Automation of CMS workflow recovery

Hamed Bakhshiansohi, Dirk Kruecker
& Tools and Integration group @ CMS

2nd Round Table on

Machine and Deep Learning at DESY

29 November 2019

Introduction

- CMS simulation and data processing are organized in “workflow” tasks, each with thousands jobs
- Workflows are interrupted due to common errors in grid jobs
- There are some workflows that can not be recovered by “Unified”
- Currently all handled manually by an operator
  - Looking into the error codes and site statuses
  - Take an action, among a few possible actions
- ML seems a natural solution to help the operator and automate the procedure

How does the operator decide ?

- Matrix of the number of each error in each site
- Site status at the time the workflow was reported as ‘needing-assistance’
- Log/Err files of failed jobs? If needed

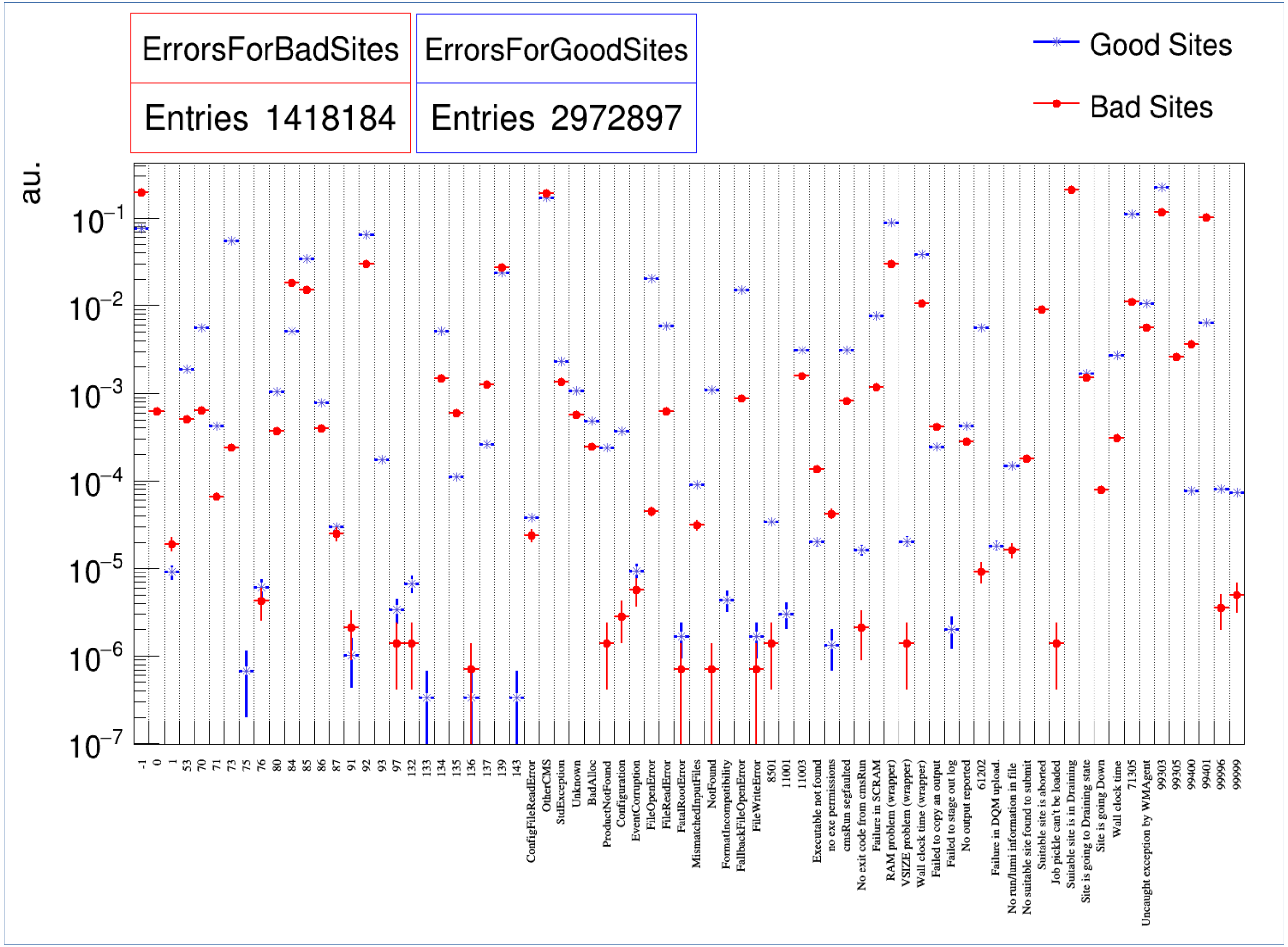
To top

	T0_CH_CERN	T1_DE_KIT	T1_ES_PIC	T1_FR_CCIN2P3	T1_IT_CNAF	T1_RU_JINR	T1_UK_RAL	T1_US_FNAL	T2_CH_CERN	T2_CH_CERNBOX	T2_CH_CERN_HLT	T2_DE_DESY	T2_ES_IFCA	T2_FR_GRIF_IRFU	T2_FR_GRIF_LLR	T2_IT_Legnaro	T2_UK_London_Brunel	T2_UK_London_IC	T2_UK_SGrid_RALPP	T2_US_Florida	T2_US_MIT	T2_US_UCSD	T2_US_Wisconsin	T3_US_FNALLPC	null
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
