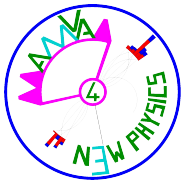
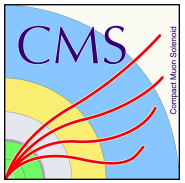






TÉCNICO
LISBOA


ATTEMPTED REPRODUCTION OF SUPER-TML METHOD

Giles Strong

3rd CMS ML Workshop, CERN - 06/19

giles.strong@outlook.com

twitter.com/Giles_C_Strong

Amva4newphysics.wordpress.com

github.com/GilesStrong

INTRODUCTION

- SuperTML: Two-Dimensional Word Embedding for the Precognition on Structured Tabular Data, March 2019, Sun et al., arxiv.org/abs/1903.06246
- Claims to get state of the art performance on the Higgs ML Kaggle challenge (simplified example of typical HEP search with strong baselines by both physicists and data scientists)
- Method is somewhat counter-intuitive
- Can their result be reproduced?

TML = Tabular Machine Learning

TRANSFER LEARNING

- In machine learning, *transfer learning* is the process of taking a model which has been trained to perform a given task on a certain dataset and using it as the basis for another model which acting on a different dataset or task
- Provided the data or task are similar enough, transfer learning is likely to provide faster convergence since many similar features will have already been learnt
- It can also mean that a more powerful model can be used on a new dataset, than would otherwise be possible given the amount of training data

TRANSFER LEARNING

- Transfer learning has a variety of common applications for certain data types:
  - Images: classification, segmentation, super-resolution
  - Text classification
- E.g. U-Net:
  - Can replace encoding path with pretrained model (e.g. ResNet)
  - Only need to learn decoder from scratch
  - Much quicker & easier to train than learning both parts from scratch

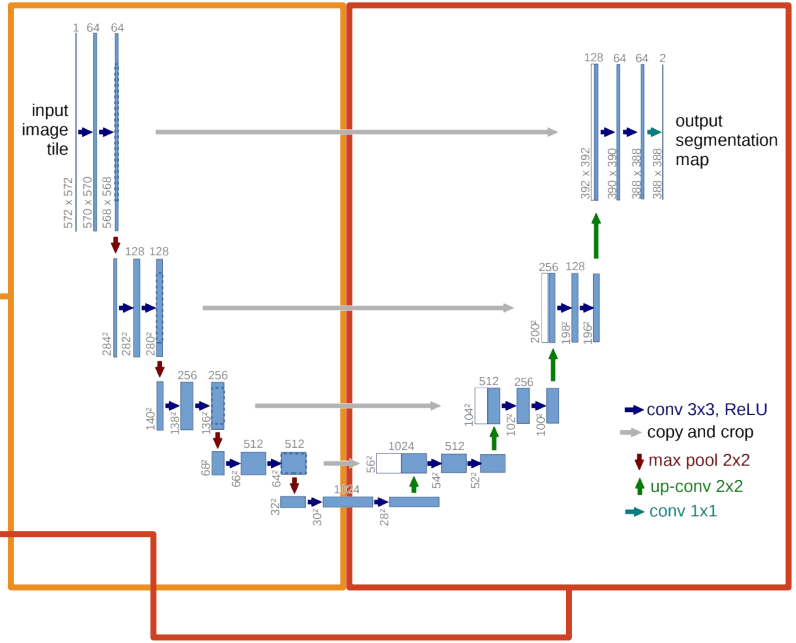


Image source: https://arxiv.org/abs/1505.04597

TRANSFER LEARNING

- Creation of the pre-trained models is normally done using a large, varied dataset, e.g. Imagenet and Wikitext
- The key point isn't that the models learn to perform a certain task well, but that in learning to perform the task they also learn task-agnostic features which are applicable to a variety of tasks, e.g:
  - Feature maps for corners, edges, textures, eyes
  - Language grammar and vocabulary

TRANSFER LEARNING

- Transfer learning for images and text is relatively easy due to:
  - The way the fields has evolved to use models which can accept a variety of input data
    - CNNs run kernels across images and adaptive max-pooling ensures the dense layers always receive their expected tensor dimension
    - RNNs can accept variable length sequences by using padding and sequence-end tokens
  - The data is all of the same format with similar *meanings*, i.e. pixel values or numerical tokens

