# Search for beyond Standard Model Higgs resonances with a parametrised neural network

Laura Pereira Sánchez

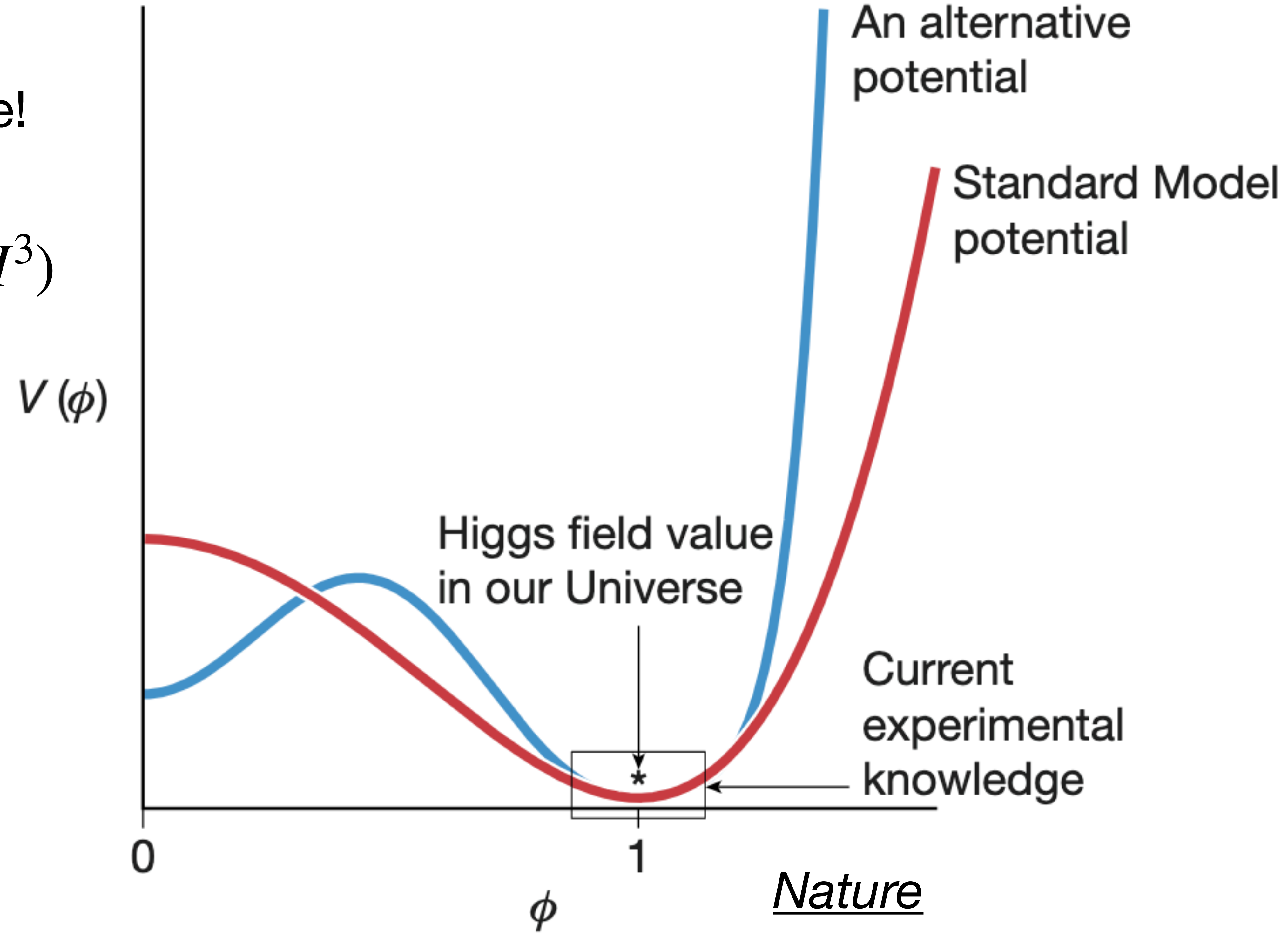
Nordic Conference on Particle Physics

Spåtind 2023

Stockholm University

# Introduction

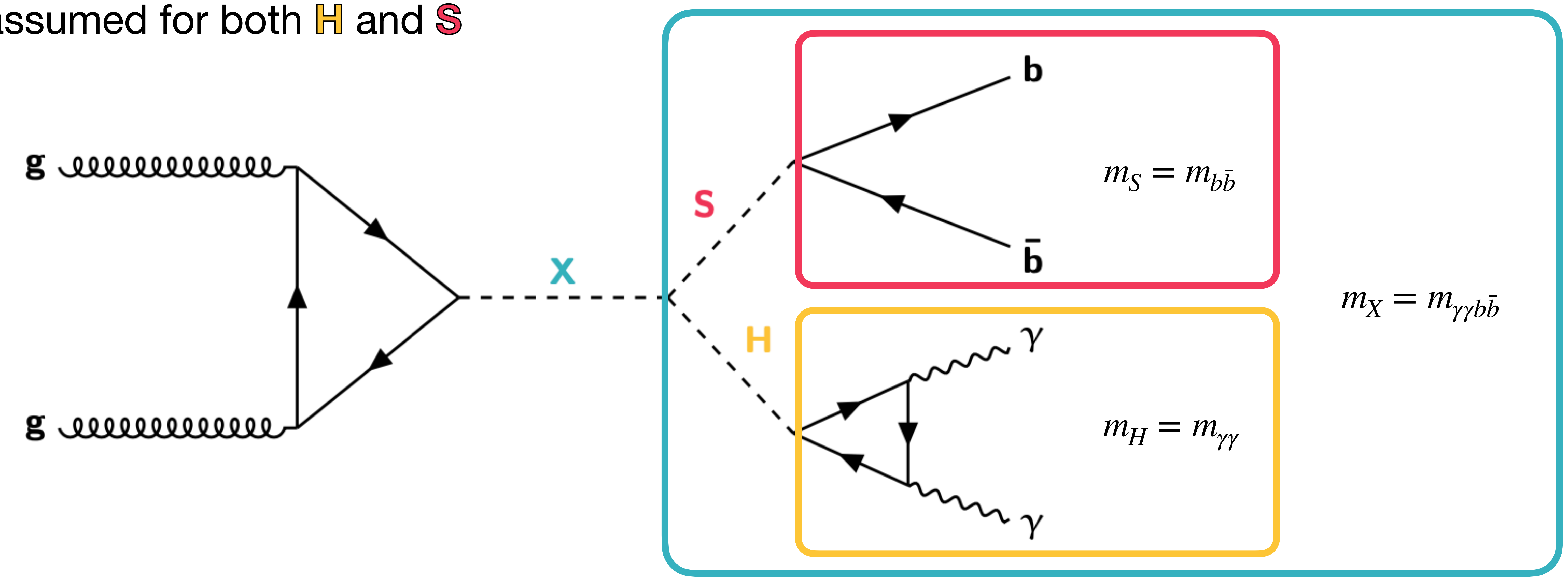
- The SM predicts the existence of the Higgs field ($\phi$) and its potential $V(\phi)$
    - In the early Universe $V(\phi)$ had a minimum at $\phi = 0$ and all particles were massless
    - A few pico seconds after the Big Bang a **phase transition** to a new ground state $\phi \neq 0$ occurred and many particles acquired mass
    - The SM **Higgs boson** (H) is a result of this spontaneous symmetry breaking (SSB)
- The shape of the potential has a big impact on the physics of the Universe!
    - In the SM the phase transition is smooth and $V(H) = \frac{1}{2}m_H^2 H^2 + O(H^3)$
    - A first order phase transition could explain the matter-anti-matter asymmetry in the Universe
- Furthermore, the Higgs field could naturally connect to dark matter and cosmology's inflaton could be some sort of extra Higgs boson!



*Nature*

# Targeted signals

- Several **BSM** models predict the existence of additional scalars with different masses and cross sections
- We search for a heavy scalar X decaying into a lighter scalar S and the SM Higgs H where S $\rightarrow b\bar{b}$ and H $\rightarrow \gamma\gamma$
- In order to keep the search as model-agnostic as possible:
  - A wide range of $m_X, m_S$ signals are targeted: $30 \leq m_S \leq 500$ GeV and $170$ GeV $\leq m_S \leq 1$ TeV
  - The production cross section is always assumed to be of 1 fb
  - SM decays are assumed for both H and S



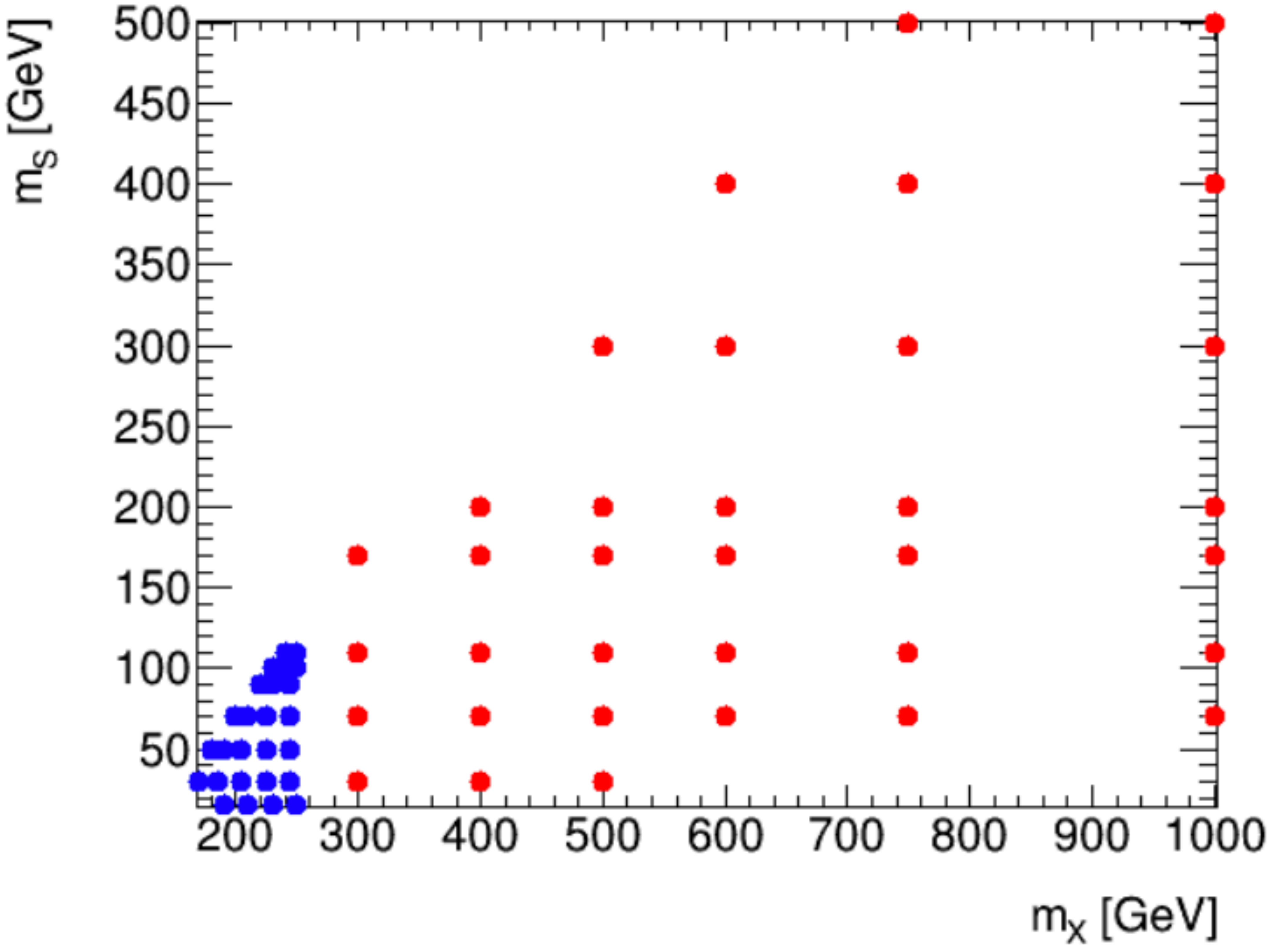
# How to target all this points?

We want to use a Neural Network (NN) to discriminate signal from background, but how to train our network?

