UNIVERSITY
OF OSLO

Master's thesis

Deployment of semi-unsupervised learning in the search for new physics at the LHC with the ATLAS detector

Benchmarking autoencoders for anomaly detection at the LHC with the ATLAS detector

Sakarias Garcia de Presno Frette

Computational Science: Physics
60 ECTS study points

Department of Physics
Faculty of Mathematics and Natural Sciences

28th April 2023

Sakarias Garcia de Presno Frette

Deployment of semi-unsupervised learning in the search for new physics at the LHC with the ATLAS detector

Benchmarking autoencoders for anomaly detection at the LHC with the ATLAS detector

Supervisors:
Professor Farid Ould-Saada
Dr. James Catmore

Abstract

Acknowledgments

> To be blunt, this exercise is the equivalent of fishing in the dark, with a blindfold on, with a net of an arbitrary size, in an ocean trying to not catch fish, but something else that may or may not look like fish that might or might not be in the ocean.
>
> *Sakarias Frette / Spring 2023*

This thesis has been a rollercoaster of a project, with its ups and downs, periods of yoy and of mild clinical depression. Never the less, it is finally done, my master project. It, like so much in my life could not have been done without help. First of all, thanks to the No Carbs Company for creating the most delicous and addictive energy drink, the NOCCO. They have served my lymbic system well and kept my need for caffeine somewhat at bay. I would also thank Bunnpris at Frederikkes plass for selling them for as long as you did, all though I have to pick a bone with you guys about the somewhat axiomatically wrong choice to stop selling them. On the other hand, I enjoyed your warm burgers as well, I guess that makes a bit up for it.

Throughout my education here at the Univesity of Oslo, I have had the pleasure of working with, befriending, and working out with several funny, smart, and close friends. William Hirst, Samir Noor, Anders Vestengen and Mikkel Metzch Jensen, you have all given me five excellent years with a lot of fun. Through difficult exams, long days, deadlines, heavy workouts, fun parties, UFC, Formula 1, movie nights and much much more we have made some beautiful bonds of friendship that I will sherish for ever. This is just the beginning.

To my mum and dad, Live Cecilie and Jarle Vidar. You have guided me through thick and thin with your excellent guidance and wisdom. You are what every parents should aspire to be. Loving, just, caring, strict but fair, full of insight and interests for your different offspring. I know I am not always easy to deal with, but you have done right by me for the last 25 years, and I am forever grateful. Dad, your longstanding fascination and curiosity of nature is admirable. Your ability to keep a cool head is envious, and you are very, very funny. You saw a potential within me to pursue the natural sciences. You have always been helpful and supportive with good advice. Thank you for this. Mum, your love knows no bounduries. You have children all over the world, as you should, for it is selfish of us to not share your heart with others. Where the natural sciences have been dad's department, the ways of the heart and soul have been yours. You are what I aspire to be in matters of the heart and soul. This I know for a fact. Thank you both for my upbringing and for the parents you are to me. I am forever in your debt.

To Ern. My lovely girlfriend, training partner and tickle monster. You are inspiring, with your hard work ethic, your ambition and your drive. I would not have thought in October how much time we would spend together, how much fun we have had, and how happy I would be. You keep me on my toes, you push me when I need it, and hugs me when I need it. Truly you are one of a kind. And all though I tend to make a face when you bite me, and it hurts a bit, I do actually like it ;). Sometimes. Love you boo.

Finally, to Professor Farid Ould-Saada, Dr. Eirik Gramstad and Dr. James Catmore. Thank you for an exiting and difficult proejct. It is funny to look back to the first weeks of my masters program and remenice about how little I actually knew about physics, the porocess of science research, machine learning applications and much much more. These past two years have been an education to say the least. The tools I bring with me are credited to you. Thanks for being patient with me, as I know I had a lot of questions. the weekly meetings, the questionairs and bugfixing are just some of the many moments I look back fondly on, and will bring with me. My hope is that my work will help the next masters students. If not then I hope you at least had some fun and interesting talks and discoveries with me. Until we meet again.

Contents

List of Figures

List of Tables

Introduction

> Not only is the Universe stranger than we think, it is stranger than we can think.
>
> *Werner Heisenberg*

So goes the quote by Werner Heisenberg, acclaimed for his work in quantum physics in the ealy to mid 20th century. The quote is a reminder and a statement of the fact that the world we live in is immensly strange, beautiful, complex and interesting. From the smallest atoms to the largest galaxy clusters, the universe is a place of wonder and mystery. It is perhaps easy to forget what science tries to do when studying nature. Its fundamental duty is to model the universe as well as possible given the tools available. These models gets better over time, but unlike what scientific absolutists might think and argue for, we will never know the whole truth. One should always remember Heisenbergs quote, for it illustrates the very point that nature is indeed stranger than we can think, and science, with its predictive power is only an approximation, one can never truly know if the model we have of nature is a hyperfitted model or the actual instruction manual for nature itself.

As physicists, we develop and extends, replaces and debunks models at the most fundamental level of nature, one of which is the Standard model. At higher energies, it is the most accurate, experimentally tested model to date, only rivalled by general relativity. From the 1930s to around 1973, there were huge leaps within the field of particle physics. However, besides experimentally verifying the existence of the Higgs boson, there have not been any major discoveries of the same order as before 1973. As the tools of science became better and better, more and more strange behavior was found around us, behavior that could not be explained by the Standard model. Attempts have been made to create theoretical frameworks that could include the strange behavior into the Standard Model, but all have yet to be experimentally verified. One could ask oneself why this is the case. Why is this new physics, what ever it may look like, so difficult to find? Is the framework wrong? Are the theories not well enough understood or wrong? Or could it be that the new physics are hiding within the data already collected, in some set of features, too subtle for the naked eye, but available through advanced data analysis tools? The answer is, perhaps not to anyone's shock, we have no earthly idea. In fact, they may all be true or false, to some proud physicist's dismay.

Luckily this story does not end here, as there are countless departments all over the world searching for this new physics. The ATLAS experiment at CERN is one organization that has taken upon itself this task, with my supervisors Professor Farid Ould-Saada and Dr. James Catmore working on searches for new physics at the University of Oslo. This thesis will take a somewhat different search approach than conventional analyses done at ATLAS. In a humble attempt, the assumption is made that if the new physics exists, it is too subtle for the naked eye to see, and that it is hiding in the data in some set of features. Further, it is assumed that by focusing on data we can label and that we know from experiment to exist, deviations from this would be of interest for more narrow searches. The hope is to filter out the events of interest, that with some certainty differs from the established theory, and then try to understand them better.

In the last decade, data analysis tools such as neural networks have become more and more available to the public, getting optimized and upgraded, extended and getting more computationally powerful every day. Google's Tensorflow[2] and Facebooks PyTorch[3] are two increasingly popular frameworks, with their own strengths and weaknesses that are used within industry, academia, and even within the particle physics community. Tensorflow was chosen for this thesis, and all the neural networks are using this framework, together with several other third party pieces of software. In the epigraph below, the author of this thesis used ChatGPT to write a description of what is attempted in this thesis.

> This endeavor resembles casting a line into the abyss, a murky world shrouded in darkness. With vision obscured and a net of unknown scale, the pursuit is not of fish, but of a nebulous entity that may or may not bear resemblance to its aquatic counterparts, if it exists in the briny deep at all.
>
> *ChatGPT | Spring 2023*

Outline of the Thesis

The master thesis is outlined in the following way. The first two chapters are dedicated to the neccesary machine learning and standard model physics background required to understand the analysis done and tools used in the thesis. The third chapter goes through the implementation of the project, where the datasets comes from, the ATLAS architecture, the programming libraries, feature choice, and so on. Chapter four goes through the results from the implementation as well as discussion and interpretation of the results, the pros and cons of the implementation, aspects for future improvement, and other thoughts around the process. The final chapter is dedicated to the conclusion, were the foundings are summarized.

Chapter 1

Data Analysis

1.1 Anomaly detection

Anomaly detection is a versatile tool that finds application in a diverse range of scenarios, including fraud detection, anomalous sensor data analysis, and time series data. The primary objective of this tool is to identify data that deviates from a predetermined standard of normal behavior. The definition of this standard can vary from situation to situation, based on the context and the expected anomalous behavior. According to Chandola, Banerjee, and Kumar [4], anomalies can be classified into three categories: *point anomalies*, *contextual anomalies*, and *collective anomalies*. *Point anomalies* represent singular or few outliers from a larger group or context, and can occur in various situations. A notable example of a point anomaly is Michael Phelps, who is able to swim intensively for longer periods due to his body producing less lactic acid. *Contextual anomalies*, on the other hand, are determined based on the context of the anomaly and data, rather than as a whole. For instance, in the case of continuous gas flow data, a sudden change in flow on a Saturday, despite being within the range of Friday's flow, could be categorized as a contextual anomaly. The third type, *collective anomalies*, represents a group of anomalies that deviate from the expected behavior of the dataset. In particle physics experiments, collective anomalies are of particular interest as there are many sources of anomalous behavior, and only collective anomalies are worth investigating. Additionally, the noise generated by numerous components in such experiments makes it essential to consider collective anomalies.

1.2 Neural Networks

In the field of machine learning, statistical algorithms are commonly employed for data analysis. Neural networks, a specific category of such algorithms, have experienced exponential growth in usage in both industry and academia over the past decade. Although most of the theory behind it was discovered as far back as the 80s, the technology has only now in the last 10 to 15 years become good enough. These statistical models are of extensive use in a variety of applications, ranging from image analysis to weather prediction.

The fundamental principle behind feed forward neural networks (FFNN) involves the data being fed forward through the network, with the end output evaluated and corrections then back propagated through the network to update the weights and biases. This training process is repeated until a certain threshold is met. Figure 1.1 depicts a general layout of a neural network, wherein the input layer consists of one node per feature in the dataset. The number of hidden layers and nodes per layer can be fine-tuned, with the last hidden layer connected to the output layer. The latter is determined by the problem being addressed, and in the case of the binary classification problem depicted in figure 1.1, the nodes interact via tunable weights w and biases b that must be trained on the dataset prior to making predictions. The neural network, expecially deep[1] ones becomes black boxes, meaning that once the training data or the test data is fed in through the input layer it is hard to know exactly what happens with the data until it comes out as a prediction. It is because of this that some[2] have shown some hesitency with using neural networks for problems of high importance. Nonetheless, they can be very effective if used correctly.

[1] Deep here refers to a neural network with more than one hidden layer, and often also a network where the layers have many neurons. One example would be a network with say a 200 - 500 - 500 - 700 - 500 - 200 - 1 network where the numbers correspond to number of nodes in their layers.

[2] In cases where a lot of money is at stake or lives are dependent on the output and performance of the neural network, it is reasonable to expect some explainability from the engineers if something goes wrong. There are other algorithms that are less "black box" machines, such as decision trees[5] or support vector machines[6], both of which have their pros and cons compared to neural networks.

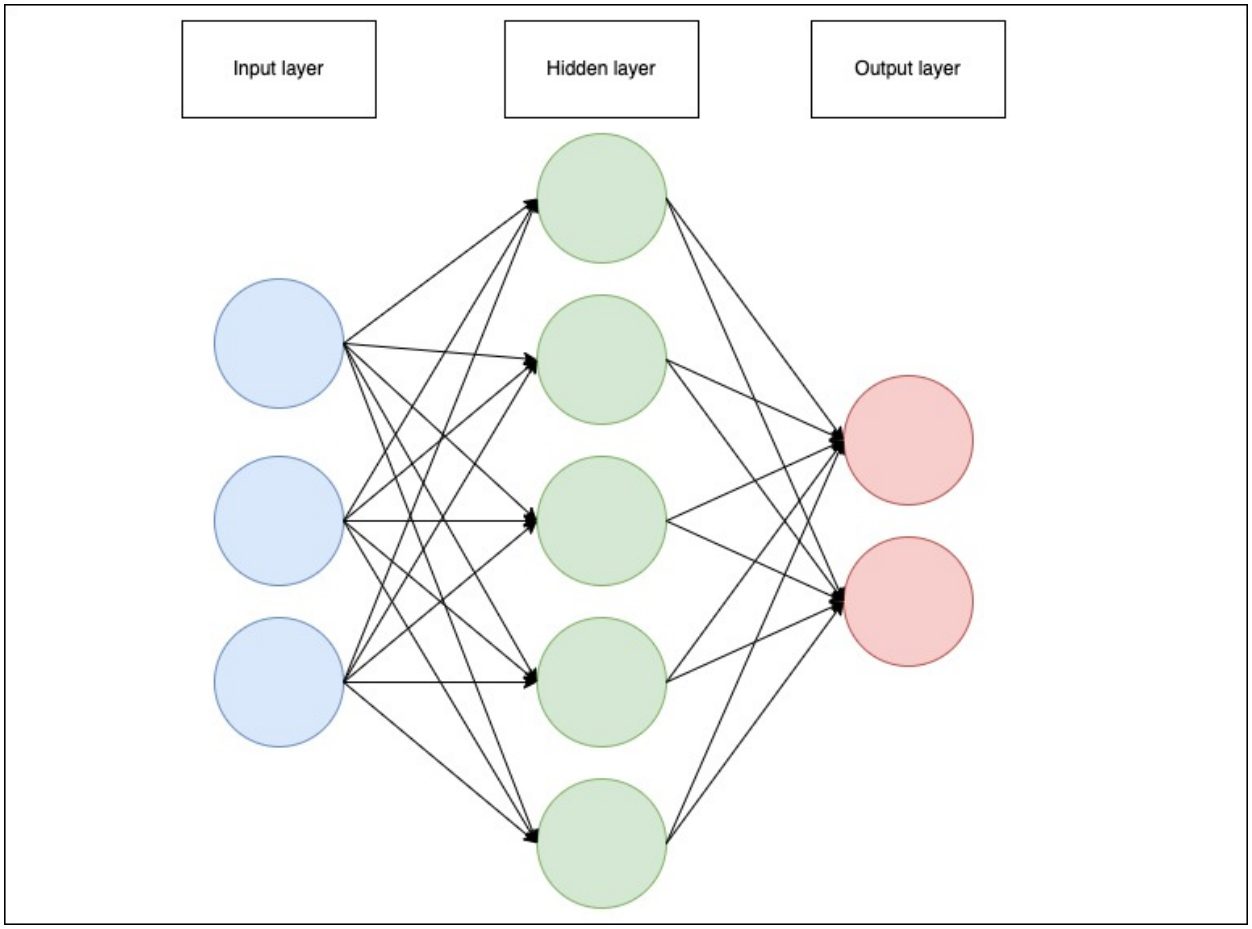


Figure 1.1: Simple neural network diagram drawm using Draw.io. Here the blue dots are the input layer, the green dots are a hidden layer, and the red dots are the output layer. The arrows show the connections between the nodes.

In order to avoid confusion, we will adhere to table 1.1 for the notation used in the following sections. Some of the following subsections contains work previously done by myself and two co-students, and can be found here[7].

Matrices and vectors		
Notation	Description	Type
X	Design Matrix (input data).	$\mathbb{R}^{N\times\#\text{features}}$
t	Target values.	$\mathbb{R}^{N\times\#\text{categories}}$
y	Model output, the prediction from our network.	$\mathbb{R}^{N\times\#\text{categories}}$
W^l	The weight matrix associated with layer l which handles the connections between layer $l-1$ and l .	$\mathbb{R}^{n_{l-1}\times n_l}$
B^l	The bias vector associated with layer l which handles the biases for all nodes in layer l.	$\mathbb{R}^{n_l\times 1}$
Elements		
w^l_{ij}	The weight connecting node i in layer $l-1$ to node j in layer l.	$\mathbb{R}$
b^l_j	Bias acting on node j in layer l.	$\mathbb{R}$
z^l_j	Node output before activation on node j on layer l.	$\mathbb{R}$
a^l_j	Activated node output on node j on layer l.	$\mathbb{R}$
Functions		
C	Cost function	
σ^l	Activation function associated with layer l.	
Quantities		
n_l	The number of nodes in layer l.	
L	Number of layers in total with $L-2$ hidden layers.	
N	Total number of data points.	
All indexing starts from 1: $i, j, k, l = 1, 2, \ldots$		

Table 1.1: Table containing notation used for deriving the mathematical formulas for the neural network fetched from previous work[7].

Gradient descent

Let us now consider a general n-dimensional problem, with parameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_n\}$. Our objective is to find the set of $\boldsymbol{\theta}$ to minimize a cost function with respect to the data and target. One way to solve this problem is using ordinary least squares. For this approach, the optimal paramters $\boldsymbol{\theta_{opt}}$ are derived from minimizing the cost function, as shown here:

$$\boldsymbol{\theta_{opt}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t},$$

where $\boldsymbol{X}$ is the design matrix containing the data, and $\boldsymbol{t}$ is the target vector. This however leads to a problem. Suppose the design matrix is sufficiently large, then the matrix inversion will get computaitonally expensive, or it might not even exist for a given $\boldsymbol{X}$. Thus, an alternative approch is to iteratively approximate the ideal parameters.

Suppose we we have a cost function $C(\boldsymbol{\theta})$ for a given problem. We can approximate the minimum of the cost function by calculating the gradient $\nabla_\theta C$ with respect to $\boldsymbol{\theta}$. The negative of this gradient indicates the direction for the minimum of C when evaluating it in a specific point $\boldsymbol{\theta}_i$ in the parameter space [7]. This is formulized as follows

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta \nabla_\theta C(\boldsymbol{\theta}_i), \tag{1.1}$$

where η is a step size, also called the learning rate. The choice of η is not a trivial case. It is one of several hyperparameters[8] that can be altered, and that highly depend on the given problem. With regard to the learning rate, there are only three situations to consider, shown in figure 1.2.

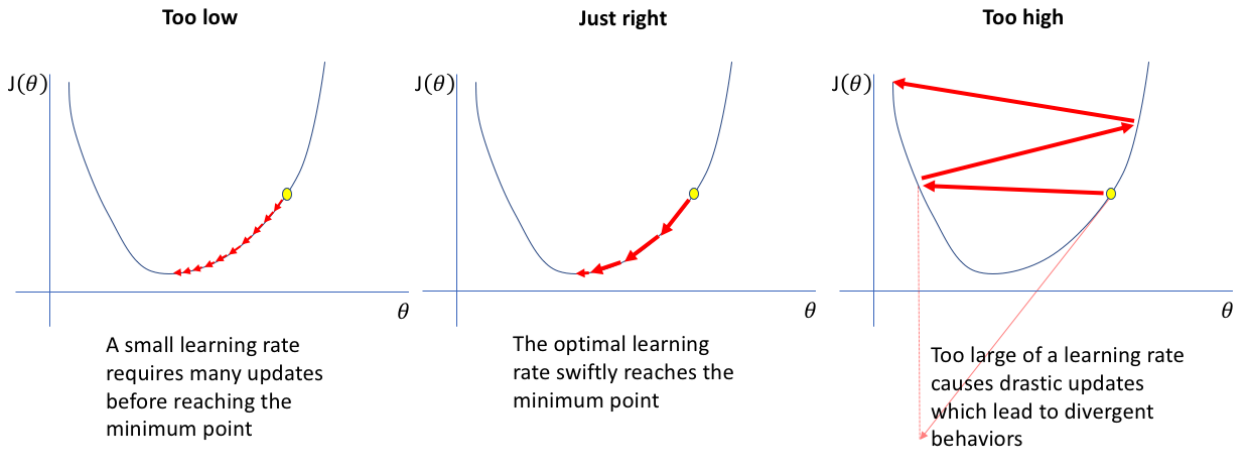


Figure 1.2: Figures showing different choices of learning rate for a given cost function, with respect to the tunable parameters. Source: Jeremy Jordan, accessed 03.10.22.

Figure 1.2 visualizes the relation between the learning rate and the cost function. In the left most figure we note that the learning rate is too small. This leads to many iterations before you reach a minimum. In the right most figure we note that the learning rate is too high, and the result is that we get divergent behavior. Thus, the goal is to find the optimal learning rate, shown in the middle figure. There are several algorithms that try to do what is shown in the middle figure above, of which is ADAM[9]

A modified and prefered version of gradient descent is the so-called stochastic gradient descent. Regular gradient descent can, for large datasets be quite slow, and is prone to getting stuck in a local minimum. To circumvent this issue, mini batches are introduced.

Feed forwarding

Inference (prediction) and training both use the same feed-forward algorithm. Let's then assume that we have generated a network. The network initializes the weights and biases usually with normal or uniformly distributed values, that can later be adjusted. The procedure is to send the data through the network, weighting each connection according to the networks architecture, and produce an output. The procedure can be summarized in the following steps[7]:

- The data is recieved by the input nodes in the network for each feature.
- Each input node weights the data value according to the connection of each node in the next layer.
- Every node in the hidden layers sums the weighted data values and adds the bias associated to the given node. The resulting number is denoted as z.
- This value z is then sent through an activation function σ, which produces the output of the node, denoted as $a = \sigma(z)$.
- This process is repeated for each hidden layer, and it is important to note that the number of nodes in the hidden layers is not dependent on the number of features in the original dataset.
- The last hidden layer then sends the activated values to the output layer, where the number of nodes and choice of activation function depends on the problem to solve.

Mathematically this is expressed as follows:

$$z_j^l = \sum_{i=1}^{n_{l-1}} w_{ij}^l a_i^{l-1} + b_j^l, \quad a_j^l = \sigma^l(z_j^l), \tag{1.2}$$

where l is the layer index, j is the node index, and i is the index of the node in the previous layer, and $l \neq 1$, as it is not used on the input layer.

Backpropagation

The way neural networks learn is conventionally by the use of the backpropagation algorithm, first proposed by Rumelhart et al[10]. This is a bit misleading, as the backpropagation algorithm actually only refers to how to compute the gradient[8]. The algorithm allows us to alter the weights and biases such thatwe get an ideal output. Assuming a cost function C, we can calculate the gradient $\nabla_{w,b}C$, and use this to back propagate the error correction. The gradient $\nabla_{w,b}C$ is comprised of two derivatives.

$$\nabla_{w,b}C = \left(\frac{\partial C}{\partial w_{i,j}^l}, \frac{\partial C}{\partial b_j^l} \right).$$

We have to use the chain rule to calculate the derivatives, and using that the last layer is $l = L$, we get the derivative with respect to the weights as

$$\frac{\partial C}{\partial w_{i,j}^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial w_{i,j}^L},$$

where

$$a_j^L = \sigma(z_j^L), \quad z_j^L = \sum_{i=1}^{n_L - 1} w_{i,j}^L a_i^{L-1} + b_j^L.$$

This then gives us

$$\frac{\partial C}{\partial w_{i,j}^L} = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L) a_i^{L-1},$$

where we defined that

$$\sigma'(z_j^L) = \frac{\partial a_j^L}{\partial z_j^L}. \tag{1.3}$$

This derivative is very easy to calculate given a specific cost function and activation function. The derivative with respect to the bias is given as follows:

$$\frac{\partial C}{\partial b_k^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \frac{\partial z_j^L}{\partial b_j^L},$$

which gives us the final expression as

$$\frac{\partial C}{\partial b_k^L} = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L).$$

We will now introduce a new notation, a local gradient commonly called the ”error”. It reflects how the rate of change of the cost function depends on the j’th node in the l’th layer.

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}.$$

Using this we get the following expression:

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L),$$

giving us the more compact forms of the derivatives with respect to the weights and biases:

$$\frac{\partial C}{\partial w_{i,j}^L} = \delta_j^L a_i^{L-1}, \quad \frac{\partial C}{\partial b_j^L} = \delta_j^L.$$

We can now let $\boldsymbol{\delta^l}$ be the vector of all the errors in the l'th layer, and $\boldsymbol{\delta^L}$ be the vector of all the errors in the last layer. The error in the l'th layer can then be expressed as a matrix equation for the last layer as follows:

$$\boldsymbol{\delta^l} = \nabla_a C \odot \frac{\partial \sigma}{\partial z^L}, \quad \nabla_a C = \left[\frac{\partial C}{\partial a_1^L}, \frac{\partial C}{\partial a_2^L}, ..., \frac{\partial C}{\partial a_{n_L}^L}\right]^T.$$

Here $\odot$ is the Hadamard product (element wise product). This local gradient can now be defined recursively for the j'th node in a layer l as a function of the local error in the next layer:

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1}. \tag{1.4}$$

We also note that

$$z_k^{l+1} = \sum_{j=1}^{n_l} w_{j,k}^{l+1} a_j^l + b_k^{l+1} = \sum_{j=1}^{n_l} w_{j,k}^{l+1} \sigma(z_j^l) + b_k^{l+1},$$

thus the partial derivative is given as

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{j,k}^{l+1} \sigma'(z_j^l), \tag{1.5}$$

using the substitution from equation 1.3. This allows us to substitute equation 1.5 into equation 1.4 to get the following expression:

$$\delta_j^l = \sum_k w_{j,k}^{l+1} \sigma'(z_j^l) \delta_k^{l+1}. \tag{1.6}$$

Using this we can derive a three-step formula for the backpropagation algorithm:

- Compute the local gradient for the last layer, δ^L.
- Recursively compute the local gradient for the remaining layers, δ^l for $l = L-1, L-2, ..., 1$.
- Update the weights and biases for all layers, $l = 1, 2, ..., L$, given the learning rate η as shown below:

$$w_{i,j}^l \leftarrow w_{i,j}^l - \eta \delta_j^l a_i^{l-1},$$

$$b_j^l \leftarrow b_j^l - \eta \delta_j^l.$$

1.3 Autoencoders

Autoencoders are a subset of neural networks. Whereas a general neural network in principle can take any shape, autoencoders are more restrictive. This restrictiveness can in its most general sence we condensed into the following points:

- Same number of output categories as input categories
- A latent space with smaller dimensionality than the input/output layer

What we end up with are two funnel shaped parts linked together. The two funnels are called the encoder (left funnel) and decoder (right funnel) respectively. This architecture is not accidental, but rather designed with a very specific solution of problems in mind, namely reconstruction. A good example to illustrate this is image reconstruction, illustrated in figure 1.3. Suppose you have an image, and want to reconstruct it. By feeding the encoder an image, and comparing the decoder output to the actual image, the autoencoder can tune itself to recreate the images it trains on.

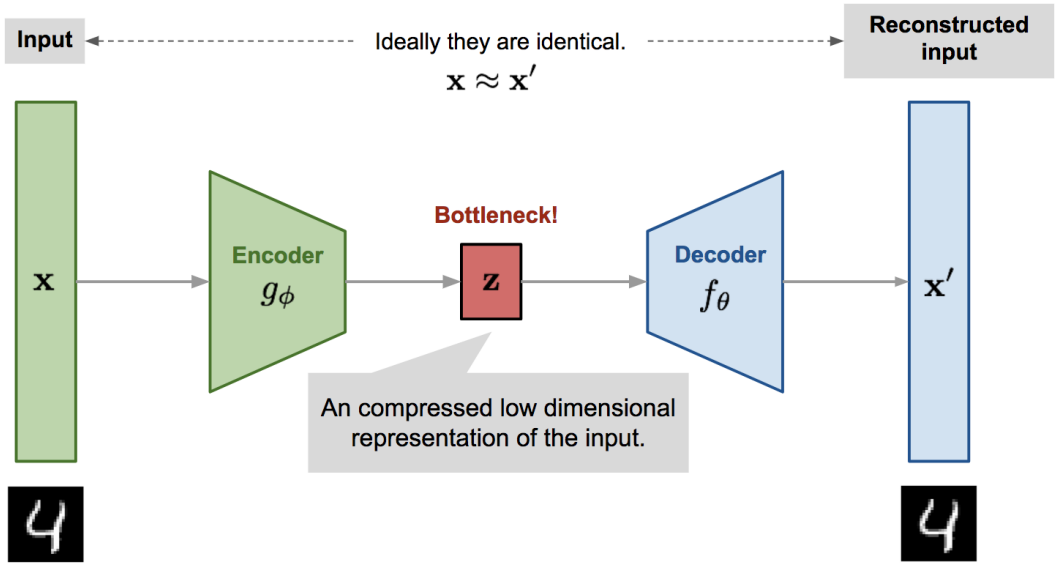


Figure 1.3: Figure depicting a model for an autoencoder. Here the input $\mathbf{x}$ is the original image, $\mathbf{x}'$ is a reconstructed version of $\mathbf{x}$, g_ϕ is the encoder, f_θ is the decoder, and z is the latent space. Found 14.01.23 here [1].

Mathematically this is is represented as follows[1]. Using the annotations of each component in figure 1.3 the decoded information is defined as follows

$$\mathbf{z} = \mathbf{g}_\phi(\mathbf{x}),$$

and the reconstruction given as

$$\mathbf{x}' = \mathbf{f}_\theta(\mathbf{g}_\phi(\mathbf{x})).$$

The parameters (ϕ, θ) are the tuneable parameters adjusted according to the loss function. In our case, the goal is reconstruction without copying, thus we can use mean squared error, given as

$$L_{AE}(\phi, \theta) = \frac{1}{N} \sum_{i=0}^{N-1} \left(\mathbf{x^i} - \mathbf{f}_\theta(\mathbf{g}_\phi(\mathbf{x^i}))\right)^2. \tag{1.7}$$

1.4 Variational autoencoders

Another popular method for reconstruction is the so-called variational autoencoder. The work by Kingman and Welling [11] showed how one can use the variational bayesian approach for efficient approximate posterior inference, leading to the use of variational autoencoders. Here, contrary to regular autoencoders, the latent space is a distribution that can be sampled from. In the context of reconstruction, this means that we want to create a latent space distribution based on the true distribution in the data, and use this latent space distribution to then generate data given that latent space.

Variational Bayes

Let's assume some dataset $X = \{x^{(i)}\}_{i=1}^N$ with N samples for a variable x, and also assume that the generation of the data comes from some random variable z. We also assume that this variable z is generated from a prior distribution $p_\theta(z)$ and that the data is then generated from a conditional distribution $p_\theta(x|z)$, and that both their respective probability density functions are sufficiently differentiable with respect to parameters θ and z. We then have that the true posterior distribution is given by $p_\theta(z|x) = p_\theta(z)p_\theta(x|z)/p_\theta(x)$. The work done by Kingman and Welling [11] was motivated by the fact that $p_\theta(z|x)$ more often than not is intractable[3], thus instead we will try to approximate the true posterior with a variational approximation $p_\theta(z|x) \approx q_\phi(z|x)$. It can from here on out be useful to think of $q_\phi(z|x)$ as a probabilistic encoder, as it for a given data point x will produce a distribution over possible values for z, the latent space, that the datapoint could have been generated from. Similarly, is can be useful to think of the $p_\theta(z|x)$ as a probabilistic decoder, as it for a given z will produce a distribution over possible values for x.

[3]Intractability here refers to the inability to evaluate or differentiate a given distribution or function, or difficulty with solving a problem due to a large parameter space or finding the global optimum of a complex function.

The variational bound

It can be shown that the marginal likelyhood can be written as follows:

$$log\, p_\theta(x^{(i)}) = D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z|x^{(i)})) + \mathcal{L}(\theta, \phi; x^{(i)}), \tag{1.8}$$

where the first term on the right-hand side is the Kullback-Leibler (KL) divergence[4] and the second term is the evidence lower bound (ELBO) on the marginal likelyhood. As the KL divergence is non-negative, the ELBO will always be equal to or less than the marginal likelihood, written below:

$$log\, p_\theta(x^{(i)}) \geq \mathcal{L}(\theta, \phi; x^{(i)}) = \mathbb{E}_{q_\phi(z|x)}[-log\, q_\phi(x|z) + log\, p_\theta(x|z)]. \tag{1.9}$$

Rewriting we get that the ELBO is given as:

$$\mathcal{L}(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z|x^{(i)})) + \mathbb{E}_{q_\phi(z|x^{(i)})}[log\, p_\theta(x^{(i)}|z)], \tag{1.10}$$

which is also known as the loss function for variational inference. As a loss function, this needs to be differentiated with respect to (ϕ, θ), but as shown by Kingman and Welling [11], the common method using Monte Carlo gradient estimator have high variance, and thus is not ideal.

The Stochastic gradient variational bayes (SGVB) estimator

Kingman and Welling [11] instead proposes a practical estimator that both deals effectivily with large datasets, aswell as the issue of intractability. This estimator is called the stochastic gradient variational bayes (SGVB) estimator, and first assumes an approximate posterior $q_\phi(z|x)$. Under certain conditions, as shown in subsection 1.4, we can reparameterize this approximate posterior to a random variable $\tilde{z} \sim q_\theta(z|x)$, by doing a differentiable transformation $g_\phi(\epsilon|x)$, such that

$$\tilde{z} = g_\phi(\epsilon|x), \quad \epsilon \sim p(\epsilon),$$

where ϵ is a noise variable.

Using Monte Carlo estimates of expectations of a function $f(z)$ with respect to $q_\theta(z|x)$, Kingman and Welling [11] shows that:

$$\mathbb{E}_{q_\theta(z|x^{(i)})}[f(z)] = \mathbb{E}_{p(\epsilon)}[f(g_\phi(\epsilon, x^{(i)}))] \approx \frac{1}{L}\sum_{l=1}^{L} f(f(g_\phi(\epsilon^{(l)}, x^{(i)}))), \tag{1.11}$$

where $\epsilon^{(l)} \sim p(\epsilon)$. Using this technique, and assuming that the KL divergence $D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z|x^{(i)}))$ can be integrated analytically, we only get one term in equation 1.10 that requires estimation by sampling: $\mathbb{E}_{q_\phi(z|x^{(i)})}[log\, p_\theta(x^{(i)}|z)]$. What we end up with is a version of the SGVB estimator: $\tilde{\mathcal{L}}^B(\theta, \phi; x^{(i)}) \backsimeq \mathcal{L}(\theta, \phi; x^{(i)})$. This is given as:

$$\mathcal{L}(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L}(\log p_\theta(x^{(i)}|z^{(i,l)})), \tag{1.12}$$

where $z^{(i,l)} = g_\phi(\epsilon^{(i,l)}, x^{(i)})$ and $\epsilon^{(i)} \sim p(\epsilon)$. What we now have is a loss function that contains two terms. The first term, the KL divergence, acts as a regularizer, whereas the other term acts as a negative reconstruction error. In fact, we have here two objectives, we want to minimize the ELBO, $\mathcal{L}(\theta, \phi; x^{(i)})$, by minimizing the KL divergence and maximizing the expected log-likelihood.

Reparameterization

If the latent space is assumed to be a continous variable sampled from a conditional continous distribution $z \sim q_\phi(z|x)$, then we can reparametrize z as a deterministic variable. This is very useful as we then can rewrite the expectation of the conditional distribution such that the Monte Carlo estimate of the expectation is differentiable with respect to the parameters ϕ. The proof can be found in the article[11]. Now let's take the example of the univariate Gaussian distribution. First, let $z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$. Then, a reparametrization of z can be $z = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. Thus

$$\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}[f(z)] \backsimeq \frac{1}{L}\sum_{l=1}^{L} f(\mu + \sigma\epsilon^{(l)}).$$

[4]The KL divergence is a measure of how one probability distribution is different from a second, reference probability distribution. It is often used as a distance measure between two probability distributions.

Variational autoencoders

Now, this variational bayes can then be used to create a variational autoencoder. This contains two neural networks, the generative model $p_\theta(x|z)$, as well as the probabilistic encoder $q_\phi(z|x)$, which is used to approximate the posterior $p_\theta(z|x)$. We now let the prior over the latent space be a multivariate Gaussian, lacking parameters: $p_\theta(z) = \mathcal{N}(z; 0, I)$. We also have that $p_\theta(x|z)$ is a multivariate Gaussian, with a distribution generated from z, whilst $p_\theta(z|x)$ is in fact intractable. If we now assume that the true (intractable) posterior is a Gaussian with approximately diagonal covariance, we can let the variational approximate posterior be a multivatiate Gaussian:

$$log\, q_\theta(z|x^{(i)}) = log\, \mathcal{N}(z; \mu^{(i)}, \sigma^{2(i)} I), \tag{1.13}$$

where the mean and standard deviation are given by the neural network. Now, we sample from the approximate posterior $z^{(i,l)} \sim q_\theta(z|x^{(i)})$ where $z^{(i,l)} = g_\phi(x^{(i)}, \epsilon^{(l)}) = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)}$, $\epsilon^{(l)} \sim (N)(0, I)$ and $\odot$ is the elementwise product. If both the prior $p_\theta(z)$ and $q_\theta(z|x)$ are Gaussian, the KL divergence can be computed analytically, and it can then be showed[11] that the variational approximate posterior is:

$$\mathcal{L}(\theta, \phi; x^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^{J} (1 + log\, ((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2) + \frac{1}{L} \sum_{l=1}^{L} log\, p_\theta(x^{(i)}|z^{(i,l)}), \tag{1.14}$$

where $z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \epsilon^{(l)}$ and $\epsilon^{(l)} \sim \mathcal{N}(0, I)$.

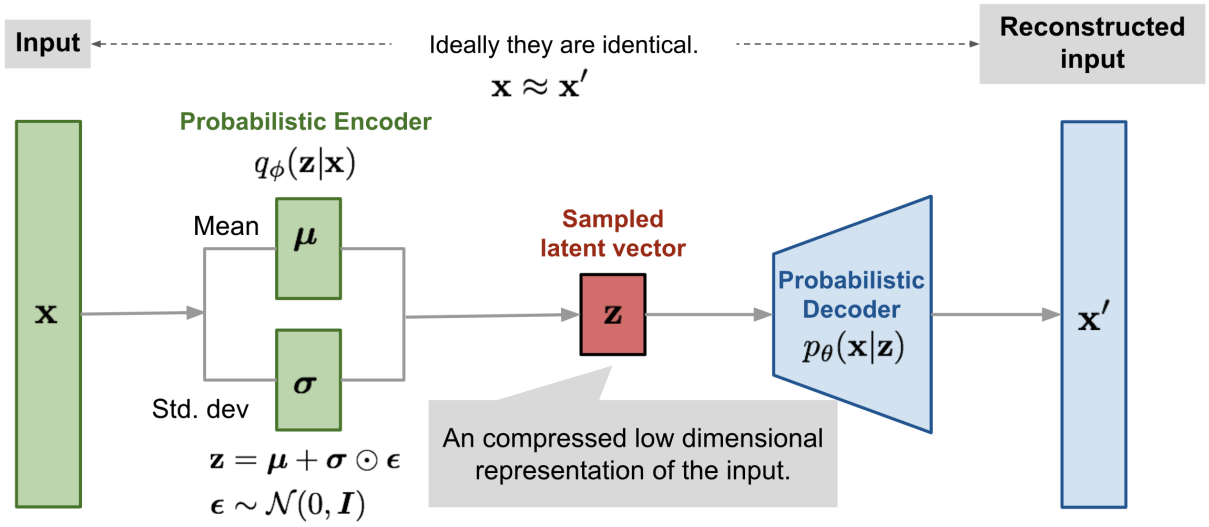


Figure 1.4: Figure depicting a model for a variational autoencoder. Found 14.01.23 here [1].

Figure 1.4 is a graphical representation of the variational autoencoder. The encoder is a neural network, which takes the input $\mathbf{x}$ and maps the input to a latent space by creating a Gaussian distribution. The decoder is a neural network, which takes the latent space $\mathbf{z}$ and outputs the parameters of a Gaussian distribution for the data. The loss function is given by equation 1.14.

1.5 Other tools and algorithms

Adaptable moment estimation (ADAM) Optimizer

Stochastic gradient descent, though very useful, lack the ability to adapt to the feature space. One algorithm that address this issue is the ADAM optimizer[9]. The Adaptable moment estimation (ADAM) algorithm uses stochastic gradient descent, but with an adaptiv learning rate. This learning rate is adjusted by calculating estimates for the first and second moment[5]. Thus, a large gradient would indicate close proximity to a minimum in feature space, thus a lower learning rate would yield a more accurate result, where as a small gradient would suggest far proximity to a local minimum, and thus a larger learning rate would increase the chance of approaching a minimum.

[5]In statistics the first moment is the expectation value for a distribution, $E[X - \mu]$. The second moment is the expectation value of the distribution squared, i.e the variance, $E[(X - \mu)^2]$

Hyperband

Hyperband is a tool for hyperparameter optimization[12]. Hyperparameter optimization is of high importance in the search for ideal structures and architectures when using neural networks, as there is not a way to find an a priori setup for a given problem. Several algorithms are used, from random search, grid search, and bayesian optimization. Hyperband is an algorithm proposed by L. Li et al. It focuses on using successful halving[13] but at the same time doing a grid search for how to allocate resources. Successive halving focuses on testing n configurations, and removing the bottom half, thus (hopefully) quickly converging to the ideal combination. However, it is not easy to a priori know the number of configurations n, and how much resources one needs, r, to quickly find the ideal set. This is where Hyperband comes in. In essence, it fetches and tries different combinations of r resources (time, data set subsampling or feature subsampling) and n configurations, to determine the ideal set of hyperparameters via successive halving, yielding 5x to 30x speedup compared to Bayesian optimization. One drawback for this algorithm is that one cannot guarantiy that the configuration is optimal, but rather that it is good enough.

Activation functions

Several activation functions are used in neural networks, and how one chooses the best combination for a given problem is not trivial. This leads often to the use of tuning. Below are a list of the functions used in this thesis, aswell as their mathematical definitions.

1. $sigmoid(x) = \frac{1}{1+e^{-x}}$
2. $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
3. $ReLU(x) = \max(0, x)$
4. $LeakyReLU(x) = \max(\alpha x, x)$
5. $Softmax(x) = \frac{e^x}{\sum_{i=1}^{n} e^x}$
6. $Linear(x) = x$

ROC curve

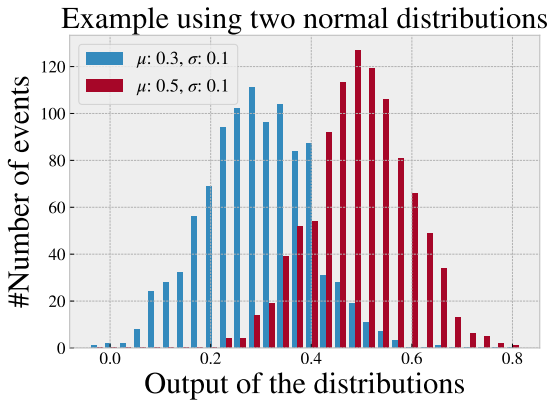
One way to measure the effectiveness of a prediction for a machine learning algorithm is with the use of a ROC curve. The ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) which will be defined shortly. Suppose a classifier $\mathcal{M}$ does a binary classification of positive and negative. If $\mathcal{M}$ predicts a positive label, and the instance is indeed positive, we define this as a *true positive*. The same goes for negative prediction of a negative instance, which is defined as a *true negative*. Then there is the case where $\mathcal{M}$ predicts a negative label when the instance is positive, defined as a *false negative*, and vice versa is then defined as a *false positive*[14]. Now, the TPR is defined as the ratio of true positives to the total number of positive instances, i.e.

$$TPR = \frac{TP}{TP + FN},$$

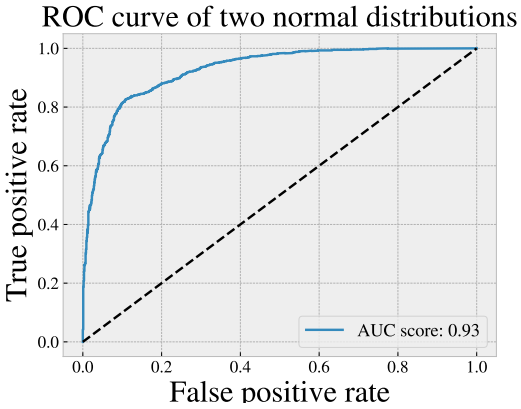
and the FPR is defined as the ratio of false positives to the total number of negative instances, i.e.

$$FPR = \frac{FP}{FP + TN}.$$

Using this, we get a plot that tells us how well the classifier performs. As the goal is usually to have a high TPR and a low FPR, the more ”north-west”[14] of the plot the curve is, the better. Below are two example of a ROC curve and the classifier output it is calculated from. For simplicity, the classifier output is in this case just two normal distributions with the mean and standard deviation listed in the legend of the distribution plots.