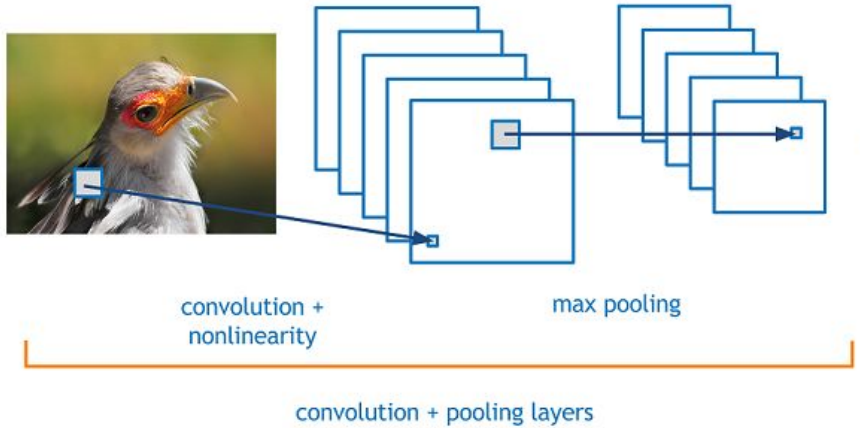


Image source: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

TRANSFER LEARNING

- Tabular data, however, can vary in terms of:
  - Numbers of features
  - Feature *meaning*, e.g. jet mass has a very different meaning to electron phi-angle
  - Whether the feature is continuous or categorical
- Applying a pretrained model to tabular data would require the same input features to used meaning that a commonly applicable model cannot be trained for all possible tabular datasets *

	0	1	2	3	4	5	6	7	8	9	...	199990
DER_mass_MMC	0.000000	0.000000	-0.495402	-0.290969	0.004018	-0.558619	0.000000	-0.608672	-0.343132	-0.309240	...	0.456663
DER_mass_transverse_met_lep	1.523702	1.189569	0.156354	-1.104284	0.123241	-1.125802	0.544919	-1.215558	-0.460711	-1.303386	...	-0.528099
DER_mass_vis	2.541261	-0.258774	-0.444344	-0.113654	0.248634	-0.822503	-0.658087	-0.391398	-0.370366	-0.104825	...	0.662791
DER_pt_h	-0.897464	-0.221461	3.283327	-0.835715	-0.154502	3.329116	0.672514	-0.027650	-0.042241	-0.903871	...	1.161695
DER_deltaeta_jet_jet	0.000000	0.000000	-0.409734	0.000000	0.000000	-0.168039	-0.662337	0.000000	0.000000	0.000000	...	0.931357
DER_mass_jet_jet	0.000000	0.000000	-0.467449	0.000000	0.000000	-0.048026	-0.677771	0.000000	0.000000	0.000000	...	0.037653
DER_prodeta_jet_jet	0.000000	0.000000	0.122037	0.000000	0.000000	0.036664	0.787169	0.000000	0.000000	0.000000	...	-0.395505
DER_deltar_tau_lep	1.797767	-0.350543	-2.087338	1.000298	-0.115392	-2.133346	0.352354	0.628401	0.006018	1.105093	...	-0.616366
DER_pt_tot	-0.808410	-0.531032	-0.797119	-0.633325	-0.097937	1.058797	2.484876	1.657872	-0.633012	-0.826575	...	1.081469
