Multi-dimensional re-weighting via Deep Set Neural Networks

NPPS joint meeting 07th May 2021

Stephen Jiggins, Fang-Ying Tsai

Project Participants for the VH(bb) analysis and Google-ATLAS R&D

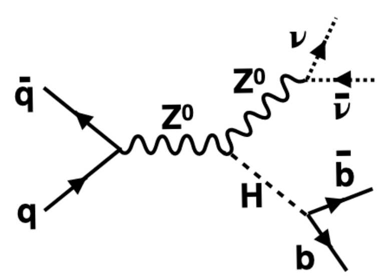
John Hobbs, Giacinto Piacquadio,
Torre Wenaus, Alexei Klimentov

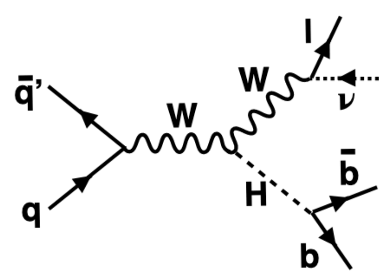
Outline

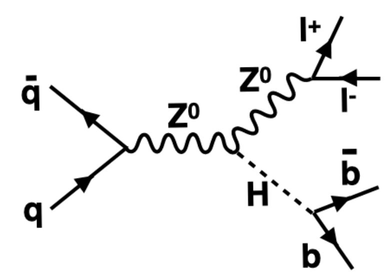
- Deep Set Neural Networks framework.
  - Motivations
  - The DSNN model
  - Closure checks
- Computing Performance
  - CPU vs GPU
  - Memory
- Next Steps

Deep Sets Neural Network reweighting (DSNNr)

- Why are we interested in DSNNr?
  - Looking for a CPU/GPUs intensive ML task for the US ATLAS Google project as a use case.
  - Ultimately we want to achieve high utilization of using CPU/GPUs for ML work.
  - We can have a generic classification for VH(bb) as well as VH(cc) analyses in both boosted and resolved regimes.
    → generate a mapping function between two MC configurations that is independent of the reconstruction scheme.

0-lep


1-lep


2-lep*


- Our framework is built based on the ParticleFlow Neural Network.
  - It’s the application of the algorithm.
  - It's a fresh technique in the analyses/ATLAS.

Deep Sets Neural Network reweighting (DSNNr)

- DSNN architecture (ref.)

$$f(\{x_1, x_2 ... x_M\}) = F(\Sigma_{i=1}^{M} \Phi(x_i))$$

- Permutation invariance sets. The NN will learn the same when permuting the input objects.

- Φ: to embed datasets into a vector space from x elements → R^ℓ

$$\{x_1, x_2 ... x_M\} \xrightarrow{\text{mapping through a NN}} \begin{bmatrix} \text{obj}_1 \\ \text{obj}_2 \\ ... \\ \text{obj}_\ell \end{bmatrix}$$

- Adding up all particle representations in multi-dimensional space.

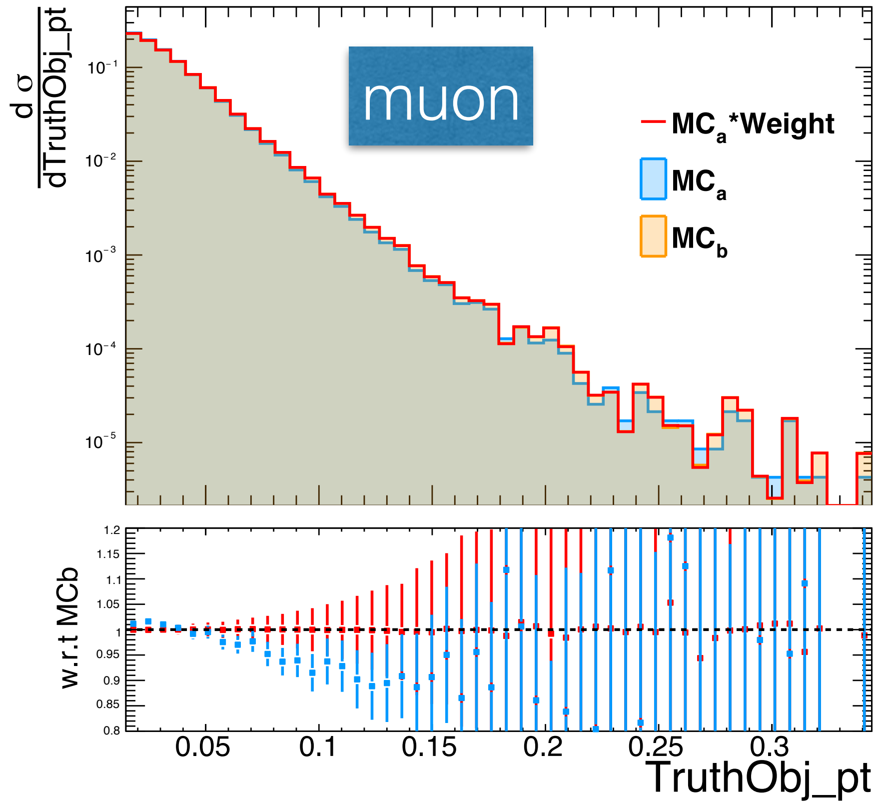
- F: applying a nonlinear transformation yielding event representations from R^ℓ elements → Y

