Airbus Ship Detection Challenge Documentation

Release 0.0.1

Michał DYzma et. al.

12/08/2018 14:32

Contents

Ostatnio zmieniony	12/08/2018 14:32

Wyzwanie zostało opublikowane na portalu Kaggle *30. lipca 2018*. Dokumentacja będzie pisana w języku Polskim, ale ze względu na specyfikę dziedziny będę stosował wiele zapożyczeń. Purystów językowych z góry przepraszam.

Wyzwanie dotyczy opracowania algorytmu, który poprawnie sklasyfikuje i oznaczy na zdjęciu satelitarnym statek. Algorytm musi być jak najbardziej wydajny i precyzyjny. Dokładny opis wyzwania znajduje się pod linkiem podanym powyżej.

CHAPTER 1

Spis treści

1.1 Szybki start

Ogólne informacje odnośnie projektu. Zakładam, że uczestnicy znają Pythona w stopniu minimum podstawowym i posiadają przynajmniej ogólną wiedzę dotyczxącą zagadnień Machine Learning i Deep Learning. W toku prac przygotuję odpowiednie "bryki" dotyczące szczegółów implementacji sieci neuronowych i zasad ich działania. Różnice, plusy, minusy.

1.1.1 Ważne linki

Nazwa	URL
GitHub	https://github.com/mdyzma/airbus-ship-detection.git
Slack	https://kagglepolska.slack.com/messages/CC4MR2S9G
Dokumentacja	https://airbus-ship-detection.readthedocs.io/en/latest/

1.1.2 Terminy

Start	**30. lipiec 2018**
Łączenie grup	**27. Wrzesień 2018**
Submisje	**27. Wrzesień 2018**
Koniec konkursu	**4. Październik 2018 23:59 UTC**

1.1.3 Nagrody

Łączna pula nagród: **$60,000**

Pierwsza nagroda	$25,000
Druga nagroda	$15,000
Trzecia nagroda	$5,000
Najszybszy algorytm	$15,000

1.1.4 Wybrane zasady

1. Pięć submisji na dobę
2. Możliwość wyboru 2 do oceny finałowej.
3. Zakaz używania zewnętrznych zestawów danych (patrz dyskusja), ale możliwość używania pre-trenowanych sieci pod warunkiem ich rejestracji na Forum.

1.1.5 Proponowane etapy i działania

Większość skryptów zmieści się w pojedynczym notebooku, który zamieszczę na Platformie Colab od Google (ze względu na możliwość trenowania na GPU). Tam też wrzucę dane treningowe. Na początku testowe. Dodam, że każdy może robić próby sam na własnym koncie google colab używając niewielkiej, losowej próbki kilku tysięcy zdjęć (w zestawie treningowym jest ich 104 000). Proponuję więc już teraz nauczyć się obsługi colab i zobaczyć jak idzie uruchamianie GPU. Możemy się też umówić na sesję treningową, którą poprowadzę i każdy zaproszony będzie mógł zobaczyć jak to wygląda w praktyce.

1. **Ściągnięcie zestawu danych (każdy musi indywidualnie)**
    - waga 26 GB
    - konto google airbus-kaggle-test
    - konto google airbus-kaggle-train
2. **Zaznajomienie z interfejsem i działaniem Google Colab**
    - uruchamianie kodu pythona
    - używanie preinstalowanych bibliotek
    - akceleracja GPU
3. **Zadania koncepcyjne:**
    - Losowanie próbki do testowania zmian obrazków (ostrość, kanały barw etc). Trochę EDA i prawdopodobieństwa.
    - dobór rodzaju modyfikacji obrazków. Tutaj trochę teorii o kodowaniu kolorów, kanałach i sposobach ich analizy oraz modyfikacji
    - dobór odpowiedniej metody/metod ewaluacji. Trochę teorii o rodzajach sieci nuronowych i ich zastosowaniach
    - dobór pre-trenowanych algorytmów. Przedstawienie istniejących rozwiązań OpenSource.
4. **Zadania programistyczne:**
    - Skrypt sprawdzający poprawność zdjęć (czy wszystkie są 768x768)
    - wprowadzenie do scikit-learn
    - Wprowadzenie do biblioteki KERAS
    - Generator danych

- Modyfikacja danych(zdjęc)
- Próba znanych pre-trenowanych sieci neuronowych i zmian danych na sub-próbie testowej (szybkość)
- Ostateczne uformowanie etapów pracy w pipeline
- Trening wszystkich danych treningowych na GPU Colab (potrzebne nowe konto, zestaw testowy ma 14.5GB)
- Ewaluacja przy użyciu danych testowych (osobne konto z przeniesioną siecią neuronową)
- Modyfikacje, prunning (opcjonalne) <- GOTO 6
- Zapisanie modelu
- Predykcja i utworzenie pliku do submisji

Każdy z tych etapów będzie wymagał jakiejś dyskusji (slack/hangout, ponieważ pisanie tekstu zajmuje więcej czasu, niż mówienie do mikrofonu). Do administracji możemy użyć trello z poziomu Slacka, lub tablicy githuba do zarządzania zadaniami. Ponieważ, to ja jestem pomysłodawcą, będę również koordynatorem wprowadzającym przedyskutowane zmiany, etapy na koncie projektu. Jednak proponuję wyznaczyć kilka osób, które również będą mogły działać w projekcie (dodam je do grupy na GH) na wypadek mojej absencji (rodzina, dom, wakacje :-))

1.1.6 Pozostałe sprawy organizacyjne

Generalnie możemy się umówić na codzienne spotkania na colab (10-15 minut max). Będziemy w miarę możliwości "symulować" metodologię scrum, ale dopasowaną do naszych potrzeb. Mamy już kanał na slack, żeby móc się komunikować (jest możliwość dzwonienia do siebie, nie wiem jak z wieloosobowymi telekonferencjami). Jeżeli ktoś ma możliwości można też podczas sesji z colab używać hangouta, tam z pewnością można zapraszać wielu uczestników. Termin spotkań do ustalenia.