A. Target groups of similar signals
   - Train a NN with multiple signals

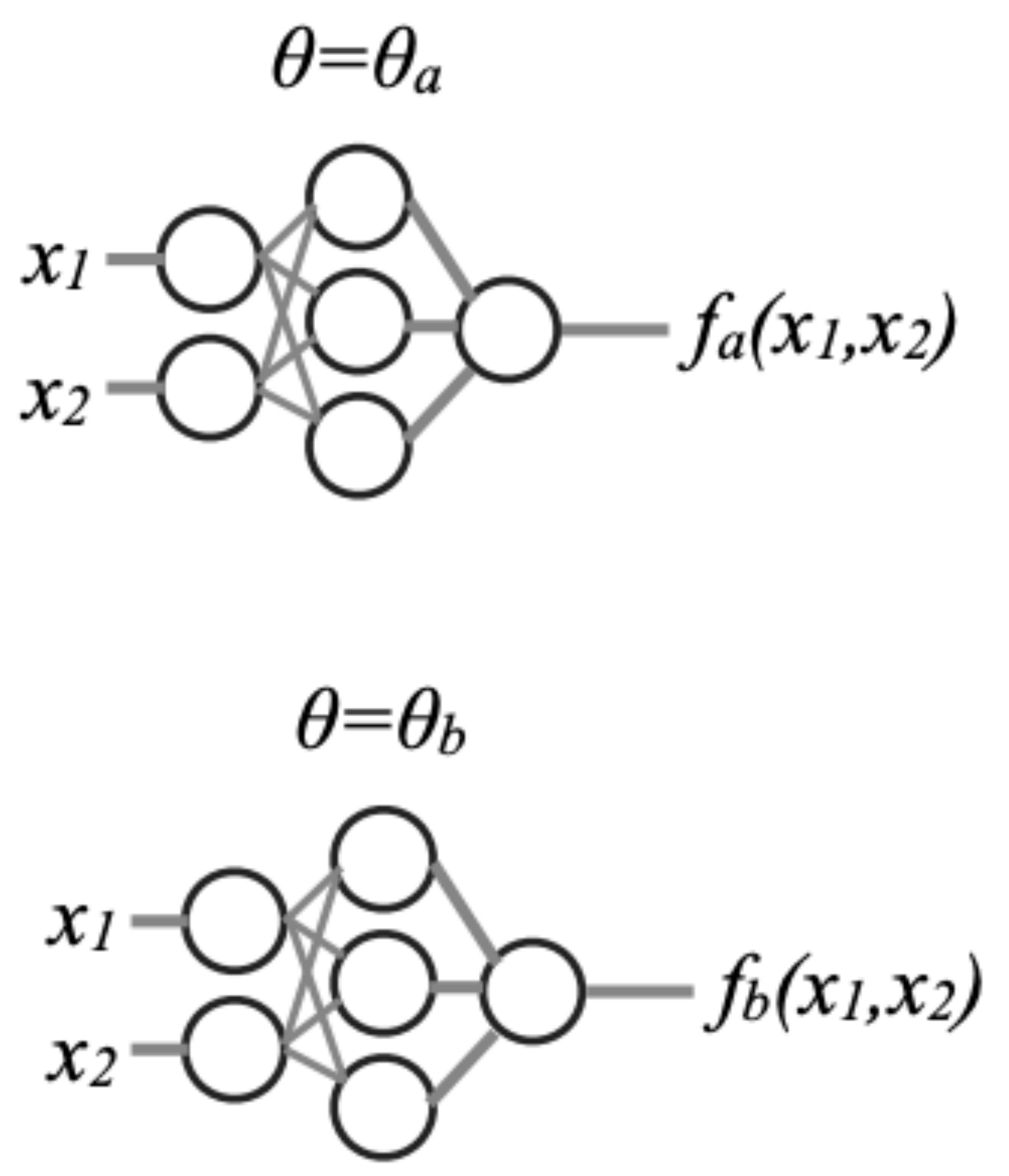
B. Target each signal individually:
   - Train an individual NN for each signal
   - Train a NN parametrised on $m_X, m_S$ (PNN)

→All these options have been tested/considered!

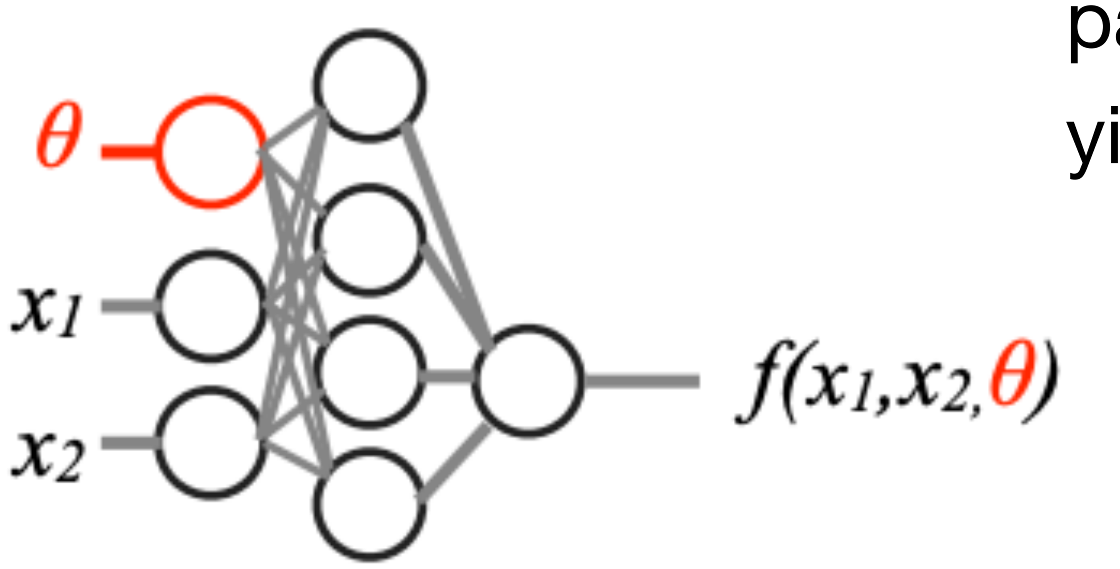


# What is a PNN?

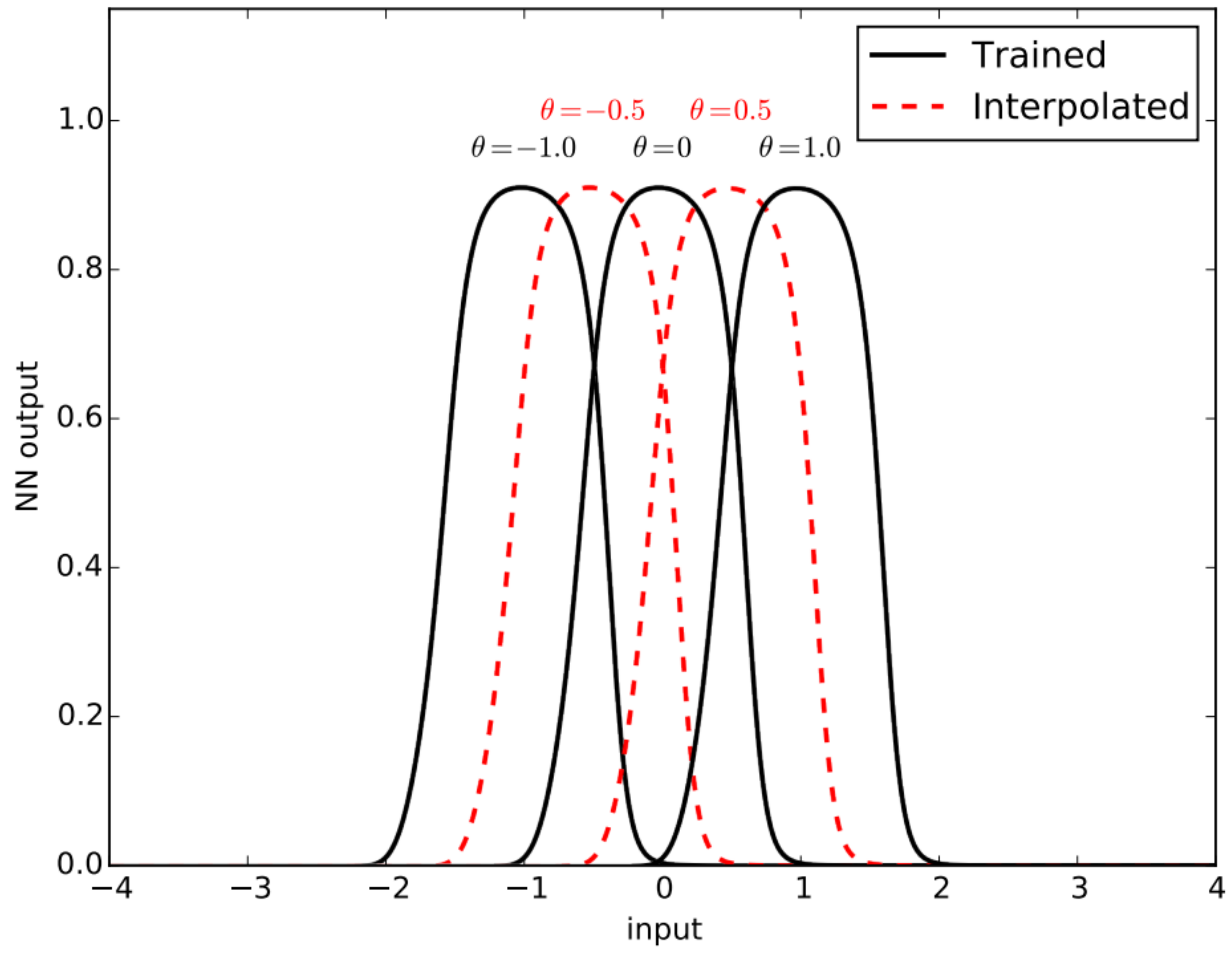
[arxiv: PML for HEP]




$\theta = m_X, m_S$




- Typical networks take a vector of features $\bar{x}$ and after training the network gives $f(\bar{x})$
- If the task is part of a larger context, described by one or more parameters $\bar{\theta}$, one can use both as input to obtain $f(\bar{x}, \bar{\theta})$ yielding different output values for different choices of $\bar{\theta}$




- Issue: Some or all of the components of $\bar{\theta}$ may not be meaningful for a particular target class (i.e. $m_X, m_S$ for background)
- Solution: randomly assign values to those components of $\bar{\theta}$ according to the same distribution used for the signal class

# Event selection

| Selection | Signal Region (SR) | Side-band (SB) |
|---|---|---|
| Number of 'tight' and isolated photons | $\geq 2$ | |
| Number of leptons | $= 0$ | |
| Number of central jets | $\in [2, 5]$ | |
| Number of b-tagged jets @ 70% WP | $\geq 1$ | |
| Number of b-tagged jets @ 77% WP | $= 2$ | |
| $m_{\gamma\gamma}$ [GeV] | $\in [120, 130]$ | $\in [105, 120] \cup [130, 160]$ |


The $m_{\gamma\gamma}$ distribution is used to divide events in a SR ($m_{\gamma\gamma} \sim m_H$) and a Control Region or SB ($m_{\gamma\gamma} \nsim m_H$)

→ The SB allows us to:

- Correct the normalisation of the large non-resonant background ($\gamma\gamma$+jets) from data
- Carefully study the full data-to-simulation comparison for the variables of interest

→ The SR gives a much better sensitivity to all of the targeted signals since in all of them $H \to \gamma\gamma$