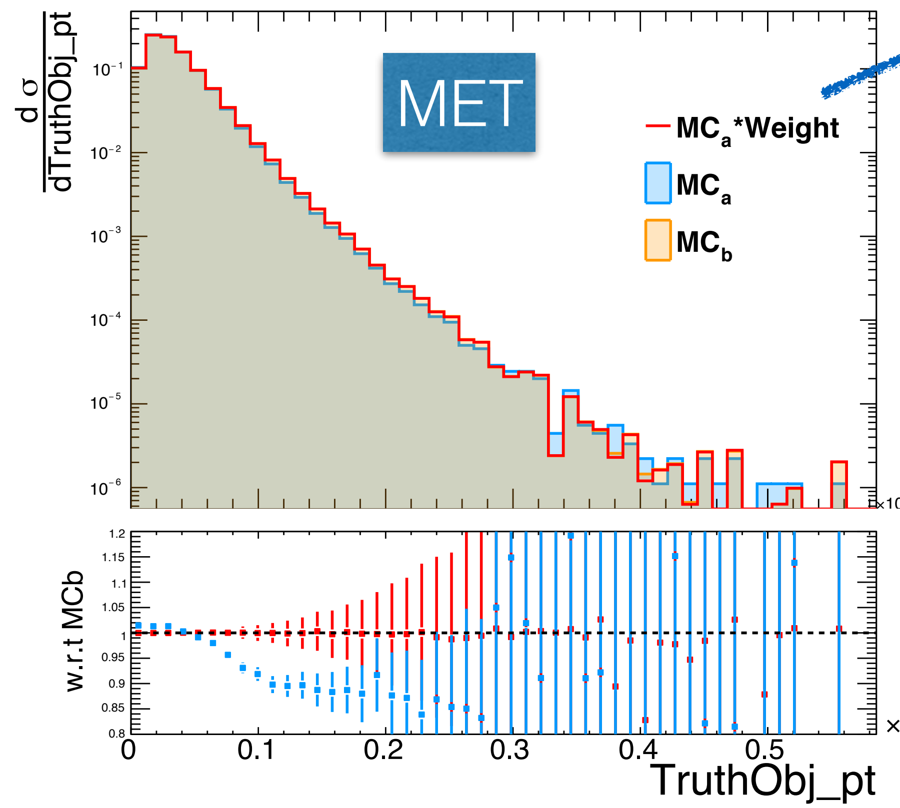
$$\begin{bmatrix} \text{obj}_1 \\ \text{obj}_2 \\ ... \\ \text{obj}_\ell \end{bmatrix} \xrightarrow{\text{The choice of the distributions through a NN}} \text{F\_signal, F\_background}$$