84	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
85	0	2	0	0	0	1	6	0	0	0	8	1	0	0	0	0	0	0	0	0	0	1	0	0	0
92	0	0	1	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	2	0	0	0	0	0	0
134	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
139	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8001	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
50110	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50660	0	0	3	2	4	1	1	1	6	0	63	1	0	1	0	1	0	1	0	0	1	0	1	0	0
50664	0	0	2	7	0	0	2	18	0	0	32	0	0	7	0	0	4	0	2	0	0	0	0	0	0
71304	0	0	0	0	0	0	0	0	0	0	102	0	0	0	0	0	0	0	0	0	0	0	0	0	0
99305	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96

Possible Actions :

- ACDC : A partial retry of a workflow, it retries only failed jobs
  - Helpful in most of the cases
- Kill and Clone
  - With new splitting
  - New settings for memory and cores
- On-hold and by-pass : very rare use cases

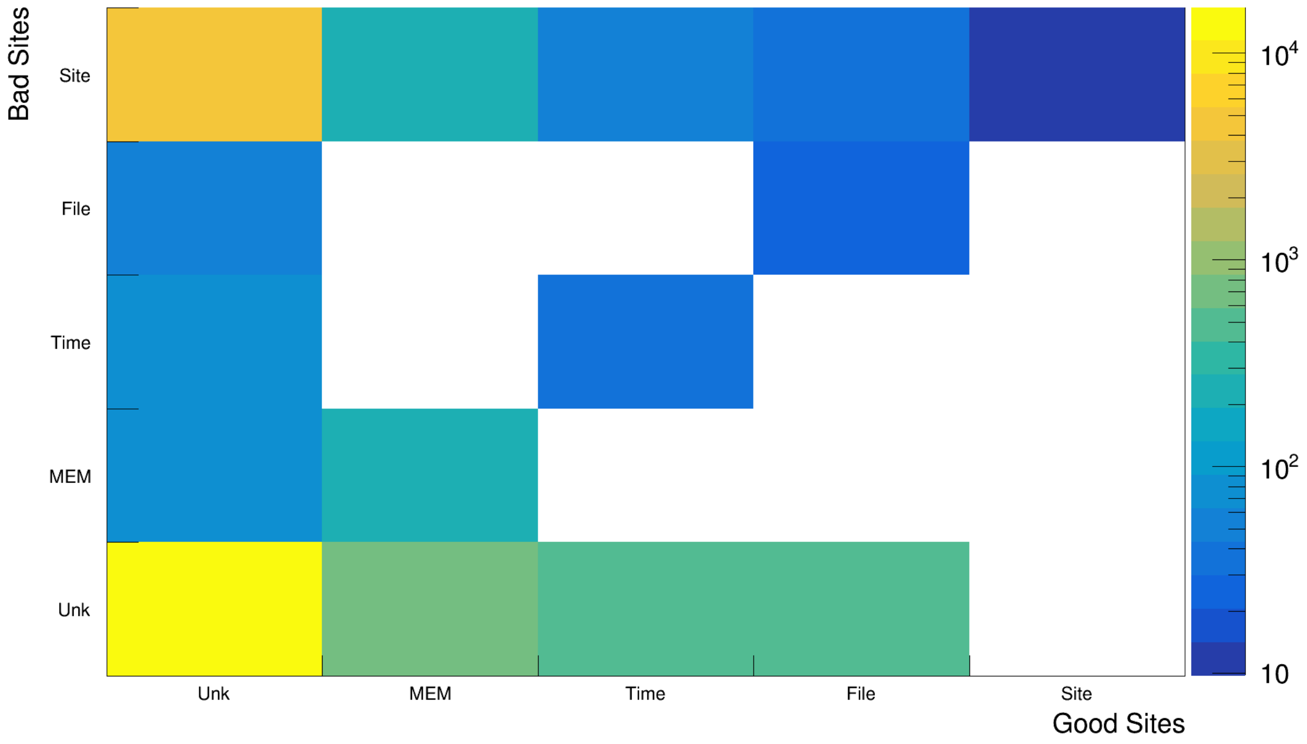
Input dataset : Error codes



- ~27K recorded actions since 2017
- Error codes:
  - 67/66 different error codes for good/bad sites
  - ~90% overlap
  - 74 in total
- Error codes are described in twiki:JobExitCodes
  - Errors are categorized according to the description
  - [MEM, FILE, TIME, SITE, Others]

