IMPROVEMENTS TO ML FOR SEARCHES AT THE LHC

- A summary of MLST:ab983a

Giles Strong

ICHEP, Prague (online) - 28/07/20

giles.strong@outlook.com

twitter.com/Giles_C_Strong

Amva4newphysics.wordpress.com

github.com/GilesStrong

INTRODUCTION

ML REQUIREMENTS AT ANALYSIS LEVEL

- Example: typical event-level classifier in a search
- Train algorithm multiple times at short notice = train time < 1 day
  - Cannot assume GPU access, must work well on CPU
- Application time depends on dataset size and number of systematics (run multiple predictions per event)
  - Typically want to 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
- Entrants included both physicists and professional data-scientists
  - Strong competition
- Top performance requires:
  - 13h using an expensive GPU
    - 110m accounting for hardware improvement
  - Or 36h on an 8-core CPU instance
- Most analysis-level researchers just have a laptop or scheduled access to shared GPUs.

	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

QUESTION

- Have there been any new methods in deep learning since 2014 which when applied to a HEP search:
  - Improve sensitivity to signal?
  - Reduce training and application time?
  - Have a lower hardware requirement?
- Let's use the HiggsML challenge as a benchmark and see!

HIGGS ML DATASET

- ATLAS 2012 MC full simulation with Geant 4
- Signal: Higgs to di-tau
- Backgrounds: $Z \rightarrow \tau\underline{\tau}$, $t\underline{t}$, and W decay
- Events selected for the semi-leptonic channel: $\tau\tau \longrightarrow (e \mid \mu) + \tau_{\mathrm{h}}$
- 250,000 labelled events for training, 550,000 unlabelled events for testing
- 31 features:
  - 3-momenta of main final-states and upto two jets (p_T ordered)
  - High-level features: angles, invariant masses, fitted di-tau mass (MMC), et cetera

CHALLENGE AIM

- Solutions must predict signal or background for each test event
- Solutions ranked via their Approximate Median Significance
  - Quick, accurate, analytical approximation of full discovery significance
  - *s* = sum of weights of true positive events (signal events determined by the solution to be signal)
  - *b* = weights of false positive events (backgrounds events determined by the solution to be signal)
  - b_r = constant term (set to 10 for the challenge)

$$\mathrm{AMS} = \sqrt{2\left(s + b + b_r\right) \log\left(\left(\left(1 + \frac{s}{b + b_r}\right) - s\right)\right)}$$

BASELINE MODEL

- The basic classifier is:
  - 4-layer 100 neuron, fully-connected network, with ReLU activations
  - Adam to minimise the weighted binary cross-entropy of event class predictions
  - Learning rate found using LR range test (Smith 2015 & 2018, see backups)
- An ensemble of 10 such classifiers is trained
- Baseline achieves metric-score of 3.664 ± 0.007

METHOD TESTING

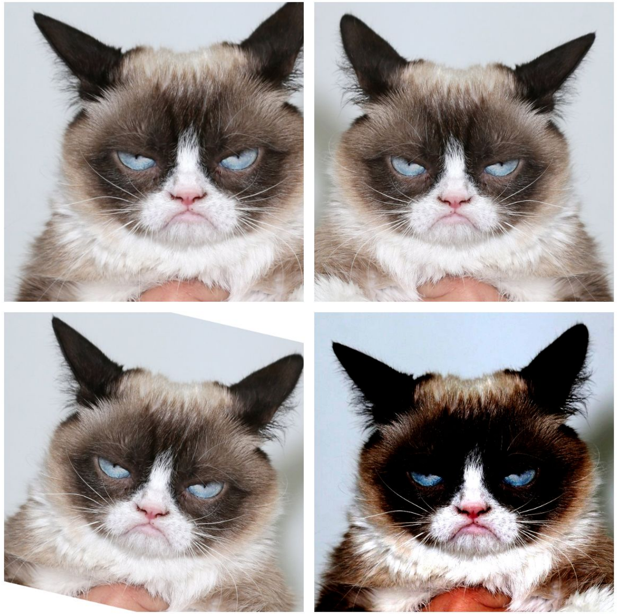
Presented in order tested, but some methods are skipped to save time

CATEGORICAL ENTITY EMBEDDING

- Guo & Berkhahn 2016 : a method of inputting categorical features without 1-hot encoding
- Gives a small improvement, but there's only one categorical feature in the dataset (number of jets)
- See paper or backups for details

