CERN openlab

# Deep Learning deployment on TPUs

**Renato Cardoso**

Sofia Vallecorsa

Google

**CERN openlab**

**Technical Workshop**

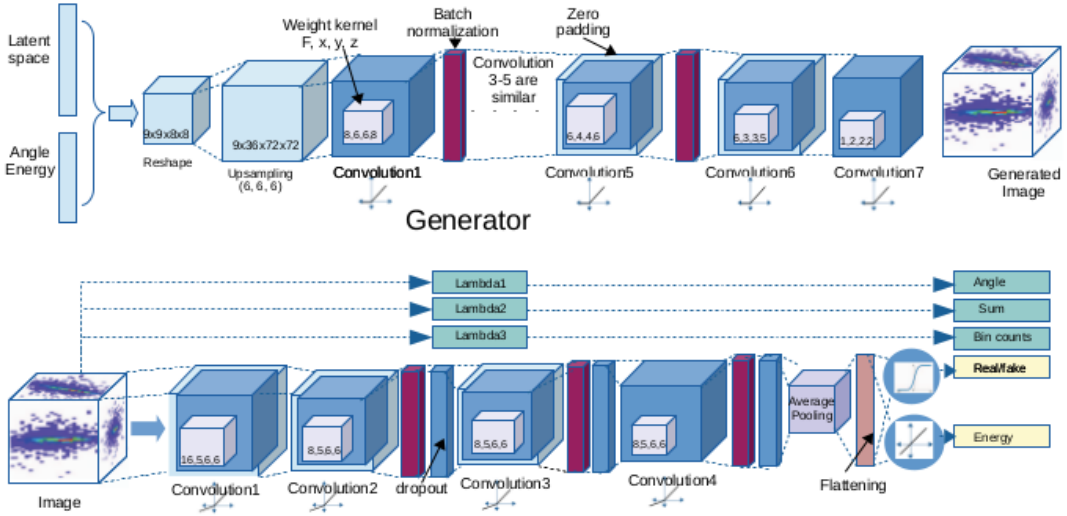
10/03/2021

# Introduction

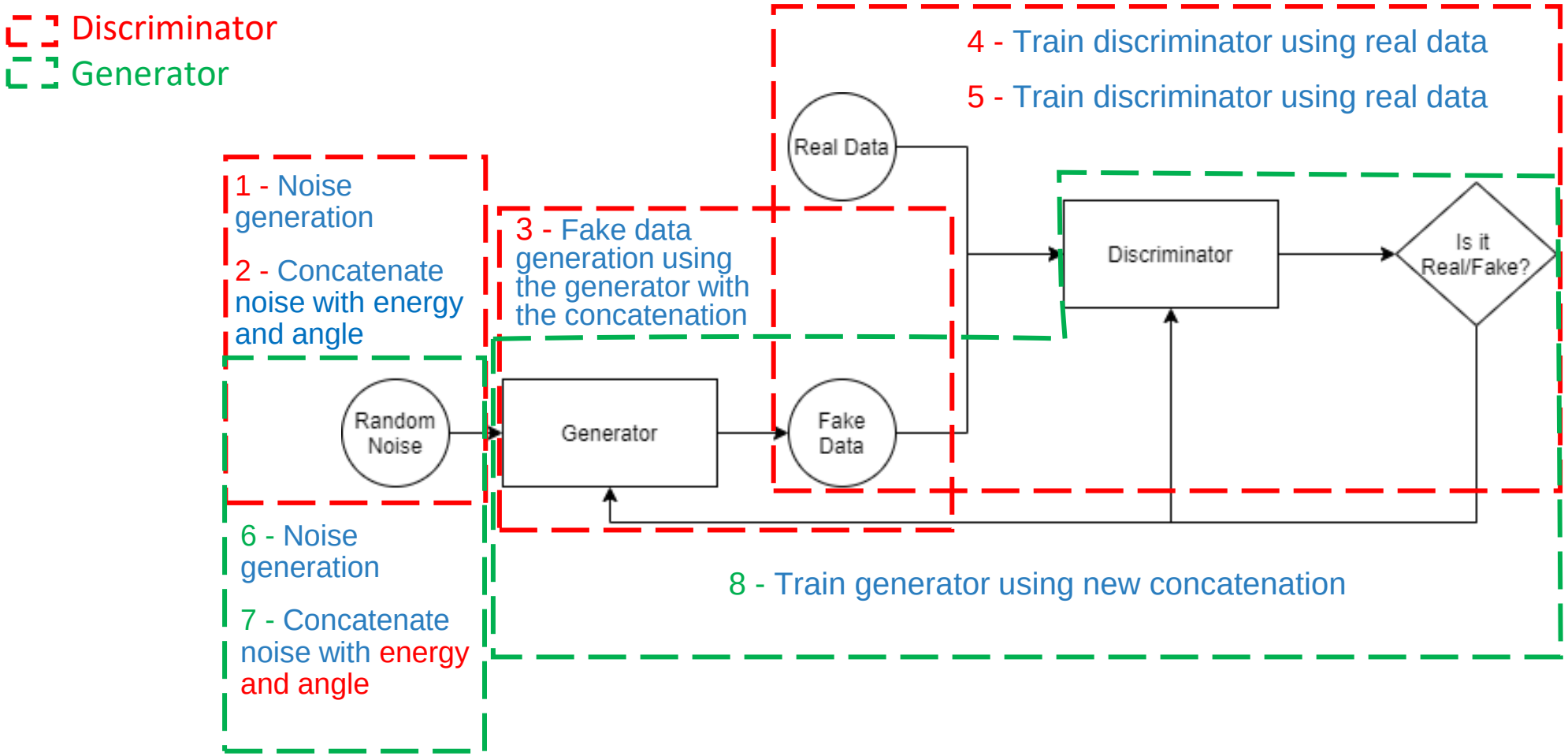
- Objective:
  - Use Deep Learning to solve different tasks related to data processing in High Energy Physics (HEP)
  - Decrease training time while still maintaining the validity of the results
- Solution:
  - Usage of efficient hardware for machine learning accelerations
  - Examples:
    - Dedicated GPUs for machine Learning Deployment (V100)
    - Tensor Processing Units (TPUs)
    - Intelligence Processing Units (IPUs)
- Problem:
  - Needs development, not out of the box deployment

# The Generative Adversarial Network

- 3D convolutional Generative Adversarial Network using physics constraints.
- Generates 51x51x25 pixels images, representing energy depositions in calorimeters, similar to the ones generated by Monte Carlo.
- This model was chosen because it is compute bound

