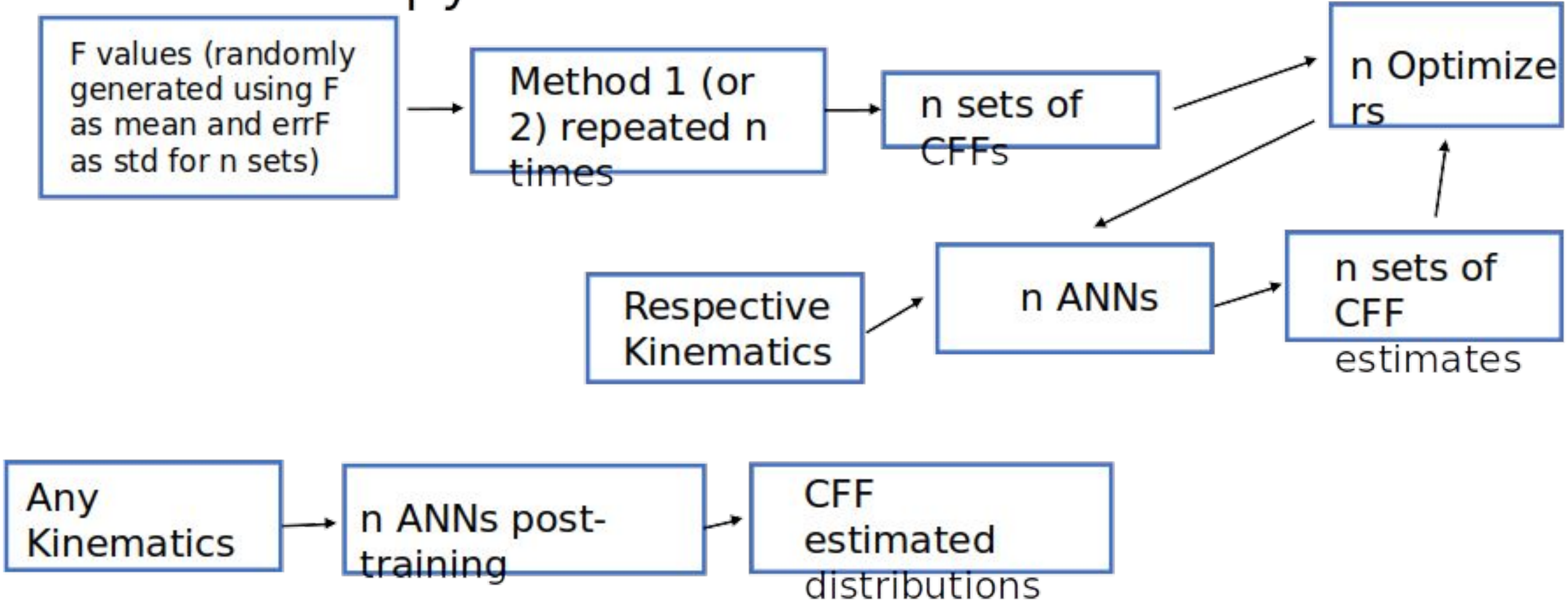
Tensorflow with GPU and CPU

Arthur Conover 7/16/21

Using Method 4 in Nick’s GitHub folder: (refresher)

- Kinematic sets are fitted locally to find CFFs, and then those CFFs are predicted using an ANN with Kinematics as inputs. An implementation can be found in "Method4.ipynb"



F values (randomly generated using F as mean and errF as std for n sets)