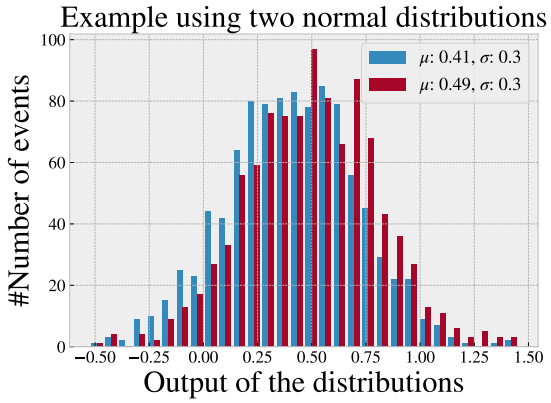


(a) Histogram showing good separation. Here the left distribution is calculated using a mean of 0.3 and a standard deviation of 0.1, and the right distribution is calculated using a mean of 0.5 and a standard deviation of 0.1.

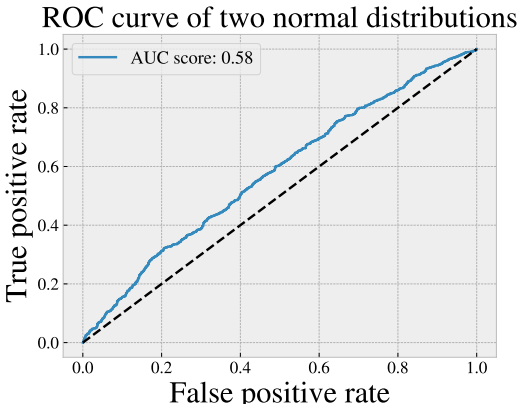


(b) ROC curve based on the two distributions to the left. Here we get an area under the curve score of 0.92, which is very good.

Figure 1.5: Histogram and ROC curve for two well separated distributions.



(a) Histogram showing bad separation. Here the left distribution is calculated using a mean of 0.41 and a standard deviation of 0.3, and the right distribution is calculated using a mean of 0.49 and a standard deviation of 0.3.



(b) ROC curve based on the two distributions to the left. Here we get an area under the curve score of 0.58, which is not a good classification score.

Figure 1.6: Histogram and ROC curve for two poorly separated distributions.

In figure 1.5 we have a histogram showing two distributions that are well separated, and the ROC curve that is calculated based on thos distributions. If we compare this to figure 1.6, we see that the better separation of the two distributions, the better ROC curve. This is a useful tool to use with the autoencoder, as it provides valueable insight beyond just looking at the output distributions.

Statistical significance

In the frequentist statistics, the Poisson distribution can be approximated with a Gaussian distribution in the limit of large number of events[15]. The expression for the significance is given then as

$$Z = \frac{s}{\sqrt{b}}, \tag{1.15}$$

where s is the amount of signal, and b is the amount of background. For low statistics we have that the significance is given as

$$Z = \sqrt{2\left[(s+b)\ln\left(1+\frac{s}{b}\right) - s\right]}. \tag{1.16}$$

It can be shown that in the limit where $s << b$, the two expressions are approimately the same[15].

Chapter 2

The Standard model and BSM physics

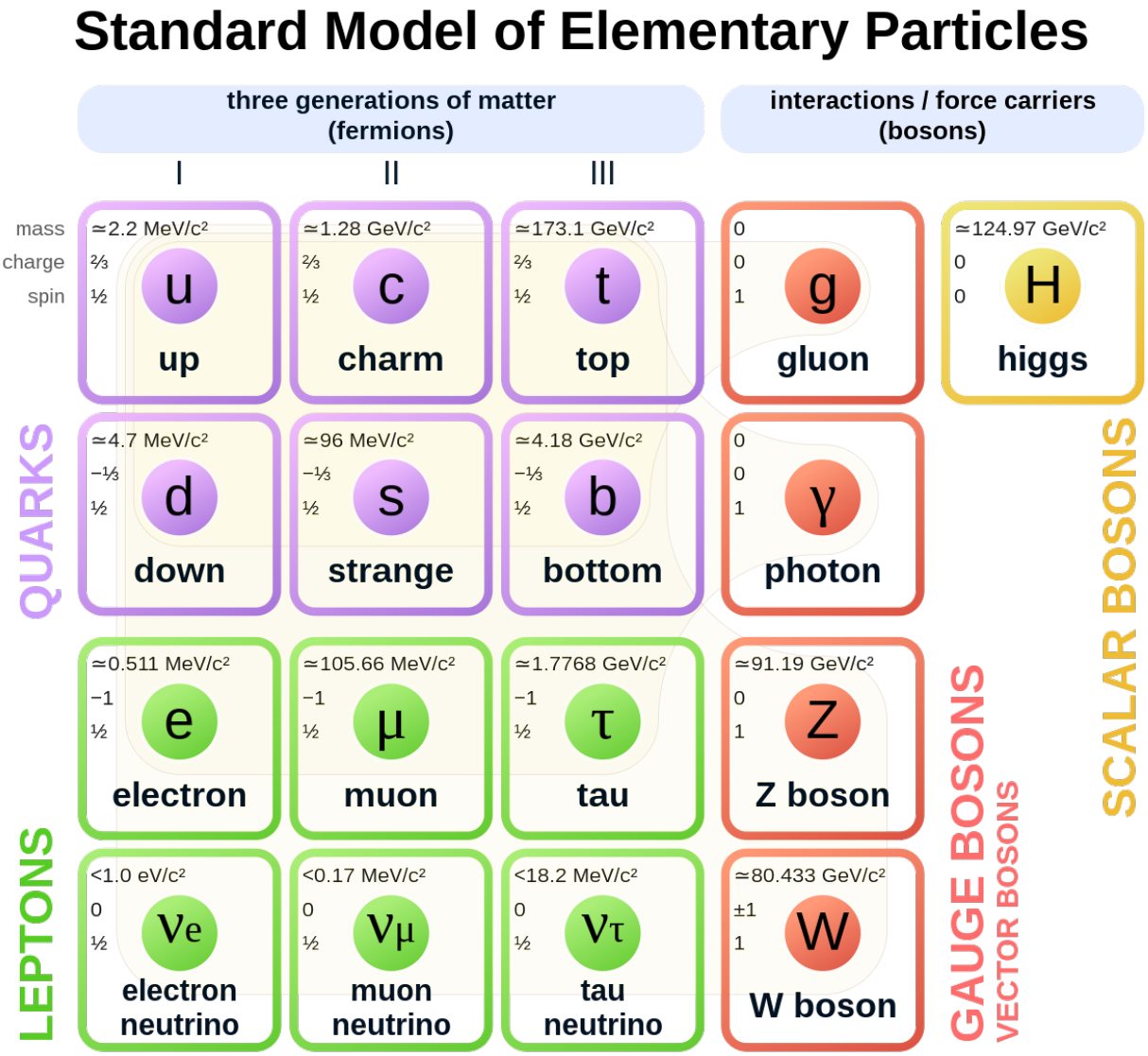


Figure 2.1: The standard model of elementary particles. Source here. Accessed 07.10.22

2.1 Structure and composition of the Standard Model

This section will describe the standard model in a phenomelogical way, as the mathematics and physical reasoning behind the theory is not of great importance to understand the work presented here, nor the results or discussion. For a more technical explanation, read (Pich, 2008)[16] for a well written paper containing some more standard model fundamentals, as well as summarizing the experimental status regarding the standard model. For more mathematical understanding of the standard model, Peskin and Schroeder's "An introduction to Quantum Field theory" (Peskin and Schroeder, 1995)[17] is highly recommended. Finally, see Thomson's "Modern Particle physics" (Thomson, 2013)[18] for a very comprehensive and up-to-date book that is easy to read and understand.

The standard model is to physicists what the periodic table is to chemists, and is to this day the most fundamental description of matter as we know it at the subatomic scale. It comprises two parent class particles, fermions and bosons, where fermions are comprised of quarks and leptons. The model contains 6 leptons, 6 quarks coming in 3 colors each, 4 gauge bosons mediating the electroweak interactions ($\gamma, W^{\pm}, Z^0$), 8 gluons behind the strong interaction and one scalar Higgs explaining particle masses, all of which are shown in figure 2.1.

Fermions

The fermions are the building blocks of matter, and contain two types of particles, leptons and quarks. The up and down quarks form protons and neutrons, which together with electrons forms the atoms. Fermions, unlike bosons, are spin half particles. The fermions are grouped into three so-called families:

$$\begin{bmatrix} \nu_e & u \\ e^- & d' \end{bmatrix}, \quad \begin{bmatrix} \nu_\mu & c \\ \mu^- & s' \end{bmatrix}, \quad \begin{bmatrix} \nu_\tau & t \\ \tau^- & b' \end{bmatrix}$$

Note that the left column contains the leptons. Whilst the right column contains the quarks. Within the left column, the subscripted ν denotes what kind of neutrino that corresponds to the given lepton. Here, the first family consists of the electron, the electron neutrino, the up and down quarks. The second family consists of the muon and the muon neutrino, the charm and strange quarks. The third family consists of the tau and the tau neutrino, the top and bottom quarks. The masses of these particles increases for each particle in the matrix as the family number increases, i.e. the muon is heavier than the electron, and the tau is heavier than the muon, and so on for the other charged fermions. It is not known wether or not the neutrinos follow this pattern as well due to their very low mass. Below is a table with specific properties of the fermions.

Table 2.1: Table showing properties of all the fermions, including name, symbol, antiparticle, spin, charge, generation and mass.

Generation	Name	Symbol	Antiparticle	Spin	Charge	Mass (MeV/c^2)
Quarks						
1	up	u	$\bar{u}$	1/2	2/3	$2.2^{+0.6}_{-0.4}$
	down	d	$\bar{d}$	1/2	$-1/3$	$4.6^{+0.5}_{-0.4}$
2	charm	c	$\bar{c}$	1/2	2/3	1280 ± 30
	strange	s	$\bar{s}$	1/2	$-1/3$	96^{+8}_{-4}
3	top	t	$\bar{t}$	1/2	2/3	172100 ± 600
	bottom	b	$\bar{b}$	1/2	$-1/3$	4180^{+40}_{-30}
Leptons						
1	electron	e^-	$\bar{e}^-$	1/2	-1	0.511
	electron neutrino	ν_e	$\bar{\nu}_e$	1/2	0	< 0.0000022
2	muon	μ^-	$\bar{\mu}^-$	1/2	-1	105.7
	muon neutrino	ν_μ	$\bar{\nu}_\mu$	1/2	0	< 0.170
3	tau	τ	$\bar{\tau}$	1/2	-1	1776.86 ± 0.12
	tau neutrino	ν_τ	$\bar{\tau}_\tau$	1/2	0	< 15.5

Another mystery regarding the neutrinos is in relation to anti particles. To each particle corresponds an antiparticle, i.e. for $l^- \to l^+$, $q \to \bar{q}$, where the anti particle and particle are different. With neutrinos however it is not known wether or not the neutrinos and antineutrinos are the same particle $\nu = \bar{\nu}$ (Majorana neutrino), or if they are different $\nu \neq \bar{\nu}$ (Dirac neutrino).

Quarks are fractional charge particles, with defined charge of either 2/3 or $-1/3$, as shown in table 2.1. They are the "main" building blocks of protons and neutrons, and are bound by the strong force, the

strongest of the four fundamental forces. The force mediator is the gluon. The other half of fermions are the leptons. They are split into the charged leptons (electrons, muons and taus), and the uncharged leptons (neutrinos). The charged leptons can interact via the electroweak force, where the Z, W bosons as well as the photon can be a mediator. Note that the neutrinos can only interact through the weak force.

Bosons

Bosons are integer number spin particles, with spin $0, 1, 2, ...$ Within bosons there are so-called elementary bosons, some of which are force carriers or mediators such as the $W^{\pm}$, Z and the photon. The Higgs boson is also an elementary scalar boson, but is not a force carrier. It provides masses for the fermions via a process called spontaneous symmetry breaking[16]. Other bosons are so-called composite bosons, mesons ($q\bar{q}$) and baryons (qqq) which are particles constructed by even or half integer spin. Bosons also have antiparticles, where (γ, g, H^0, Z^0) are equal to their respective antiparticle, and the $W^- \rightarrow W^+$ are not equal.

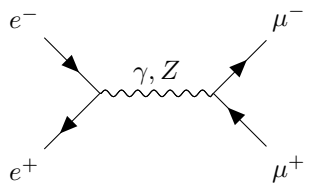
Left and right handedness

Particles in the standard model are subject to a quantum mechanical property called chirality. Chirality is a property that describes a particle's ability to be superimposed on its mirror image. If a particle has chirality, it cannot be superimposed on its mirror image by any combination of translation, rotation, and reflection operations. [19]

Feynman diagrams

A graphical way to understand particle interactions are through so-called Feynman diagrams. Feynman diagrams are drawn based on the Feynman rules for a given Lagrangian[16][20], and each component can be linked to a part in the Lagrangian for the system.

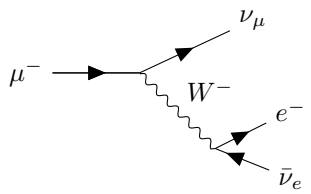
Figure 2.2: Feynman diagram of muon pair production from electron scattering. Here, both Z and γ can work as the propagator.



In figure 2.2 we have a Feynman diagram describing electron-positron annihalition into muon-antimuon pairs. In this thesis, all diagrams will be interpreted from left to right, i.e. figure 2.2. The diagram contains all the components in the Lagrangian, and arrows, curly lines and so on all have its own meaning. A straight line with an arrow usually means a fermion, where the direction of the arrow tells if the particle (arrow towards the vertex) is an anti particle (arrow away from the vertex). There is often also a propagator between the left and right side of the Feynman diagram, and they depend on the processes we want to study. In the diagram above we have lepton scattering, thus we can both have the photon and the Z-boson as a propagator. This process is called a neutral current[16], as the total charge coming out of the interaction is 0. As with neutral currents, we also have so-called charged currents, where the sum of charge is not 0. Note that we only require charge conservation, thus there is nothing wrong with either having a neutral or a charged current, as long as charge conservation is preserved.

Feynman diagrams is not only used for visualizing scatterings, they can also visualize decays. An example is provided in figure 2.3.

Figure 2.3: Muon decay into an electron, an electron neutrino and a muon neutrino via the W^- boson. Read the graph from left to right.



Here we have a decay of a muon into and electron and two neutrinos, through a charged current.

The examples above in figure 2.2 and 2.3 show interactions with the electroweak force, but along with the electroweak interactions, are also the quantum chromodynamics (QCD), responsible for interactions between quarks and gluons. A strange property of QCD, is that the coupling constant α_S, unlike the α_{EM} for electromagnetism, gets stronger as the energy decreases. This is because QCD (and weak interactions) are based on non-abelian groups[17], thus to study such interactions, one needs to create collisions at very high energies.

Proton-proton collisions

Proton-proton collisions require high energy to collide. During an interaction, both the quarks and gluons in the protons colliding can interact in proton-proton collisions, as shown below:

Figure 2.4: Proton-proton collision with lepton pair production via the Z boson or photon. Read from left to right.

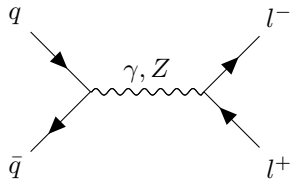
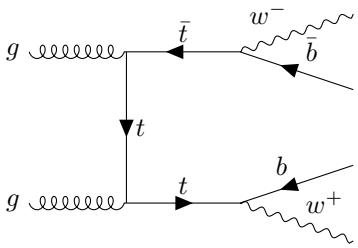


Figure 2.5: Proton-proton collision showing the $t\bar{t}$ channel. Read from left to right



In figure 2.4 and figure 2.5 we have two examples of Feynman diagrams for possible interactions in proton-proton collisions. The first figure displays a lepton pair production via the Z boson or photon from quark-antiquark annihilation, and the second figure displays the $t\bar{t}$ production via gluon-gluon fusion.

Some limitations

Although the standard model has had great success comparing with experiments, there are still several problems not addressed by it. One example is gravity, the standard model as described above, does not and cannot incorporate gravity in a quantized way. There are models that try, without success so far, to address this problem, but they supplement the standard model, and does not derrive it from it. Another problem with the standard model is a curious property of the weak interaction, namely that parity is broken. Parity as a mathematical operation is equivalent to the spatial inversion through the origin[18]:

$$x \to -x. \tag{2.1}$$

In other words, parity can be thought of as left-right symmetry, or mirror symmetry. Breaking of parity is observed in weak currents, where the mediator of the charged currents, $W^{\pm}$ only interacts with left-handed fermions and right-handed antifermions. In the standard model, neutrinos are assumed to be massless, and the righthanded neutrinos are sterile, i.e. they do not interact in the standard model.

This asymmetry is strange, and hints towards new physics that perhaps can restore the parity breaking at much higher energies. Another note to make is that it has been experimentally verified that the neutrinos are massive[21], with an upper limit on the mass for the anti electron neutrino of $m_\nu < 0.8eVc^2$ at 90% confidence level. This is somewhat problematic, as the tiny masses of the neutrinos are not predicted by the standard model.

2.2 BSM model physics

Heavy neutrinos

Parity symmetry is suggested to be restored at high energies by the introduction of a right-handed weak symmetry, leading to right-handed weak charged bosons, $W_R^{'\pm}$. These mediators, as with the rest, decay faster than the detection ability at the LHC[6], thus evidence of such a boson would come from the detection of the decay of a heavy neutrino. Heavy neutrinos can be produced in proton-proton collisions through either the right-handed $W_R^{'\pm}$ bosons or the SM $W^{\pm}$ bosons, under given conditions.

Figure 2.6: Proton-proton collision with heavy neutrino production via SM $W^{\pm}$ boson into 3 lepton final state. Read the graph from left to right.

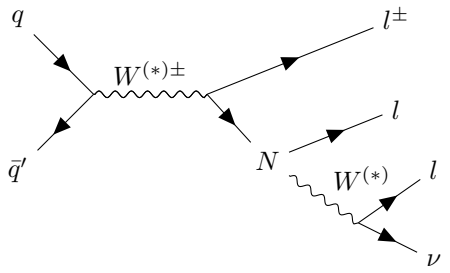
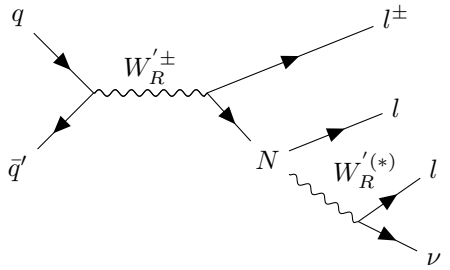


Figure 2.7: Proton-proton collision with heavy neutrino production via right-handed $W_R^{\pm}$ boson into 3 lepton final state. Read the graph from left to right.



Note that the right most W and W_R bosons have an asterix as a superfix. This is because the bosons can be virtual. This means that if the mass of the heavy neutrino is less than the W boson, it cannot produce it, but it can produce a virtual one such that the decay still happens.

Supersymmetry

Supersymmetry is another BSM theory that attempts to solve two other problems that the standard model has. First, the hierarchy problem. As the standard model is a perturbative theory, the Higgs mass increases at higher energies. The problem is that when you approach higher and higher energies, this mass goes to infinity, which is not physical. Supersymmetry solves this problem. The theory introduces a supersymmetric partner to each particle in the standard model. The result is that the contributions to the Higgs mass from fermions and bosons mainly cancel, thus fixing the hierarchy problem. Another problem we have with the standard model is that is does not have a field for dark matter. Some supersymmetry models have a dark matter candidate, thus the Supersymmetry search is quite large at CERN.

[6]Large Hadron Collider at CERN

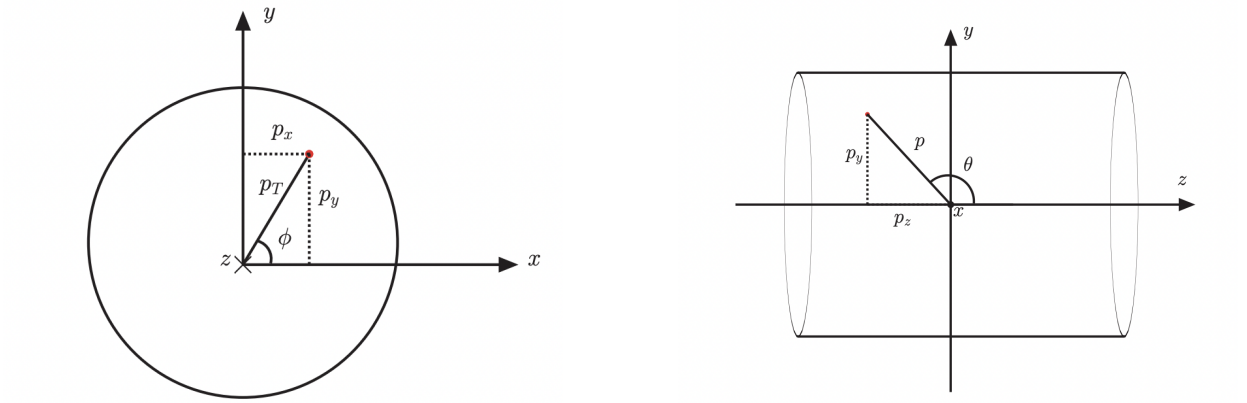
Chapter 3

Implementation

3.1 The ATLAS detector

Kinematics and detector geometry

Before one can analyse the data, it has to be collected and processed in a detector. The data used in this thesis are generated from proton-proton collisions in the ATLAS detector at the LHC. The ATLAS inner detector itself is a solenoid, and the kinematic variables are measured based on the following coordinate system. The z-axis is defined to go along the center axis of the solenoid, whereas the y-axis points upwards in the detector and the x-axis radialy outwards from the center axis. This allows for all transverse variables to be defined in the x-y plane[22]. From this we contruct the azimuthal and polar angles ϕ and θ, where the azimuthal angle ϕ[23] is the angle around the z-axis, and the polar angle θ is the angle from the z-axis, as shown in figure 3.1.



(a) Transverse plane of detector

(b) Longitudal plane of detector

Figure 3.1: Spherical coordinate definitions with the azimuthal and polar angles ϕ and θ. Here figure 3.1a of the transverse plane shows the z-axis into the paper, where as figure 3.1b of the longitudal plane shows the positive x-axis going out of the paper. Source here. Accessed 16.03.23

In figure 3.1 the kinematic variables for proton-proton collisions are described by the energy, rest mass and momentum, given as E, m, and $\mathbf{p} = (p_x, p_y, p_z)$ respectively. As the particles move with very high energy, we will use the relativistic 4 momentum[7], given as $\mathbf{P} = (E, \mathbf{p})$. We also have that

$$\gamma = \frac{1}{\sqrt{1-\beta^2}},$$

where γ is the Lorentz factor, and $\beta = \frac{v}{c}$, which gives us the following definitions for energy $E = \gamma m$ and momentum $\mathbf{p} = \beta\gamma m$[22]. From this we can derive the energy momentum formula:

[7] In special relativity, 4 momentum is used as both energy and momentum are conserved, and thus by creating the 4 momentum, you achieve a Lorentz invariant quantity[18].

$$\begin{aligned}\mathbf{p}^2 &= \beta^2\gamma^2 m^2 \\ \mathbf{p}^2 + m^2 &= m^2(\beta^2\gamma^2 + 1) \\ \mathbf{p}^2 + m^2 &= m^2\gamma^2 \\ \mathbf{p}^2 + m^2 &= E^2\end{aligned}$$

$$E = \sqrt{p^2 + m^2}. \tag{3.1}$$

It can be shown that the phase space of a particle is given by[24]:

$$d\mathbf{p} = dp_x dp_y dp_z = p^2 dp d\Omega = dp_z p_T dp_T d\phi, \tag{3.2}$$

where p_z is the momentum along the beam direction, p_T is the projected momentum on the transverse plane, and Ω is the solid angle. An analog to the relativistic longitudal velocity is the rapidity y. To define this we have that the relativistic generalization of equation 3.2 is given by:

$$d^4p\delta(E^2 - p^2 - m^2) = d\mathbf{p}\frac{1}{E} = p_T dp_T d\phi dy, \; dy = \frac{dp_z}{E}.$$

Using the fact that $p = \sqrt{p_T^2 + p_z^2}$ and equation 3.1 we can integrate dy to get the rapidity:

$$\begin{aligned}\int dy &= \int \frac{dp_z}{\sqrt{p_T^2 + p_z^2 + m^2}} \\ y &= \cosh^{-1}\left(\frac{E}{\sqrt{p_T^2 + m^2}}\right)\end{aligned} \tag{3.3}$$

For particles with little to no mass relative to the transverse momentum, we have that $p_T^2 + m^2 \approx p_T^2$ where $p_T = E\sin(\theta)$, which gives us the following relations:

$$\begin{aligned}\cosh(y) &= \frac{1}{\sin(\theta)}, \\ \sinh(y) &= \frac{1}{\tan(\theta)}, \\ \tanh(y) &= \cos(\theta).\end{aligned}$$

which can be used to show that $e^{-y} = \tan\frac{\theta}{2}$. From this we define the pseudorapidity η as:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right), \tag{3.4}$$

which is, in the relativistic limit, the same as the rapidity y. A useful property of the pseudorapidity is that the phase space of a single particle is uniformly distributed for both η and ϕ, making them good features to compare overlap between SM MC and ATLAS data[22].

Data collection

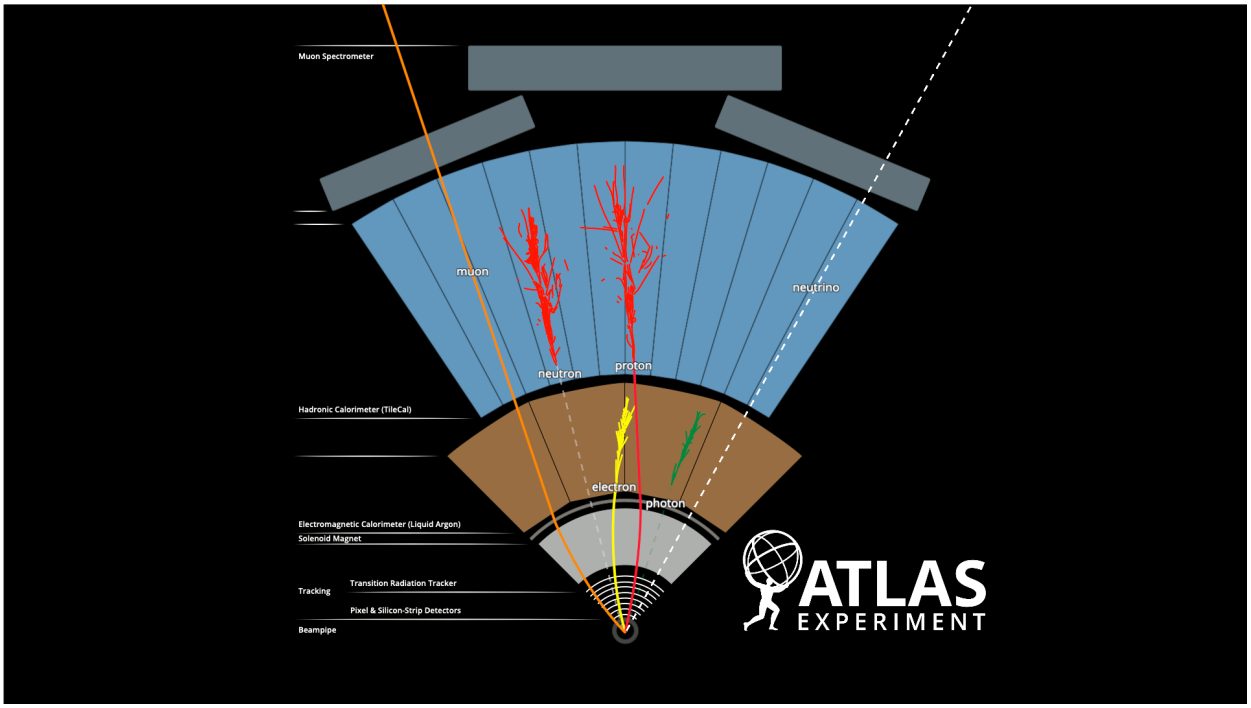


Figure 3.2: Figure describing how particles are detected at ATLAS, fetched from ATLAS detector slice (and particle visualisations), by Sascha Mehlhase [25] .

The features used in this analysis are computed with or fetched from the features from the detector itself. Such features include the momentum, energy, angles etc., all of which are either directly measured or computed based on the measurements in the detector. In figure 3.2 a visualization shows how different particles move through the detector and how they are detected. For example, energy deposits are measured using calorimeters, and the different particles have calorimeters specially designed for them. Charged particles leave tracks in the inner tracking device. Thus, electrons leaves tracks in the inner tracker and deposits energy in the elctromagnetic calorimeter, the muons leaves tracks in the inner detector and deposits energy in the muon detector. The photons only deposits energy in the electromagnetic calorimeter. Hadrons like protons and neutrons deposits energy in the electromagnetic and hadronic calorimeters, and charged hadrons also leaves tracks in the inner tracker. This is shown in figure 3.2.

The ATLAS detector have a few thousand selection stages before the data is stored. In order to reach the highest intensity of collisions, the LHC accellerates packets of around 10^{11} protons, and collides them at a rate of 25 nanoseconds, yeilding a collision rate of 40 MHz[26]. [27]

Data preparation

During datagathering at the ATLAS detector, triggers on the hardware and software level select out the events that are of most interest. On the order of 99% or more of the recorded events are discarded, with about 1 in 40000 events being accepted, as the amount of recorded events simply as too high to realistically analyse. Also, a lot of the events are of no interest for new physics analysis anyway as they for example might have too little energy. Once the trigger selection is done, the data is reconstructed. This means that the objects in the recorded events are through software algorithms reconstructed into particles, jets, photons etc. The reconstruction is done based on the tracks and measurements in the detector, but it is not perfect, and can lead to fake leptons or jets. By fake, it is meant that an object might look like a lepton but is in reality a jet or vice versa[28]. Once the reconstruction is done, further slimming of the n-tuples are done. Derivations are slimmings of the n-tuples where the selection of events are further reduced to match the needs of the different analysis groups.

The SM MC go through parts of the same process. These events are first generated and then run through the detector to simulate actual events. Once that is done, the events can be reconstructed and go through derivations just like the pp-collision data, to be used in analysis.

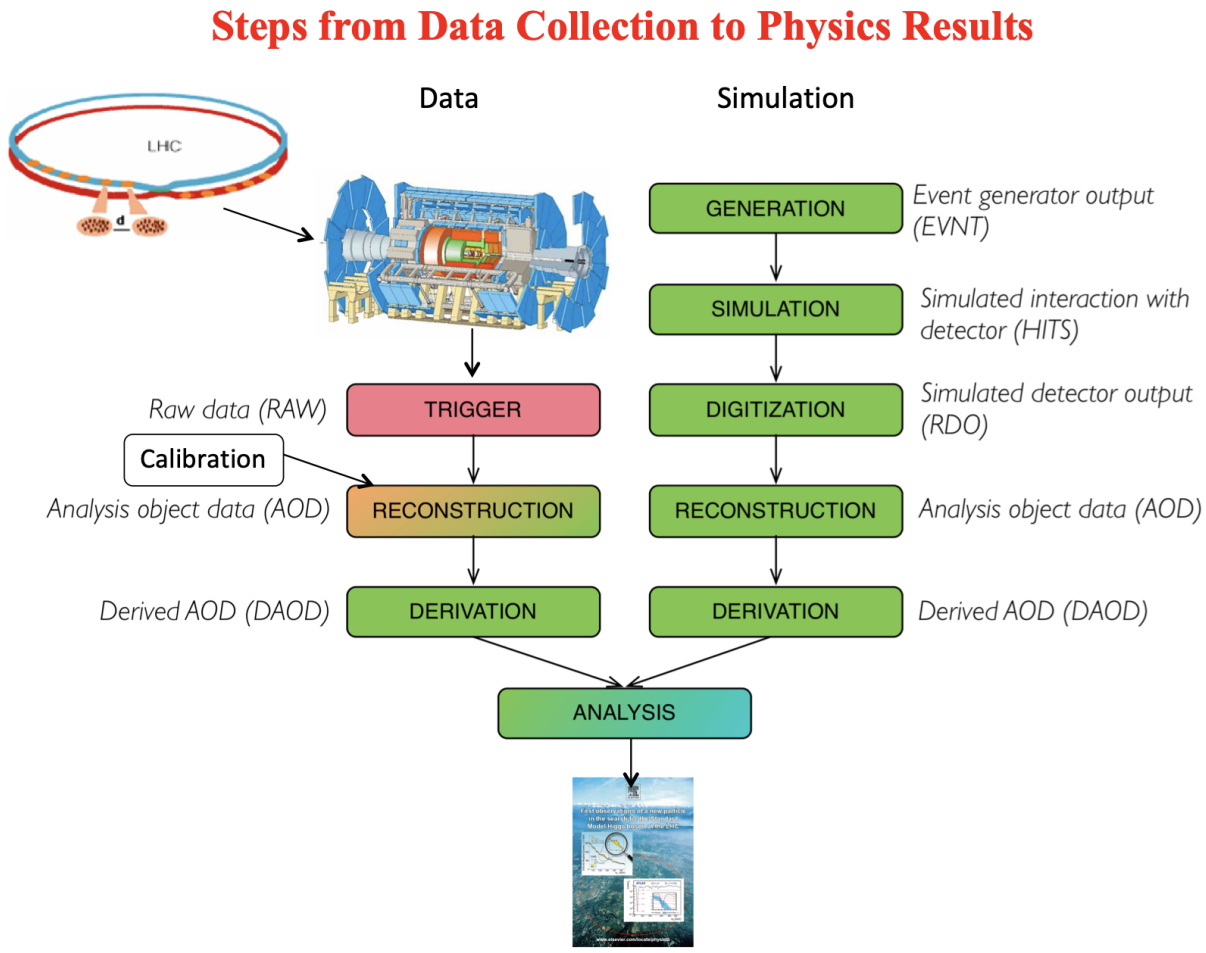


Figure 3.3: Figure describing the steps to take for data collection at ATLAS, fetched from Hybrid ATLAS Induction Day + Software Tutorial workshop, part Computing and Data preparation, held by S.M Wang [26] .

Jets

Photons and electrons are detected in the electromagnetic calorimeter, and are easy to track and detect as they separate easily. Quarks, however, are bound by QCD and thus cannot be seperated as individual particles. An illustration of how quarks and gluons materialise as jets during a proton-proton collision is shown below in figure 3.4.

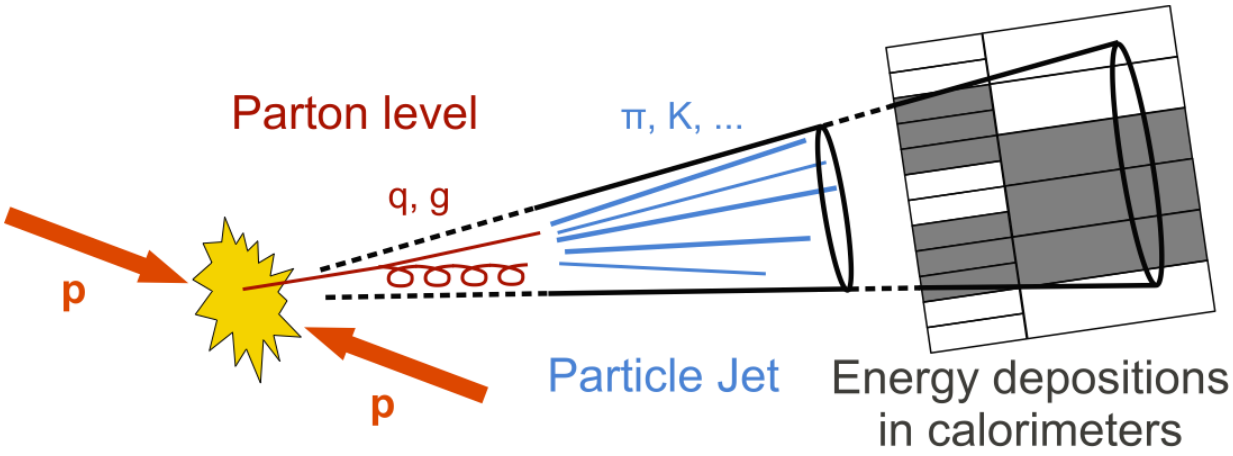


Figure 3.4: Figure describing how quarks and gluons are treated in the detector, and thus why we name them jets, fetched from the CMS webpage.

In a proton-proton collision, the quarks and gluons forms stable or unstable hadrons such that the color confinement[8] is upheld. These then decay to other stable hadrons that can be tracked, and these tracks are called jets. This is particularly difficult because one wants to isolate which hadrons came from the original quark in the Feynman diagram. Another point to make is that some quarks are of higher interest than others. For example, the b jet, coming from a b quark, is a good indicator for certain processes, thus identifying suchs particles is of huge interest.

3.2 ROOT

N tuples

The main datastructure of ROOT is the so called N tuple structure. This datastructure contains each property for each type of particle in a given event, yeilding a ragged structure.

	$jetP_T$	$jetPhi$	$lepP_T$	$lepPhi$
0	$[120.2, 57]$	$[1.2, 0.5]$	$[223.3, 57.5, 9.7]$	$[0.545, 0.2, -0.3]$
1	$[,]$	$[,]$	$[121.343, 89.323]$	$[0.886, -0.855]$
2	$[86.112]$	$[86.112]$	$[57.75, 34.5]$	$[0.33, 0.255]$

Rowlength
10
4
6

RDataFrame

[29]

RDataFrame's main purpose is to make reading and handling of root files easier, especially in relation to modern machine learning tools and their respective frameworks and environments. This is done by creating a dataframe like structure of the root n-tuples, and then lazily[9] apply contraints to the data. Using PyROOT, RDataFrame can be accessed in Python, as the functionality is wrapped around a C++ class. Below is an example of how to create a RDataFrame object, apply a cut and then create a column for later use. Here, good leptons are defined first, denoted as "ele_SG" and "muo_SG". A cut is then applied where we require that the number of good leptons is always 3. Finally, a column is created where the combination of type of leptons in the 3 lepton system is stored, as well as creating a histogram containing the results for that given channel[10] k. Notice here that if the variable already exist as a column in the dataframe, arithmetic and logic can be applied directly using those columns to create new one. More complicated variables, such as the flavor combination for the leptons, or the invariant mass of two particles must be found or calculated using C++ functions. An example of such a function is shown below the python code listing.

[8] Add link to a source or explanation for this.

[9] In this context lazily means that the functions and or cuts are done first after all have been registered, see ROOT guidelines for more.

[10] A channel here refers to a certain decay channel. The standard model has several, and some look more alike than others. One example is the Higgs decay channel, with possible decays such as two photons, W bosons or Z bosons.

```
1  import ROOT as R
2
3  R.EnableImplicitMT(200)
4  R.gROOT.ProcessLine(".L helperFunctions.cxx+")
5  R.gSystem.AddDynamicPath(str(dynamic_path))
6  R.gInterpreter.Declare(
7      '#include "helperFunctions.h"'
8  )  # Header with the definition of the myFilter function
9  R.gSystem.Load("helperFunctions_cxx.so")  # Library with the myFilter function
10
11 df_mc = getDataFrames(mypath_mc)
12 df_data = getDataFrames(mypath_data)
13 df = {**df_mc, **df_data}
14
15 for k in df.keys():
16
17     # Signal leptons
18     df[k] = df[k].Define(
19         "ele_SG",
20         "ele_BL && lepIsoLoose_VarRad && lepTight && (lepD0Sig <= 5 && lepD0Sig >= -5)",
21     )
22     df[k] = df[k].Define(
23         "muo_SG",
24         "muo_BL && lepIsoLoose_VarRad && (lepD0Sig <= 3 && lepD0Sig >= -3)",
25     )
26     df[k] = df[k].Define("isGoodLep", "ele_SG || muo_SG")
27     df[k] = df[k].Define(
28                 "nlep_SG", "ROOT::VecOps::Sum(ele_SG)+ROOT::VecOps::Sum(muo_SG)"
29             )
30
31     df[k] = df[k].Filter("nlep_SG == 3", "3 SG leptons")
32
33     # Define flavor combination based on
34     df[k] = df[k].Define("flcomp", "flavourComp3L(lepFlavor[ele_SG || muo_SG])")
35     histo[f"flcomp_{k}" ] = df[k].Histo1D(
36         (
37             f"h_flcomp_{k}",
38             f"h_flcomp_{k}",
39             len(fldic.keys()),
40             0,
41             len(fldic.keys()),
42         ),
43         "flcomp",
44         "wgt_SG",
45     )
46
```

In the code listing above we see an example of how RDataframe can be used for event selection. Line 1-9 are settings for ROOT, how many threads to use in the parallellization, extra helper functions written in C++ with .h and .so files and the path to the folder. The nex three lines creates a dictionary containing the ROOT RDataFrames to do event selection on. These are categorized by channel name. The loop then does event selection for each channel sample, defining new variables in the RDataFrame, applying filters, and creating histograms. Some variables are constructed using variables already in the ROOT files, such as energy and mass and so on, which through custom C++ functions can be added. In the codelisting below we see a custom C++ function which is used in this thesis.

```
double getM(VecF_t &pt_i, VecF_t &eta_i, VecF_t &phi_i, VecF_t &e_i,
            VecF_t &pt_j, VecF_t &eta_j, VecF_t &phi_j, VecF_t &e_j,
            int i, int j)
{
    /* Gets he invariant mass between two particles, be it jets or leptons */

    const auto size_i = int(pt_i.size());
    const auto size_j = int(pt_j.size());

    if (size_i == 0 || size_j == 0){return 0.;}
    if (i > size_i-1){return 0.;}
    if (j > size_j-1){return 0.;}

    TLorentzVector p1;
    TLorentzVector p2;

    p1.SetPtEtaPhiM(pt_i[i], eta_i[i], phi_i[i], e_i[i]);
    p2.SetPtEtaPhiM(pt_j[j], eta_j[j], phi_j[j], e_j[j]);

    double inv_mass = (p1 + p2).M();

    return inv_mass;
}
```

The C++ function listed above creates Lorentzvectors for two particles, and then constructs the invariant mass based on the parameters sent in. this function will be used on all the leptons in a given event, and in the case that one particle or both do not exist, the C++ function will return 0 as the invariant mass.

```
import pandas as pd

cols = df.keys()

for k in cols:

    print(f"Transforming {k}.ROOT to numpy")
    numpy = df[k].AsNumpy(DATAFRAME_COLS)
    print(f"Numpy conversion done for {k}.ROOT")
    df1 = pd.DataFrame(data=numpy)
    print(f"Transformation done")


    df1.to_hdf(
        PATH_TO_STORE + f"/{k}_3lep_df_forML_bkg_signal_fromRDF.hdf5", "mini"
    )
```

Once eventselection is done, the features have been chosen and histograms have been drawn, the Rdataframe can be converted to a pandas dataframe, which is a very popular choice for data structure when doing data analysis in python. This is done through an intermediary step of converting the RDataframe to a numpy filestructure, which then can be converted to a pandas[30] dataframe, or some other framwork. Here the new pandas dataframe is stored as hdf5[31] files to be used later, as the hdf5 format has a very good compression ratio, and is very fast to read and write.

