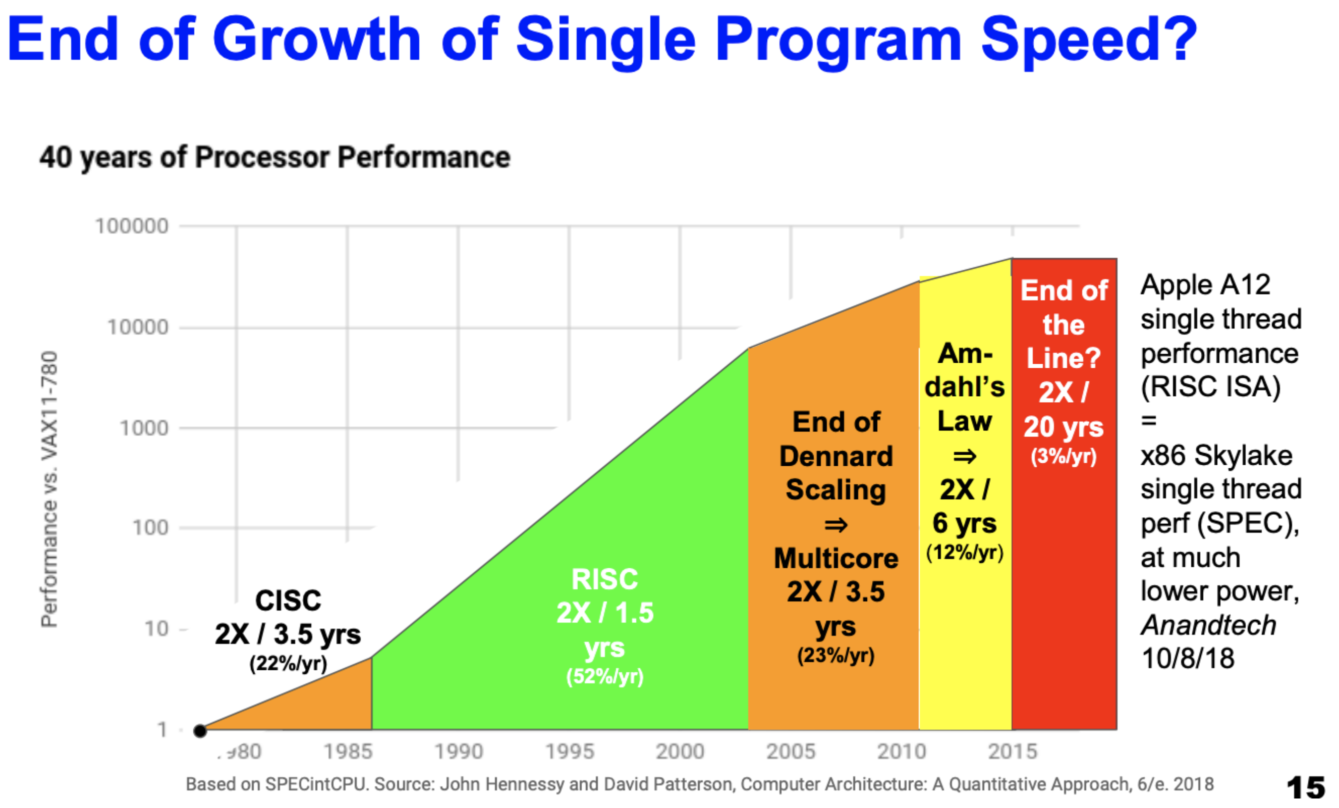
End of Moore’s Law

End of Moore’s Law

Perhaps the best source to describe what is happening with chips is Dr. David Patterson a professor at UC Berkeley and an architect for the TPU (Tensorflow Processing Unit).



End of Growth of Single Program Speed?

40 years of Processor Performance

Apple A12 single thread performance (RISC ISA) = x86 Skylake single thread perf (SPEC), at much lower power, *Anandtech* 10/8/18

Based on SPECintCPU. Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6/e. 2018