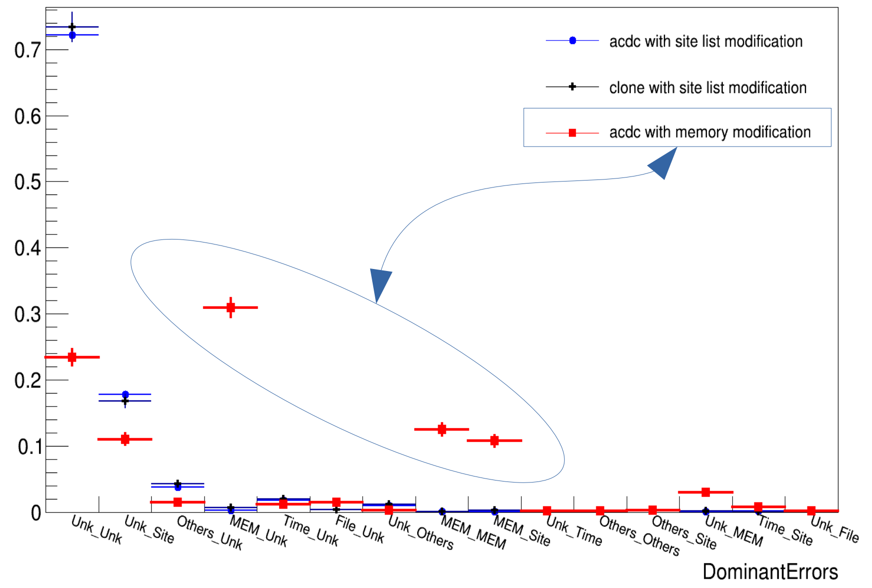
Dominant error in good/bad sites

- Site related errors happen rarely in “good sites”
- Memory, CPU time, File related errors are ~symmetric in good/bad sites



Actions

action	splitting	Site list	memory	rate
ACDC	None	Modified	Not set	87.53%
clone	None	Not modified	Not set	4.61%
ACDC	None	Modified	> 20GB	3.28%
ACDC	None	Modified	> 10GB	1.12%
ACDC	None	Modified	< 10GB	0.75%
ACDC	10x	Modified	Not set	0.74%
Other actions (27 more rows)				< 2%



- Obvious correlation between dominant error codes and memory settings
- Similar trend for ACDC/clone actions

Tools and framework

- TensorFlow 2.0.0 and the embedded Keras are used for training
- Data split

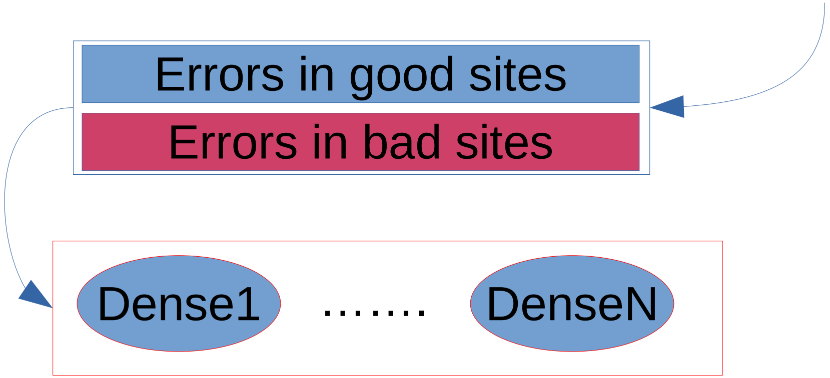
- Talos: hyperparameter optimization
  - nLayers, nNeurons, Activation functions, batch_size, L2 regularization
  - Learning rate is the most important optimized parameter, as expected

Simplest approach: ignore site names

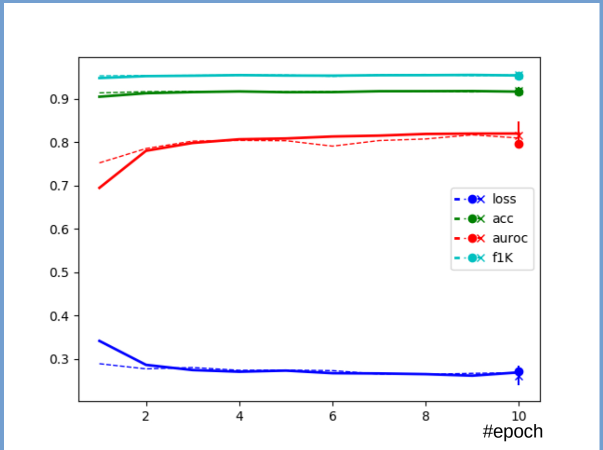
- Sum errors over sites
  - separated by site-status
- Binary [ACDC(sites modified), others] classification
- Weighted and normal cross entropy loss function
- Optimized networks
  - Unweighted:
    - 4 layers each with 50 neurons
    - batch_size = 500
  - Weighted:
    - 6 layers each with 100 neurons
    - L2 Regularization for some layers
    - batch_size = 5k