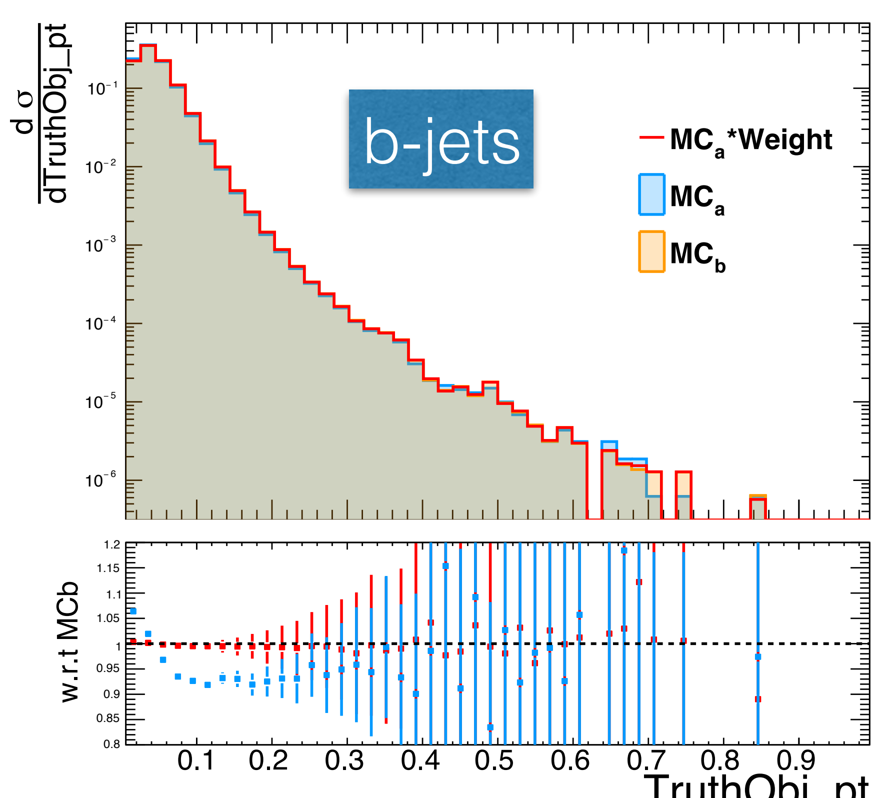
DSNNr Features

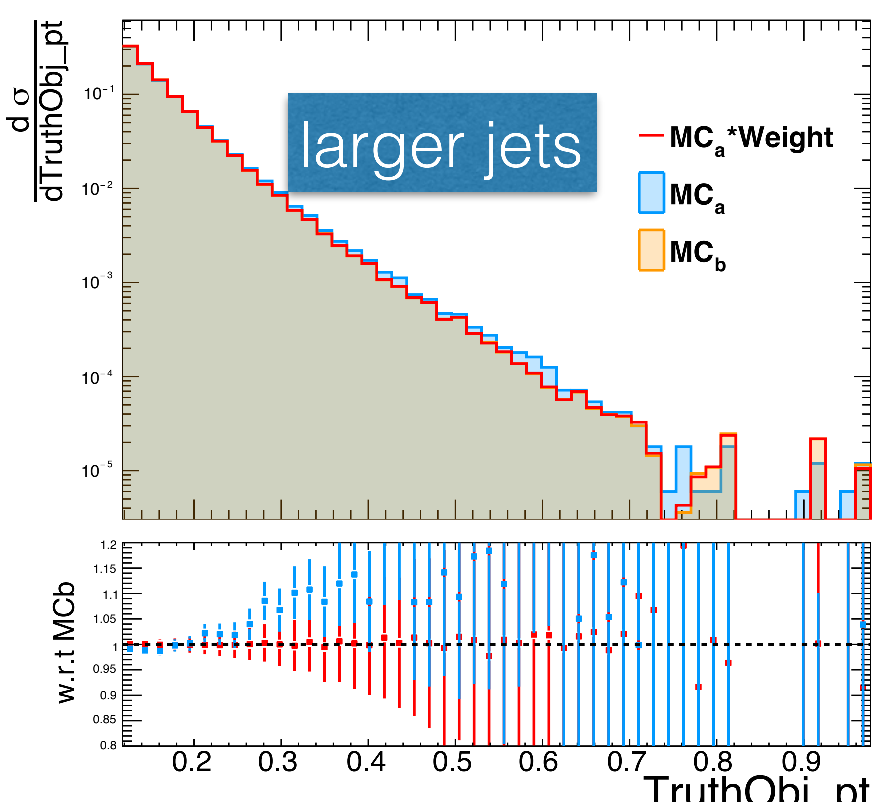
- Sanity checks: MCa v.s. MCa x dR(bb) weights. (full features, pt, eta phi, mass, see <u>here</u>.)