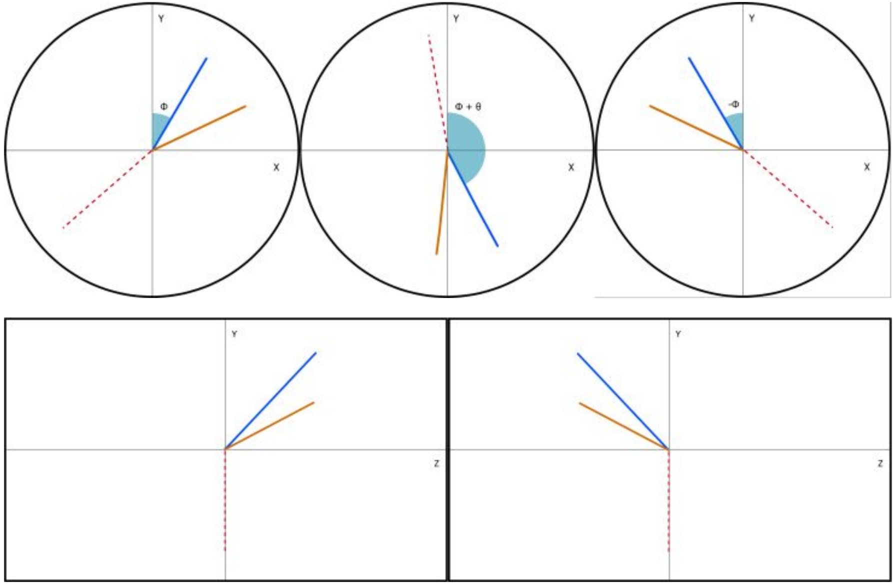
DATA AUGMENTATION

- Copy data by exploiting invariances between input and target:
  - E.g. can flip, zoom, rotate, & adjust image pixels but object does not change class
- Applied at train-time to artificially increase dataset size e.g Krizhevsky et al. 2012
- Applied at test-time to get multiple predictions per datapoint and average



DATA AUGMENTATION

- At the CMS and ATLAS detectors at the LHC, can exploit the azimuthal and longitudinal invariance of events:
  - Rotate in ϕ, flip in η, and flip in either *x* or *y* axis
- Alternative is to remove symmetries by setting common alignment for events
  - E.g. rotate & flip events such that leptons are always at $\phi = 0$, $\eta > 0$, and taus are always at $\phi > 0$
- Using data augmentation results in:
  - Large performance improvement
  - Very large increase in train & application time (but still reasonable to use)

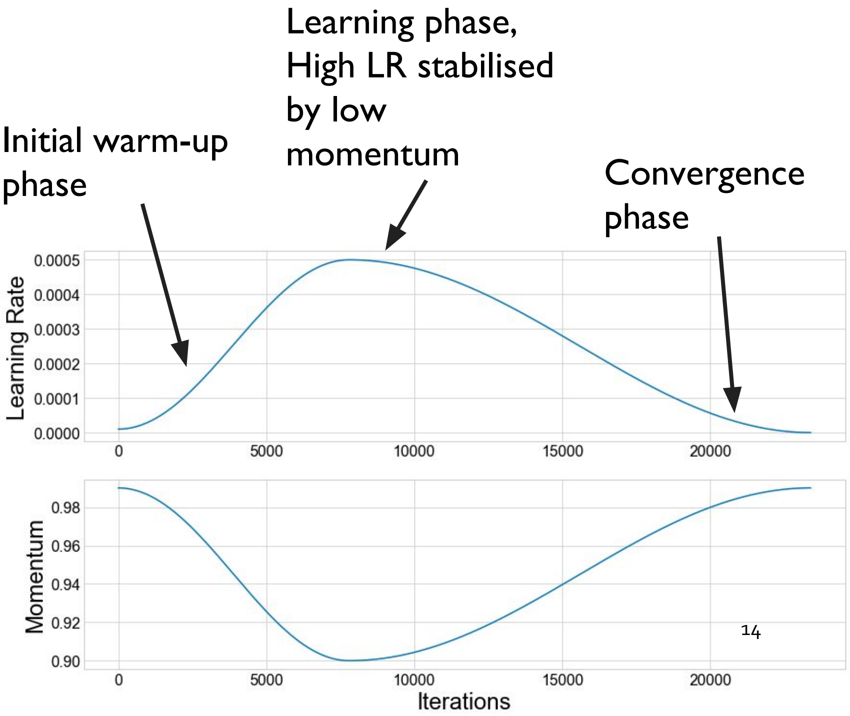


SKIPPED METHODS

- Cosine annealed LR schedule (Loshchilov and Hutter, 2016)
  - Slight improvement in performance, but replaced with 1cycle (coming up soon)
- Swish activation function (Ramachandran et al., 2017)
  - Small performance improvement
- Advanced ensembling: Snapshot ensembling, Fast geometric ensembling, Stochastic weight averaging
  - SWA gave slight improvement in performance, but replaced with 1cycle (coming up soon)