DER_sum_pt	-0.739397	-0.498472	2.602965	-0.797298	-0.250211	2.461114	-0.117803	-0.807528	-0.286888	-0.768555	...	0.864952
DER_pt_ratio_lep_tau	1.061345	-0.270989	1.156173	-0.642004	-0.157196	-0.306550	-0.263877	-0.792543	-0.390710	-0.773578	...	-1.190821
DER_met_phi_centrality	-0.999251	-0.441209	1.211129	-1.077175	0.659791	0.805586	1.291568	1.171748	1.291568	-1.069634	...	0.265978
DER_lep_eta_centrality	0.000000	0.000000	-0.865579	0.000000	0.000000	-0.579713	-1.148939	0.000000	0.000000	0.000000	...	1.356158
PRI_met_sumet	-0.874380	-0.805011	1.101020	-0.827281	0.329617	2.581124	1.525718	-0.646100	-0.549804	-1.294556	...	0.287222
PRI_jet_all_pt	-0.745191	-0.336127	1.991812	-0.745191	-0.424295	2.419050	0.158949	-0.745191	-0.160587	-0.745191	...	0.771600
PRI_tau_px	0.367290	0.895332	1.750334	-0.713812	-0.422239	2.355970	-0.328423	-0.738359	-0.281934	-0.863724	...	2.599004
PRI_tau_py	-0.061403	-0.018709	0.738016	-0.865355	0.590245	-0.955789	-0.582495	-0.628173	0.079096	-1.124600	...	0.616154
PRI_tau_pz	-0.921071	-0.565094	0.212954	0.446815	-0.055135	-0.549117	-0.188430	-0.613286	-0.041830	0.575070	...	-0.965968
PRI_jet_subleading_px	0.000000	0.000000	-0.604195	0.000000	0.000000	0.875979	0.956430	0.000000	0.000000	0.000000	...	-0.725265
PRI_jet_subleading_py	0.000000	0.000000	0.059578	0.000000	0.000000	-0.762735	-0.133058	0.000000	0.000000	0.000000	...	-0.302338
PRI_jet_subleading_pz	0.000000	0.000000	-0.313236	0.000000	0.000000	-0.435841	-0.204145	0.000000	0.000000	0.000000	...	-0.209193
PRI_jet_leading_px	0.000000	0.987067	-2.310585	0.000000	0.590479	-2.939375	-0.043377	0.000000	0.416243	0.000000	...	-0.027400

