How To Run CUDA C or C++ on Google Colab.

medium.com/@p190036/how-to-run-cuda-c-or-c-on-google-colab-c1e5d342f65f

Muhammad Abdullah April 9, 2022

Muhammad Abdullah

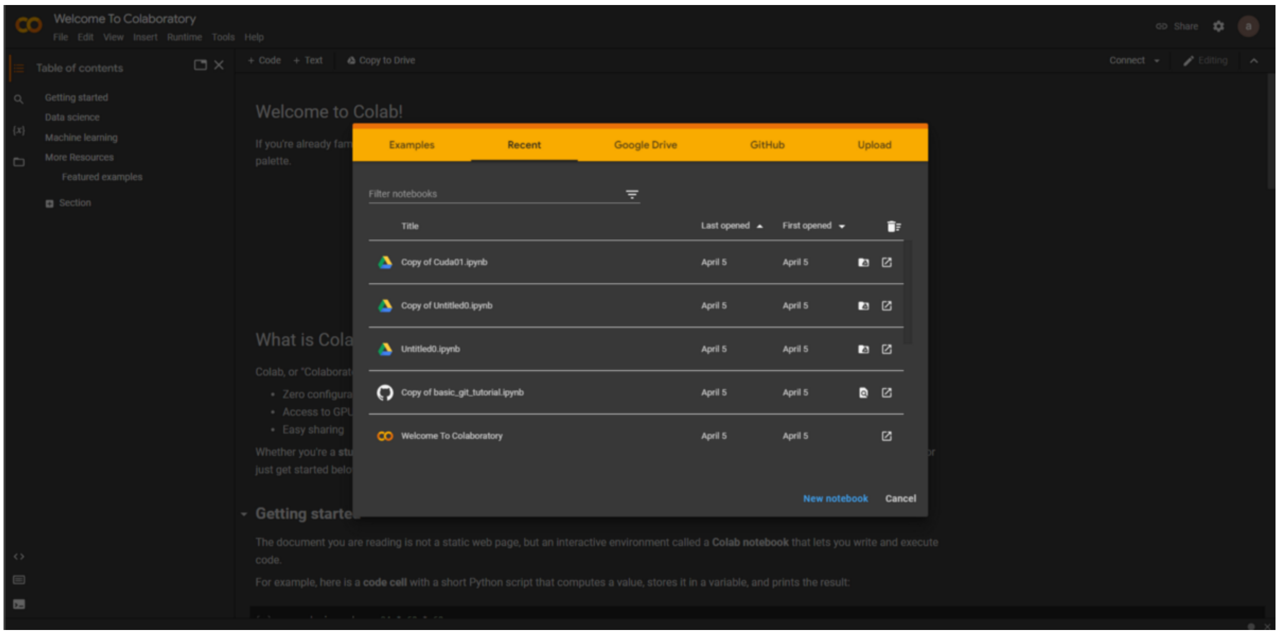
Apr 7

·

3 min read

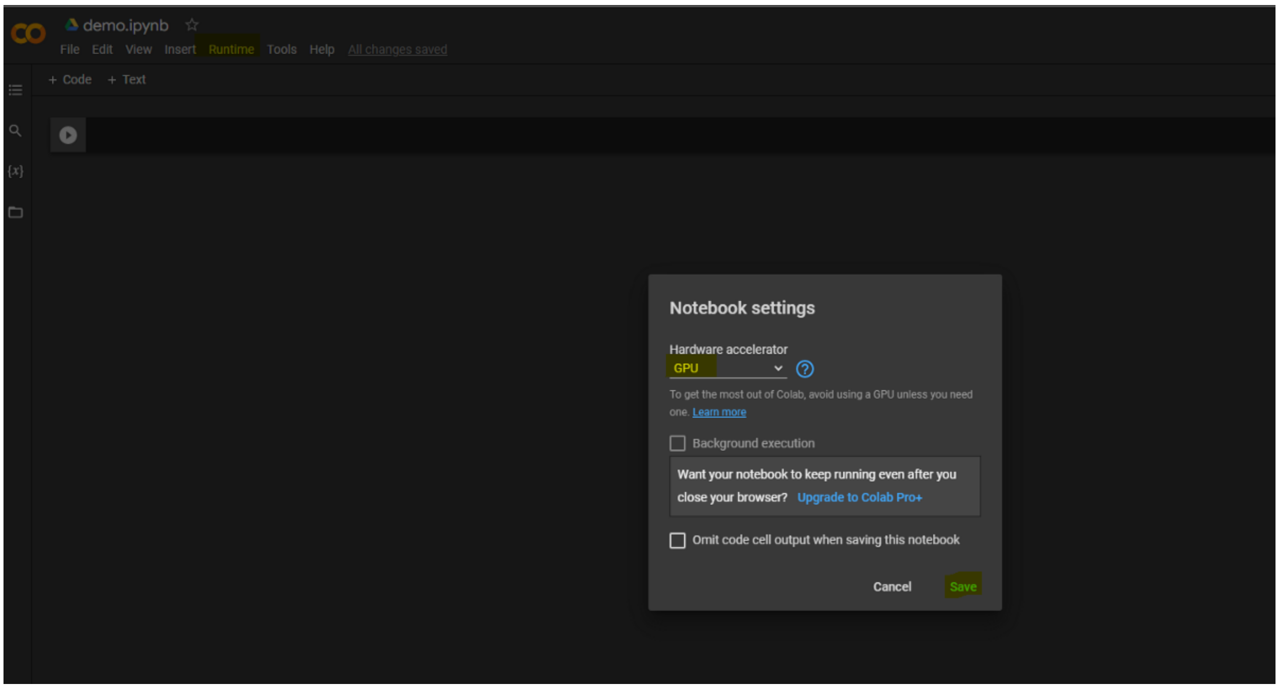
CUDA code doesn't run on AMD CPU or Intel HD graphics unless you have a NVIDIA Hardware inside your Machine. If you're interested in running CUDA on NVIDIA hardware you can check the following article: How To Run CUDA C or C++ on Microsoft Visual Studio. | by Muhammad Abdullah | Apr, 2022 | Medium

Step 1: Go to https://colab.research.google.com in Browser and Click on New Python 3 Notebook



Title	Last opened	First opened
Copy of Cuda01.ipynb	April 5	April 5
Copy of Untitled0.ipynb	April 5	April 5
Untitled0.ipynb	April 5	April 5
Copy of basic_git_tutorial.ipynb	April 5	April 5
Welcome To Colaboratory	April 5	April 5

Step 2: Click to Runtime > Change > Hardware Accelerator GPU .



Notebook settings

Hardware accelerator: GPU

To get the most out of Colab, avoid using a GPU unless you need one. Learn more

☐ Background execution

Want your notebook to keep running even after you close your browser? Upgrade to Colab Pro+

☐ Omit code cell output when saving this notebook

To check which GPU you're using, run the following command.

!nvidia-smi

Step 3: Refresh the Cloud Instance of CUDA On Server [write code in a Seprate code Block and Run that]

```
!apt-get — purge remove cuda nvidia* libnvidia-*
!dpkg -l | grep cuda- | awk '{print $2}' | xargs -n1 dpkg — purge
!apt-get remove cuda-*
!apt autoremove
!apt-get update
```

Step 4: Install CUDA Version 9 [write code in a Seprate code Block and Run that]

```
!wget https: -O cuda-repo-ubuntu1604–9–2-local_9.2.88–1_amd64.deb
!dpkg -i cuda-repo-ubuntu1604–9–2-local_9.2.88–1_amd64.deb
!apt-key add /var/cuda-repo-9–2-local/7fa2af80.pub
!apt-get update
!apt-get install cuda-9.2
```

Step 5: Check the Version of CUDA by : running the command below to get the following output :

```
!nvcc — version
```

