

# Введение в искусственный интеллект. Современное компьютерное зрение

Тема семинара: Введение. GPU, фреймворки и Colab

Бабин Д.Н., Иванов И.Е., Петюшко А.А.

кафедра Математической Теории Интеллектуальных Систем



## ① Организационные вопросы



# План семинара

- 1 Организационные вопросы
- 2 Обзор семинарских занятий

# План семинара

- 1 Организационные вопросы
- 2 Обзор семинарских занятий
- 3 Стек технологий для работы с компьютерным зрением

# План семинара

- 1 Организационные вопросы
- 2 Обзор семинарских занятий
- 3 Стек технологий для работы с компьютерным зрением
- 4 Google CoLab

- 1 Организационные вопросы
- 2 Обзор семинарских занятий
- 3 Стек технологий для работы с компьютерным зрением
- 4 Google CoLab
- 5 Сдача домашних заданий через Telegram-бота



- Данный курс является частью программы **SHARE**
  - **SHARE** = School of Huawei Advanced Research Education, или Школа опережающего научного образования Хуавэй
  - e-mail: [share@intsys.msu.ru](mailto:share@intsys.msu.ru)
  - Сайт SHARE: <http://sharemsu.ru>
  - Канал SHARE: <https://t.me/joinchat/9IzmCnQIyvs2NjUy>



# Оценки за курс

- Оценки за курс будут выставляться в соответствии с данными о посещении и набранными баллами за выполнение домашних заданий.





# Оценки за курс

- Оценки за курс будут выставляться в соответствии с данными о посещении и набранными баллами за выполнение домашних заданий.
- В ходе курса будут предложены домашние задания трёх типов:
  - теоретические
  - практические
  - соревнования



# Оценки за курс

- Оценки за курс будут выставляться в соответствии с данными о посещении и набранными баллами за выполнение домашних заданий.
- В ходе курса будут предложены домашние задания трёх типов:
  - теоретические
  - практические
  - соревнования
- В конце семестра состоится экзамен, на котором при желании можно будет повысить свою оценку



# Оценки за курс

- Оценки за курс будут выставляться в соответствии с данными о посещении и набранными баллами за выполнение домашних заданий.
- В ходе курса будут предложены домашние задания трёх типов:
  - теоретические
  - практические
  - соревнования
- В конце семестра состоится экзамен, на котором при желании можно будет повысить свою оценку
- Предварительная шкала оценок:

Оценка	Процент выполненных заданий
Отлично	80 %
Хорошо	60 %
Зачет	40 %



- Списывать (у других студентов) категорически запрещается!



- Списывать (у других студентов) категорически запрещается!
- При подозрении на списанную работу ставится 0 баллов:
  - Списавшему
  - Давшему списать



- Списывать (у других студентов) категорически запрещается!
- При подозрении на списанную работу ставится 0 баллов:
  - Списавшему
  - Давшему списать
- При использовании дополнительных источников (ресурсы в Интернете, учебники) обязательно ссылаться на них



- Страница курса: <https://github.com/mlcoursemm/cv2022spring>
- Главный ресурс по курсам “Введение в компьютерный интеллект”:  
<https://github.com/mlcoursemm>
- Почта курса: [mlcoursemm@gmail.com](mailto:mlcoursemm@gmail.com)



# Что будет в этом курсе [еще раз]

## Теоретическая часть

- Задачи компьютерного зрения
  - Классификация, детекция, сегментация
- Известные сверточные нейросети
  - AlexNet, VGG, ResNet, R-CNN, Mask R-CNN
- Генеративные модели
  - Генеративные состязательные сети, (вариационный) автоэнкодер
- Обучение нейросетей
  - Градиентный спуск, обратное распространение ошибки, инициализация





# Что будет в этом курсе [еще раз]

## Теоретическая часть

- Задачи компьютерного зрения
  - Классификация, детекция, сегментация
- Известные сверточные нейросети
  - AlexNet, VGG, ResNet, R-CNN, Mask R-CNN
- Генеративные модели
  - Генеративные состязательные сети, (вариационный) автоэнкодер
- Обучение нейросетей
  - Градиентный спуск, обратное распространение ошибки, инициализация

## Практическая часть

- Обработка изображений и нейросетевые фреймворки
  - Scikit-Image, Tensorflow, Keras
- Соревнования по компьютерному зрению

