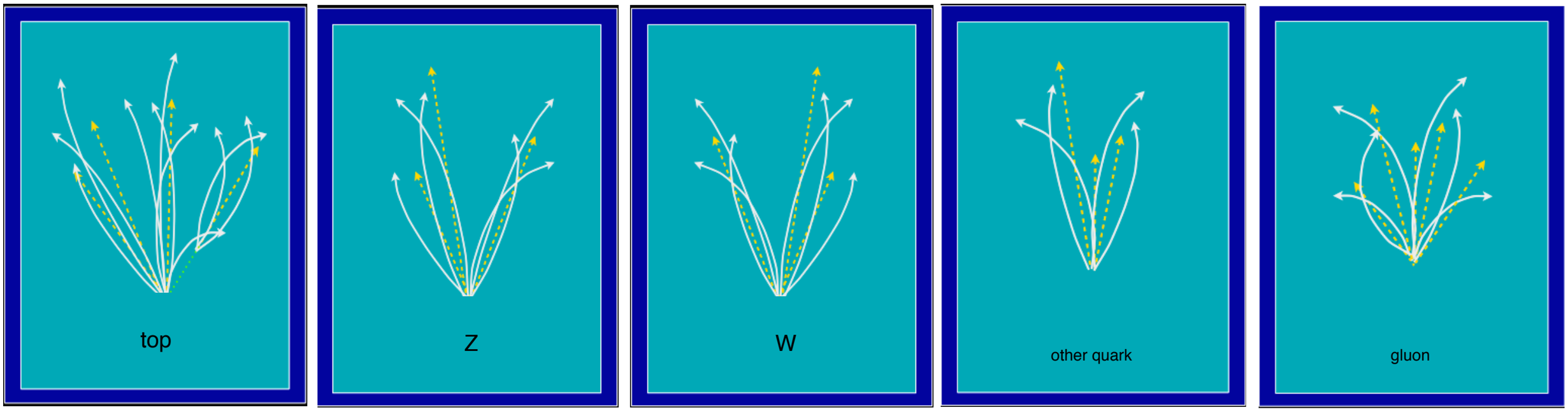
# Description of available models

# Physics case: jet tagging

Study a (**multi-)classification task to be implemented on FPGA:** discrimination between highly energetic (boosted) ***q, g, W, Z, t*** initiated jets

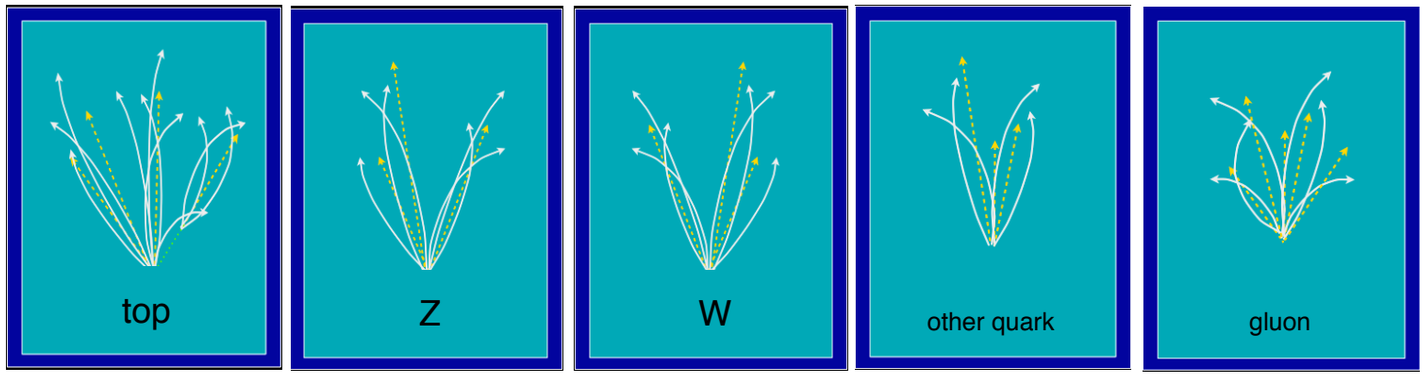


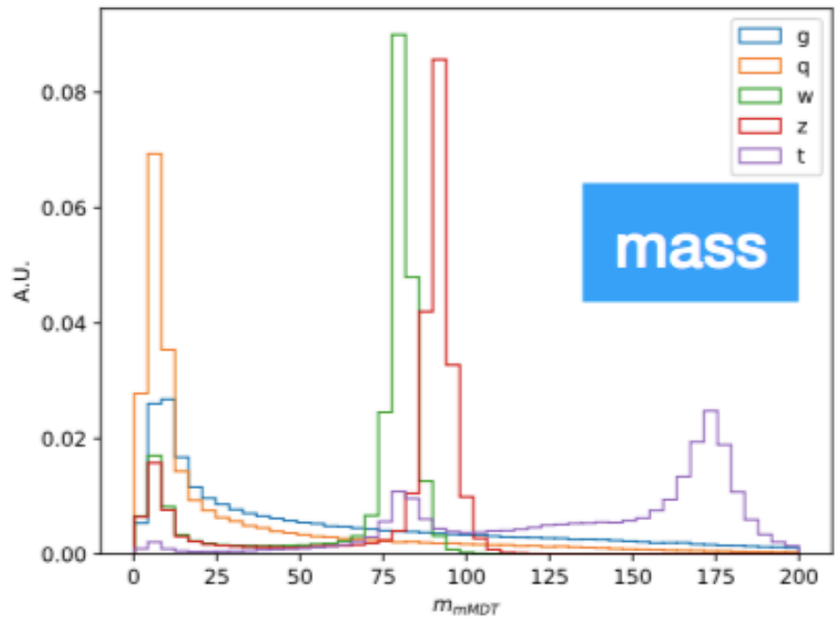

| **t→bW→bqq** | **Z→qq** | **W→qq** | **q/g background** |
|---|---|---|---|
| 3-prong jet | 2-prong jet | 2-prong jet | no substructure and/or mass ~ 0 |