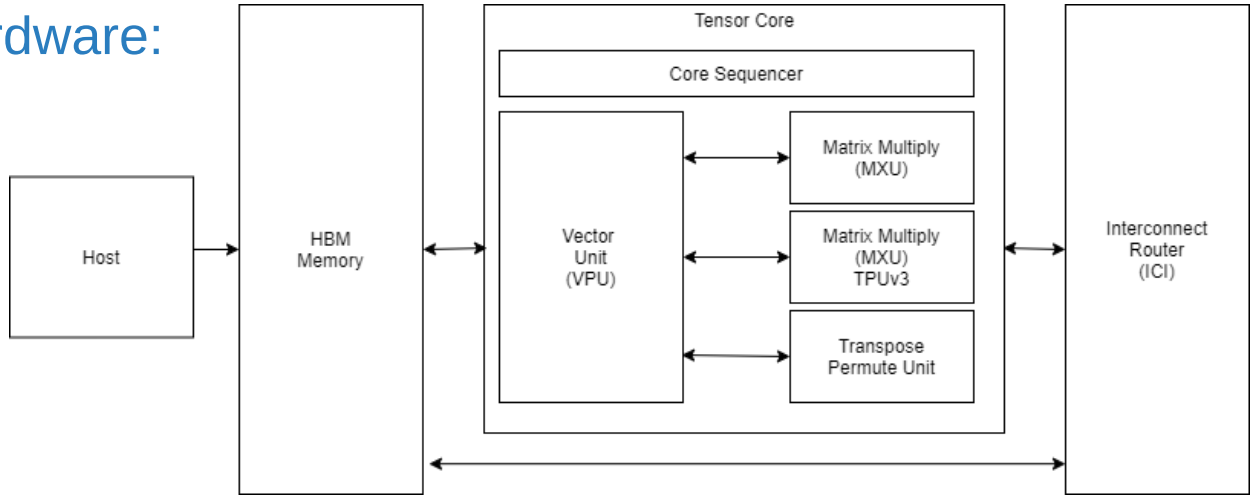


# GAN Training



# TPUs – Tensor Processing Units

- What are TPUs:
  - Tensor Processing Units (TPUs) are application-specific integrated circuits (ASICs) developed by Google in order to accelerate the machine learning workload.
- What is a TPU hardware:

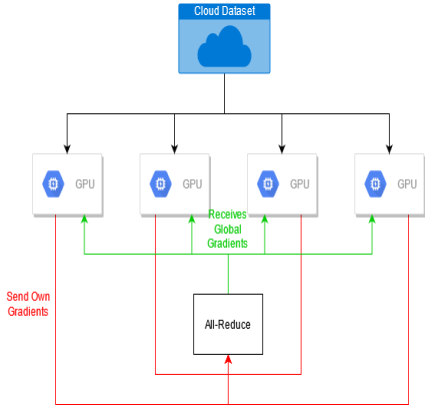



  - A large two-dimensional matrix multiply unit (MXU) size of 128x128

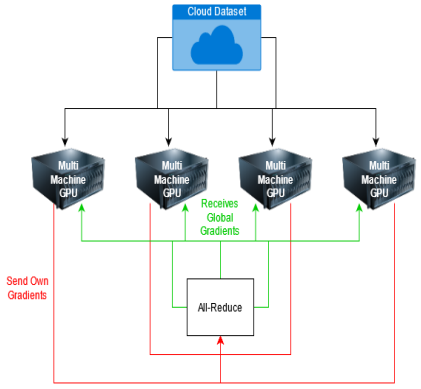
# Distributed training in TensorFlow

- Achieve distributed training using TensorFlow distributed Strategy tf.distribute.Strategy:
  - A TensorFlow API to distribute training across multiple GPUs, multiple machines with GPUs or TPUs.
- Can be done using a high level TensorFlow training function such as model.fit or by using a custom training loop:
  - Model.fit is not optimal due the GAN architecture
  - ➡ Use a custom training to include all the training steps