Weight = Prob_MCb/Prob_MCa





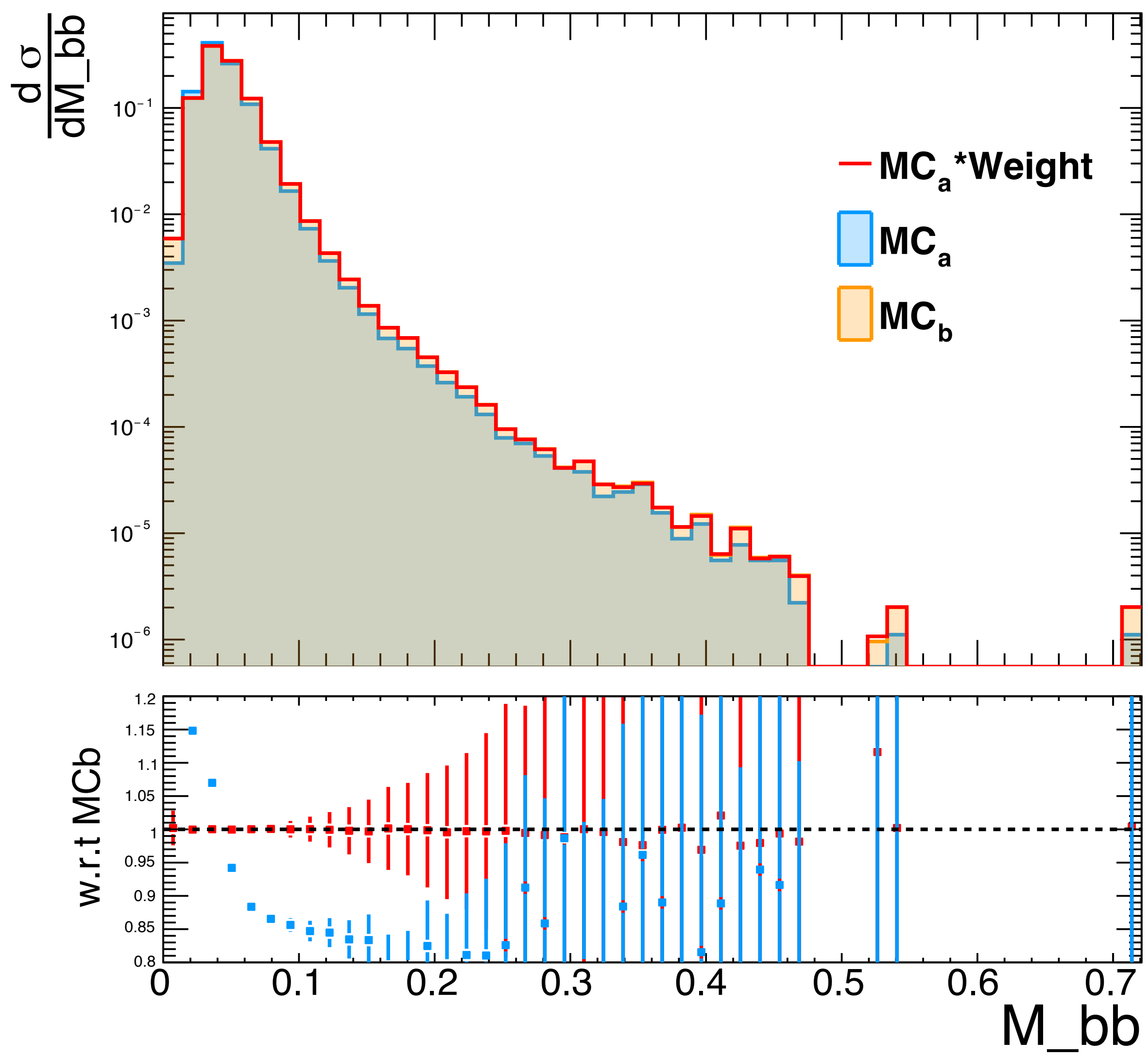


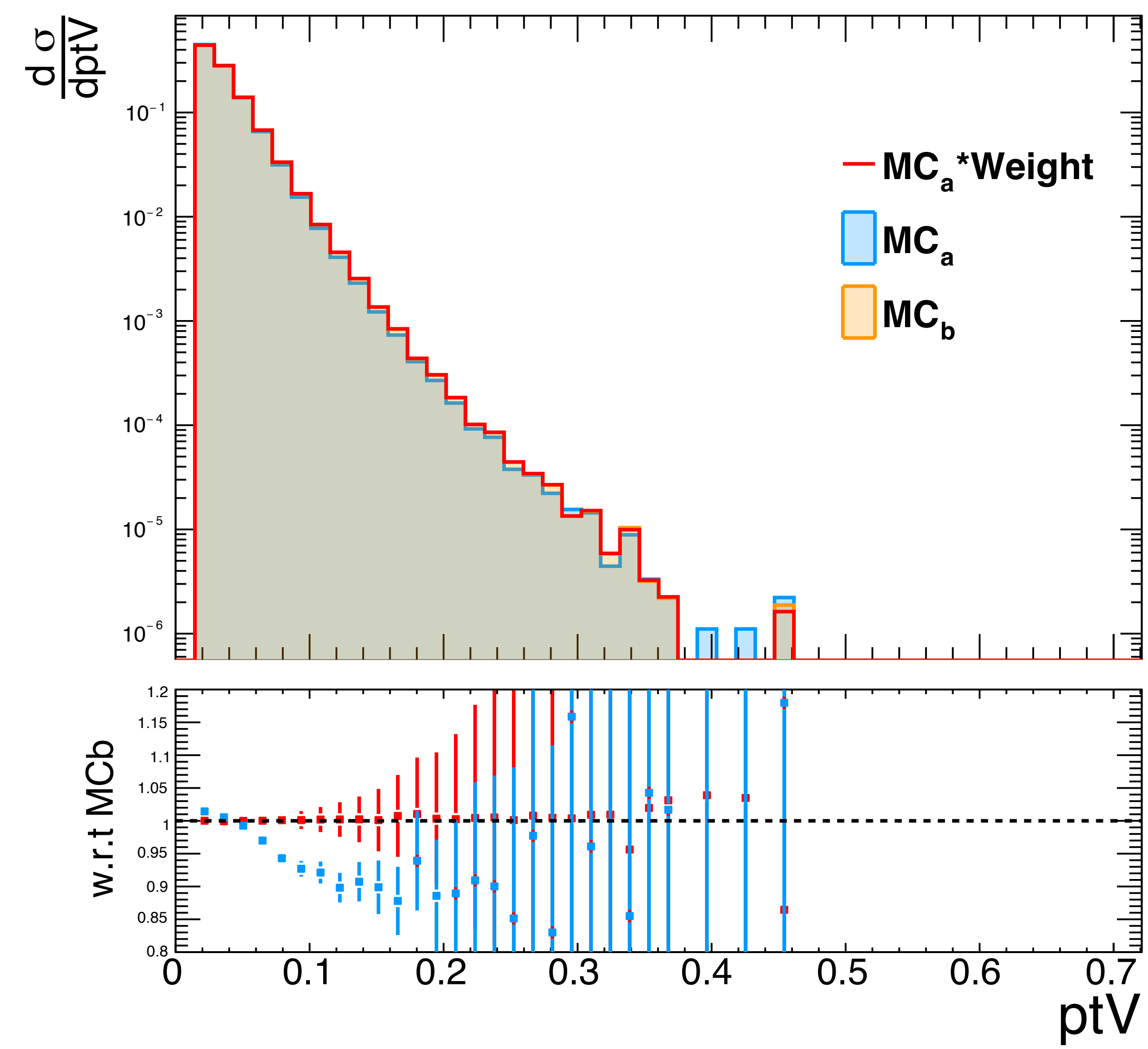


☆ The NN is able to learn mapping these two perfectly!

DSNNr Observables