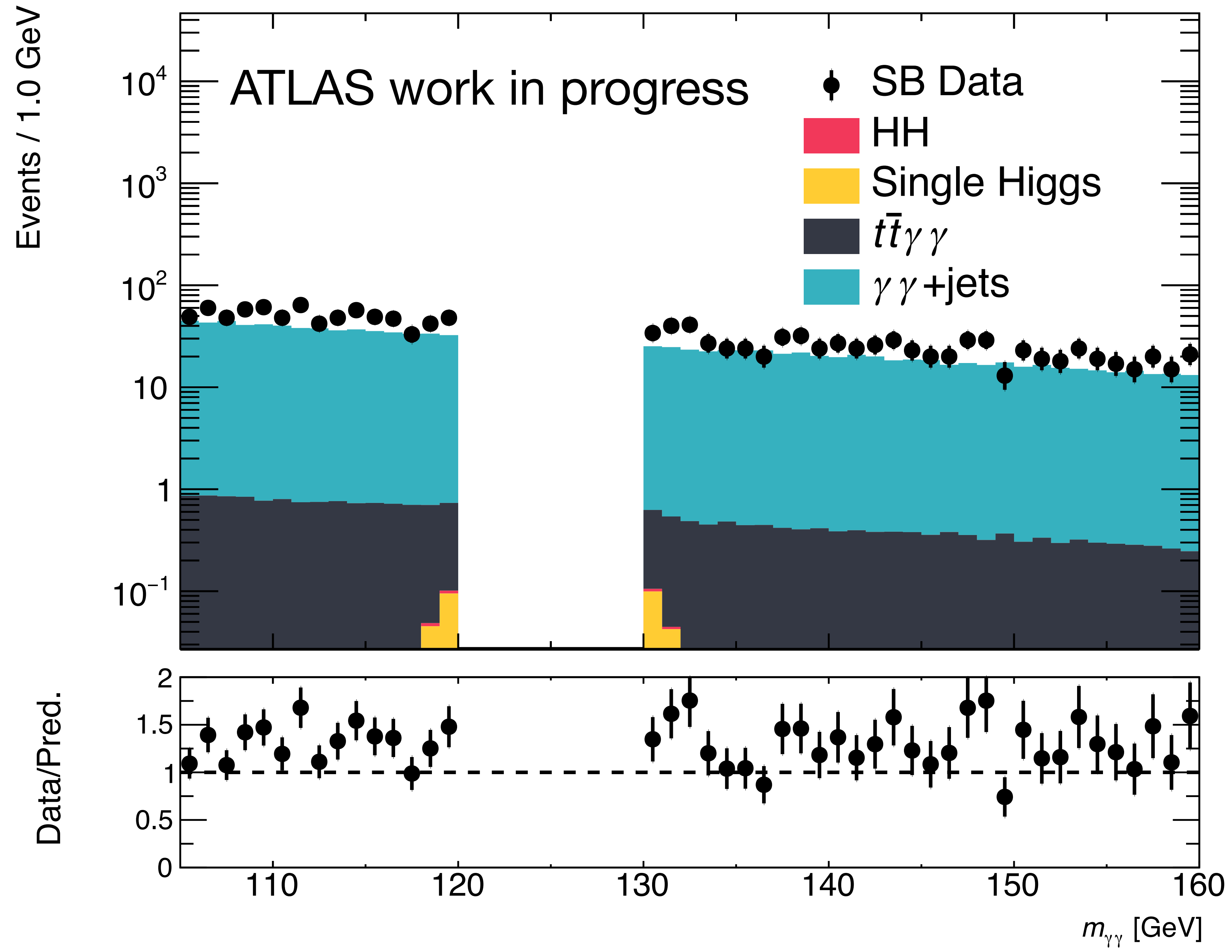
# Predicted number of events

ATLAS work in progress

| | Preselection | Side-band | SR |
|---|---|---|---|
| $HH$ | 1.615 ± 0.004 | 0.026 ± 0.001 | 1.589 ± 0.004 |
| $VBFH$ | 0.68 ± 0.012 | 0.014 ± 0.002 | 0.666 ± 0.011 |
| $W^-H$ | 0.084 ± 0.002 | 0.002 ± 0.0 | 0.082 ± 0.002 |
| $W^+H$ | 0.122 ± 0.003 | 0.003 ± 0.001 | 0.119 ± 0.003 |
| $ZH$ | 2.796 ± 0.01 | 0.057 ± 0.001 | 2.739 ± 0.01 |
| $bbH$ | 0.615 ± 0.023 | 0.016 ± 0.004 | 0.598 ± 0.023 |
| $ggH$ | 5.41 ± 0.065 | 0.108 ± 0.009 | 5.302 ± 0.064 |
| $ggZH$ | 0.865 ± 0.007 | 0.016 ± 0.001 | 0.85 ± 0.007 |
| $tHjb$ | 0.962 ± 0.029 | 0.021 ± 0.004 | 0.94 ± 0.028 |
| $tWH$ | 0.13 ± 0.005 | 0.003 ± 0.001 | 0.127 ± 0.005 |
| $ttH$ | 8.247 ± 0.014 | 0.204 ± 0.002 | 8.043 ± 0.014 |
| $t\bar{t}\gamma\gamma$ all-had | 12.804 ± 0.045 | 10.253 ± 0.04 | 2.551 ± 0.02 |
| $t\bar{t}\gamma\gamma$ no-allhad | 15.304 ± 0.097 | 12.22 ± 0.086 | 3.084 ± 0.043 |
| $\gamma\gamma$+jets | 1388.15 ± 4.489 | 1109.74 ± 4.014 | 278.419 ± 2.01 |
| Total SM | 1437.783 ± 4.491 | 1132.684 ± 4.015 | 305.108 ± 2.012 |
| Data | - | 1482 | - |
| $\mathrm{NF}_{\gamma\gamma\text{+jets}}$ | - | 1.31 | - |
| $m_{X,S} = (1000, 300)$ | 31.034 ± 0.216 | 0.87 ± 0.036 | 30.164 ± 0.213 |
| $m_{X,S} = (190, 50)$ | 1.211 ± 0.042 | 0.059 ± 0.009 | 1.152 ± 0.041 |
| $m_{X,S} = (250, 110)$ | 9.721 ± 0.119 | 0.461 ± 0.026 | 9.261 ± 0.116 |
| $m_{X,S} = (300, 110)$ | 11.079 ± 0.127 | 0.494 ± 0.027 | 10.585 ± 0.124 |
| $m_{X,S} = (600, 170)$ | 23.446 ± 0.187 | 0.738 ± 0.033 | 22.708 ± 0.184 |
| $m_{X,S} = (750, 300)$ | 28.018 ± 0.205 | 0.843 ± 0.036 | 27.174 ± 0.202 |

The largest non-resonant and resonant backgrounds are $\gamma\gamma$+jets, $ttH$, $ggH$ and $ZH$

High mass signals have a much larger selection efficiency than low mass



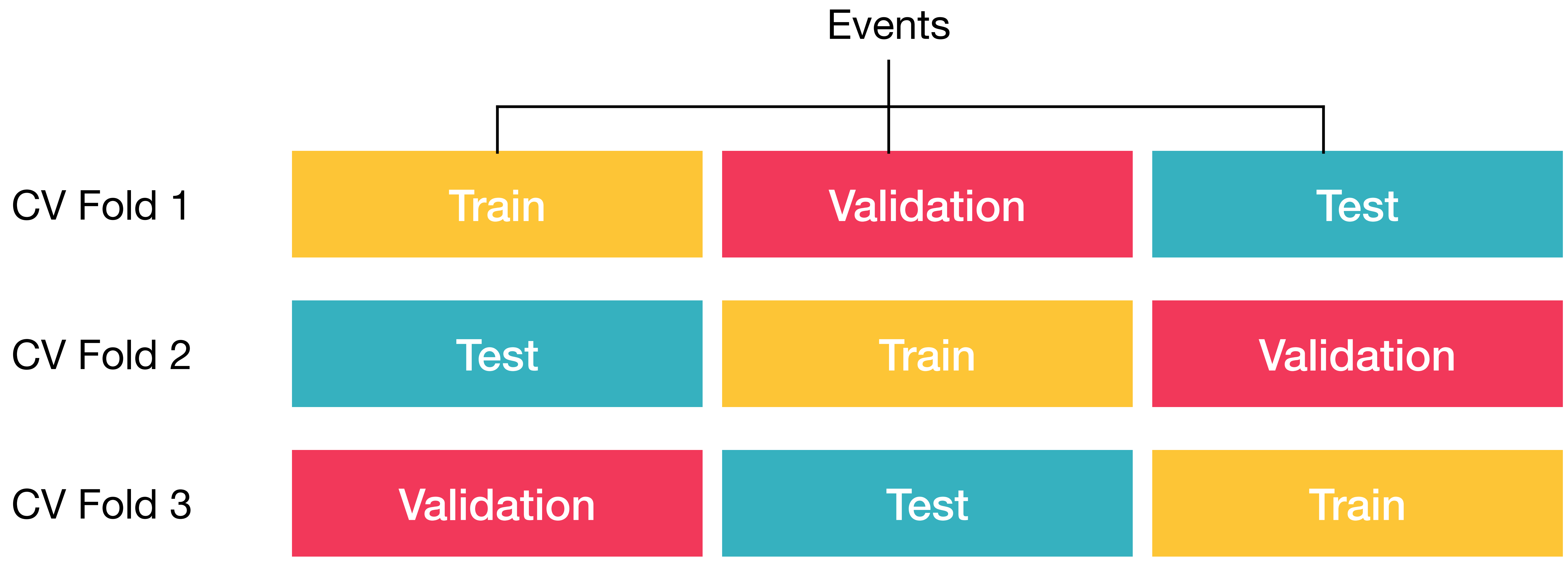

$m_{\gamma\gamma}$

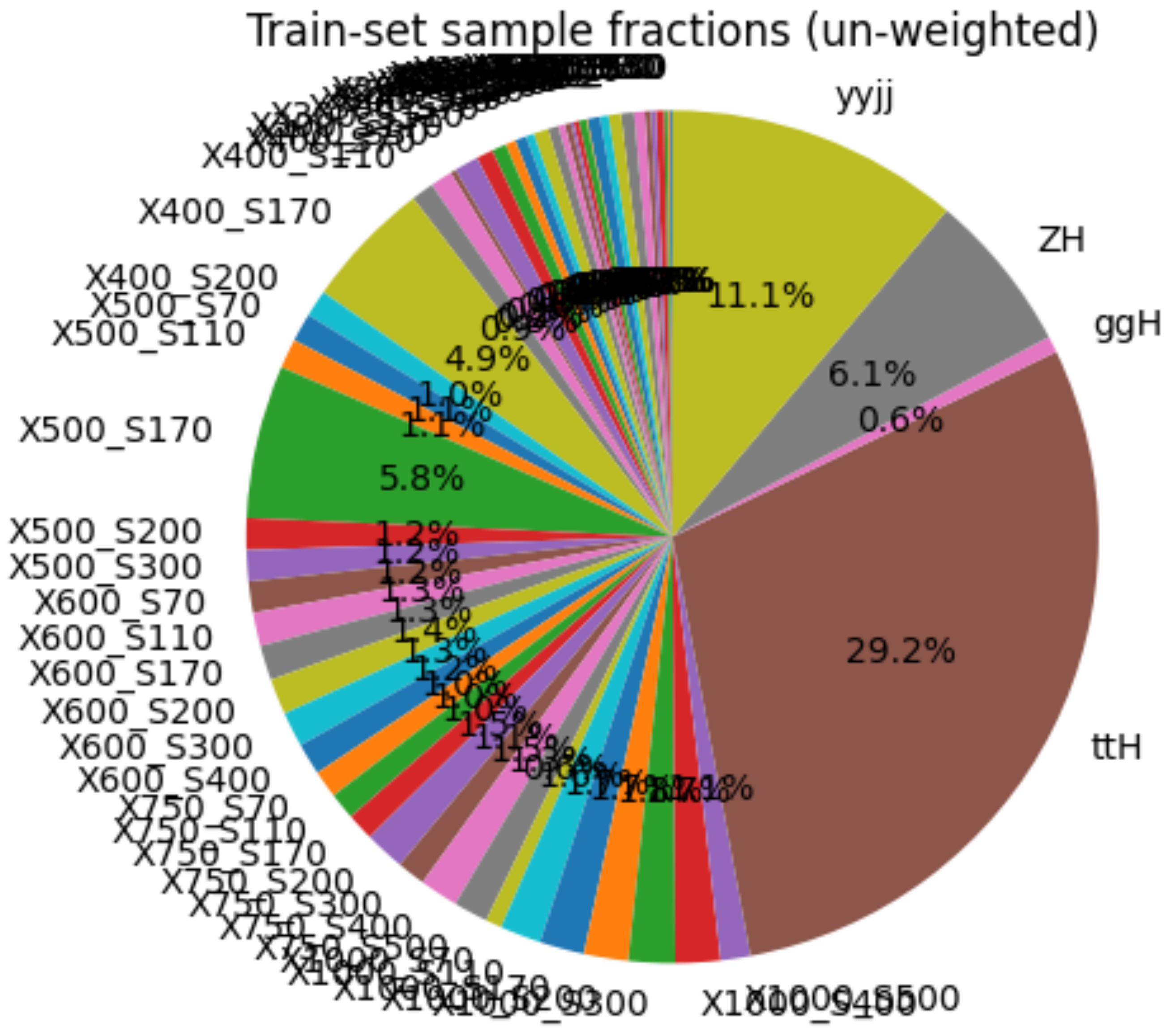
# Set splitting

- The amount of available labeled data is limited so we need to split it properly to assess the performance of our network!