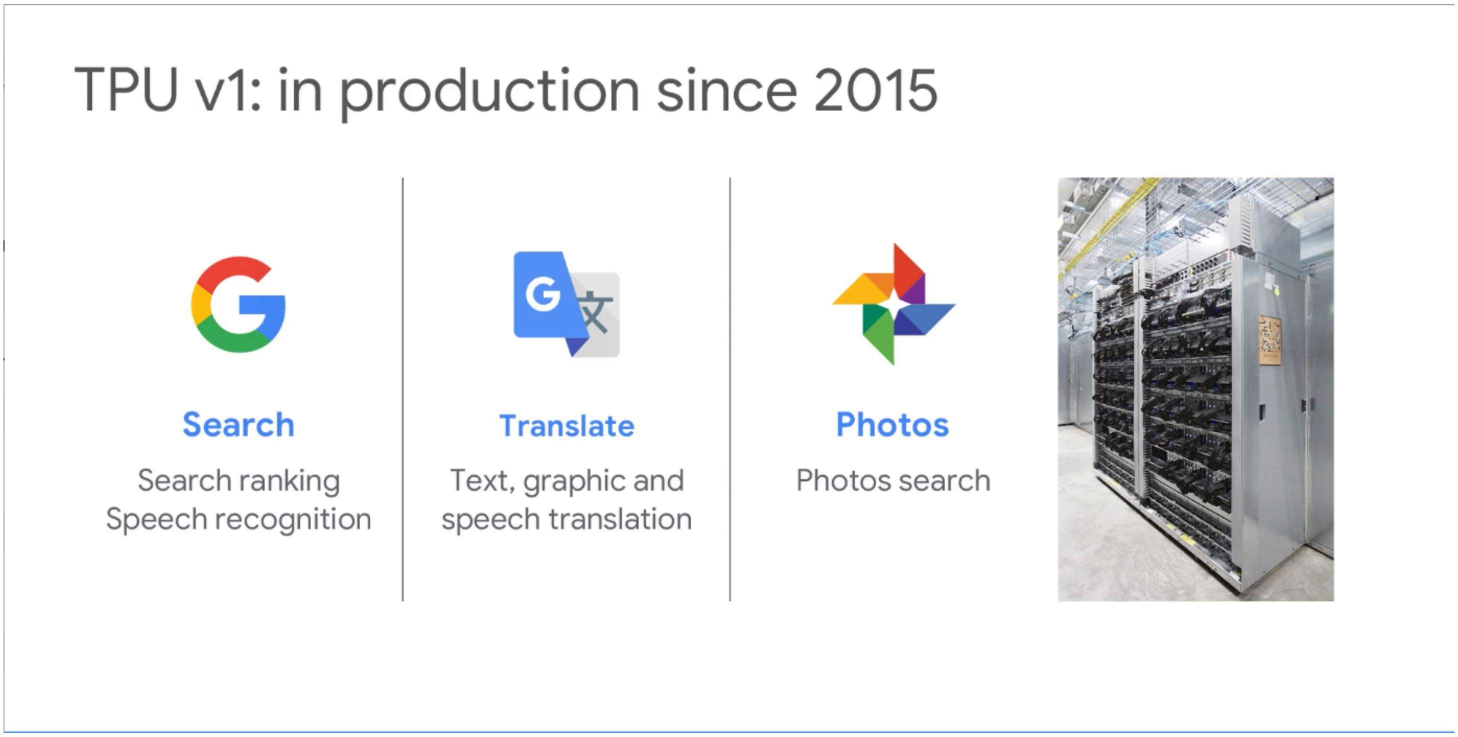
15

The high-level version is that CPU clock speed is effectively dead. This opens up an opportunity for new solutions. These solutions involve both cloud computing and also specialized chips called ASICs.