Kolejne rozdziały będą opisem tego jak co zrobić, lub zbiorem linków do przydatnych tutoriali i artykułów w sieci.

Przewiduję też sesje "na żywo" w colab, gdzie będziemy rozwiązywać problemy na bieżąco.

1.2 Google Colab i Kaggle

W trakcie realizacji zadania mamy możliwość używania Notebooka serwowanego przez Gogole Colaboratory, lub portal Kaggle. Oba mają swoje wady i zalety. Poniżej przedstawię je i opiszę jak z nich korzystać.

1.2.1 Google Colab

Dokument opisuje pokrótce jak podłączyć i używać produktu Google zwanego colab, które jest po prostu notatnikiem ipythona (ipython notebook) z warstwą współdzielenia podobną do Google Docs i opierającym się na autentykacji via konto google. Pliki zapisywane są na naszym koncie w Google Drive. Niewątpliwą zaletą Colab jest udostępnianie za darmo całej przestrzeni dyskowej przypisanej do danego konta Google (około 15-17 GB) i udostępnianie pojedynczej jednostki GPU (Nvidia Tesla K80) do obliczeń. Colab posiada też znaczą ilość preinstalowanych bibliotek pythona nakierowanych na machine learning (listing z 08.08.2018):

```
!pip list

Package                  Version
------------------------ ---------
absl-py                  0.3.0
altair                   2.1.0
```

(continues on next page)

(continued from previous page)

```
astor                     0.7.1
beautifulsoup4            4.6.1
bleach                    2.1.3
cachetools                2.1.0
certifi                   2018.4.16
chardet                   3.0.4
crcmod                    1.7
cycler                    0.10.0
decorator                 4.3.0
entrypoints               0.2.3
future                    0.16.0
gast                      0.2.0
google-api-core           1.3.0
google-api-python-client  1.6.7
google-auth               1.4.2
google-auth-httplib2      0.0.3
google-auth-oauthlib      0.2.0
google-cloud-bigquery     1.1.0
google-cloud-core         0.28.1
google-cloud-language     1.0.2
google-cloud-storage      1.8.0
google-cloud-translate    1.3.1
google-colab              0.0.1a1
google-resumable-media    0.3.1
googleapis-common-protos  1.5.3
grpcio                    1.14.0
h5py                      2.8.0
html5lib                  1.0.1
httplib2                  0.11.3
idna                      2.6
ipykernel                 4.6.1
ipython                   5.5.0
ipython-genutils          0.2.0
Jinja2                    2.10
jsonschema                2.6.0
jupyter-client            5.2.3
jupyter-core              4.4.0
Keras                     2.1.6
Markdown                  2.6.11
MarkupSafe                1.0
matplotlib                2.1.2
mistune                   0.8.3
mpmath                    1.0.0
nbconvert                 5.3.1
nbformat                  4.4.0
networkx                  2.1
nltk                      3.2.5
notebook                  5.2.2
numpy                     1.14.5
oauth2client              4.1.2
oauthlib                  2.1.0
olefile                   0.45.1
opencv-python             3.4.2.17
packaging                 17.1
pandas                    0.22.0
pandas-gbq                0.4.1
pandocfilters             1.4.2
```

(continues on next page)

(continued from previous page)

```
patsy                   0.5.0
pexpect                 4.6.0
pickleshare             0.7.4
Pillow                  4.0.0
pip                     18.0
plotly                  1.12.12
pluggy                  0.7.1
portpicker              1.2.0
prompt-toolkit          1.0.15
protobuf                3.6.0
psutil                  5.4.6
ptyprocess              0.6.0
py                      1.5.4
pyasn1                  0.4.4
pyasn1-modules          0.2.2
Pygments                2.1.3
pyparsing               2.2.0
pystache                0.5.4
python-dateutil         2.5.3
pytz                    2018.5
PyWavelets              0.5.2
PyYAML                  3.13
pyzmq                   16.0.4
requests                2.18.4
requests-oauthlib       1.0.0
rsa                     3.4.2
scikit-image            0.13.1
scikit-learn            0.19.2
scipy                   0.19.1
seaborn                 0.7.1
setuptools              39.1.0
simplegeneric           0.8.1
six                     1.11.0
statsmodels             0.8.0
sympy                   1.1.1
tensorboard             1.9.0
tensorflow              1.9.0
tensorflow-hub          0.1.1
termcolor               1.1.0
terminado               0.8.1
testpath                0.3.1
toolz                   0.9.0
tornado                 4.5.3
tox                     3.1.2
traitlets               4.3.2
typing                  3.6.4
uritemplate             3.0.0
urllib3                 1.22
vega-datasets           0.5.0
virtualenv              16.0.0
wcwidth                 0.1.7
webencodings            0.5.1
Werkzeug                0.14.1
wheel                   0.31.1
xgboost                 0.7.post4
```