3.3 Background samples

To look for anomalies in the three lepton final state we need to train on background MonteCarlo samples with that specific final state. This means in a sense that we want the autoencoder to learn what is expected from the Standard model in terms of this final state. The 2 and 3 lepton + e_T^{miss} background MonteCarlo contains the following channels:

Table 3.1: SM MC channels for both the 2 and 3 lepton + e_T^{miss} final state background.

3 lepton + e_T^{miss}	**2 lepton + e_T^{miss}**
Channel names:	
Wjets	Wjets
ttbar	ttbar
Singletop	Singletop
ZeeJets	ZeeJets
ZmmJets	ZmmJets
ZttJets	ZttJets
Higgs	Diboson
Triboson	
TopOther	
Diboson2L	
Diboson3L	
Diboson4L	

Below are three Feynamn diagrams, two of which are both represented in the 2 and 3 lepton + e_T^{miss} background MonteCarlo shown in table 3.1. The selected ones are likely Feynman diagrams for $t\bar{t}$, Higgs and Zeejets channels.

Figure 3.5: Proton-proton collision showing the $t\bar{t}$ channel. Here the w bosons decay leptonically and one or more jets can be misreconstructed as fake leptons by the detector.

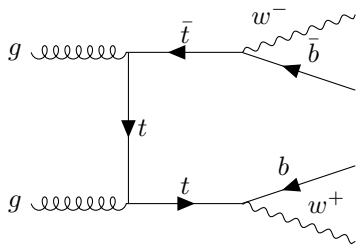


Figure 3.6: Proton-proton collision showing the Higgs channel. Here the Z bosons decay leptonically.

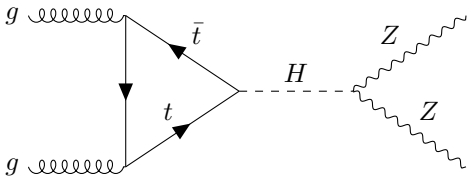
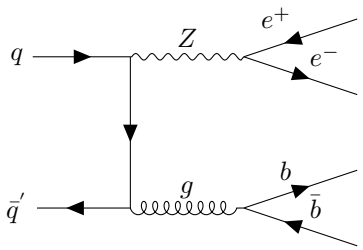


Figure 3.7: Proton-proton collision showing the Zeejets channel. Here one of the Z bosons decay leptonically and the gluon decays hadronically.



3.4 The dataset features

The Rapidity-Mass matrix (RMM)

Most of the features in the analysis are elements in the so called Rapidity-Mass matrix (RMM) inspired by the work of Chekanov [32].

The RMM is a convenient structure to create a feature space for the dataset. It contains information of various reconstructed objects and their combinations about mass, rapidity, momenta and missing transverse energy, all of which are useful in searches for new physics[33] in HEP. One example of an analysis that have used some features from the RMM is demontrated in [34]. The main reason however for using this structure is the systematic layout and automated featurespace, that maintains low to no corrolation between the cells in the matrix, as this is ideal when using neural networks.

Its composition is determined as a square matrix of $1 + \sum_{i=1}^{T} N_i$ columns and rows, where T is the total number of objects (i.e. jets, electrons etc.), and N_i is the multiplicity of a given object. In the case of the same number of a given object for all objects, we can denote the RMM matrix as a TmNn matrix, where m is the multiplicity of T, and n is the number of particle per type. Thus, there is already room for evaluation, as the combination of number of objects and the number of each object type highly affects the analysis as well as computational resources. Each cell in the matrix contains information about either single og two particle properties. This could in principle be generalized to three particle properties, which would make a three-dimensional RMM. The scope of thesis will for simplicity only cover the two-dimensional case. An example is shown in matrix 3.5.

$$\begin{pmatrix} \boldsymbol{e}_T^{miss} & m_T(j_1) & m_T(j_2) & m_T(e_1) & m_T(e_2) \\ h_L(j_1) & \boldsymbol{e_T}(j_1) & m(j_1,j_2) & m(j_1,e_1) & m(j_1,e_2) \\ h_L(j_2) & h(j_2,j_1) & \delta\boldsymbol{e_T}(j_2) & m(j_2,e_1) & m(j_2,e_2) \\ h_L(e_1) & h(e_1,j_1) & h(e_1,j_2) & \boldsymbol{e_T}(e_1) & m(e_1,e_2) \\ h_L(e_2) & h(e_2,j_1) & h(e_2,j_2) & h(e_2,e_1) & \delta\boldsymbol{e_T}(e_2) \end{pmatrix} \tag{3.5}$$

In matrix 3.5 we have the RMM matrix for a T2N2 system, in other words we have two types of particles, jets [11] and electrons, where each type has two particles. The matrix itself is partitioned into three parts. The diagonal represents energy properties, the upper triangular represents mass properties, and the lower triangular represents longitudal properties related to rapidity. The diagonal has three different properties, $\boldsymbol{e_T^{miss}}$, $\boldsymbol{e_T}$ and $\delta\boldsymbol{e_T}$. $\boldsymbol{e_T^{miss}}$ is placed in the $(0,0)$ position in the matrix. It accounts for the missing transverse energy for the system, which is of high interest for this analysis due to the search for heavy neutrinos. $\boldsymbol{e_T}$ is the transverse energy defined as

$$\boldsymbol{e_T} = \sqrt{m^2 + p_T^2} \tag{3.6}$$

but for light particles such as electrons, this can be approximated to $\boldsymbol{e_T} \approx p_T$. $\delta\boldsymbol{e_T}$ is the transverse energy imbalance. It is defined as

$$\delta\boldsymbol{e_T} = \frac{E_T(i_n - 1) - E_T(i_n)}{E_T(i_n - 1) + E_T(i_n)},\ n = 2, ..., N. \tag{3.7}$$

The first columm in the RMM matrix, with the exeption of the first element, is related to the longitudal property of the given particle. It is defined as

$$h_L(i_n) = C(\cosh(y) - 1),$$

where C is a constant to ensure that the average $h_L(i_n)$ values do not deviate too much from the ranges of the invariant masses of the transverse masses, found to be 0.15, as it ensures that rapidity ranges in the range $[-2.5, 2.5]$ produces $h_L(i_n)$ values in the $[0, 1]$ interval[32]. y is the rapidity of the particle, and i_n is the particle number. On the lower triangle we have the longitudal properties of the combinations of particles. Similar to $h_L(i_n)$, this property is defined as

$$h(i_n, j_k) = C(\cosh(\Delta y) - 1),$$

where $\Delta y = y_{i_n} - y_{j_k}$ is the rapidity difference for particle i_n and j_k.

Tabular and sparce data

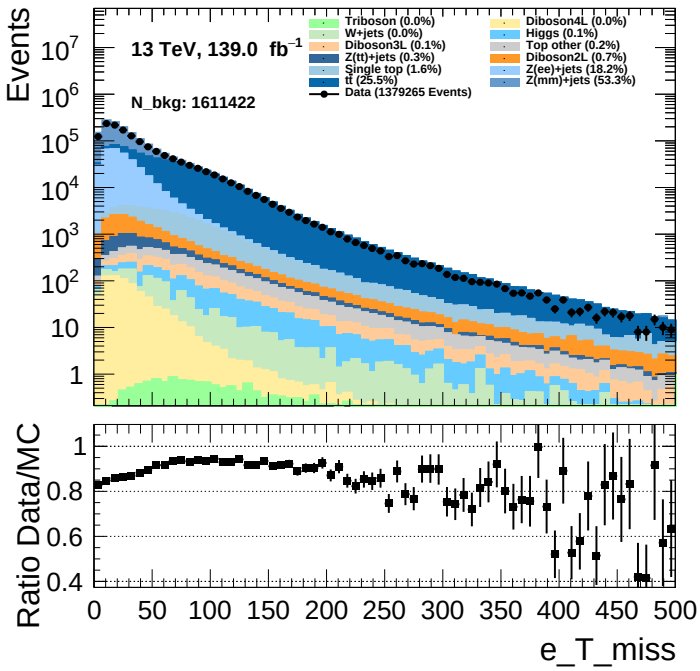
A concequence of using the RMM structure is that the data and Monte Carlo are sparce. This is due to the fact that the RMM allows for the variaty of final states of the reconstructed events, i.e. that one event has two ljets, zero bjets, one electron and two muons, whereas another event can have 4 ljet, 3 bjets and three electrons. This means that the RMM matrix for each event will have a different size, and for neural networks this is a problem. To solve this problem, Chekanov simply pads the missing values with 0s[32].

[11] Jets here can both be b- or ljets. Ljet is defined as jets with jetdl1r < 0.665, whereas bjet77 is defined as jets with jetdl1r>=2.195, where jetdl1r is a machine learning output from a network trained to destinguish b- and ljets.

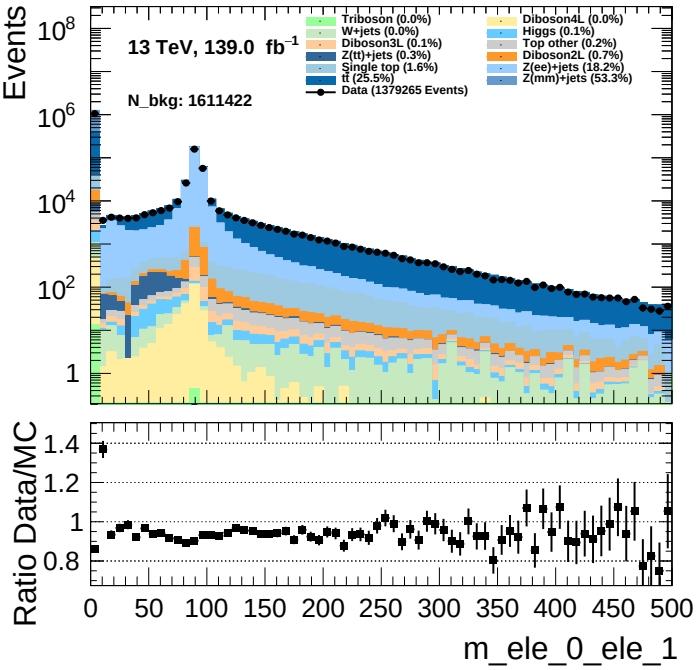
MonteCarlo and data comparison

Before we can start the analysis, we need to compare the MonteCarlo and data. This is done to ensure that the measured features used are reconstructed well by the MonteCarlo training samples we use. As described by R. Stuart Geiger et al. [35], the concept of ”Garbage in, garbage out” is of key importance in computer science, and indeed important in high energy physics. To ensure that the models we train actually learn physical processes, the training set must represent the physics ”status quo”. If the training samples do not match the physical reality, we regard it, in the context of high energy physics, as garbage in, which will in turn give garbage out. The Monte Carlo standard model simulations are indeed very good, but they are numerical approximations, and can sometimes be off. Thus, every feature that will be used for training have to be checked before being used. This is done by comparing the distributions of the features in the MonteCarlo and ATLAS data. MonteCarlo simulations are based on the actual theory itself, and comparisons with data taken from ATLAS and other detectors alike are neccesary to prove that the standard model is a good model.

Now, if we compare all SM MonteCarlo and ATLAS data, we would usually expect there to be a good overlap. To ensure that the standard model MonteCarlo actualy represents the physics, we create signal and background regions to optimize for a signal and or background. If we can create a background region where we believe with very high certainty that only standard model processes can occur, and we get a good match, we usually conclude that the MonteCarlo is good enough. Now, for this thesis, simply comparing all ATLAS data to all standard model MonteCarlo is enough, as this data batch has been analysed by the ATLAS collaboration for multiple years without finding any new physics, concluding that if the signals are there, they are too small for so-called visual cuts. Traditional searches have only exluded some models, which is why machine learning is getting more popular. The hope is that the signal, whatever it might be, can be revealed with clever feature engineering and smart machine learning algorithms. Particle physics differs here from more day-to-day machine learning as the target data is unlabeled.



(a) Missing transverse energy for the three lepton final state in GeV. The histogram contains the entire Run 2 dataset.

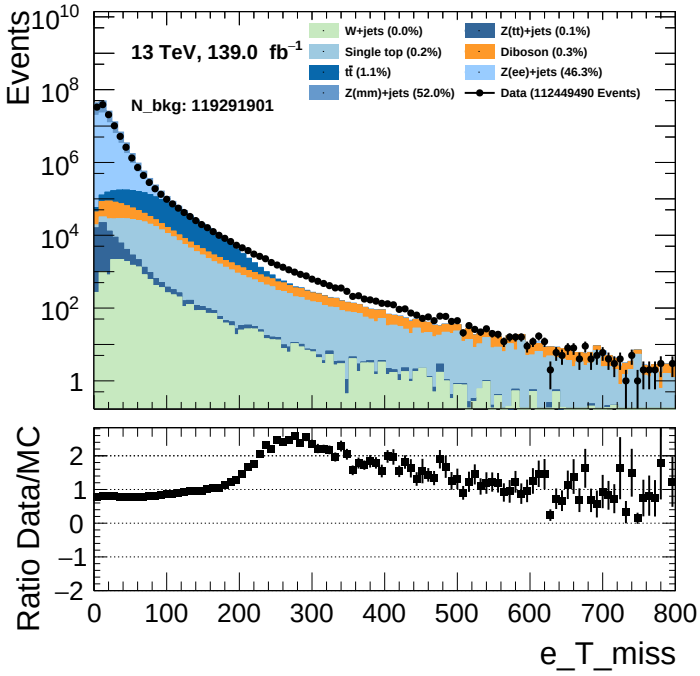


(b) Invariant mass for the first and second electron in GeV. The histogram contains the entire Run 2 dataset.

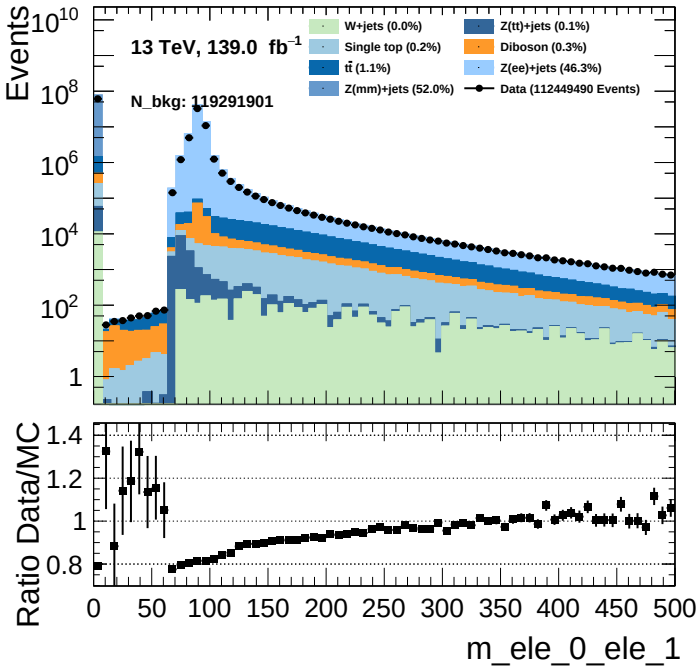
Figure 3.8: Comparison of the MonteCarlo and data for the three lepton + e_T^{miss} final state with the features e_T^{miss} and flavor composition.

In figure 3.8 two features have been selected to vizualize the comparison between Monte Carlo and ATLAS data, e_T^{miss} and $m(ele_0, ele_1)$ in the 3 lepton + e_T^{miss} dataset. We see that both e_T^{miss} and $m(ele_0, ele_1)$ satisfy a good ratio between Monte Carlo and ATLAS data, thus we can safely move forward with the analysis. All

features were checked, and can be found in the GitHub repository for this thesis at Figures/Histo_var_check[12] under the 3lep folder.



(a) Missing transverse energy for the three lepton final state in GeV. The histogram contains the entire Run 2 dataset. Note here the lack of good overlap from about 200-500 GeV.



(b) Invariant mass for the first and second electron in GeV. The histogram contains the entire Run 2 dataset. Note the sharp reduction of events from 10-70 GeV.

Figure 3.9: Comparison of the MonteCarlo and data for the 2 lepton + e_T^{miss} final state with the features e_T^{miss} and flavor composition.

[12]Full link: *https://github.com/Gadangadang/MasterThesis/tree/main/Figures/histo_var_check/LEP3*

The same checks were done for the 2 lepton + e_T^{miss} dataset. In figure 3.9 we see that the features e_T^{miss} and $m(ele_0, ele_1)$ do not satisfy the same ratio between SM MC and ATLAS data, as with the 3 lepton + e_T^{miss} case. It appears that RDataframe struggles with some diboson events, leading to a discrepency. This could be a trigger matching issue, or something else. The samples were run using both C++ ROOT event selection and RDataFrame by one of the supervisors, and the issue seems to only occure with RDataFrame. Thus, for all results regarding the 2 lepton + e_T^{miss} dataset should be interpreted with this in mind. One attempt can be shown in figure 3.9b where a cut on the invariant mass above 70 GeV of at least the two leptons with the highest energy, to try to acommodate the trigger issue. It did not seem to work. All features were checked, and can be found in the GitHub repository for this thesis at Figures/Histo_var_check[13] under the 2lep folder.

Triggers

The following triggers where implemented for the 2 lepton + e_T^{miss} and 3 lepton + e_T^{miss} dataset are written below:

Table 3.2: Triggers used in the 2015 MonteCarlo and ATLAS data samples for the 2 lepton 0 e_T^{miss} dataset.

	Name
2015	$HLT\_2e15\_lhvloose\_nod0\_L12EM13VH$
	$HLT\_2e12_{l}hloose_{L}12EM10VH$
	HLT_2mu10
	$HLT\_mu18\_mu8noL1$
	$HLT\_e17\_lhloose\_mu14$
	$HLT\_e7\_lhmedium\_mu24$

Table 3.3: Triggers used in the 2016 MonteCarlo and ATLAS data samples for the 2 lepton 0 e_T^{miss} dataset.

	Name
2016	$HLT\_2e15\_lhvloose\_nod0\_L12EM13VH$
	$HLT\_2e17\_lhvloose\_nod0$
	HLT_2mu10
	HLT_2mu14
	$HLT\_mu20\_mu8noL1$
	$HLT\_mu22\_mu8noL1$
	$HLT\_e17\_lhloose\_nod0\_mu14$
	$HLT\_e24\_lhmedium\_nod0\_L1EM20VHI\_mu8noL1$
	$HLT\_e7\_lhmedium\_nod0\_mu24$

Table 3.4: Triggers used in the 2017 MonteCarlo and ATLAS data samples for the 2 lepton 0 e_T^{miss} dataset.

	Name
2017	$HLT\_2e17\_lhvloose\_nod0\_L12EM15VHI$
	$HLT\_2e24\_lhvloose\_nod0$
	HLT_2mu14
	$HLT\_mu22\_mu8noL1$
	$HLT\_e17\_lhloose\_nod0\_mu14$
	$HLT\_e26\_lhmedium\_nod0\_mu8noL1$
	$HLT\_e7\_lhmedium\_nod0\_mu24$

[13]Full link: *https://github.com/Gadangadang/MasterThesis/tree/main/Figures/histo_var_check/LEP2*

Table 3.5: Triggers used in the 2018 MonteCarlo and ATLAS data samples for the 2 lepton 0 e_T^{miss} dataset.

	Name
2018	$HLT\_2e17\_lhvloose\_nod0\_L12EM15VHI$
	$HLT\_2e24\_lhvloose\_nod0$
	HLT_2mu14
	$HLT\_mu22\_mu8noL1$
	$HLT\_e17\_lhloose\_nod0\_mu14$
	$HLT\_e26\_lhmedium\_nod0\_mu8noL1$
	$HLT\_e7\_lhmedium\_nod0\_mu24$

In tables 3.2, 3.3, 3.4, 3.5 we have the triggers used for the different data generations used for the 2 lepton + e_T^{miss}, i.e. the data taken from the 4 years of data collection in Run 2. Each trigger is designed to detect certain base expectations in the detector, for example reconstructed leptons. In the tables above, the triggers are layed out, showing which were used for a given data generation year. For a more in depth understanding of trigger, it is recommended to look at the "Performance of the ATLAS Trigger System in 2015" paper[36], as well as the paper for the electron and photon triggers paper[37] and the muon trigger paper[38]. The most important thing to note about these triggers is that the second argument in the trigger, in other words the component after "HLT_" tells you about the leptons and their transverse momentum criteria. The trigger "HLT_2mu14" requires two reconstructed muons with a transverse momentum of at least 14 GeV to be triggered for a given event. The other componentents in some other triggers are more complicated, and describe quality of reconstruction and more.

For the 3 lepton + e_T^{miss} final state, the trigger system did not implement well in RDataframe, thus a simple cut of requiring at least two leptons with a p_T above 20 GeV was implemented. Note that further triggers would refine the dataset even more.

3.5 Code implementation

Machine learning implementation

The machine learning analysis was written with Keras[39] using the Tensorflow api[2]. The machine learning structure was written using a functional structure[14]. In practise, this model could just as well have been written as a Sequential model[15], but at a cost of flexibility and lack of potential non-linear structure in the architecture. The code consists of one general class for the autoencoder, where the different testing cases are different classes inheriting from the parent class.

Construction of a neural network in Tensorflow

Using the functional structure, a general neural network in the Tensorflow API can be constructed as shown below.

[14]Functional structure uses a function call for layers, i.e. for layers a, b, then b(a) will connect the two layers, and equals a sequential link a → b. This allows for more flexible structures. More on the functional api can be found here.

[15]Sequential structure adds layers in sequence, i.e. for layers a, b, c we have that a → b → c, with a strict structure. This allows for more organized code. More on sequential models can be found here.

```
import tensorflow as tf


inputs = tf.keras.layers.Input(shape=data_shape, name="input")

# First hidden layer
First_layer = tf.keras.layers.Dense(
    units=30,
    activation="relu"
)(inputs)

# Second hidden layer
Second_layer = tf.keras.layers.Dense(
    units=45,
    activation="relu"
)(First_layer)

# Second hidden layer
output_layer = tf.keras.layers.Dense(
    units=1,
    activation="sigmoid"
)(Second_layer)


# Model definition
nn_model = tf.keras.Model(inputs, output_layer, name="nn_model")

hp_learning_rate = 0.0015
optimizer = tf.keras.optimizers.Adam(hp_learning_rate)
nn_model.compile(loss="mse", optimizer=optimizer, metrics=["mse"])
```

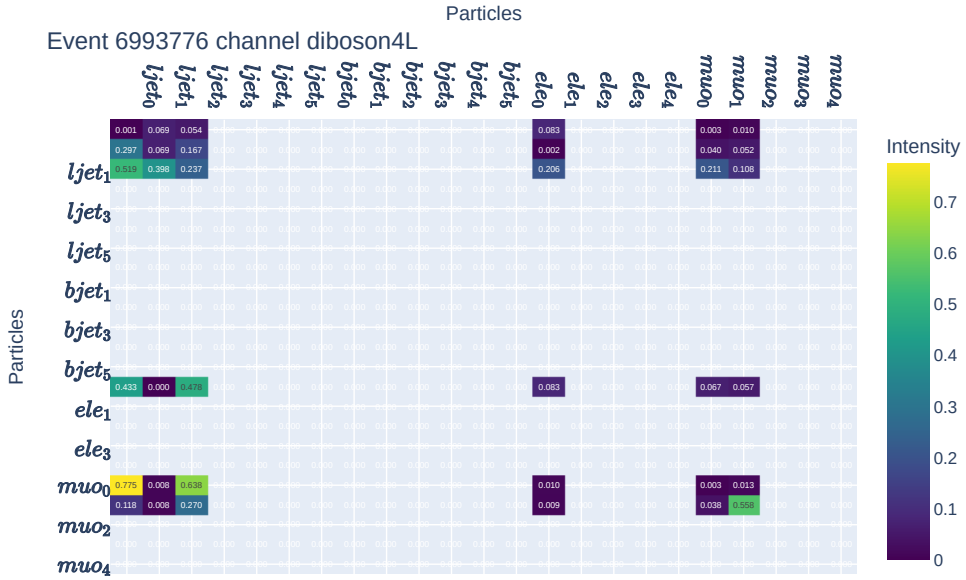
The neural network here contains one input layer, two hidden layers, and an output layer. The choice of nodes and activation functions are arbitrary here as the use case has not been defined. Note that this is exactly the same as the previous example, but using the sequential structure.

```
import tensorflow as tf

nn_model = tf.keras.Sequential(
    [
        tf.keras.layers.Dense(30, activation="relu", input_shape=data_shape),
        tf.keras.layers.Dense(45, activation="relu"),
        tf.keras.layers.Dense(1, activation="sigmoid"),
    ]
)

hp_learning_rate = 0.0015
optimizer = tf.keras.optimizers.Adam(hp_learning_rate)
nn_model.compile(loss="mse", optimizer=optimizer, metrics=["mse"])
```

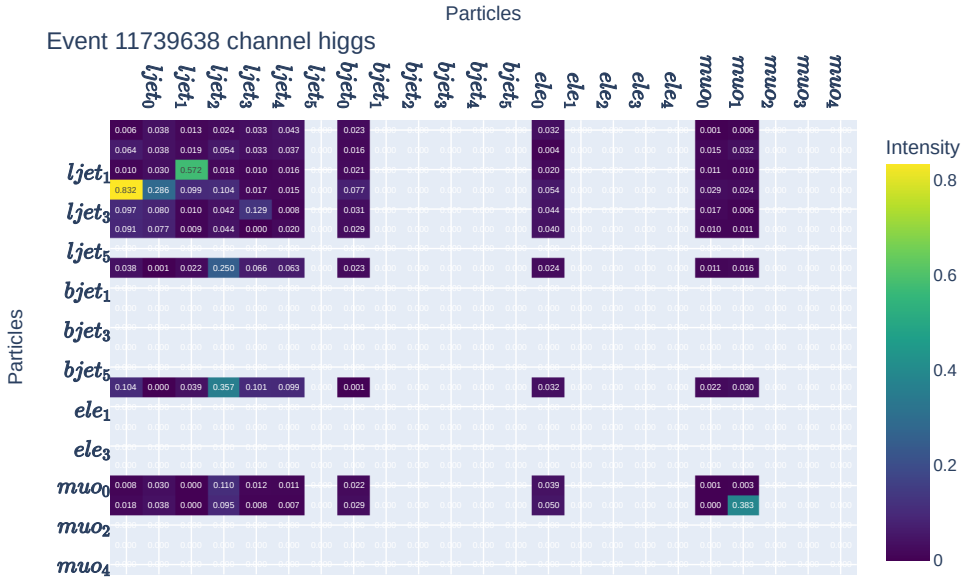
Data handling python side

Implementation of the RMM matrix

An example of the RMM matrices used in this thesis is shown in figure 3.10 below:



(a) RMM matrix for event number 6993776 from the MonteCarlo diboson4L sample. Each feature is scaled based on a fit for that feature for all events in the training set ($\approx 80\%$ of total MC). This sample contains two ljets, one electron and two muons.



(b) RMM matrix for event number 11739638 from the MonteCarlo Higgs sample. Each feature is scaled based on a fit for that feature for all events in the training set ($\approx 80\%$ of total MC). This sample contains five ljets, one bjet, one electron and two muons.

Figure 3.10: Note here that the y-axis for the RMM's lack every other label, due to lack of space in the y-axis of the plot. If looked more closely upon, one can see that each figure have all RMM cells, just that the labels, which are identical to the x-axis label, only show every other. Note also here that the labels only tell which particle are used for that row/column, i.e. in figure 3.10b in row 0 column 3 we have the invariant mass of the first and second ljet. The RMM plots were created using Plotly[40].

In figure 3.10 we see two RMM matrices created from two different channels in the MonteCarlo samples. This RMM is of type T4N5[16]. For easier interpretability, the gray area corresponds to a missing value, leading to so-called "islands" in the RMM matrix.

Setup for 3 lepton dataset

The 3 lepton dataset is about 96 Giga bytes of data when implementing the RMM structure for 6 b- and ljets, 5 electrons and 5 muons. The dataset is converted from a ROOT RDataframe to a Pandas dataframe[30] for further preprocessing. Having added the channel column in the dataset[17], the channel categories, weights, missing transverse energy and trilepton mass[18] as well as the RMM structure are divided into a training and validation/test set in an 80-20 split. This was done using the "$.fit\_transform()$" and "$.transform()$" functions from the Scikit-learn library[41]. The training and validation/test set are then converted to numpy arrays[42] for faster loading and easier indexing, and saved as ".npy" files. This allows for faster reuse of the arrays.

Setup via iterative training for 2 lepton dataset

The two lepton dataset contains about 1.5 - 2 Terra bytes of data when implementing the RMM structure for 6 b- and ljets, 5 electrons and 5 muons, as for the 3 lepton dataset. This is too much to hold in memory at the same time, thus it had to be split into several smaller datasets, called megasets. Below is a figure visualizing the structure used.

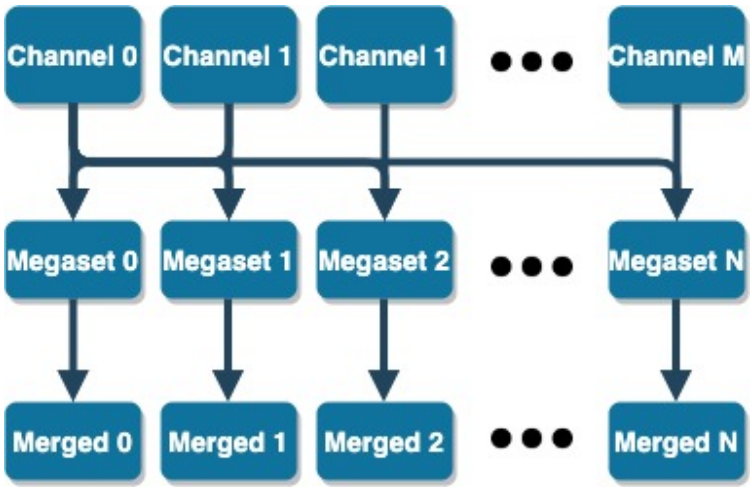


Figure 3.11: Megaset structure for the 2lep dataset. This figure generalises to M channels, and N megasets. The more you increase the number of megasets, the smaller each megaset will be in bytesize, but in order to keep the SM MC distribution, it is not recommended to make too small sets.

In figure 3.11 we see a generalised dividing structure. In the case of this thesis, each channel was divided into 10 equal parts, and stored in their respective folder. Pandas is not built for very large datasets, not running in a parallalized way. To handle this, the library Polars [19] [43] was used instead. When all channels were split, a merging was done combining all the channels in a given megaset to a separate dataset. The selection of events from each channel was done randomly, which is important, as we want to the best of our ability keep the distribution signature of the entire dataset in each megaset. If not, the model will be biased towards those datasets with the most events. Once each of the megasets where merged, the training could begin in an iterative fashion. Because Tensorflow is statically compiled, you cannot call the fit function over and over again. Instead, the weights trained based on one megaset is stored and reloaded into a new model, thus the weights are still trained on the entire set, but in a batch like manner.

[16] T4 → 4 particle types: bjets, ljets, electrons and muons. N5 → 5 particles per particle type. Note here that we have 5 particles only for the leptons, and 6 particles for each of the types of jets.

[17] This column would in a fully supervised setting be used as a target vector, but in this thesis it will only be used for legends in histograms and to index out certain channels in the validation and training set.

[18] Invariant mass of three leptons. This is assured to exist from event selection in the 3 lepton dataset.

[19] Polars uses all available cores on the system and has excellent memory handling capability, see Polars User Guide

```
for megaset in range(totmegasets):

    #* Load model

    autoencoder = getModel()

    if megaset != 0:
        autoencoder.load_weights('./checkpoints/Megabatch_checkpoint')


    #* Run Training
    with tf.device("/GPU:0"):

        tf.config.optimizer.set_jit("autoclustering")

        autoencoder.fit(
            xtrain,
            xtrain,
            epochs=epochs,
            batch_size=b_size,
            validation_data=(xval, xval),
            sample_weight=x_train_weights,
        )


    AE_model.save_weights('./checkpoints/Megabatch_checkpoint')
```

3.6 The chosen neural network architectures

The regular Autoencoder

Below are the two models used for the regular autoencoder.

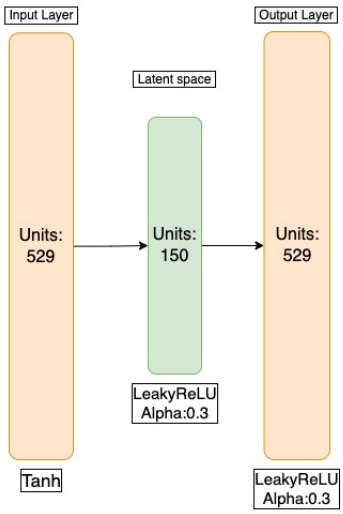


Figure 3.12: Small autoencoder architecture.

In figure 3.12 we have the small autoencoder. It consists of an input and output layer of 529 nodes, with one latent space layer of 150 nodes. The activation functions for the input, latent space and output are the Tanh and LeakyReLU with $\alpha = 0.3$ respectively,

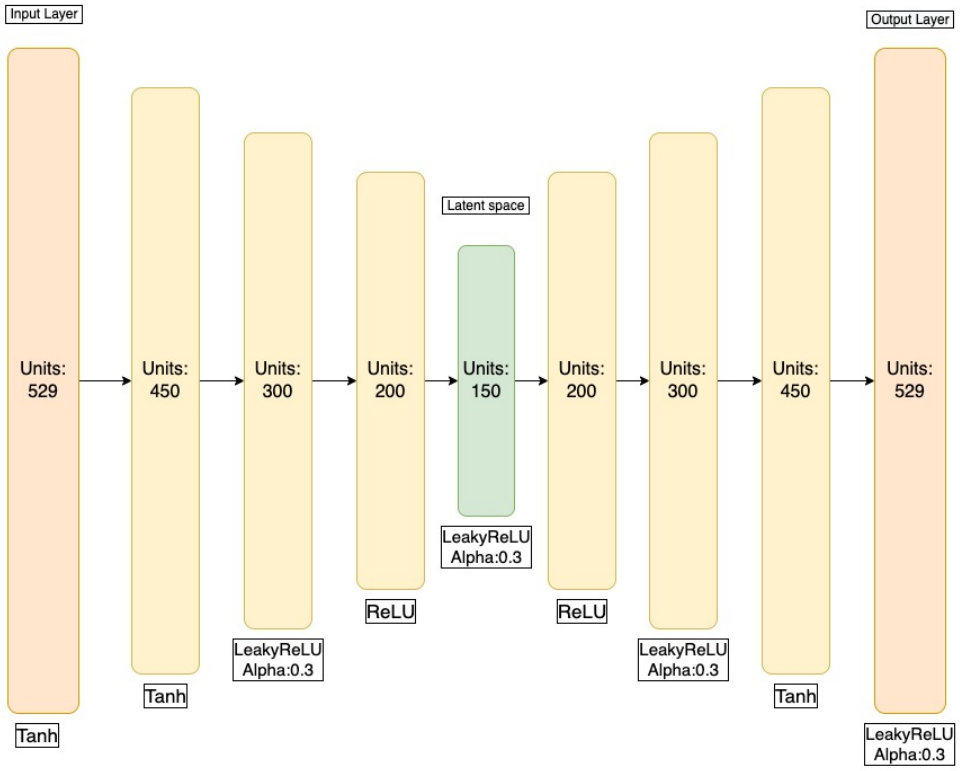


Figure 3.13: Large autoencoder architecture.

In figure 3.13 we have the large autoencoder. It consists of an input and output layer of 529 nodes, with three hidden layers of 450, 300 and 200 nodes respectively in the encoder and three hidden layers of 200, 300 and 450 respectively. The activation functions for the input and ouput layers are the Tanh and LeakyReLU with $\alpha = 0.3$ respectively. The hidden layers in the encoder have the activation functions Tanh, LeakyReLU with $\alpha = 0.3$ and ReLU respectively. The hidden layers in the decoder have the activation functions ReLU, LeakyReLU with $\alpha = 0.3$ and Tanh respectively. The latent space has 150 nodes, with the LeakyReLU activation function with $\alpha = 0.3$.

The variational Autoencoder

Below are the two models used for the variational autoencoder.

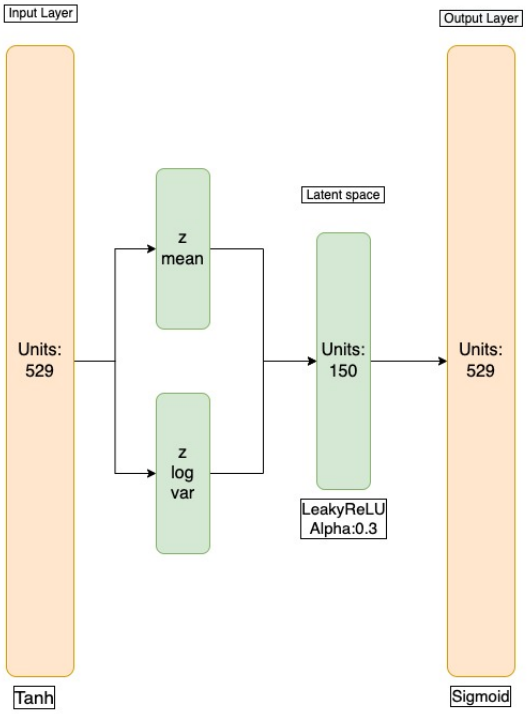


Figure 3.14: Small variational autoencoder architecture.

In figure 3.14 we have the small variational autoencoder. It consists of an input and output layer of 529 nodes, with one latent space layer of 150 nodes sampling from a mean and variance layer of same size. The activation functions for the input and output are the Tanh and LeakyReLU with $\alpha = 0.3$ respectively.

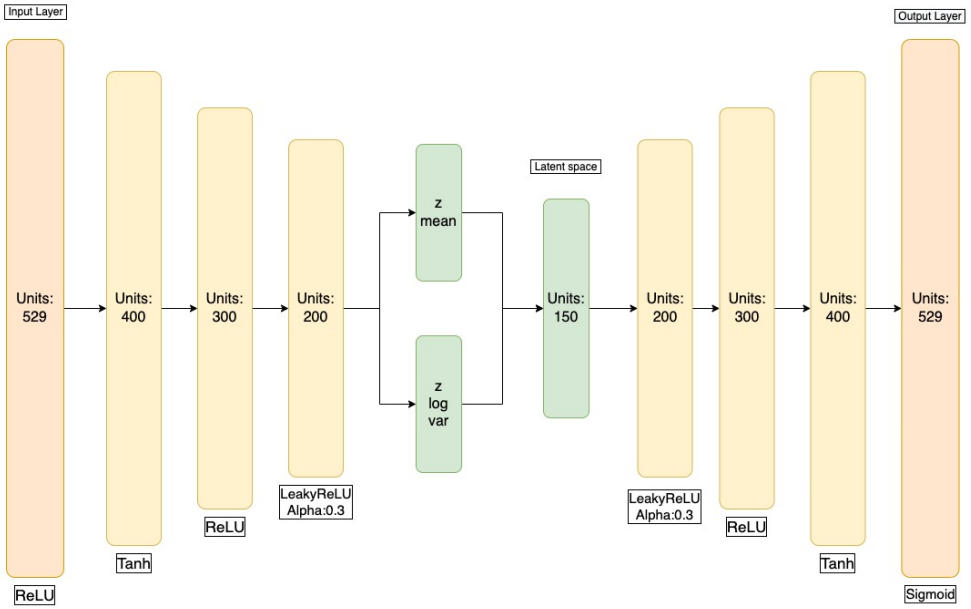


Figure 3.15: Large variational autoencoder architecture.

In figure 3.15 we have the large autoencoder. It consists of an input and output layer of 529 nodes, with three hidden layers of 400, 300 and 200 nodes respectively in the encoder and three hidden layers of 200, 300 and 400 respectively. The activation functions for the input and ouput layers are the ReLU and Sigmoid respectively. The hidden layers in the encoder have the activation functions Tanh, ReLU, LeakyReLU with $\alpha = 0.3$ respectively. The hidden layers in the decoder have the activation functions LeakyReLU with $\alpha = 0.3$, ReLU and Tanh respectively.

3.7 The search strategy

The strategy used to look for anomalies in the three lepton + missing energy finals state is presented as follows. First, the MonteCarlo and ATLAS data for training and inference are constructed with the RMM from section 3.4 as features. After scaling and splitting, 80% of the MC will be used for training the neural network, and the remaining 20% will be used for inference. To separate the anomalies, the reconstruction error of the input data will create a distribution. The same will then be done for a test signal. The ATLAS data is completely unlabeled which makes validation difficult. However, the test signals are labeled, thus we can analyze the reconstruction error distribution of those signals as a validation of performance when scientists at ATLAS do analysis on ATLAS data.

Standard analyses create what is called a signal region. The signal region is a region in the feature space where the signal is maximized. We use this region to calculate the significance of a result in the search, which is really the only metric that is of use. The statistical uncertainty and noise is proportional to the amount of SM MC, thus with lower amounts of SM MC, the better the significance will be. Using the reconstruction error, the autoencoder can create its own signal region, namely the areas of high reconstruction error. A cut here will then be used to separate the anomalies from the Standard model in for example missing transverse energy, or other features of interest.