* A possible limited use in HEP could be pretraining a model on Delphes and refining on GEANT

SUPER TML

- SuperTML: Two-Dimensional Word Embedding for the Precognition on Structured Tabular Data, March 2019, Sun et al., arxiv.org/abs/1903.06246
- Proposes to transform tabular data into images by printing feature values as text on black backgrounds, either with fixed font-size or varying font-size with feature importance
- Pretrained CNNs can then be refined to perform the desired task as normal

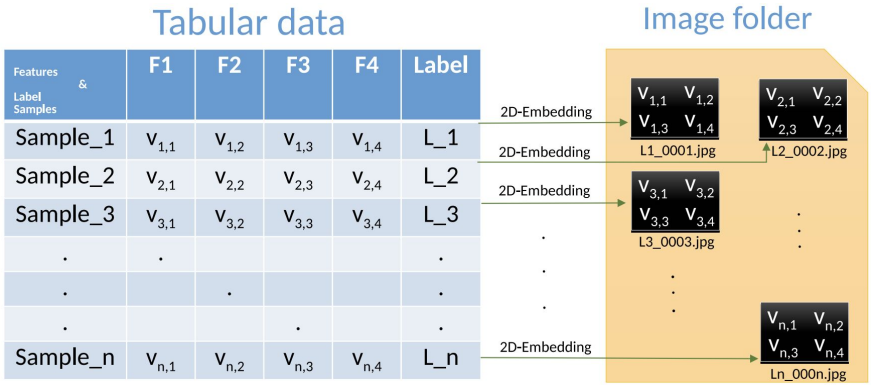


Image source: arxiv.org/abs/1903.06246

EXPERIMENTS

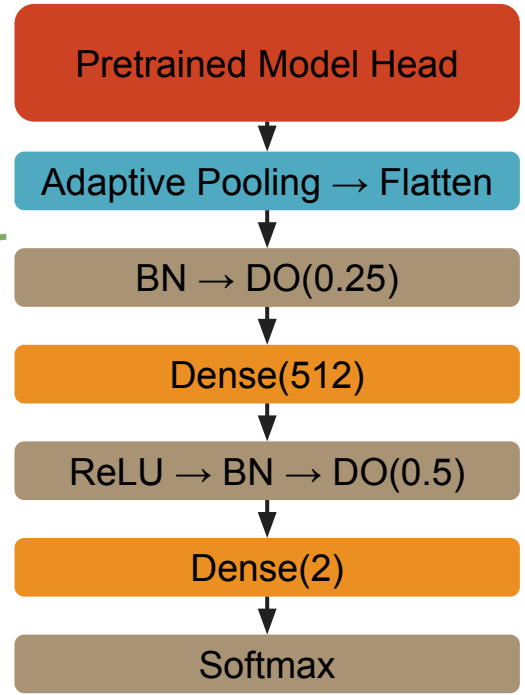
- The paper considers four datasets:
  - Iris: classification of iris species using data of their petals and sepals
  - Wine: classifying wine types by their properties
  - Adult: predicting whether someone earns more then $50,000 based on personal data
  - Higgs ML: Maximising the Approximate Median Significance for Higgs boson discovery in the di-tau channel using final-state properties
- In each case, the authors claim to beat the previous state of the art
- Of greatest interest to us is the Higgs ML score, where they claim an AMS of 3.979 (c.f. winning score 3.806)

HIGGS ML SOLUTION

- Their best solution uses:
  - 224x224 images
  - Fixed font-size
  - Squeeze-Excitation Net 154 (foundation of winning entry of Imagenet 2017)
    - SE block is effectively a plugin for architectures which learns weights for every feature map, rather than treating them all equally.
- Given the huge difference between their result and the winning solution, and the somewhat counter-intuitive method, it is necessary to see whether it can be reproduced

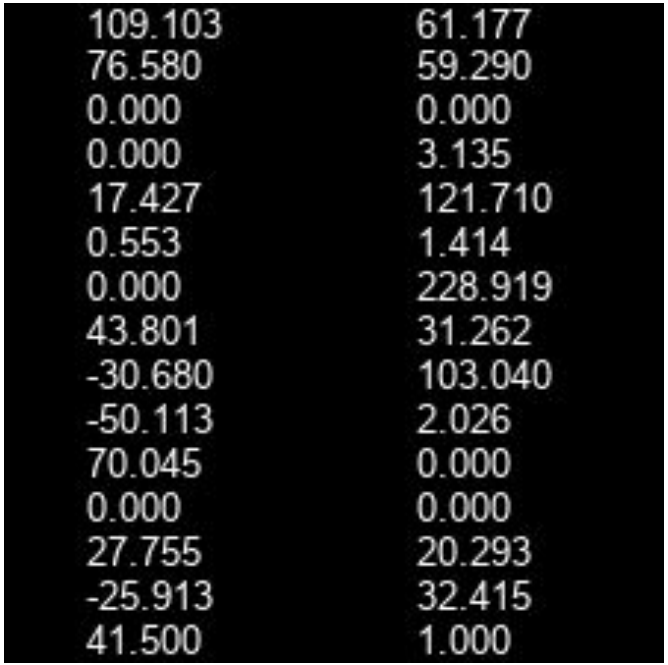
MODEL TRAINING

- Models trained in Fastai (PyTorch wrapper)
- Images adjusted to account for Imagenet channel means and standard deviations
- Unweighted categorical cross-entropy
- Final dense layer of pretrained model removed and replaced with two dense layers with two output nodes
- Two step process:
  - First only the final two dense layers are trained (4 epochs with 1-cycle policy, LR set using LR finder)
  - Next, the entire network is trained (4 epoch 1-cycle, LR set using LR finder, and scaled according to layer depth (discriminative LR))