- ML Stack



- ML Stack
- Арифметика сверток



- ML Stack
- Арифметика сверток
- Прочие слои нейросетей



- ML Stack
- Арифметика сверток
- Прочие слои нейросетей
- Обратное распространение ошибок



- ML Stack
- Арифметика сверток
- Прочие слои нейросетей
- Обратное распространение ошибок
- Введение в фреймворк глубокого обучения



- ML Stack
- Арифметика сверток
- Прочие слои нейросетей
- Обратное распространение ошибок
- Введение в фреймворк глубокого обучения
- mAP и другие метрики детекции / сегментации



- ML Stack
- Арифметика сверток
- Прочие слои нейросетей
- Обратное распространение ошибок
- Введение в фреймворк глубокого обучения
- mAP и другие метрики детекции / сегментации
- Работа с изображениями





- ML Stack
- Арифметика сверток
- Прочие слои нейросетей
- Обратное распространение ошибок
- Введение в фреймворк глубокого обучения
- mAP и другие метрики детекции / сегментации
- Работа с изображениями
- Построение GAN



- ML Stack
- Арифметика сверток
- Прочие слои нейросетей
- Обратное распространение ошибок
- Введение в фреймворк глубокого обучения
- mAP и другие метрики детекции / сегментации
- Работа с изображениями
- Построение GAN
- Методы аугментации





- **Фреймворк** — программная платформа, определяющая структуру программной системы
- Отличие от библиотеки — инверсия управления



- **Фреймворк** — программная платформа, определяющая структуру программной системы
- Отличие от библиотеки — инверсия управления
  - Приложение вызывает функции (классы) библиотеки и получает управление после вызова



- **Фреймворк** — программная платформа, определяющая структуру программной системы
- Отличие от библиотеки — инверсия управления
  - Приложение вызывает функции (классы) библиотеки и получает управление после вызова
  - Приложение может реализовывать конкретное поведение, встраиваемое в более общий, “абстрактный” код фреймворка, который вызывает функции (классы) пользовательского кода



- **Фреймворк** — программная платформа, определяющая структуру программной системы
- Отличие от библиотеки — инверсия управления
  - Приложение вызывает функции (классы) библиотеки и получает управление после вызова
  - Приложение может реализовывать конкретное поведение, встраиваемое в более общий, “абстрактный” код фреймворка, который вызывает функции (классы) пользовательского кода
- Наиболее популярные фреймворки для компьютерного зрения (КЗ):



- **Фреймворк** — программная платформа, определяющая структуру программной системы
- Отличие от библиотеки — инверсия управления
  - Приложение вызывает функции (классы) библиотеки и получает управление после вызова
  - Приложение может реализовывать конкретное поведение, встраиваемое в более общий, “абстрактный” код фреймворка, который вызывает функции (классы) пользовательского кода
- Наиболее популярные фреймворки для компьютерного зрения (КЗ):
  - PyTorch





- **Фреймворк** — программная платформа, определяющая структуру программной системы
- Отличие от библиотеки — инверсия управления
  - Приложение вызывает функции (классы) библиотеки и получает управление после вызова
  - Приложение может реализовывать конкретное поведение, встраиваемое в более общий, “абстрактный” код фреймворка, который вызывает функции (классы) пользовательского кода
- Наиболее популярные фреймворки для компьютерного зрения (КЗ):
  - PyTorch
  - TensorFlow (посредством Keras) — удобен в качестве обучения



- **Фреймворк** — программная платформа, определяющая структуру программной системы
- Отличие от библиотеки — инверсия управления
  - Приложение вызывает функции (классы) библиотеки и получает управление после вызова
  - Приложение может реализовывать конкретное поведение, встраиваемое в более общий, “абстрактный” код фреймворка, который вызывает функции (классы) пользовательского кода
- Наиболее популярные фреймворки для компьютерного зрения (КЗ):
  - PyTorch
  - TensorFlow (посредством Keras) — удобен в качестве обучения
  - Вычисления в них понимаются как **потоки в графе вычислений**, где результат операции — **тензор** (многомерный массив)



- В нейросетевых фреймворках нужно реализовать:



- В нейросетевых фреймворках нужно реализовать:
  - построение графа вычислений



- В нейросетевых фреймворках нужно реализовать:
  - построение графа вычислений
  - метод его обучения



- В нейросетевых фреймворках нужно реализовать:
  - построение графа вычислений
  - метод его обучения
  - подачу данных