- Sanity checks: MCa v.s. MCa x dR(bb) weights.
- The framework is working (and we also learn a lot about our dataset)!

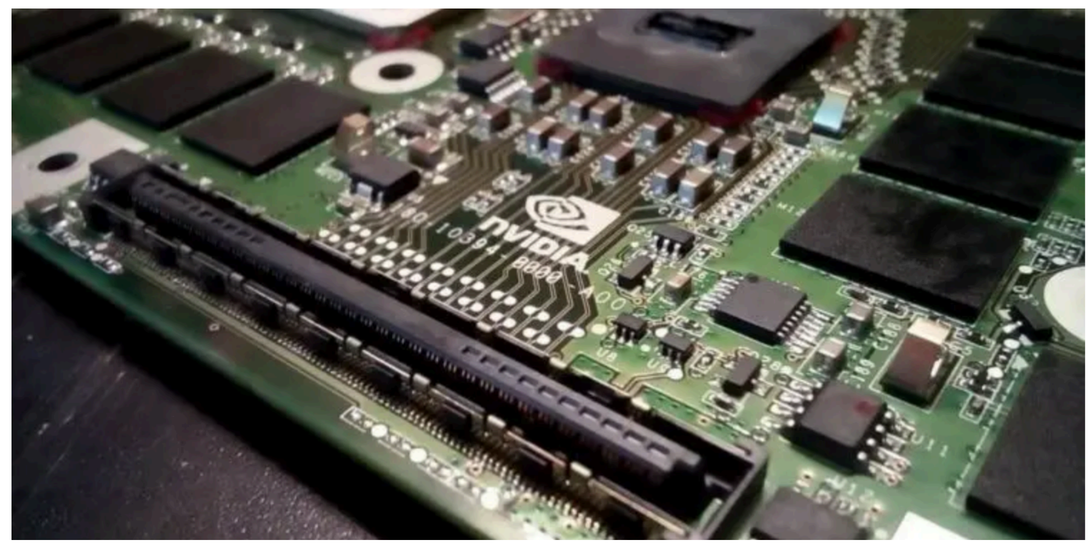




Computing Performance among CPUs & GPUs

- Our system Information:

- We run on Lxplus with a GPU/CPU via a Docker image.
- HTCondor batch jobs.
- TensorFlow version: 2.4.1
- Python version: 3.6.9
- CUDA v.11.2 /cuDNN



cuDNN

☆ Each epoch's time consuming is from ~100s (CPU) to ~7s (GPU)!
☆ Running the whole application with ~100,000 events takes 40 mins on CPUs+GPUs, and 3 hours on CPUs. (~50 epochs)
☆ About 10 GB memory usage (1M events each).

Computing Performance among CPUs & GPUs

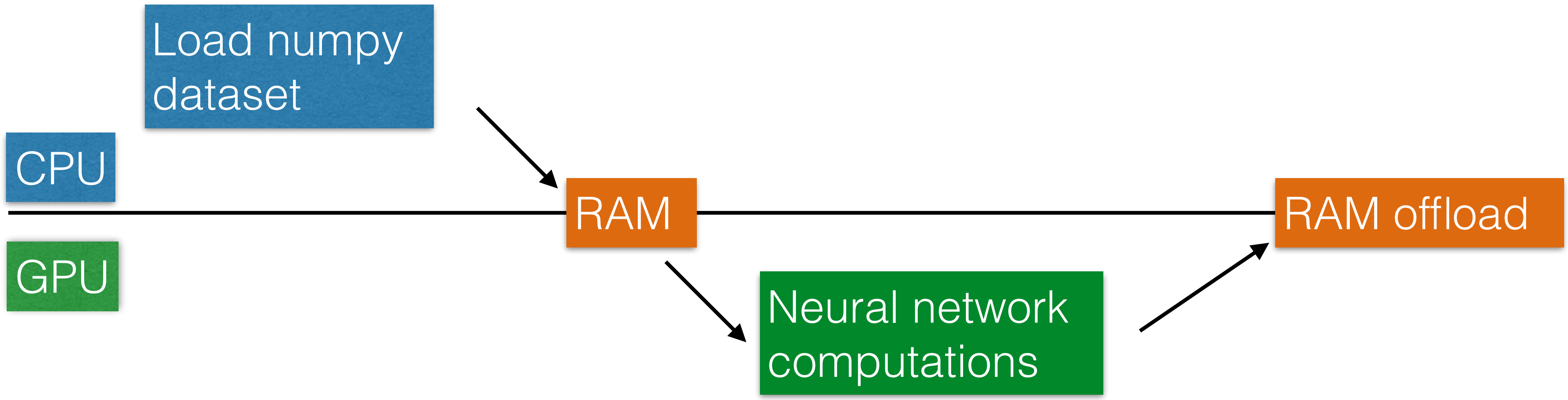
- Training the dense neural network with 5 layers of 20 nodes v.s. a 20 layer network with 100 nodes is approximately the same amount of time.
  -> Having more GPUs would benefit us little.

☆ Increase GPU utilization:
  batch size = 2000 takes 7 sec.
  batch size = 100,000 takes 2 sec.

1. If enough memory to fit.
2. Warning of increasing batch size, here

☆ Wait for about 1-2 sec to go from epoch to the next.

For 1 epoch,



What Next?

- Aim to get this DSNN project done in 3 months (by the end of June or so).
  - Train on different samples, e.g. ttbar, W+jets…
  - Compare it with BDTr performance.
  - Validation.
- Deploy this technique to wherever it could come in handy.
  - VH(bb/cc) analysis framework.
  - Instead of generating 3*nominal events (nominal + variations) to make uncertainties, we can store the weights into a vector once the NN learns the differences.
    -> Save the disk storage, save the CPU usage!
- Run this DSNN framework with the BNL facilities
  - submit jobs to the SLURM cluster. (Thanks to Doug Benjamin)
  - use Jupyter to access the resources.
- Run this DSNN framework with Google facilities.
  - will start running on CPUs using a small fraction of the dataset. (Fernando?)

