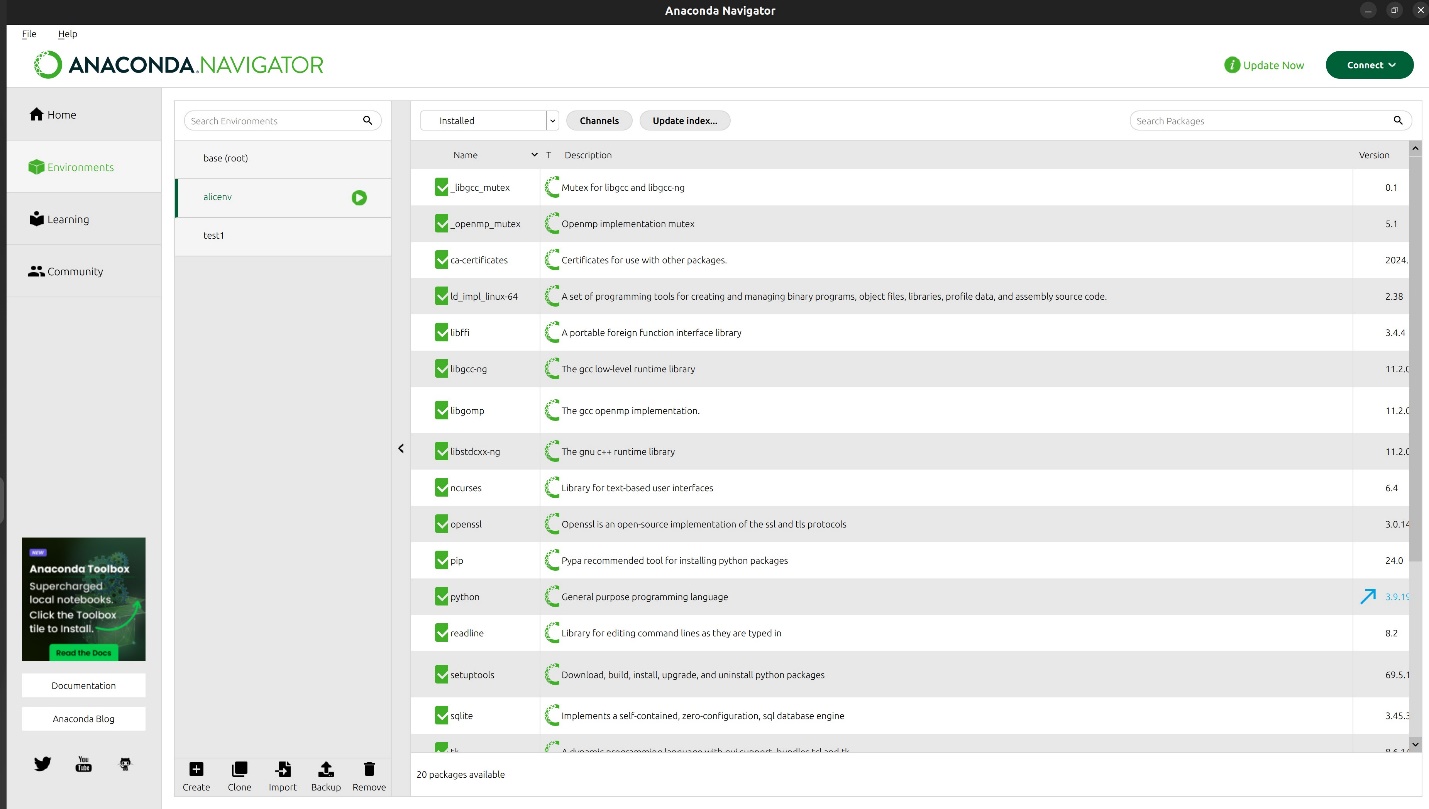


Рисунок 20 – Environments в Anaconda

На рисунке 21 изображена проверка работы видеокарты в Anaconda Jupyter. Все работает корректно.