- В нейросетевых фреймворках нужно реализовать:
  - построение графа вычислений
  - метод его обучения
  - подачу данных
- Это делается с помощью кода приложения, которое написано на языке программирования (желательно) высокого уровня: например, Python, C/C++



- В нейросетевых фреймворках нужно реализовать:
  - построение графа вычислений
  - метод его обучения
  - подачу данных
- Это делается с помощью кода приложения, которое написано на языке программирования (желательно) высокого уровня: например, Python, C/C++
- При этом можно использовать дополнительные подходящие библиотеки (например, SciKit-Learn). В любом случае, пригодятся библиотеки:
  - Для работы с изображениями (Scikit-Image, OpenCV)
  - Для работы с тензорами (Numpy)





- На нижнем уровне приложение будет взаимодействовать с операционной системой (ОС), устройствами (например, видеокартой), а также драйверами для этих устройств



- На нижнем уровне приложение будет взаимодействовать с операционной системой (ОС), устройствами (например, видеокартой), а также драйверами для этих устройств
- Зачастую для удобства развертывания системы, а также изоляции родительской системы, применяются различные способы виртуализации:

- На нижнем уровне приложение будет взаимодействовать с операционной системой (ОС), устройствами (например, видеокартой), а также драйверами для этих устройств
- Зачастую для удобства развертывания системы, а также изоляции родительской системы, применяются различные способы виртуализации:
  - Удаленный доступ к рабочей станции (лучший учебный вариант)
  - Виртуальное окружение на уровне приложений (conda, virtualenv): можно попробовать дома
  - Виртуализация на уровне ОС (docker): продвинутый “индустриальный” способ



- На нижнем уровне приложение будет взаимодействовать с операционной системой (ОС), устройствами (например, видеокартой), а также драйверами для этих устройств
- Зачастую для удобства развертывания системы, а также изоляции родительской системы, применяются различные способы виртуализации:
  - Удаленный доступ к рабочей станции (лучший учебный вариант)
  - Виртуальное окружение на уровне приложений (conda, virtualenv): можно попробовать дома
  - Виртуализация на уровне ОС (docker): продвинутый “индустриальный” способ
- Иногда концепции используют несколько уровней в иерархии (например, CUDA)



- CUDA: **C**ompute **U**nified **D**evice **A**rchitecture



- CUDA: **C**ompute **U**nified **D**evice **A**rchitecture
- Это **программно-аппаратная** архитектура параллельных вычислений
  - То есть объединяет как драйвера для устройства (в данном случае — видеокарты),
  - Так и API (Application Programming Interface) для вызова специфических функций, как в библиотеке,
  - Так и конкретное устройство, которой поддерживает только определенную версию CUDA



- CUDA: **C**ompute **U**nified **D**evice **A**rchitecture
- Это **программно-аппаратная** архитектура параллельных вычислений
  - То есть объединяет как драйвера для устройства (в данном случае — видеокарты),
  - Так и API (Application Programming Interface) для вызова специфических функций, как в библиотеке,
  - Так и конкретное устройство, которой поддерживает только определенную версию CUDA
- Вывод: CUDA как библиотека неотделима от устройства



# Стек технологий: зачем нужны GPU

- В последнее время все больше задач КЗ решаются с помощью GPU





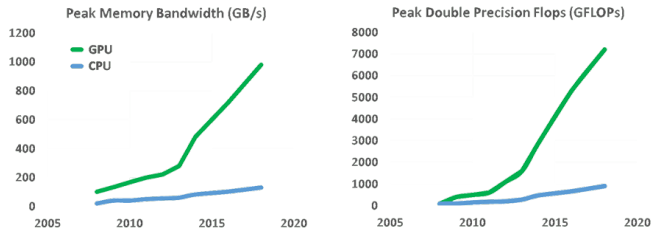
# Стек технологий: зачем нужны GPU

- В последнее время все больше задач КЗ решаются с помощью GPU
- GPU: **G**raphics **P**rocessing **U**nit (видеокарта)

