NEURAL NETWORKS ON A BUDGET

Giles Strong

Supervisors: M. Gallinaro, J. Varela

LIP PhD Students Workshop, Online - 25/05/20

giles.strong@outlook.com

twitter.com/Giles_C_Strong

Amva4newphysics.wordpress.com

github.com/GilesStrong

MACHINE LEARNING IN HEP

- Many analyses and experiment software now aim to benefit from using machine learning approaches; often necessary to achieve competitive performance
- ML is now an integral part of HEP, and well recognised as such:
  - Establishment of dedicated forums & groups (IML, ATLAS & CMS ML groups)
  - Identified in 2020 update of the European Strategy for Particle Physics as essential R&D

European Strategy Update

2020 Strategy Statements

4. Other essential scientific activities for particle physics

Computing and software infrastructure

- There is a need for strong community-wide coordination for computing and software R&D activities, and for the development of common coordinating structures that will promote coherence in these activities, long-term planning and effective means of exploiting synergies with other disciplines and industry
- A significant role for artificial intelligence is emerging in detector design, detector operation, online data processing and data analysis
- Computing and software are profound R&D topics in their own right and are essential to sustain and enhance particle physics research capabilities
- More experts need to be trained to address the essential needs, especially with the increased data volume and complexity in the upcoming HL-LHC era, and will also help in experiments in adjacent fields.

d) Large-scale data-intensive software and computing infrastructures are an essential ingredient to particle physics research programmes. The community faces major challenges in this area, notably with a view to the HL-LHC. As a result, the software and computing models used in particle physics research must evolve to meet the future needs of the field. *The community must vigorously pursue common, coordinated R&D efforts in collaboration with other fields of science and industry to develop software and computing infrastructures that exploit recent advances in information technology and data science. Further development of internal policies on open data and data preservation should be encouraged, and an adequate level of resources invested in their implementation.*

GENERAL ML REQUIREMENTS IN CMS

- During data collection (Level 1 and high-level triggers):
  - Train algorithm once, use for several years = train time not a concern
    - Probably GPUs available
  - Application time, extremely quick (40 MHz & 30 kHz) = quantification, pruning, lookup-tables
    - L1 Hardware-based: FPGA, ASIC
    - HLT Software based: CPU cluster

GENERAL ML REQUIREMENTS IN CMS

- During data processing (reconstruction):
  - Train algorithm once, use for a year = train time not a concern
    - Probably GPUs available
  - Application time, moderately quick, but high volume = CPU cluster
- During data analysis:
  - Train algorithm multiple times at short notice = train time < 1 day
    - Cannot assume GPU access, must work well on CPU
  - Application time, can be slow = must process entire dataset in under a few hours
    - Cannot assume GPU access, must work well on CPU

HIGGS ML SOLUTIONS

- 2014 Higgs ML Kaggle competition simulated a typical data-analysis level application of ML in HEP
- Top performance requires:
  - 13h using a 1000 euro GPU
    - 110m accounting for hardware improvement
  - Or 36h on an 8-core CPU instance
- Most analysis-level researchers just have a laptop...

	1^{st} place	2^{nd} place	3^{rd} place
Method	70 DNNs	Many BDTs	108 DNNs
Train-time (GPU)	12 h	N/A	N/A
Train-time (CPU)	35 h	48 h	3 h
Test-time (GPU)	1 h	N/A	N/A
Test-time (CPU)	???	???	20 min
Score	3.80581	3.78913	3.78682

PROBLEMS FACING ANALYSIS-LEVEL RESEARCHERS

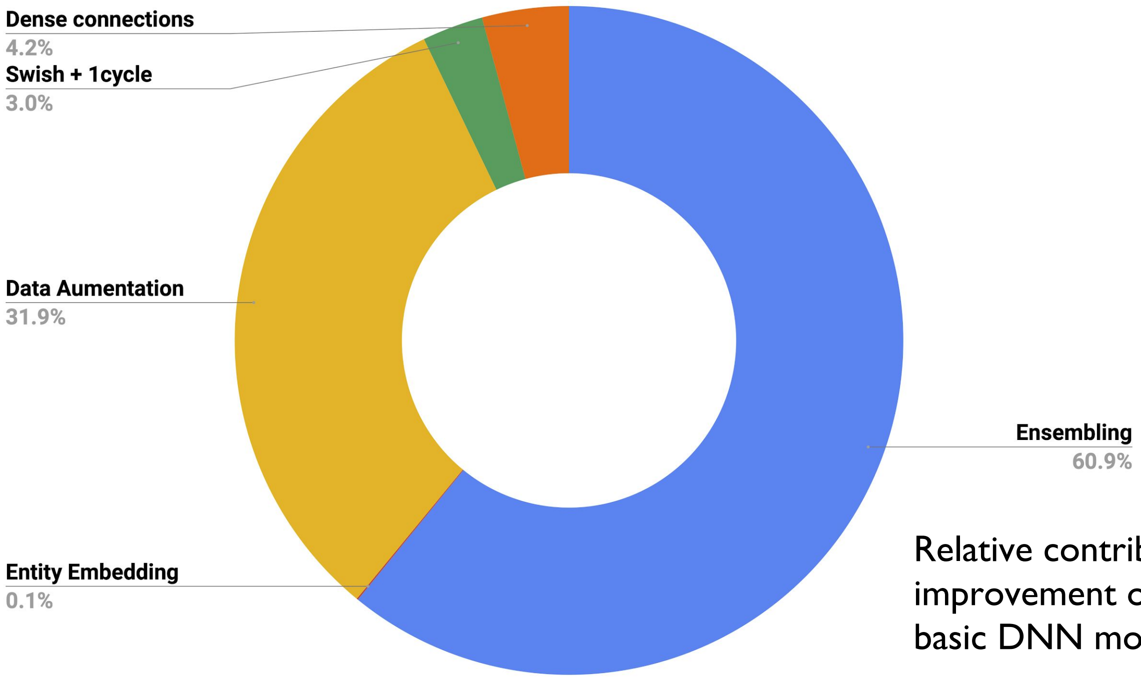
1. How can competitive performance be achieved in a short time?
2. And can this be done without dedicated hardware?