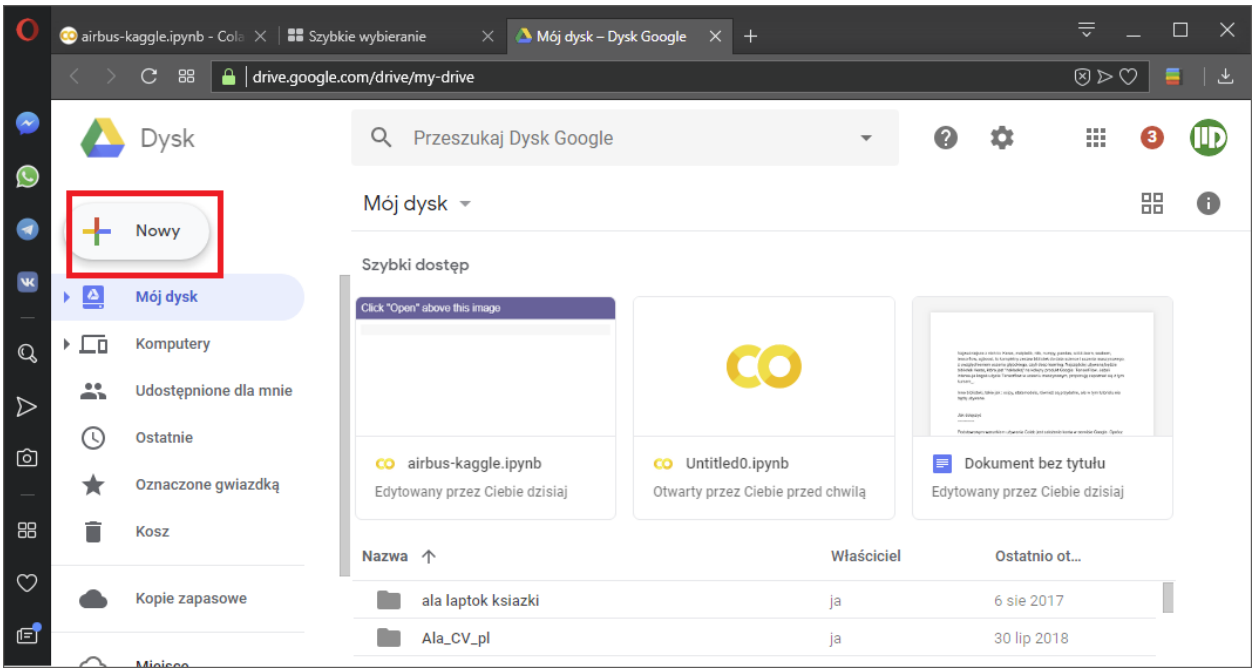
Najważniejsze z nich to: Keras, matplolib, nltk, numpy, pandas, scikit-learn, seaborn, tensorflow, xgboost, to kom-

pletny zestaw bibliotek do data science i uczenia maszynowego, z uwzględnieniem uczenia głębokiego, czyli deep learning. Najczęściej używaną będzie bibliotek Keras, która jest "nakładką" na kolejny produkt Google: TensorFlow. Jeżeli interesuje kogoś użycie Tensorflow w uczeniu maszynowym, proponuję zapoznać się z tym kursem.

Inne biblioteki, takie jak : scipy, statsmodels, również są przydatne, ale w tym tutorialu nie będą używane.

Jak dołączyć?

Podstawowym warunkiem używania Colab jest założenie konta w serwisie Google. Oprócz dziesiątek aplikacji sieciowych, właściciel takiego konta może dodać aplikację Colab. W momencie, kiedy już mamy konto Google (prawie każdy z telefonem android je ma, nie liczę osób z custom ROM :)) przechodzimy do aplikacji Google Drive i klikamy na nowy element –> Więcej i odnajdujemy Colaboratory na liście. To wszystko. Prawda, że proste?



Interfejs

Mamy panel z boku, który zbiera linki komórek markdown w spis treści. Dodatkowo wszystkie możliwości ipython notebook. Kmórki kodu, komórki mardown z pełna obsługą LaTeXa. Ewaluacja komórek to Shift+Enter.

