Введение в искусственный интеллект. Современное компьютерное зрение

Семинар 1. Вводный. GPU, фреймворки и Colab

Бабин Д.Н., Иванов И.Е., Петюшко А.А.

кафедра Математической Теории Интеллектуальных Систем

16 февраля 2021 г.

План семинара

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

SHARE

- 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`
  - Сайт SHARE: `http://sharemsu.ru`
  - Канал SHARE: `https://t.me/joinchat/AAAAAE_r4XKzEDaUKy1FwA`
  - Чат SHARE: `https://t.me/joinchat/AAAAAEnwHm0FStzFxKtS8w`

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 %

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 %

За посещение каждого занятия балл увеличивается примерно на 1%.

Кодекс чести SHARE

- Списывать (у других студентов) **категорически запрещается**!

Кодекс чести SHARE

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

Кодекс чести SHARE

- 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

Полезные ресурсы

- Страница курса: `https://github.com/mlcoursemm/cv2021spring`
- Главный ресурс по курсам “Введение в компьютерный интеллект”: `https://github.com/mlcoursemm`
- Телеграмм-канал: `https://t.me/joinchat/AAAAAEUmx5cJLOdLXsOt8g`
- Группа обсуждения: `https://t.me/joinchat/AAAAAEx8IrWw-nYJPo6smQ`
- Почта курса: `mlcoursemm@gmail.com`
  - Именно сюда нужно будет посылать свои домашние задания!

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

Обзор семинаров

- ML Stack

Обзор семинаров

- ML Stack
- Арифметика сверток

Обзор семинаров

- ML Stack
- Арифметика сверток
- Прочие слои нейросетей

Обзор семинаров

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

Обзор семинаров

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

Обзор семинаров

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

Обзор семинаров

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

Обзор семинаров

- 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

Обзор семинаров

- 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

Стек технологий

Стек технологий: верхний уровень

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

Стек технологий: верхний уровень

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

Стек технологий: верхний уровень

- **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

Стек технологий: верхний уровень

- **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):

Стек технологий: верхний уровень

- **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

Стек технологий: верхний уровень

- **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

Стек технологий: верхний уровень

- **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)

Стек технологий: средний уровень

- В нейросетевых фреймворках нужно реализовать:

Стек технологий: средний уровень

- В нейросетевых фреймворках нужно реализовать:
  - построение графа вычислений

Стек технологий: средний уровень

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

Стек технологий: средний уровень

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

Стек технологий: средний уровень

- 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++

Стек технологий: средний уровень

- 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)

Стек технологий: нижний уровень

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

Стек технологий: нижний уровень

- 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:

Стек технологий: нижний уровень

- 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

Стек технологий: нижний уровень

- 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)

Стек технологий: что такое CUDA

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

Стек технологий: что такое 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

Стек технологий: что такое 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

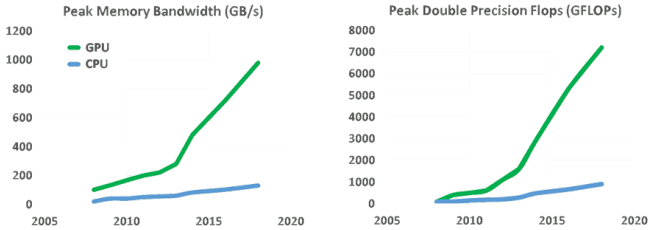
- В последнее время все больше задач КЗ решаются с помощью GPU