  - Train set: used for training
  - Validation set: used for assessing the performance on unseen data
  - Test set: used to test the final performance in a completely unbiased data

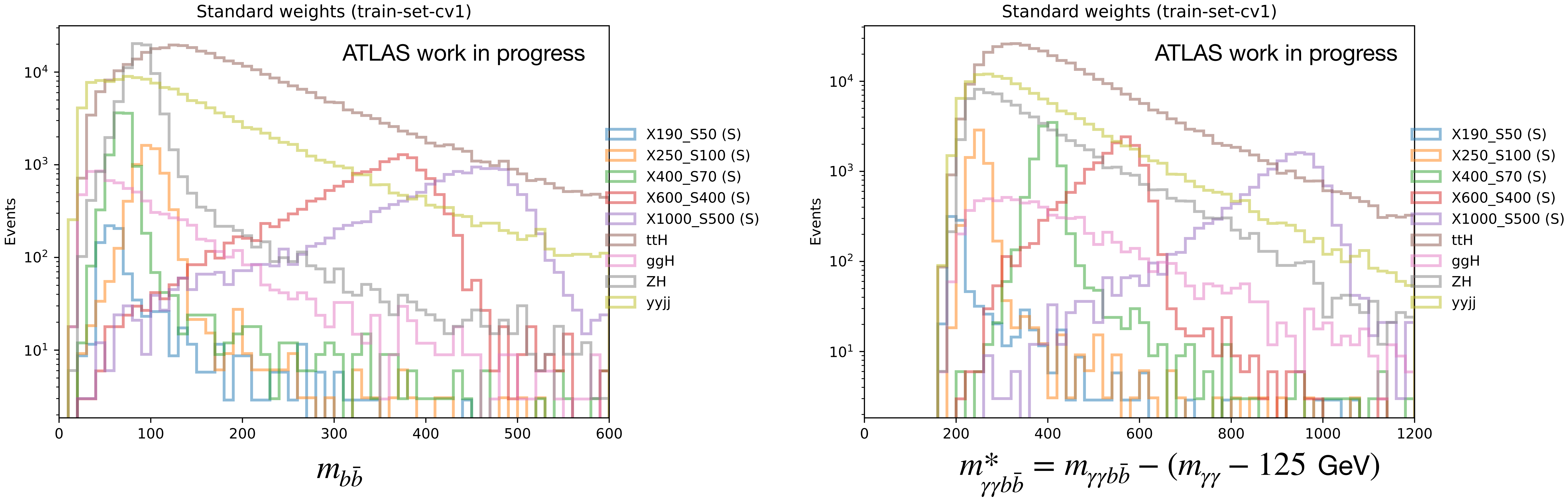
- Cross validation (CV) is used to recover all of the events which will be needed for the statistical analysis



# Input variables

- The natural choice of input variables for a PNN with $\theta = m_X, m_S$ are $m_{\gamma\gamma b\bar{b}} \sim m_X$ and $m_{b\bar{b}} \sim m_S$
- To avoid correlations with $m_{\gamma\gamma}$, which is used to define the SR and SB, the modified $m^*_{\gamma\gamma b\bar{b}} = m_{\gamma\gamma b\bar{b}} - (m_{\gamma\gamma} - 125$ GeV) is used instead of $m_{\gamma\gamma b\bar{b}}$




- **Less is more**: We tested including other variables to the trainings but they seem to confuse the PNN more than help it - we think this is due to our limited amount of data and training time

# Hyperparameter optimisation

Parameters are optimised by Keras tuner (maximises the Area-Under-the-Curve of the validation set)

| Parameter | Range | Sampling Mode |
|---|---|---|
| Number of hidden layers | [1-6] | 1 |
| Dropout rate (per-layer) | [0, 0.2] | 0.05 |
| Learning rate | [0.0001 - 0.1] | Log |

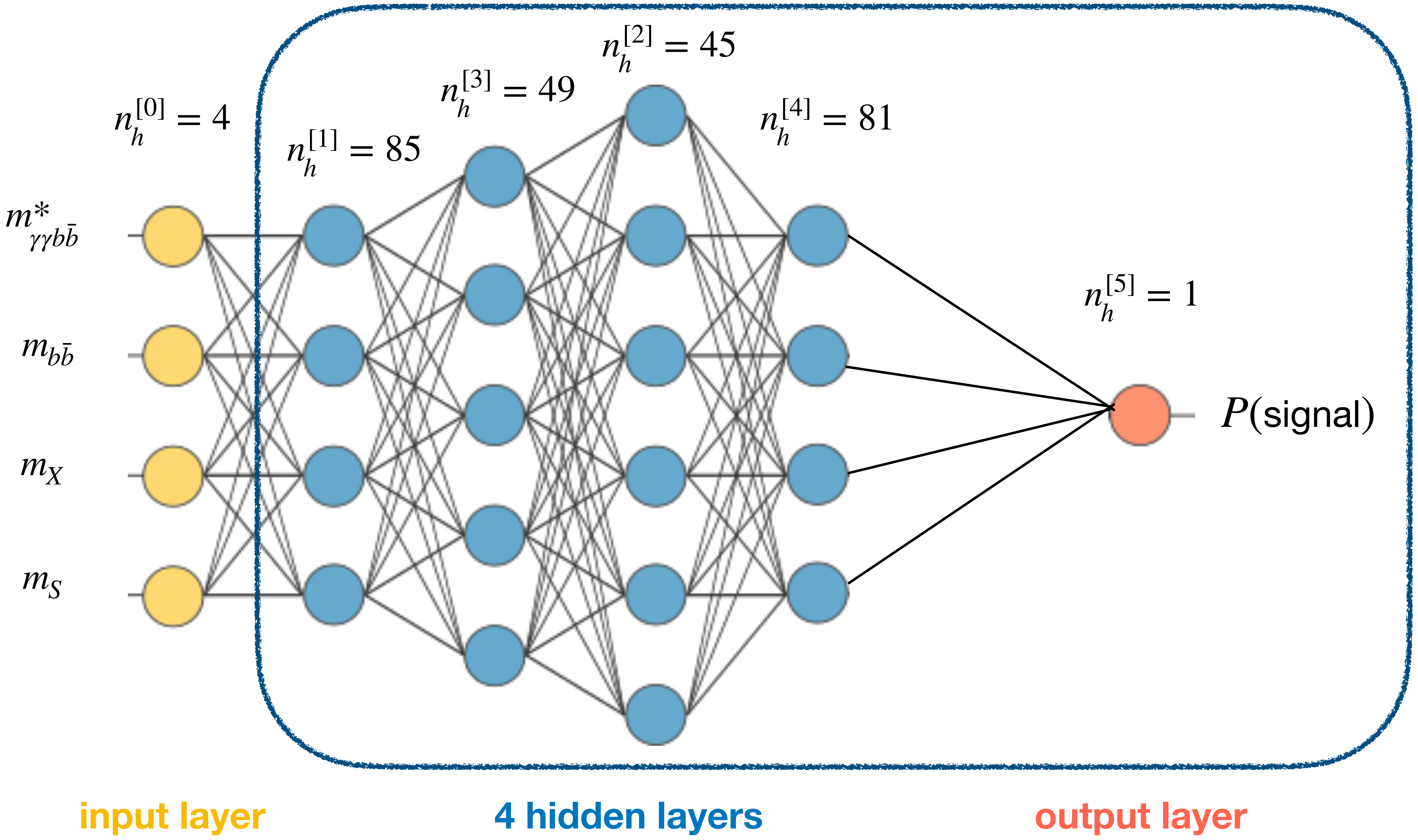
The dataset is very unbalanced - the number of background events is much larger than the number of signal events - the following parameters are defined to help a NN or PNN learn to separate signal from background:

- Class weight: $\text{weight}(i) = \dfrac{\#\ \text{total events}}{2 \cdot \#i\ \text{events}}$ for $i =$ signal or background
- Initial bias in output layer: $b = \dfrac{\#\text{signal events}}{\#\text{backgorund events}}$
- Batch size: Minimum size such that there are ~ 200 events of each signal (avoids confusing the network due to statistical fluctuations in the number of signal events)

# PNN architecture

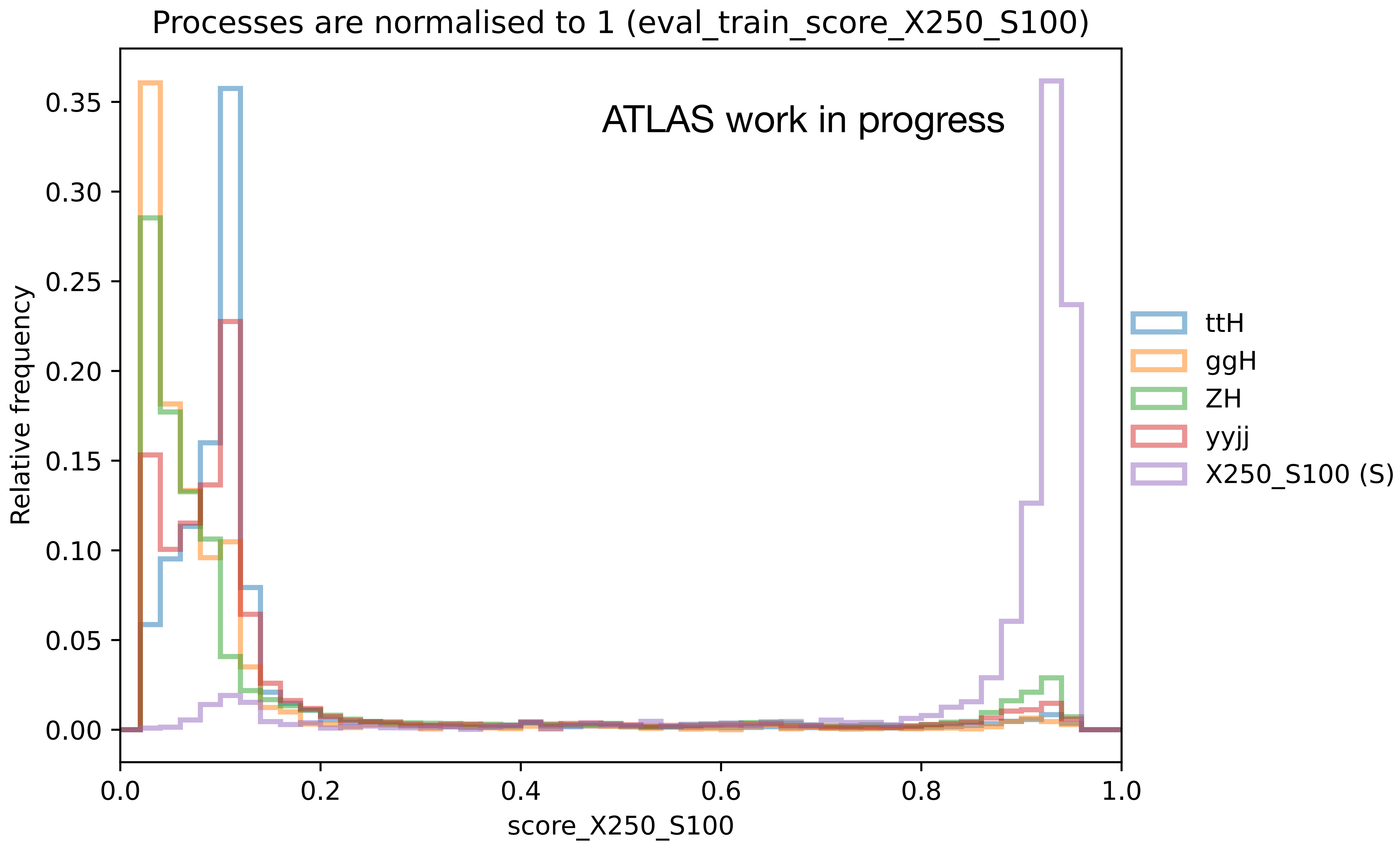
Chosen hyper-parameters

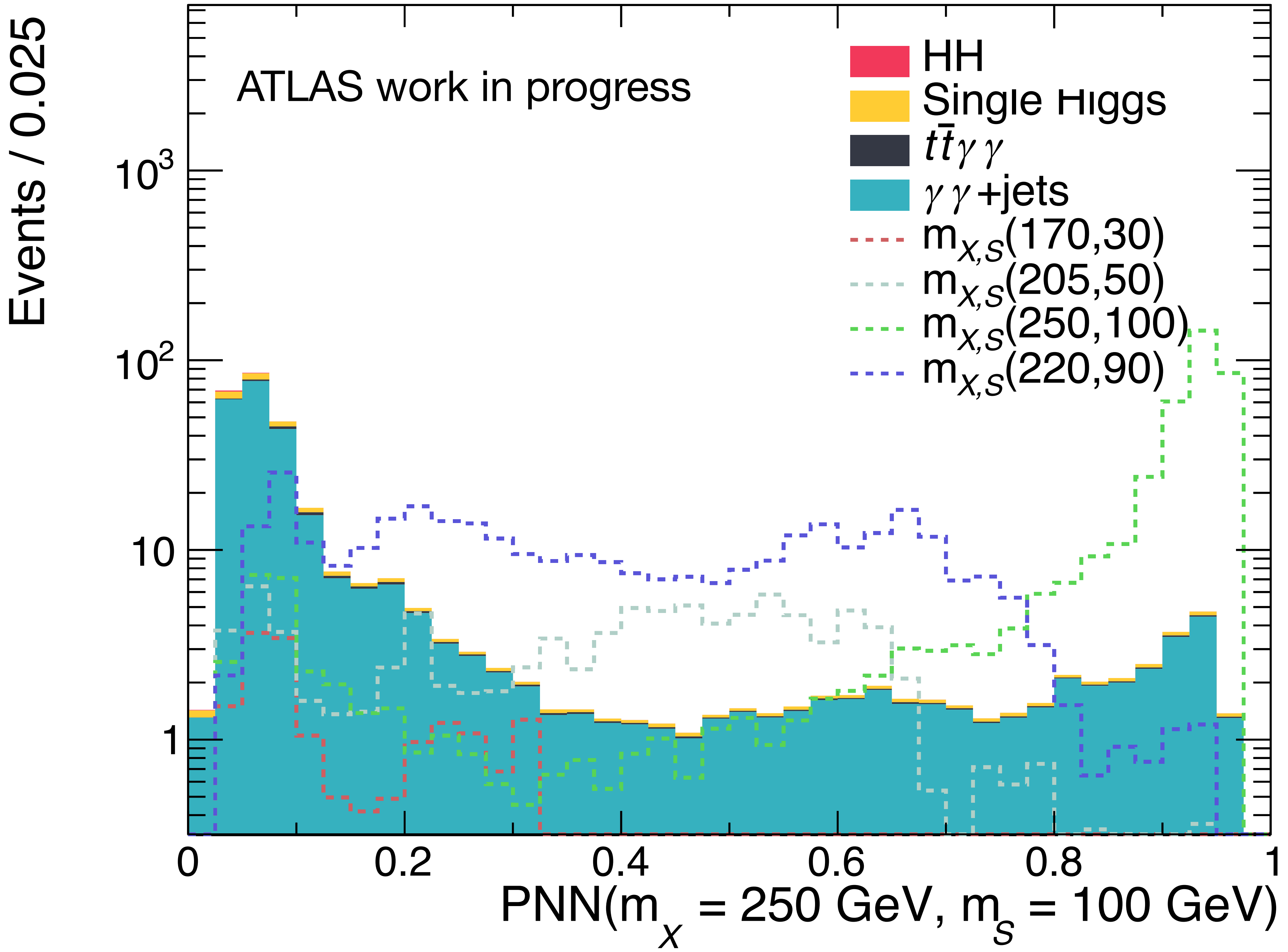
| | |
|---|---|
| Number of hidden layers | 4 |
| Layer 1 dropout rate | 0.05 |
| Layer 2 droport rate | 0.1 |
| Layer 3 dropout rate | 0.2 |
| Layer 4 dropout rate | 0.1 |
| Learning rate | 0.009137 |
| 0ptimizer | Adam |
| Loss function | Binary Loss |
| Initial bias | 0.118 |
| Signal class weight | 0.945 |
| Background class weight | 1.062 |
| Batch size | 212613 |
| Number of batches | 2 |
| Number of epochs | 2000 |