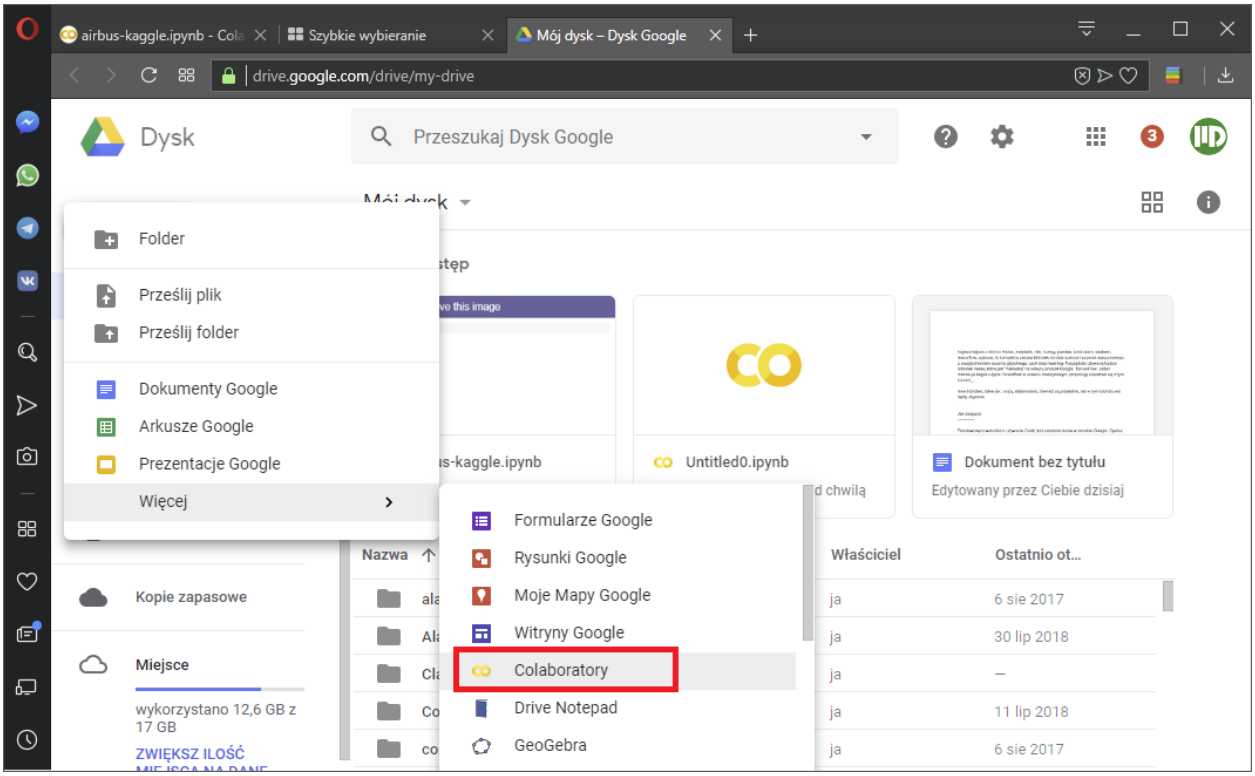
Akceleracja GPU

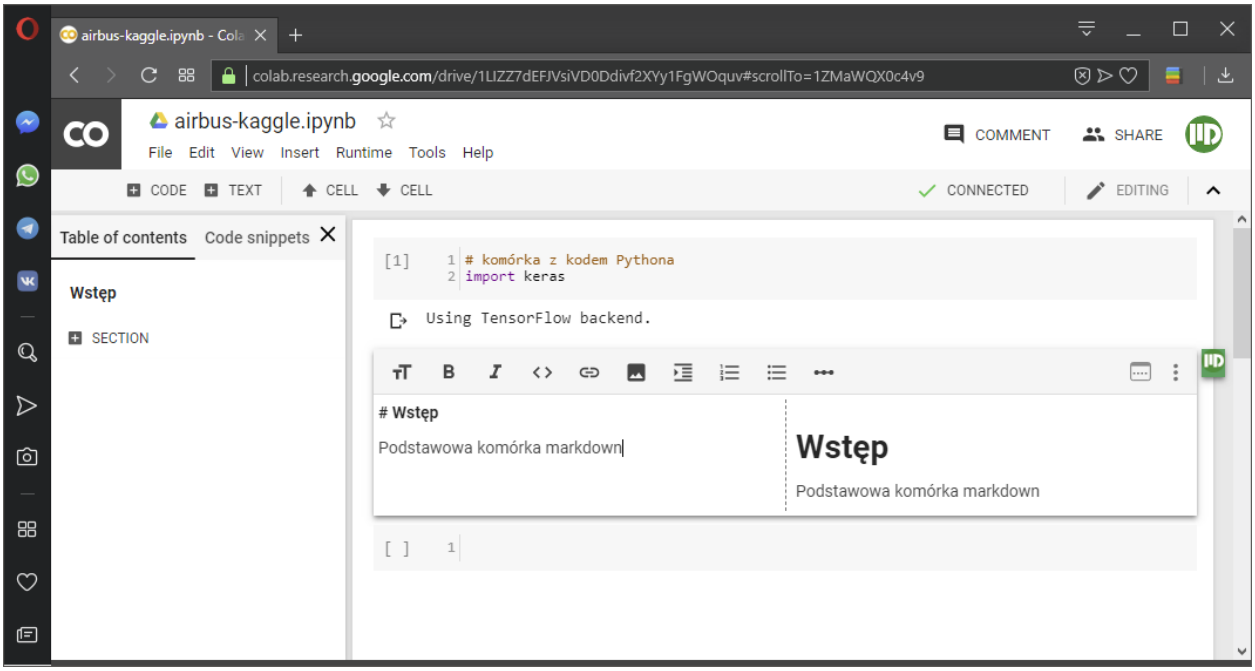
Aby móc skorzystać z GPU musimy najpierw aktywować tę opcję w ustawieniach notebooka. Przechodzimy do Edit –> Notebook Settings