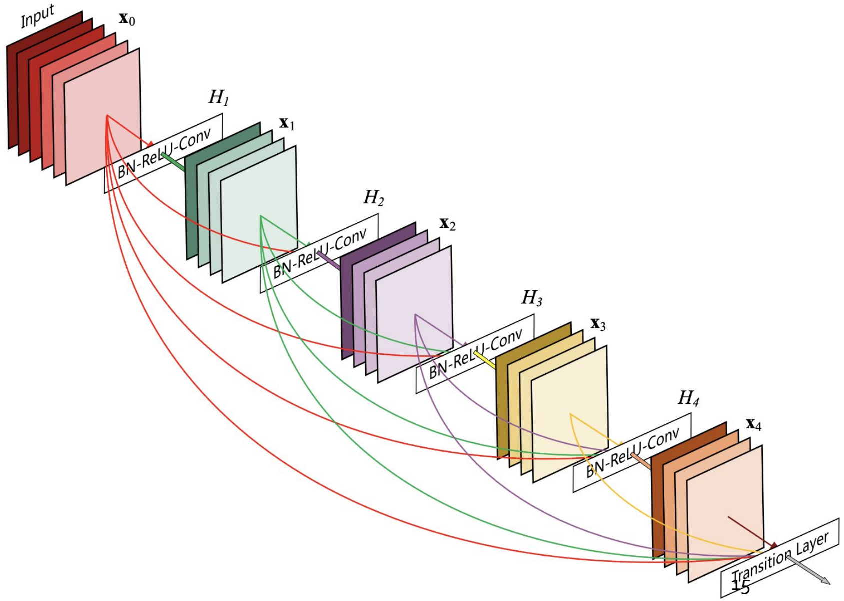
1CYCLE SCHEDULE

- Smith 2018 introduces the 1cycle schedule
  - Adjusts the learning rate and momentum of the optimiser during training
  - Original paper used linear interpolation
  - FastAI found a cosine interpolation was better, as illustrated
- Reduces training time by over 50% with no change in performance!



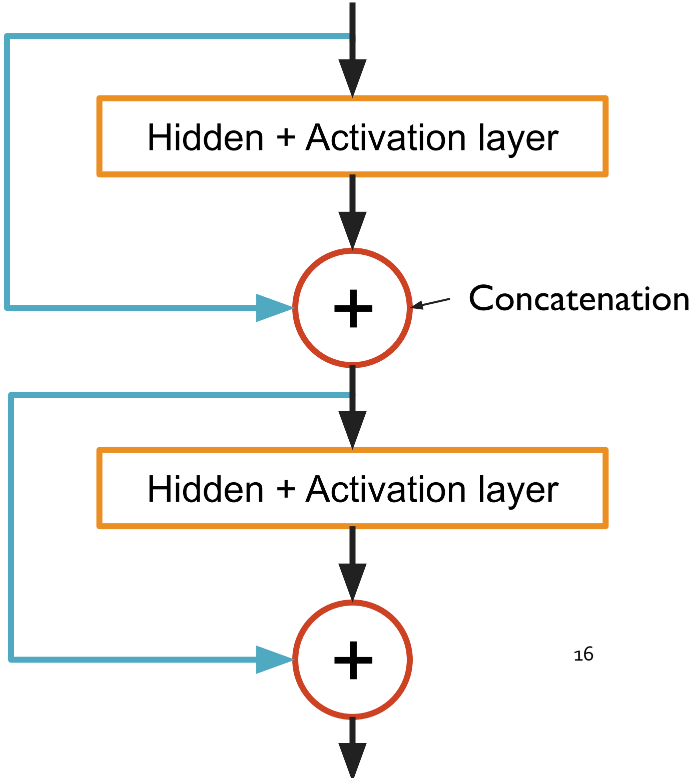
DENSE CONNECTIONS

- Huang et al. 2016 presents Densenet, a CNN architecture in which channel-wise concatenation is used to pass all the feature-maps from all previous layers to all subsequent layers
- Information is never 'lost', i.e. each layer has access to all the original inputs and weights have more direct gradient flow
- Reduces required number of free-parameters and enables 'deep supervision'



DENSE CONNECTIONS

- DNNs here are not convolutional
  - Instead use width-wise concatenation of previous hidden states
- Places less reliance on exact settings of width and depth of network layers by protecting against over-parametrisation
  - Reduced layer widths to number of inputs (33)
  - Increased number of layers to 6 (was 4)
  - Reduces number of free parameters by a third
- Provides:
  - Small performance improvement
  - Small increase in train time