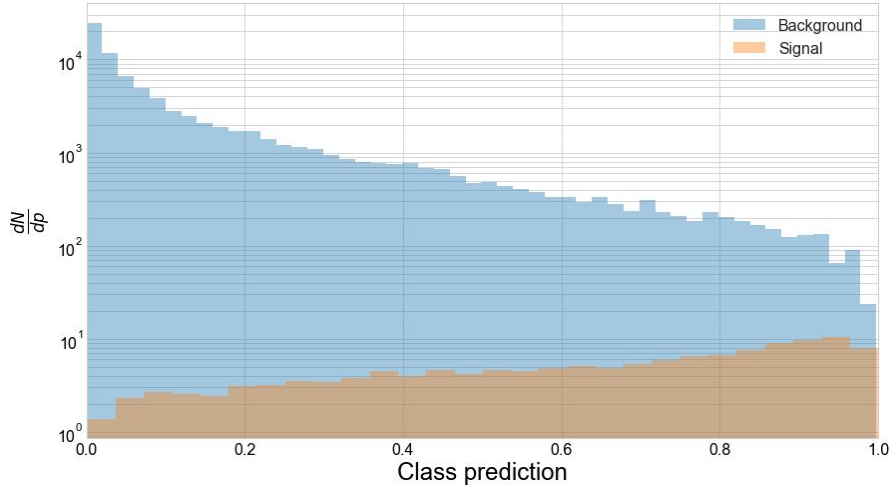
INITIAL ATTEMPT

- Settings:
  - 224x224 images
  - Equal font-size
  - ResNet34
- Surprisingly, it seemed to learn something
- Validation AMS around 3
- Test AMS around 2.75
  - N.B. Test set checked each time; we're trying to confirm a result rather than develop a model
- Train-time about 47 minutes
- Data size: 13 GB



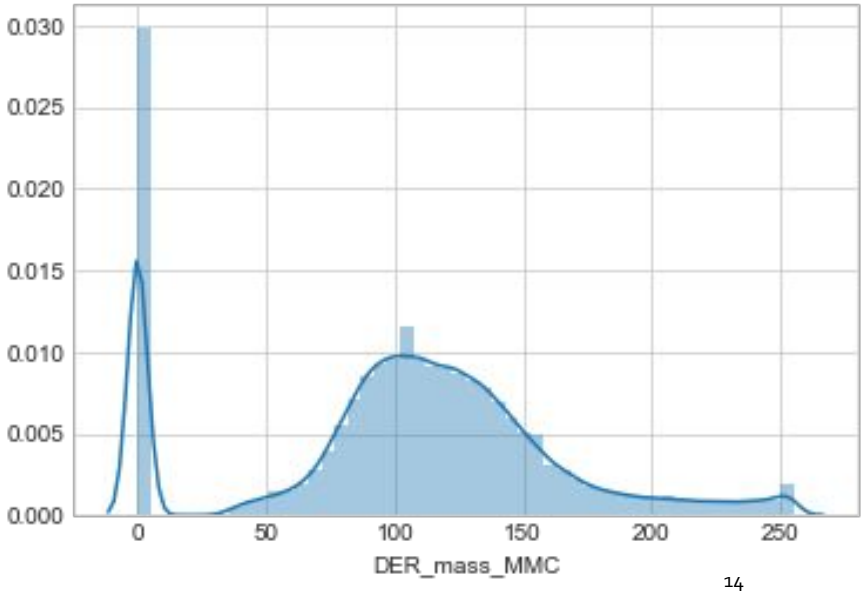
VAL. PREDICTION DISTRIBUTION

- Looks relatively smooth
- Clustered not flat or scattered



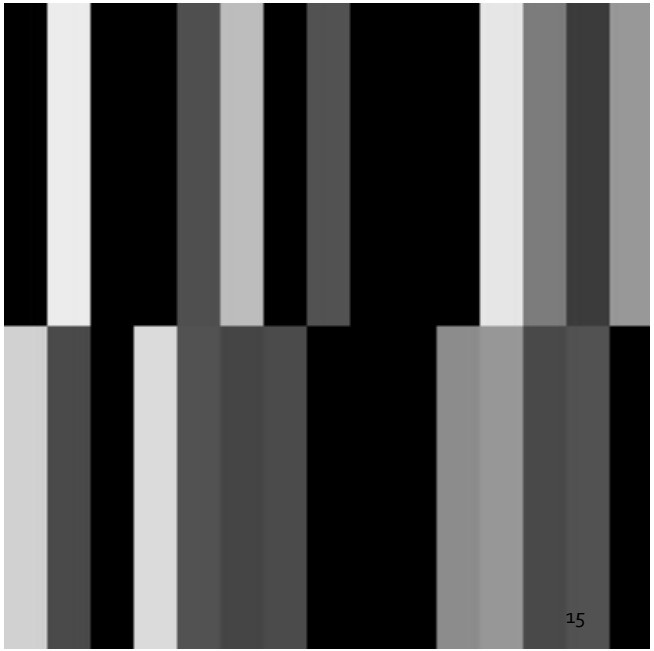
ENCODING CONSIDERATIONS

- Taking numerical information and encoding it as an abstract representation (written digits) and requiring the model to reinterpret the text pixels back into numerical information seems inefficient
- The features could instead be encoded as blocks of pixels with the same intensity
- This can be achieved by:
  - Standardising and normalising data (set mean to zero and standard deviation to one)
  - Passing values through a sigmoid (soft-clamps values between zero and one)
  - Timesing values by 255 and converting to integer