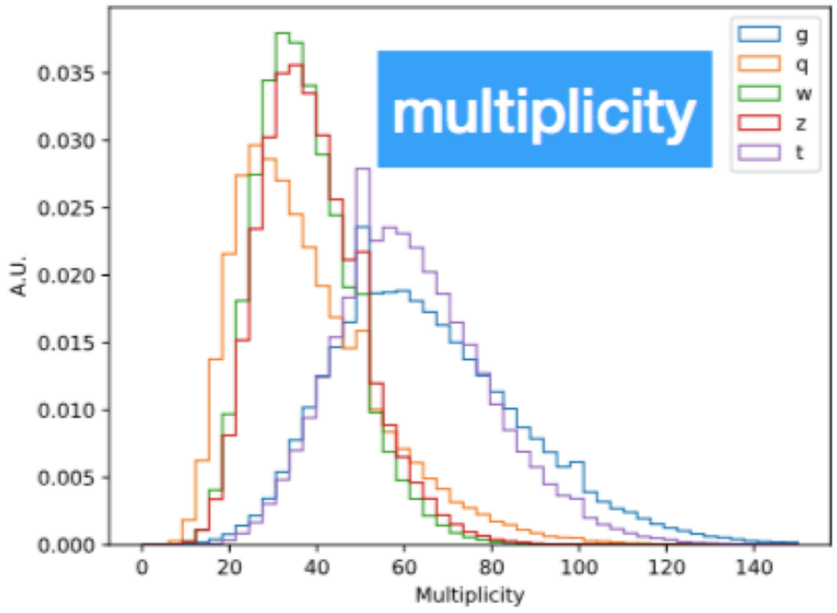
Reconstructed as one massive jet with substructure

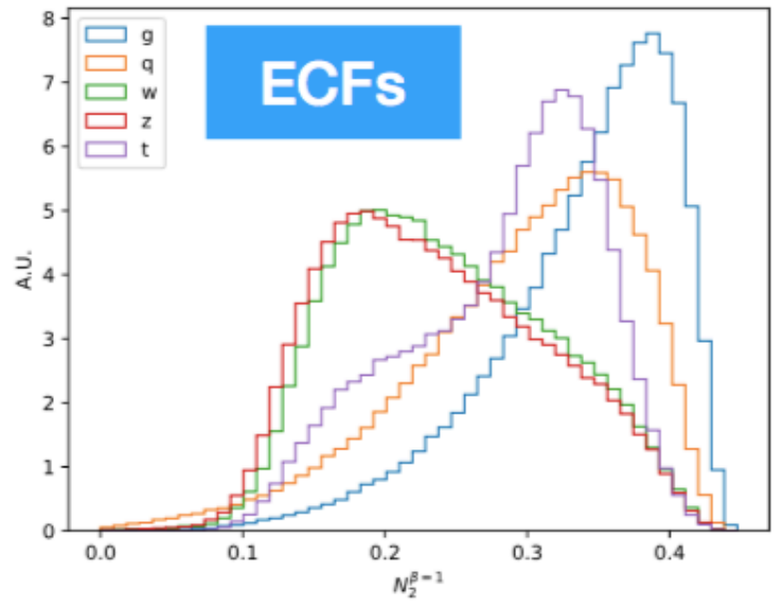
# Physics case: jet tagging

- Dataset: sample of events with a mixture of two boosted WW/ZZ/tt/qq/gg anti-$k_T$ jets (200,000 events for each final state)



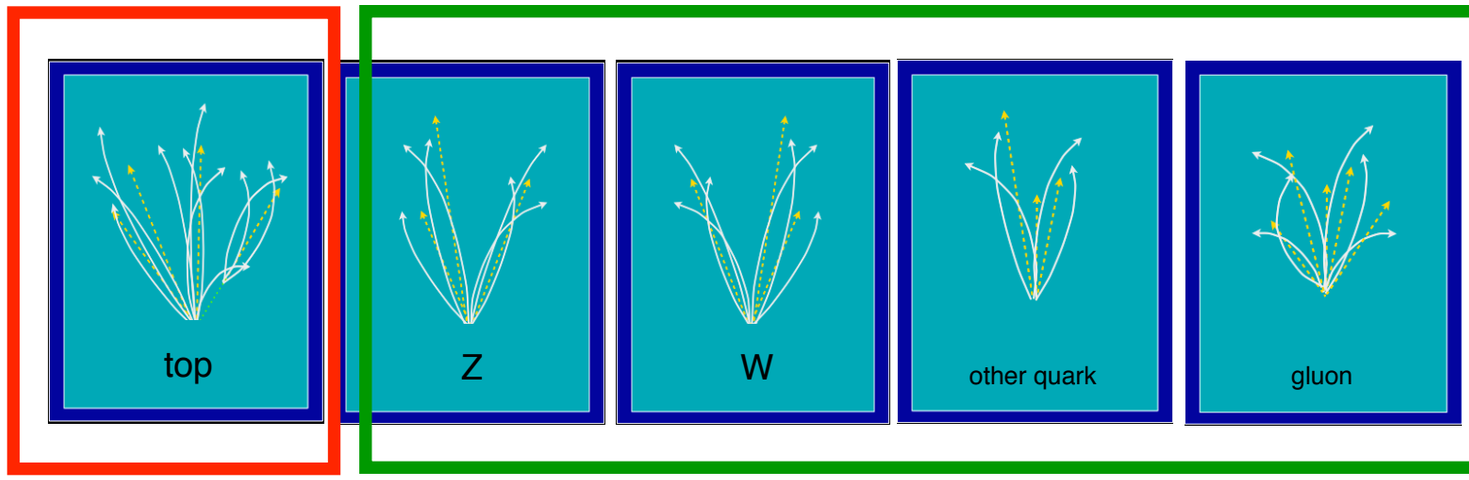

**Input variables: several obervables known to have high discrimination power from offline data analyses and published studies [*]**

[*] D. Guest at al. PhysRevD.94.112002, G. Kasieczka et al. JHEP05(2017)006, J. M. Butterworth et al. PhysRevLett.100.242001, etc..