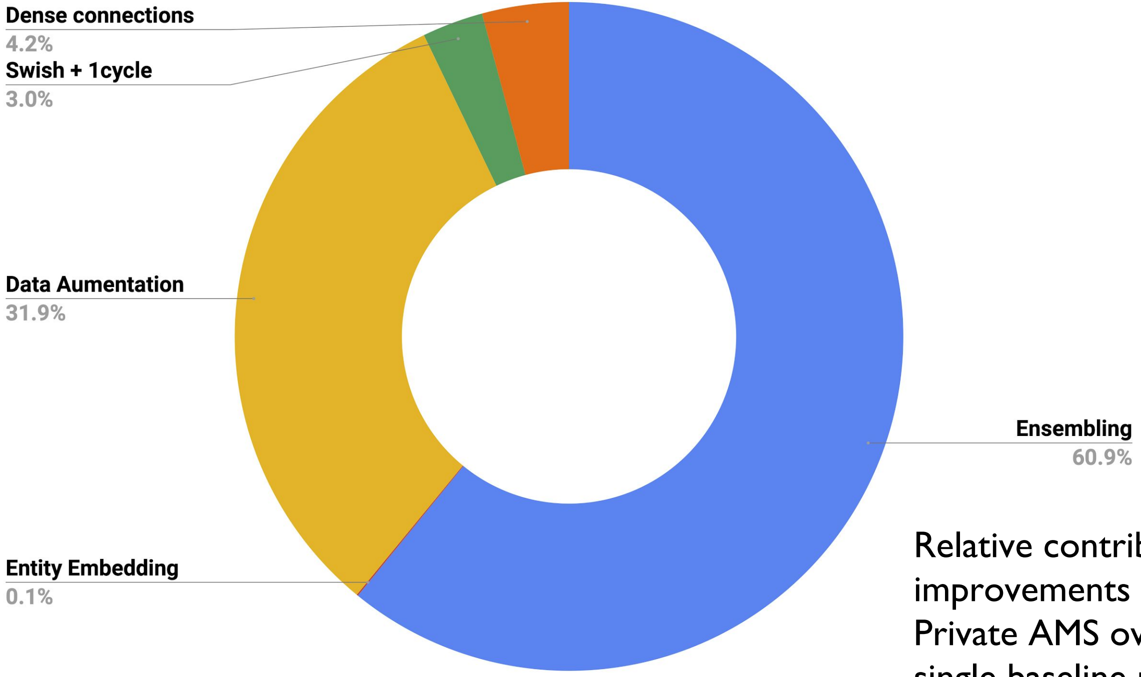


TESTING

- Model fixed and private AMS computed
  - Solution here matches 1st-place performance
- Hardware for mine:
  - GPU: Nvidia 1080 Ti
  - CPU: Intel i7-8559U (MacBook Pro 2018)
  - More hardware timings in backups
- Accounting for difference in GPU (Titan → 1080 Ti) processing power, 1st-place:
  - Trains in 100 minutes (mine 8 minutes = 92% quicker on GPU)
  - Tests in 8 minutes (mine 15 seconds = 97% quicker on GPU)
  - N.B. Doesn't include software changes (LISP→PyTorch)

	Our solution	1st place	2nd place	3rd 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 improvements to Private AMS over single baseline model

SUMMARY

SUMMARY

- Algorithms can be further improved by staying up-to-date with the field of deep-learning
- HiggsML study showed new methods:
  - Bring genuine improvements in performance
  - Reduce train and application time
  - Reduce hardware requirements: can run powerful algorithms on a laptop CPU
- Solutions developed in LUMIN (PyTorch wrapper)
  - Study code
- Accepted manuscript, Preprint (no watermark)