Activation functions: ReLu for all hidden layers and a sigmoid for output layer

# PNN output



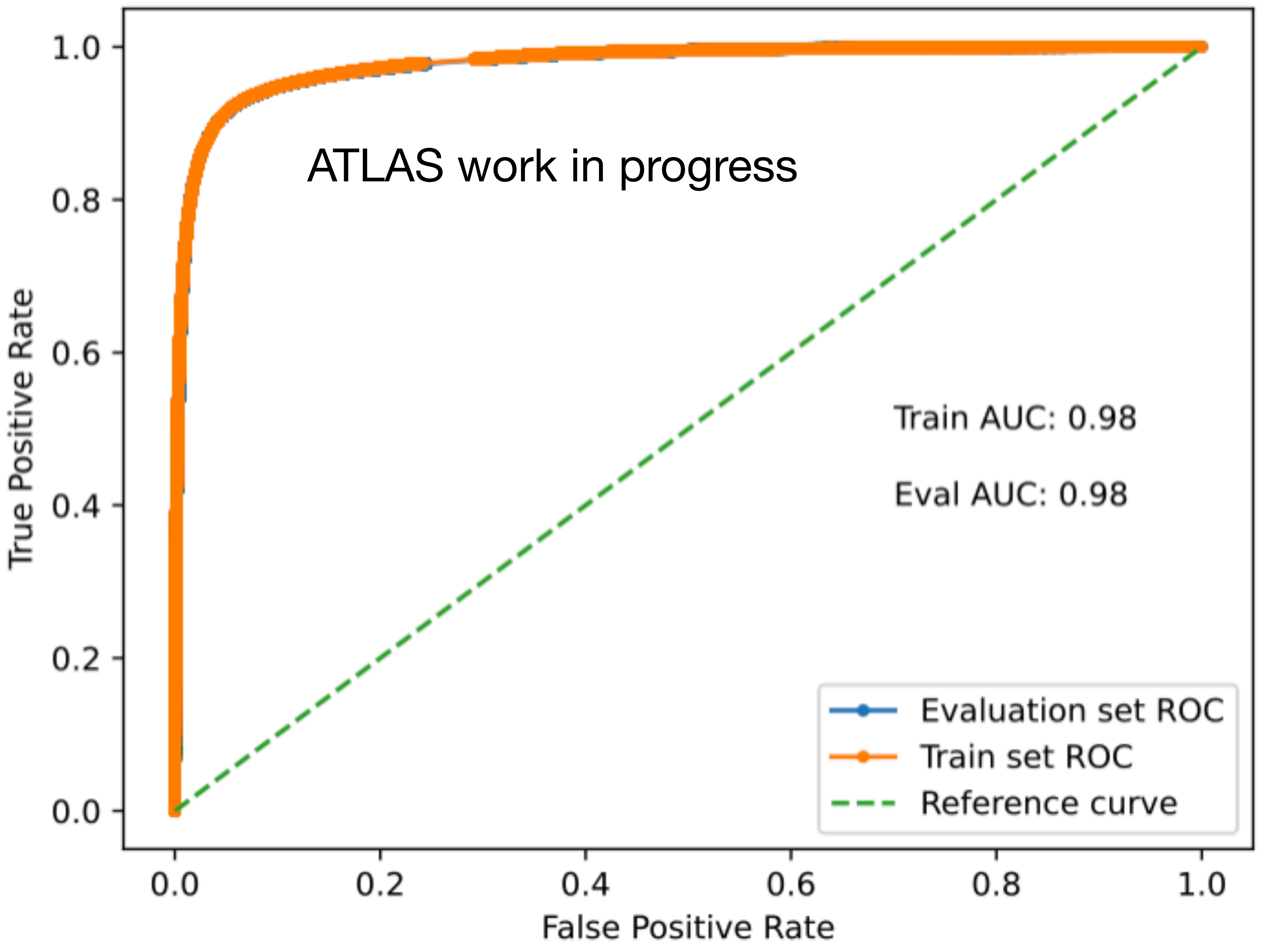




- The PNN score fit is capable of discriminating the targeted $\theta$ $(m_X, m_S)$ signal from background
- Signals with $m_X, m_S$ values far away from $\theta$ are easily classified as background ($P$(signal) = 0= while signals with $m_X, m_S$ ~ $\theta$ result in $P$(signal) ~ 0.5

# PNN performance

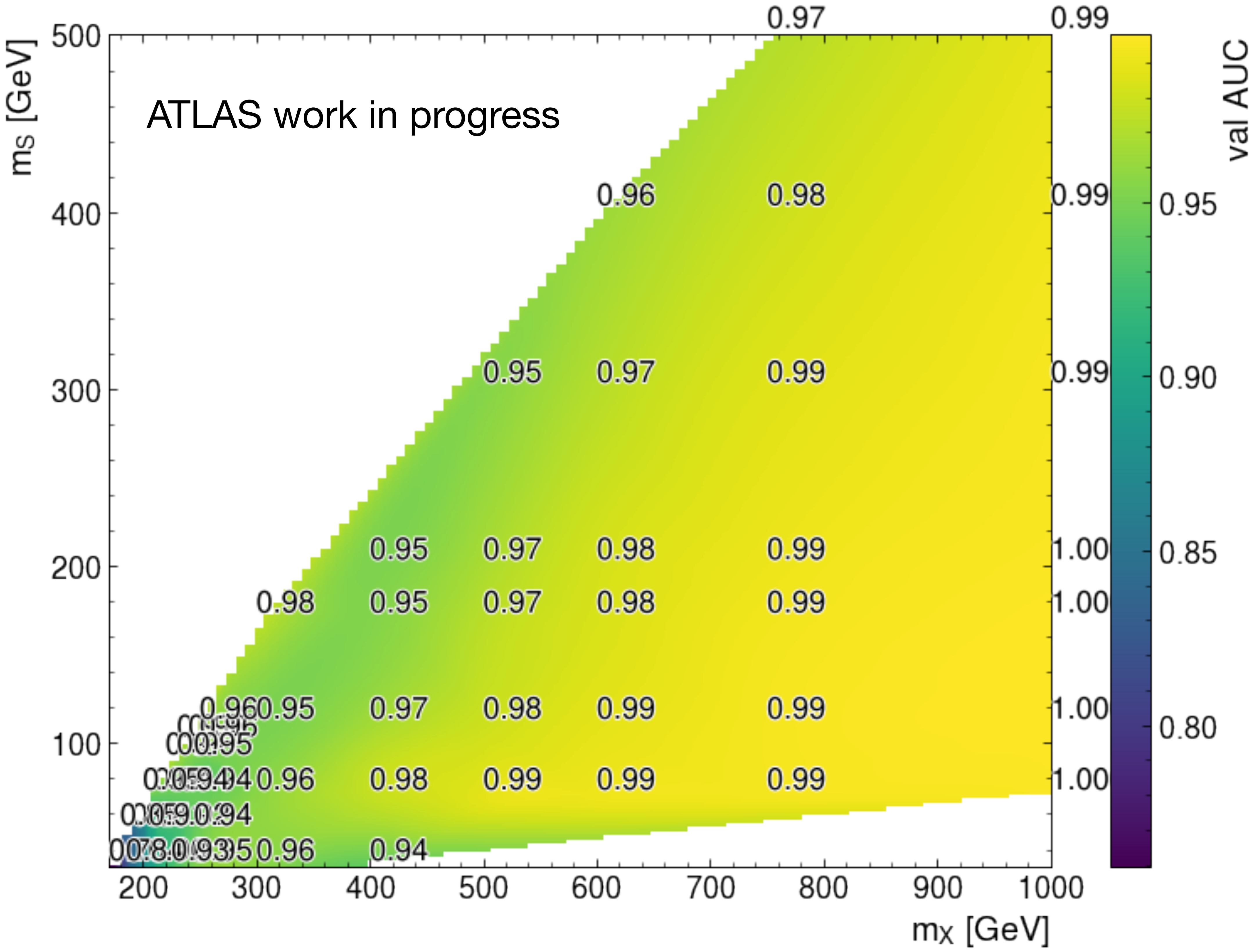
CV fold 1 (similar results for folds 2 and 3)

AUC for train and validation sets



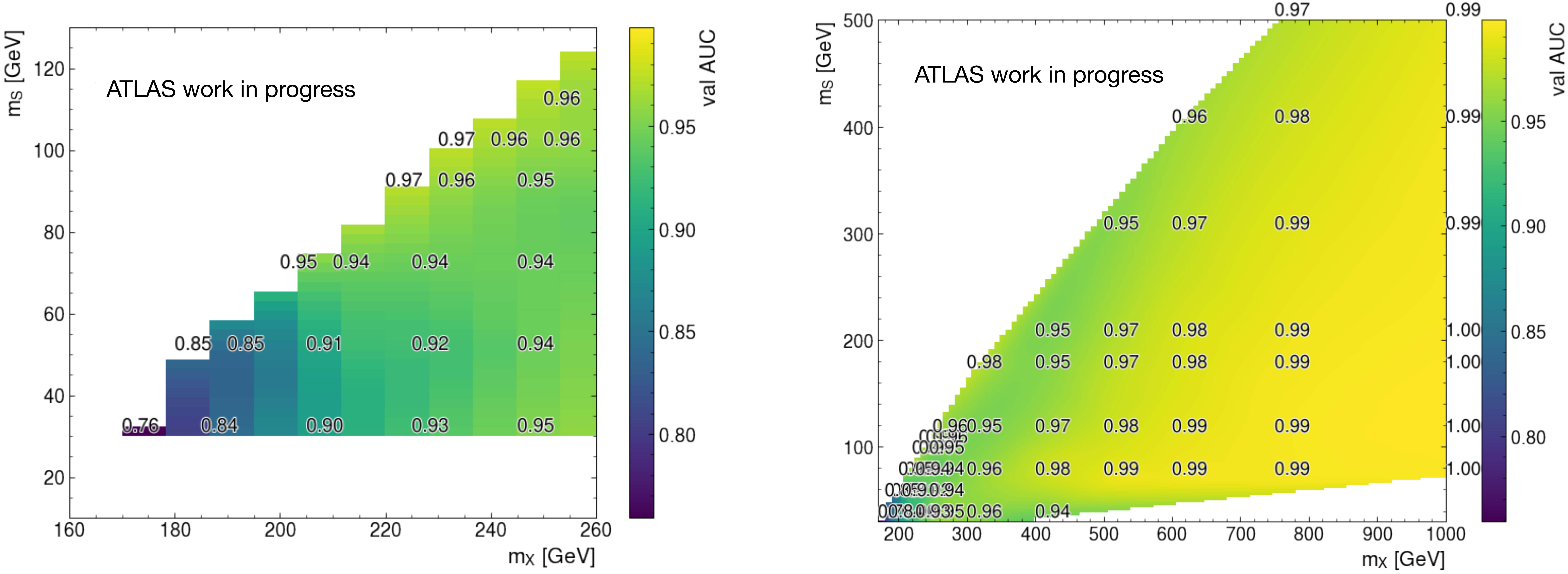

Independent of the $(m_X, m_S)$ parameters

Validation set AUC$(m_X, m_S)$




- Very similar AUC$(m_X, m_S)$ values are obtained for the train and validation sets indicating that there is no overtraining

# Let's take a closer look!




- The PNN is better at separating better signals with high $m_X$ than with low
- Would a NN do better?