# 1-layer Dense NN

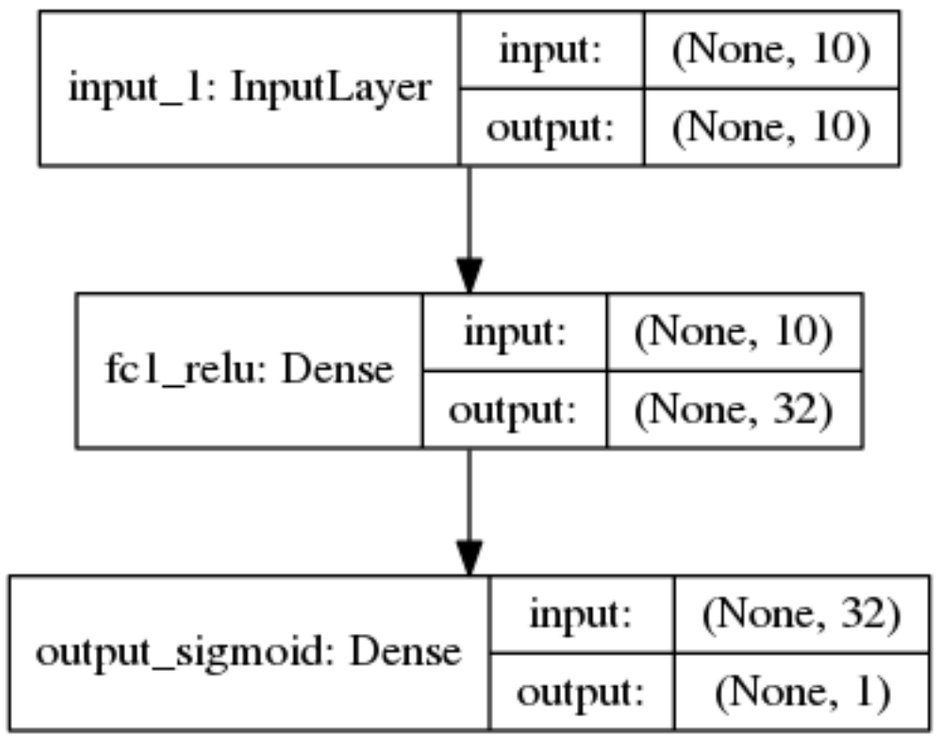
*Simple classification task: discriminate top-quark jets from the others*