The signal region for the regular autoencoder models were created by calculating the median m_{err} of the reconstruction error. Then, 3 cuts were made, starting at $m_{err} + im_{err}/5$ for $i = 1, 2, 3$. This was a direct result of the shape of the background reconstruction error, being a hill like shape. The median then became a good place to start to remove alot of the background. This is however just a guess for an optimal signal region, as the true signal is unknown, and the method has to be as unbiased as possible. There is however an issue to keep in mind here. The method to find the cut is in some sense based on the slope shape of the SM MC reconstruction error distribution, thus three cuts based on the median seems like a good choice. However, if one then uses all the event in the signal region, it might be that one misses the ideal amount of background and signal to create the significance. Thus, for each reconstruction error cut, there is an associated graph showing the significance as a function of e_T^{miss}. More specifically, the function calculates from a point and outwards, for all values in the e_T^{miss} distribution. Thus, you can find the ideal cut, within the signal region, for where to choose the amount of background and signal to get a better significance.

To avoid bias by the author, some test signals should contain some signal samples that the creator of the model has not seen before, to ensure no changes have been made to the network to adjust for that signal. ROC curves will then be used to evaluate the binary classification ability of the autoencoder.

Testing strategy

Before testing the AE and VAE on signal samples, it is of interest to test the sensitivity of both the regular and variational autoencoders on alterings on the Monte Carlo. This can be thought of as initial testing.

Channel removing

As the goal of the autoencoder is to reconstruction data is has looked on, one idea was to remove on of the channels in the standard model. The idea was that some of the channels differs enough in the final states they produce and thus the RMMs for the events in the given selection. All channels where tested on as signal, but one can expect some to have more similar results than others.

Altering transverse momentum

Another idea for anomaly detection testing with the MonteCarlo was to alter the transverse momentum of some of the particles. Random events where selected and had the transverse energy changed, in accordance with equation 3.6. The hope is that especially events with above 5 time increase in transverse momentum should be picked up. Note that by changing the transverse momentum, the change in transverse energy also changes. From equation 3.7 we have that a scale change k in p_T yields the following new relation:

$$\delta \boldsymbol{e}_{\boldsymbol{T}}^{\boldsymbol{k}} = \frac{kE_T(i_n - 1) - E_T(i_n)}{kE_T(i_n - 1) + E_T(i_n)}, \, n = 2, ..., N. \tag{3.8}$$

Thus, both the transverse energy and the change in transverse energy is changed for this test.

Signal testing

It would not technically be true to call the machine learning used in this thesis as unsupervised learning. This is mostly due to the fact that we show it labeled MonteCarlo to train on, and thus essentially gives the models a target to aim for. But, sinse we do not show it any signals in training, it can be allowed to name the learning strategy as semi-supervised learning. Because of this, one does not bias oneself too much if one then tests the algorithm on one or more signal samples. This type of testing can be separated into two different categories, the open signal testing and the blind signal testing.

The open signal testing first takes a trained model that has only seen SM MC and then runs inference on both SM MC and the given signal. Then the search strategy mentioned in the previous section is done on those collected reconstruction error distributions. The blind signal testing is the final test, where a dataset is prepared incapsulating SM MC, and one or multiple signals in a mix. The labels are kept, for verification. That way, if one were to do open signal testing on many signal samples from many different theories, one could in principle bias oneself too much. This way one can get an unbiased measurement of the performance.

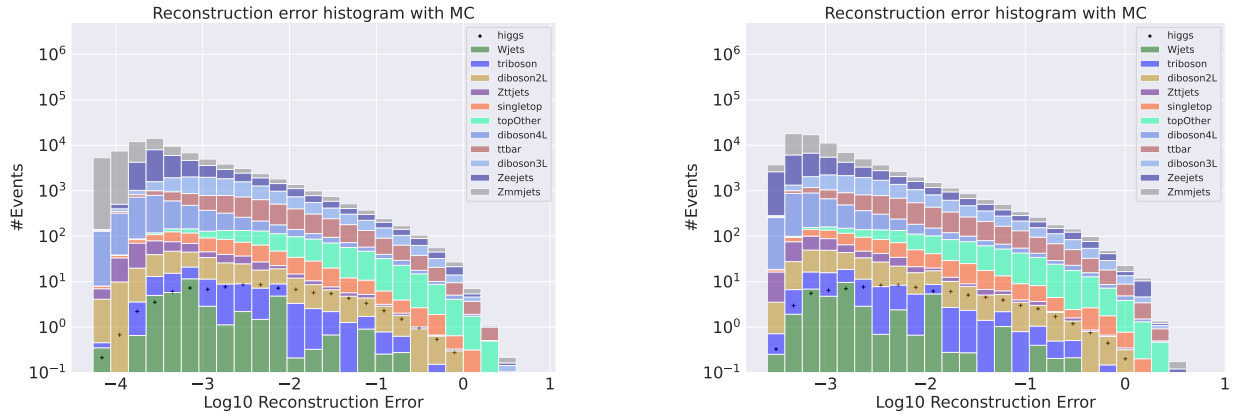
Chapter 4

Results and Discussion

4.1 Non signal testing of the regular and variational Autoencoder

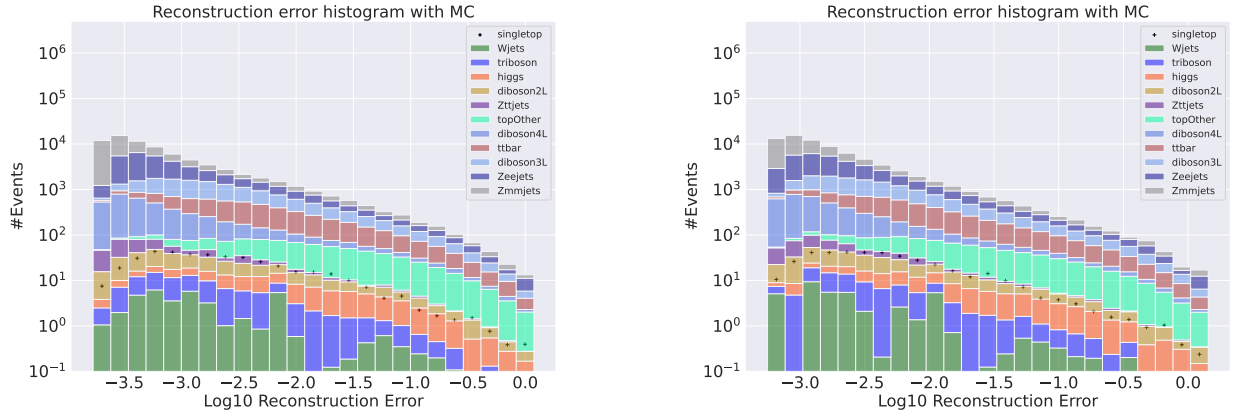
Channel removing

Both the large and small regular and variational autoencoder produced results, and are shown below. The Higgs, singletop and ttbar channels have been selected here, as they are from a physics stand point the channels that looks most unlike the other channels.



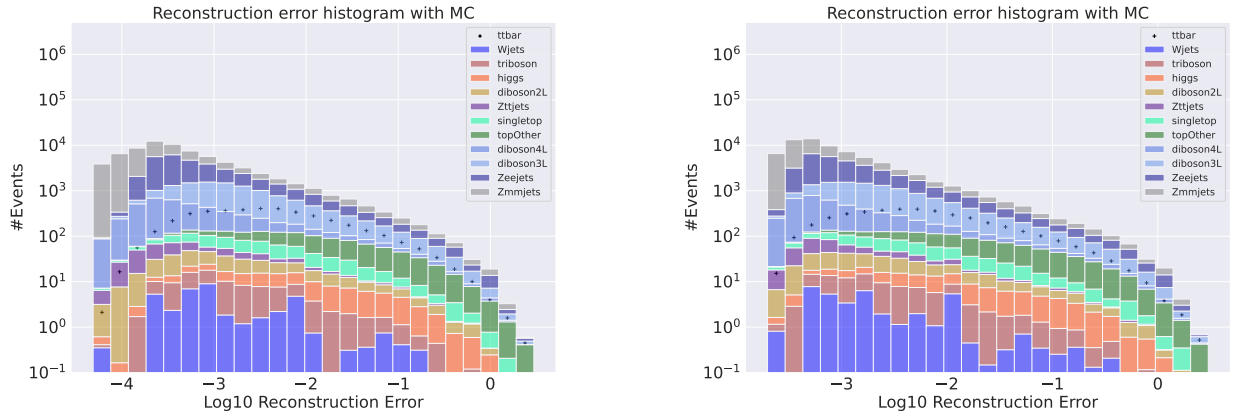
(a) **(b)**

Figure 4.1: Reconstruction error on validation SM MC from the small (left) and large (right) Autoencoders. Here the higgs channel has been removed from training and is used as signal. No significant difference in distributions is found.



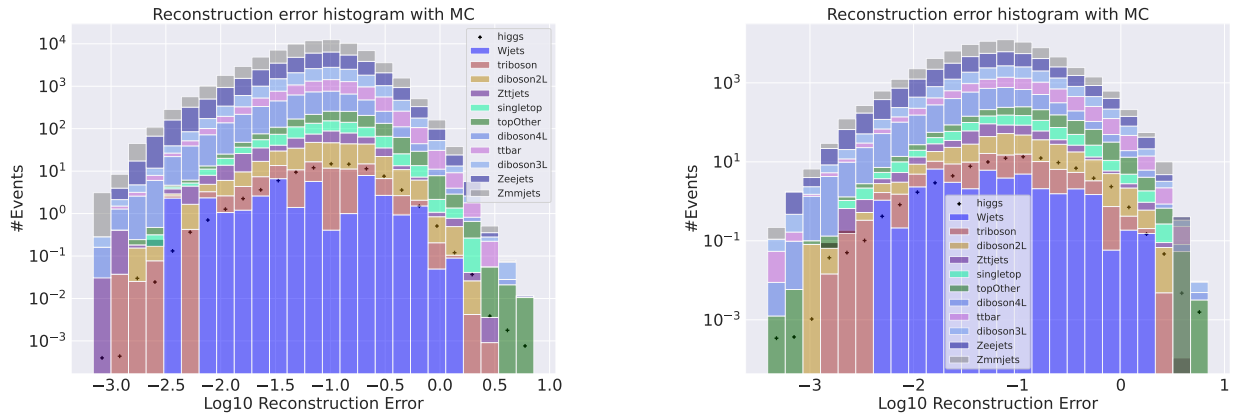
(a) **(b)**

Figure 4.2: Reconstruction error on validation SM MC from the small (left) and large (right) Autoencoders. Here the singletop channel has been removed from training and is used as signal. No significant difference in distributions is found.



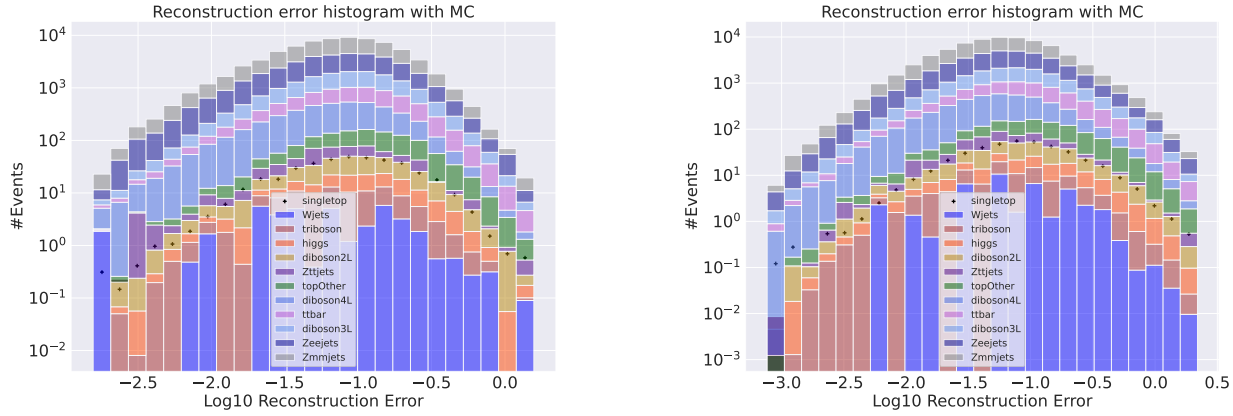
(a) **(b)**

Figure 4.3: Reconstruction error on validation SM MC from the small (left) and large (right) Autoencoder. Here the ttbar channel has been removed from training and is used as signal. No significant difference in distributions is found.



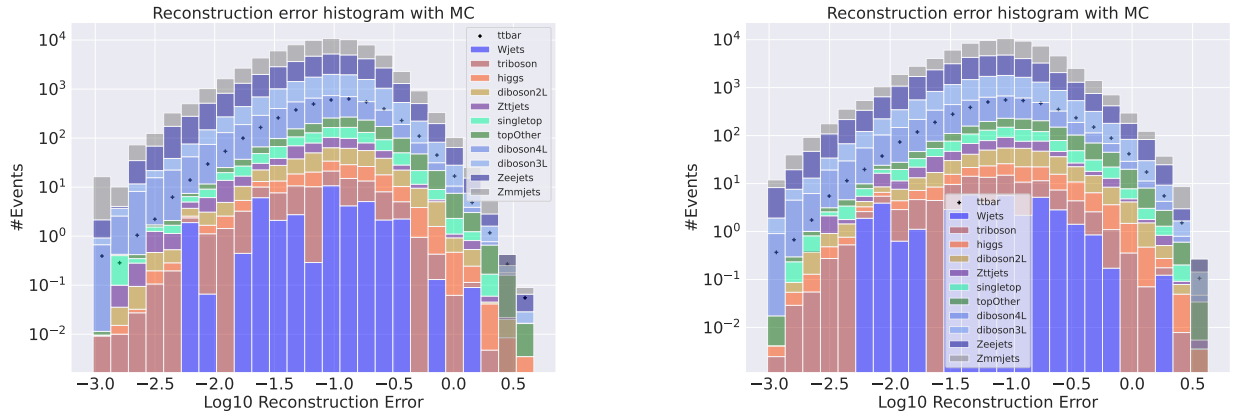
(a) **(b)**

Figure 4.4: Reconstruction error on validation SM MC from the small (left) and large (right) variational Autoencoder. Here the higgs channel has been removed from training and is used as signal. No significant difference in distributions is found.



(a) **(b)**

Figure 4.5: Reconstruction error on validation SM MC from the small (left) and large (right) variational Autoencoder. Here the singletop channel has been removed from training and is used as signal. No significant difference in distributions is found.



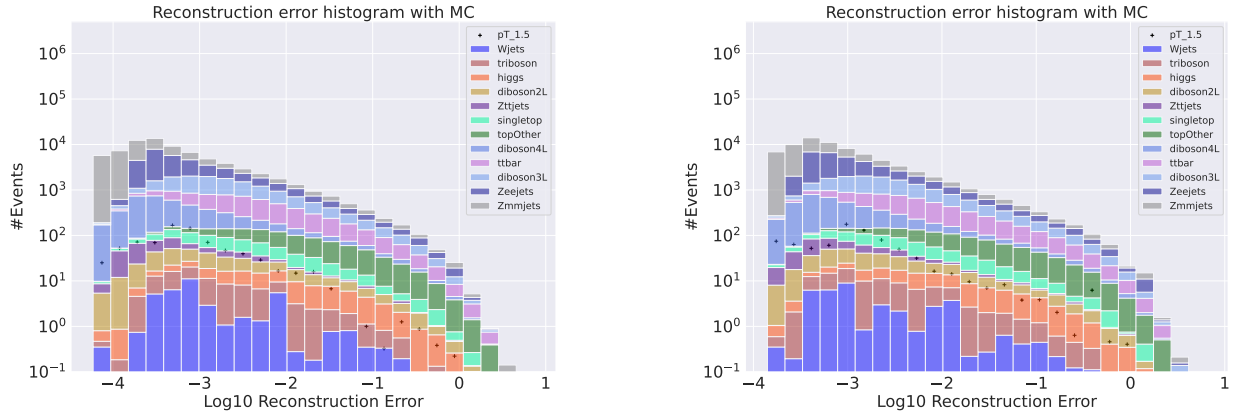
(a) **(b)**

Figure 4.6: Reconstruction error on validation SM MC from the small (left) and large (right) variational Autoencoder. Here the ttbar channel has been removed from training and is used as signal. No significant difference in distributions is found.

In figures 4.1 to 4.6 we have the reconstruction error distributions for the SM MC using Higgs, singletop and ttbar as signal samples for the small (to the left) and the big (to the right) regular and variational autoencoder. It is clear from figures that the reconstruction error distributions for the removed SM channel and the remainding SM MC are very similar. The same behavior is also shown for the other channels, which can found in appendix 2. Although the two models struggled to get a separation of reconstruction error distributions, the regular autoencoder seems to have a more shifted pattern of reconstruction to lower values than the variational autoencoder. In fact, the regular autoencoder's reconstruction error distribution peak is about 3 order of magnitude lower than the variational autoencoder's error distribution peak. There could be several reasons for this, one of which is that the variational autoencoder requires more input data to approximate the distribution, as well as the fact that the balance between the KL divergence and MSE loss can be difficult to handle[44]. This can lead to poor performance of the network. It should be notet that although the three channels selected here, as well as the rest in appendix 2, do not differ that much from one another, and the hopes were not high that the networks would be able to separate the two distributions created. However, it does provide us with a baseline, as well as insight for what to expect if we were to test on signals that looks a lot like some channels.

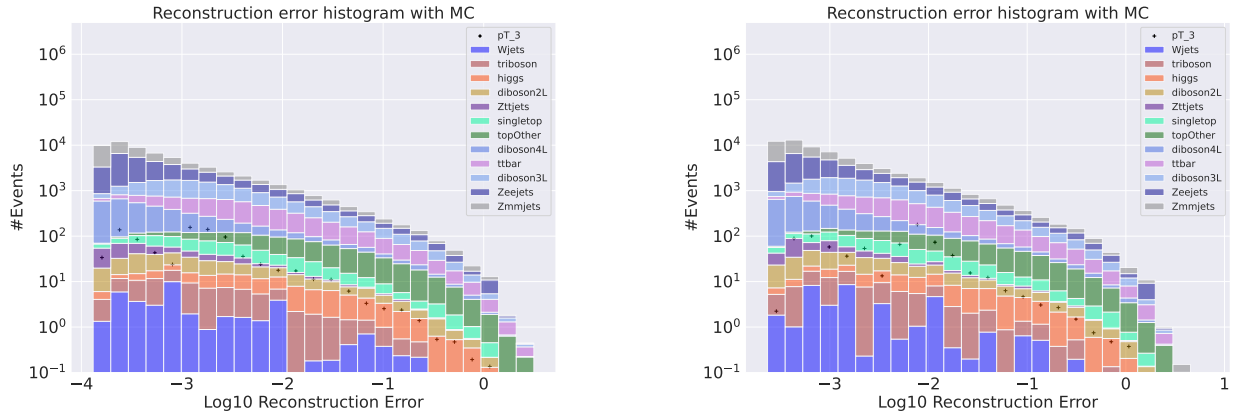
Altering of transverse momentum

Altering the transverse momentum of some particles would in the extreme be anomalous, and the hypothesis was that some of those trends would be picked up by the autoencoder. Several scales for the transverse momentum were used, with the following results.



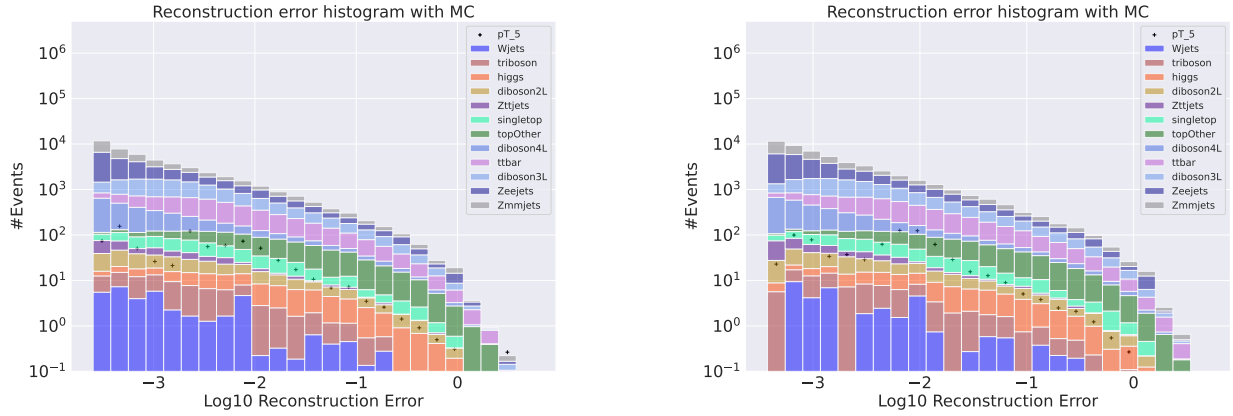
(a) **(b)**

Figure 4.7: Reconstruction error on validation SM MC from the small (left) and large (right) regular Autoencoder. Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 1.5. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.



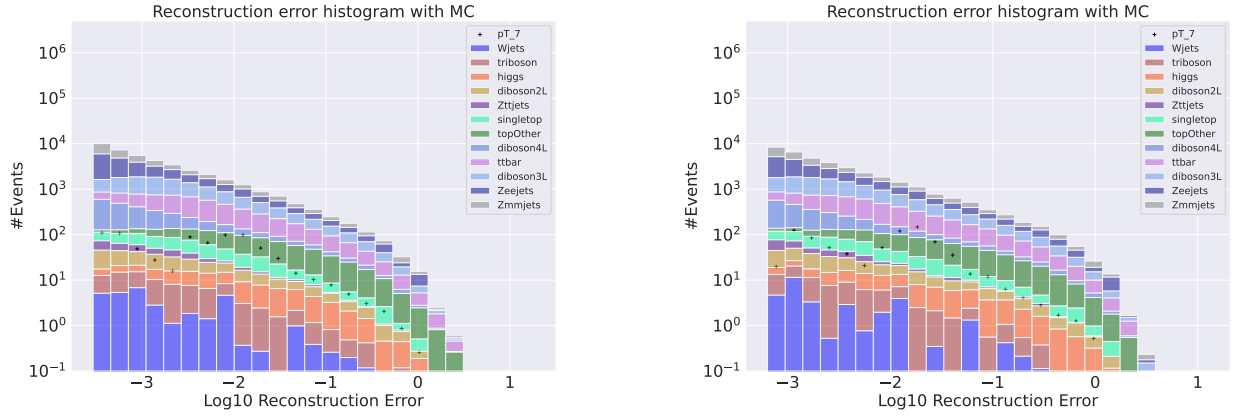
(a) **(b)**

Figure 4.8: Reconstruction error on validation SM MC from the small (left) and large (right) regular Autoencoder. Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 3. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.



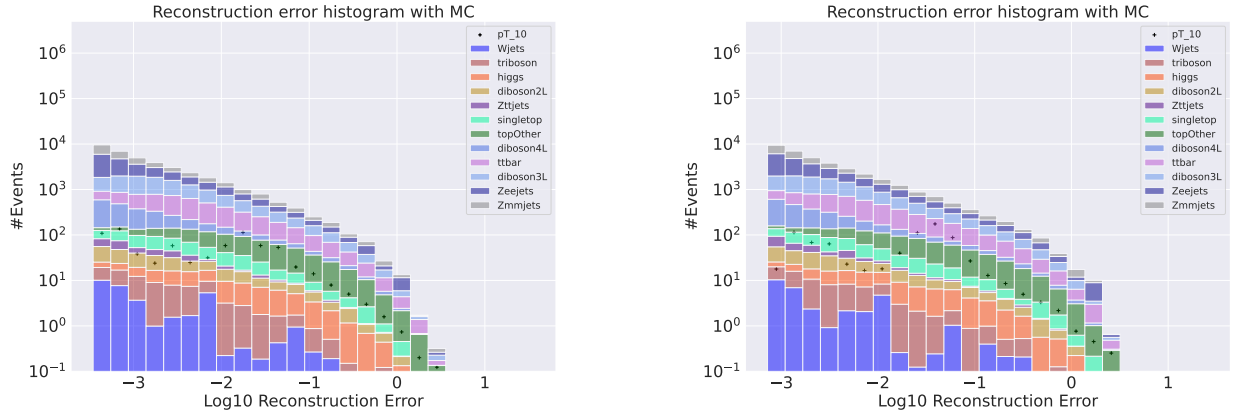
(a) **(b)**

Figure 4.9: Reconstruction error on validation SM MC from the small (left) and large (right) regular Autoencoder. Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 5. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.



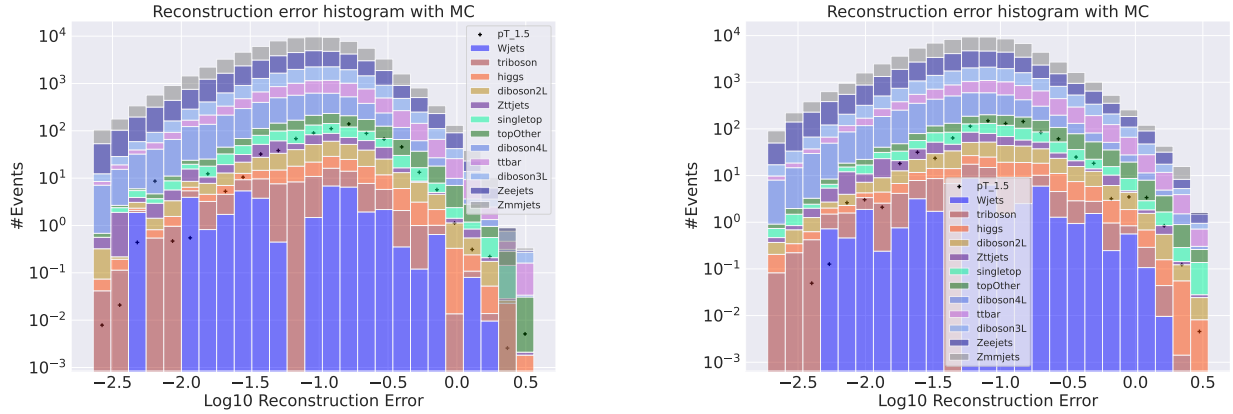
(a) **(b)**

Figure 4.10: Reconstruction error on validation SM MC from the small (left) and large (right) regular Autoencoder. Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 7. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.



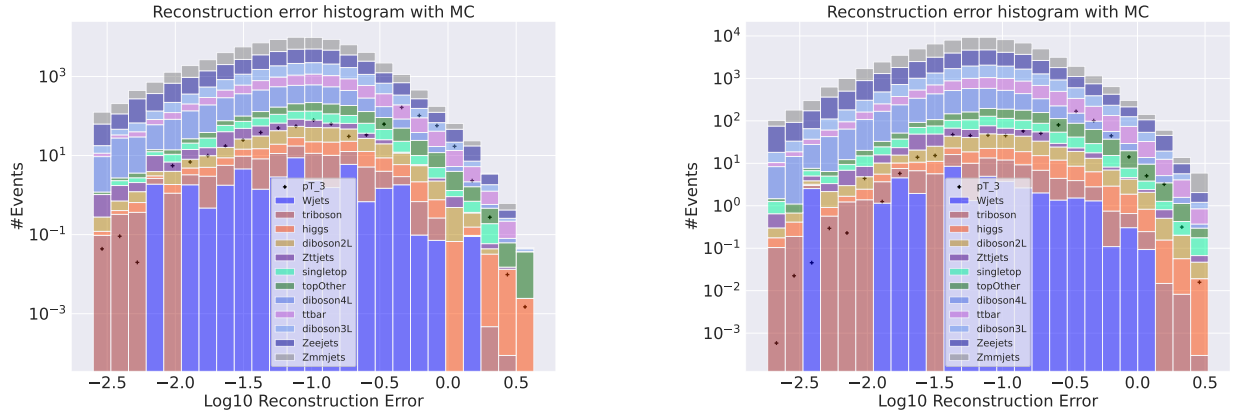
(a) **(b)**

Figure 4.11: Reconstruction error on validation SM MC from the small (left) and large (right) regular Autoencoder. Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 10. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.



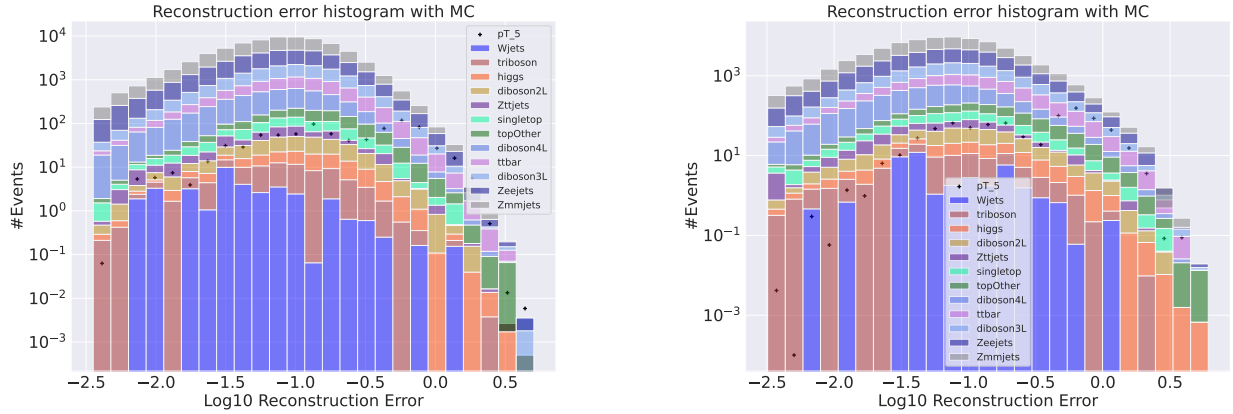
(a) **(b)**

Figure 4.12: Reconstruction error on validation SM MC from the small (left) and large (right) variational Autoencoder. Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 1.5. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.



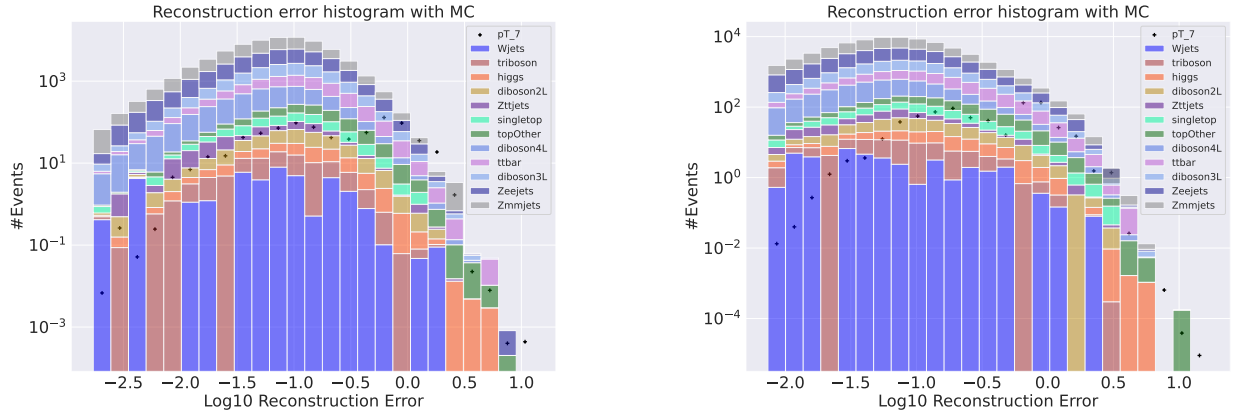
(a) **(b)**

Figure 4.13: Reconstruction error on validation SM MC from the small (left) and large (right) variational Autoencoder. Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 3. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.



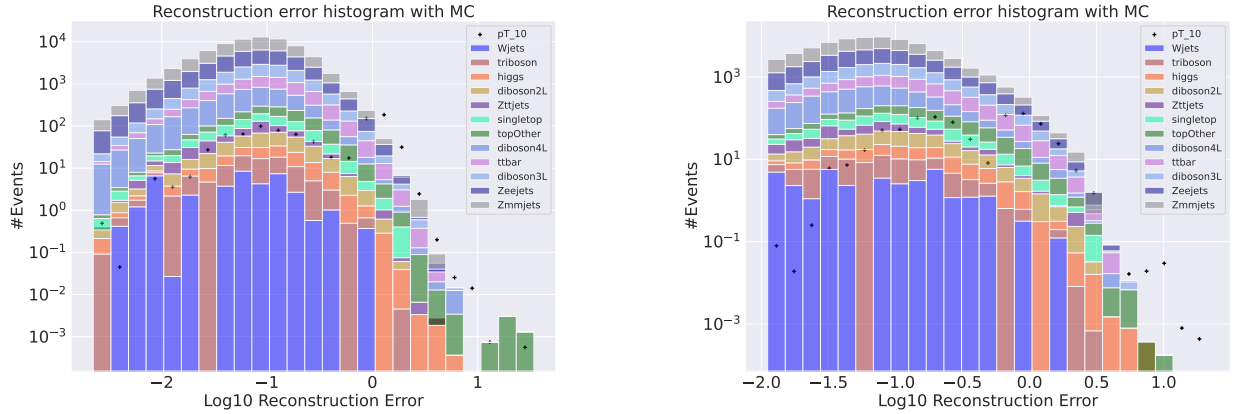
(a) **(b)**

Figure 4.14: Reconstruction error on validation SM MC from the small (left) and large (right) variational Autoencoder.Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 5. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.



(a) **(b)**

Figure 4.15: Reconstruction error on validation SM MC from the small (left) and large (right) variational Autoencoder. Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 7. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.



(a) **(b)**

Figure 4.16: Reconstruction error on validation SM MC from the small (left) and large (right) variational Autoencoder. Here the signal is a subsample of the validation set where the transverse momentum of the first electron and the first muon has been increased with a scale of 10. The change of transverse energy has thusly also been changed according to the scaling of transverse momentum. No significant difference in distributions is found.

The first thing to notice here is that, as with the channel removal, the distribution shape of the output from the regular autoencoder compared to the variational autoencoder is different. The shape from the regular autoencoder is clearly shifted to the left end, with low reconstruction error compared to the shape from the variational autoencoder, where the peak of the distribution is around 1. From the p_T altering test we also see here that the variational autoencoder falls short to the regular one. It is also interesting to observe here that the high p_T events are picked up as anomalous from both autoencoders, with both shallow and deep structure. In the regular autoencoder it is arguably even separated distributions. This is also a good sign as it indicates that the network has learned which ranges the p_T should be in for the 3 lepton plus missing energy final state. The most extreme case is as expected in the case where the p_T is multiplied by 10, and the δe_T^{10}, there is a clear separation in reconstruction error distributions for the signal and background for the deep regular autoencoder, and a slightly more subtle separation of reconstruction error distributions for the shallow regular autoencoder. In the variational autoencoder, the separation is not as clear, but the signal is still picked up as anomalous. The peaks are separated, but not to the same degree.

4.2 3 lepton training for bump search testing

To better accertain and map the usefulness of the regular and variational autoencoder in the search for new physics, two MonteCarlo signals were used as test cases. They both are 3 lep $+e_T^{miss}$ final state SUSY signals, and thus are fit to use for testing. The two autoencoders will be tested on four metrics.

- Low reconstruction error on SM MC
- Background to signal ratio in e_T^{miss} signal region
- Possible resonance in trilepton signal region
- Significance search in e_T^{miss} signal region

Regular autoencoder output

Below are the reconstruction error distributions for the two signals for both the shallow and deep regular autoencoder.

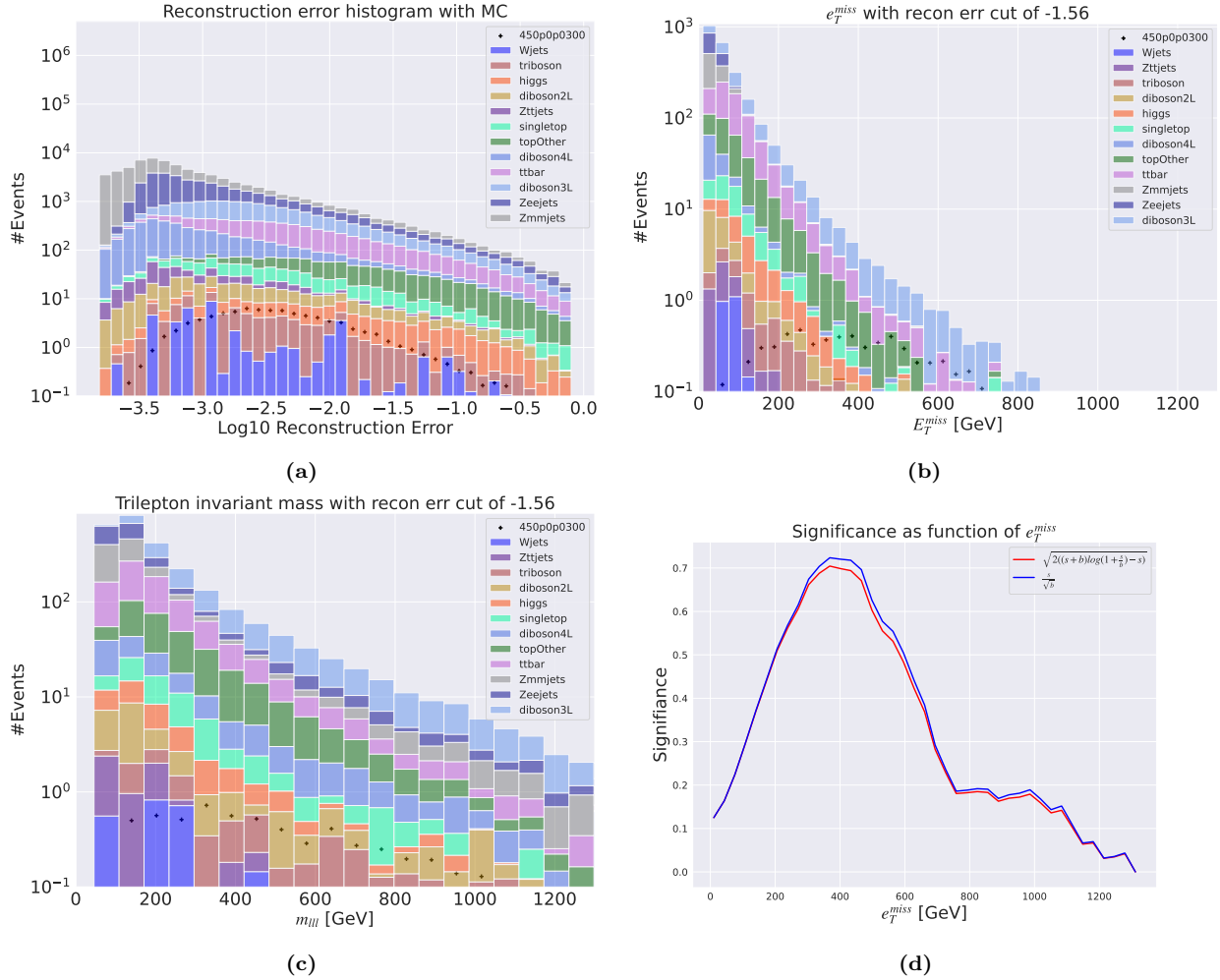


Figure 4.17: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.

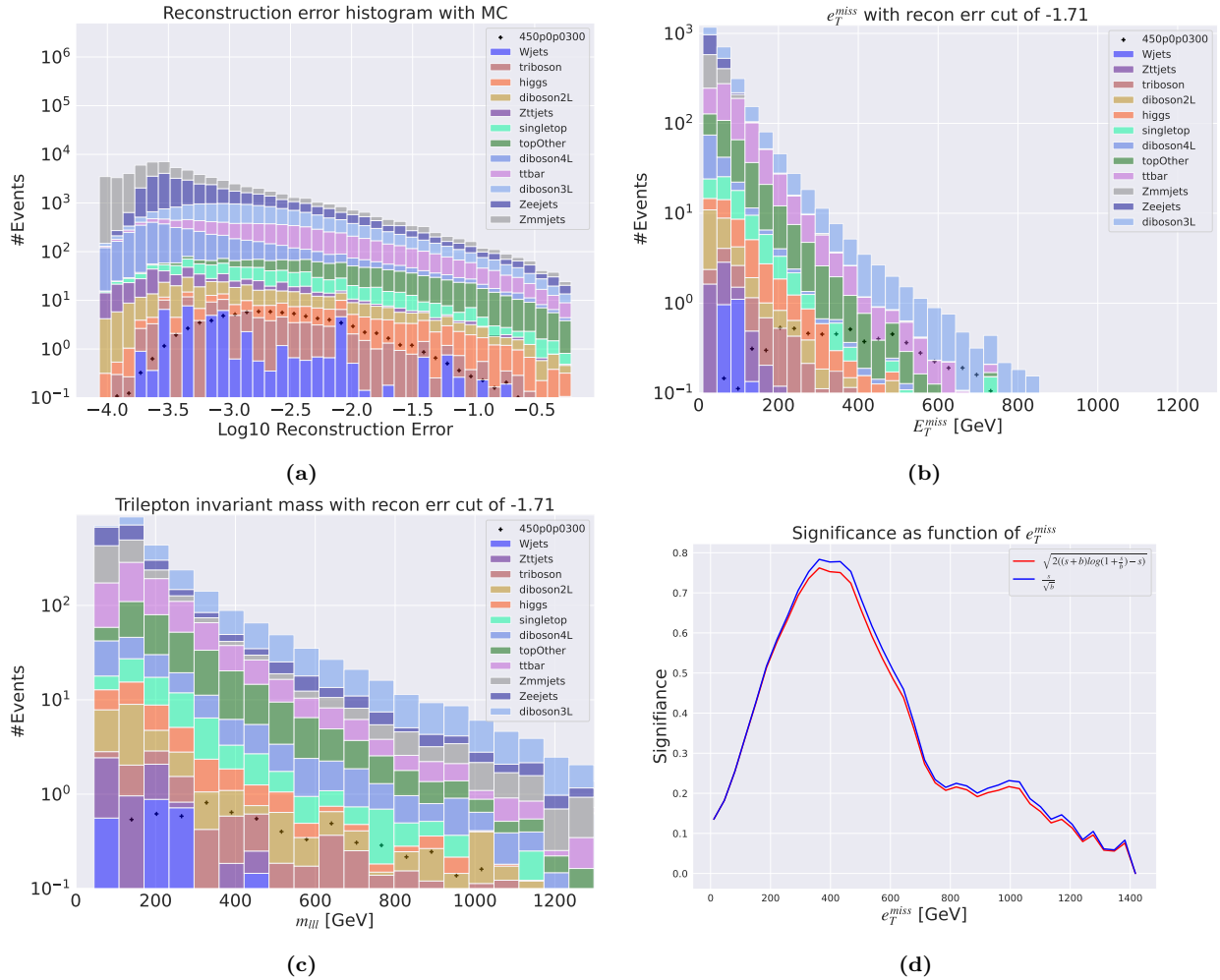


Figure 4.18: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 450p300 model is used.