PIXEL ENCODING

- To be as similar as possible to text approach, blocks set to be thin rectangles in two columns
- Data was quicker to produce and file size dropped to 9 GB
- Large improvements to both validation and test AMS: 3.62, 3.3 respectively
- Also tried data augmentation by randomly varying pixel intensity, but it didn't improve



SE-NET 50

- Moved to using the smallest squeeze-excitation pretrained model, SE-net 50
- Model size meant batch size had to be halved - train-time → 2.3 hours
- Validation AMS of 3.71
- Public : Private AMS: 3.35 : 3.48

SE-NET 154

- Moved to using the same model they used
- Again, batch-size had to be reduced
- Train-time = 12 hours
- Validation AMS of 3.72
- Public : Private AMS at cut around 3.5

FURTHER DATA CONSIDERATIONS

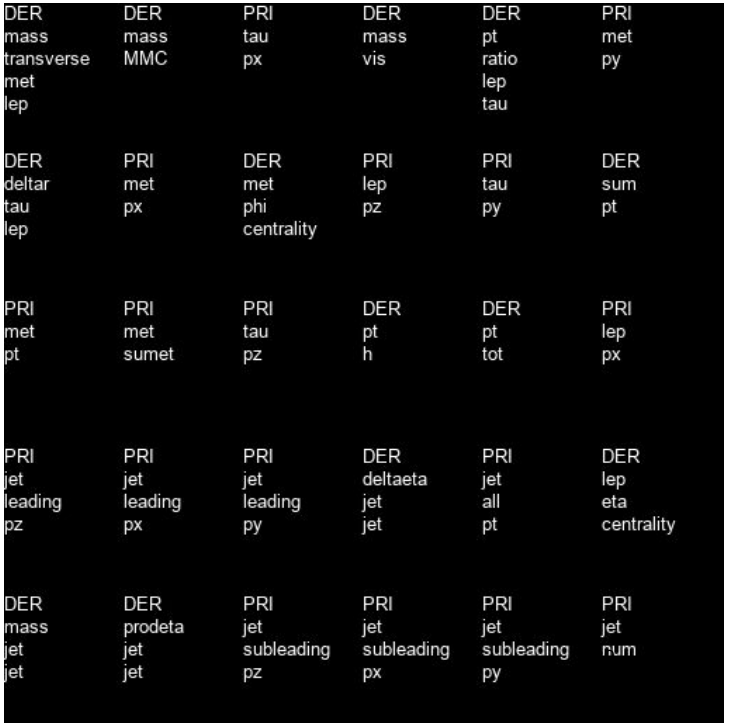
- Unlike Imagenet data, there's no way the data here can contain more information by increasing the resolution beyond a certain point
- Stride-2 (halves data dimensions) convolutions in pretrained model, however, can mean that performance can still degrade
- Still, interesting to see whether the resolution can be reduced below the common size of 224

SMALLER IMAGES

- This rectangles changed to approximate squares
- Image size reduced to 56x56
- Data size → 3.1 GB
- Moved back to SE-net 50
- Was able to move to larger batch size (512), train-time → 15 minutes
- Slight performance drop:
  - Validation AMS 3.66
  - Test AMS between 3.3 - 3.5
- Have also tried balancing data classes by copying signal data once - provides slight improvements