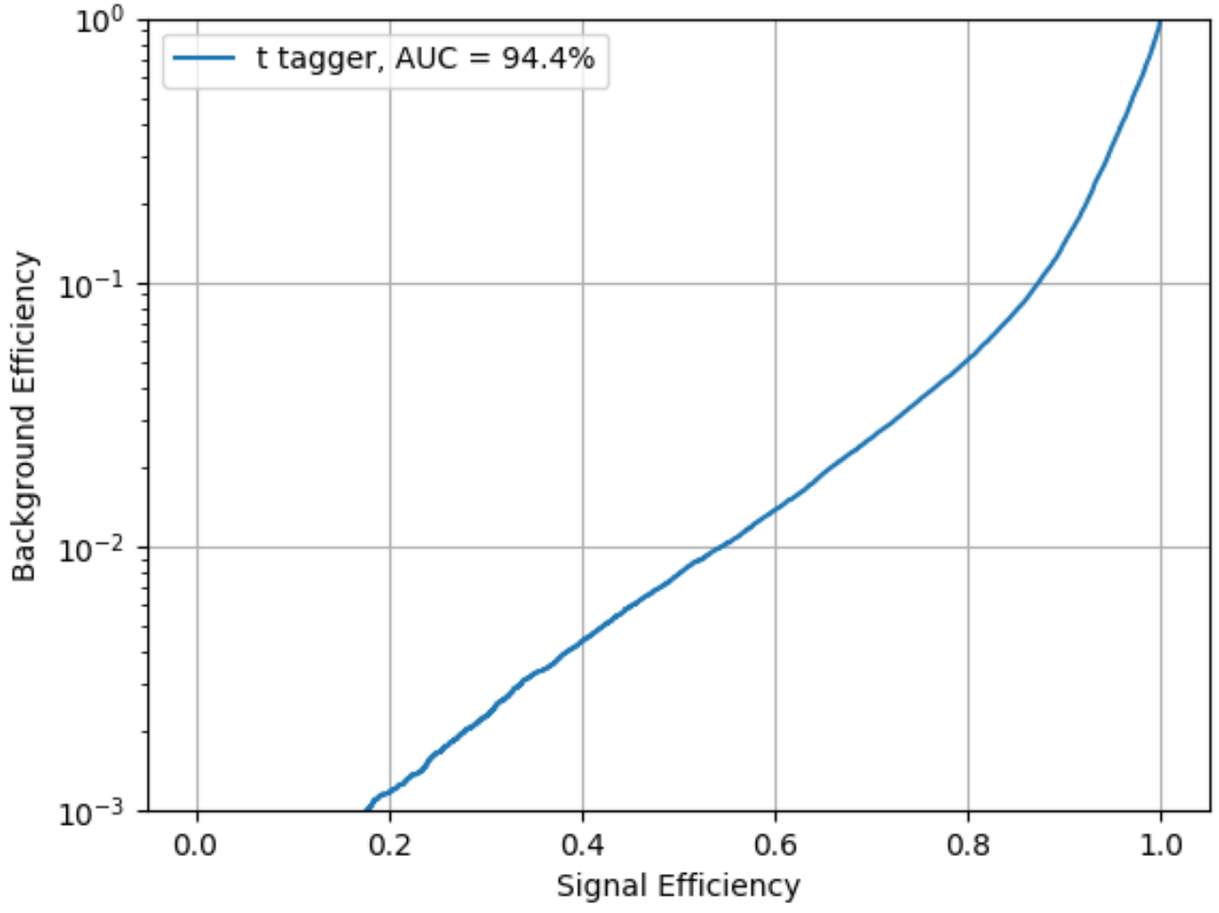
signal:
top-quark jets




background:
all other jets

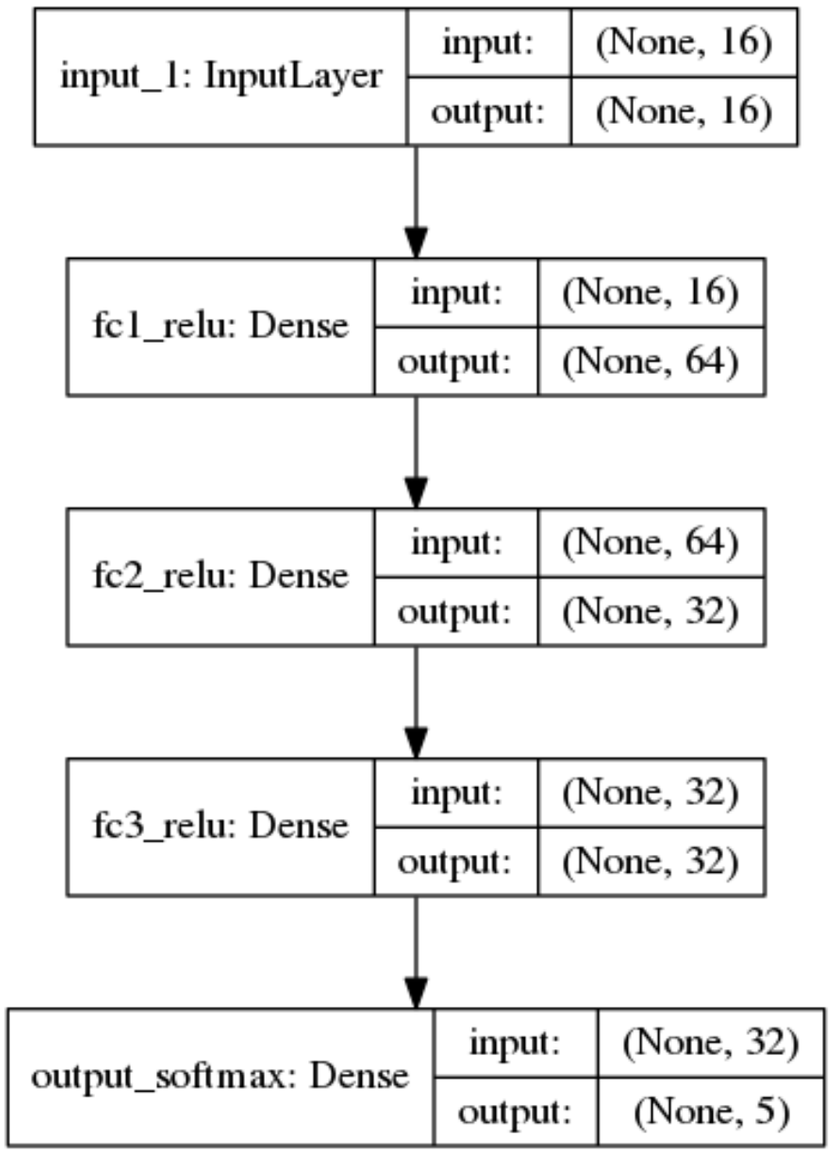
hls4ml

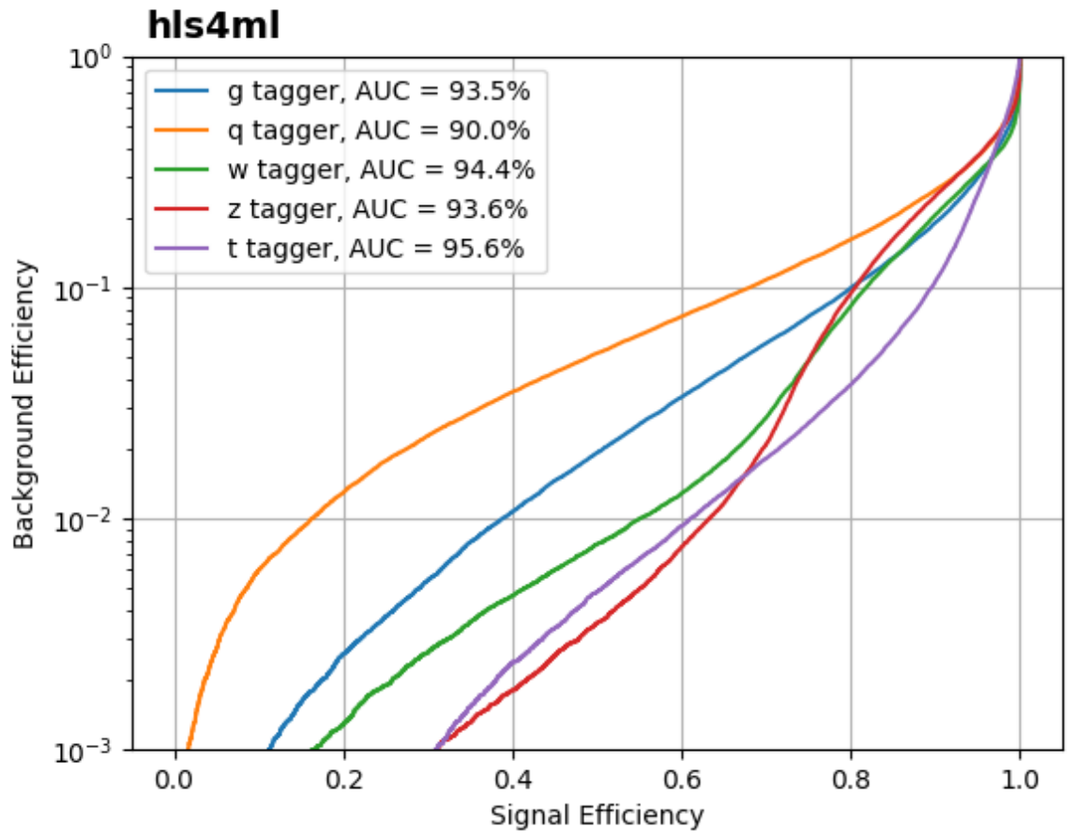


