

LSTM Tagger

Philipp Windischhofer

January 7, 2017

The Setup

Goal

Train a binary neural-network based classifier that can distinguish between b - and non- b -jets, using the raw jet data as input.

- use *tracks* as the primary source of information
- number of tracks is unknown a-priori \rightarrow cannot use an architecture that expects a fixed number of inputs
- currently looking into recurrent neural networks / LSTM networks

Recurrent Neural Networks

The Workflow

Training

- match tracks to their associated jets (contained in different ROOT trees)
- for each track in the jet, feed all 8 available track parameters into the classifier network during training
 - ▶ use p_T ordering, i.e. hardest track first
- supervised training: provide a binary (0/1) output value for each jet (from MC)

Now running on Piz Daint:

- roughly $2\text{-}3 \times$ improvement in execution speed compared to PSI/Tier-3
- limited by Jet-Track-matching, which is handled by the CPU
- possible workaround: train multiple classifiers during the same run “in parallel”

The Workflow

Evaluation

- compare performance to cMVA tagger as “gold-standard”
- obtain ROC curves for both classifiers, correlation plots of the outputs
- currently: validation data is disjoint from training data, but from the same MC-run (i.e. contains a similar event signature)

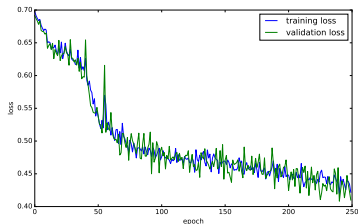
Results so far

- trained a number of LSTM networks, scanned the hyperparameters:
 - ▶ number of nodes in each layer
 - ▶ number of layers
 - ▶ number of training epochs

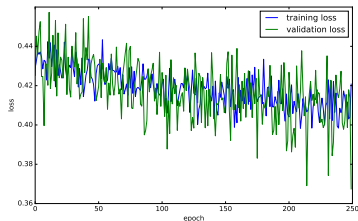
Details of the training:

- read training data in chunks of 10k jets
- use 8k jets for training, 2k jets to monitor performance during each epoch

64 nodes / layer, 1 layer



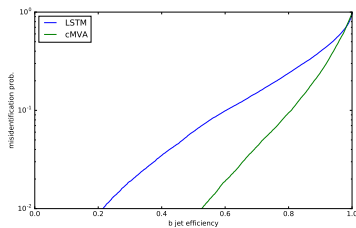
(a)



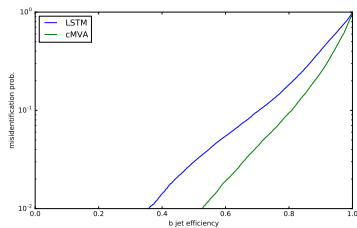
(b)

Figure: test

64 nodes / layer, 1 layer



(a)

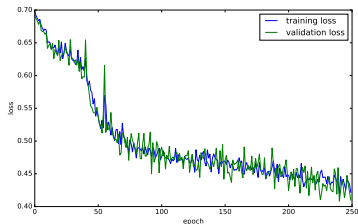


(b)

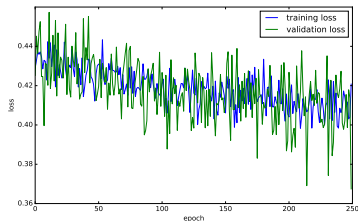
Figure: test

- $\text{AUC}(\text{cMVA}) = 0.9233$
- $\text{AUC}(\text{RNN}) = 0.8794$

64 nodes / layer, 3 layers



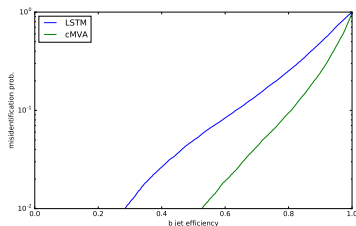
(a)



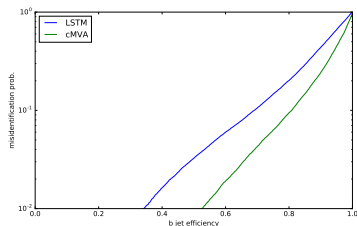
(b)