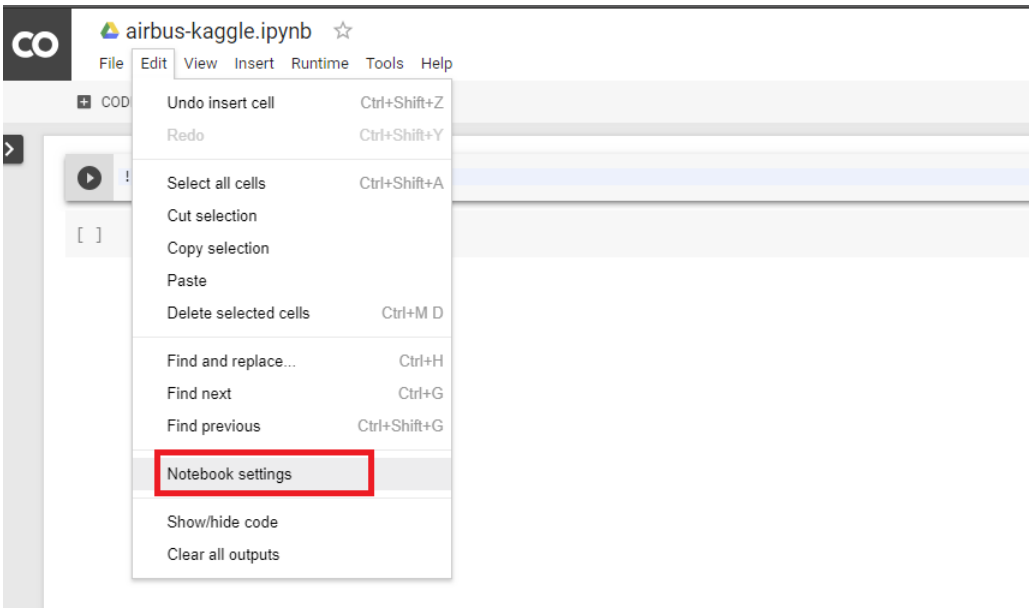
a następne zaznaczamy:

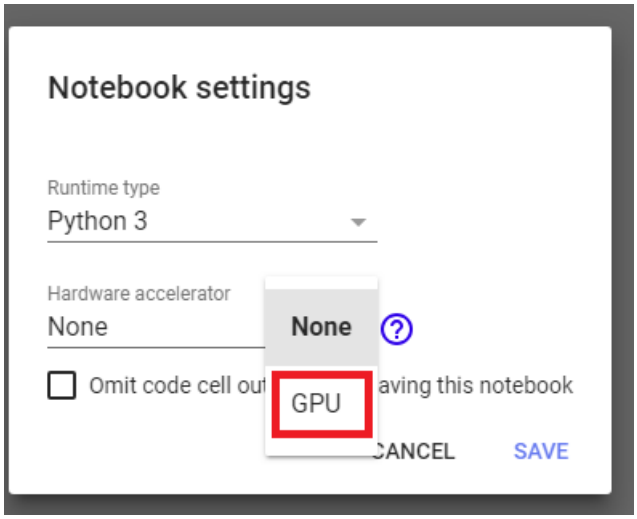
Gotowe!

Żeby sprawdzić, czy operacja się udała wpisujemy w wolną komórkę z kodem:


airbus-kaggle.ipynb - Cola
Szybkie wybieranie
Mój dysk – Dysk Google
drive.google.com/drive/my-drive
Dysk
Przeszukaj Dysk Google
Folder
Prześlij plik
Prześlij folder
Dokumenty Google
Arkusze Google
Prezentacje Google
Więcej
Formularze Google
Rysunki Google
Moje Mapy Google
Witryny Google
Colaboratory
Drive Notepad
GeoGebra
Kopie zapasowe
Miejsce
wykorzystano 12,6 GB z 17 GB
ZWIĘKSZ ILOŚĆ
Untitled0.ipynb
Dokument bez tytułu
Edytowany przez Ciebie dzisiaj
Nazwa
Właściciel
Ostatnio ot...
6 sie 2017
30 lip 2018
11 lip 2018


airbus-kaggle.ipynb - Cola
colab.research.google.com/drive/1LIZZ7dEFJVsiVD0Ddivf2XYy1FgWOquv#scrollTo=1ZMaWQX0c4v9
airbus-kaggle.ipynb
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE
TEXT
CELL
CELL
CONNECTED
EDITING
Table of contents
Code snippets
Wstęp
SECTION
[1] 1 # komórka z kodem Pythona
2 import keras
Using TensorFlow backend.
Wstęp
Podstawowa komórka markdown
Wstęp
Podstawowa komórka markdown
[ ] 1


airbus-kaggle.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Undo insert cell Ctrl+Shift+Z
Redo Ctrl+Shift+Y
Select all cells Ctrl+Shift+A
Cut selection
Copy selection
Paste
Delete selected cells Ctrl+M D
Find and replace... Ctrl+H
Find next Ctrl+G
Find previous Ctrl+Shift+G
Notebook settings
Show/hide code
Clear all outputs