MODERN DEEP-LEARNING TECHNIQUES

- Top HiggsML solution used relatively simple DNNs; main benefit = 70 DNNs
- In Strong, 2020 I studied the impact of new DNNs techniques on performance and timing
- Solution matched top performance, but trained in 14 minutes on a laptop CPU
  - 86% effective speedup over 1st-place GPU
- Hardware for mine:
  - GPU: Nvidia 1080 Ti
  - CPU: Intel i7-8559U (MacBook Pro 2018)
  - More hardware timings in paper

	Our solution	1^{st} place	2^{nd} place	3^{rd} place
Method	10 DNNs	70 DNNs	Many BDTs	108 DNNs
Train-time (GPU)	8 min	12 h	N/A	N/A
Train-time (CPU)	14 min	35 h	48 h	3 h
Test-time (GPU)	15 s	1 h	N/A	N/A
Test-time (CPU)	3 min	???	???	20 min
Score	3.806 ± 0.005	3.80581	3.78913	3.78682

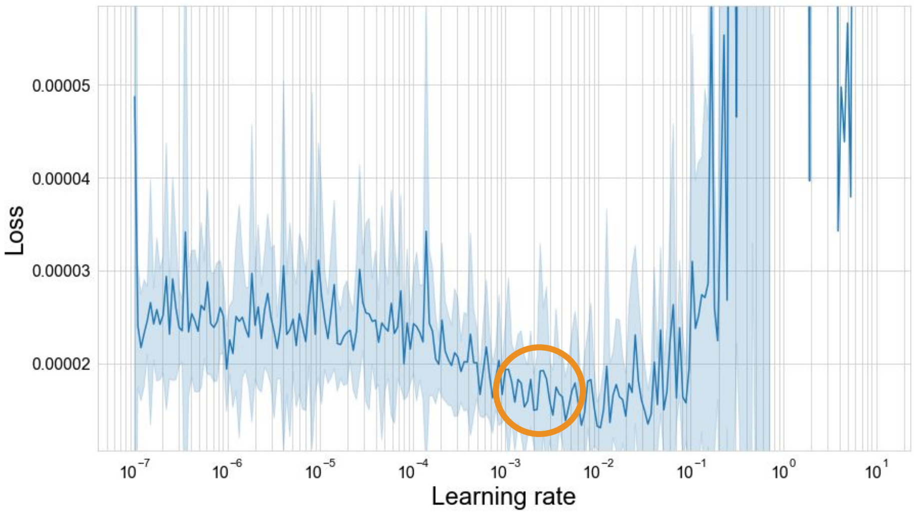
IMPROVEMENT CONTRIBUTIONS



Relative contributions to improvement over a single, basic DNN model

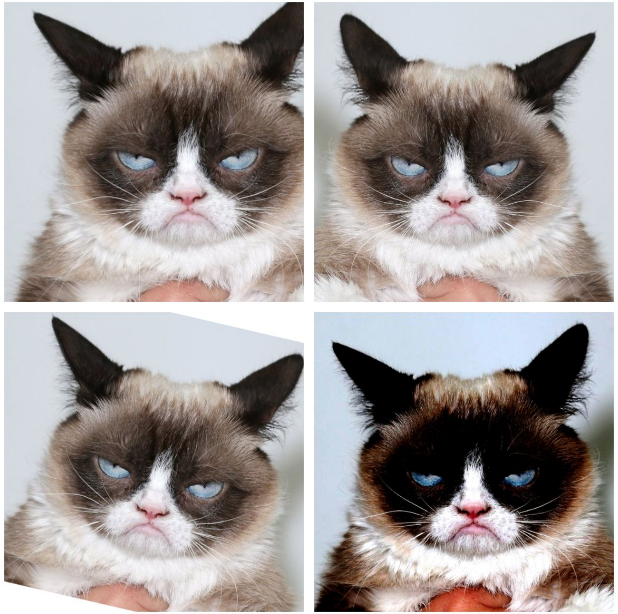
LEARNING RATE FINDER

- “*[The Learning Rate] is often the single most important hyperparameter and one should always make sure that it has been tuned*” - Bengio, 2012
- Previously this required running several different trainings using a range of LRs
- The LR range test (Smith 2015 & 2018) can quickly find the optimum LR using a single epoch of training (e.g. a few seconds)