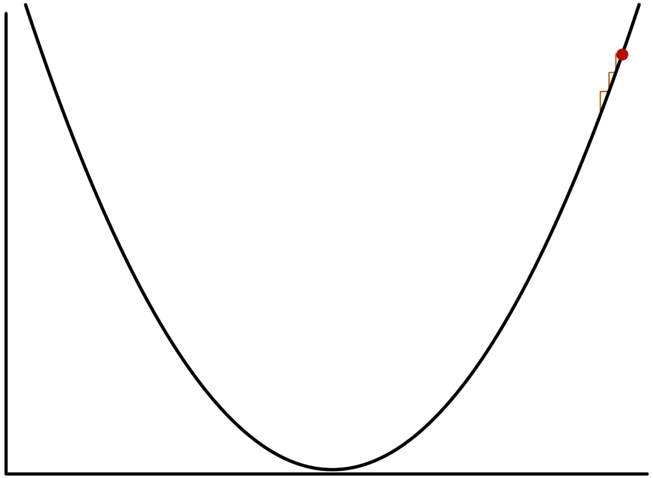
BACKUPS

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

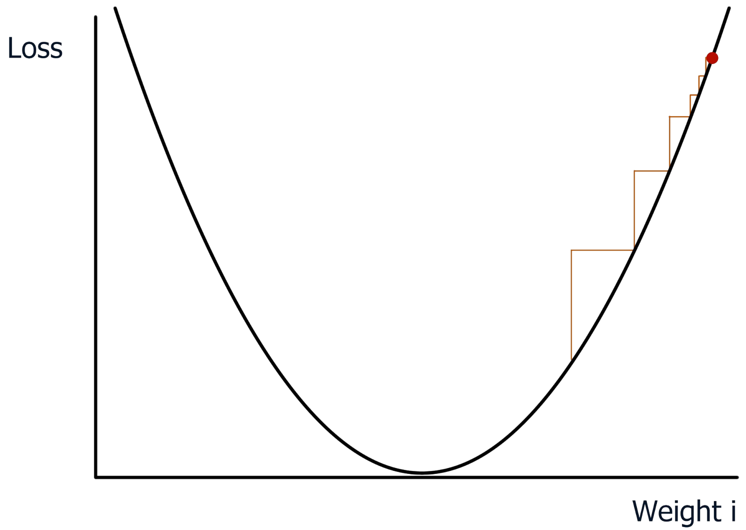
LEARNING RATE FINDER

1. Starting from a tiny LR (~1e-7), the LR is gradually increased after each minibatch



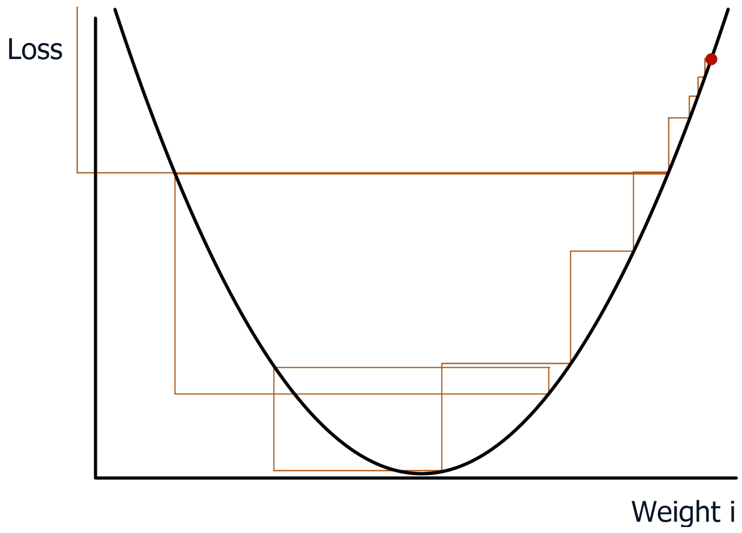
LEARNING RATE FINDER

1. Starting from a tiny LR (~1e-7), the LR is gradually increased after each minibatch
2. Eventually the network starts training (loss decreases)



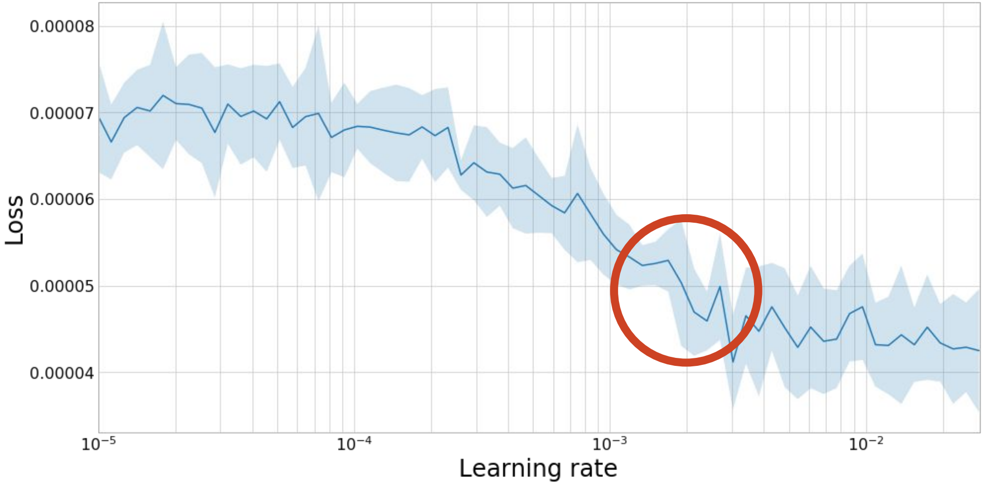
LEARNING RATE FINDER

1. Starting from a tiny LR (~1e-7), the LR is gradually increased after each minibatch
2. Eventually the network starts training (loss decreases)
3. At a higher LR the network can no longer train (loss plateaus), and eventually the network diverges (loss increases)



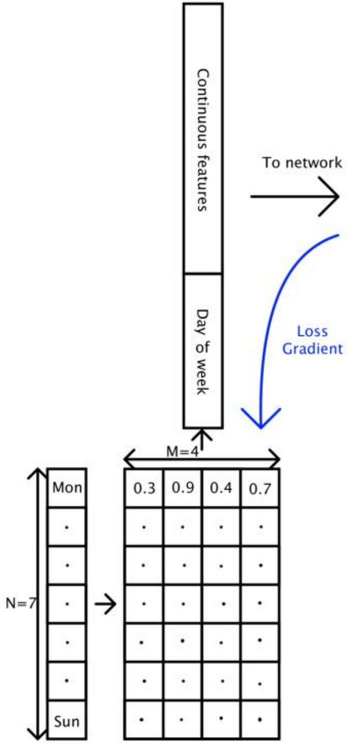
LEARNING RATE FINDER

- The optimum LR is the highest LR at which the loss is still decreasing
- Further explanation in this lesson



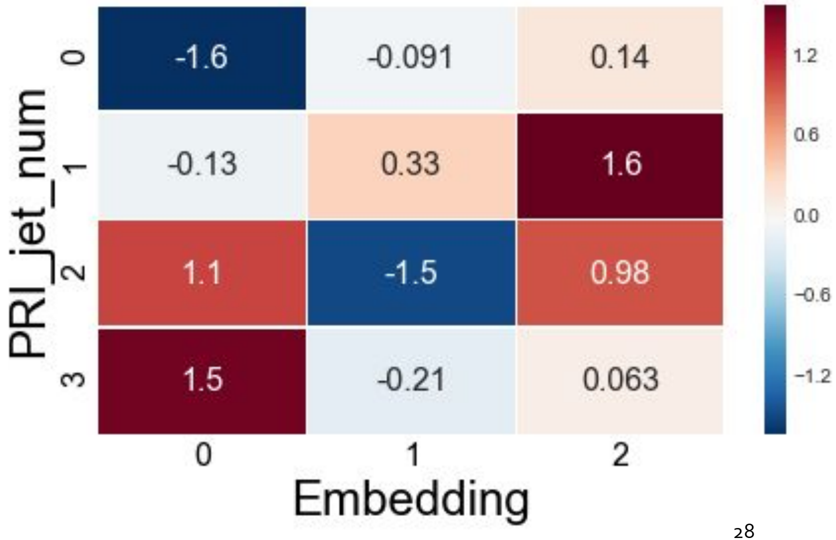
CATEGORICAL ENTITY EMBEDDING

- Categorical features = features with discrete values and no numerical comparison
- Normal to 1-hot encode as Boolean vector (Monday → 1000000)
- But potentially means a large number of extra inputs to NN (day of year = 365 inputs)
- Guo & Berkhahn 2016 learns lookup tables which provide a compact, but rich, representation of categorical values as vector of floats (Monday → 0.3,0.9,0.4,0.7)