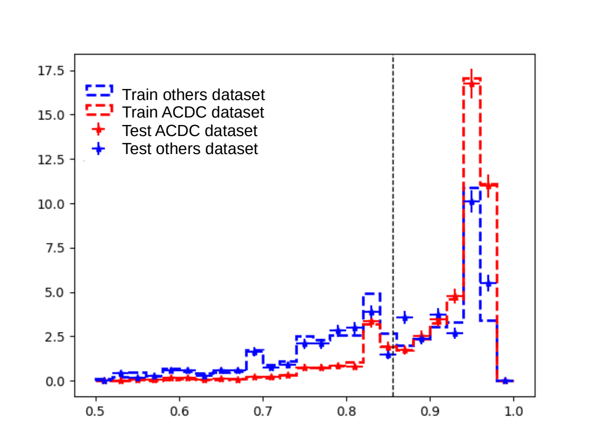
To top

	T0_CH_CERN	T1_DE_KIT	T1_ES_PIC	T1_FR_CCIN2P3	T1_IT_CNAF	T1_RU_JINR	T1_UK_RAL	T1_US_FNAL	T2_CH_CERN	T2_CH_CERNBOX	T2_CH_CERN_HLT	T2_DE_DESY	T2_ES_IFCA	T2_FR_GRIF_IRFU	T2_FR_GRIF_LLR	T2_IT_Legnaro	T2_UK_London_Brunel	T2_UK_London_IC	T2_UK_SGrid_RALPP	T2_US_Florida	T2_US_MIT	T2_US_UCSD	T2_US_Wisconsin	T3_US_FNALPC	null
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
84	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
85	0	2	0	0	0	1	6	0	0	0	8	1	0	0	0	0	0	0	0	0	0	1	0	0	0
92	0	0	1	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	2	0	0	0	0	0	0
134	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
139	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8001	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
50110	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50660	0	0	3	2	4	1	1	1	6	0	63	1	0	1	0	1	0	1	0	0	1	0	1	0	0
50664	0	0	2	7	0	0	2	18	0	0	32	0	0	7	0	0	4	0	2	0	0	0	0	0	0
71304	0	0	0	0	0	0	0	0	0	0	102	0	0	0	0	0	0	0	0	0	0	0	0	0	0
99305	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96

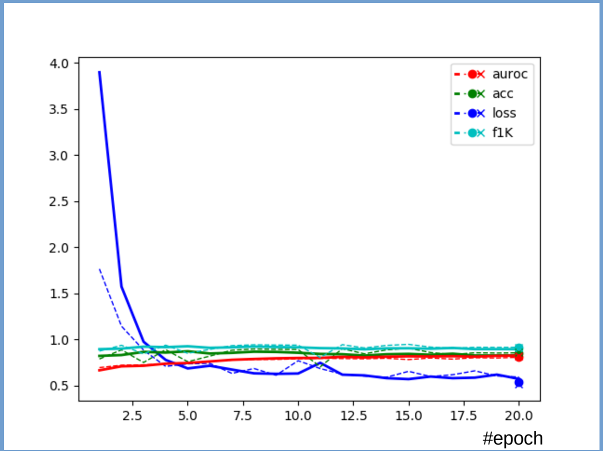


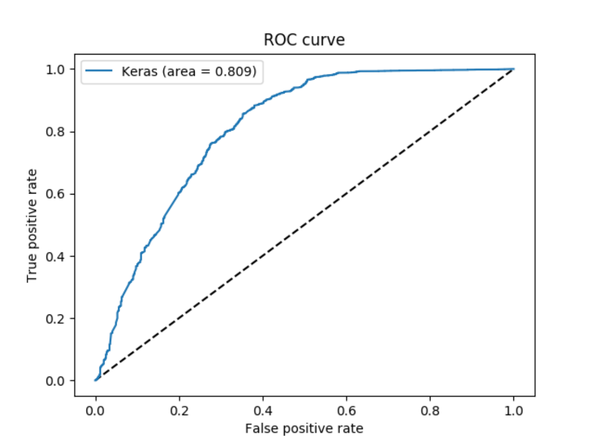
Unweighted
Stable results
Accuracy ~ 90%
AUROC ~ 80%