# NN vs PNN?

- NNs have been trained with the same input variables as the PNN for 3 specific signals.
- The hyper-parameter optimisation is performed in order to find the best architecture for each case.

| AUC | PNN val (train-set) | NN val (train-set) |
|---|---|---|
| X170_S130 | 0.76 (0.74) | 0.83 (0.82) |
| X250_S100 | 0.96 (0.96) | 0.95 (0.95) |
| X750_S110 | 0.99 (0.99) | 0.99 (0.99) |

→ The NN performs 9% better than the PNN

→ Similar performance

- The NNs also perform worst at low $m_X$ → This signal is simply harder to separate from background
- The PNN performs better than the NN at intermediate $m_X$ values → The PNN can learn from other masses improving the performance with respect to the NN

# Conclusions

- A single NN parametrised as a function of 2 resonant masses $(m_X, m_S)$ can be used to separate multiple resonant signals from background
- Given the difficulty of the task, the network performs better when the bare minimum amount of variables are given
  $\rightarrow$ **LESS is MORE**
- The PNN performs equal to or better than an individual NN trained with the same input variables for most $(m_X, m_S)$ values
  - The NN is only 9% better than the PNN for the signal which is the most difficult to separate from background
  - Training a single NN to target multiple signals (not shown today) gives the worst performance

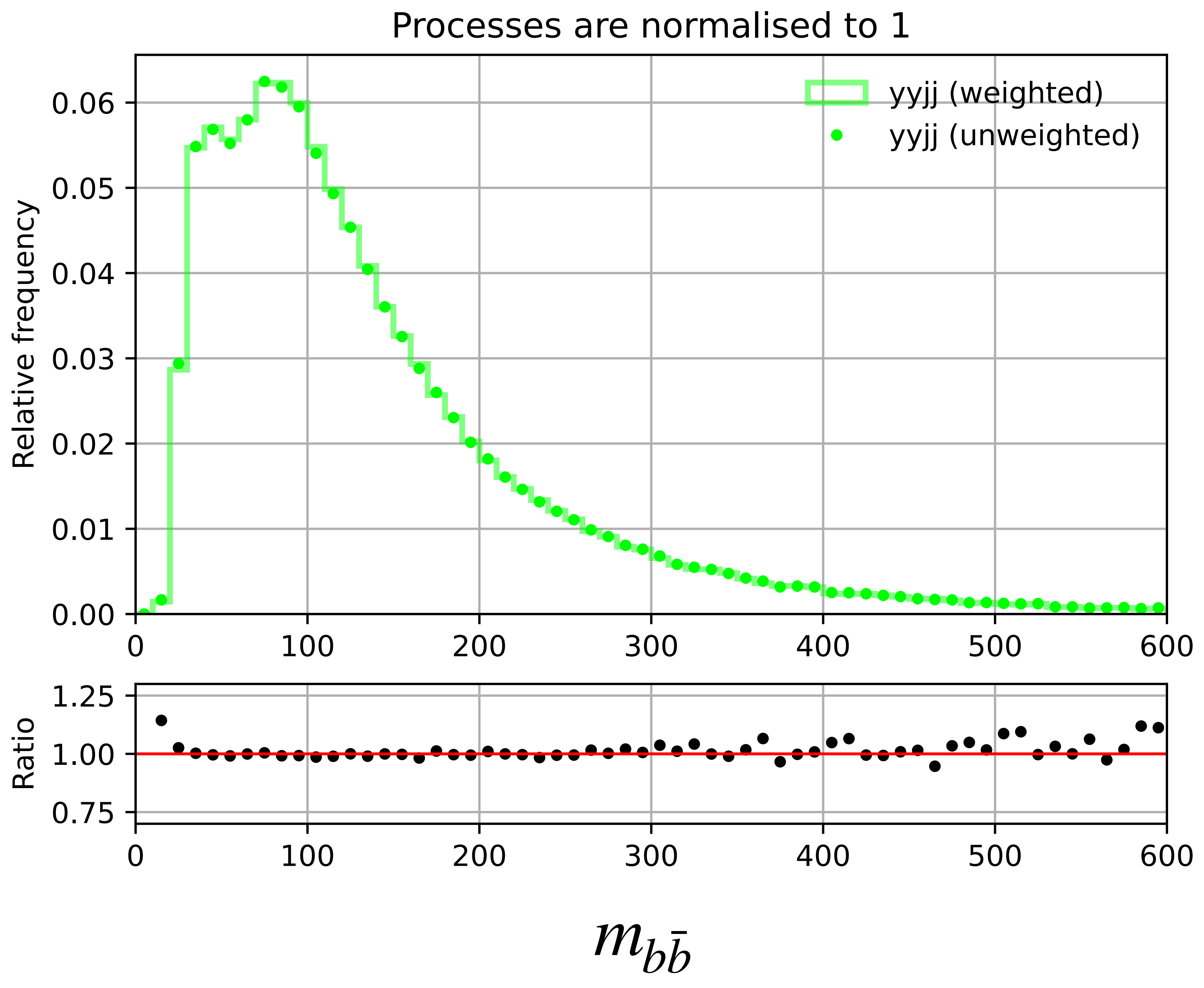
# Backup

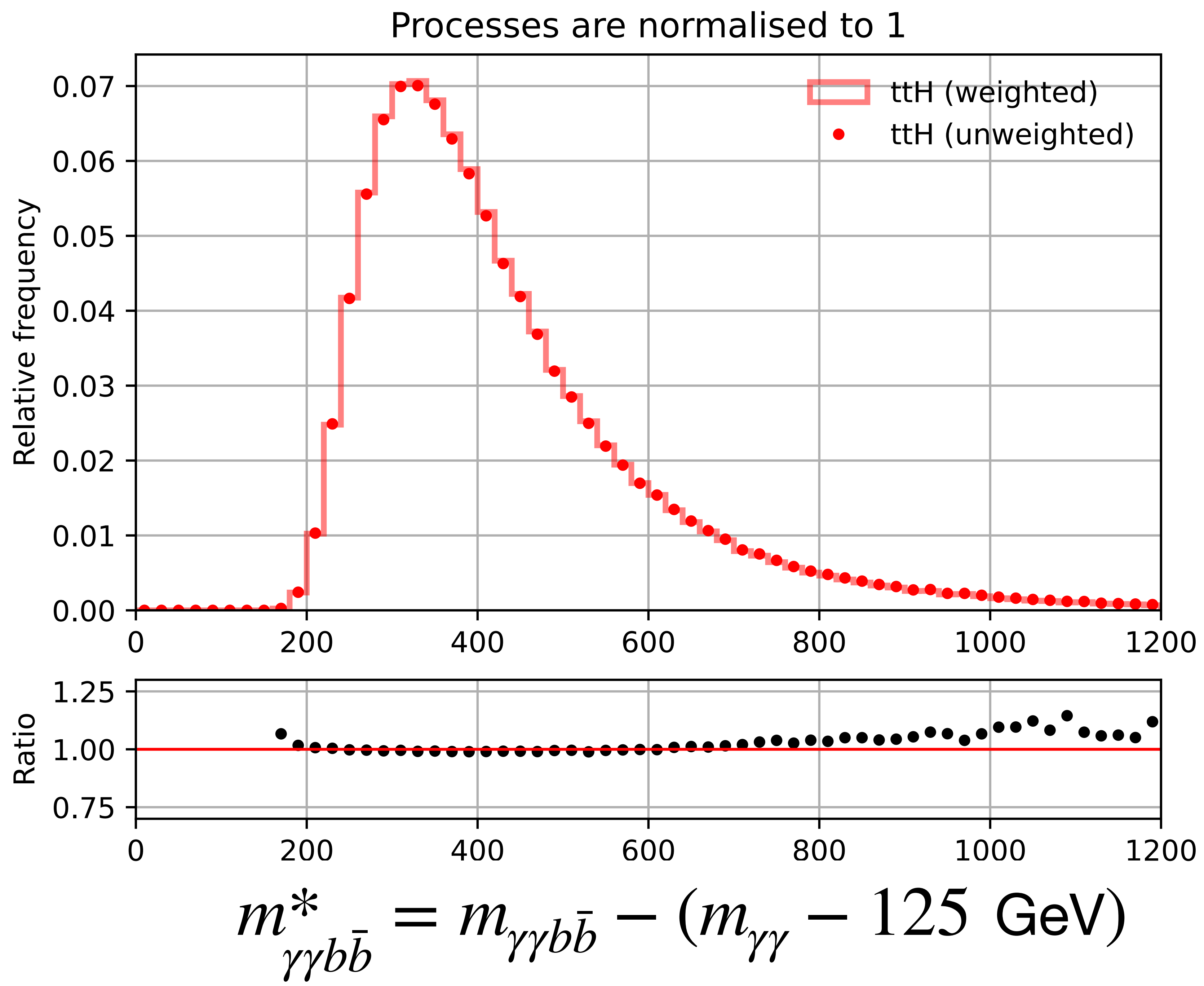
# Training with MC event weights - Yay or Nay?

The NNs shown today do not use MC events weights because

- Not including event weights speeds the training of the NN considerably
- The network learns better when not using event-weights (improvement in performance)

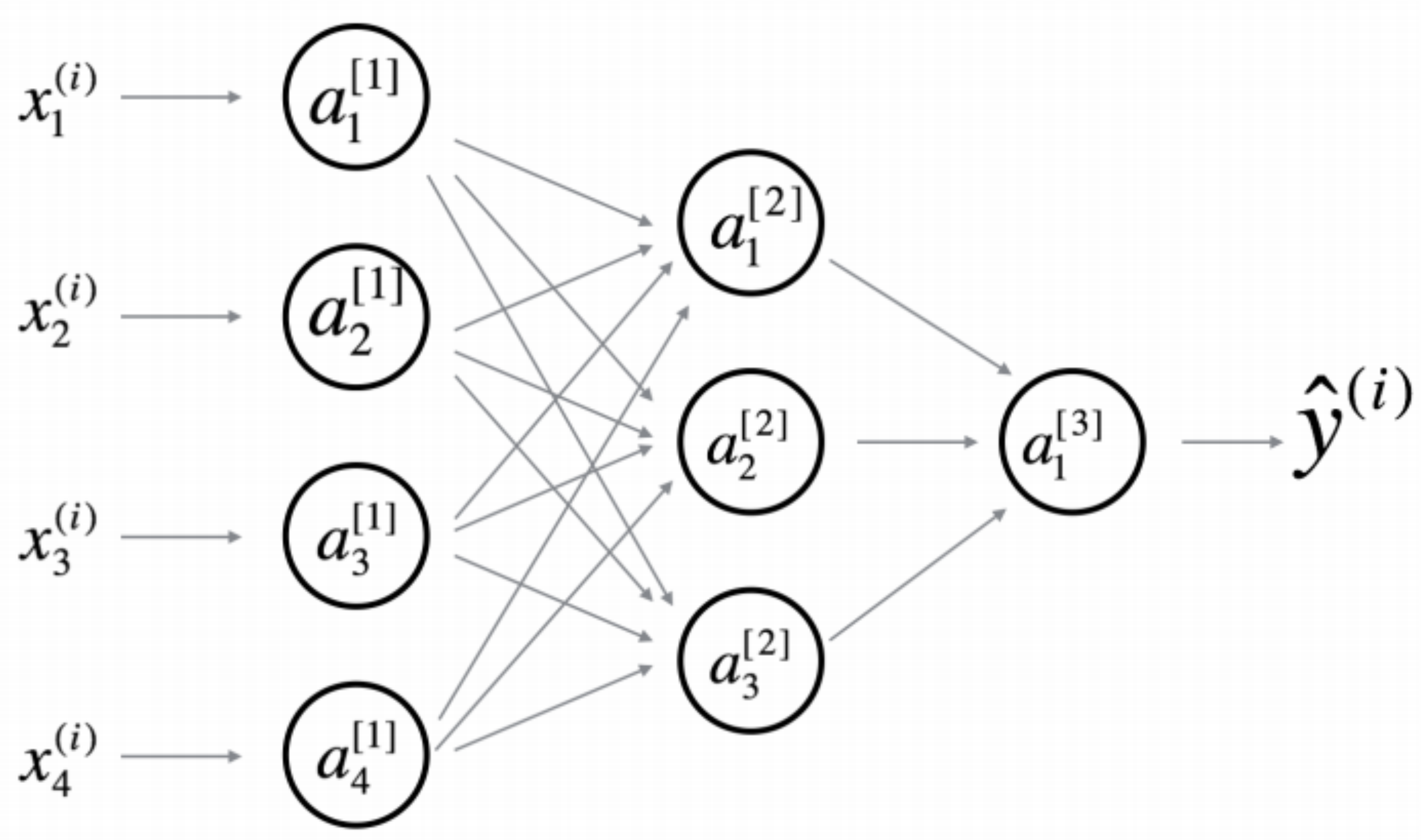
My suggestion → Consider using class weights instead!