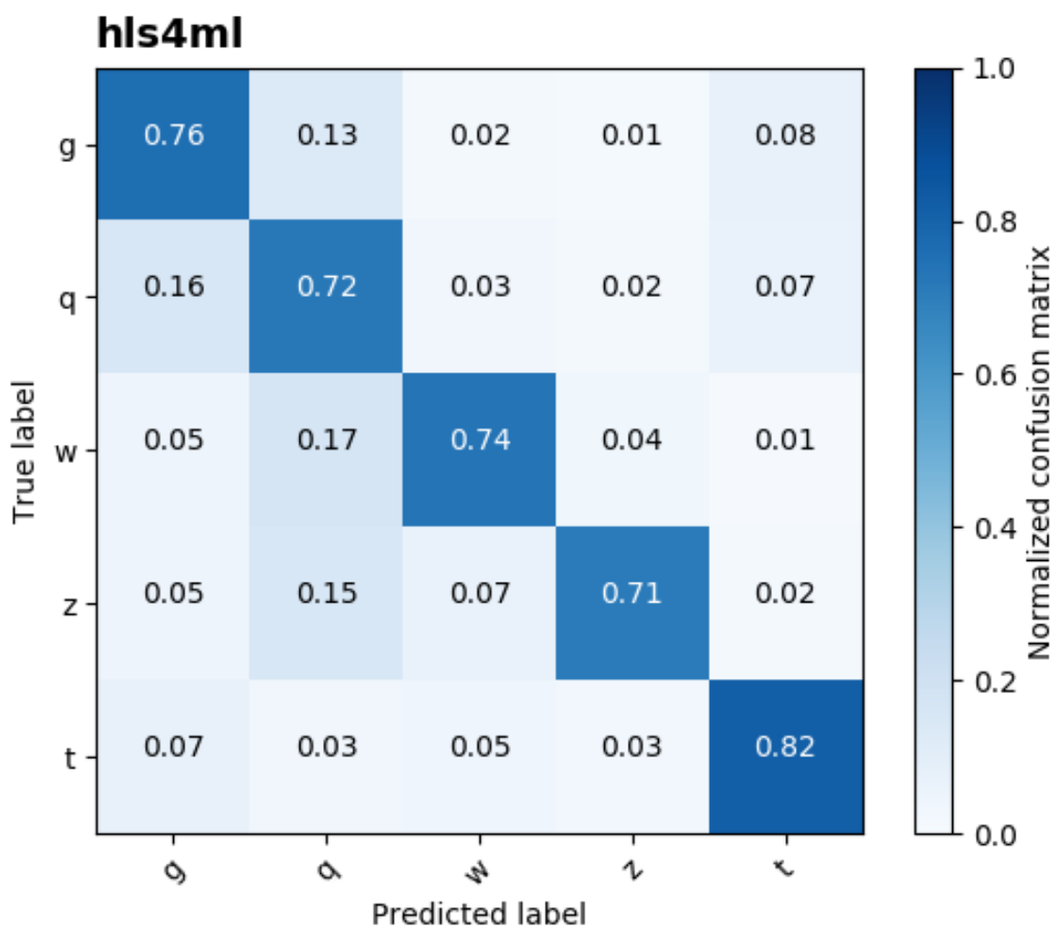



# 3-layers Dense NN (1)

*Multi classification task: discrimination between the 5 classes of jets (t,W,Z,q,g)*



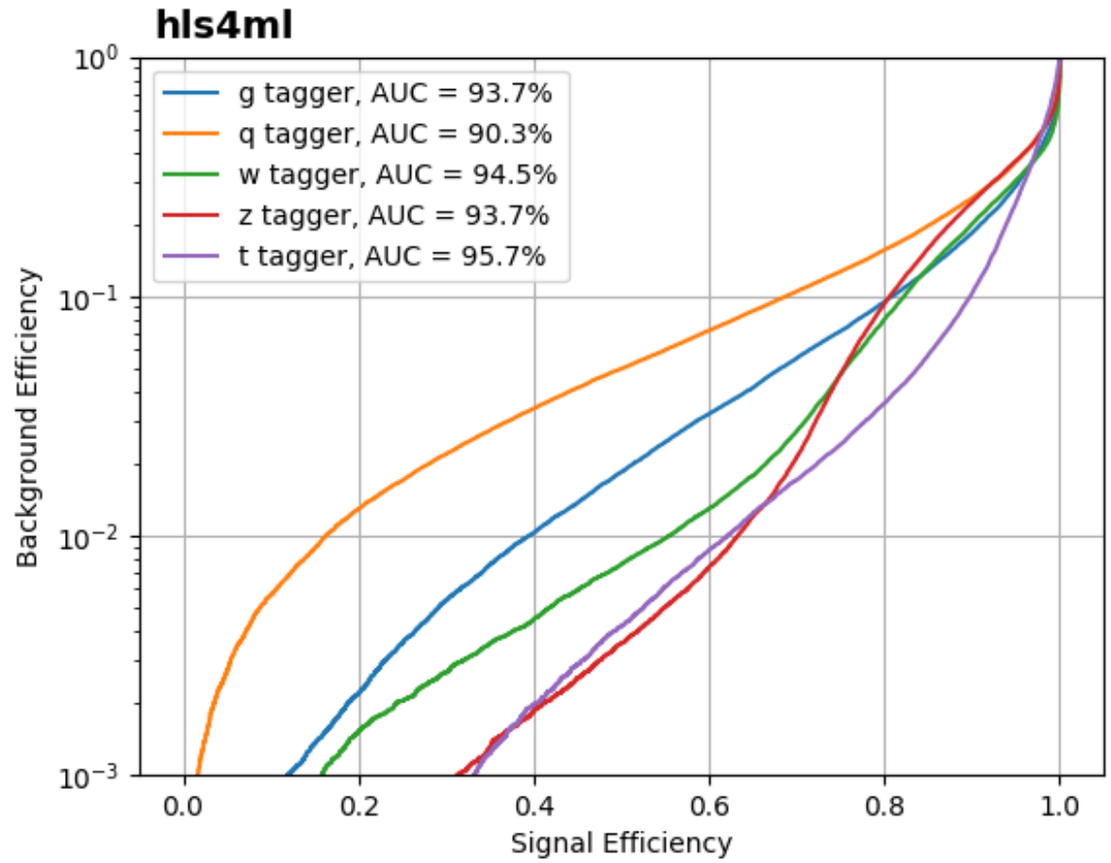
# 3-layers Dense NN (2)