CATEGORICAL ENTITY EMBEDDING

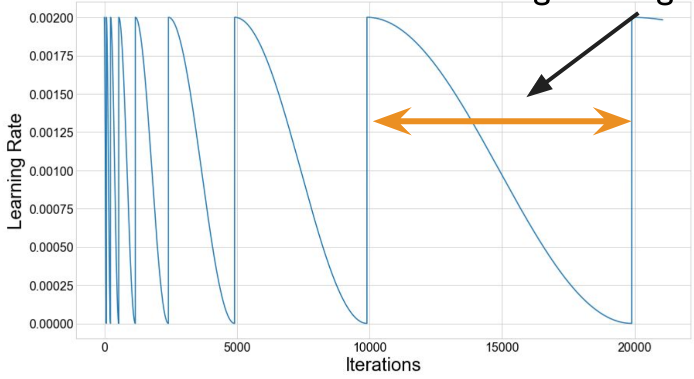
- Embedding values start from random initialisation
- Receive gradient during backpropagation and are learnt just like any other network parameter
- Embedding of the number of jets in each event gives:
  - Moderate performance improvement 3.664±0.007→3.71±0.02
  - Small increase in train & application time

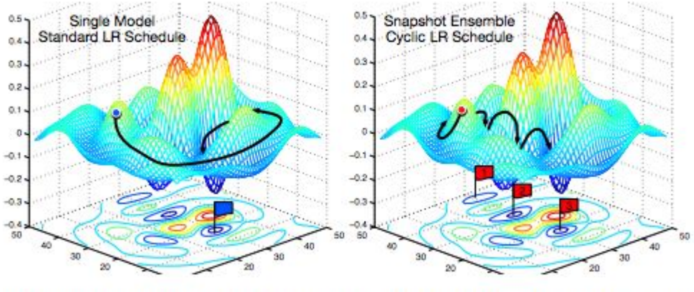


SGD WITH WARM RESTARTS

- Adjusting the LR during training is a common technique for achieving better performance
- Normally this involves decreasing the LR once the validation loss becomes flat
- Loshchilov and Hutter 2016 instead suggests that the LR should be decay as a cosine with the schedule restarting once the LR reaches zero
  - cosine annealing
- Huang et al. 2017 later suggests that the discontinuity allows the network to discover multiple minima in the loss surface

Can change cycle length during training

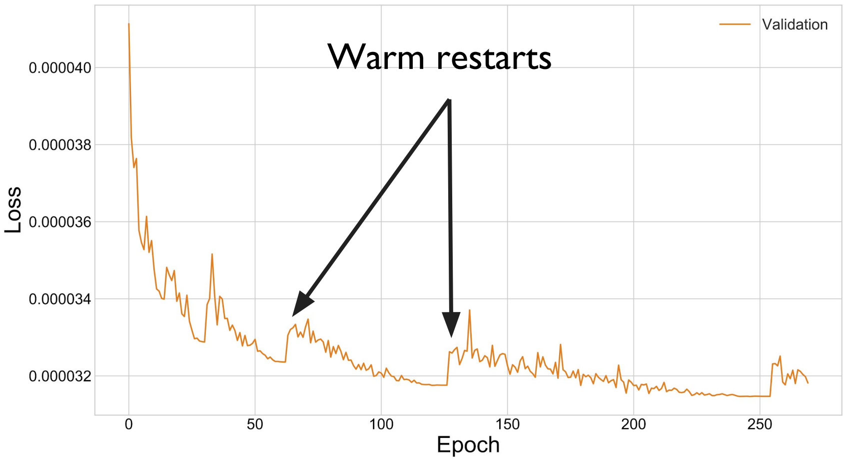




Lower figure - Huang et al., 2017, arXiv:1704.00109

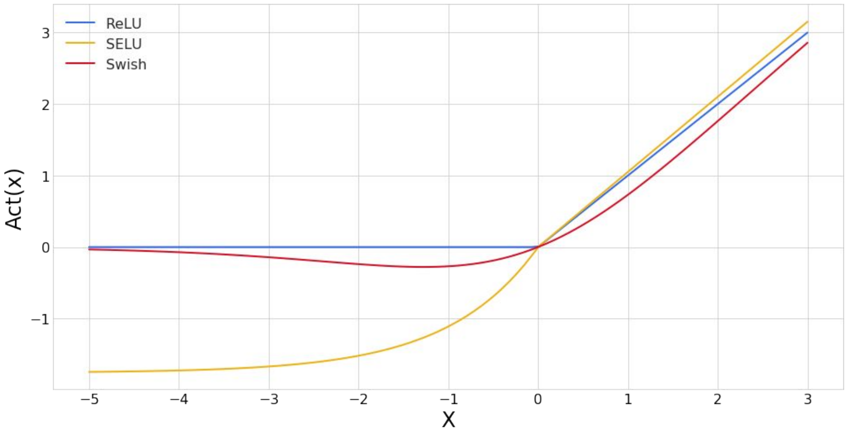
SGD WITH WARM RESTARTS

- Used cosine annealing and doubled the cycle-length with each restart
- Results in
  - Small performance improvement 3.79 ±0.01→3.80±0.02
  - Very large increase in train time (but still reasonable to use)