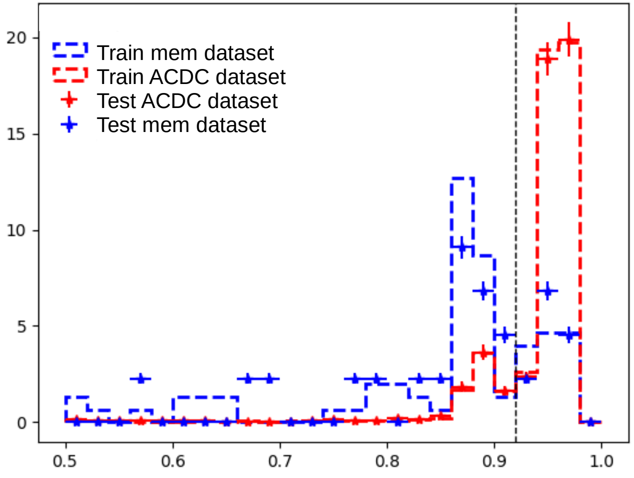
Weighted
No significance improvement

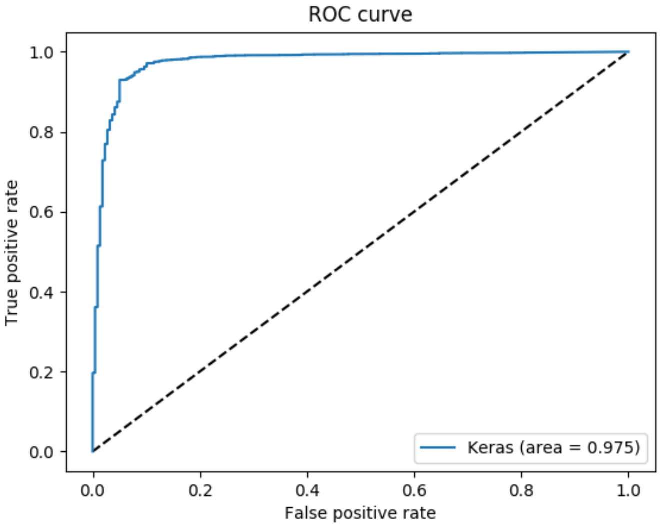




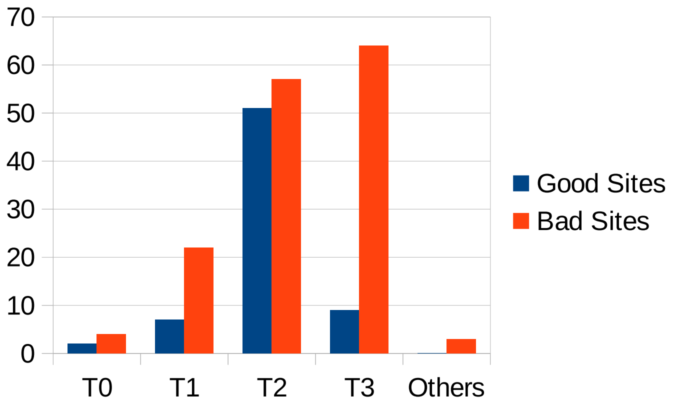
Predict if MEMORY re-configuration is needed

- Binary classification : ACDC(with memory configuration) vs. others
- Weighted cross entropy loss function
- After optimization:
  - AUROC : 97.5%





- Add Tier information to the input matrix for training.

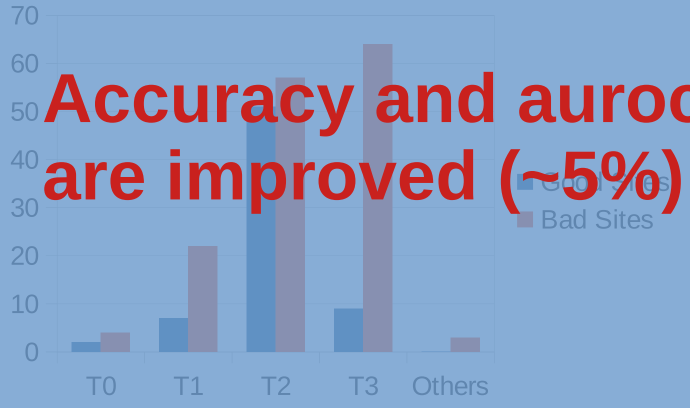


- Binary classification : [ACDC(sites modified), others]
- Weighted loss function
- Optimized Network

To top

	T0_CH_CERN	T1_DE_KIT	T1_ES_PIC	T1_FR_CCIN2P3	T1_IT_CNAF	T1_RU_JINR	T1_UK_RAL	T1_US_FNAL	T2_CH_CERN	T2_CH_CERNBOX	T2_CH_CERN_HLT	T2_DE_DESY	T2_ES_IFCA	T2_FR_GRIF_IRFU	T2_FR_GRIF_LLR	T2_IT_Legnaro	T2_UK_London_Brunel	T2_UK_London_IC	T2_UK_SGrid_RALPP	T2_US_Florida	T2_US_MIT	T2_US_UCSD	T2_US_Wisconsin	T3_US_FNALLPC	null
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
84	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
85	0	2	0	0	0	1	6	0	0	0	8	1	0	0	0	0	0	0	0	0	0	1	0	0	0
92	0	0	1	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	2	0	0	0	0	0	0
134	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
139	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
8001	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
50110	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50660	0	0	3	2	4	1	1	1	6	0	63	1	0	1	0	1	0	1	0	0	1	0	1	0	0
50664	0	0	2	7	0	0	2	18	0	0	32	0	0	7	0	0	4	0	2	0	0	0	0	0	0
71304	0	0	0	0	0	0	0	0	0	0	102	0	0	0	0	0	0	0	0	0	0	0	0	0	0
99305	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96