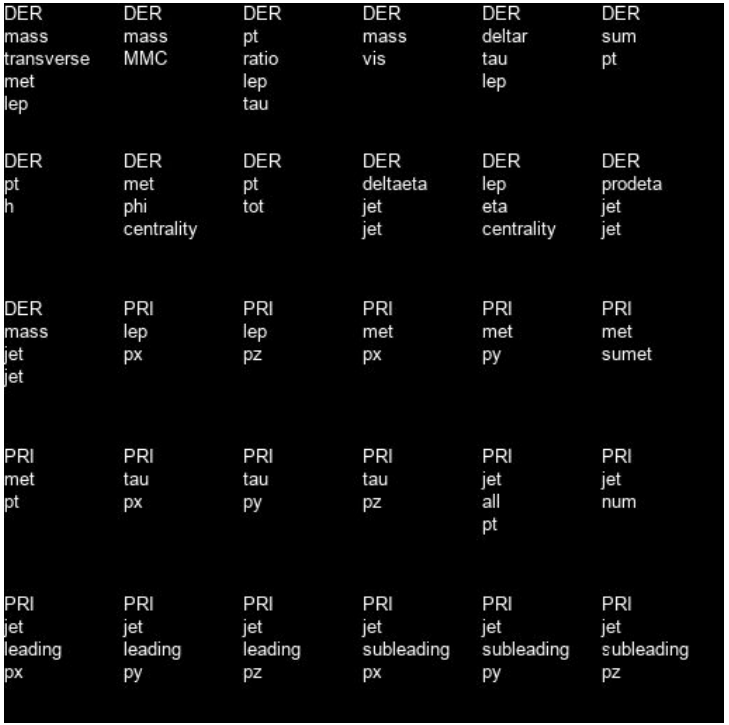
FEATURE ORDERING

- Feature ordering has a heavy impact on performance; randomising order yields very different trainings
- Default order works well:
  - High-level features are sequential
  - As are 3-momenta for each particle
- Paper doesn't say how they order features
- Ordering by permutation importance seems to be a reliable method
- Hypothesis: training is easier (possible) when filters can cover several important features at the same position



FEATURE ORDERING

- Possible downside of ordering all features is the potential limitation of the NN to utilise low-level information (importance computed by Random Forests)
- Alternative:
  - Order high-level features
  - Prescribe low-level ordering maximising filter overlap between final-state 3-momenta
- Rough testing seems to indicate similar performance compared to ordering all features



CURRENT STATUS

- Have tried a variety of encoding methods and models and have not been able to recover their 3.979 score
- Github repo with solutions available here: https://github.com/GilesStrong/SuperTML_HiggsML_Test
- Running solutions requires a high capacity GPU (e.g. Nvidia 1080 Ti)

POSSIBLE EXPLANATIONS - TEST SIZE

- Could the result be obtained on a smaller reweighted test set?
- Paper text is inconsistent:
  - Section 1 says the test size is 550,000 events (full Kaggle set)
  - Section 3.4 states the size test size is 55,000 events
- Splitting and reweighting the test data into 10 random subsamples and checking the AMSs at the chosen cut finds that a maximum AMS of 3.94 public and 3.67 private can be achieved using senet154

POSSIBLE EXPLANATIONS - MAXIMISE AMS ON TEST SET

- Could the paper have maximised the AMS on the test set?
- Entrants were required to provide Boolean class predictions for test data, i.e. signal/background cut had to be chosen using validation data
- However the paper uses the CERN OpenData version which allows the AMS to be optimised on the test set
- Again, splitting the test data and allowing the AMS to be optimised each time shows a maximum public : private AMS of 5.79 : 3.87 can be achieved

DISCUSSION

- The paper method (text encoding) seems questionable:
  - How should numbers be rounded?
  - Do other numeral systems work?
- Presumably the inefficiency of the data transforming is offset by the benefits of transfer learning
  - But what does it **really** learn on Imagenet which is useful to be transferred?
    - Very little: training from scratch gets to about the same performance in same number of epochs
- Encoding as pixel intensities seems experimentally to be better than text
- Current reimplementation performance does not indicate improvement over traditional methods

DISCUSSION

- If the claimed improvement is realised, the method would allow the benefits of transfer learning to be applied to most data analysis work at CERN
- However:
  - Best performance (so far) takes much longer to achieve compared to traditional methods, and requires specialised hardware
  - Data storage requirements are much higher (but images could be generated on demand)
  - Size and train-time can be reduced at cost of performance (reducing the benefit compared to easier to implement methods), but useful for tuning setup quickly