SWISH ACTIVATION FUNCTION

- The Swish activation function (Ramachandran et al., 2017) found via reinforcement learning
  - Provides a region of negative gradient
  - Shown to provide incremental improvement over other activation functions
- Provides:
  - Small performance improvement 3.80 ±0.02→3.81±0.02
  - Small increase in train and application time
- N.B. Had previously tested SELU (Klambauer et al., 2017) , but Swish performed better



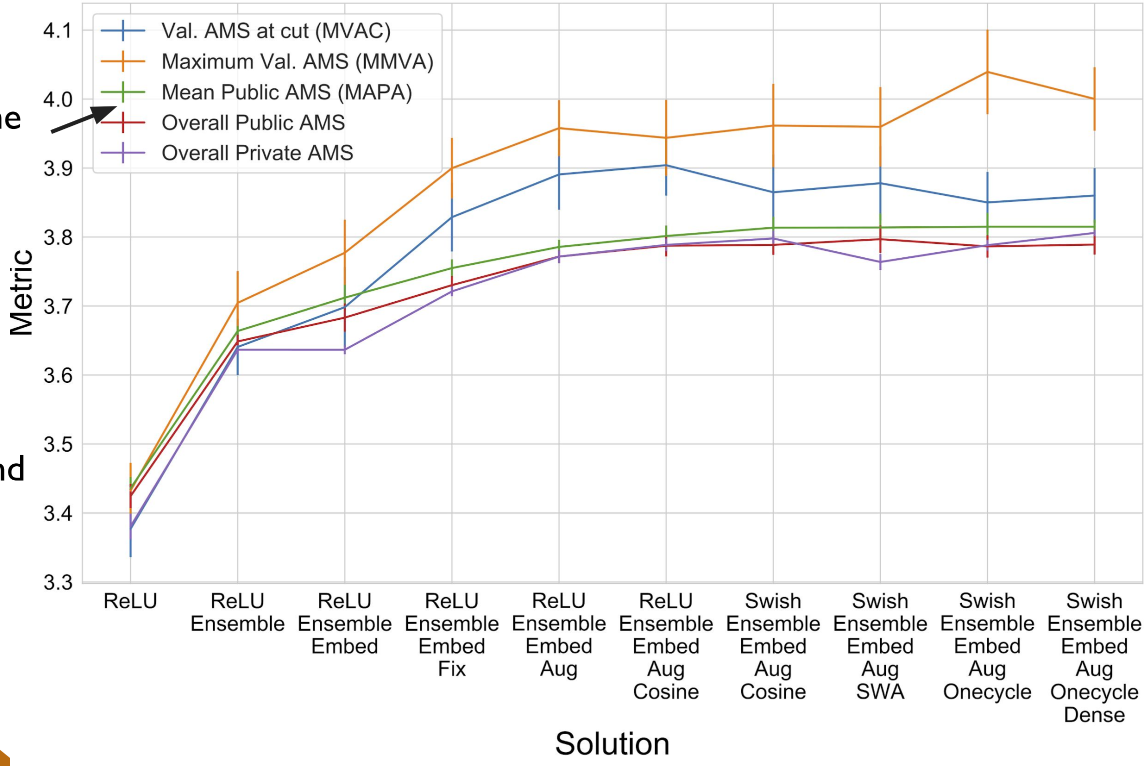
ADVANCED ENSEMBLING

- Tested several methods:
  - Huang et al. 2017 (Snapshot ensembling (SSE))
    - Produces ensembles in a single training
  - Garipov et al. 2018 (Fast geometric ensembling (FGE))
    - Produces larger ensembles in a single training
  - Izmailov et al. 2018 (Stochastic weight averaging (SWA))
    - Approximates FGE in a single model
- SWA provided reduced training time and replaced cosine annealing
  - Was then replaced by 1cycle (coming up next)
  - See Sec. 4.8 of paper for details