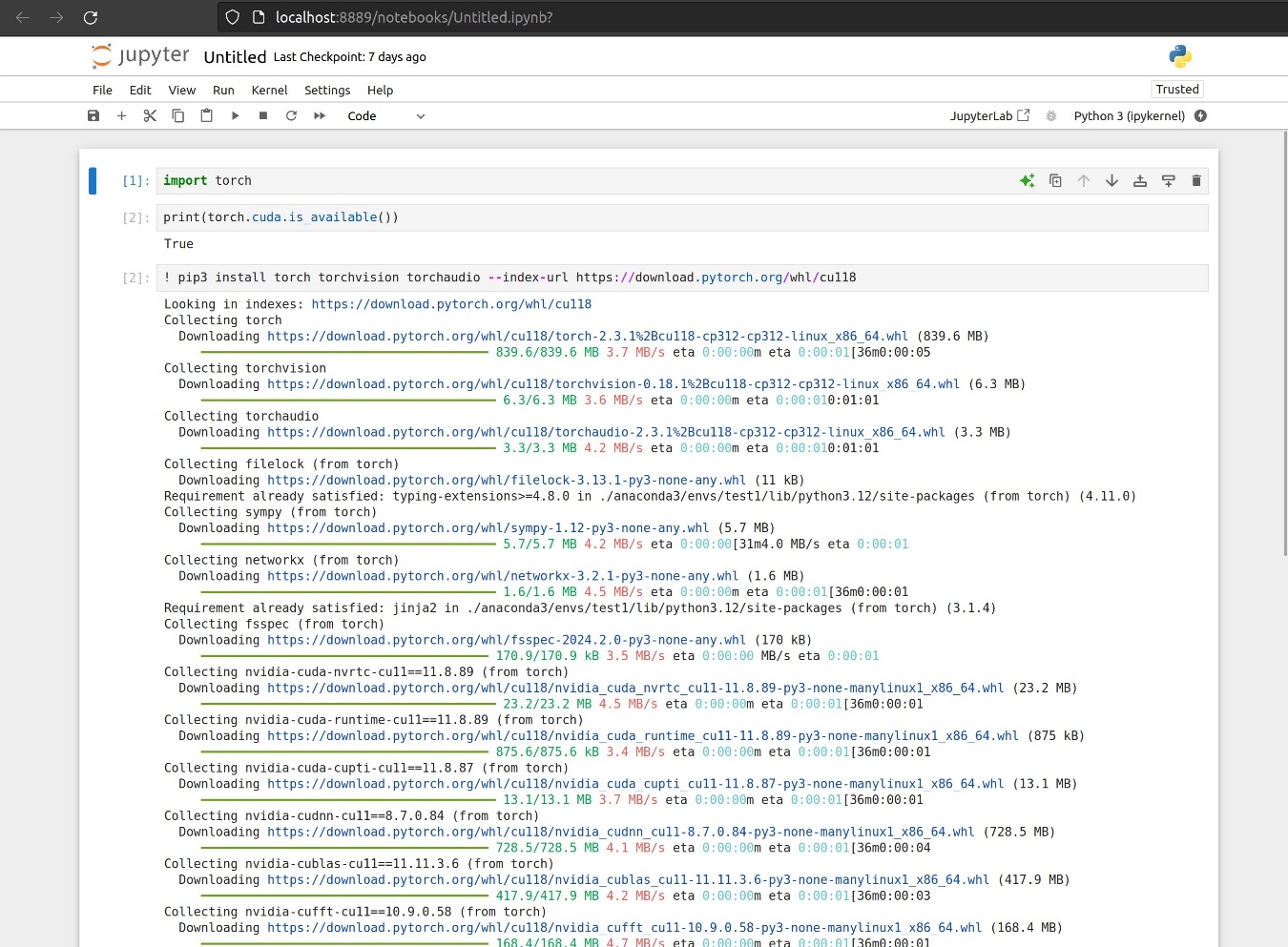


Рисунок 21 – Проверка работы видеокарты в Anaconda Jupyter

# Заключение

Цель учебной технологической практики заключалась в установке и настройке дистрибутива ubuntu, нужных драйверов для правильной работы видеокарты, anaconda, а также Docker.

Выполнение в учебной технологической практике задач:

1. Установлена и настроена Ubuntu 22.04 LTS.
2. Установлены драйвера видеокарты;
3. Установлена и протестирована Nvidia CUDA
4. Установлена и протестирована Nvidia CUDnn
5. Установлены Nvidia Container Toolkit и Docker
6. Установлена и настроена Anaconda

Поставленные в работе задачи решены в полном объеме.

# Список использованных источников

1. NVIDIA CUDA Installation Guide for Linux. URL: https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html (дата обращения: 7.01.2020).
2. Installing cuDNN on Linux. URL:https://docs.nvidia.com/deeplearning/cudnn/latest/installation/linux.html(дата обращения: 7.01.2020).
3. Docker. URL: https://docs.docker.com/( дата обращения: 7.01.2020).
4. Jupyter Docs. URL: https://docs.jupyter.org/en/latest/( дата обращения: 7.01.2020).
5. Pytorch Docs. URL: https://pytorch.org/get-started/locally/( дата обращения: 7.01.2020).
6. Anaconda Docs. URL: https://docs.anaconda.com/anaconda/install/linux/( дата обращения: 7.01.2020).