ASICS: GPUs, TPUs, FPGA

ASIC vs CPU vs GPU

TPU in Production



TPU v1: in production since 2015

Search

Search ranking
Speech recognition

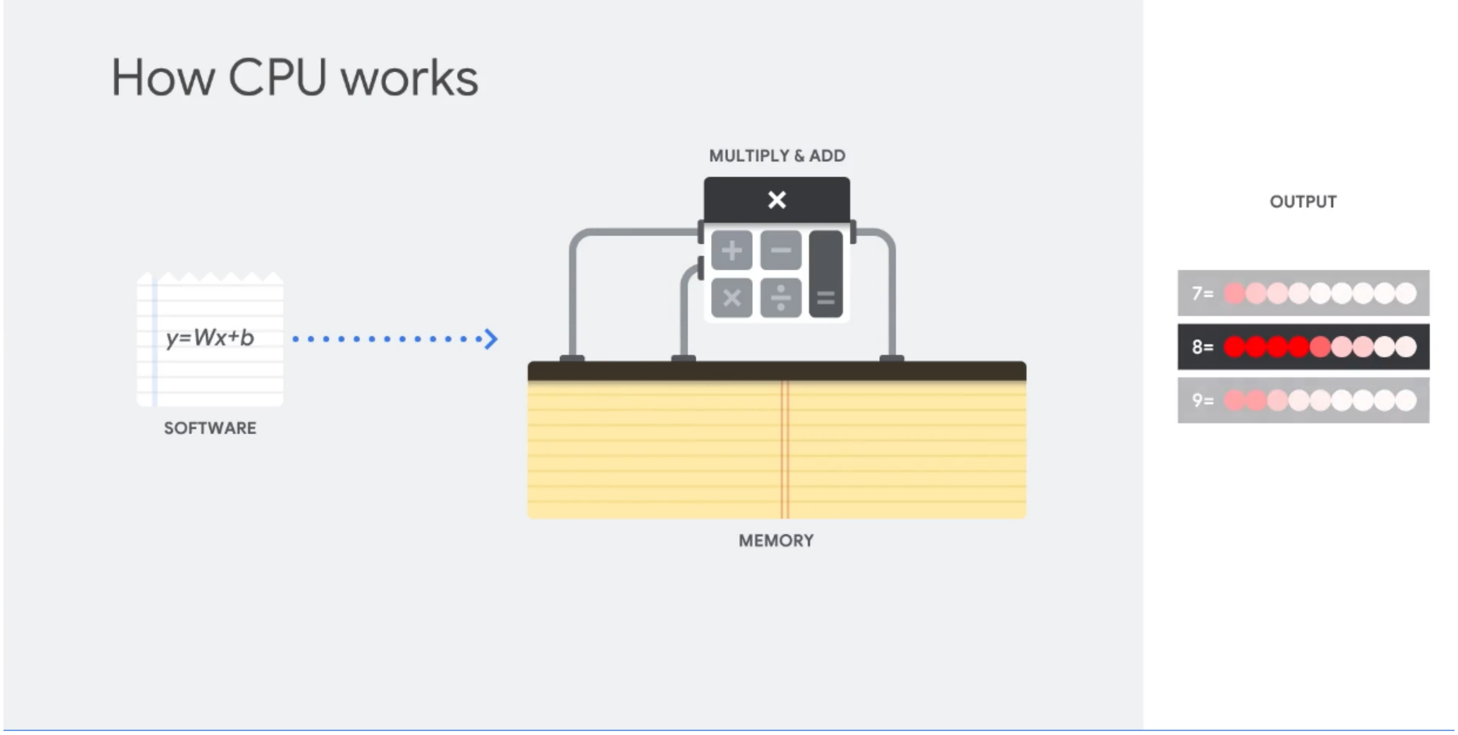
Translate

Text, graphic and
speech translation

Photos

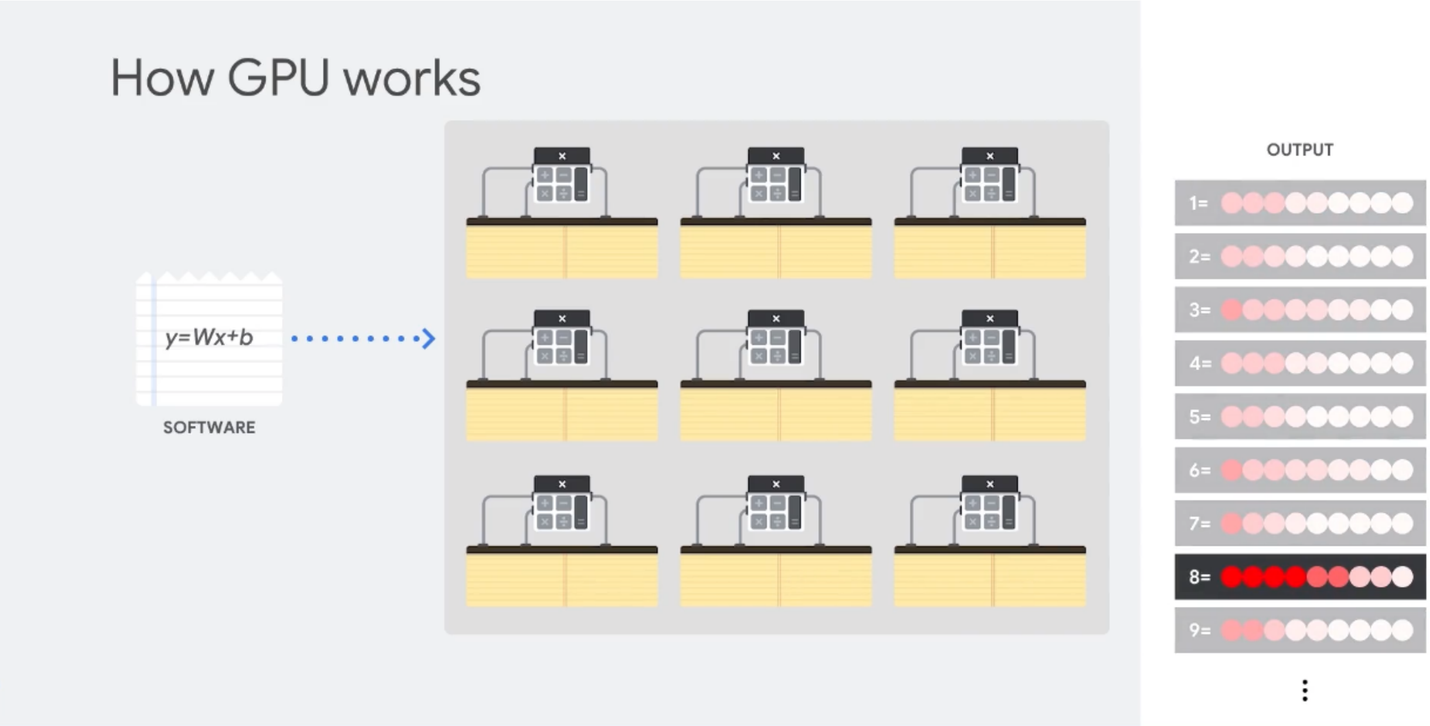
Photos search

CPU



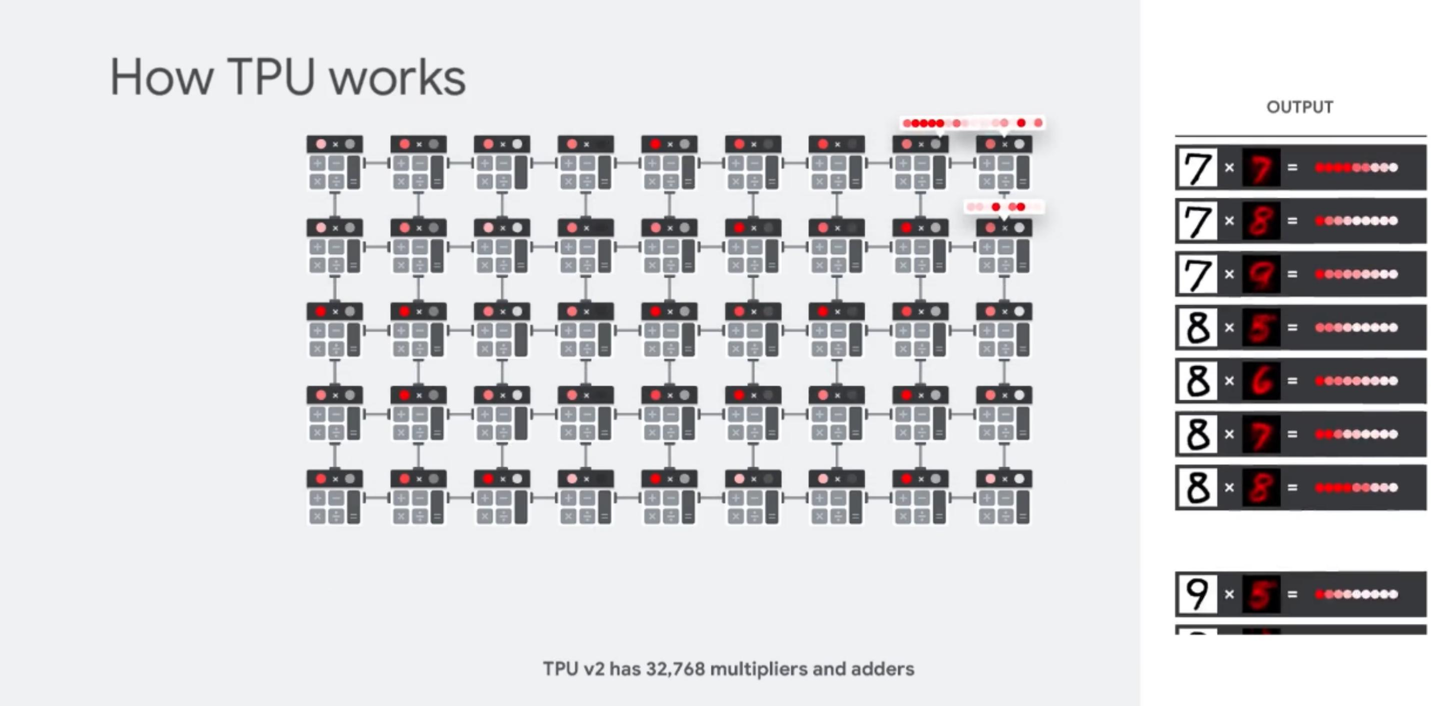
How CPU works

GPU



How GPU works

TPU



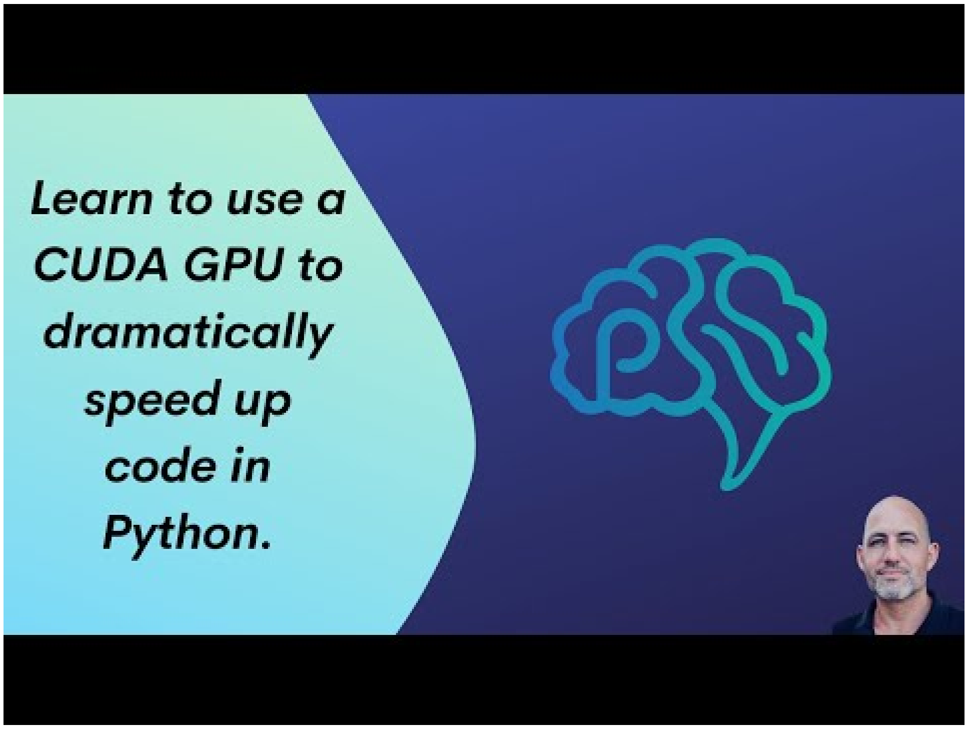
How TPU works

TPU v2 has 32,768 multipliers and adders

Sources: https://storage.googleapis.com/nexttpu/index.html

