Deploying Models with Big Data

What is big data?

Big data is essentially a large volume of data

- Structured
- Semi-structured
- Unstructured

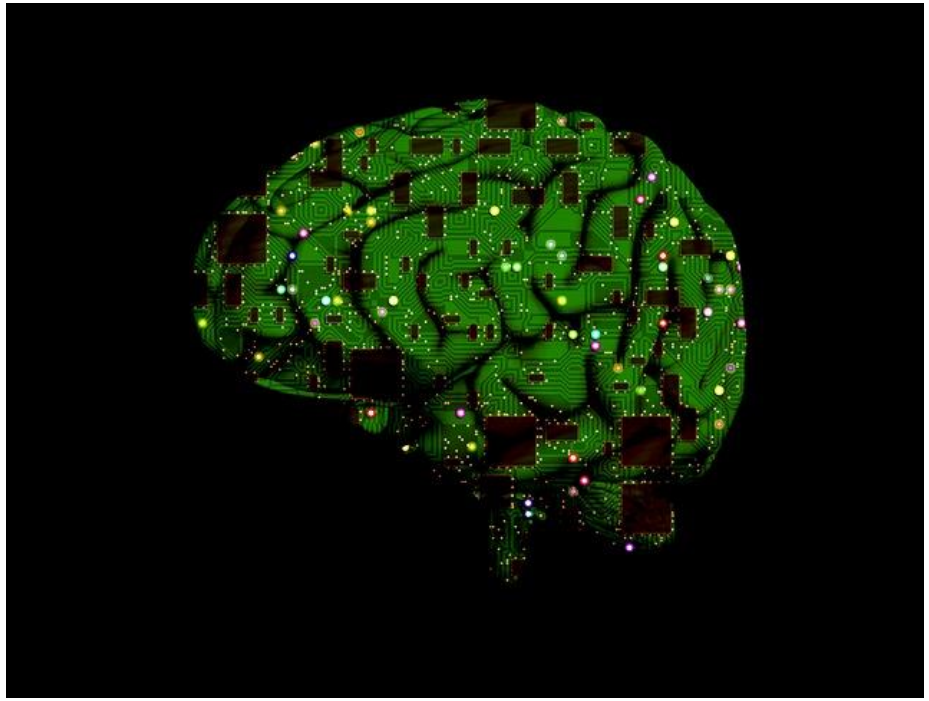
Normally used for analytics and machine learning

Typically terabytes and above

Challenges of using big data in ML

Big Data

- Text - large documents
- Images
- Complex Time Series



Deep Learning

Challenges of using big data in ML

- Neural networks can be extremely compute heavy.
- During training takes up a lot of compute power.
- When scoring can also take a lot of computer power
- Difficult to scale
- Normally rely on GPUs.
- GPUs can be scarce and expensive

Deployment pipeline for CNN

- Coding Challenges
- Deployment Challenges

V2 Plant Seedlings Dataset

V2 Plant Seedlings Dataset

[Kaggle website]

CNN - Production Code

V2 Plant Seedlings Dataset - Production Code

Download and navigate the attached code to follow the video

Reproducibility in Neural Networks

Reproducibility - why it matters

- Reproducibility ensures models' gain in performance are genuine.
  - Not from hidden sources of randomness
- Reproducibility reduces or eliminates variations when re-running experiments
  - Essential for testing and continuous integration and iterative refinement of models.
- Reproducibility is increasingly important as sophisticated models and real-time data streams push us towards distributed training across clusters of GPUs.
  - More sources of non-determinism behaviour during model training.

What causes non-reproducibility?

- **Random initialization of layer weights:** the weights of the different layers are intialised randomly
- **Shuffling of datasets:** The dataset is randomly shuffled for example if we leave 10% of the samples for cross-validation within model.fit
- **Noisy hidden layers:** Dropout layers, which exclude the contribution of a particular neuron, are initialised randomly.
- **Changes in ML frameworks:** Different versions of ML libraries can lead different behavior.

What causes non-reproducibility?

- **Non-deterministic GPU floating point calculations:** If using GPUs, certain functions in cuDNN, the Nvidia Deep Neural Network library for GPUs, are stochastic, which means that they are initialised randomly at each run.
- **CPU multi-threading:** CPU parallelization when using Tensorflow

How to control for random initialisation?

- Keras gets its source of randomness from the NumPy random number generator
  - Seed the numpy random generator both for Theano or TensorFlow backend.
- Tensorflow uses multiple threads, which may cause non-reproducible results
  - Force TensorFlow to use single thread.
- **In python 3, you need to set a flag before running your script**
  - **PYTHONHASHSEED=0**
- **Set the** cuDNN as deterministic

How do I do that?

- In the next article, I have included code from various sources that allows you to seed a neural network in its multiple initialisation instances