Output

```
nvcc: NVIDIA (R) Cuda compiler driver Copyright © 2005–2020 NVIDIA Corporation Built on Mon_Oct_12_20:09:46_PDT_2020 Cuda compilation tools, release 11.1, V11.1.105 Build cuda_11.1.TC455_06.29190527_0
```

Step 6: Execute the given command to install a small extension to run nvcc from Notebook cells [write code in a Seprate code Block and Run that]

```
!pip install git+https://github.com/andreinechaev/nvcc4jupyter.git
```

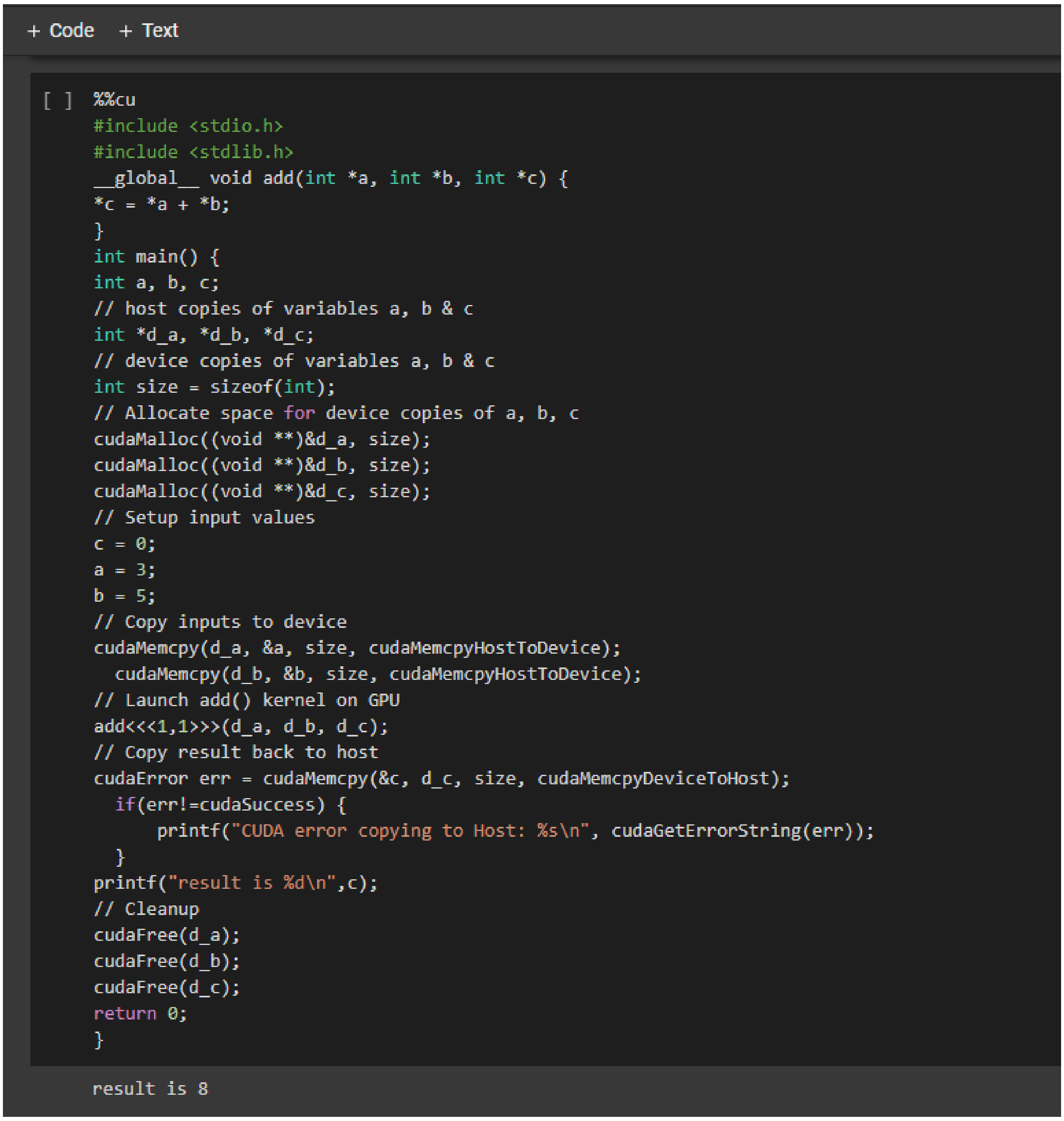
Step 7: Load the extension using this code:[write code in a Seprate code Block and Run that]

```
%load_ext nvcc_plugin
```