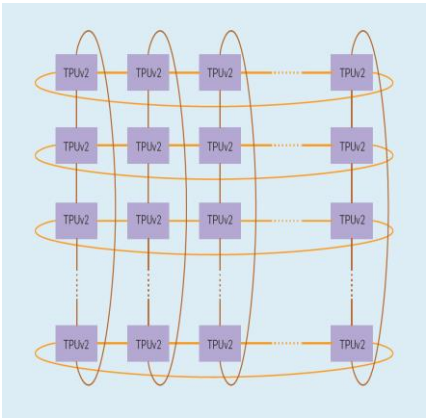
Mirrored Strategy




Multi Worker Mirrored Strategy

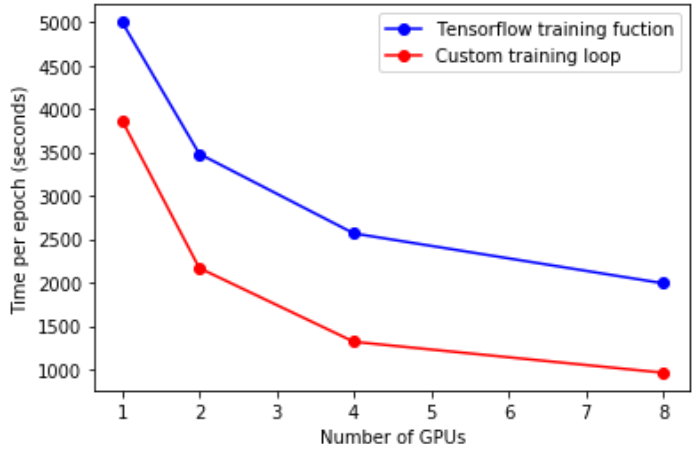


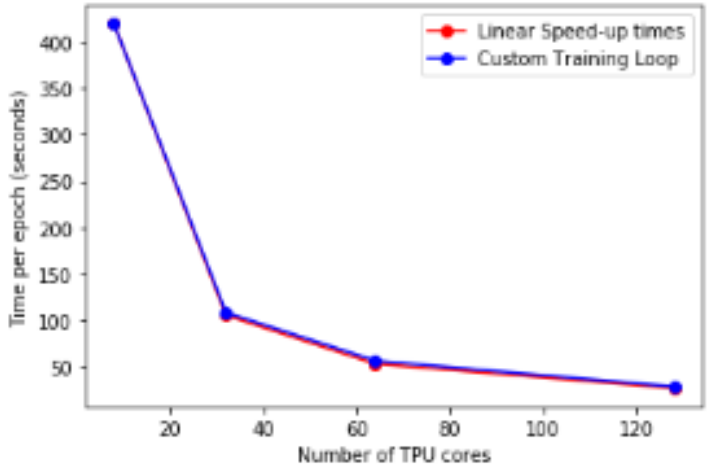

TPU Strategy



# Parallel Approach

- Advantages:
  - Much faster training times
  - Training times are stable with the increase of the number of GPUs
  - The training process can be fully distributed
- Disadvantages:
  - Hard to deploy
  - Needs to change the core training code
  - Can’t use eager functions.
- Final Bottleneck is the dataset preparation and distribution
  - Dataset is preprocessed for each file every epoch
  - Dataset needs to be read to the host machine and then preprocessed
  - Remove Sequential part of the code



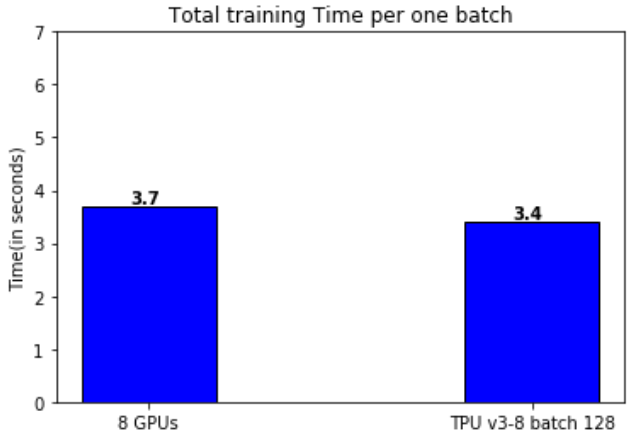

Results obtained with collaboration with CloudBank EU

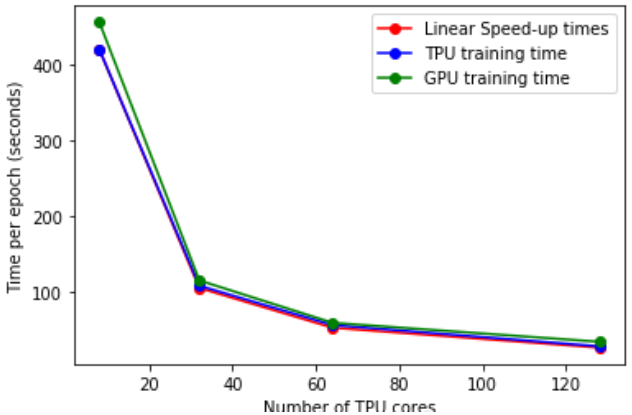
# GPUs Vs TPUs

Times:

- TPU 128 cores
  - 28 seconds per epoch
- 128 GPU
  - 34 seconds per epoch
- A difference of 20%

Near Linear Speed-Up for both TPUs and GPUs

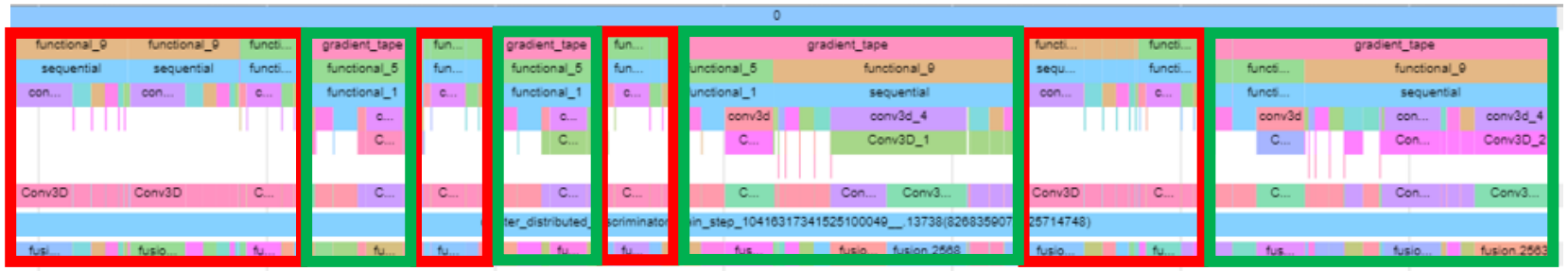


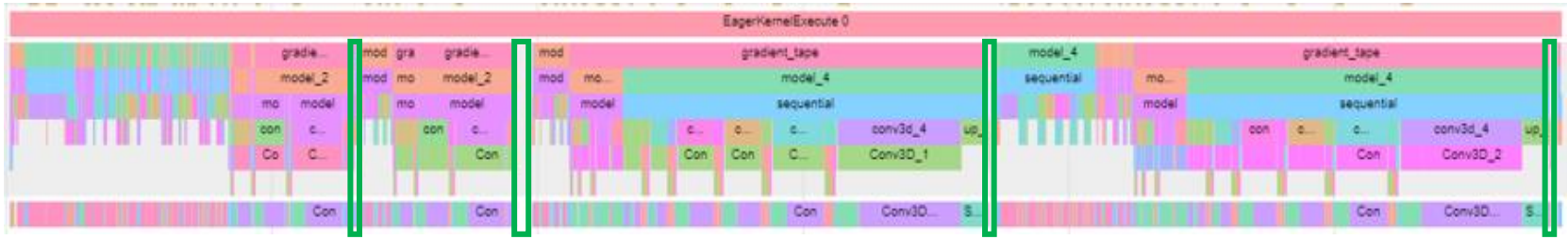




Results obtained with collaboration with CloudBank EU

# Further Analysis

- TPUs:
  - TPUs are only idle 0.7% of the time, with 28.5% of idle time use for All-Reduce Operations
  - 38% of the time being used for the **forward-pass**
  - 61% **for back-propagation**
- GPUs:
  - GPUs are only idle 2.9% of the time, mostly used for All reduce operations
  - Similar percentages for forward and backward propagation as the TPUs.
- Program is not input bound, 0% of the training step time was spent waiting for input
- With this profile it is possible to verify that the model is compute bound




4 gradient tapes
- After each gradient tape there is one all reduce
- Followed by a RMSprop and the corresponding updates

# Future Work

Continue the distributed deployment on different models/architectures

- Collaboration with UNISAT
  - Progressive GAN for generation of satellite images
- Branch out to different architectures:
  - Data classification
  - Auto-encoders
  - Etc.

# Thank You

# QUESTIONS?

*renato.cardoso@cern.ch*

Other contact (e.g. Twitter)