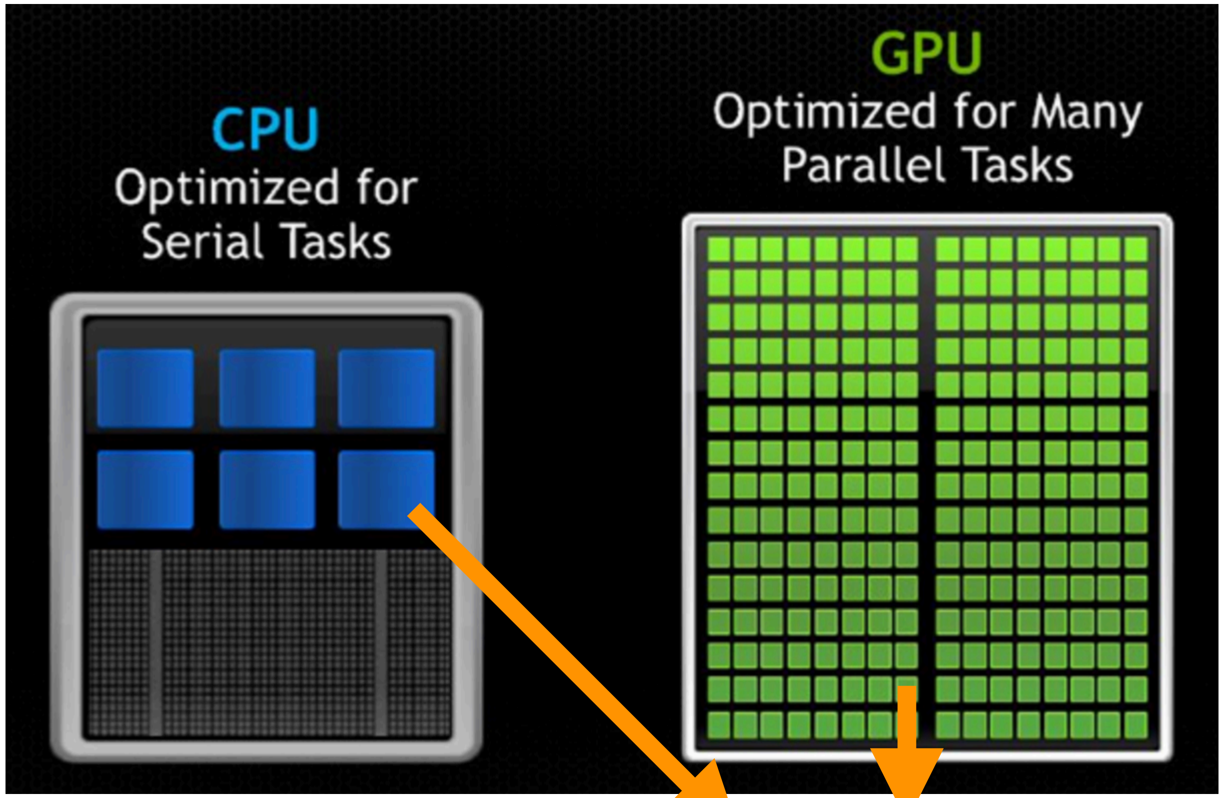
Backup

Discussions - CPU vs GPU

- Why GPUs are better?

  - CPUs tend to be working on single program -> increase the **core clock** speed to get better performance.
  - GPUs were originally designed for 3D rendering -> increase the **memory clock**.
  - GPU wins against CPU in execution throughput of massively parallel programs.

Fig.



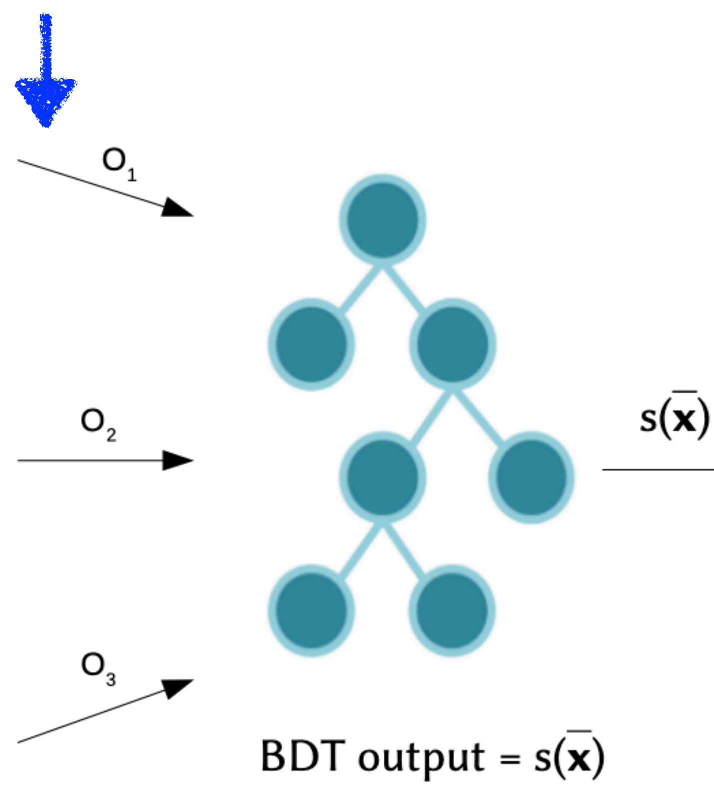
Can each core of CPU vs GPU be compared in terms of computing cost?