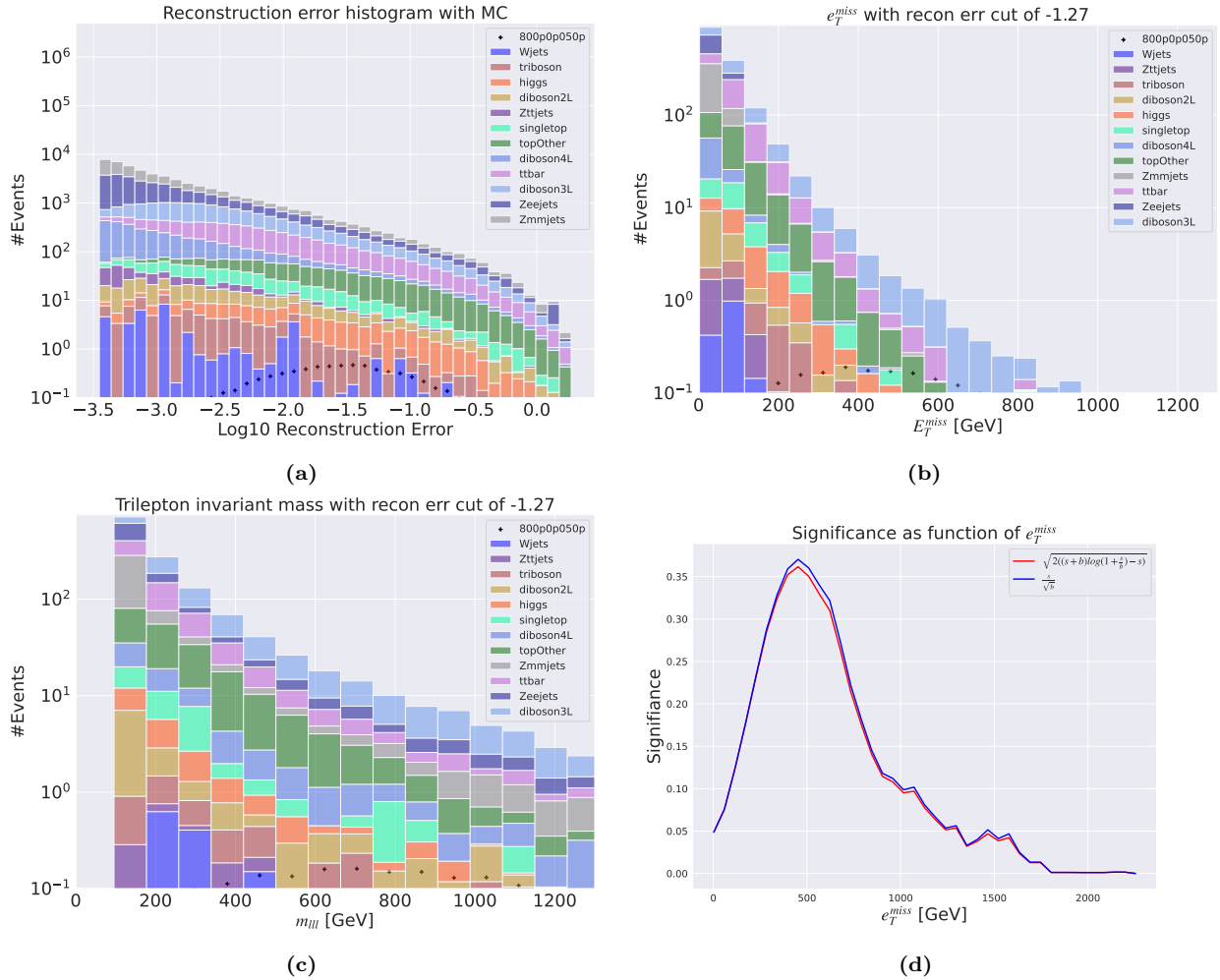


Figure 4.19: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.

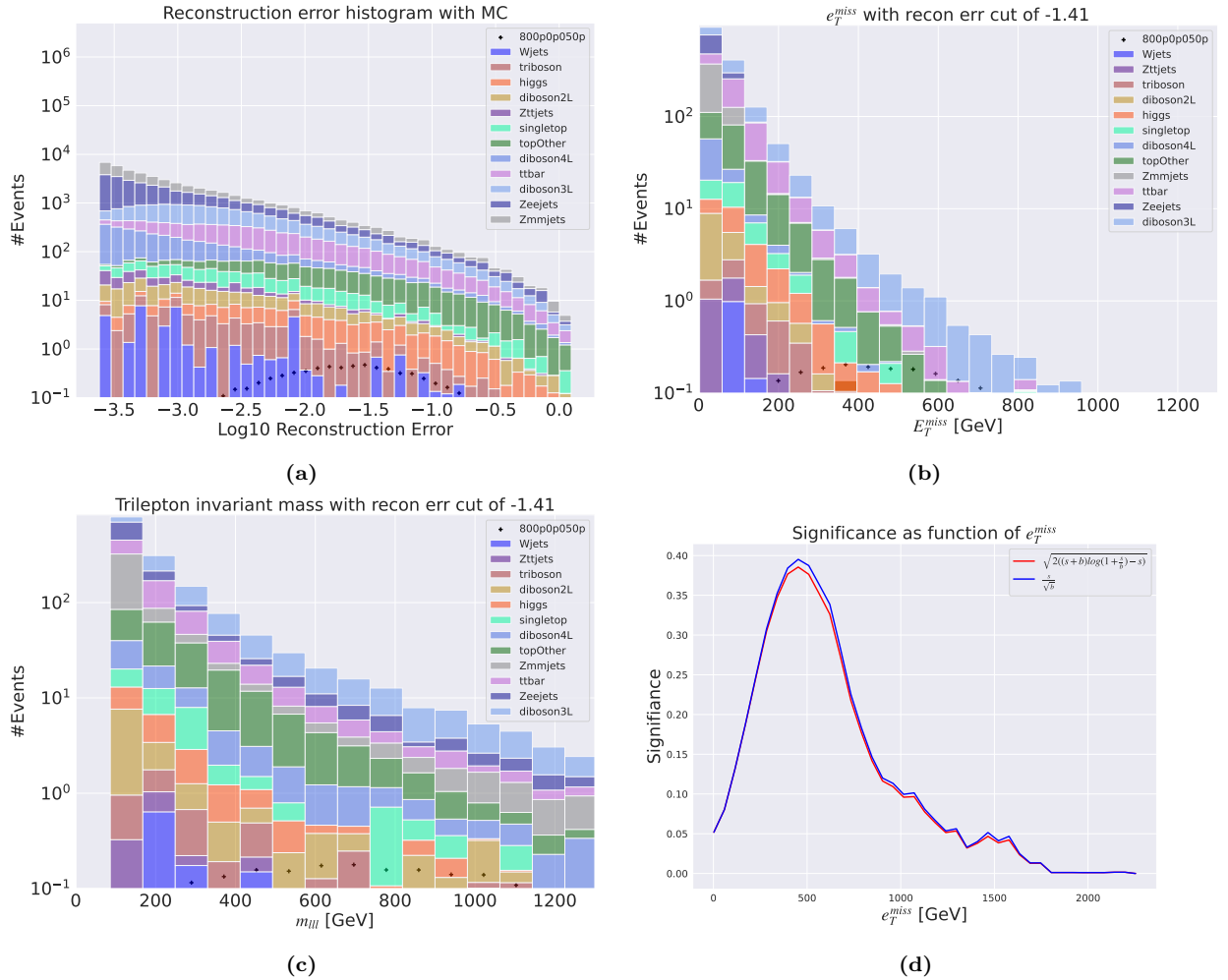


Figure 4.20: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 450p300 model is used.

In figures 4.17, 4.18, 4.19 and 4.20 we have four subplots containing the total reconstruction error distributions, the e_T^{miss} signal region, the m_{lll} signal region and the significance as function of e_T^{miss} curve respectively for the shallow and deep regular autoencoder. From figures 4.17a, 4.18a, 4.19a, 4.20a it is clear here that the autoencoder manages to somewhat separate the signals from the background. The SUSY 800p50 signals has the best separation of peaks, which is expected considering its rather extreme tail in the e_T^{miss} distribution of the signal. It is also interesting to obeserve the slope trend in the SM MC reconstruciton error distribution. The steepness of the slope seems to be somewhat similar for both the shallow and deep regular autoencoder, indicating at least that the difference in these two models do not differ much when it comes to performance. The bulk of the events are below 10^{-2} reconstruction error, indicating that the autoencoder has learned a lot of the internal RMM structures for the 3 lepton + e_T^{miss} final state MC.

In figures 4.17b, 4.18b, 4.19b and 4.20b we have a signal region created for each of the SUSY models from both the shallow and deep regular autoencoder. The cuts were created using the median and then iteratively increasing the error requirement, as explained in section 3.7. Only one of the three cuts done are shown here, the rest can be found in the appendix. The histograms in figures 4.17b, 4.18b, 4.19b and 4.20b displays the e_T^{miss} distribution in the signal region for the SM MC and the SUSY signals. They represent the signal region with the most relaxed cut, in other words, the least strict and thus the signal region with the most amount of total events, both SM MC and signal. This is to illustrate the difficulty of this method since we only have the reconstruction error to go on. Thus, too strict a cut will most likely eliminate all possible signal, whereas too loose a cut and the SM MC background will dominate completely. As the final state signal has 3 leptons, it was thought that there might be a resonance[20] in the trilepton (m_{lll}) distributions in the signal region. Thus, the figure below diplays the m_{lll} distributions for both the shallow and deep regular autoencoder for both SUSY signals.

[20]A resonance in this context means a peak that is significantly different from the distribution it is found in, for example the Z mass resonance around 91 GeV shown in a dilepton mass histogram.

In figures 4.17c, 4.18c, 4.19c and 4.20c we have the m_{lll} distributions in the signal regions from both the shallow and deep regular autoencoder for both SUSY signals. As with the e_T^{miss} distributions, the m_{lll} distributions with the least strickt cut in the signal region are shown here, with the rest shown in the appendix. From these figures it is not easy to see any resonances in the distributions. There is also little separation between signal and background distributions in the m_{lll} feature, allthough there is some for the SUSY 800p50 model. It is also shown that the difference between the shallow and deep neural network architecture has little to no effect here or in the e_T^{miss} distributions.

In figures 4.17d, 4.18d, 4.19d and 4.20d we have the significance as a function of the e_T^{miss}. It displays that a further cut on the e_T^{miss} will get an even better significance, the best case provided from the SUSY 450p300 signal with the deep autoencoder, using a cut on the e_T^{miss} around 400 GeV.

Variational autoencoder output

In the Variational autoencoder case, we have the same distributions as above. Below we have the full reconstruction error distributions for both signals from the shallow and deep variational autoencoder.

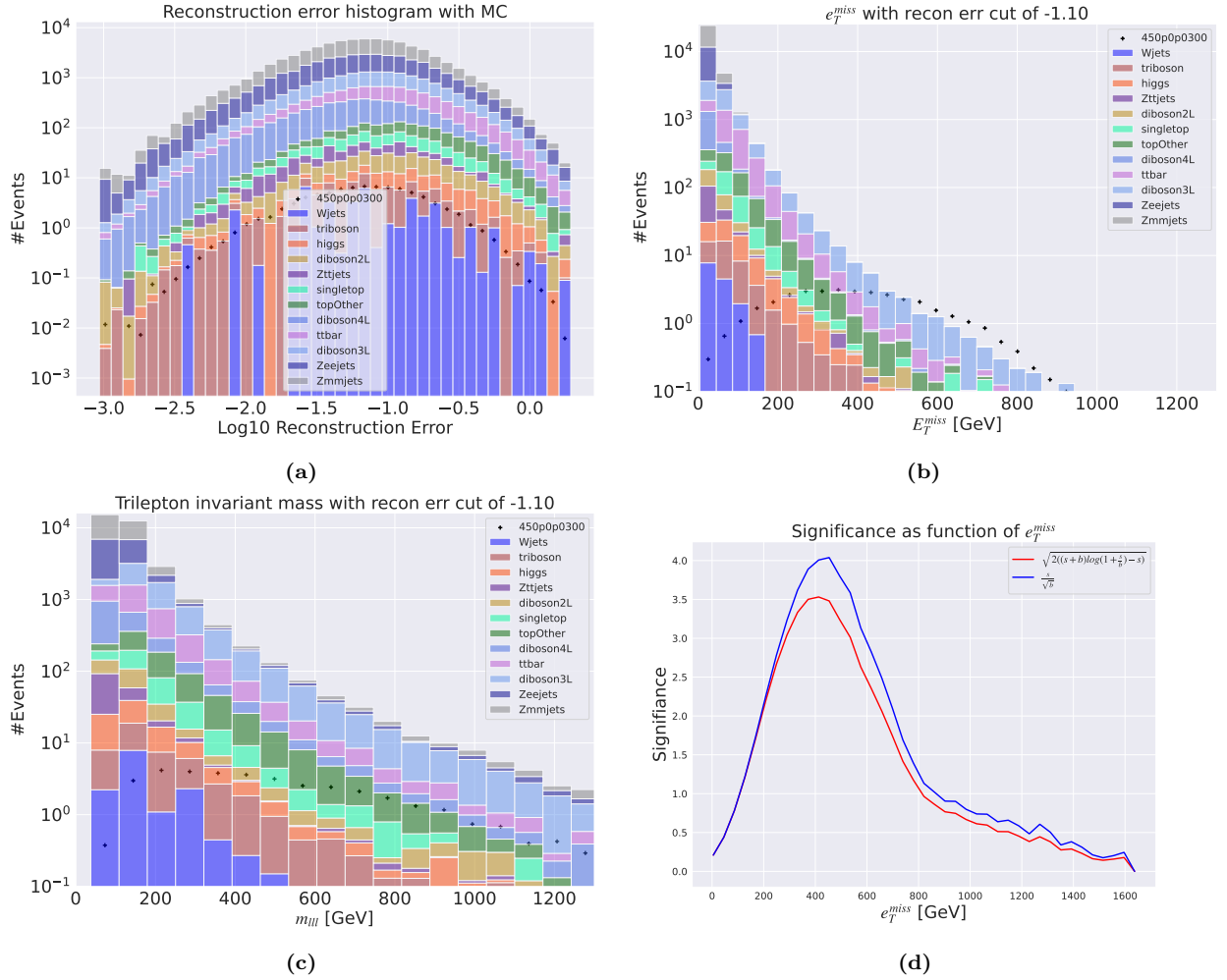
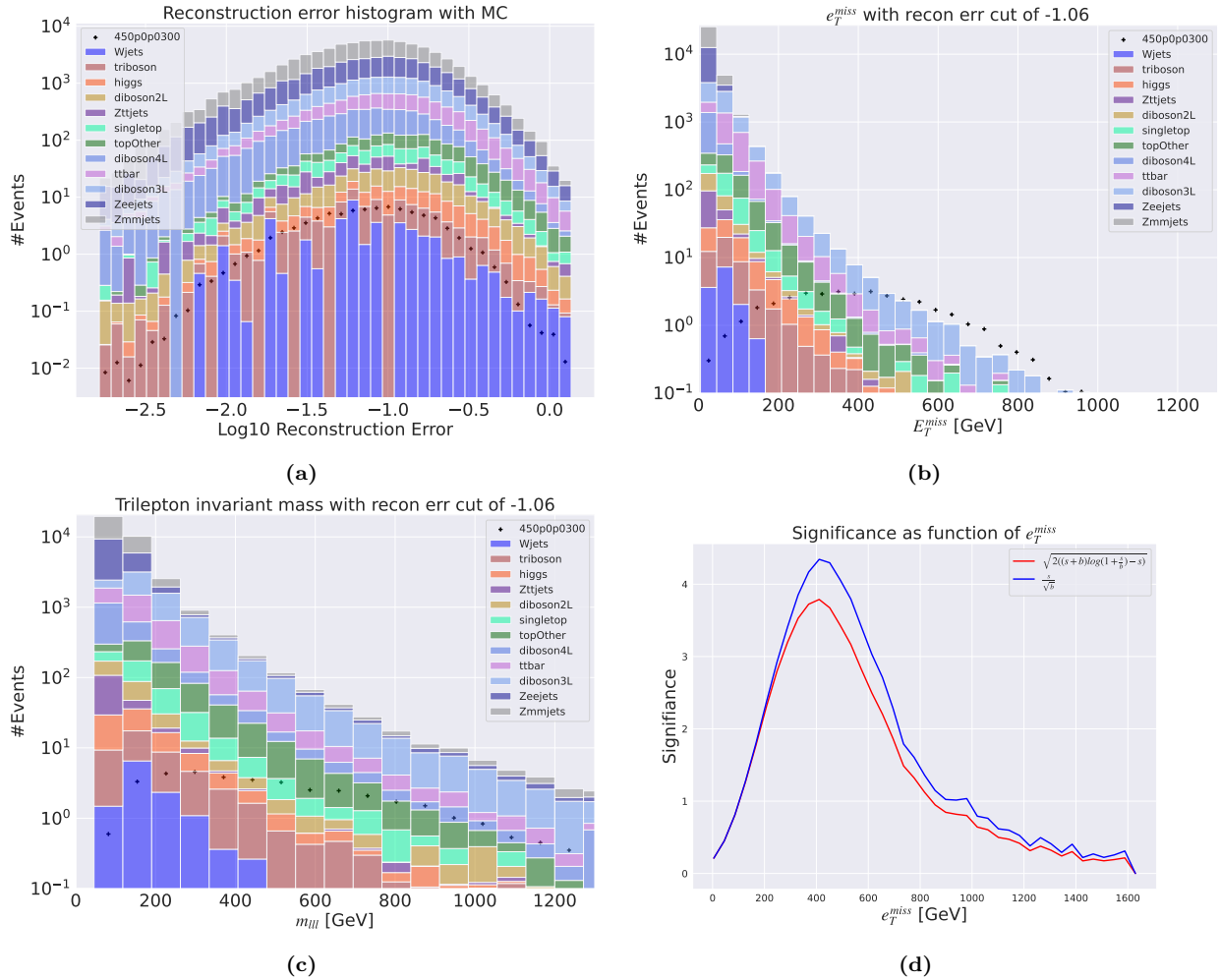


Figure 4.21: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep variational autoencoder. Here the SUSY 450p300 model is used.



(a) **(b)** **(c)** **(d)**

Figure 4.22: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow variational autoencoder. Here the SUSY 450p300 model is used.

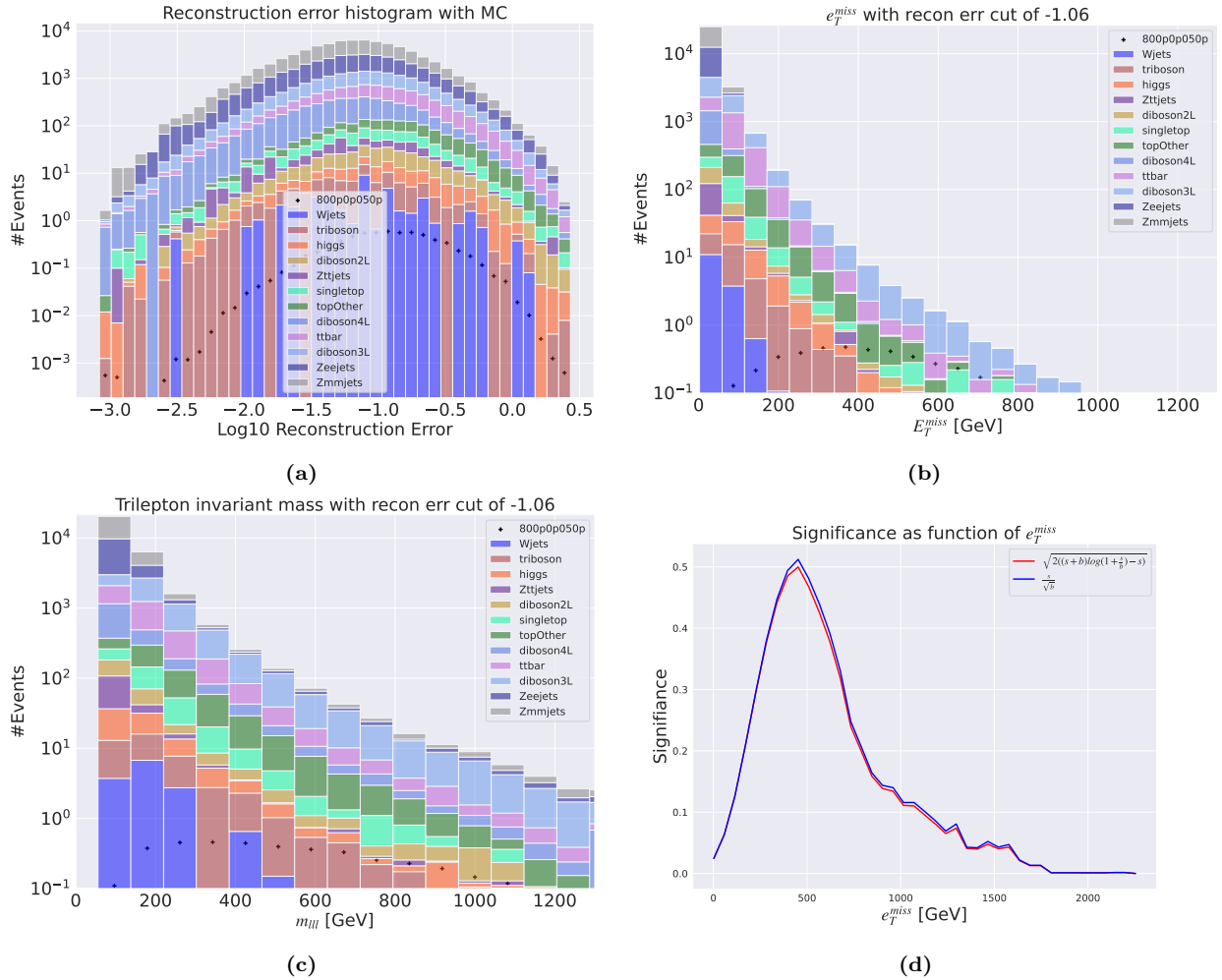


Figure 4.23: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep variational autoencoder. Here the SUSY 450p300 model is used.

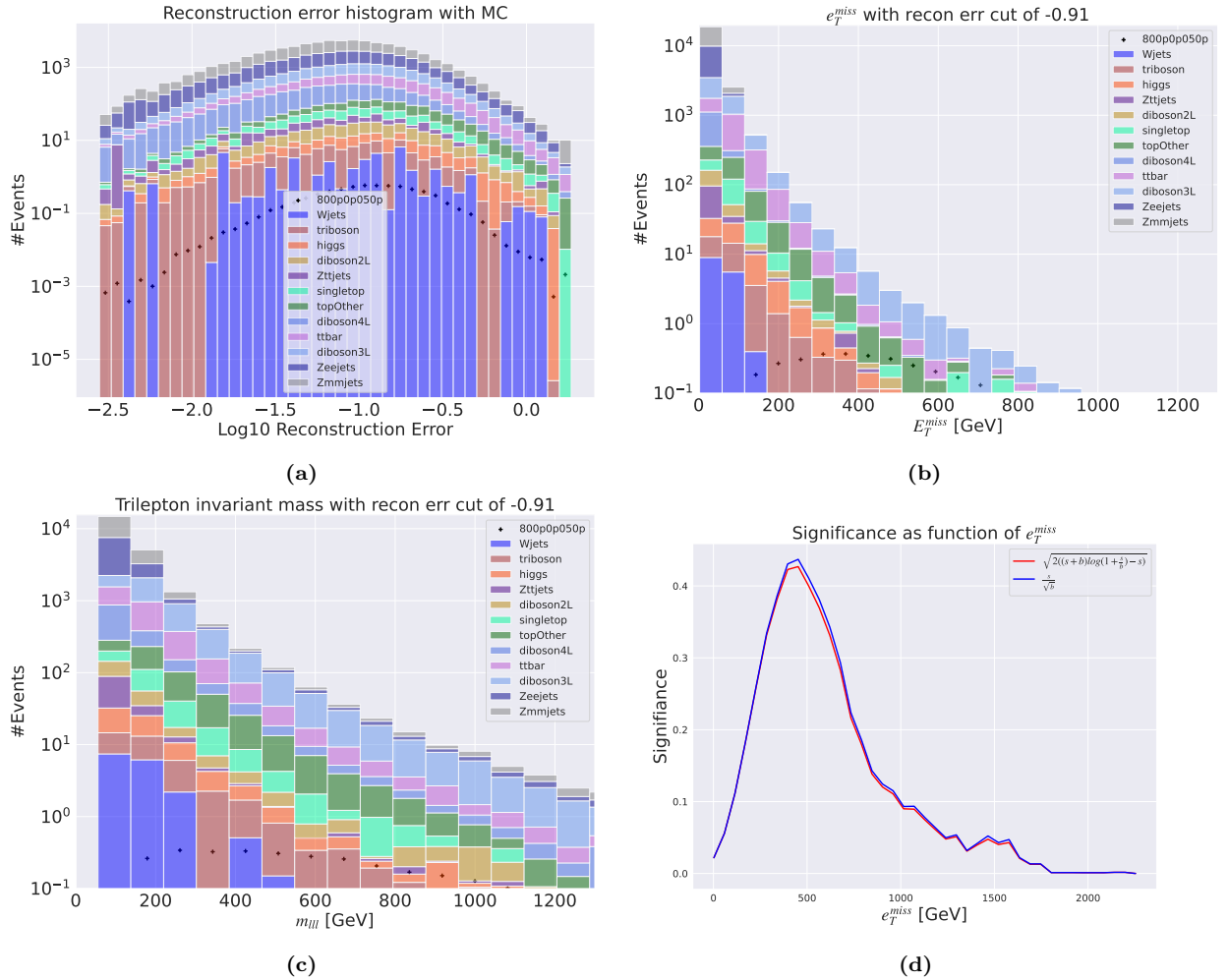


Figure 4.24: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow variational autoencoder. Here the SUSY 450p300 model is used.

In figures 4.21, 4.22, 4.23 and 4.24 we have four subplots containing the total reconstruction error distributions, the e_T^{miss} signal region, the m_{lll} signal region and the significance as function of e_T^{miss} curve respectively for the shallow and deep variational autoencoder. From figures 4.21a, 4.22a, 4.23a, 4.24a we observe that the variational autoencoder seems to struggle with differentiating between background and signal, having the two distributions laying on top of each other. There is however a slight difference in the shape of the distributions from the shallow and deep network. The shallow has a more narrow and slightly more left leaning shale, whereas the deep network has a slightly more broad distribution shifted a bit to the right. The bulk of the events are between 10^{-2} and $10^{-0.5}$ reconstruction error, indicating that the autoencoder struggles to learn the internal RMM structures for the 3 lepton + e_T^{miss} final state MC. The somewhat Gaussian distribution differs between the two SUSY samples. As the networks were retrained each time, it could indicate that the distribution in part is due to the sampling in the latent space.

In figures 4.21b, 4.22b, 4.23b and 4.24b we have the reconstruction error cut imposed on the SM MC and the signal samples. Interestingly, one should note that although the total reconstruction error distributions are not well separated, the signal regions show a separation in distribution peaks. Now, unlike in the regular autoencoder case, the variational autoencoder allows for almost two orders of magnitude more background events in the signal region. This is because of the algorithm used to determine the reconstruction error cut by means of finding the median and iteratively increasing the criterion. Bacause the two distributions are on top of each other, we do get a lot of background events, but we also get the peak of the signal distributions, which is why we see the clear separation in the subfigures 4.21b, 4.22b, 4.23b and 4.24b.

In figures 4.21c, 4.22c, 4.23c and 4.24c we have the signal region in the m_{lll} feature. Here, as with the e_T^{miss} distribution, the least strict reconstruction error cut has been imposed on the region. There are little to no separation between the peaks to observe.

In figures 4.21d, 4.22d, 4.23d and 4.24d we have the significance as a function of the e_T^{miss}. Interestingly, for the SUSY 450p300 case, we have that both the deep and shallow autoencoder manages to get a significance

of around 4, which is much better than the regular autoencoder which got around 0.7. Likewise for the other SUSY signal, but here the difference was not so impressive. From these figures a further cut on the e_T^{miss} on around 400 GeV provides the best significance. Also note the tails of the significance figures, as they all contain little background and or signal and are thus not of interest.

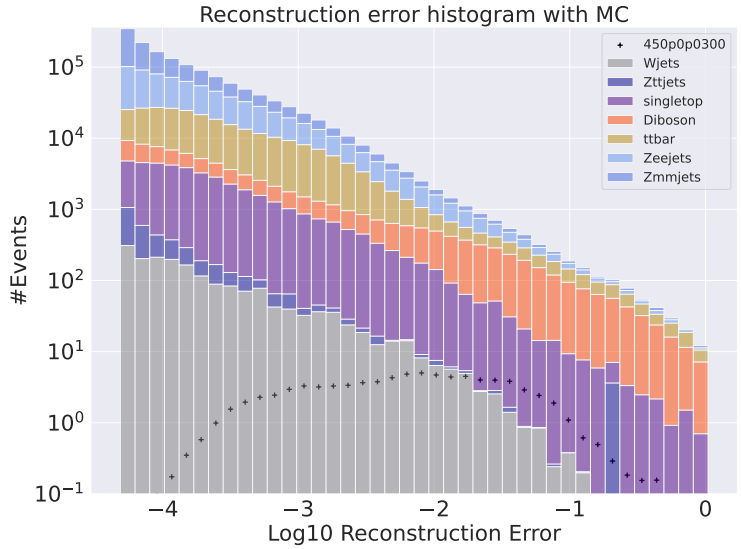
4.3 2 lepton training for bump search testing

The 3 lepton + e_T^{miss} dataset has fewer events, and thus allows for less training of the neural networks. Thus, the 2 lepton + e_T^{miss} dataset was tried as well. The event selection was done choosing atleast 2 leptons, meaning that the RMM signatures of some events will look similar to the RMM signatures of the 3 lepton + e_T^{miss} dataset. A consequence of this is that the signal samples for the 3 lepton case would be a good start point for testing. The two autoencoders will be tested on three of the four metrics used for the 3 lepton + e_T^{miss} case.

- Low reconstruction error on SM MC
- Background to signal ratio in e_T^{miss} signal region
- Significance search in e_T^{miss} signal region

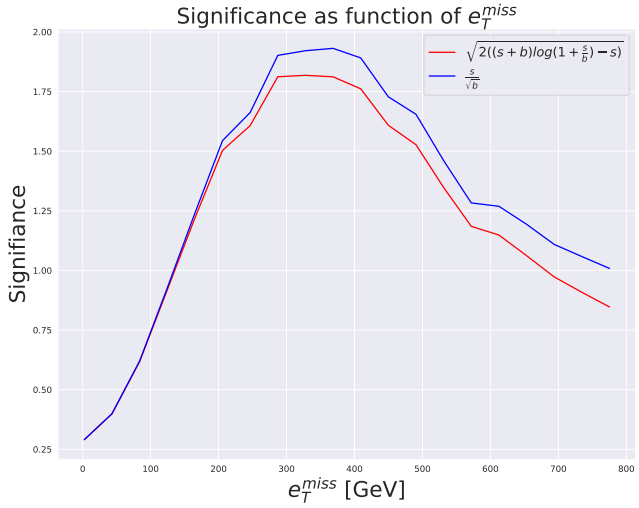
Regular autoencoder performance

Below are some results from training on the 2 lepton case with the regular autoencoder, using the same two SUSY signals as test cases.



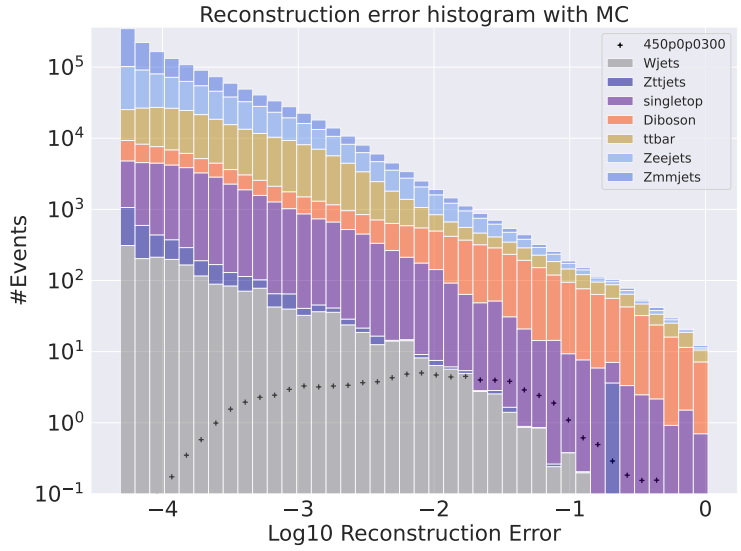
(a)

(b)

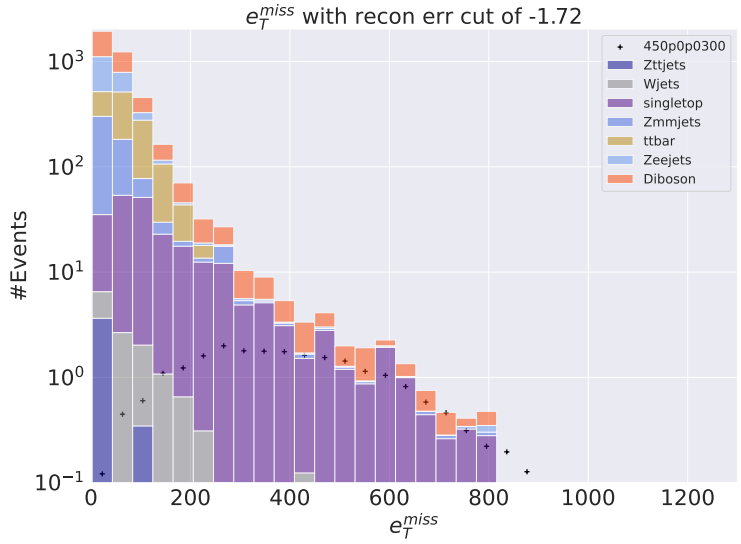


(c)

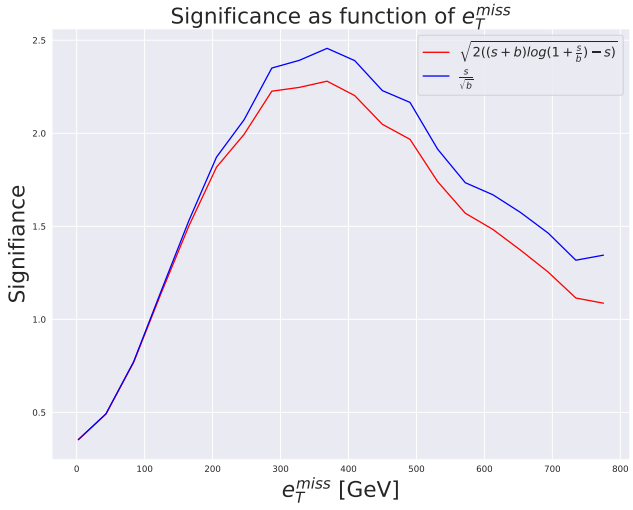
Figure 4.25: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a)

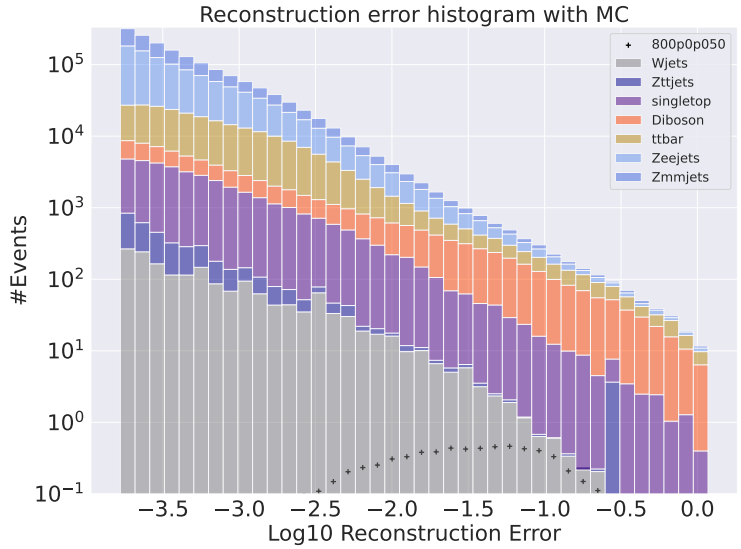


(b)

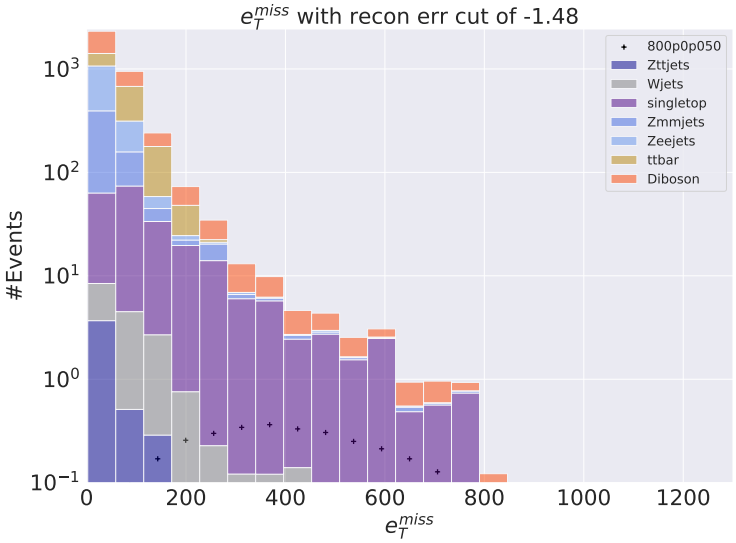


(c)

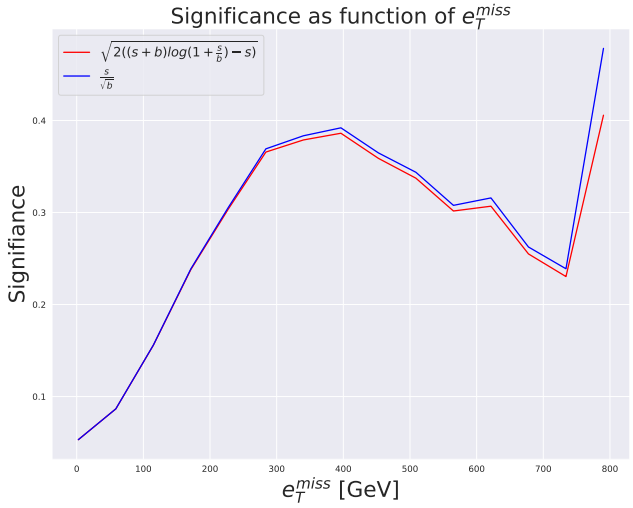
Figure 4.26: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a)

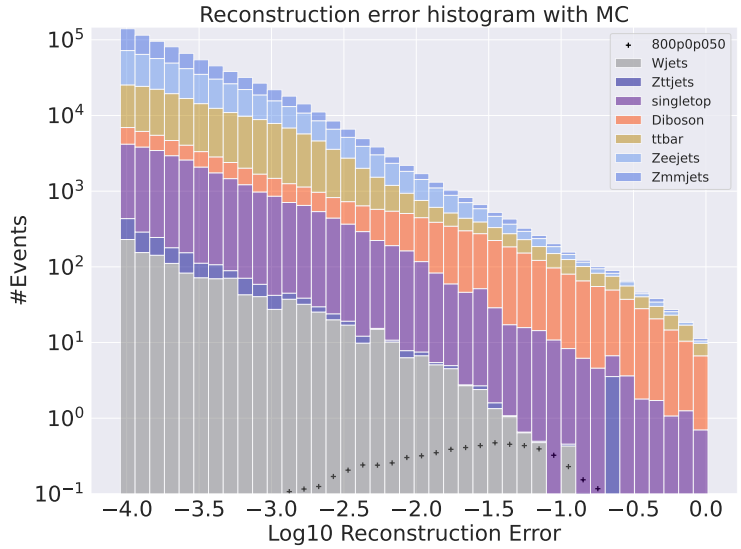


(b)

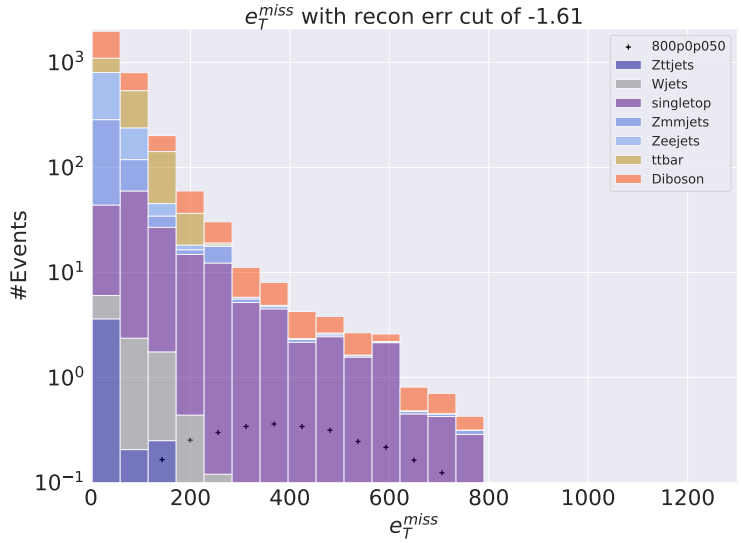


(c)

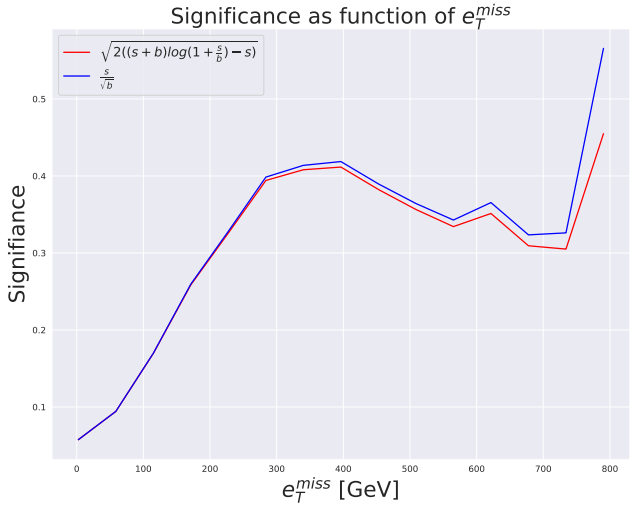
Figure 4.27: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY $800p50$ model is used.



(a)



(b)



(c)

Figure 4.28: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 800p50 model is used.