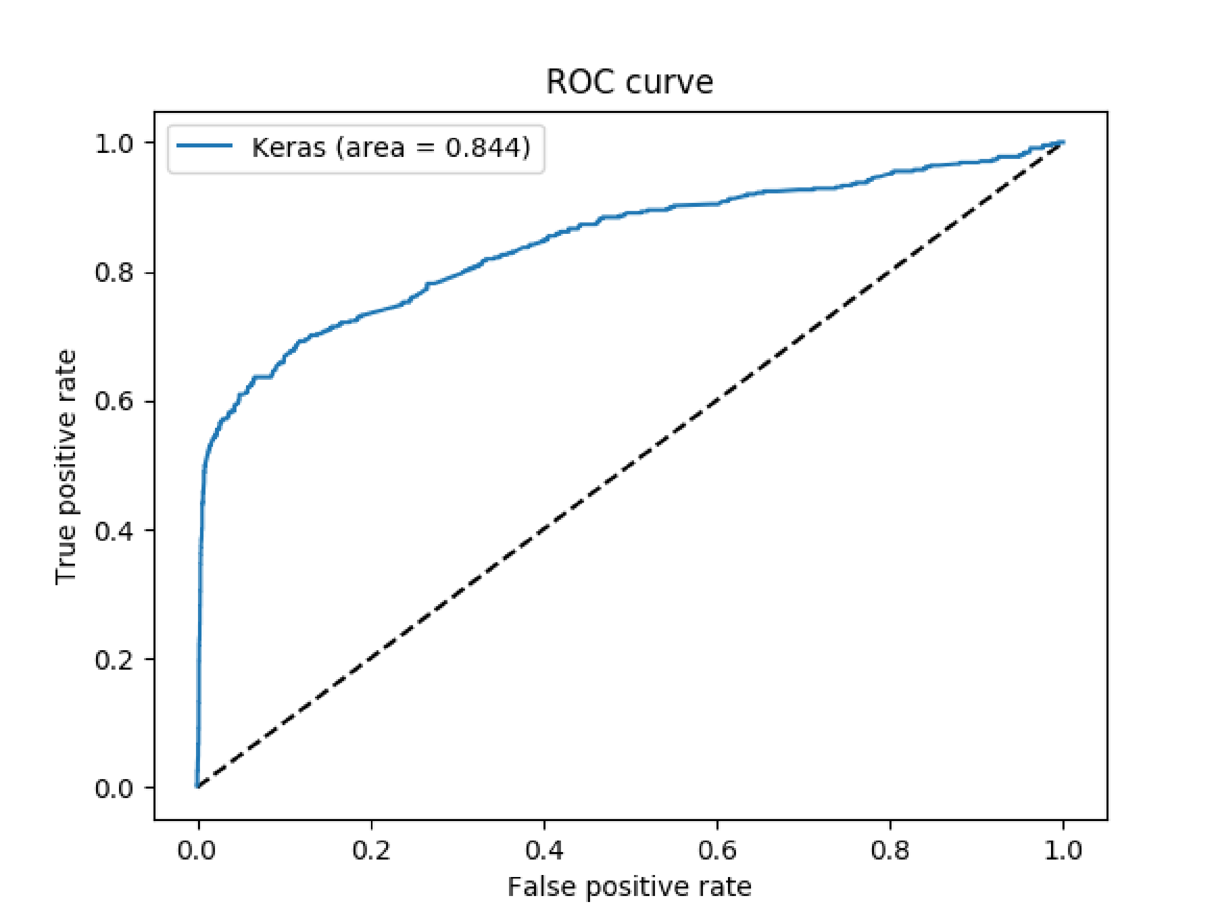
- Add Tier information to the input matrix for training.



Accuracy and auroc are improved (~5%)

To top

	T0_CH_CERN	T1_DE_KIT	T1_ES_PIC	T1_FR_CCIN2P3	T1_IT_CNAF	T1_RU_JINR	T1_UK_RAL	T1_US_FNAL	T2_CH_CERN	T2_CH_CERNBOX	T2_CH_CERN_HLT	T2_DE_DESY	T2_ES_IFCA	T2_FR_GRIF_IRFU	T2_FR_GRIF_LLR	T2_IT_Legnaro	T2_UK_London_Brunel	T2_UK_London_IC	T2_UK_SGrid_RALPP	T2_US_Florida	T2_US_MIT	T2_US_UCSD	T2_US_Wisconsin	T3_US_FNALLPC	
-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
84	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	
85	0	2	0	0	0	1	6	0	0	0	8	1	0	0	0	0	0	0	0	0	0	1	0	0	
92	0	0	1	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	2	0	0	0	0	0	
134	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	
139	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
8001	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
8004	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
50110	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
50660	0	0	3	2	4	1	1	1	6	0	63	1	0	1	0	1	0	1	0	0	1	0	1	0	
50664	0	0	2	7	0	0	2	18	0	0	32	0	0	7	0	0	4	0	2	0	0	0	0	0	
71304	0	0	0	0	0	0	0	0	0	0	102	0	0	0	0	0	0	0	0	0	0	0	0	0	
99305	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96



Using the full matrix, Including site names

- Has been studied in detail (Poster @ CHEP 2018)
- To overcome class imbalance
  - SMOTE (synthetic data by an knn approach)
  - A simple re-samplng of minority class events
- Bayesian hyper-parameter tuning
- ~80% AUROC and ~90% accuracy achieved