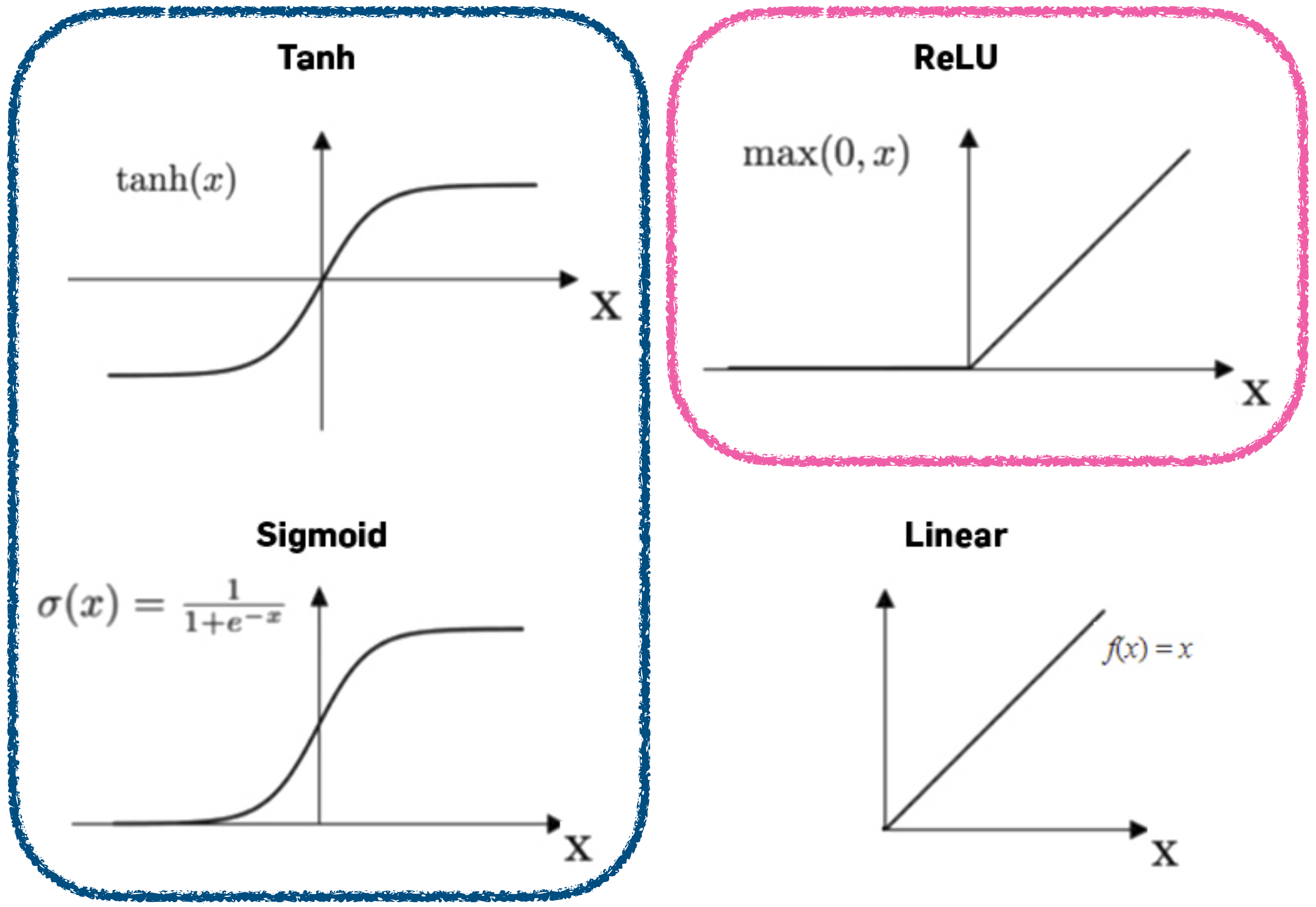


# Activation functions

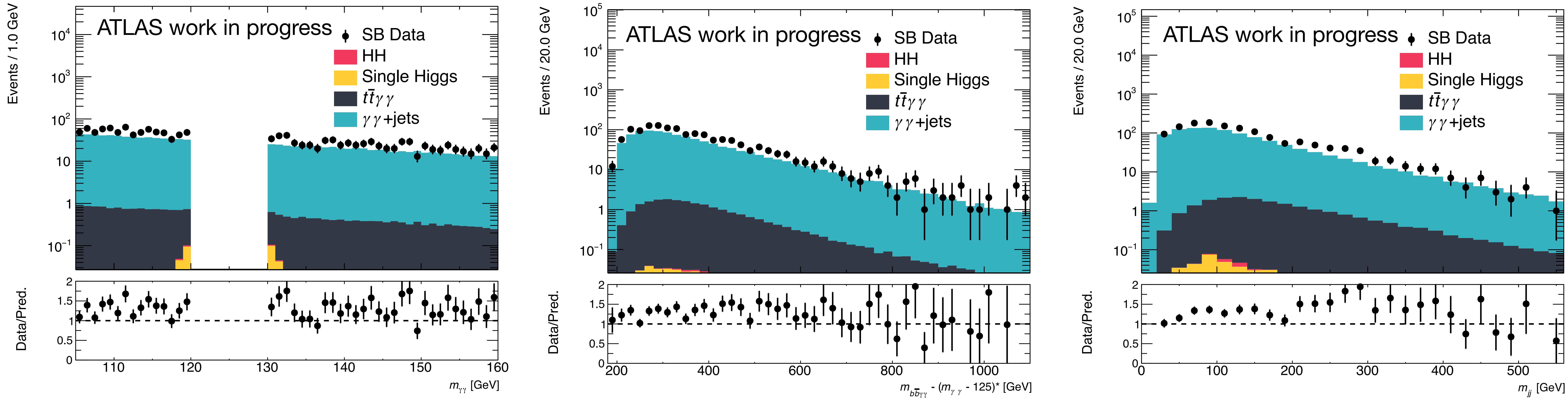
- Each neural network neuron has an activation function.
- Different functions are often used in different layers
- These functions are used to transform
  - $Z^{[k]} = w_k^{[k]} \cdot a^{[k-1]} + b^{[k]}$




- ReLu activation functions are most commonly used in the hidden layers.
- Sigmoid/tanh activation functions are often used in the output layers because they predict values $\in [0,1]$ $([-1,1])$. They also tend to slow down the training when used in hidden layers of deep networks.