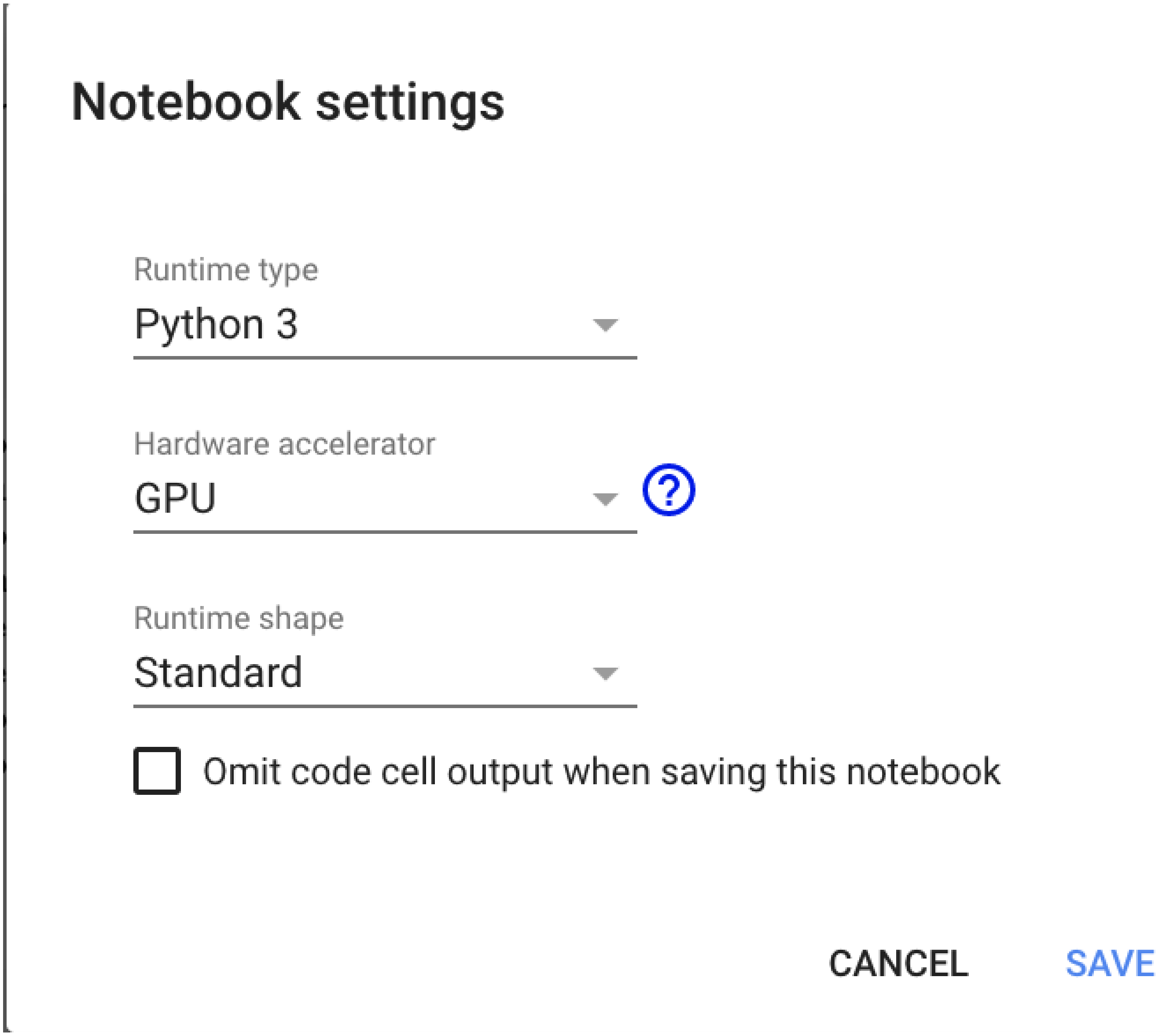
Using GPUs and JIT

Screencast


Learn to use a CUDA GPU to dramatically speed up code in Python.

One of the easiest ways to use a Just in Time compiler (JIT) or a GPU is to use a library like numba and a hosted runtime like Google Colab.

There is a step by step example of how to use these operations in the following notebook (https://github.com/noahgift/cloud-data-analysis-at-scale/blob/master/GPU_Programming.ipynb) [https://github.com/noahgift/cloud-data-analysis-at-scale/blob/master/GPU_Programming.ipynb]. The main high level takeaway is that the GPU runtime in colab must be enabled.



Notebook settings

Runtime type
Python 3

Hardware accelerator
GPU

Runtime shape
Standard

□ Omit code cell output when saving this notebook

Next up install `numba` and double check the CUDA `.so` libraries are available.

```
!pip install numba
!find / -iname 'libdevice'
!find / -iname 'libnvvm.so'
```

You should see something like this.

```
/usr/local/cuda-10.0/nvvm/libdevice
/usr/local/cuda-10.1/nvvm/libdevice
/usr/local/cuda-10.0/nvvm/lib64/libnvvm.so
/usr/local/cuda-10.1/nvvm/lib64/libnvvm.so
```

Next up try one of the methods for speeding up your code.

GPU Workflow

[TO DO: Create GPU vectorize workflow diagram]

Exercise

- Topic: Go through colab example here
- Estimated time: 20-30 minutes
- People: Individual or Final Project Team
- Slack Channel: #noisy-exercise-chatter
- Directions:
  - Part A: Get code working in colab
  - Part B: Make your own GPU or JIT code to speed up a project you are working on. Share in slack and/or create a technical blog post about it.