Ideas for improvement

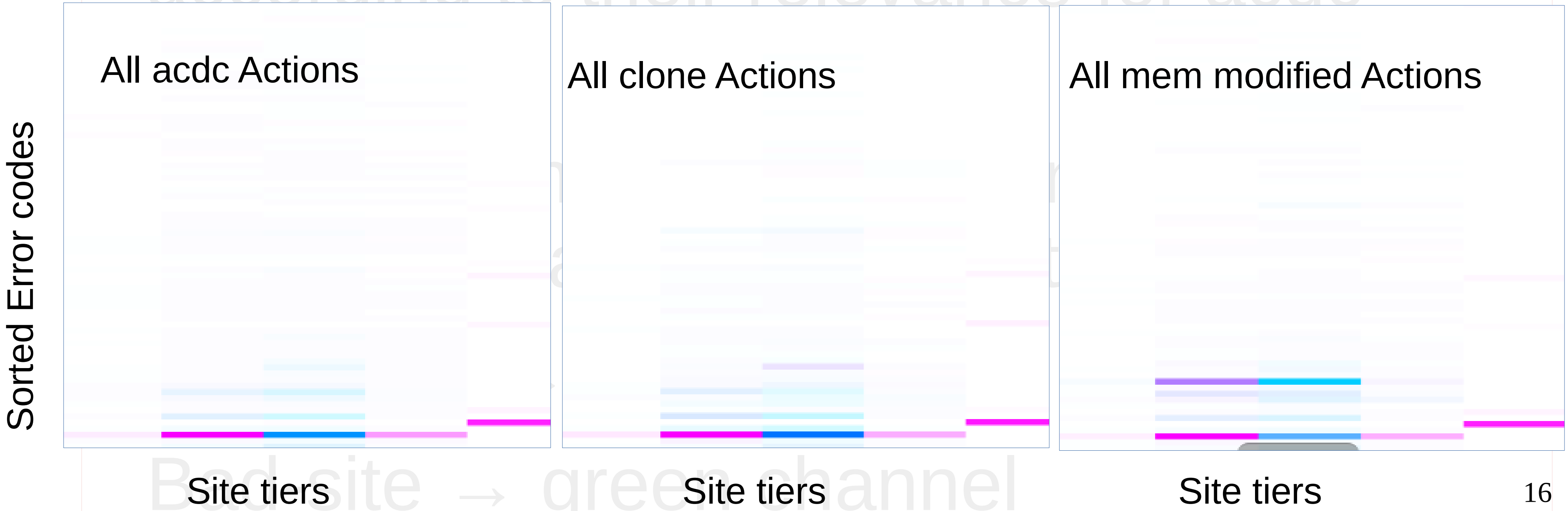
- Best results so far by grouping site data into tiers
- What about grouping error codes?
  - Error codes should be sorted
- How to weight data for grouping ?
- Convert (sorted) error/site matrix to image and use standard CNN methods

Visualization

- Error codes are sorted manually according to their relevance for acdc actions
- Average number of errors in each site-tier for different actions are plotted
- Good-site → red channel
- Bad-site → green channel

Visualization

Error codes are sorted manually according to their relevance for acdc



Bad site → green channel

Extended Neural Network

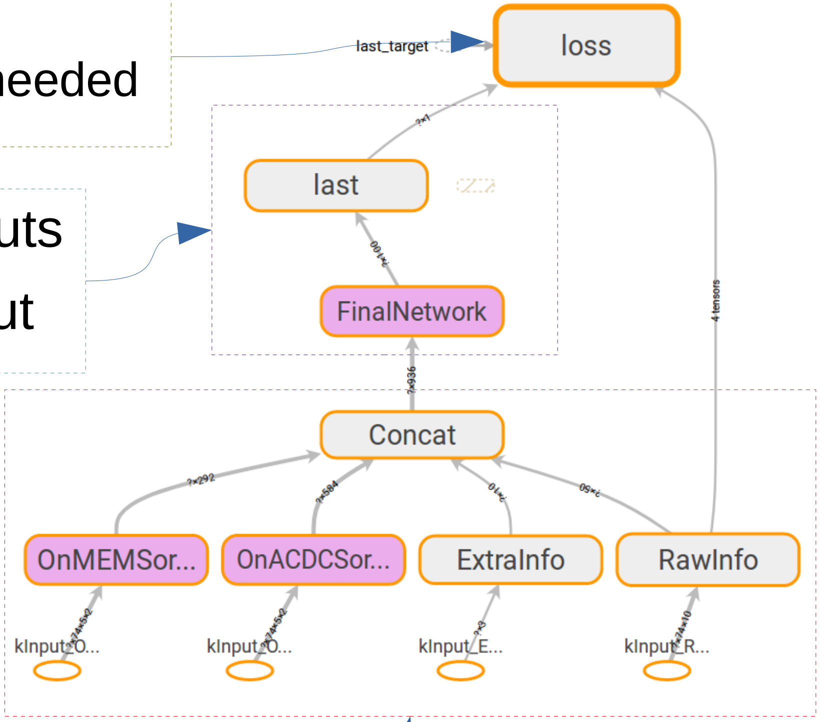
- Simple CNN on image presentation gives similar results to DNN
  - Depends heavily on the sorting of the error codes
- There are extra information, like total number of Jobs, missing from the site/error matrix
- An extended NN developed to include all the inputs

Extended Neural Network

- Binary cross entropy
- Target labels: if memory modification is needed

- Deep dense layer on top of all the outputs
- Last sigmoid layer to make binary output