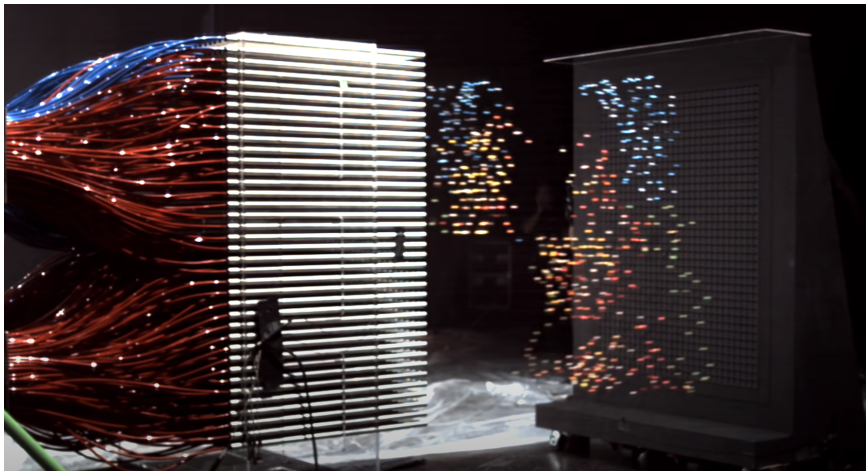


# Стек технологий: зачем нужны GPU

- В последнее время все больше задач КЗ решаются с помощью GPU
- GPU: **G**raphics **P**rocessing **U**nit (видеокарта)
- На данный момент GPU в разы превосходит CPU (процессор):
  - По пропускной способности памяти,
  - По количеству операций в секунду



# Стек технологий: зачем нужны GPU



# Стек технологий: почему нужны не только GPU

- Почему же тогда не переходят на GPU повсеместно?



# Стек технологий: почему нужны не только GPU

- Почему же тогда не переходят на GPU повсеместно?
- Дело в том, что потрясающая быстрота GPU достигается благодаря:
  - Специализированной архитектуре под узкий набор задач,
  - Ограниченному набору команд



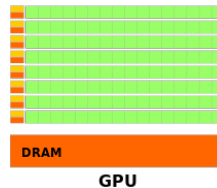
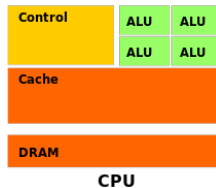
# Стек технологий: почему нужны не только GPU

- Почему же тогда не переходят на GPU повсеместно?
- Дело в том, что потрясающая быстрота GPU достигается благодаря:
  - Специализированной архитектуре под узкий набор задач,
  - Ограниченному набору команд
- То есть быстро на GPU перемножать матрицы можно, а писать код со множественными ветвлениями — уже нет



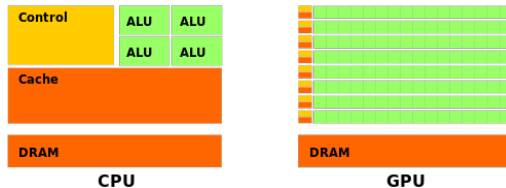
# Стек технологий: почему нужны не только GPU

- Почему же тогда не переходят на GPU повсеместно?
- Дело в том, что потрясающая быстрота GPU достигается благодаря:
  - Специализированной архитектуре под узкий набор задач,
  - Ограниченному набору команд
- То есть быстро на GPU перемножать матрицы можно, а писать код со множественными ветвлениями — уже нет



# Стек технологий: почему нужны не только GPU

- Почему же тогда не переходят на GPU повсеместно?
- Дело в том, что потрясающая быстрота GPU достигается благодаря:
  - Специализированной архитектуре под узкий набор задач,
  - Ограниченному набору команд
- То есть быстро на GPU перемножать матрицы можно, а писать код со множественными ветвлениями — уже нет



- Особенности:
  - Большое число ядер вычислений (для выполнения простых параллелизуемых вычислений)
  - Много потоков, в каждом из которых свой небольшой кэш и набор команд
  - Процесс только один (хотя возможна эмуляция с помощью контекста)



# Стек технологий: какие GPU наиболее распространены

- В настоящее время наиболее распространены GPU от NVIDIA (в дополнение к Google TPU / Huawei AI Chip)



# Стек технологий: какие GPU наиболее распространены

- В настоящее время наиболее распространены GPU от NVIDIA (в дополнение к Google TPU / Huawei AI Chip)
- По существу они различаются:
  - По области применения (серверные и десктопные),
  - По используемой архитектуре



# Стек технологий: какие GPU наиболее распространены

- В настоящее время наиболее распространены GPU от NVIDIA (в дополнение к Google TPU / Huawei AI Chip)
- По существу они различаются:
  - По области применения (серверные и десктопные),
  - По используемой архитектуре
- Также важным параметром является объем внутренней памяти (от 10 ГБ, чем больше — тем лучше)