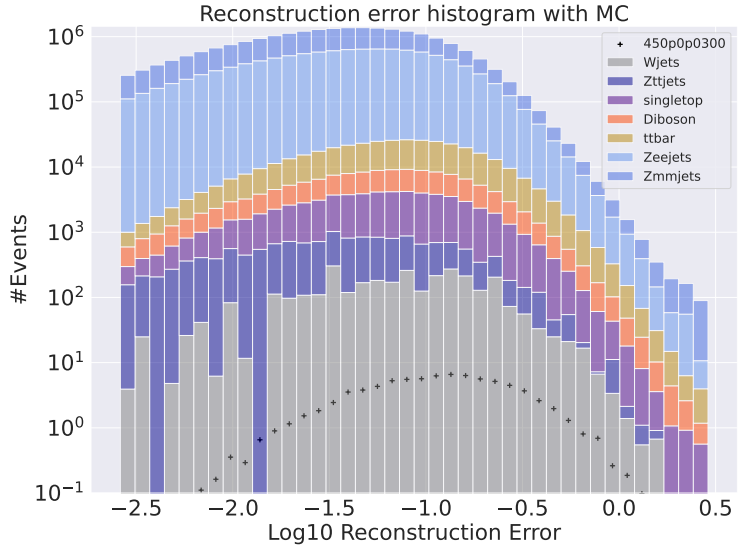
In figures 4.25, 4.26, 4.27 and 4.28 we have three subplots containing the total reconstruction error distributions, the e_T^{miss} signal region, and the significance as function of e_T^{miss} curve respectively. They were created using the shallow and deep regular autoencoder with the 2 lepton + e_T^{miss} dataset. In figures 4.25a, 4.26a, 4.27a, 4.28a we observe that there is a general trend here for both the small and large regular autoencoder where the background distributions are highly shifted to the lower end of the reconstruction error range. This is a continuing trend following the 3 lepton + e_T^{miss} shown in figures 4.25a, 4.18a, 4.19a, 4.20a. This indicates that as we increase the statistics, in other words the amount of background events used for training, the ability of the autoencoder to learn the internal structure increases. It is also interesting to observe here that as the reconstruction error increases, the amount of each sample in a given bin changes alot. As expected, Zmmjets and Zeejets along with ttbar are the events with the highest statistics in the 2 lepton + e_T^{miss} dataset, thus it should be easier to learn to better reconstruct those events. However, note the amount of Diboson in the higher end of the reconstruction error histograms, as well as in the e_T^{miss} post reconstruction error cut distributions. One explanation for this could be the sample discrepency explained in section 3.4. It would appear that too much of the diboson samples is cut, leading to not enough samples for it to appropriatly learn the RMM structure for that channel, leading to the high reconstruction error values for some of those events. Note also here that the reconstruction threshold where this effect is getting more notisable around 10^{-2} and larger, where the events are on the order of a 1000 or less per bin. Thus, it is not alot of events, but still enough to note.

Using the signal region definition from above we set cuts on the reconstruction error and then calculate the significance of the signal.

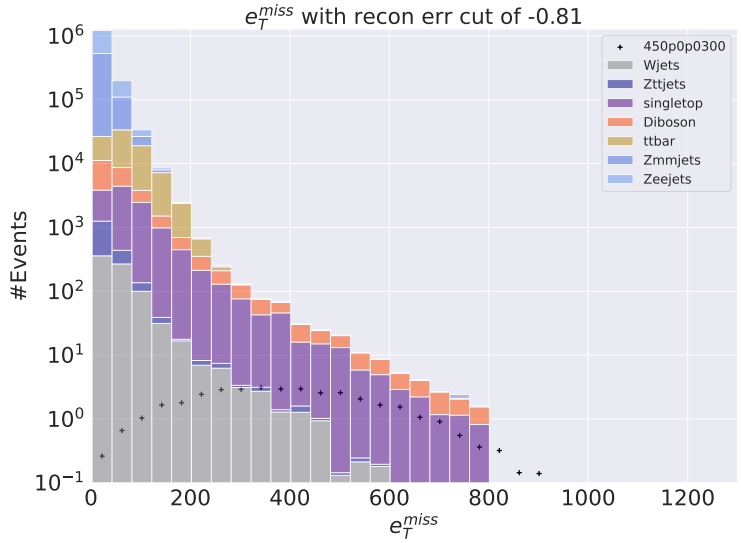
In figures 4.25b, 4.26b, 4.27b and 4.28b we have the e_T^{miss} distributions for the least strict cuts for the regular autoencoder models. Now, one indication that the search strategy used in this thesis could work is if the models can improve the significance from just looking at the e_T^{miss} distributions of the background and the signal in mind. Thus, we want to compare the pre reconstruction error significance with the signal region based on the autoencoder output. The significance in the pre reconstruction error case were for the SUSY 450p300 signal 0.017 using both the small and large statistics formula, and 0.0014 for the SUSY 800p50 signal. Note here that for e_T^{miss}, no physics informed cuts have been used, but there are certain cuts that can increase the significance, so the significance in should be considered to possibly be somewhat higher.

From figures 4.25c, 4.26c, 4.27c and 4.28c we have the significance as a function of e_T^{miss} for both signals using the shallow and deep autoencoder. This graph shows how implementing another cut in the signal region, namely where in the e_T^{miss} distribution to select event, allowss for higher significance. We see here that the shallow autoencoder gets a significance of almost 2.5 in the SUSY 450p300 case, being the highest score, and beating the pre reconstruction error significance by alot. Another interesting point to note here is the tail ends, especially for the SUSY 800p50 signal. This model has some events of very high e_T^{miss}, and in the tail end, there might be very few or none background events. It should be noted that the significance for the 800p50 signal model is a lot smaller than for the 450p300 signal model, even though the separation shown in figures 4.25a, 4.26a, 4.27a, 4.28a would suggest otherwise. Although the peak in both signal models are fairly separated from the peak of the SM MC, the SUSY 800p50 signal model is shifted a lot more. This is consistant with expectation, for the same reason that the significance tails are the way they are. Now, The reason for the low significance is most likely the fact that the signal sample contains low statistics, in other words, the weights are just a lot lower, compared to the other signal sample.

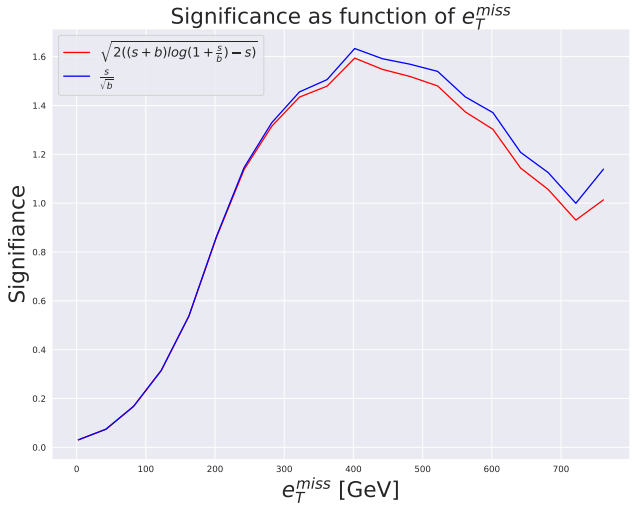
Variational autoencoder performance



(a)

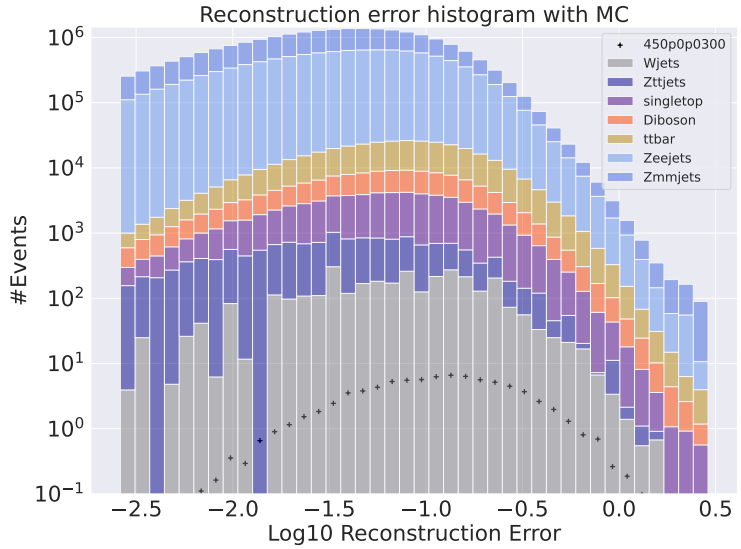


(b)

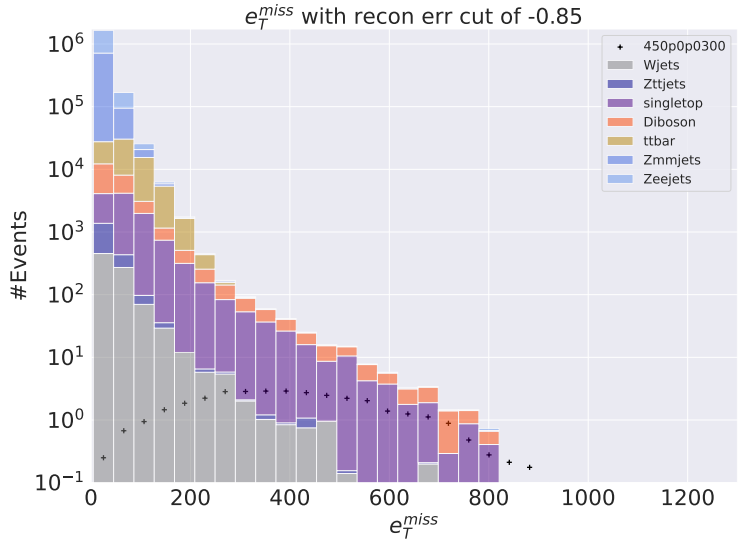


(c)

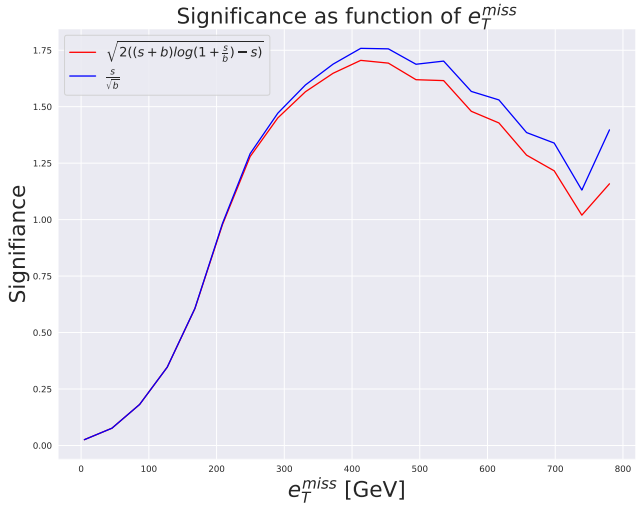
Figure 4.29: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a)

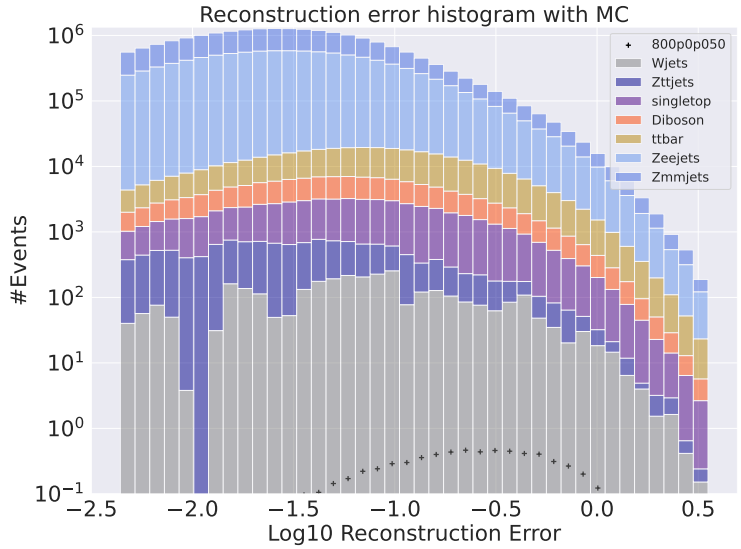


(b)

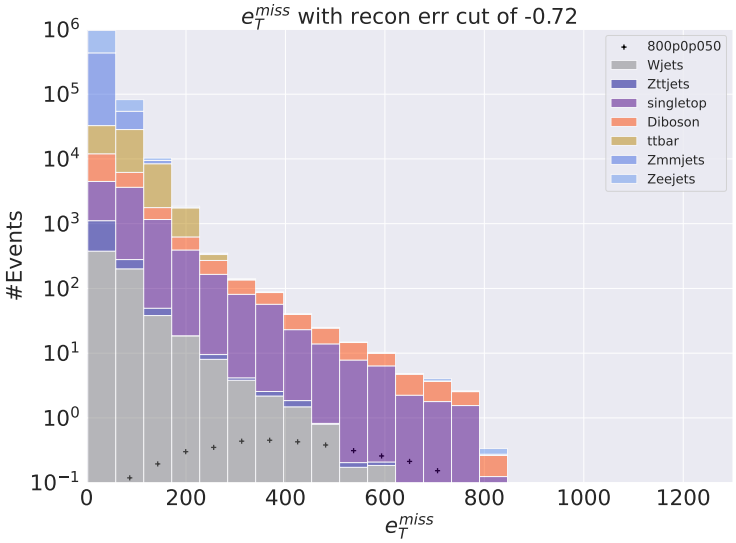


(c)

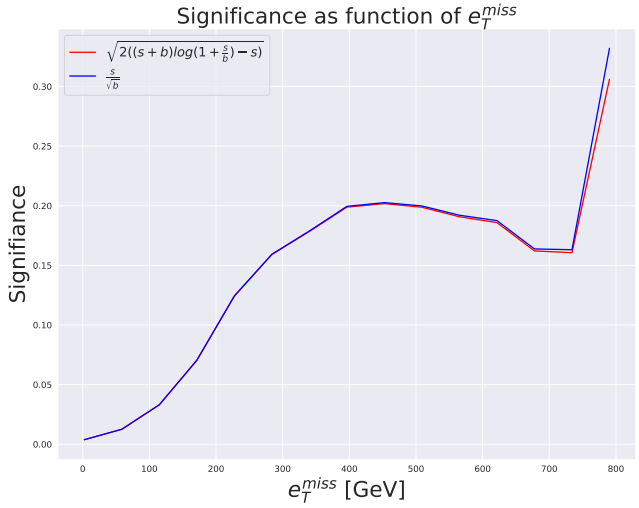
Figure 4.30: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 450p300 model is used.



(a)

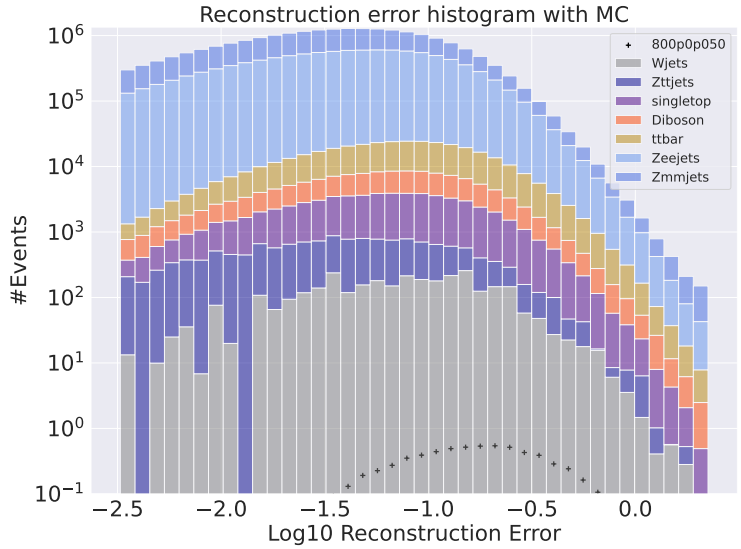


(b)

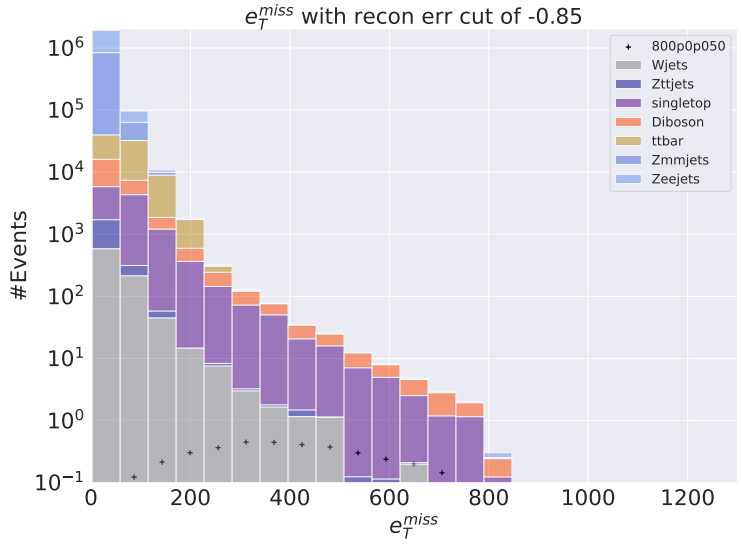


(c)

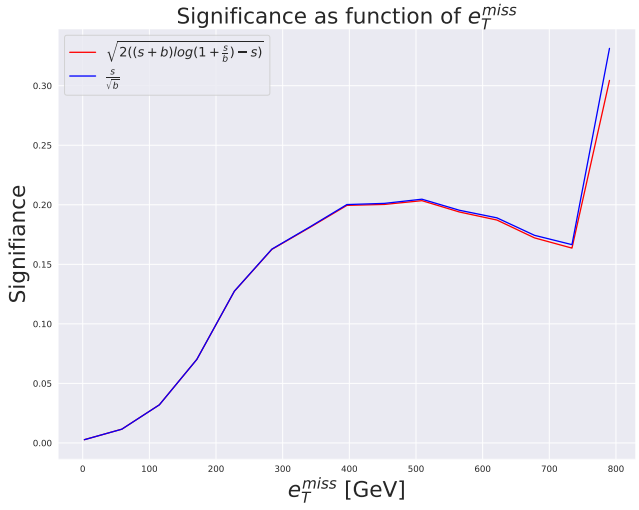
Figure 4.31: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep variational autoencoder. Here the SUSY 800p50 model is used.



(a)



(b)



(c)

Figure 4.32: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the shallow variational autoencoder. Here the SUSY 800p50 model is used.

In figures 4.29, 4.30, 4.31 and 4.32 we have three subplots containing the total reconstruction error distributions, the e_T^{miss} signal region, and the significance as function of e_T^{miss} curve respectively. They were created using the shallow and deep regular autoencoder with the 2 lepton + e_T^{miss} dataset. In figures 4.29a, 4.30a, 4.31a, 4.32a we have the reconstruction error distributions for both SUSY signals for the small and large variational autoencoder. Here we observe that the peak of the distributions for the SM MC in all four cases are somewhat centered in the middle of the reconstruction error range, which differs from the steep slope we saw in figures 4.25a, 4.26a, 4.27a, 4.28a. Interestingly, we see here that the deepness of the neural network here plays a role, which is different from the regular autoencoder output, where both the small and large autoencoder made a steep slope shape of the SM MC reconstruction error distribution. The peak of the distribution here is slightly shifted to the left for the shallow autoencoder model, and slightly shifted to the right of the center with the deep autoencoder model. One possible reason for this somewhat Gaussian like distribution could be that the variational autoencoder, via the reparametization trick from section 1.4, samples from a Gaussian distribution that has yet to be trained on enough data to produce a good enough error distribution. This is also supported with the fact that the shape is even more Gaussian like in the 3 lepton + e_T^{miss} case shown in figures 4.21a, 4.22a, 4.23a, 4.24a. It could also be that the batch size is too large, and that the model has to train on smaller batches to get a better result.

In figures 4.29b, 4.30b, 4.31b and 4.32b we have the e_T^{miss} distribution for the least strict cut for each signal. We see that the cuts are somewhat similar to the regular autoencoder, but with two key differences. First, because the peaks of the distributions from figures 4.29a, 4.30a, 4.31a, 4.32a are so close, the cuts allowed for more background events in the signal region. Here, as with the regular autoencoder output, m_{err} was used, but was not a good descriminator for the background events. One could perhaps make a new strategy for setting the cuts in the event one has a "fat gaussian" shape. Still, because we set cuts based on reconstruction error to mimimize the background in the signal region, if one does not have a slope like shape, the results will be poor in comparison.

Secondly, the background that remains are slightly different from the signal region from the regular autoencoder. In the lower energy range there is a large excess of Zeejets, Zmmjets and ttbar events that have a high reconstruction error, which is not the case for the variational autoencoder, dominated by diboson events in all bins. In the higher energy range, diboson are largest contributer to the background, but the number of bins are exceptionally smaller than the Zmmjets/Zeejets/ttbar events, by a magnitude of 3 at the most.

4.4 Final remarks on the results

In sections 4.1, 4.2 and 4.3 the performance of the regular and variational autoencoder with respect to usage in BSM searches has been shown through various tests. The final remarks can be summed up in four points:

- Shape of SM MC reconstruction error distribution
- Dataset altering testing
- Megaset changes
- Network architecture

First, it appears that the shape of the reconstruction error is more sensitive to the choice between variational and regular autoencoder than it is to the amount of training samples it uses. However, the steepness of the shape in the case of the regular autoencoder increases with the increase of training data. From this obervation it is reasonable to assume that to increase the ability of the autoencoder to perform, one needs large amounts of training samples. This is also shown in the case of the variational autoencoder, where if one increases the amount of training data by going from the 3 lepton + e_T^{miss} case to the 2 lepton + e_T^{miss} case we observe the peak to move more to the left end of the reconstruction error distribution. It was not well investigated why the outputs of the two models are so different, but one reason could be that the decoder that sampels from the latent space distribution in the variational autoencoder needs even more training data to get good separation.

Secondly, with the increase in training data it would be interesting to do the "altering p_T" test, as well as some other SM MC altering tests with especially the regular autoencoder. Based on the signal tests and the "altering p_T" test done with the 3 lepton + e_T^{miss} trained regular autoencoder, it is reasonable that the autoencoder would perform even better. Other tests could for example be to swap certain features in the RMM in order to create unphysical events, or swap events from one decay channel with another in order to create unphysical events. Essentially, by making these anomalous events and testing the autoencoder, one would get a better picture on the reach and ability of the autoencoder. This is of importance given the fact

that the target signal or signals could in theory look very strange or perhaps in some feature space very similar to the SM MC.

A third point to note is in regard to the megaset training done in the 2 lepton + e_T^{miss} case. An arbitrary choice of 10 megasets were chosen, but it might be better to choose a larger or smaller number of megasets, and that might have an impact on how well the autoencoders learn the SM signatures. A key criterion to uphold when doing that training is to ensure that the megasets all keep the natural distribution of the entire training set, and thus the standard model. If not, the algorithm will learn with a bias that is not contructive.

The fourth point to draw from the results are that it appears that the results are not too sensitive to the architecture of the networks. The two different architectures chosen were one model with only a latent space layer between the input and output layer, and a model with three layers on each side of the latent space. The choice of layers and nodes per layer was somewhat arbitrary, and after inference it was also clear that really only the latent space was sensitive to change with respect to performance in reconstruction error. This is though more related to the size of the input layer and the complexity of the data, more than the number of events in the training and test set.

4.5 Challenges with the search method and tools

In the previous sections, the output and results of using the autoencoder for anomaly detection have been shown. The method and results, as produced and shown in this thesis, have yielded some promising results given the nature of the search method. However, it should be known what the challenges of the task actually are to truly understand why the results are only promising, and not great compared to other search methods. The challenges can be divided into three main points, all of which are entangled together. The three challenges are listed below.

- Model independence
- Reconstruction error minimization
- Feature engineering

Model independence is the first challenge. By model, it is here referring to signal models, which itself might be self-explanatory, but is an incredibly difficult criteria to uphold. As mentioned in the theory section for the Standard Model, the Standard Model, although very successful in certain predictions, lack the ability to explain a whole number of behavior around us, and so there have been made many suggested solutions to the issues. These new models are often called extensions to the Standard Model, and although mathematically consistent, not neccesarly physically possible. And even if they are physically possible, in other words, they adhere to certain fundamental physical principles, they still might not exsist, as several searches at ATLAS have exluded but never found any new physics. The search method is inherently biased as one assumes that the new physics looks like the signal, and thus do analysis, data preparation etc. with that signal in mind. But we do not know, at all, what the new physics looks like, even the assumption that we are looking for particles are implying a bias that might not be true. From the collisions in the detector to the analysis, there are biased descisions, we cannot avoid them, but we can minimize them as much as possible, which is a goal with the search strategy in this thesis.

This leads us to the second challenge, which is reconstruction error minimization. The proposed method in this thesis is to learn the signature of the standard model so well, that even subtle anomalous behavior will be picked up by the analysis tool. The autoencoder learns the signature of the standard model, by mimimizing the reconstruction error, and then hopefully, the anomalous data will be picked up and skewed to the right end of the reconstructino error distribution in a signal region. One problem with this is that one first blinds oneself to signals that might be very, very similar to the standard model in some feature space, but with very low statistics. These events will for a given set of features, never be found.

The third challenge is the choice of features. This thesis utilized the RMM structure by Chekanov et al., as it maximizes the amount of information in the input data by using almost completely uncorrolated features. However, as we do not know the signals we are looking for, there is no way to know if this choice is the optimal choice for new physics. In fact, even if we found an ideal set of features, based on some physical principles or something else, it is not trivial that the reconstruction error calculation should weight the error of each feature equally. It might very well be that some features are less important than others. Essentially, the goal is to optimize for a signal we do not know, using features we don't know are optimal, and weighting them as "unbiased" as possible, simply taking the average, to dictate how the autoencoder learns and updates its internal weights and biases.

4.6 Future work

Autoencoders and their applications to anomaly detection in high energy physics are not well understood, and this thesis on scraped on some issues and attempts that needs to be done. The work done in this thesis has led to three areas of focus that should be explored more, listed below.

- Computational bottlenecks
- Data and feature engineering
- Network and architecture engineering

Computational bottlenecks

From sections 4.2 and 4.3 there seems to be a positive trend in increasing the amount of training data for the autoencoders, both regular and variational. With a continual increase in data from Run3 and onward [45], so will the loading and storing time used for doing analyses increase.

Figure 4.33: LHC nominal luminosity projections for the next 2 decades

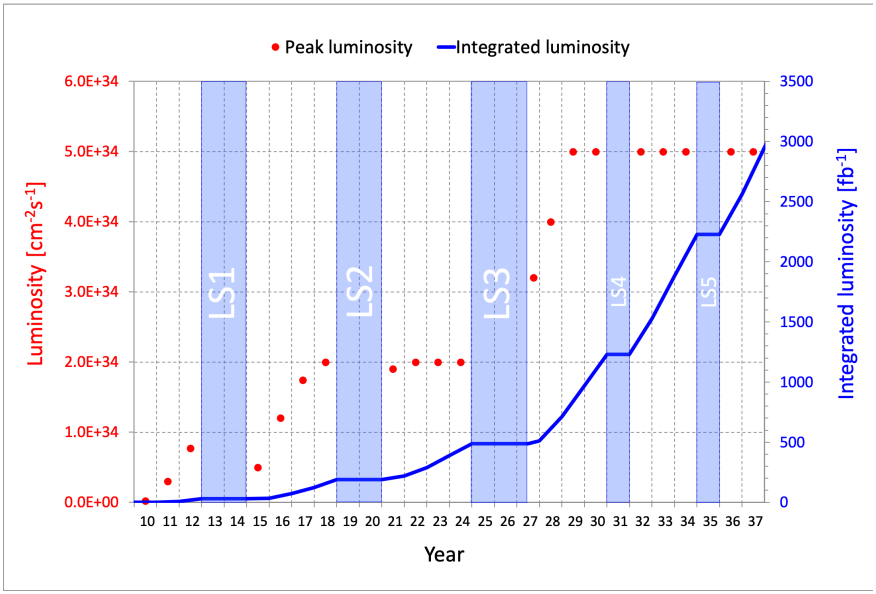
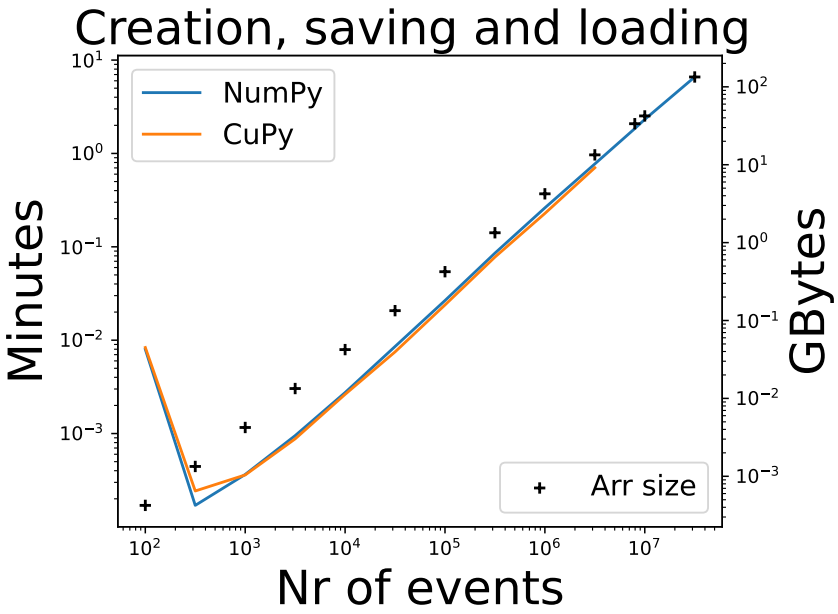


Figure 4.33 shows the expected time periods for data collection and the amount of data expected for the next two decades.

One bottleneck that was discovered during the work in this thesis was the use of Pandas pre the 2.0 version release, which is written on top of NumPy. Although Pandas has many very useful features and is the industry standard for data analysis and feature engineering, it is somewhat slow. NumPy is not parallelized, thus all operations are done in serial. Polars has proven useful as a replacement. Though in its early stages still, it unlike Pandas has a Rust backend, which natively parallelizes. Thus, if one works with rather large datasets one should use Polars, or some other parallalized tool. Another tool used for training and inference was to store training and test data post-processing in numpy arrays such that one can just load them up. NumPy makes this very simple, but as mentioned above it is serial in its execution and thus takes long time for large arrays. In the 2 lepton case the clear bottleneck both in processing of the training and testing data, as well as inference and training of the models, was the loading and storing time using NumPy's load and save functions. Sadly there is not much one can do per now as NumPy's load and save functions are the fastest, provided one has a lot of storage available. CuPy[46] is a new tool that takes NumPy functions and parallalize them using CUDA. Provided one have a lot of GPU memory, and upscale the amount allowed for CuPy to allocate, it could perhaps save you some time, but as shown below, for now there is not much gained from using CuPy over Numpy.

Figure 4.34: NumPy vs CuPy time comparison. Left y-axis shows time in minutes for creating, writing and reading from disk as function of number of events used in the *events* $\times$ 529 dimentional array. Right y axis shows array size in Gigabytes for number of events, visualized by the black crosses.



In figure 4.34 we have a comparison in runtime between NumPy and CuPy. The test checks how long time in minutes both libraries take to create a uniformly distributed array of size $Events \times Features$, where the number of features were set to 529, the number of features in the RMM structure used in this thesis. The number of events were set in powers of 10 from 2 to 8, with a half increment, i.e $[2, 2.5, 3, ..., 7.5]$. The GPU used in the test is the same used in the rest of the thesis, an NVIDIA A100-PCIE-40GB. The left y-axis shows the amount of minutes each array took, where the right yaxis shows the array size in Gigabytes. From this figure we really see that this bottleneck is not something one can do much about, unless one can aquire large amounts of GPU memory. An even then, Numpy is still incredibly fast.

Another bottleneck is computation time of training and inference. As many machine learning scientists knows, there is a continuing battle between speed and accuracy in terms of choosing an appropriate batchsize. The choice of batchsize for this thesis is 8192. It is common to choose a batchsize that is a multiple of two, as this is most optimized for GPU's. 8192 is 2^{13}, and it a rather large batchsize. Allthough this allows for epoch training time to be as low as less than 2 minutes for 2 lepton case and less than 1 minute for 3 lepton case, it does decreacy the model's ability to learn more about the dataset. With better and faster equipment this could be improved whilst decreasing the batchsize.

Data and feature engineering

Scaling

One aspect that was not explored in this thesis is the choice of scaling for the data. The choice used for this thesis was the MinMax scaling algorithm from Sci-kit learn. From previous work done on the ATLAS Open data [47] it appeared that MinMax was better for over all accuracy than Standard scaling. Still, it should be looked at and more understood why this is or is not the case for the Run2 dataset used in this thesis.

Interpretation and possible feature changes

This thesis used the Rapidity-Mass matrix described by Chekanov[32] as features in the dataset. Whilst containing features that are very uncorrolated it creates a feature signature that could be used by the autoencoder to learn underlying structures in the standard model. There are however some aspects to discuss. First, every nonexisting entry in the RMM for a given event is replaced with a zero value. This creates "islands" in the RMM structure with contributes to create the signatures for the events. It is however not clear how these zero values propagate through the network, and how they affect the performance and really the autoencoder's ability to learn the underlying structures in the standard model. In cases where one uses tools like decision trees, this is not an issue, as they can just remove the feature if it holds a certain value,

but as we need a tabular and rectangular shape to do the matrix multiplication that is neural network feed forwarding and backpropagation, a value has to be put in the missing entries, and it is not obvious or known what the choice of using zeros does.

Another choice to consider is the amount of particles in the RMM. This thesis tried allowing for 5 electrons and 5 muons, and 6 b- and ljets, yeilding a total of 529 elements in the RMM. This choice was an arbitrary one, but it is not unreasonable to think that an even larger RMM would be more benefitial, as it contains more information. Some events might have even more jets in them, or leptons. But with a larger RMM comes a larger amount of zero valued features as well. And also, with a larger RMM, the memory needed for a given array increases, which in the case of the 2 lepton case, would mean more megasets of smaller size. This is of course manageable, and can be automated with some light scripting. The computation time, from event selection and conversion from Rdataframe to Numpy, and training and inference, would increase a lot.

Architecture engineering

The regular autoencoder appears to provide a better anomaly detection performance than the variational autoencoder. The architecture is somewhat simple, given the incredibly hard task it is meant to perform. It might be possible to add some more complex layer structures like CNN autoencoders or PCA to make a more complicated and better performing network architecture. This is of course only speculation, and was not investigated much in depth. General trends were found to vary depending on the size of the dataset, and it is not trivial that for example a deep neural network is benefitial as the training sample increases. What was important though was to ensure that the latent space was large enough to store enough information for the decoder. In the early attempts the latent space were set between 10 and 50 nodes, which was way too little for the encoded information to be useful. As there is no clear way to find the optimal architecture, an educated guess on the number of nodes were done. In theory, one can create a tunable network where the number of layers and nodes in each layer are hyperparameters, but the share number of combinations would lead to much longer computation time than what was available during this thesis. Tools like Keras-Tuner could facilitate this search, but one should have significant hardware. Based on the experience from working on this thesis it seems that educated guesses are the best way to find a good architecture, but given the overall search problem it is hard or perhaps no point in doing searches for optimal architectures, more than educated guesses.

Another point to think about is how the autoencoder is trained. As explained in section 1.3 the autoencoder is trained by how well it manages to reconstruct the input data. This is done by calculating the error of each feature for a given event, and averaging the error for all features. This has its pros and cons, and should be investigated further. Suppose one has an RMM which is very sparse, with 10 bjets, 10 ljets, 10 electrons, 10 muons, and 10 photons, i.e a T5N10 matrix of 2601 colums and rows. In this case, one can easily assume that the RMM would be very sparce, as few if any SM events produced as the current energy level as the LHC contains that many particles. Thus, on could argue that some features are more important to learn correctly than others, and thus should be weighted more. It is however difficult to know which features that should be in focus, as the signatures for each event's might differ, and some might look completely different from one another. One way to deal with this issue is to calculate the average RMM for a given channel, and then see if there are general trends in the RMM for those channels. Then one could perhaps weigh the most used features more than the more sparce areas.

Chapter 5

Conclusion

The main goal of this thesis was to benchmark and investigate the performance and usage of autoencoders in beyond standard model searches. The analysis and testing was done using n-tuples from ATLAS that was converted to python dataframe structures. We argued for using the Rapidity-Mass matrix as features in our input data, with 6 bjets and 6 ljets, and 6 of each lepton. The original goal was to test and understand the performance in the case where we have a 3 lepton + e_T^{miss} final state. In testing it was shown that the performance was not too impressive, thus the choice was made to use the 2 lepton + e_T^{miss} dataset which contains much more data. Due to the large size of the total dataset, we proposed a solution where the natural distribution in the total set was conserved in smaller batches, called megasets. Several tests were deviced to benchmark the autoencoders, making anomalous events by altering the p_T of standard model events and testing on two supersymmetric signal models. We showed that the autoencoders performance increased with larger training samples, but argue for more testing as these methods are not well understood yet. The performance was measured in three categories, how well it reconstructs the test dataset, how much background and signal is left in the signal region, and the significance it achieves when performing cuts in the signal region. It was showed that for the first category, the regular autoencoder is much better than the variational autoencoder, creating a slopelike shape. In the second category the regular autoencoder is much better at reducing the background, but not that much better at increasing the amount of signal. In the third category the regular autoencoder performs better than the variational autoencoder. It was also shown that with an increase in training data, the first category was improved for both the regular autoencoder and the variational autoencoder.

We also argue for future work and challenges with the method. Amongst other issues where computational bottlenecks related to writing and loading of data from training and inference. It is recommended to further investigate the RMM, as well as alter the training process by physics informed or machine learning informed choices, such as a weighted MSE.

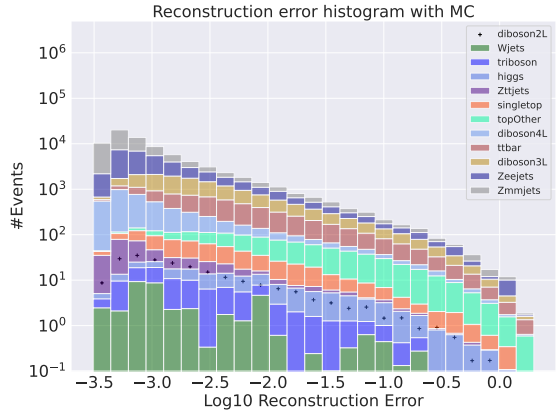
Appendices

Appendix A

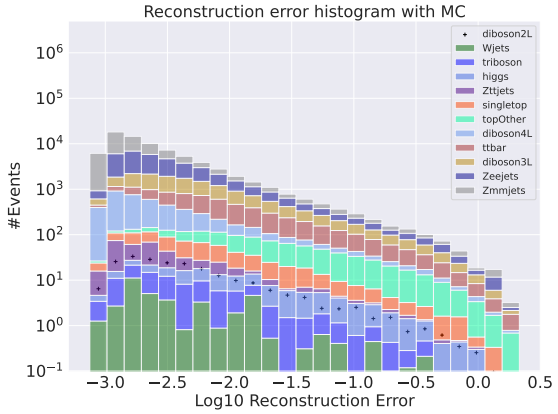
A.1 Channel removal testing

In this section the remaining histograms for the channel removal test are shown. Each figure contains the same subfigure structure as shown in the results and discussion section.

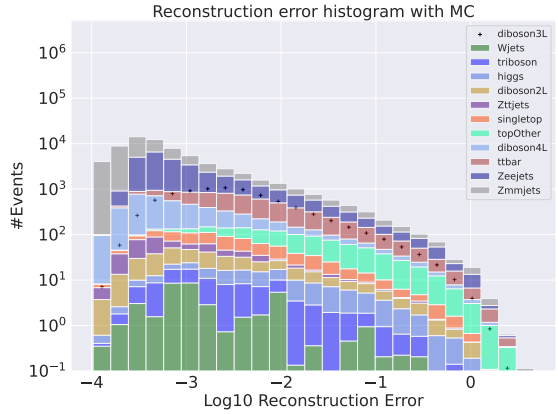
Regular autoencoder



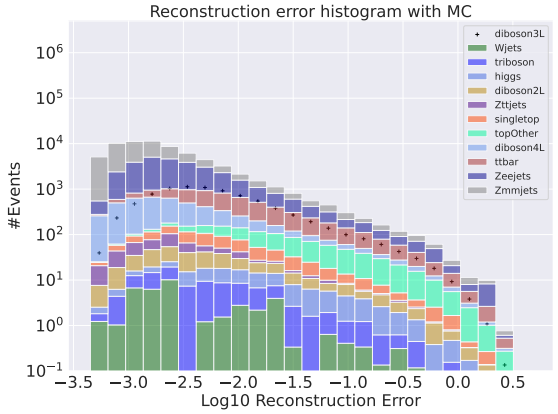
(a) Reconstruction error on validation SM MC from the small Autoencoder. Here the diboson2l channel has been removed from training and is used as signal. No significant difference in distributions is found.



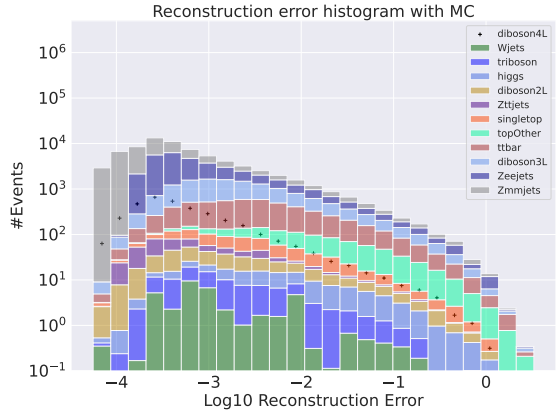
(b) Reconstruction error on validation SM MC from the big Autoencoder. Here the diboson2l channel has been removed from training and is used as signal. No significant difference in distributions is found.



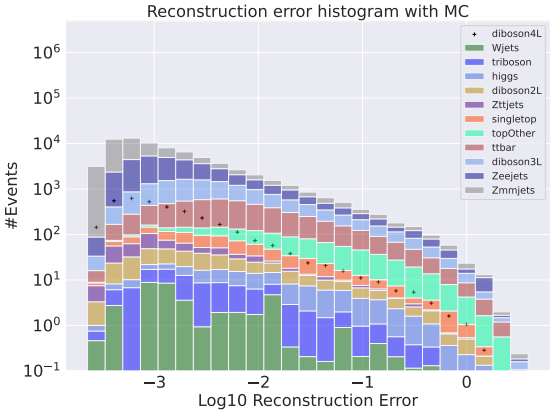
(c) Reconstruction error on validation SM MC from the small Autoencoder. Here the diboson3l channel has been removed from training and is used as signal. No significant difference in distributions is found.



(d) Reconstruction error on validation SM MC from the big Autoencoder. Here the diboson3l channel has been removed from training and is used as signal. No significant difference in distributions is found.