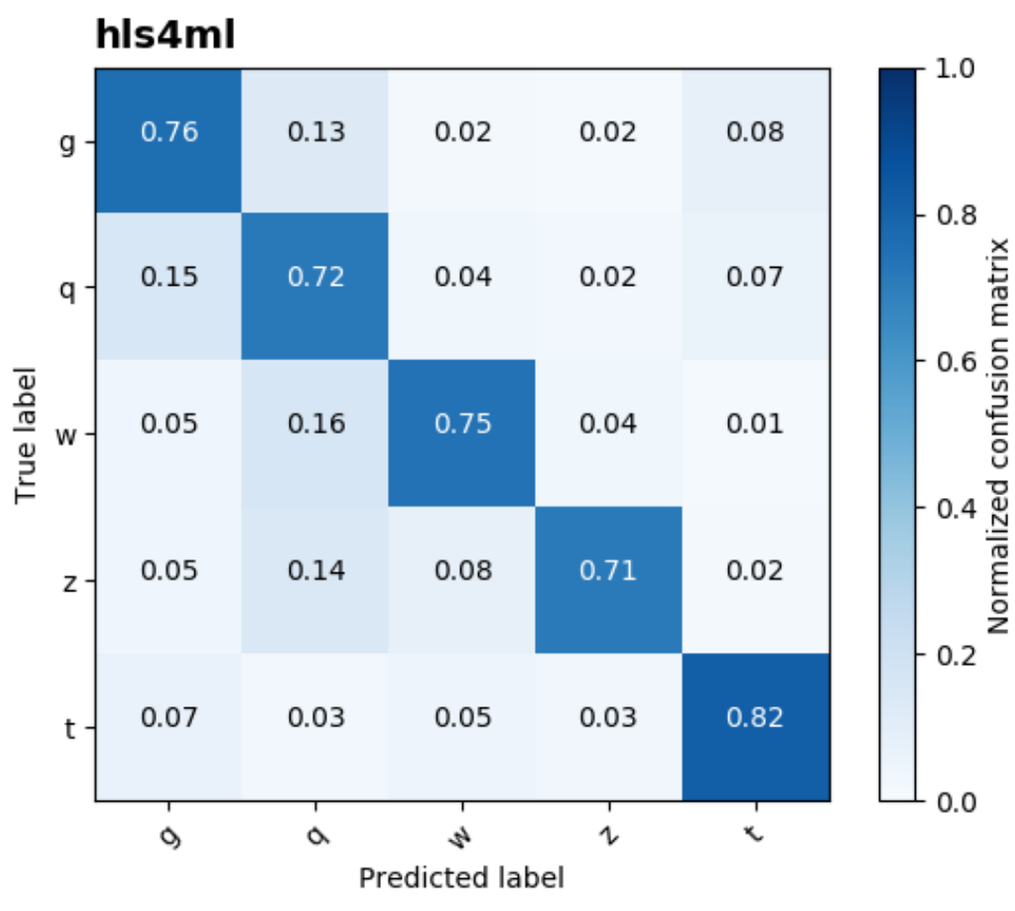
*As previous slide but adding batch normalization layers [*]*

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | (None, 16) | 0 |
| fc1_relu (Dense) | (None, 64) | 1088 |
| **bn1 (BatchNormalization)** | **(None, 64)** | **256** |
| relu1 (Activation) | (None, 64) | 0 |
| fc2_relu (Dense) | (None, 32) | 2080 |
| **bn2 (BatchNormalization)** | **(None, 32)** | **128** |
| relu2 (Activation) | (None, 32) | 0 |
| fc3_relu (Dense) | (None, 32) | 1056 |
| **bn3 (BatchNormalization)** | **(None, 32)** | **128** |
| relu3 (Activation) | (None, 32) | 0 |
| output_softmax (Dense) | (None, 5) | 165 |
| **bn4 (BatchNormalization)** | **(None, 5)** | **20** |
| softmax (Activation) | (None, 5) | 0 |

Total params: 4,921
Trainable params: 4,655
Non-trainable params: 266

*[*] nb, we choose here to do batch normalization before the activation but there are different ways to do this.This results in one more Dense layer (output_softmax).*