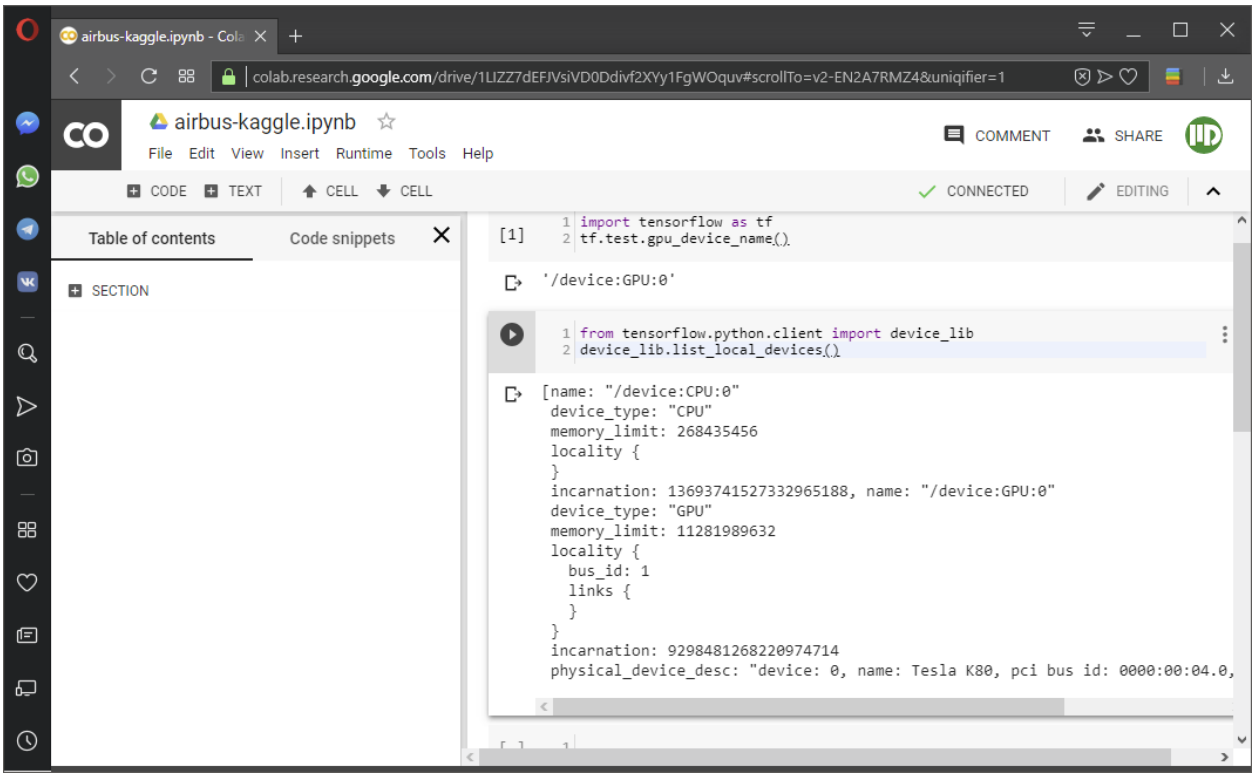

Notebook settings
Runtime type
Python 3
Hardware accelerator
None
None
GPU
Omit code cell out
aving this notebook
CANCEL
SAVE

```
import tensorflow as tf
tf.test.gpu_device_name()

'/device:GPU:0'
```

Powinniśmy uzyskać string z miejscem montowania GPU. Jeżeli pojawił sie pusty string, to oznacza, że aktywacja GPU nie powiodła się.

Aby dokładnie poznać parametry dostępnej karty graficznej, wpisujemy:

```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()

[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 13693741527332965188, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 11281989632
locality {
  bus_id: 1
  links {
  }
}
incarnation: 9298481268220974714
physical_device_desc: "device: 0, name: Tesla K80, pci bus id: 0000:00:04.0, compute
↪capability: 3.7"
```



1.2.2 Kaggle Notebook

Podobnie jak Google, portal Kaggle również udostępnia własną implementację Jupyter Notebook. Notebooki Kaggla od niedawna również umożliwiają użycie GPU do przyspieszenia obliczeń. Również jest to Nvidia Tesla K80. Kaggle Notebook oferuje również znaczną ilość pre-instalowanych paczek (524 paczki):

```
!pip list

Package                            Version                  Location
---------------------------------- ------------------------ -------------------------
↪----------
absl-py                            0.2.2
alabaster                          0.7.10
algopy                             0.5.7
altair                             2.1.0
anaconda-client                    1.6.5
anaconda-navigator                 1.6.9
anaconda-project                   0.8.0
annoy                              1.12.0
appdirs                            1.4.3
arrow                              0.12.1
asn1crypto                         0.22.0
astor                              0.7.1
astroid                            1.5.3
astropy                            2.0.2
attrs                              18.1.0
audioread                          2.1.6
Babel                              2.5.0
backports.shutil-get-terminal-size 1.0.0
Baker                              1.3
basemap                            1.1.0
bayesian-optimization              0.6.0
bayespy                            0.5.17
bcolz                              1.2.1
beautifulsoup4                     4.6.0
...
```

Co ciekawe Kaggle ma pre-instalowane paczki związane nie tylko z uczeniem maszynowym, ale również z wieloma specyficznymi dziedzinami takimi jak biologia obliczeniowa (biopython), astronomia (astropy). Z ciekawszych, które udało mi się zauważyć można wymienić cały zestaw paczek do map i GIS (basemap, Cartopy, Fiona, GDAL, geopandas etc.), paczki do przyspieszenia Pythona (numba, Cython), manipulacji danymi (odo, pandas, dask), wizualizacji danych (matplotlib, bokeh, plotly, seaborn, vega). Do wyboru, do koloru.