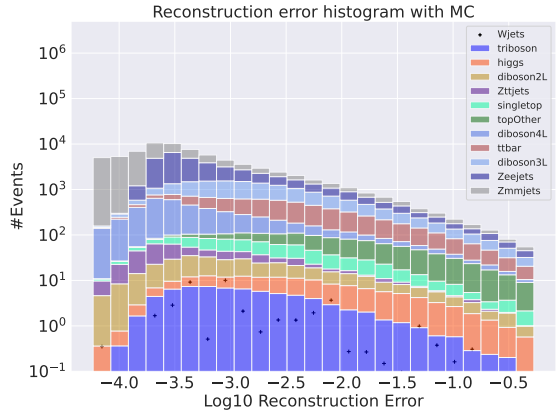


(e) Reconstruction error on validation SM MC from the small Autoencoder. Here the diboson4l channel has been removed from training and is used as signal. No significant difference in distributions is found.

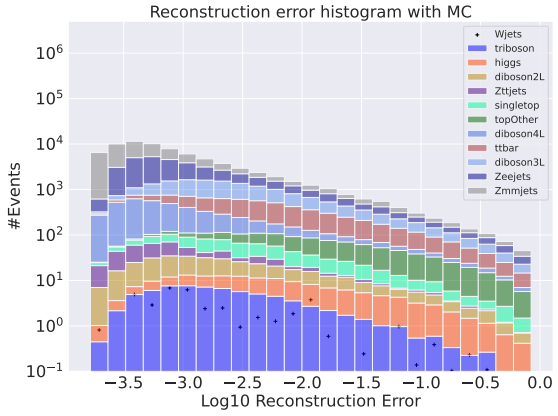


(f) Reconstruction error on validation SM MC from the big Autoencoder. Here the diboson4l channel has been removed from training and is used as signal. No significant difference in distributions is found.

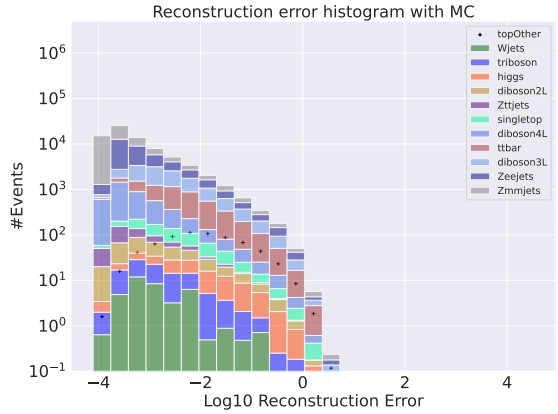
Figure 1: Reconstruction error on validation SM MC from the small and big Autoencoder. Here the diboson2l, diboson3l and diboson4l have been used for the small (left) and large (right) regular autoencoder



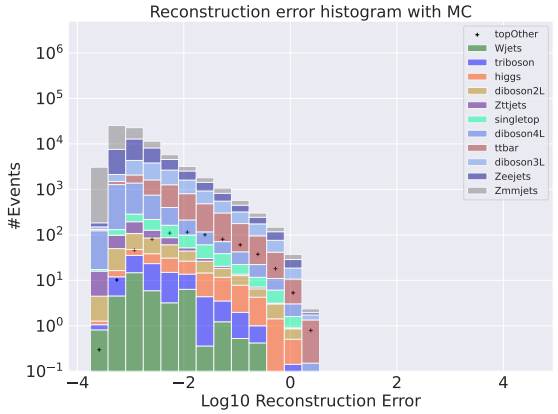
(a) Reconstruction error on validation SM MC from the small Autoencoder. Here the Wjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



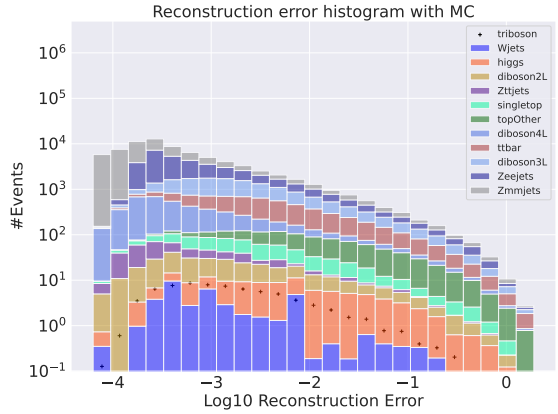
(b) Reconstruction error on validation SM MC from the big Autoencoder. Here the Wjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



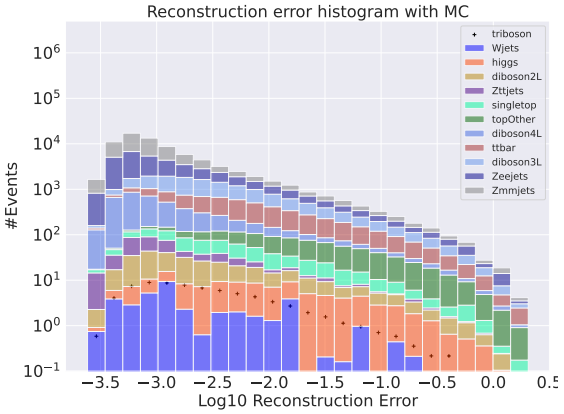
(c) Reconstruction error on validation SM MC from the small Autoencoder. Here the topOther channel has been removed from training and is used as signal. No significant difference in distributions is found.



(d) Reconstruction error on validation SM MC from the big Autoencoder. Here the topOther channel has been removed from training and is used as signal. No significant difference in distributions is found.

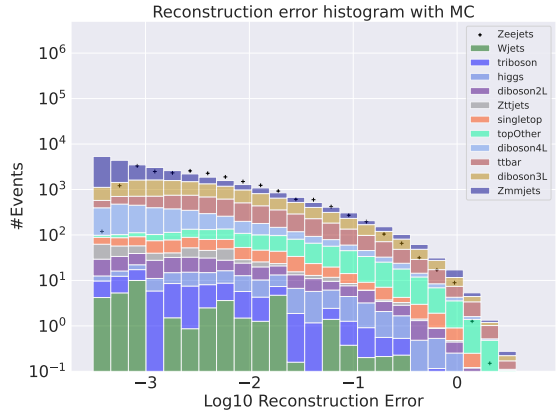


(e) Reconstruction error on validation SM MC from the small Autoencoder. Here the triboson channel has been removed from training and is used as signal. No significant difference in distributions is found.

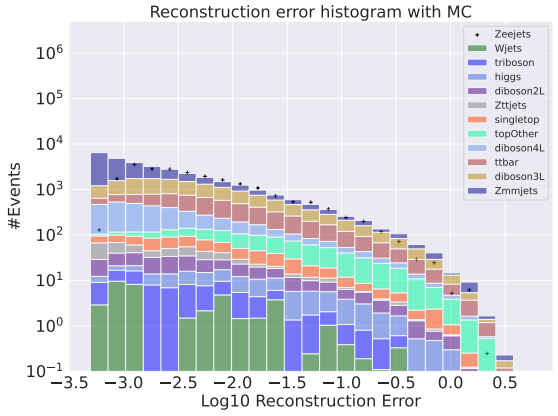


(f) Reconstruction error on validation SM MC from the big Autoencoder. Here the triboson channel has been removed from training and is used as signal. No significant difference in distributions is found.

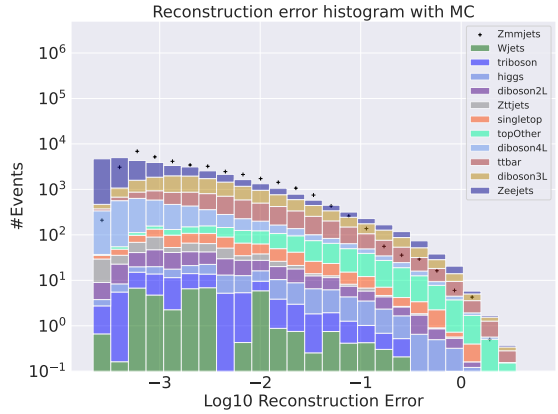
Figure 2: Reconstruction error on validation SM MC from the small and big Autoencoder. Here the Wjets, topOther and triboson have been used for the small (left) and large (right) regular autoencoder



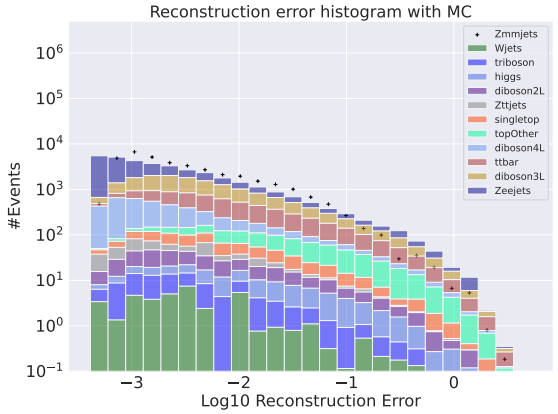
(a) Reconstruction error on validation SM MC from the small Autoencoder. Here the Zeejets channel has been removed from training and is used as signal. No significant difference in distributions is found.



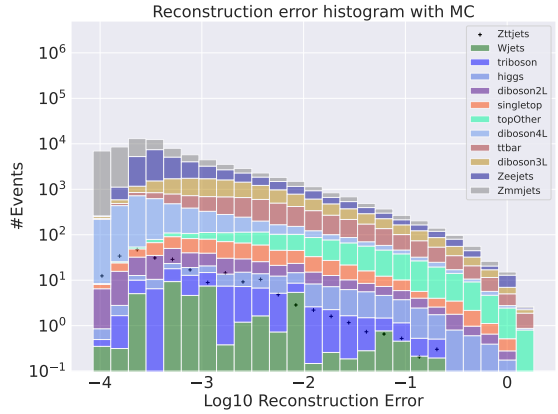
(b) Reconstruction error on validation SM MC from the big Autoencoder. Here the Zeejets channel has been removed from training and is used as signal. No significant difference in distributions is found.



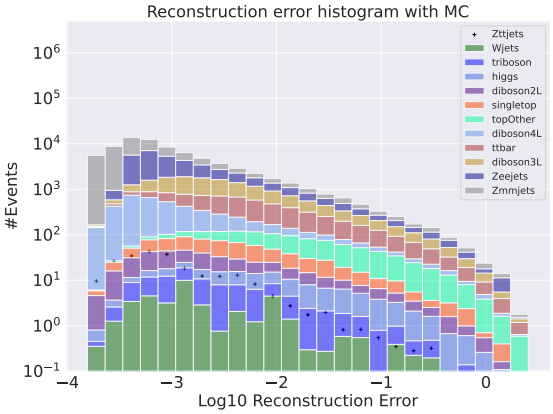
(c) Reconstruction error on validation SM MC from the small Autoencoder. Here the Zmmjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



(d) Reconstruction error on validation SM MC from the big Autoencoder. Here the Zmmjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



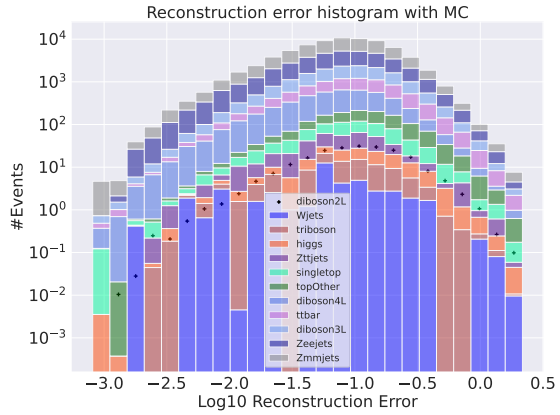
(e) Reconstruction error on validation SM MC from the small Autoencoder. Here the Zttjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



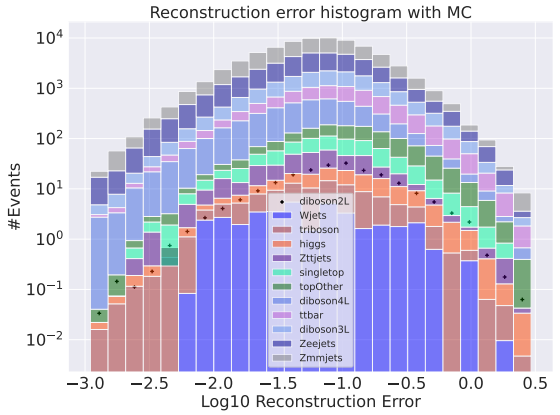
(f) Reconstruction error on validation SM MC from the big Autoencoder. Here the Zttjets channel has been removed from training and is used as signal. No significant difference in distributions is found.

Figure 3: Reconstruction error on validation SM MC from the small and big Autoencoder. Here the Zeejets, Zmmjets and Zttjets have been used for the small (left) and large (right) regular autoencoder

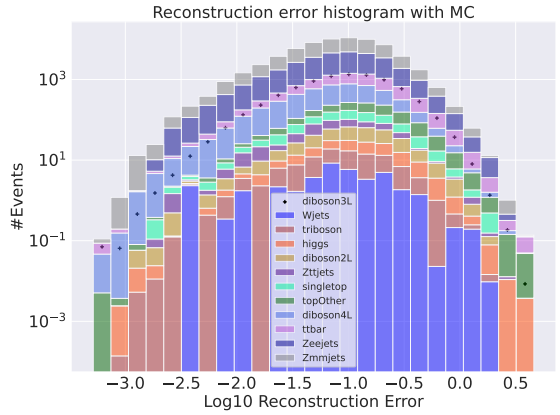
Variational autoencoder



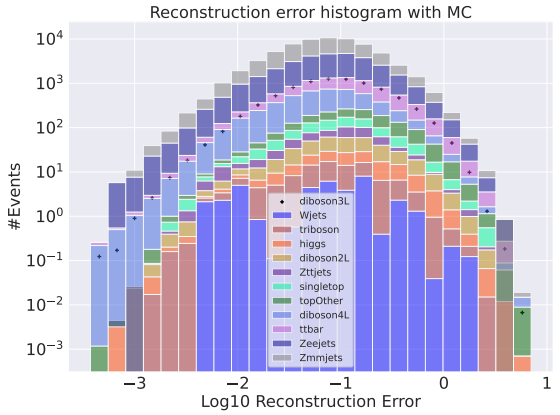
(a) Reconstruction error on validation SM MC from the small Autoencoder. Here the diboson2l channel has been removed from training and is used as signal. No significant difference in distributions is found.



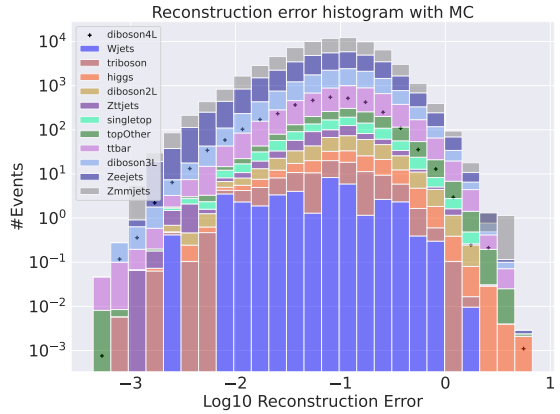
(b) Reconstruction error on validation SM MC from the big Autoencoder. Here the diboson2l channel has been removed from training and is used as signal. No significant difference in distributions is found.



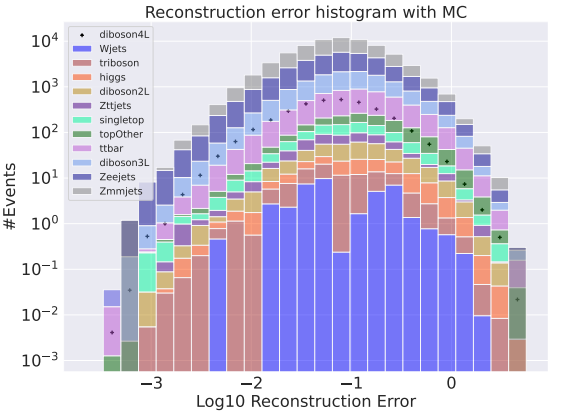
(c) Reconstruction error on validation SM MC from the small Autoencoder. Here the diboson3l channel has been removed from training and is used as signal. No significant difference in distributions is found.



(d) Reconstruction error on validation SM MC from the big Autoencoder. Here the diboson3l channel has been removed from training and is used as signal. No significant difference in distributions is found.

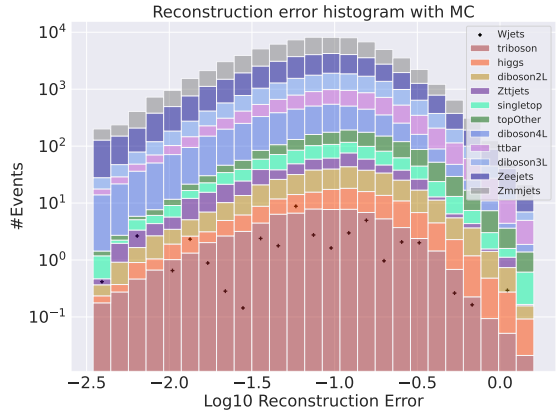


(e) Reconstruction error on validation SM MC from the small Autoencoder. Here the diboson4l channel has been removed from training and is used as signal. No significant difference in distributions is found.

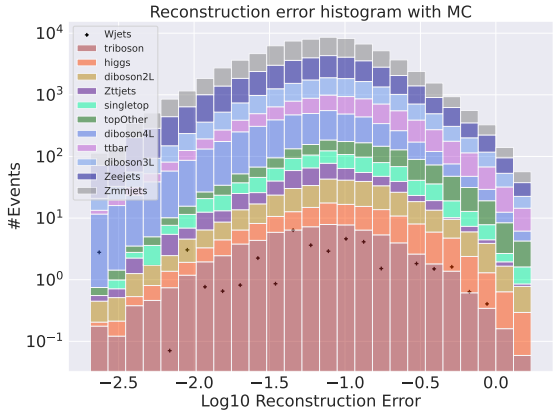


(f) Reconstruction error on validation SM MC from the big Autoencoder. Here the diboson4l channel has been removed from training and is used as signal. No significant difference in distributions is found.

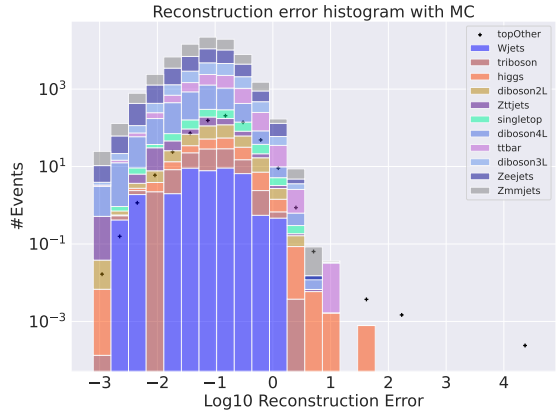
Figure 4: Reconstruction error on validation SM MC from the small and big Autoencoder. Here the diboson2l, diboson3l and diboson4l have been used for the small (left) and large (right) variational autoencoder



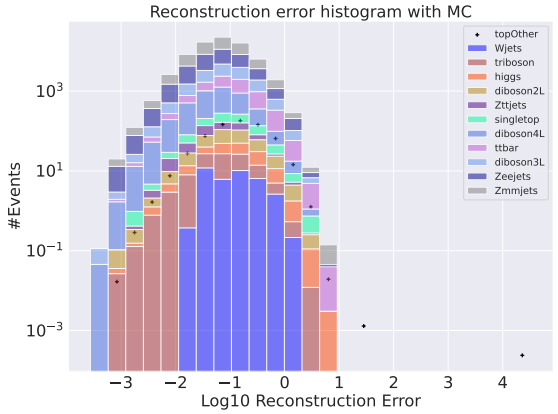
(a) Reconstruction error on validation SM MC from the small Autoencoder. Here the Wjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



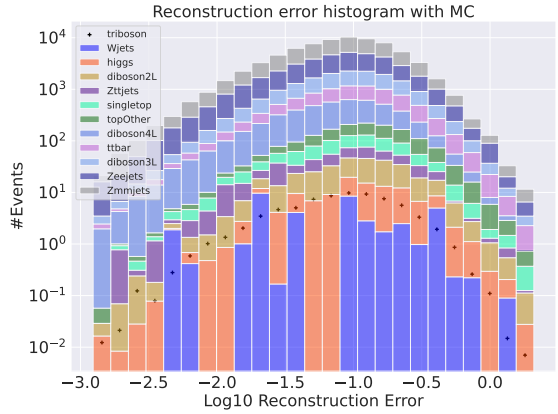
(b) Reconstruction error on validation SM MC from the big Autoencoder. Here the Wjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



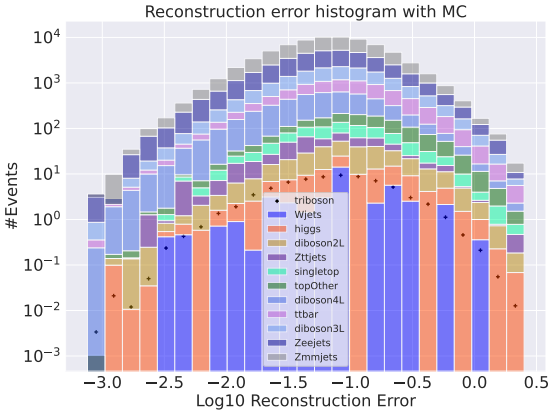
(c) Reconstruction error on validation SM MC from the small Autoencoder. Here the topOther channel has been removed from training and is used as signal. No significant difference in distributions is found.



(d) Reconstruction error on validation SM MC from the big Autoencoder. Here the topOther channel has been removed from training and is used as signal. No significant difference in distributions is found.

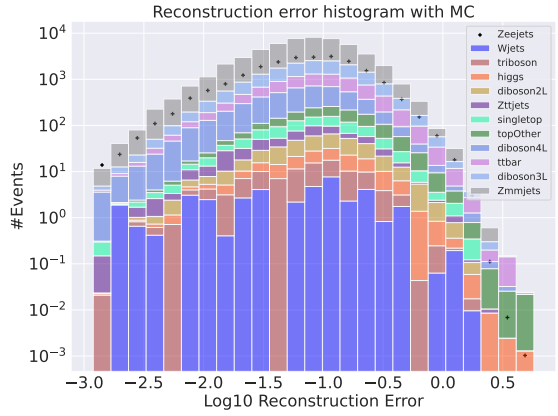


(e) Reconstruction error on validation SM MC from the small Autoencoder. Here the triboson channel has been removed from training and is used as signal. No significant difference in distributions is found.

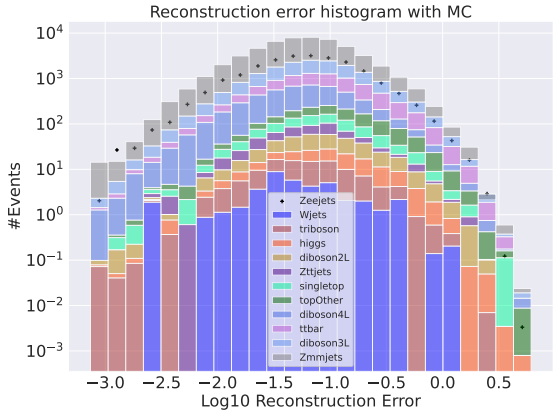


(f) Reconstruction error on validation SM MC from the big Autoencoder. Here the triboson channel has been removed from training and is used as signal. No significant difference in distributions is found.

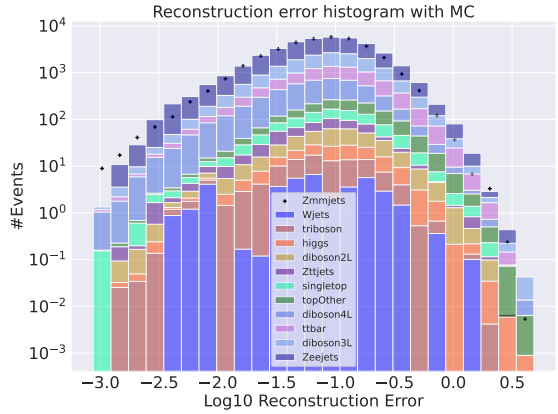
Figure 5: Reconstruction error on validation SM MC from the small and big Autoencoder. Here the Wjets, topOther and triboson have been used for the small (left) and large (right) variational autoencoder



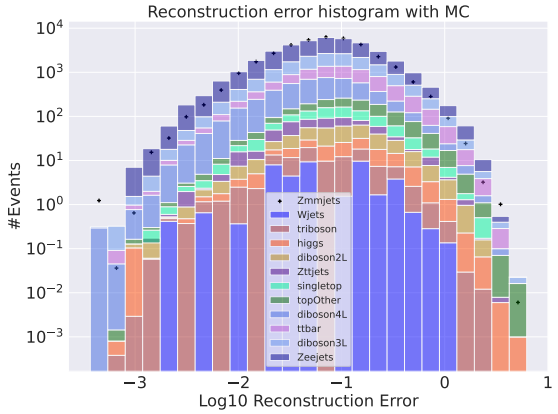
(a) Reconstruction error on validation SM MC from the small Autoencoder. Here the Zeejets channel has been removed from training and is used as signal. No significant difference in distributions is found.



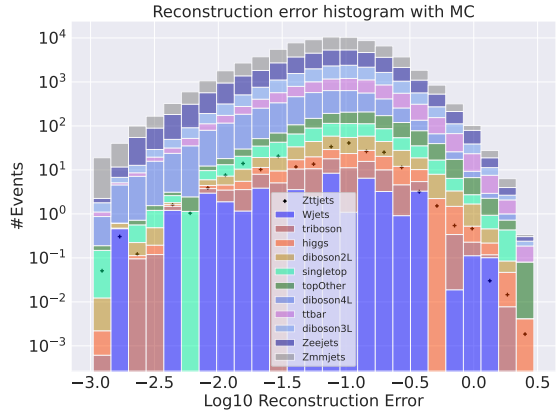
(b) Reconstruction error on validation SM MC from the big Autoencoder. Here the Zeejets channel has been removed from training and is used as signal. No significant difference in distributions is found.



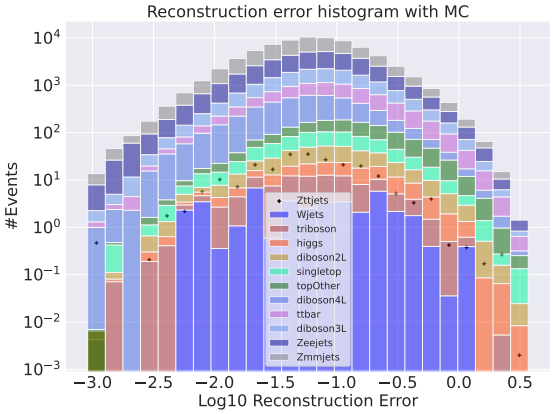
(c) Reconstruction error on validation SM MC from the small Autoencoder. Here the Zmmjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



(d) Reconstruction error on validation SM MC from the big Autoencoder. Here the Zmmjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



(e) Reconstruction error on validation SM MC from the small Autoencoder. Here the Zttjets channel has been removed from training and is used as signal. No significant difference in distributions is found.



(f) Reconstruction error on validation SM MC from the big Autoencoder. Here the Zttjets channel has been removed from training and is used as signal. No significant difference in distributions is found.

Figure 6: Reconstruction error on validation SM MC from the small and big Autoencoder. Here the Zeejets, Zmmjets and Zttjets have been used for the small (left) and large (right) variational autoencoder

A.2 Reconstruction error cuts for 3 leptons + e_T^{miss}

In this section the remainding two reconstruction error cuts in the 3 lepton + e_T^{miss} are shown. The figures here are shaped with the same subfigure structure as the figures in the results and discussion section.

Regular autoencoder output

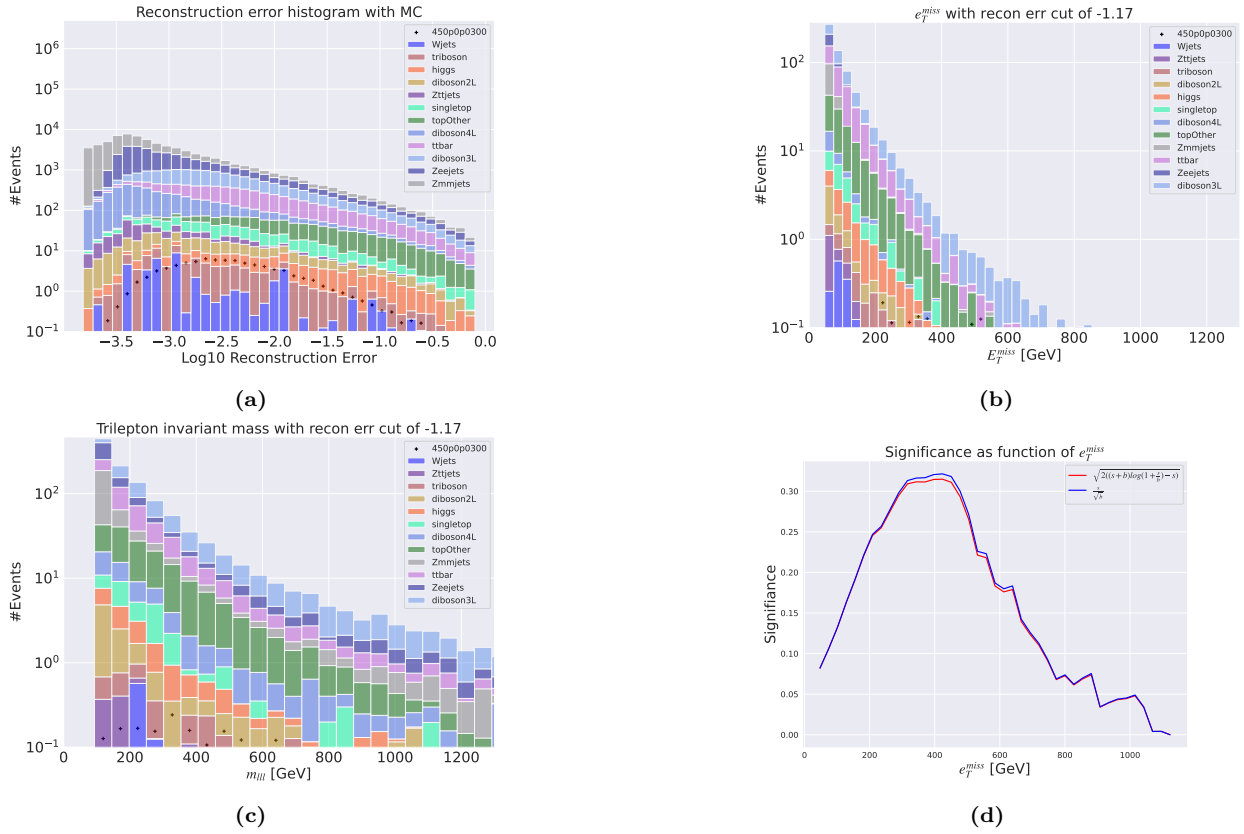
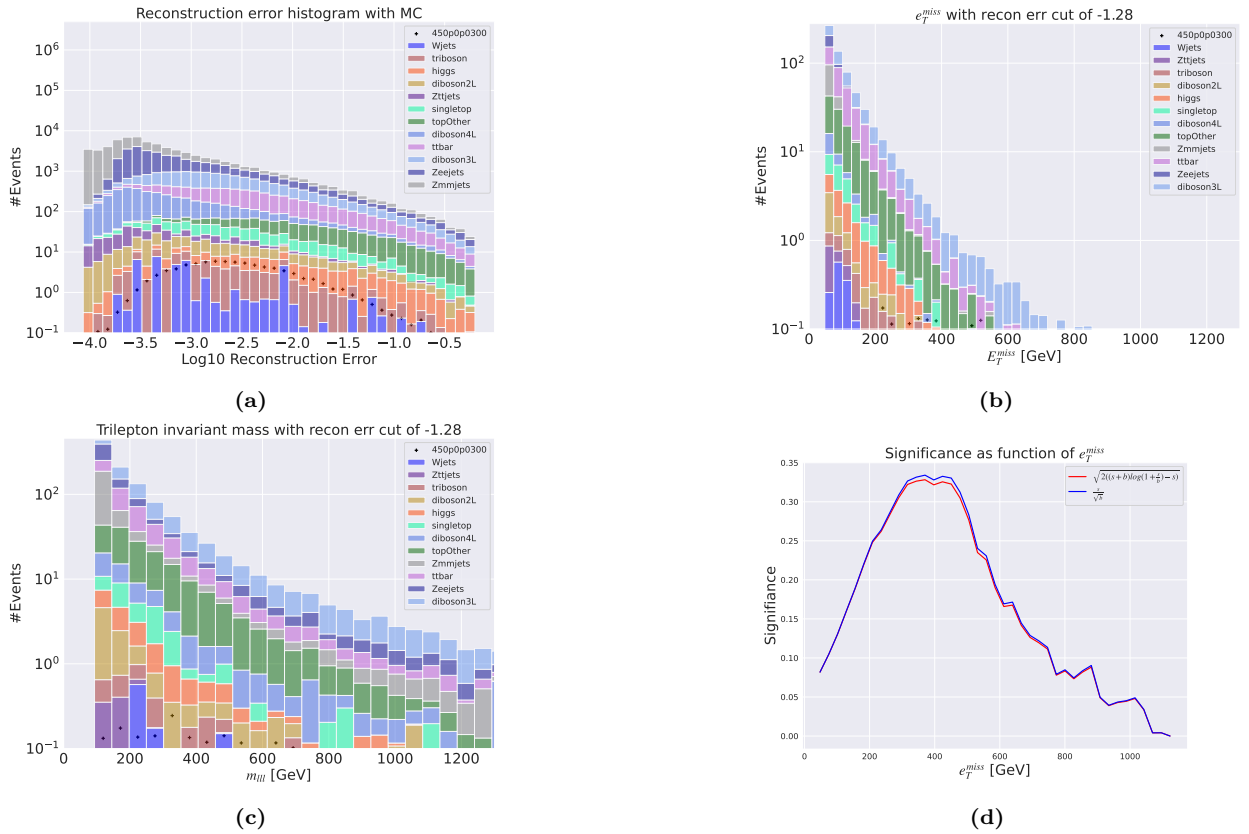
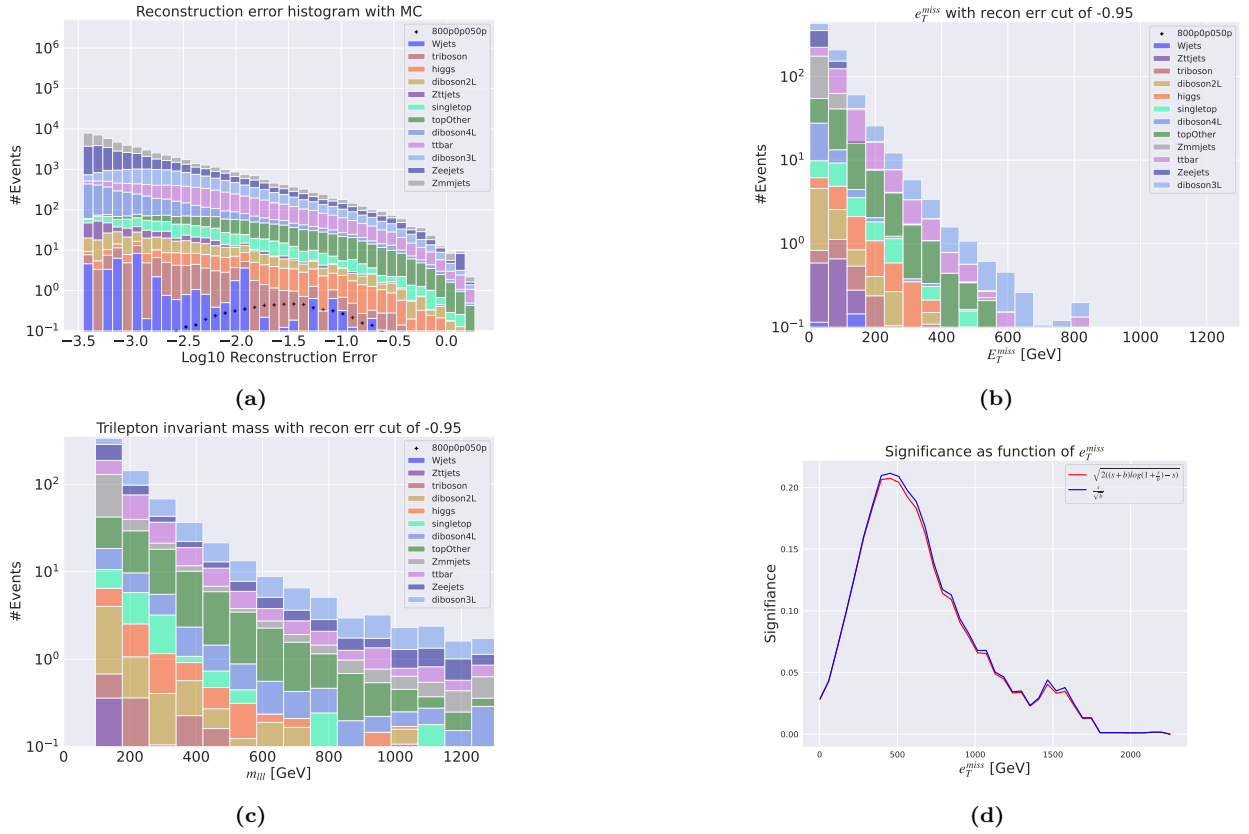


Figure 7: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a) (b) (c) (d)

Figure 8: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 450p300 model is used.



(a) (b) (c) (d)

Figure 9: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.

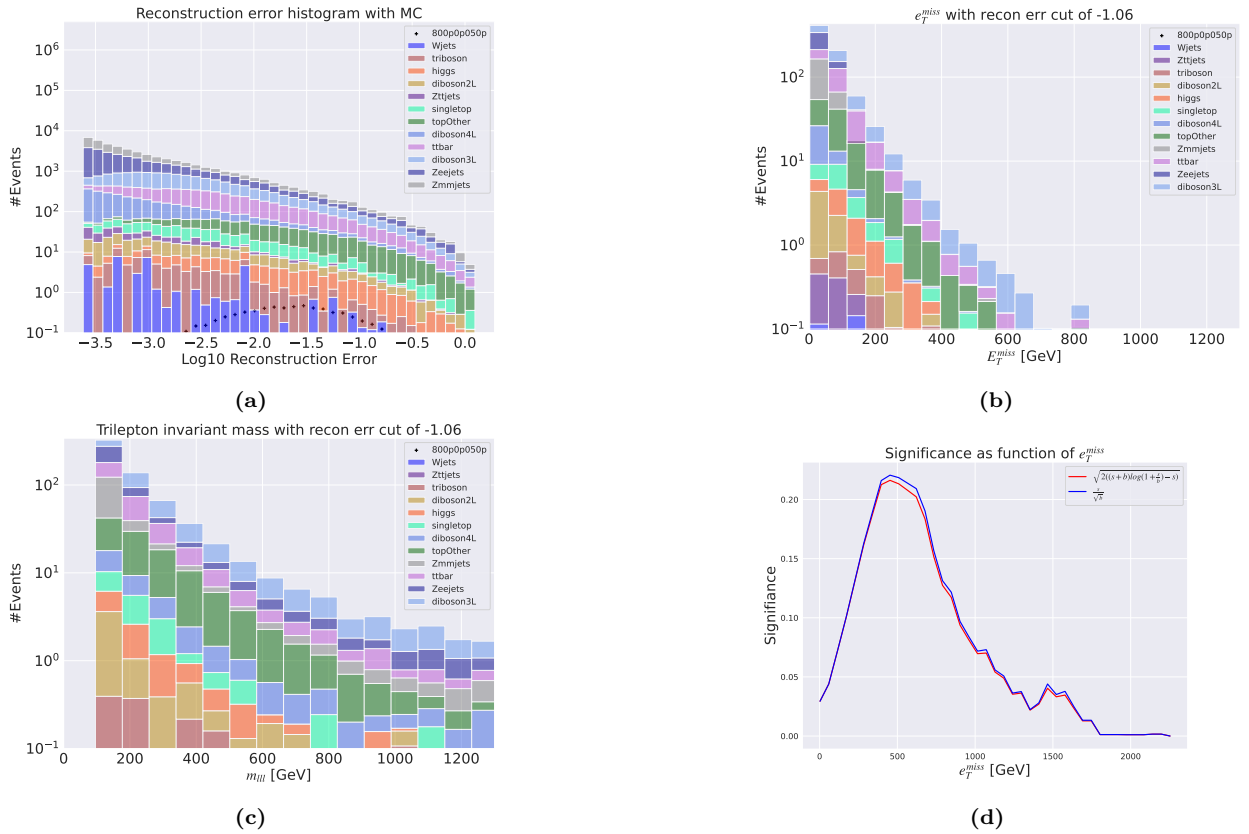


Figure 10: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 450p300 model is used.