# 3-layers Binary Dense

*3-layers with batch normalization as in previous slides but:*

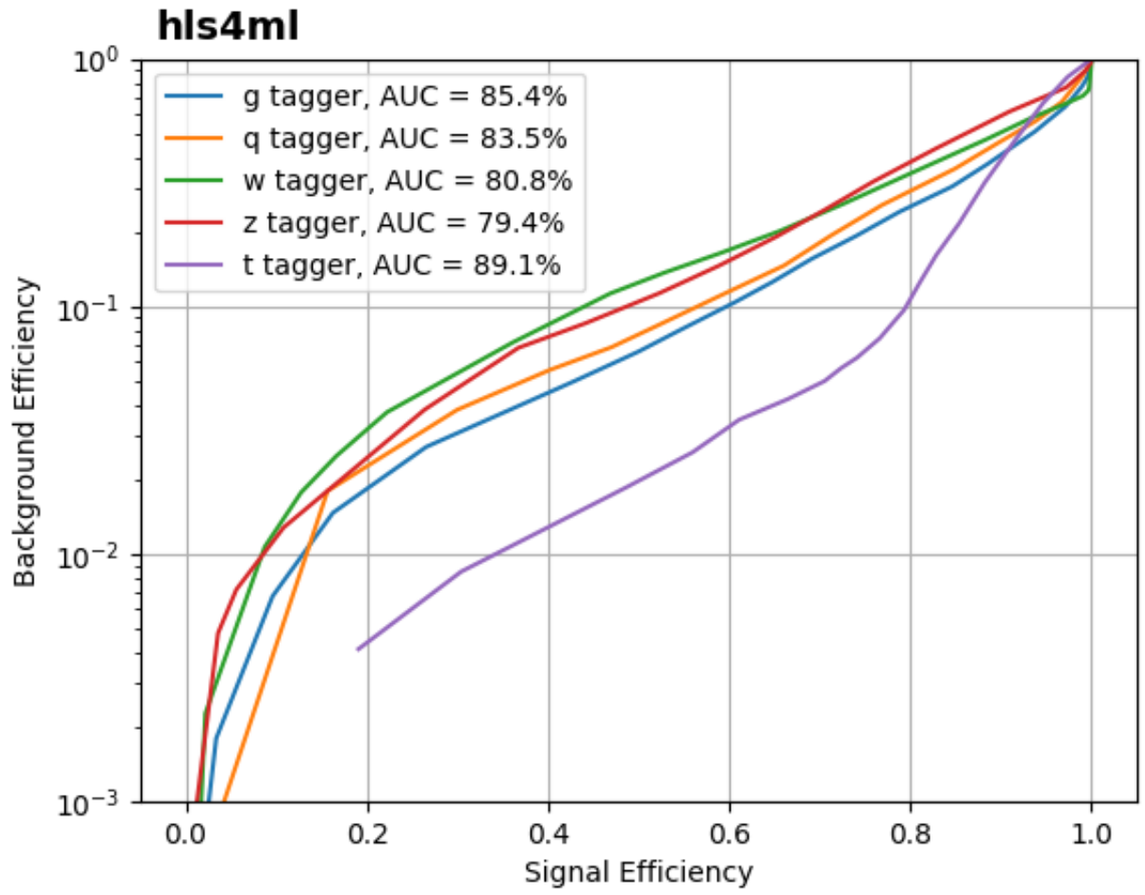
***weights = +/- 1***

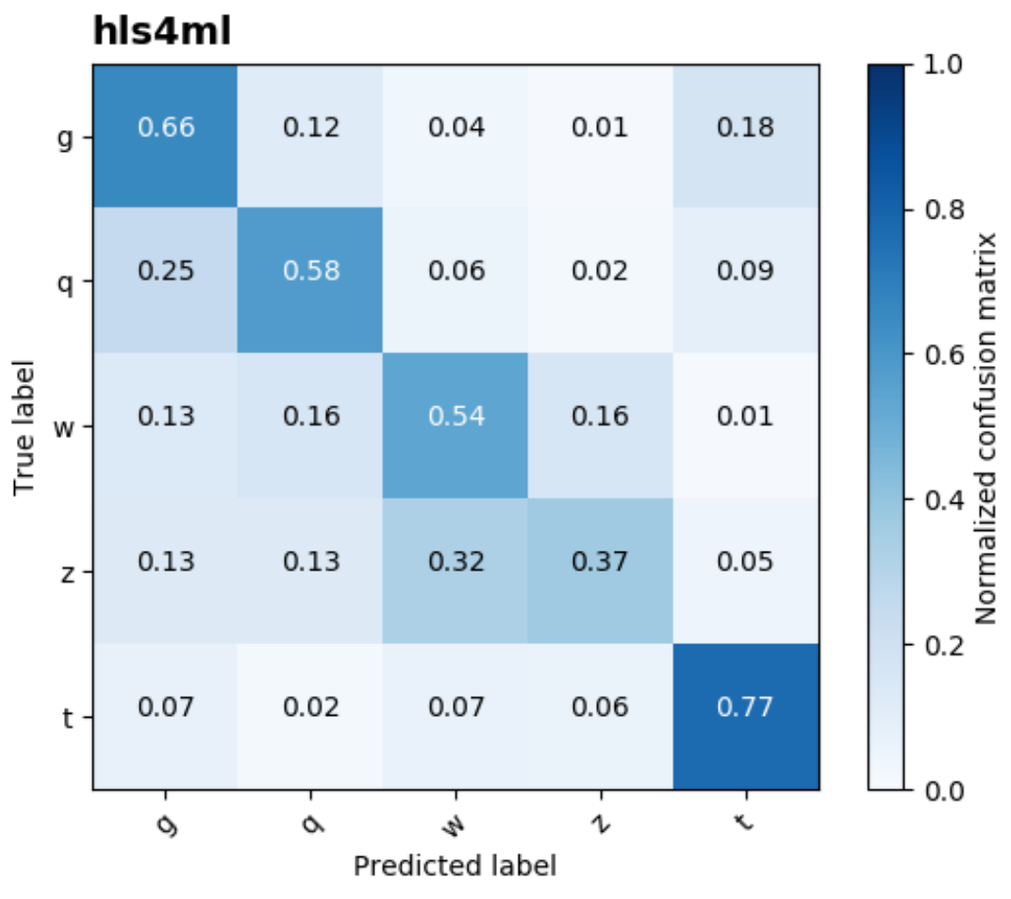
***binary activation function***

***no output softmax***

| Layer (type) | Output Shape | Param # |
|---|---|---|
| **fc1 (BinaryDense)** | **(None, 64)** | **1024** |
| bn1 (BatchNormalization) | (None, 64) | 256 |
| act1 (Activation) | (None, 64) | 0 |
| **fc2 (BinaryDense)** | **(None, 32)** | **2048** |
| bn2 (BatchNormalization) | (None, 32) | 128 |
| act2 (Activation) | (None, 32) | 0 |
| **fc3 (BinaryDense)** | **(None, 32)** | **1024** |
| bn3 (BatchNormalization) | (None, 32) | 128 |
| act3 (Activation) | (None, 32) | 0 |
| **output (BinaryDense)** | **(None, 5)** | **160** |
| bn (BatchNormalization) | (None, 5) | 20 |