METRIC EVOLUTION

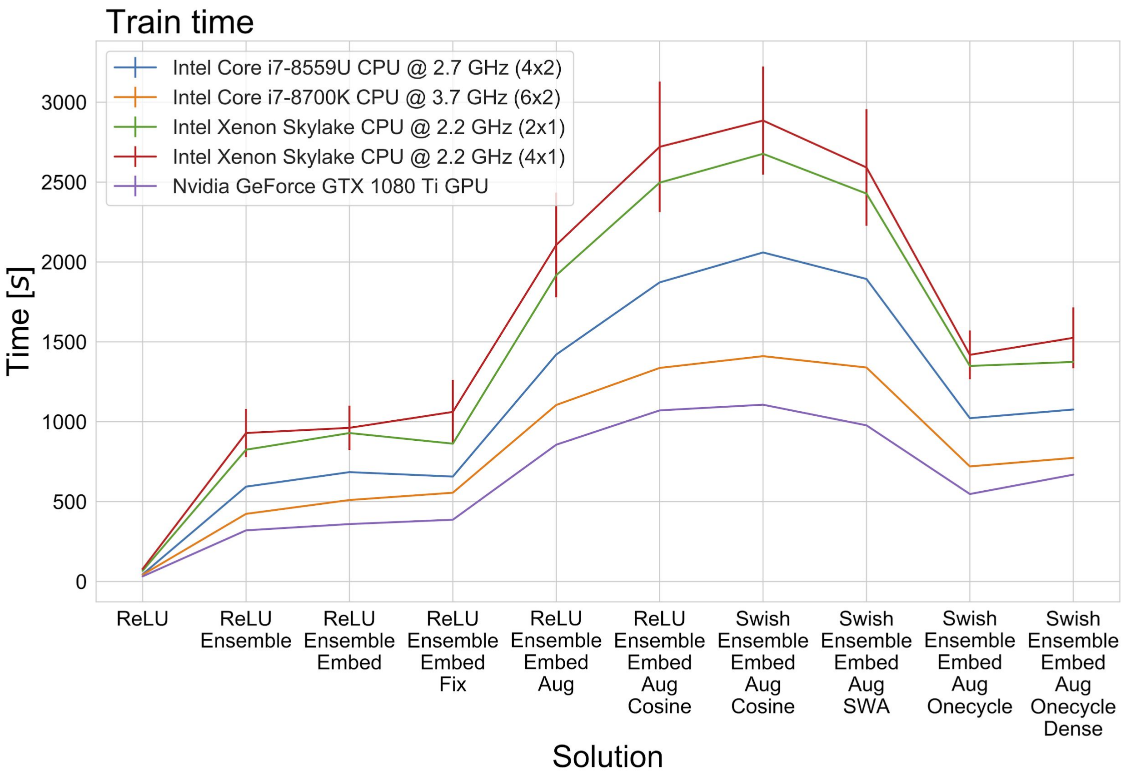
MAPA was the main optimisation metric

Overall Public|Private AMSs only checked at end

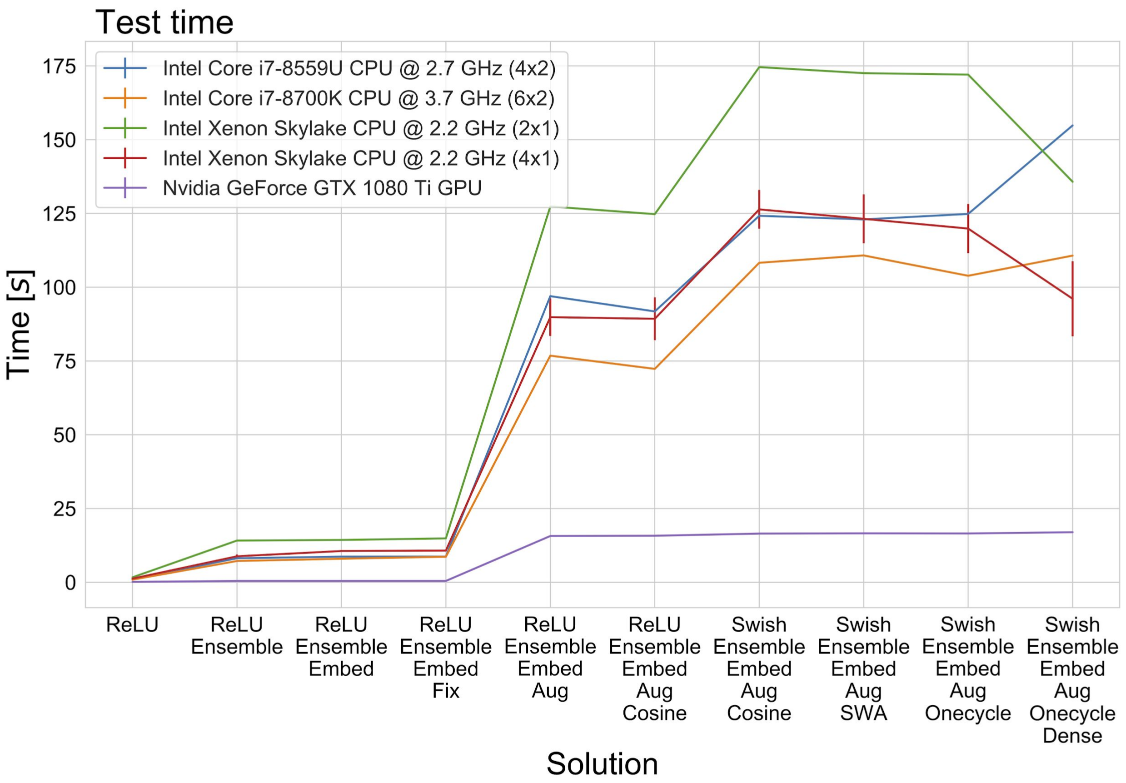


MVAC & MMVA were two other optimisation metrics, but were known to be optimistic

TRAINING TIME



TESTING TIME



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!