Figure: test

64 nodes / layer, 3 layers



(a)



(b)

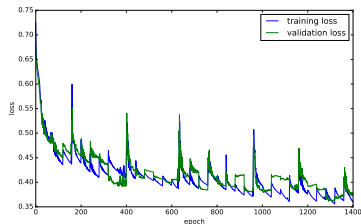
Figure: test

- $\text{AUC}(\text{cMVA}) = 0.9233$
- $\text{AUC}(\text{RNN}) = 0.8704$

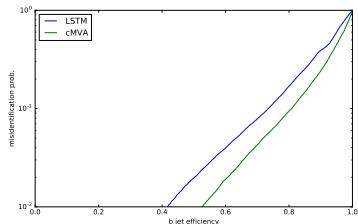
64 nodes / layer, 3 layers

A different training strategy

Use 40 training epochs per 10k chunk:



(a)



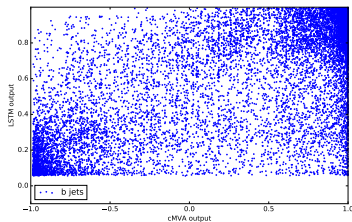
(b)

Figure: test

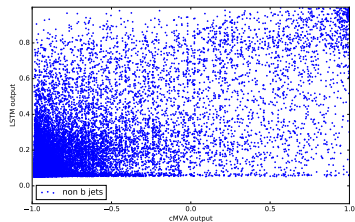
- $AUC(cMVA) = 0.9233$
- $AUC(RNN) = 0.888$

64 nodes / layer, 3 layers

Output compared to cMVA



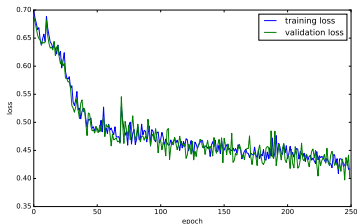
(a)



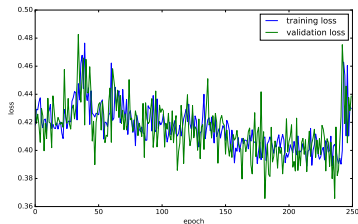
(b)

Figure: test

128 nodes / layer, 1 layer



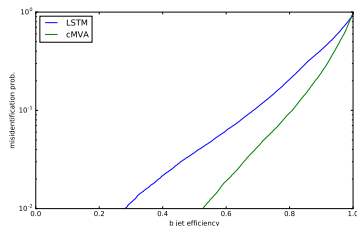
(a)



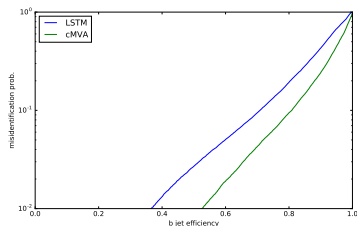
(b)

Figure: test

128 nodes / layer, 1 layer



(a)



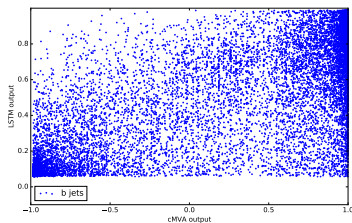
(b)

Figure: test

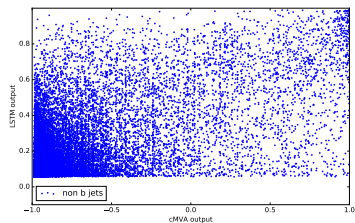
- $AUC(cMVA) = 0.9233$
- $AUC(RNN) = 0.8737$

128 nodes / layer, 1 layer

Output compared to cMVA



(a)



(b)

Figure: test

Conclusions

- basic infrastructure seems to be in place and working
- classifier performance is very similar across the different networks that were evaluated
 - ▶ training just not complete? Use still more epochs even if loss doesn't seem to improve much anymore?
 - ▶ or is performance limited by the data representation / preprocessing rather than the network architecture?

Future Experiments

- more sophisticated preprocessing?
- different representation (other than the raw track data)?
- try removing individual track parameters to see how performance degrades