Стек технологий: зачем нужны GPU

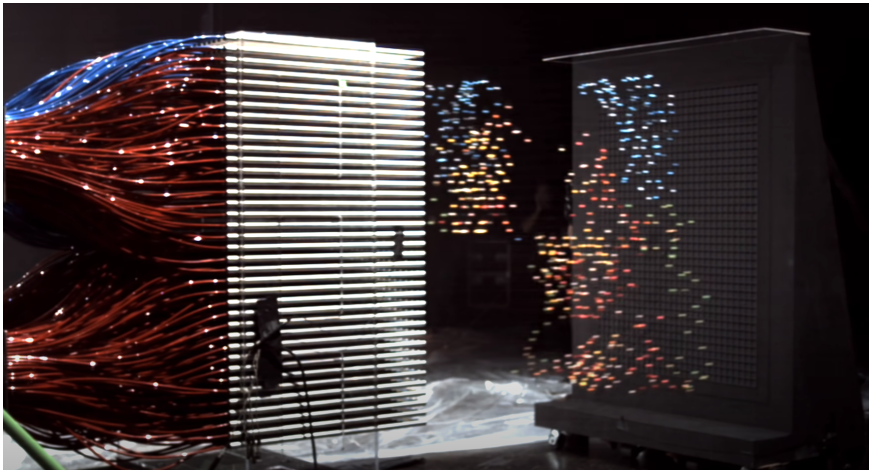
- В последнее время все больше задач КЗ решаются с помощью GPU
- GPU: **G**raphics **P**rocessing **U**nit (видеокарта)

Стек технологий: зачем нужны GPU

- В последнее время все больше задач К3 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 (десктопные)

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+

Google Colab[1]

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

[1] https://colab.research.google.com/notebooks/intro.ipynb

Google Colab[1]

- 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

[1] `https://colab.research.google.com/notebooks/intro.ipynb`

Google Colab[1]

- 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

[1] https://colab.research.google.com/notebooks/intro.ipynb

Возможности Colab

- Виртуальная машина с Ubuntu 18.04

Возможности Colab

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

Возможности Colab

- Виртуальная машина с Ubuntu 18.04
- Серверная видеокарта Tesla T4 с 16 ГБ памяти
- Важные предустановленные пакеты Python:
  - h5py 2.10.0
  - Keras 2.4.3
  - numpy 1.19.5
  - opencv-python (+contrib) 4.1.2.30
  - pandas 1.1.5
  - Pillow 7.0.0
  - scikit-image 0.16.2
  - scikit-learn 0.22.2.post1
  - scipy 1.4.1
  - tensorflow 2.4.1

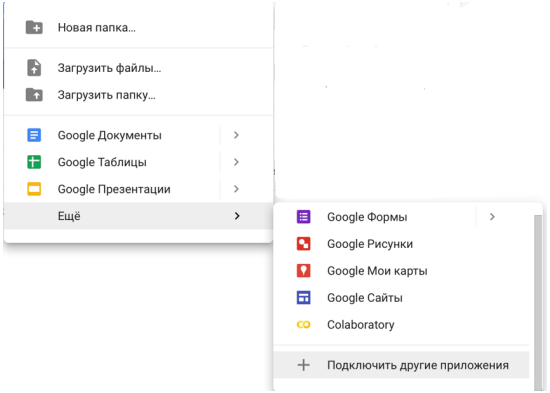
Подключение Colab

- Нужно подключить Colab как приложение к своему диску Google[2] через правую кнопку мыши в интерфейсе Google Drive

[2] Google Drive

Подключение Colab

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



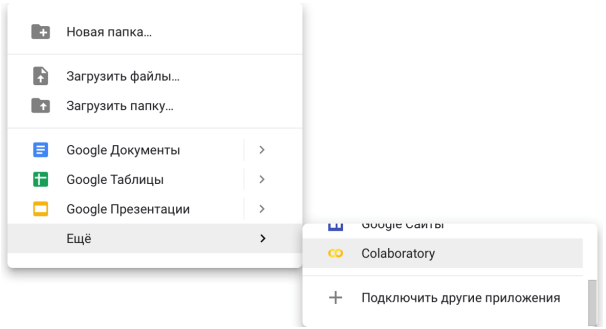
[2]Google Drive

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
```

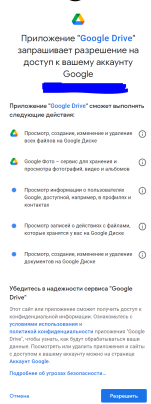
Обмен данными с Colab (1)

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

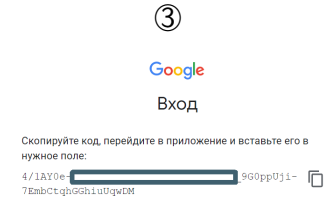
```
from google.colab import drive
drive.mount('/content/drive/')
```

①

②



③



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`”

Спасибо за внимание!