Total params: 4,788
Trainable params: 4,522
Non-trainable params: 266



# 3-layers Ternary Dense

*3-layers with batch normalization as in previous slides but:*

***weights = +/- 1 or 0***

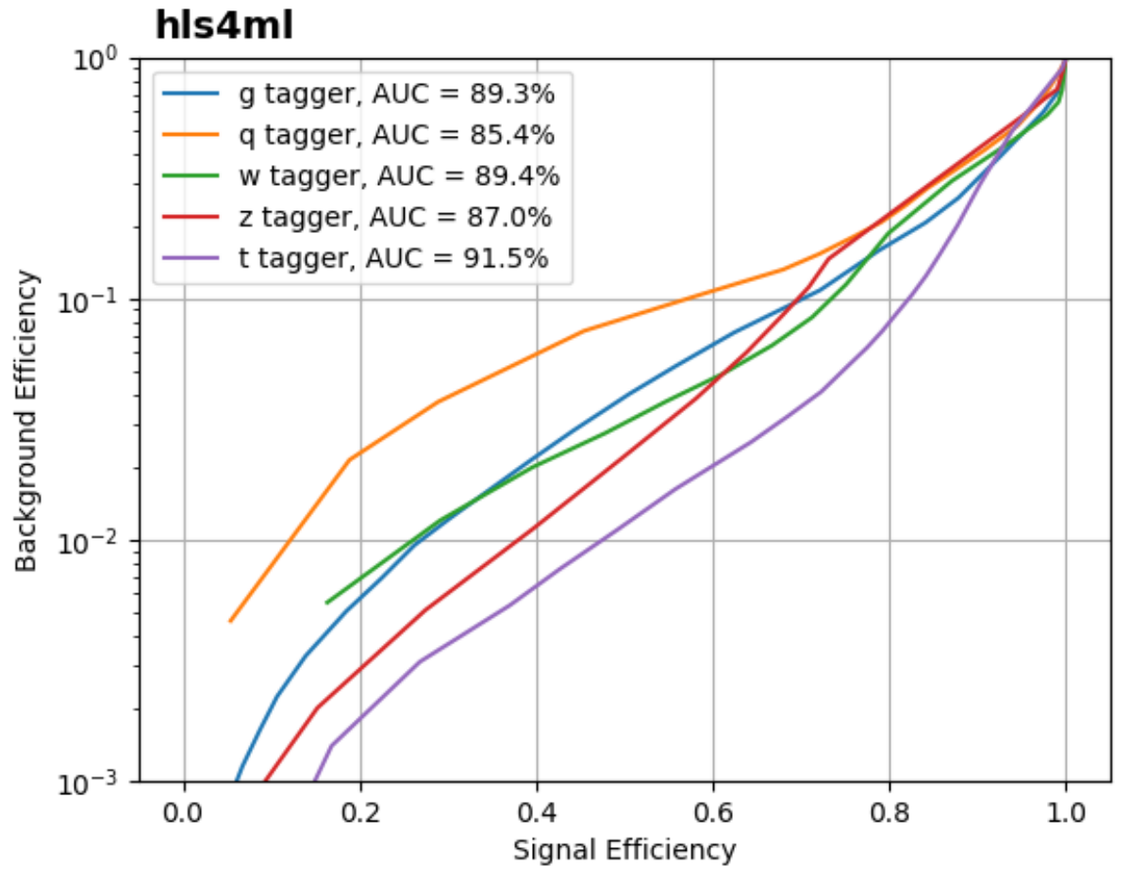
***ternary activation function***

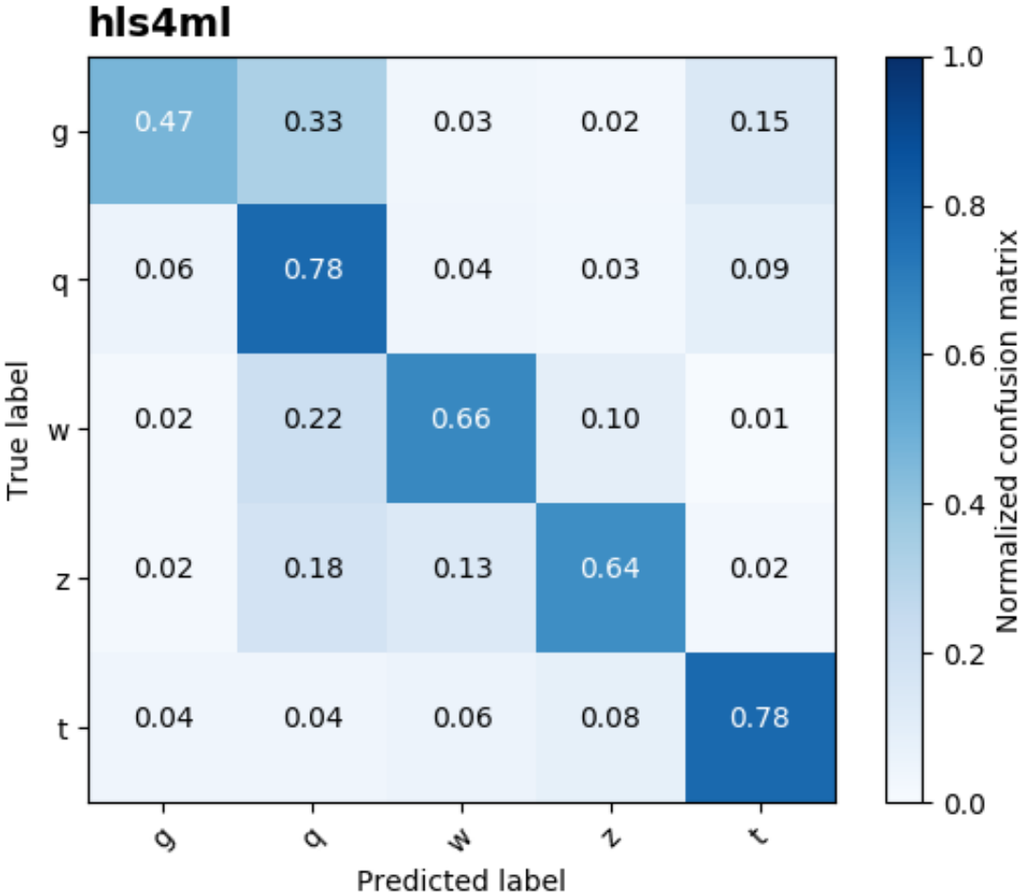
***no output softmax***

| Layer (type) | Output Shape | Param # |
|---|---|---|
| **fc1 (TernaryDense)** | **(None, 64)** | **1024** |
| bn1 (BatchNormalization) | (None, 64) | 256 |
| act1 (Activation) | (None, 64) | 0 |
| **fc2 (TernaryDense)** | **(None, 32)** | **2048** |
| bn2 (BatchNormalization) | (None, 32) | 128 |
| act2 (Activation) | (None, 32) | 0 |
| **fc3 (TernaryDense)** | **(None, 32)** | **1024** |
| bn3 (BatchNormalization) | (None, 32) | 128 |
| act3 (Activation) | (None, 32) | 0 |
| **output (TernaryDense)** | **(None, 5)** | **160** |
| bn (BatchNormalization) | (None, 5) | 20 |

Total params: 4,788
Trainable params: 4,522
Non-trainable params: 266

[*] more info: https://arxiv.org/abs/1605.04711



# List of available models

- Find the weigths (.h5 file) and model architecture (.json file) under the `example-keras-model-files`

  - **1-layer Dense NN:**
    `KERAS_1layer.json, KERAS_1layer_weights.h5`

  - **3-layers Dense NN:**
    `KERAS_3layer.json, KERAS_3layer_weights.h5`

  - **3-layers Dense NN + BatchNormalization:**
    `KERAS_3layer_batch_norm.json, KERAS_3layer_batch_norm_weights.h5`

  - **3-layers Binary Dense NN:**
    `KERAS_3layer_binary.json, KERAS_3layer_binary_weights.h5`

  - **3-layers Ternary Dense NN:**
    `KERAS_3layer_ternary.json, KERAS_3layer_ternary_weights.h5`

**Pick your favourite one to implement it on FPGA!**