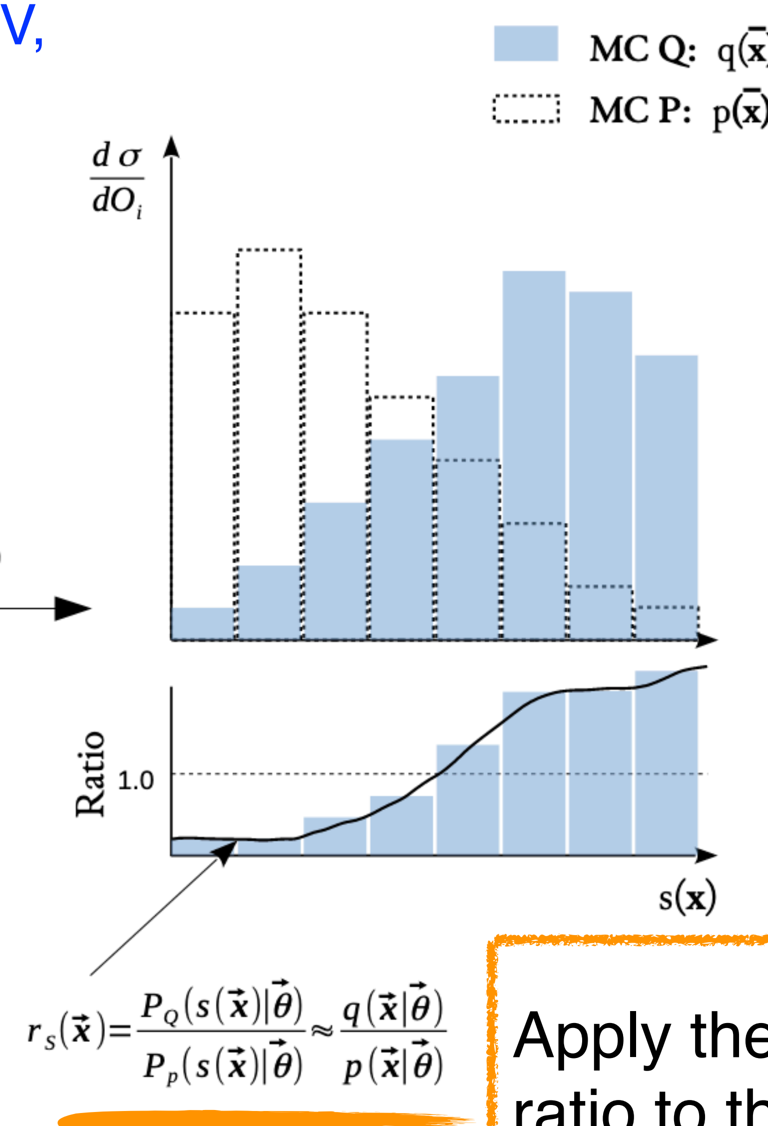
BDTr recap

- The modeling uncertainty is calculated through Boosted Decision Tree reweighting (BDTr) technique in the current VH(bb) analysis. (Stephen's talk) - map two MC predictions and take their differences as a systematic.

events (MC Q + MC P) are sorted by pTV, mass, dR… etc. 13 observables.

☆ Each layer of the tree is a cut on the variable.
☆ BDT algorithm decides the cut priorities based on the variable and cut boundary that yields the best S/B.





train classifier on (MC_P * pTV weights) vs MC_Q
- Trees are built based on the weighted events from the previous tree.
- Average of all the trees to get s(x).

Apply the classifier ratio to the nominal to obtain the systematic shape.

- It works well.