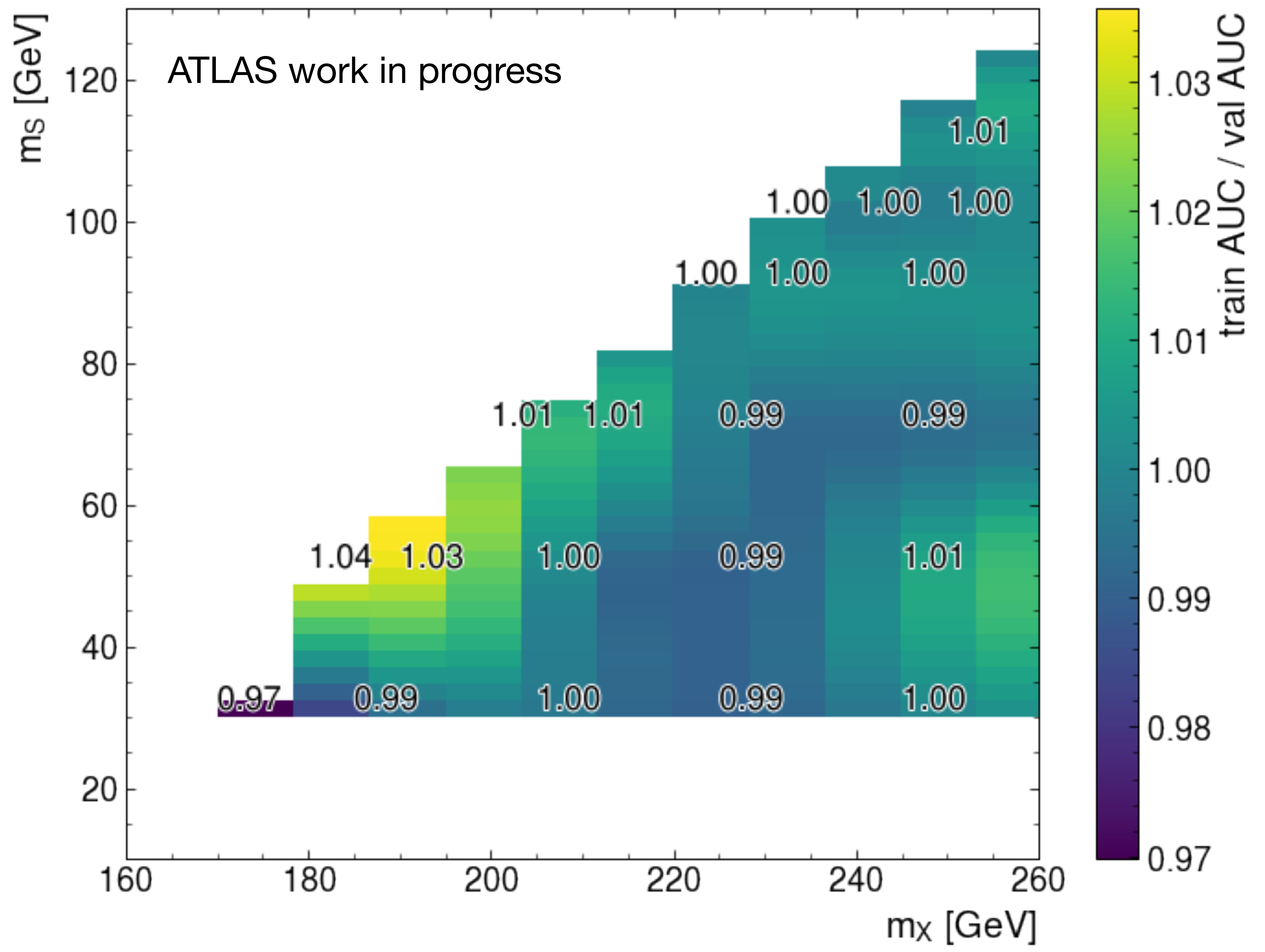
# Data/MC in SB

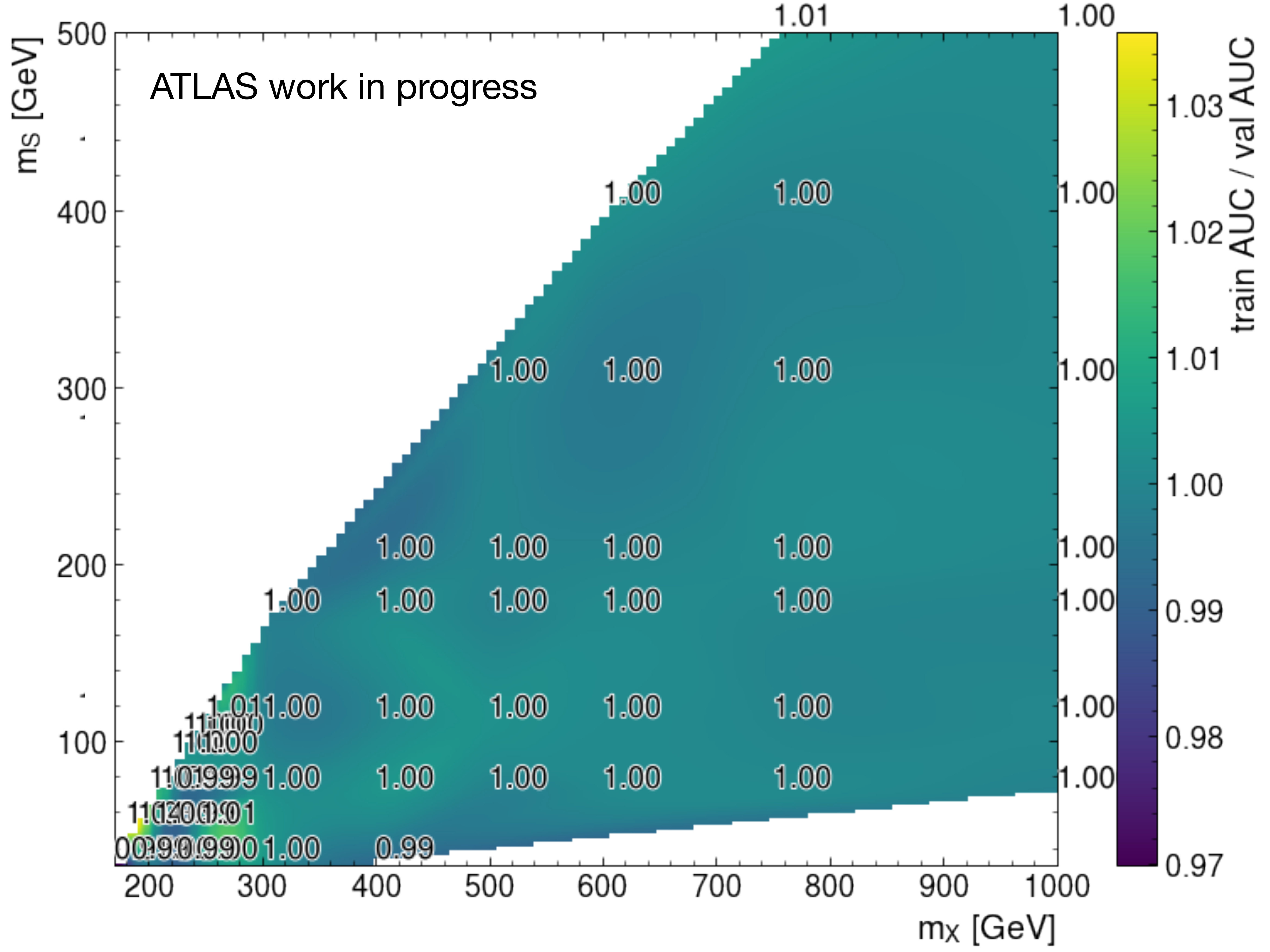



$m_{\gamma\gamma}$ $m^*_{b\bar{b}\gamma\gamma}$ $m_{jj}$

# Is there overtraining for any $(m_X, m_S)$?

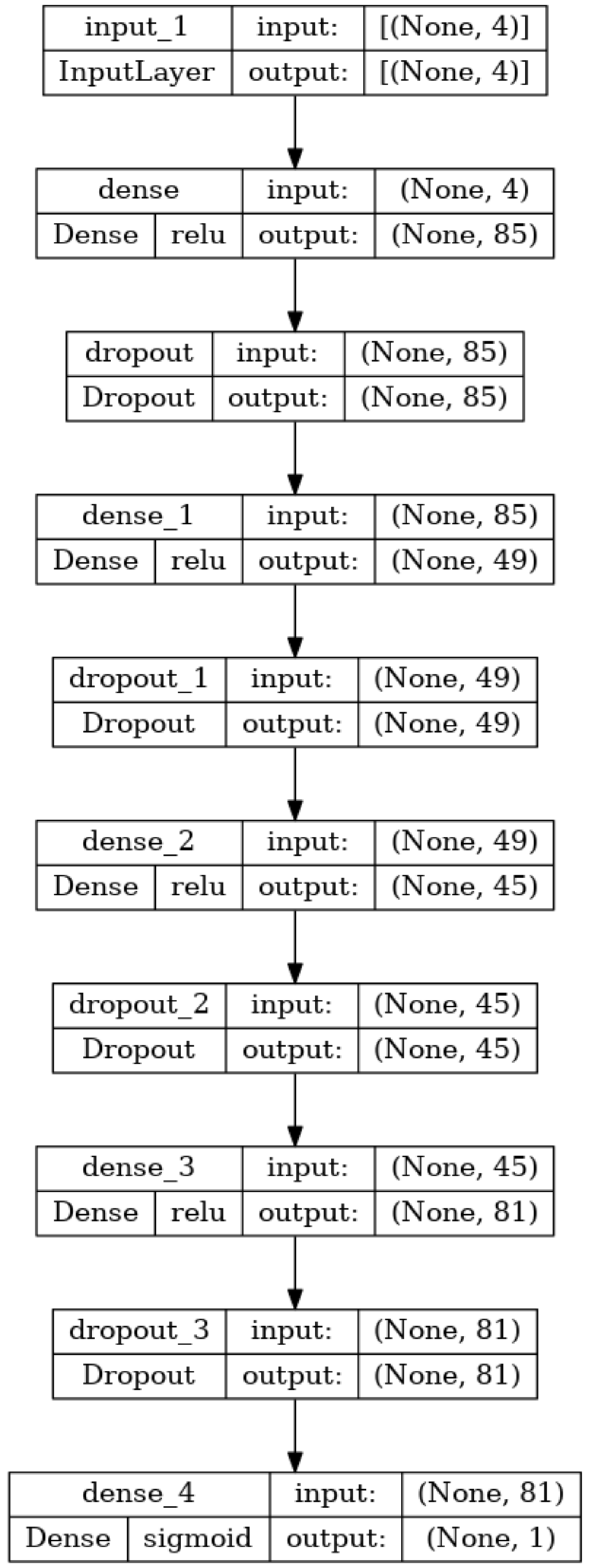
- Train AUC / Val AUC is always very close to 1 so it doesn't suggest any overtraining



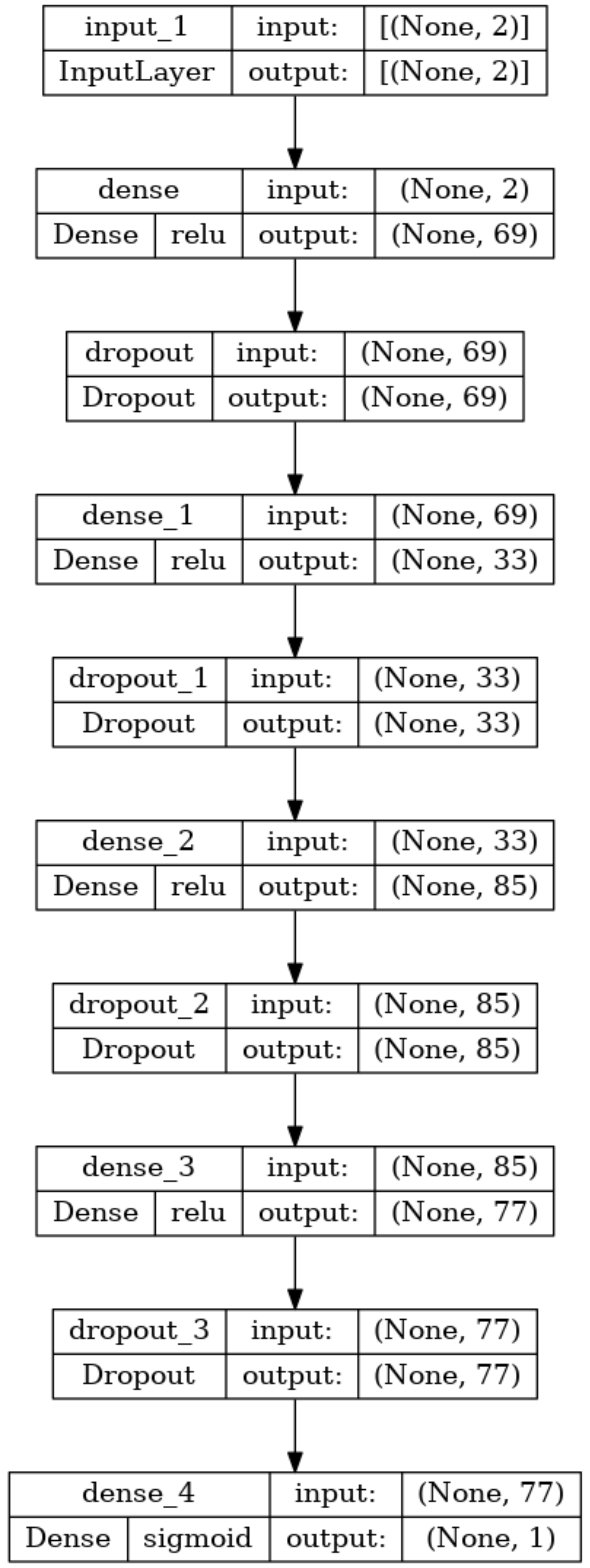



# Comparing NN architectures

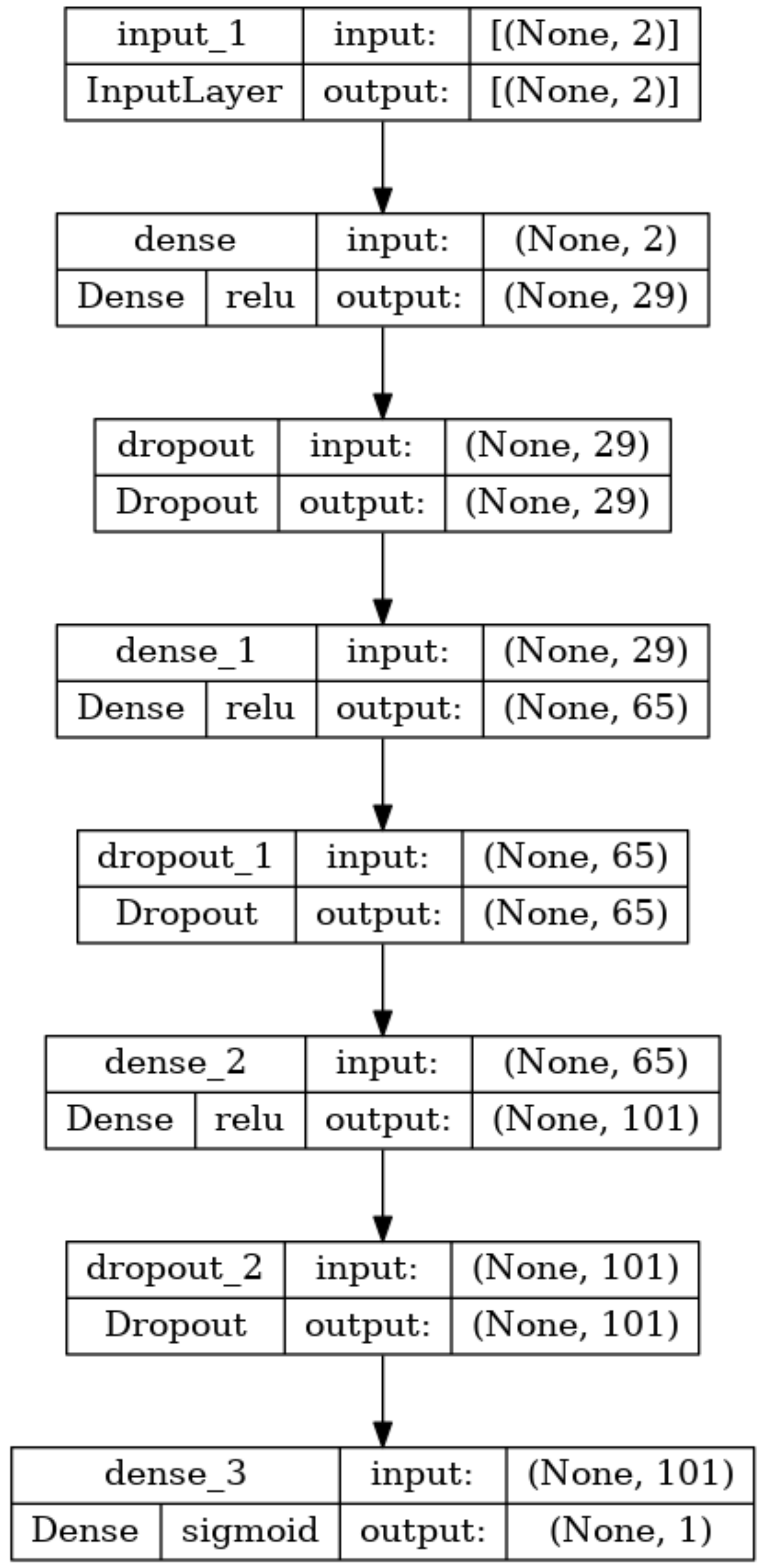
## PNN



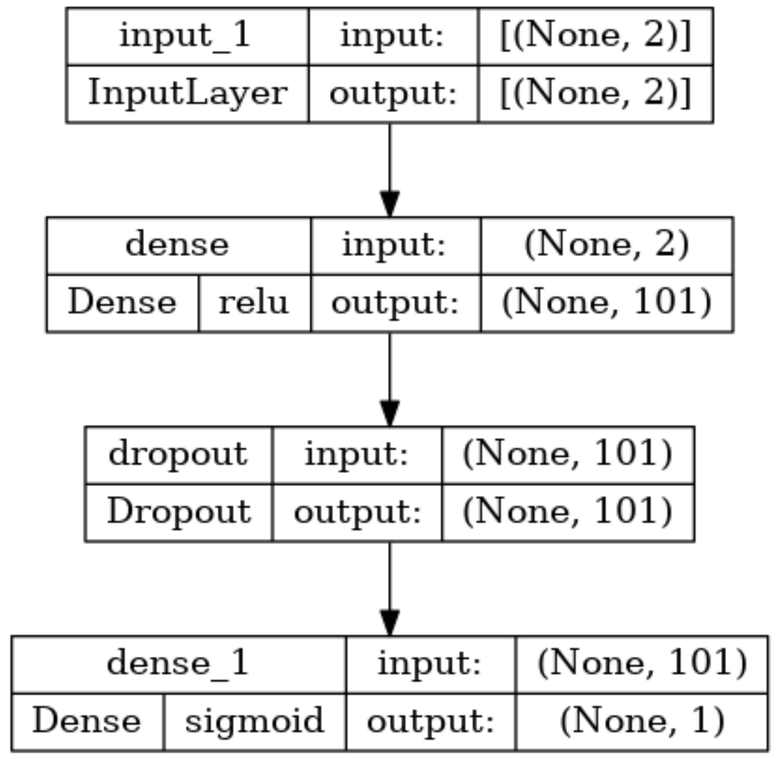

## NN X170_S30




## NN X250_S100




## NN X750_S110



# $m_X, m_S$ parameter labels in the train set