# Стек технологий: какие GPU наиболее распространены

- В настоящее время наиболее распространены GPU от NVIDIA (в дополнение к Google TPU / Huawei AI Chip)
- По существу они различаются:
  - По области применения (серверные и десктопные),
  - По используемой архитектуре
- Также важным параметром является объем внутренней памяти (от 10 ГБ, чем больше — тем лучше)
- Интерфейс — PCIe



# Стек технологий: какие GPU наиболее распространены

- В настоящее время наиболее распространены GPU от NVIDIA (в дополнение к Google TPU / Huawei AI Chip)
- По существу они различаются:
  - По области применения (серверные и десктопные),
  - По используемой архитектуре
- Также важным параметром является объем внутренней памяти (от 10 ГБ, чем больше — тем лучше)
- Интерфейс — PCIe
- Эволюция архитектур:
  - 2016 — Pascal
  - 2017 — Volta
  - 2018 — Turing
  - 2020 — Ampere



# Стек технологий: список подходящих GPU (desktopные)

GPU	Архитектура	Память	CUDA	CUDA SDK
GTX 1080 Ti	Pascal	11 Гб	6.1	8.0+
Titan X(p)	Pascal	12 Гб	6.1	8.0+
Titan V	Volta	12/32 Гб	7.0	9.0+
RTX 2080 Ti	Turing	11 Гб	7.5	10.0+
Titan RTX	Turing	24 Гб	7.5	10.0+
RTX 3080	Ampere	10 Гб	8.6	11.0+
RTX 3090	Ampere	24 Гб	8.6	11.0+

# Стек технологий: список подходящих GPU (серверные)

GPU	Архитектура	Память	CUDA	CUDA SDK
P40	Pascal	24 Гб	6.1	8.0+
P100	Pascal	12/16 Гб	6.0	8.0+
V100	Volta	16/32 Гб	7.0	9.0+
T4	Turing	16 Гб	7.5	10.0+
A100	Ampere	40 Гб	8.6	11.0+

- Сервис от Google: Colaboratory (совместная лаборатория), или сокращенно **Colab**

---

<sup>1</sup><https://colab.research.google.com/notebooks/intro.ipynb>



- Сервис от Google: Colaboratory (совместная лаборатория), или сокращенно **Colab**
- Что такое Colab:
  - Доступ к виртуальной машине (можно понимать как работу в докер-контейнере) с Linux
  - Python с большим количеством предустановленных пакетов

---

<sup>1</sup><https://colab.research.google.com/notebooks/intro.ipynb>

- Сервис от Google: Colaboratory (совместная лаборатория), или сокращенно **Colab**
- Что такое Colab:
  - Доступ к виртуальной машине (можно понимать как работу в докер-контейнере) с Linux
  - Python с большим количеством предустановленных пакетов
- Что дает Colab:
  - Возможность работать в jupyter-notebook подобному интерфейсу взаимодействия удаленно (не имея на компьютере даже Python)
  - Возможность использования мощной видеокарты
  - Возможность доустановки необходимых пакетов
    - `pip install`
    - `apt-get install`
    - `make / cmake`

---

<sup>1</sup><https://colab.research.google.com/notebooks/intro.ipynb>

- Виртуальная машина с Ubuntu 18.04



- Виртуальная машина с Ubuntu 18.04
- Серверная видеокарта Tesla T4 с 16 ГБ памяти



- Виртуальная машина с Ubuntu 18.04
- Серверная видеокарта Tesla T4 с 16 ГБ памяти
- Важные предустановленные пакеты Python:
  - h5py 3.1.0
  - Keras 2.7.0
  - numpy 1.19.5
  - opencv-python (+contrib) 4.1.2.30
  - pandas 1.3.5
  - Pillow 7.1.2
  - scikit-image 0.18.3
  - scikit-learn 1.0.2
  - scipy 1.4.1
  - tensorflow 2.7.0



# Подключение Colab

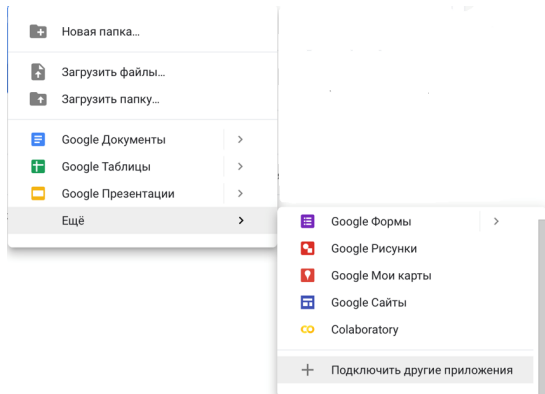
- Нужно подключить Colab как приложение к своему диску Google<sup>2</sup> через правую кнопку мыши в интерфейсе Google Drive