Important : To check the Code run the following snippet in [write code in a Seprate code Block and Run that only again when changing the code and re running it]. Also to run cuda programs you need to add %%cu at the start of your code.

```
%%cu
#include <stdio.h>
#include <stdlib.h>
__global__ void add(int *a, int *b, int *c) {
*c = *a + *b;
}
int main() {
int a, b, c;
// host copies of variables a, b & c
int *d_a, *d_b, *d_c;
// device copies of variables a, b & c
int size = sizeof(int);
// Allocate space for device copies of a, b, c
cudaMalloc((void **)&d_a, size);
cudaMalloc((void **)&d_b, size);
cudaMalloc((void **)&d_c, size);
// Setup input values
c = 0;
a = 3;
b = 5;
// Copy inputs to device
cudaMemcpy(d_a, &a, size, cudaMemcpyHostToDevice);
cudaMemcpy(d_b, &b, size, cudaMemcpyHostToDevice);
// Launch add() kernel on GPU
add<<<1,1>>>(d_a, d_b, d_c);
// Copy result back to host
cudaError err = cudaMemcpy(&c, d_c, size, cudaMemcpyDeviceToHost);
if(err!=cudaSuccess) {
printf(“CUDA error copying to Host: %s\n”, cudaGetErrorString(err));
}
printf(“result is %d\n”,c);
// Cleanup
cudaFree(d_a);
cudaFree(d_b);
cudaFree(d_c);
return 0;
}
```

The Ouptut should be 8



```
%%cu
#include <stdio.h>
#include <stdlib.h>
__global__ void add(int *a, int *b, int *c) {
*c = *a + *b;
}
int main() {
int a, b, c;
// host copies of variables a, b & c
int *d_a, *d_b, *d_c;
// device copies of variables a, b & c
int size = sizeof(int);
// Allocate space for device copies of a, b, c
cudaMalloc((void **)&d_a, size);
cudaMalloc((void **)&d_b, size);
cudaMalloc((void **)&d_c, size);
// Setup input values
c = 0;
a = 3;
b = 5;
// Copy inputs to device
cudaMemcpy(d_a, &a, size, cudaMemcpyHostToDevice);
  cudaMemcpy(d_b, &b, size, cudaMemcpyHostToDevice);
// Launch add() kernel on GPU
add<<<1,1>>>(d_a, d_b, d_c);
// Copy result back to host
cudaError err = cudaMemcpy(&c, d_c, size, cudaMemcpyDeviceToHost);
  if(err!=cudaSuccess) {
      printf("CUDA error copying to Host: %s\n", cudaGetErrorString(err));
  }
printf("result is %d\n",c);
// Cleanup
cudaFree(d_a);
cudaFree(d_b);
cudaFree(d_c);
return 0;
}
```

```
result is 8
```

For the next time you just have to run the following two commands(**Step 6** & **Step 7**)

!pip install git+https://github.com/andreinechaev/nvcc4jupyter.git

%load_ext nvcc_plugin

If you're interested in more examples of CUDA code you can check them on the following link NVIDIA/cuda-samples: Samples for CUDA Developers which demonstrates features in CUDA Toolkit (github.com)

Refernces :

Google Colab — The Beginner's Guide | by Vishakha Lall | Lean In Women In Tech India | Medium

How to Use Google Colab for Deep Learning — Complete Tutorial — neptune.ai

How To Run CUDA C or C++ on Google Colab or Azure Notebook | by Harshit Yadav | Medium