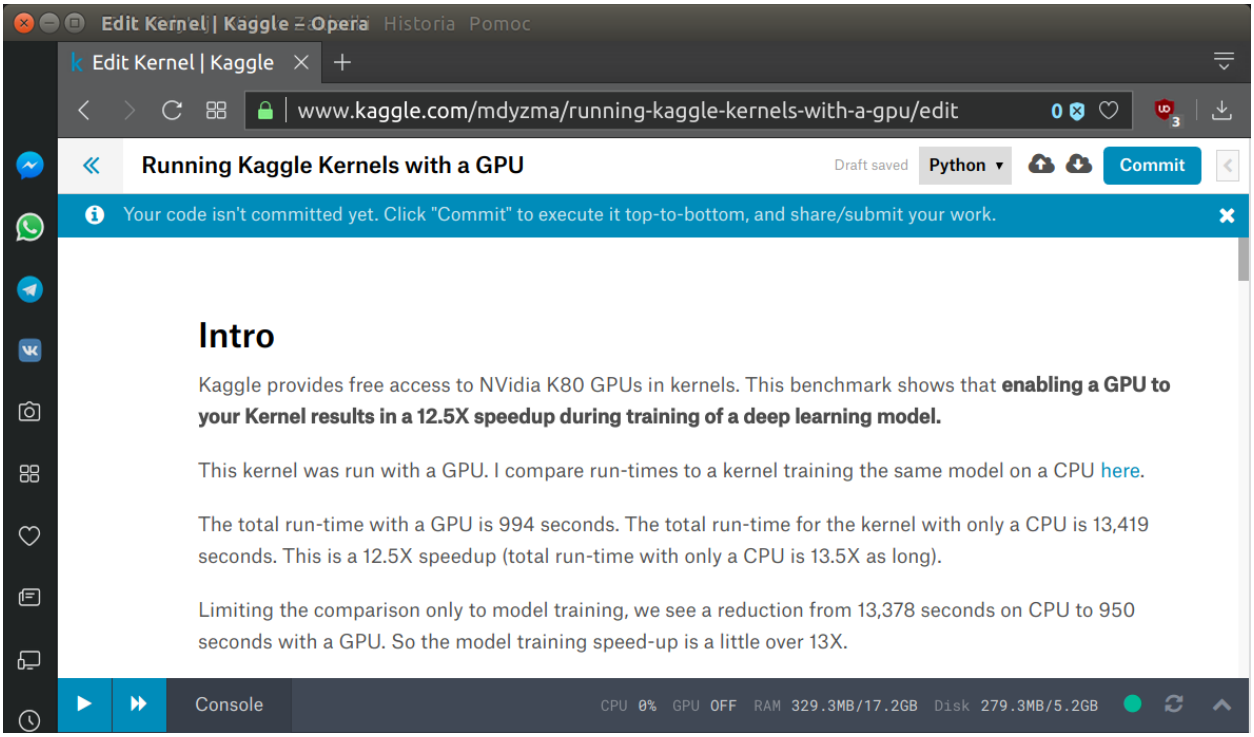
Interfejs

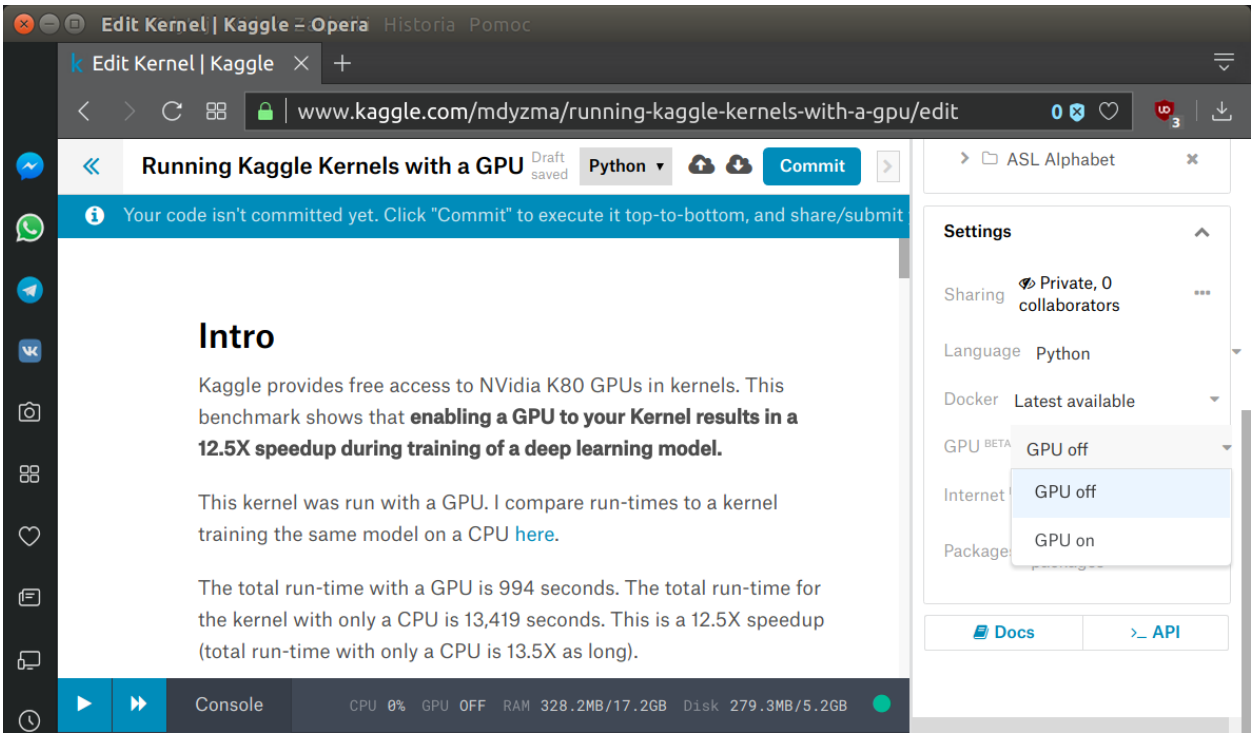
Interfejs jest prosty i bardzo podobny do colabowego. Działają nawet skróty z jupyter notebook.