Method 1 (or 2) repeated n times

n sets of CFFs

n Optimizers

Respective Kinematics

n ANNs

n sets of CFF estimates

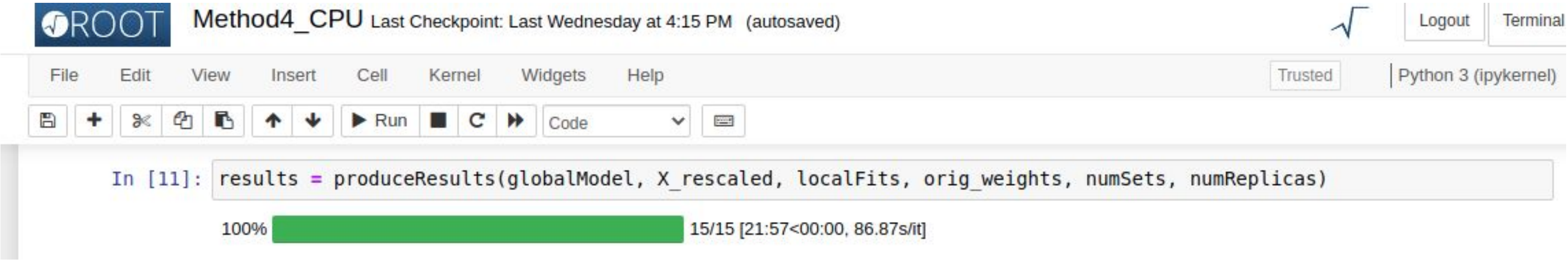
Any Kinematics

n ANNs post-training

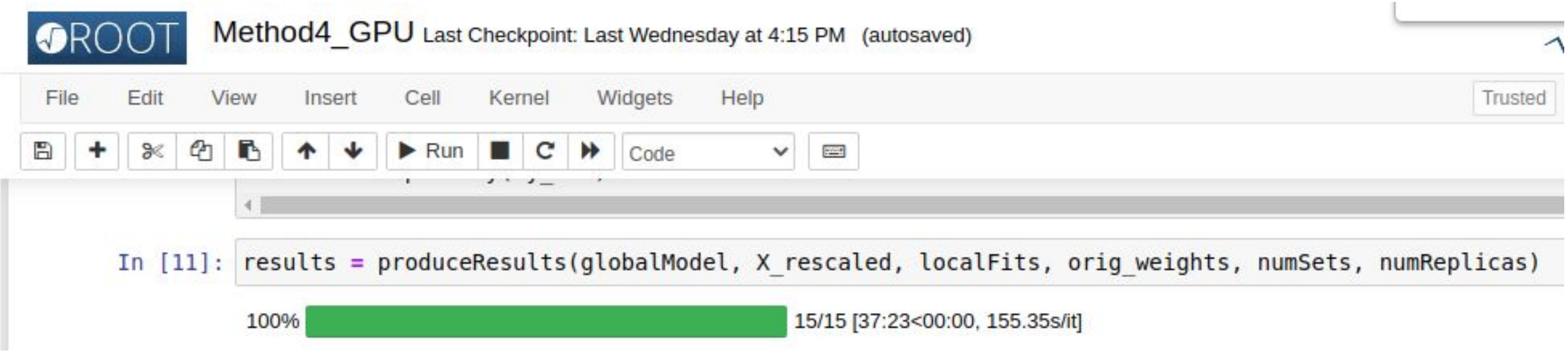
CFF estimated distributions

When CUDA is installed, GPU used automatically

This isn’t necessarily quicker:

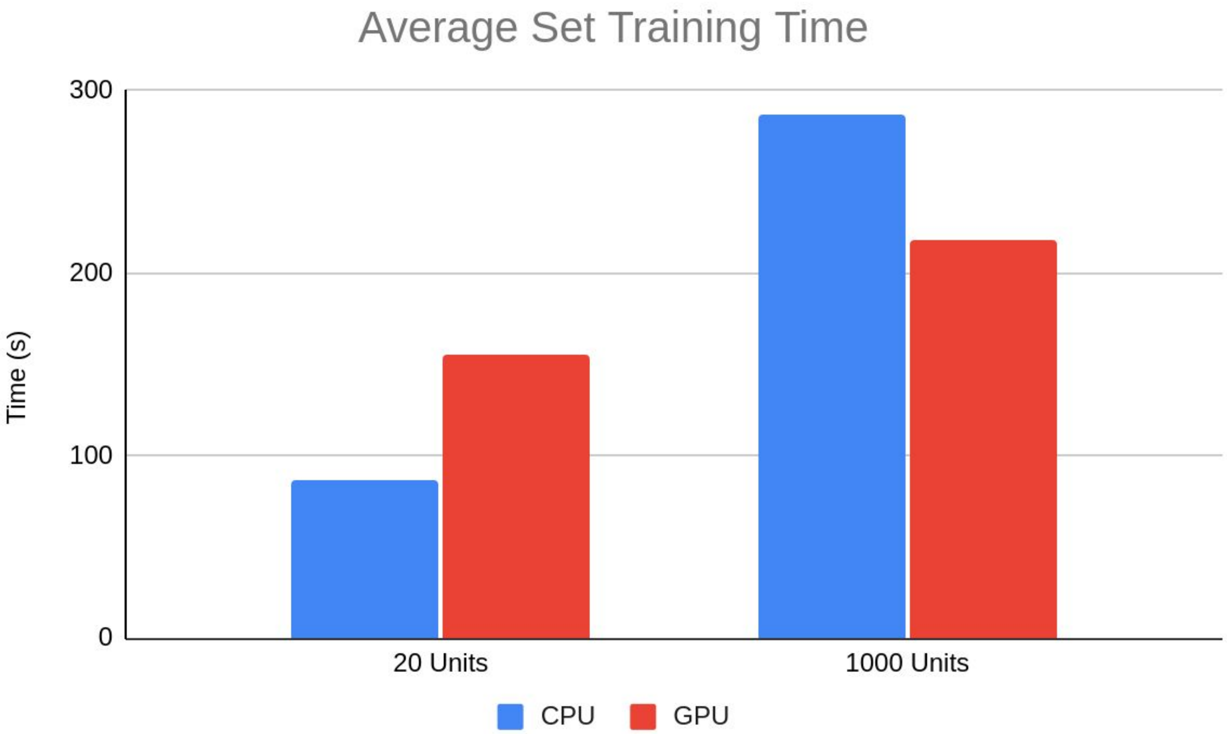


```
In [11]: results = produceResults(globalModel, X_rescaled, localFits, orig_weights, numSets, numReplicas)

         100%  15/15 [21:57<00:00, 86.87s/it]
```


ROOT
Method4_GPU Last Checkpoint: Last Wednesday at 4:15 PM (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Run
Code

```
In [11]: results = produceResults(globalModel, X_rescaled, localFits, orig_weights, numSets, numReplicas)

         100%  15/15 [37:23<00:00, 155.35s/it]
```

More complex ANNs benefit from GPU



Didn’t have time to make a plot

Adding additional layers also makes GPU have an advantage over CPU.

In summary, if GPU enabled, make sure to switch to CPU for simple ANNs but use GPU for more complicated ANNs.

This is done by adding this code at the beginning of the the script:

```
import os

os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
```