---

<sup>2</sup>Google Drive

# Подключение Colab

- Нужно подключить Colab как приложение к своему диску Google<sup>2</sup> через правую кнопку мыши в интерфейсе Google Drive
- Это нужно сделать только один раз



<sup>2</sup>Google Drive

# Запуск Colab (1)

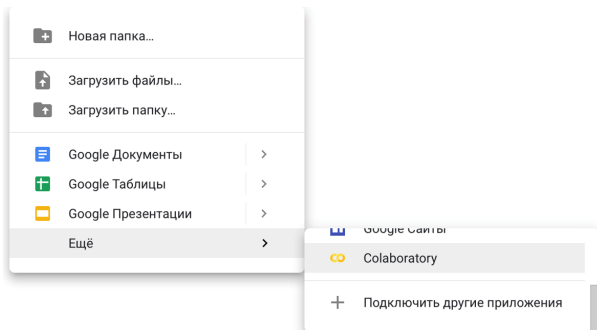
- Запускаем виртуальную машину и одновременно ноутбук для доступа к ней





# Запуск Colab (1)

- Запускаем виртуальную машину и одновременно ноутбук для доступа к ней
- Ноутбук будет сохранен в диске Google



## Запуск Colab (2)

- Также можно запускать любой `.ipynb` ноутбук из github:



## Запуск Colab (2)

- Также можно запускать любой `.ipynb` ноутбук из github:
  - Пусть ноутбук находится по адресу: `https://github.com/mlcoursemm/ml2020autumn/blob/master/seminars/seminar01-intro_cv_bv.ipynb`



## Запуск Colab (2)

- Также можно запускать любой .ipynb ноутбук из github:
  - Пусть ноутбук находится по адресу: `https://github.com/mlcoursemm/ml2020autumn/blob/master/seminars/seminar01-intro\_cv\_bv.ipynb`
  - Для его открытия на Colab переходим по адресу `https://colab.research.google.com/github/mlcoursemm/ml2020autumn/blob/master/seminars/seminar01-intro\_cv\_bv.ipynb`

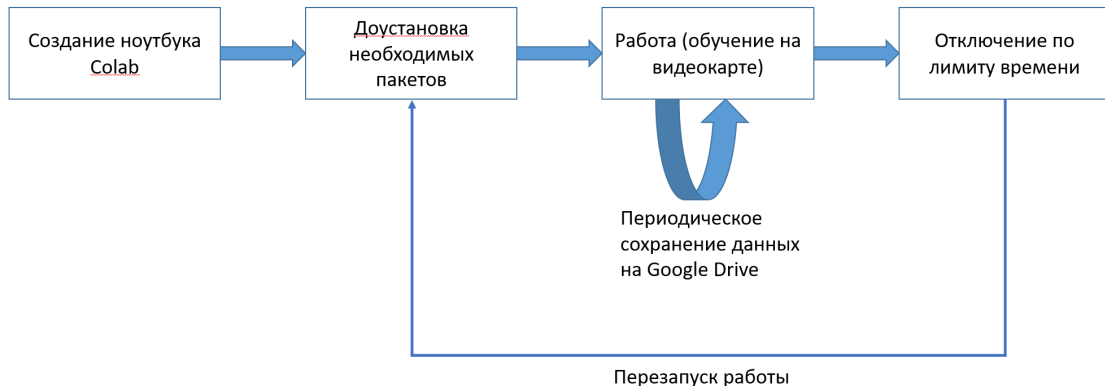


## Запуск Colab (2)

- Также можно запускать любой `.ipynb` ноутбук из github:
  - Пусть ноутбук находится по адресу: `https://github.com/mlcoursemm/ml2020autumn/blob/master/seminars/seminar01-intro_cv_bv.ipynb`
  - Для его открытия на Colab переходим по адресу `https://colab.research.google.com/github/mlcoursemm/ml2020autumn/blob/master/seminars/seminar01-intro_cv_bv.ipynb`
  - Чтобы иметь возможность сохранять изменения и / или результаты работы, то лучше сразу копировать ноутбук к себе на диск: **File** → **Save a copy in Drive**
    - Он будет помещен в папку "Colab Notebooks" в вашем диске