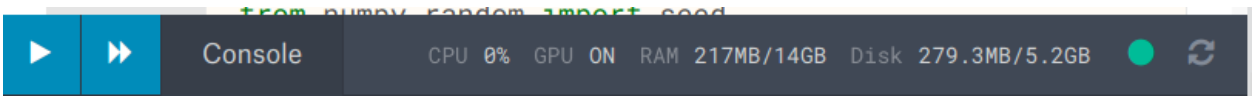
Akceleracja GPU

Jak aktywować jednostkę GPU? Podczas tworzenia nowego "kernela" (notebooka) otwieramy menu ustawień schowane pod prawym, bocznym panelem i przechodzimy do

Następnie wybieramy opcję **GPU on** i gotowe. Na dolnym pasku ze statusem naszego notebooka powinna się pojawić kontrolka "GPU ON".


Edit Kernel | Kaggle
www.kaggle.com/mdyzma/running-kaggle-kernels-with-a-gpu/edit
Running Kaggle Kernels with a GPU
Draft saved
Python
Commit
Your code isn't committed yet. Click "Commit" to execute it top-to-bottom, and share/submit your work.
Intro
Kaggle provides free access to NVidia K80 GPUs in kernels. This benchmark shows that enabling a GPU to your Kernel results in a 12.5X speedup during training of a deep learning model.
This kernel was run with a GPU. I compare run-times to a kernel training the same model on a CPU here.
The total run-time with a GPU is 994 seconds. The total run-time for the kernel with only a CPU is 13,419 seconds. This is a 12.5X speedup (total run-time with only a CPU is 13.5X as long).
Limiting the comparison only to model training, we see a reduction from 13,378 seconds on CPU to 950 seconds with a GPU. So the model training speed-up is a little over 13X.
Console
CPU 0% GPU OFF RAM 329.3MB/17.2GB Disk 279.3MB/5.2GB


Edit Kernel | Kaggle
www.kaggle.com/mdyzma/running-kaggle-kernels-with-a-gpu/edit
Running Kaggle Kernels with a GPU
Draft saved
Python
Commit
Your code isn't committed yet. Click "Commit" to execute it top-to-bottom, and share/submit
Intro
Kaggle provides free access to NVidia K80 GPUs in kernels. This benchmark shows that enabling a GPU to your Kernel results in a 12.5X speedup during training of a deep learning model.
This kernel was run with a GPU. I compare run-times to a kernel training the same model on a CPU here.
The total run-time with a GPU is 994 seconds. The total run-time for the kernel with only a CPU is 13,419 seconds. This is a 12.5X speedup (total run-time with only a CPU is 13.5X as long).
Console
CPU 0% GPU OFF RAM 328.2MB/17.2GB Disk 279.3MB/5.2GB
ASL Alphabet
Settings
Sharing
Private, 0 collaborators
Language
Python
Docker
Latest available
GPU BETA
GPU off
GPU off
GPU on
Internet
Package
Docs
API


Console
CPU 0% GPU ON RAM 217MB/14GB Disk 279.3MB/5.2GB

1.2.3 Podsumowanie

Jak wspomniałem oba rozwiązania mają swoje plusy i minusy. Pokrótce:

Cecha	**Kaggle**	**Colab**
Zauploadowane dane	tak	nie
Możliwość współpracy	nie (??)	tak
GPU	tak	tak
Preinstalowane biblioteki	tak(++)	tak

Ponieważ nasza grupa nastawiona jest na edukację razem, moim zdaniem opcja współpracy Google Colab przeważa i warto zainwestować trochę czasu w upload próbki zdjęć na konto Google i móc dzielić Notebooka z całą grupą.

1.2.4 Ważne linki

Nazwa	URL
Colab Welcome	https://colab.research.google.com/notebooks/welcome.ipynb
Colab GPU	https://colab.research.google.com/notebooks/gpu.ipynb
Google ML Kurs	https://developers.google.com/machine-learning/crash-course/ml-intro
Colab Tutorial	https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d
Kaggle GPU	https://www.kaggle.com/dansbecker/running-kaggle-kernels-with-a-gpu

1.3 Pre-processing danych

1.3.1 Wczytywanie danych

1.3.2 Wizualizacje

Obrazki na wykresach

Wyświetlanie kanałów RGB

Histogramy

1.3.3 Modyfikacje (augmentacja)

Translacja

Skalowanie

Rotacja

Odbicie

Manipulacje oświetleniem

Manipulacje paletami barw

1.3.4 Ważne linki

Nazwa	URL
Tutorial	https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9

1.4 Sieci Neuronowe

1.4.1 Podstawowe zagadnienia

Overfitting

Underfitting

Backpropagation

Warstwy

Neuron

Liniowy

ReLU

Sigmoidalny

Transfer learning

Co to jest sieć neuronowa

Rodzaje sieci neuronowych

1.4.2 Miary jakości

Miara jakości uczenia

Miara klasyfikacji

Dokładność (Accuracy)

Confusion matrics

1.4.3 Konwolucyjne Sieci Neuronowe (CNN / ConvNets)

Warstwy Konwolucyjne

Warstwy agregująca (pooling)

W pełni połączone

1.4.4 Ważne linki

Nazwa	URL

CHAPTER 2

Indices and tables

- genindex
- modindex
- search