- CNN: On image representations
  - different error code sortings
- Small one layer dense network on the “matrix of extra information”
- Deep NN on the “full matrix of error/site codes”
- Concatenate all outputs



Extended Neural Network
Optimization and results

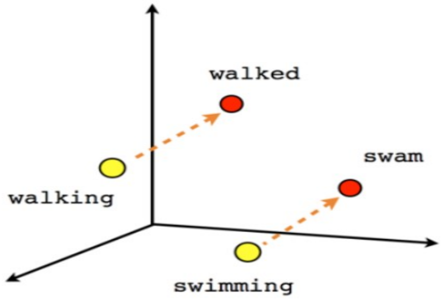
- Structure of the network is optimized using random/ bayesian search in KerasTuner package
- Results to predict if “memory configuration” is needed
  - AUC is improved ~1%
- No improvement in separating ‘clone’ and ‘acdc’ jobs

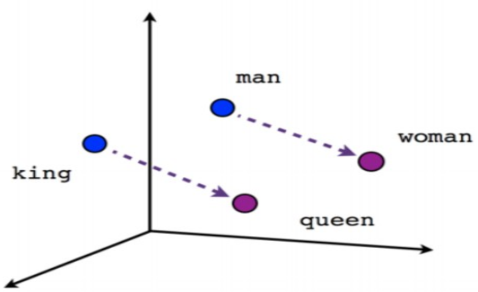
Error/Log files as input

- Add the information of the log files to the error/site matrix for the machine learning

- Use the NLP algorithm word2vec to map the log files words to high dimensional vectors

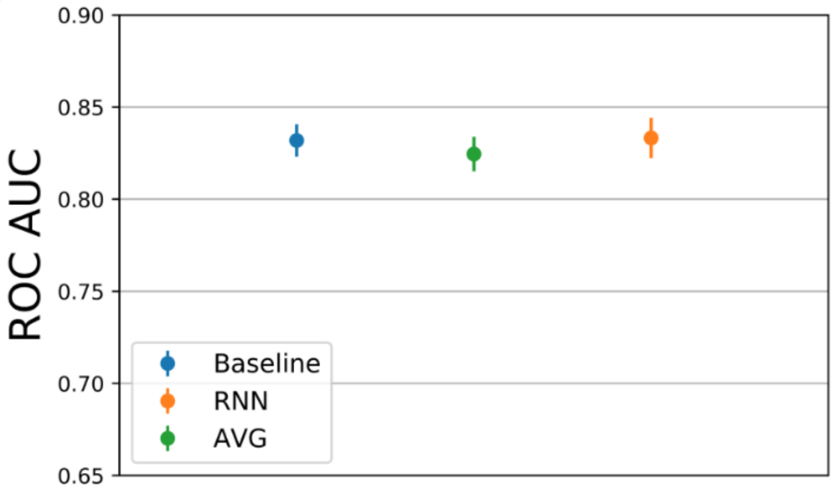
- Words that share common context in the corpus are located in close proximity to one another in space





Results

ROC AUC for acdc w.o. modification vs. other

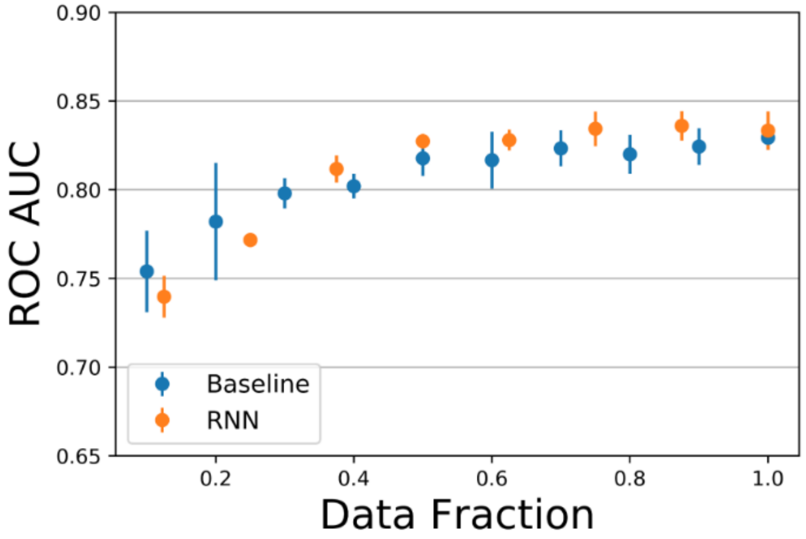


Baseline: FF with counts only

AVG: FF for averaged w2v + counts

RNN: RNN for embeddings + counts

ROC AUC as a function of the fraction of the total data set used for training



➡ **Successful training** of first complex NLP models

➡ **Similar results** as baseline - performance **improving with more data**

➡ **Work ongoing** → Full potential not yet exploited

Summary and outlook

- Attempts toward automation of “workflow recovery” were presented
- Site-Error matrices were summarized
- Reducing the input matrix to site-tier level gives better results
  - Extended Neural Net and CNN on image representation
  - Marginal improvement
- Log/Err files used for training
  - Average of word vectors / RNN to feed all the words
  - No improvement