# Цикл работы с Colab



# Особенности работы в Colab (1)

- Нужно понимать, что эксклюзивный доступ к виртуальной машине дается на ограниченное время (в районе 8–12 часов, более того, иногда может выкинуть раньше указанного времени), после чего ее состояние сбрасывается и необходимо заново ее запускать и все устанавливать
- Поэтому необходимо:
  - Все данные (например, checkpoints обучаемых моделей) **периодически сохранять** на свой диск (например, раз в полчаса, или каждые 1000 итераций)
  - Каждый раз в начале работы в ноутбуке **ДОустанавливать необходимые пакеты** (если это необходимо)
- Если виртуальная машина зависла и не отвечает, ее можно принудительно перезапустить через

```
! kill -9 -1
```



## Особенности работы в Colab (2)

- Вы в виртуальной машине имеете все права (являетесь root), и даже нет нужды писать “sudo” перед каждой командой
- Если необходимо использовать видеокарту, то это нужно сделать в самом начале работы (машина перезапустится при попытке подключения): **Edit → Notebook settings → Hardware accelerator → GPU** либо **Runtime → Change runtime type → Hardware accelerator → GPU**
- По файловой системе лучше перемещаться через

```
%cd /path
```

или

```
import os  
os.chdir('/path')
```

не доверяя переходам по папкам через

```
!cd /path
```





# Обмен данными с Colab (1)

- Основной способ — через свой Google диск
- Диск будет смонтирован в “/content/drive/MyDrive”

```
from google.colab import drive
drive.mount('/content/drive/')
```

①

```
from google.colab import drive
drive.mount('/content/drive/')
```

Go to this URL in a browser: [https://accounts.google.com/o/oauth2/auth?client\\_id=...](https://accounts.google.com/o/oauth2/auth?client_id=...)

Enter your authorization code:

③

Вход

Скопируйте код, перейдите в приложение и вставьте его в нужное поле:

4/1AY0e- [redacted] 9G0ppUji-

7EmbCtqhGGhiuUqwDM

②

Приложение "Google Drive" запрашивает разрешение на доступ к вашему аккаунту Google

Приложение "Google Drive" сможет выполнить следующие действия:

- Просмотр, создание, изменение и удаление всех файлов на Google Диске
- Google Фото – сервис для хранения и просмотра фотографий, видео и альбомов
- Просмотр информации о пользователе Google, доступной, например, в профилях и контактах
- Просмотр записей о действиях с файлами, которые хранятся у вас на Google Диске
- Просмотр, создание, изменение и удаление документов на Google Диске

Убедитесь в надежности сервиса "Google Drive"

Этот сайт или приложение сможет получить доступ к конфиденциальной информации. Соглашаетесь с условиями использования и политики конфиденциальности приложения "Google Drive", чтобы узнать, как будут обрабатываться ваши данные. Посмотреть или удалить приложение и сайты с доступом к вашему аккаунту можно на странице [Аккаунт Google](#).

Подробнее об угрозах безопасности...

Отмена Продолжить

## Обмен данными с Colab (2)

- Другой способ — через веб-интерфейс
- Загрузить к себе на компьютер:

```
from google.colab import files  
files.download('hello.py')
```

- Загрузить в виртуальную машину со своего компьютера:

```
from google.colab import files  
uploaded = files.upload()
```

После загрузки дополнительно в словаре “uploaded” будут созданы записи вида “filename : bytecontent”

- Ну и всегда можно воспользоваться “wget” / “git clone”



# Telegram-bot (1)

- Домашние задания: отправлять через Telegram-бота под именем “@cv2022sharebot”
- Сначала нужно зарегистрироваться:

```
/register Familia Imia Otchestvo
```

- Затем можно посмотреть список доступных заданий:

```
/start
```

- Получить информацию о домашнем задании:

```
/info seminar01
```



- Получить домашнее задание:

```
/get seminar01
```

- Затем загрузить результат выполнения домашнего задания с правильной подписью (*caption*) “**seminar01**”, нужным именем файла и в срок (см. пункт /info)
- Проверить результат выполнения (**последний загруженный файл с правильным именем файлом**):

```
/check seminar01
```





Спасибо за внимание!