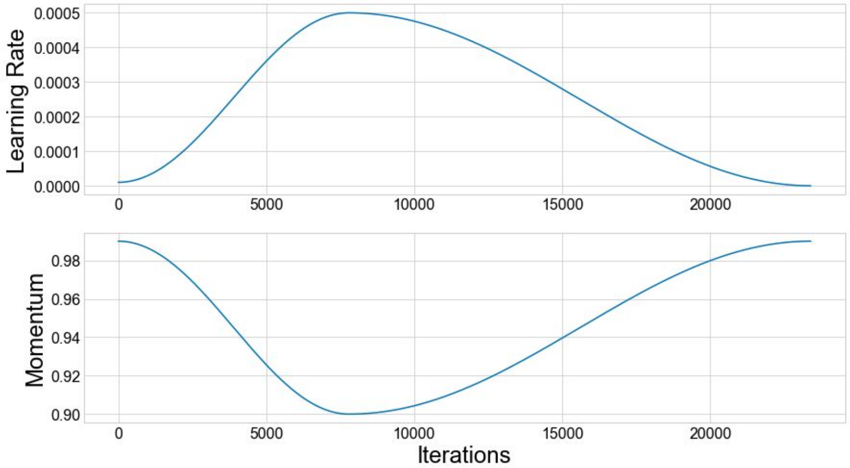
DATA AUGMENTATION

- Data augmentation = class-preserving input transformations
    - For CMS/ATLAS = rotating and flipping events
- Artificially increase the amount of training data (train-time augmentation), e.g Krizhevsky et al. 2012
- Can be applied at test time by predicting the class of a range of augmented data and then taking an average
- Increased performance considerably
- Also increased train and test time



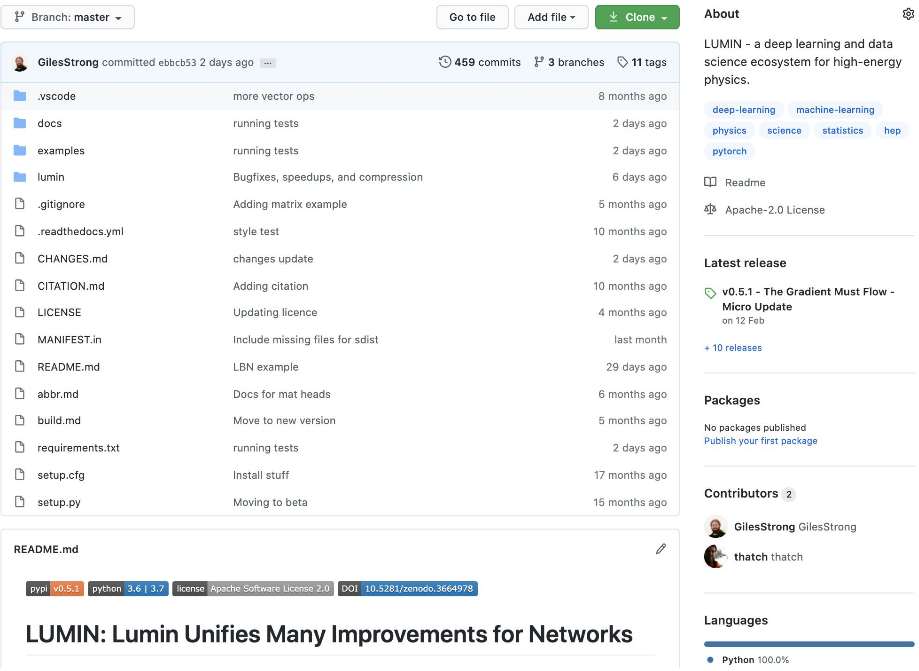
1CYCLE SCHEDULE

- Smith 2018 introduces the 1cycle schedule
- This involves running through a single cycle of increasing and then decreasing the lR, with a similar, inverted schedule applied to momentum/$beta_1$
  - Provides very fast convergence due to high-LR balanced by momentum
  - Original paper used linear interpolation
  - FastAI found a cosine interpolation was better
- Reduces training time by over 50% with no change in performance



LUMIN

- LUMIN is a PyTorch wrapper library that provides implementations for these methods
- Also includes other useful methods & classes for working with HEP data and columnar data in general, and more
  - E.g. recent update adds RNNs, CNNs, and a few graph-nets
- Links:
  - Docs
  - Github
  - Colab examples
  - Issues - contributions welcome!



SUMMARY

- ML is an integral part of HEP, and necessary to achieve competitive performance
- Hardware and training-time requirements can be lowered significantly by using modern techniques
  - See Strong, 2020 for complete study
- An open-source software package (LUMIN) is available to allow researchers to easily apply these methods
  - You can help to develop it further