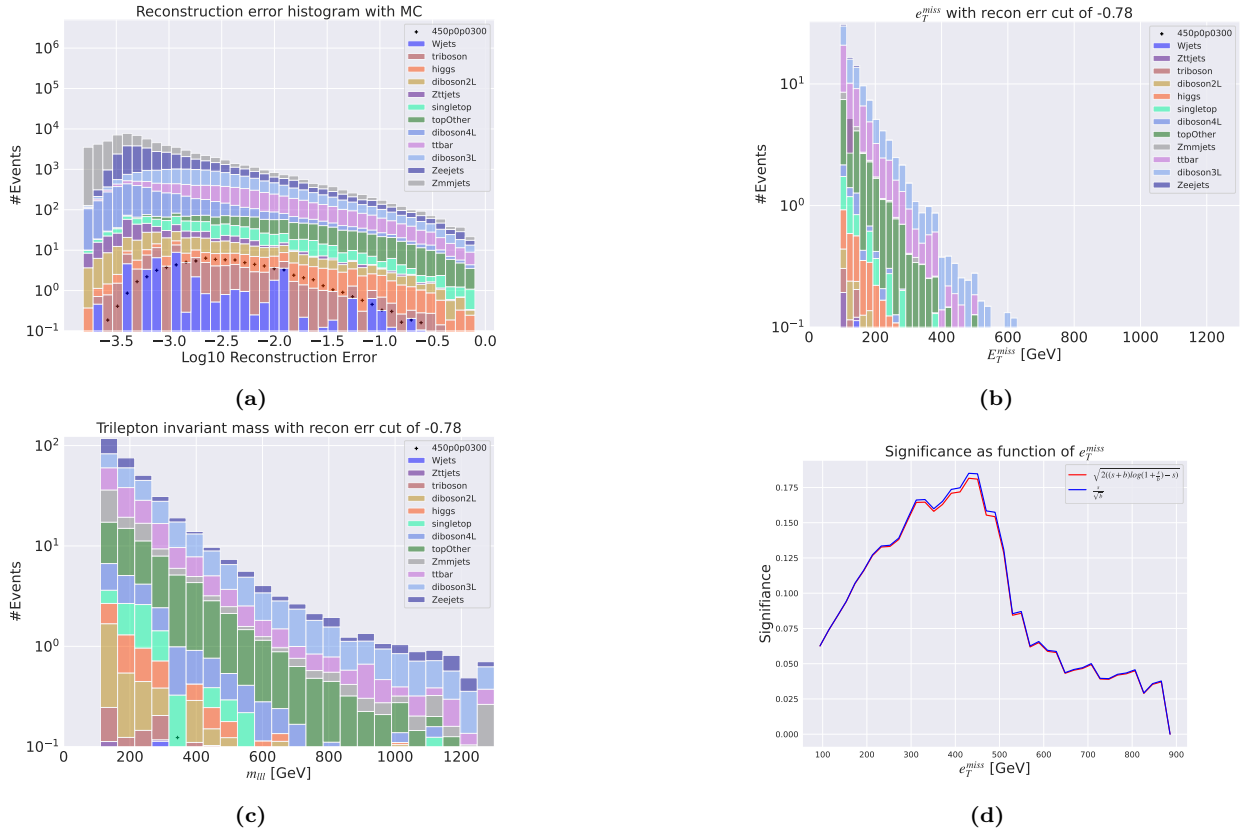
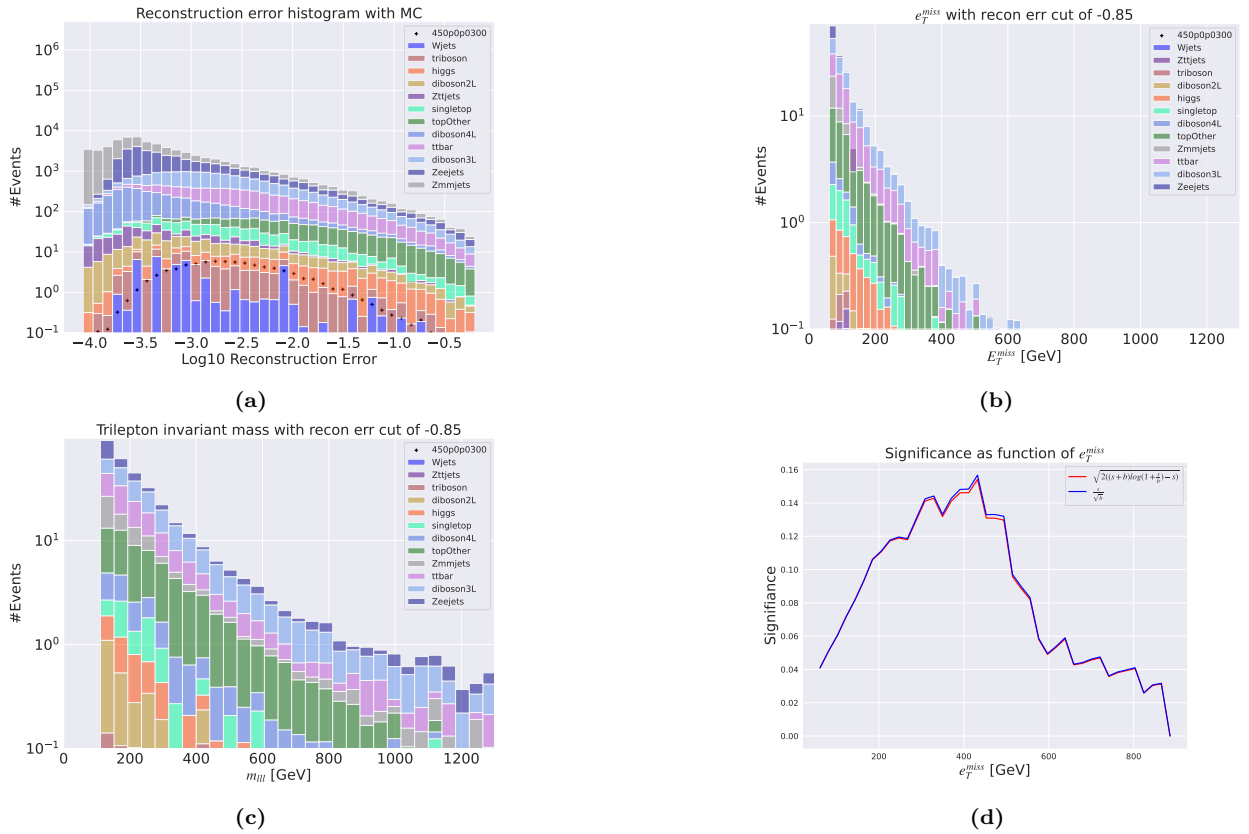
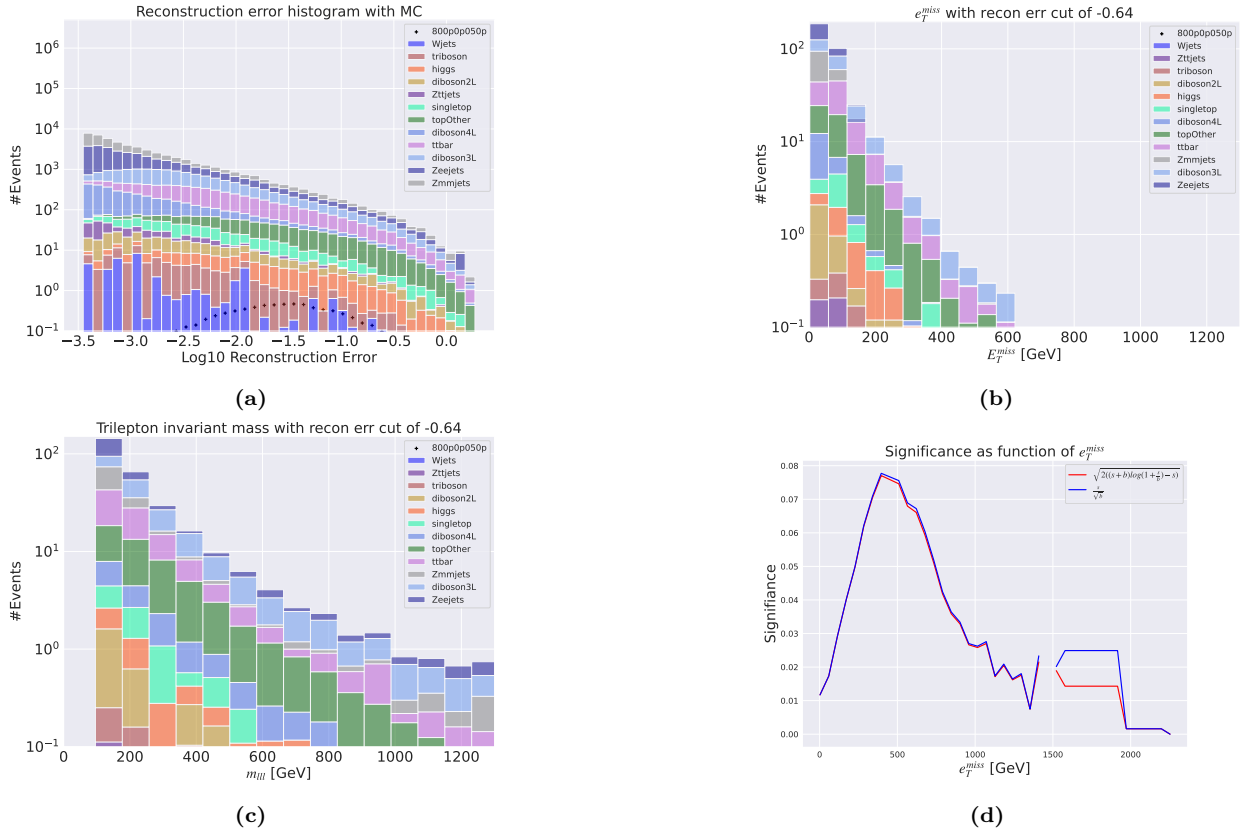


Figure 11: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a) **(b)** **(c)** **(d)**

Figure 12: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 450p300 model is used.



(a) **(b)** **(c)** **(d)**

Figure 13: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.

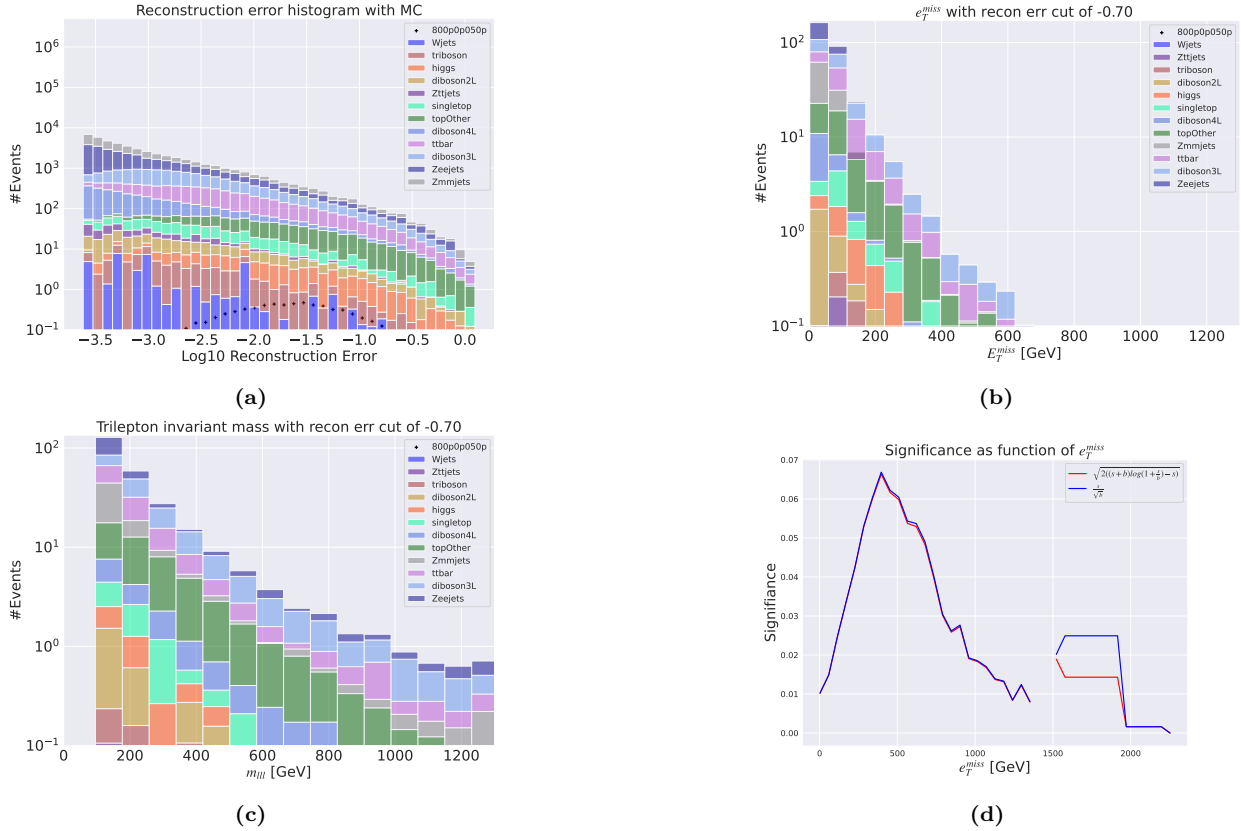
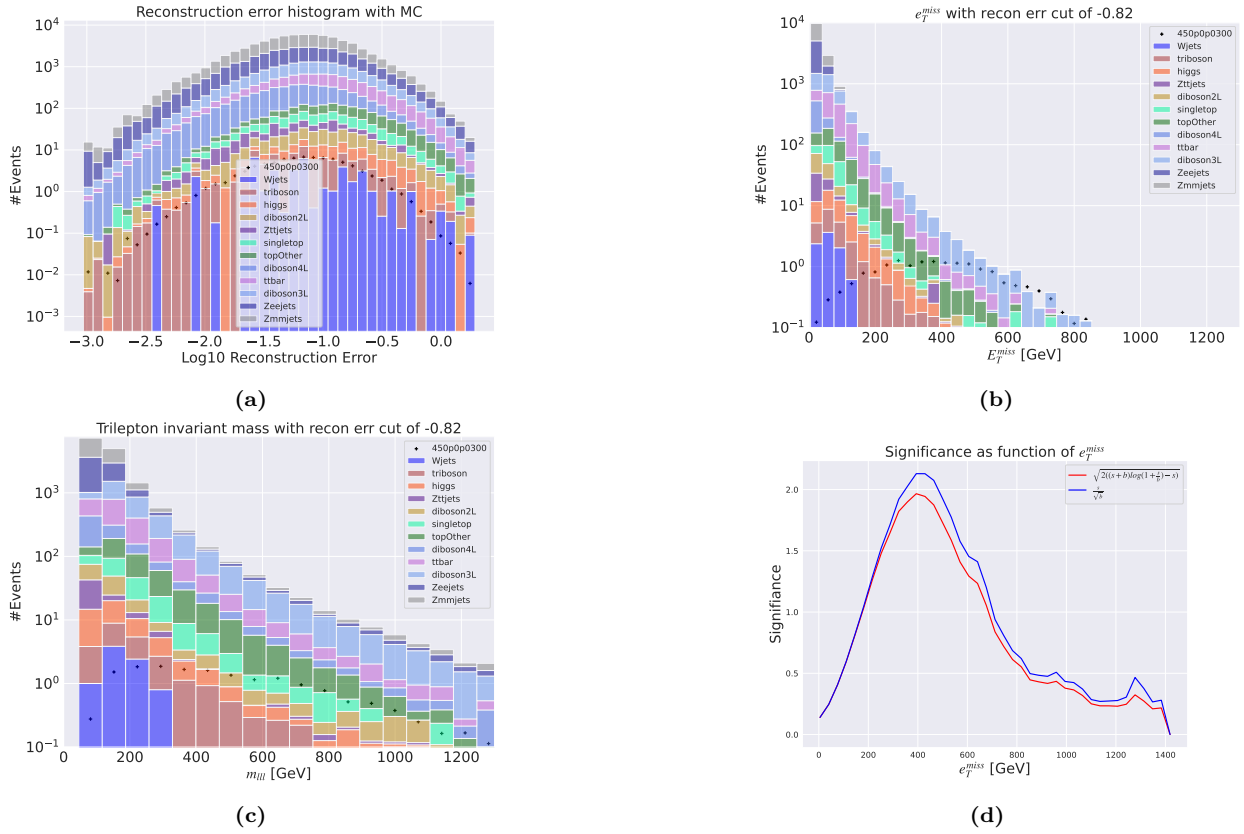


Figure 14: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY $450p300$ model is used.

Variational autoencoder output



(a) **(b)** **(c)** **(d)**

Figure 15: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep variational autoencoder. Here the SUSY 450p300 model is used.

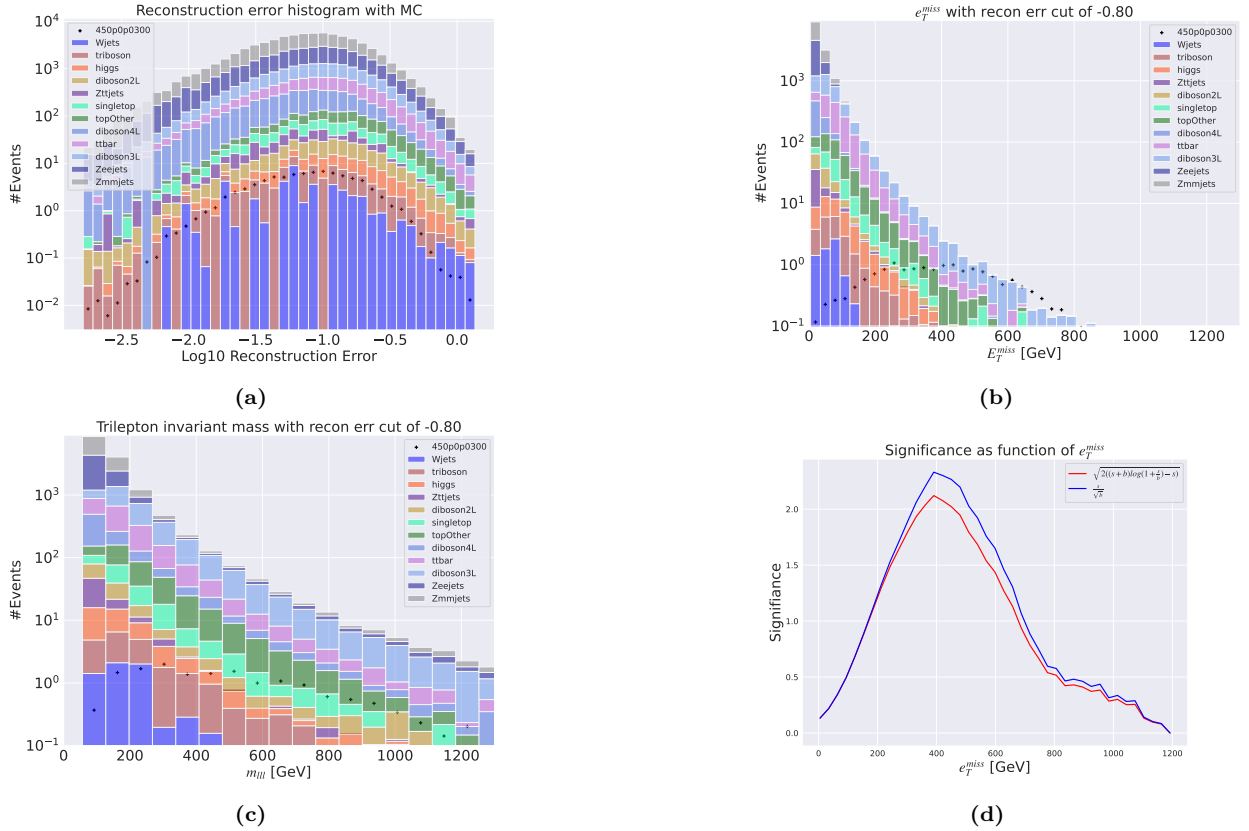


Figure 16: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow variational autoencoder. Here the SUSY 450p300 model is used.

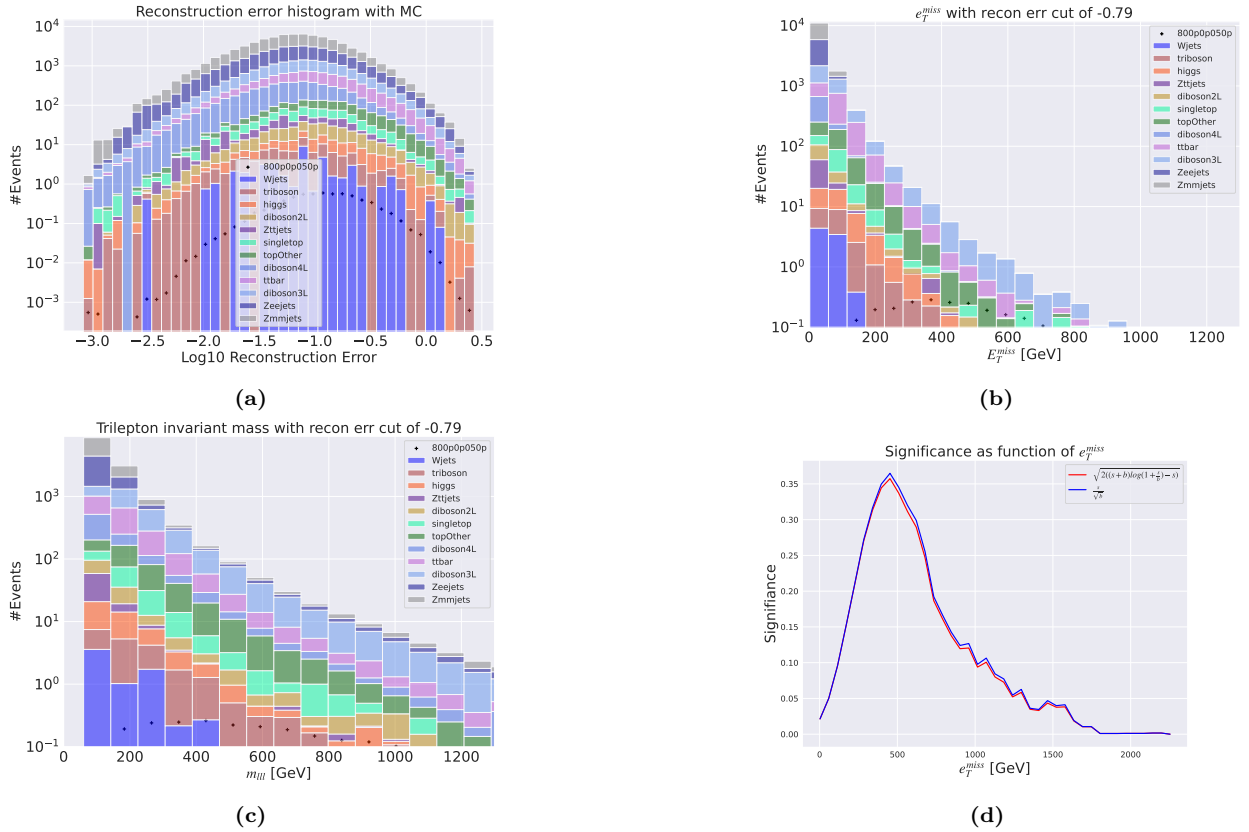


Figure 17: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep variational autoencoder. Here the SUSY 450p300 model is used.

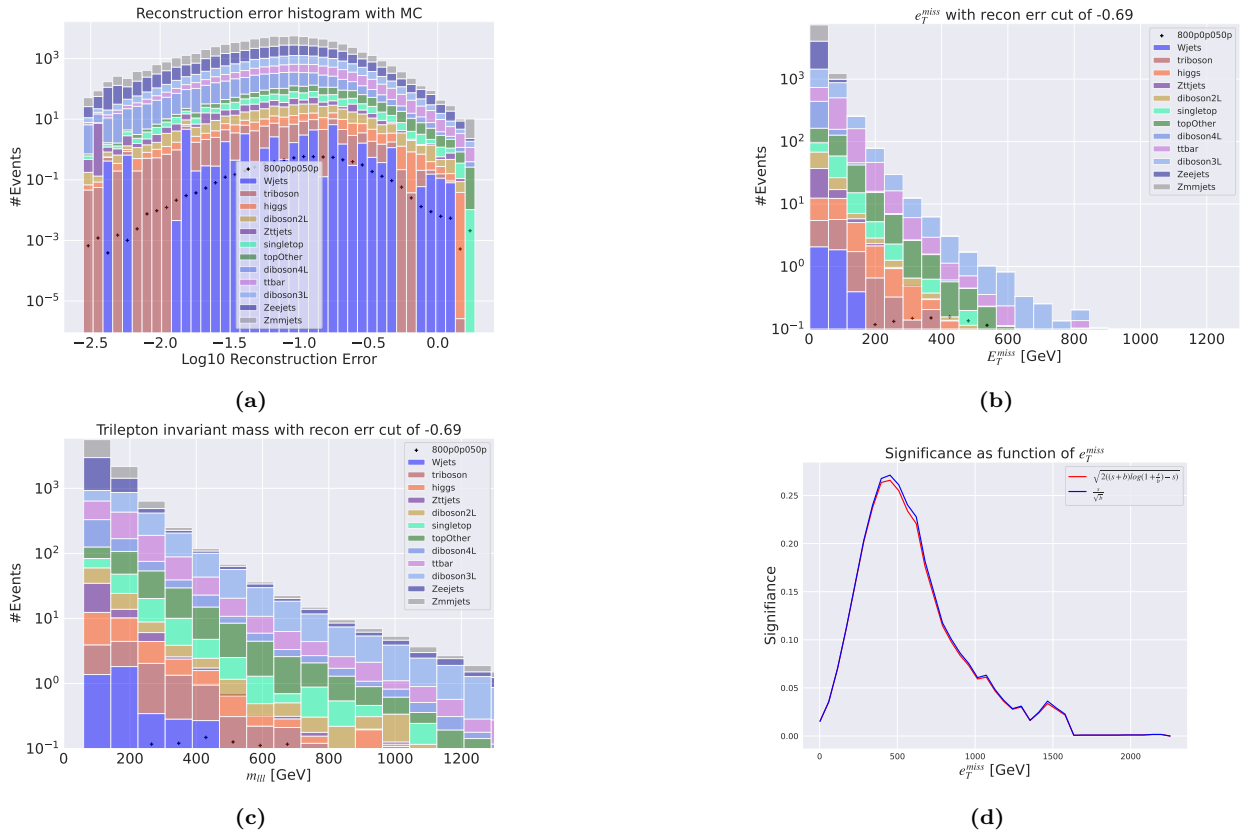


Figure 18: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow variational autoencoder. Here the SUSY 450p300 model is used.

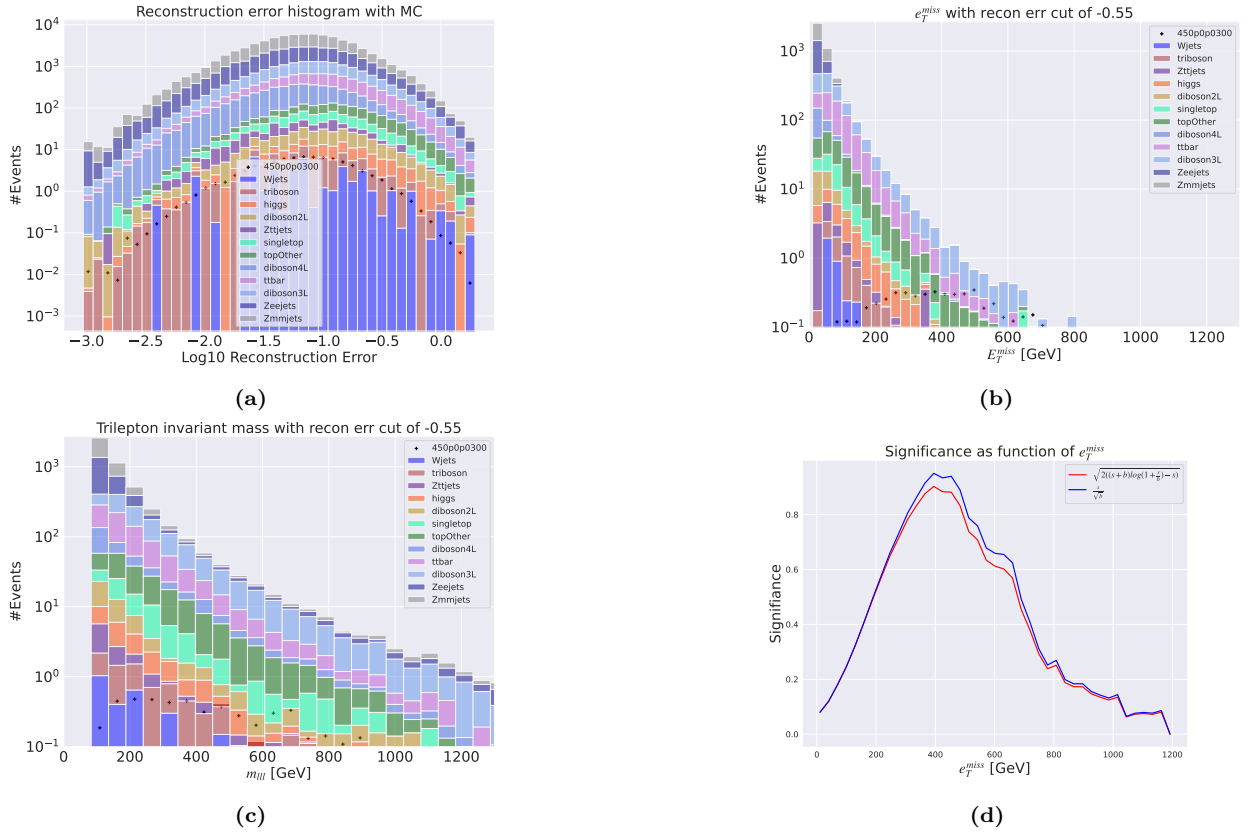


Figure 19: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep variational autoencoder. Here the SUSY 450p300 model is used.

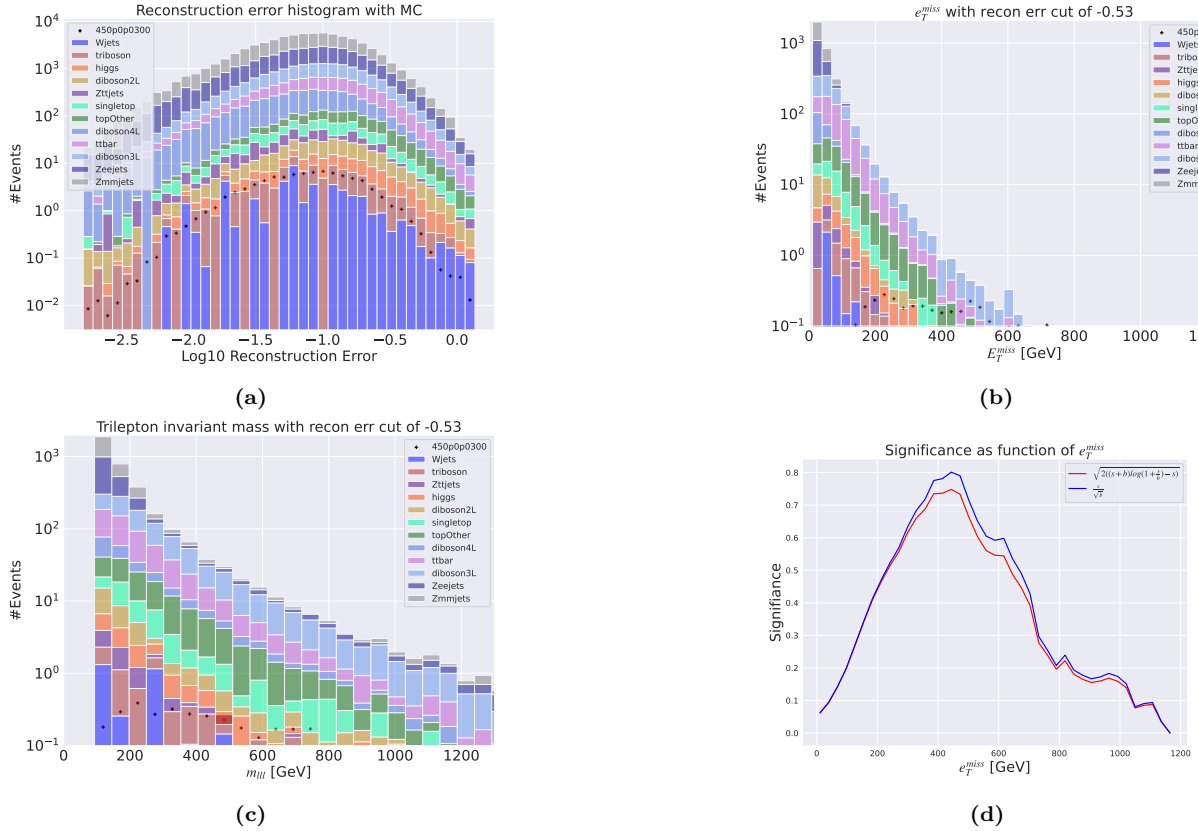


Figure 20: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow variational autoencoder. Here the SUSY 450p300 model is used.

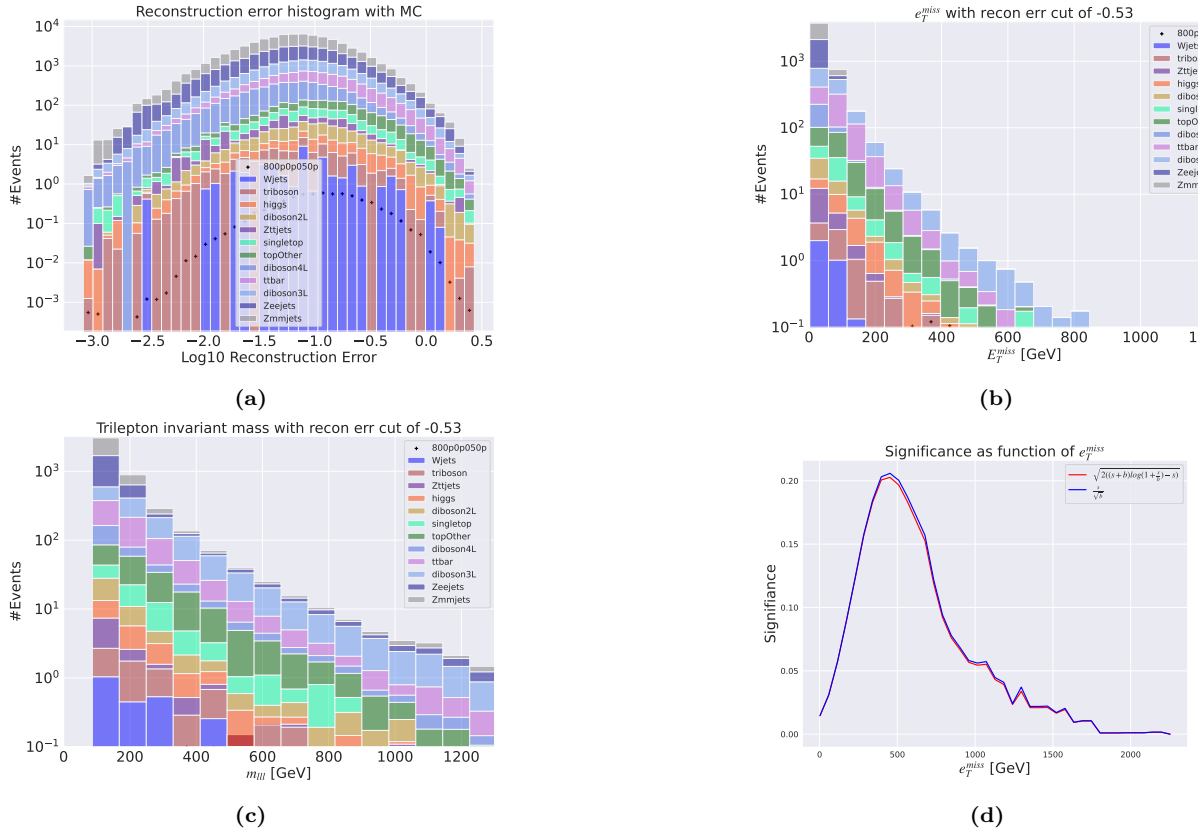


Figure 21: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep variational autoencoder. Here the SUSY 450p300 model is used.

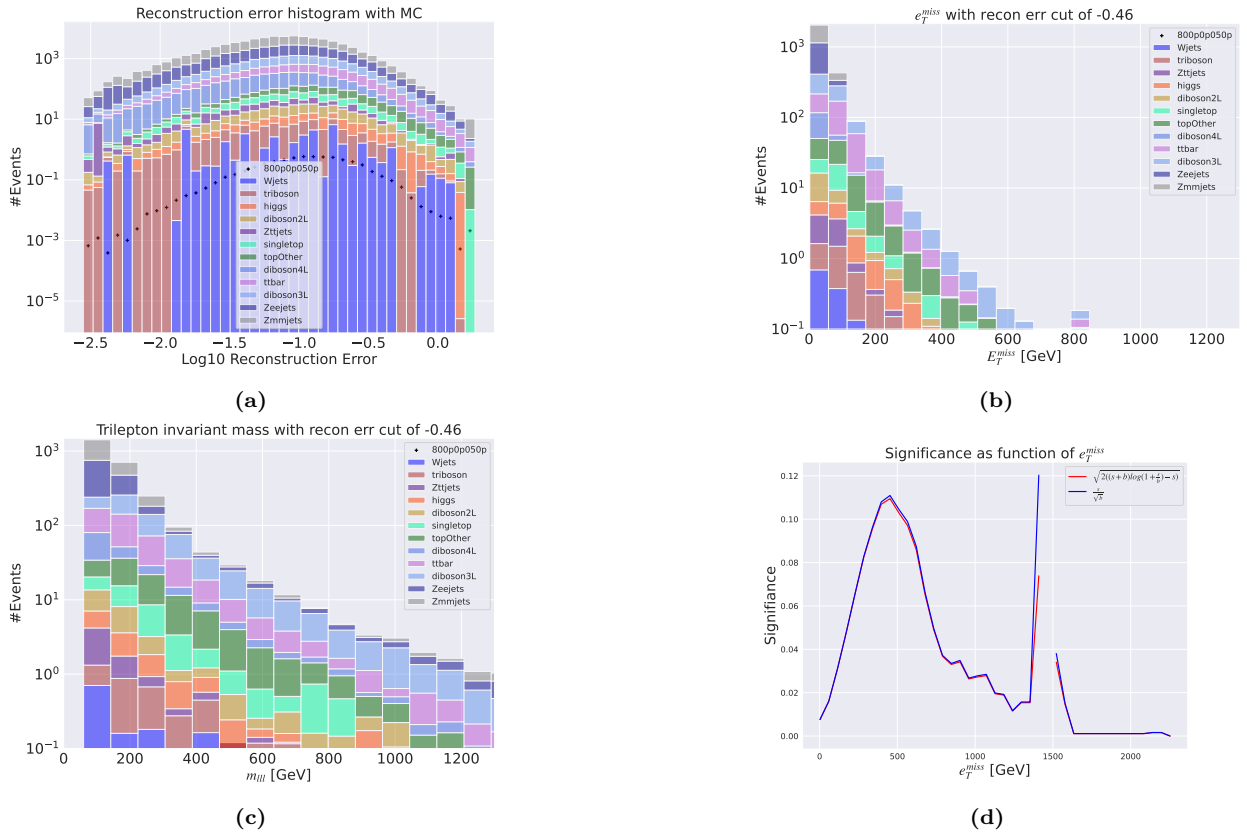
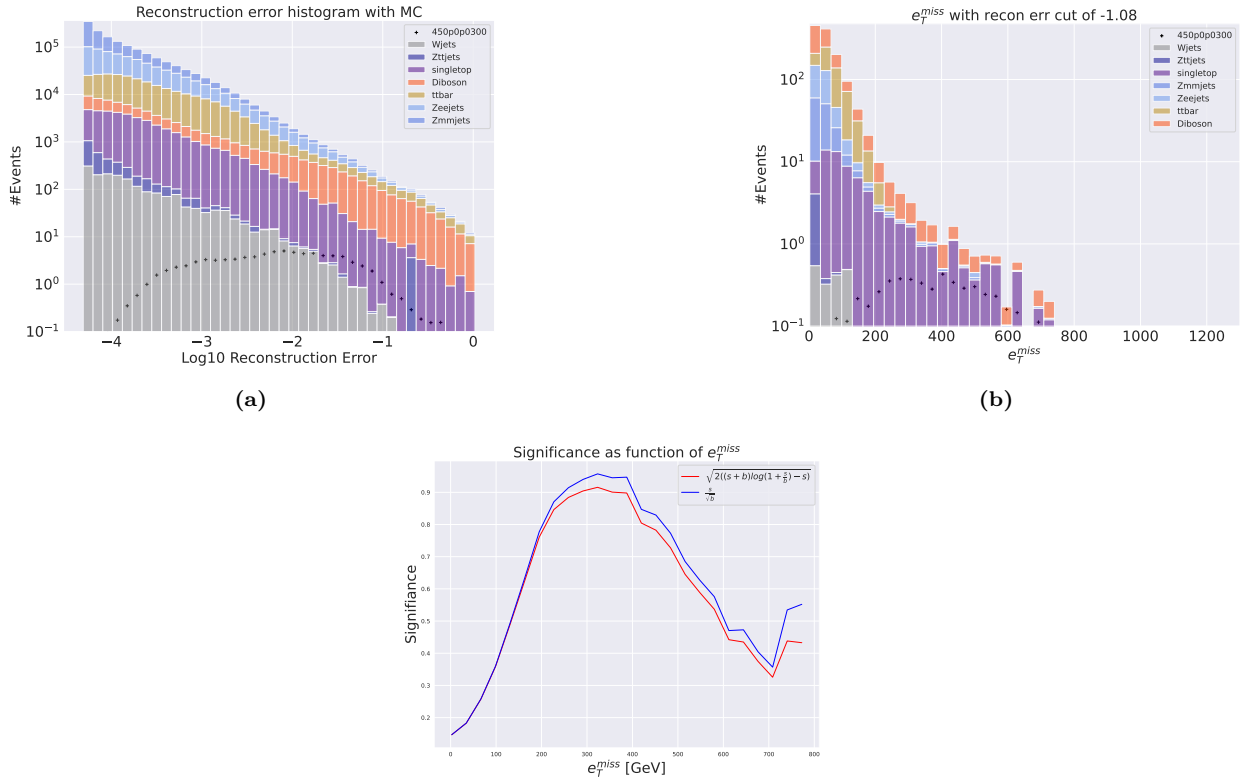


Figure 22: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow variational autoencoder. Here the SUSY 450p300 model is used.

A.3 Reconstruction error cuts for 2 leptons + e_T^{miss}

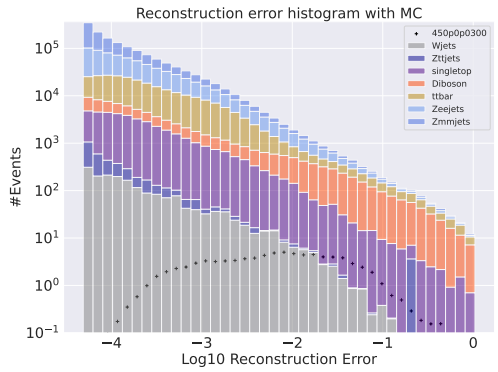
In this section the remainding two reconstruction error cuts in the 2 lepton + e_T^{miss} are shown. The figures here are shaped with the same subfigure structure as the figures in the results and discussion section.

Regular autoencoder output

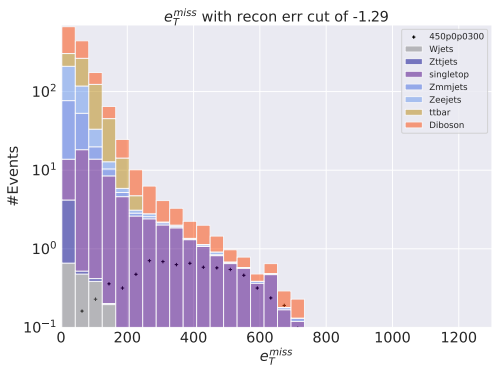


(a) **(b)** **(c)**

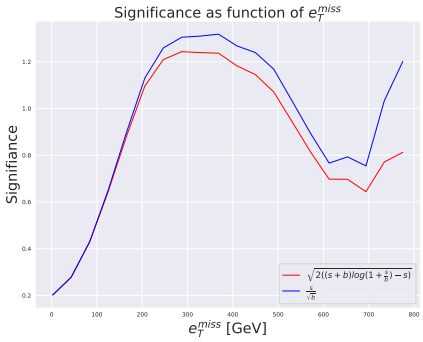
Figure 23: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a)

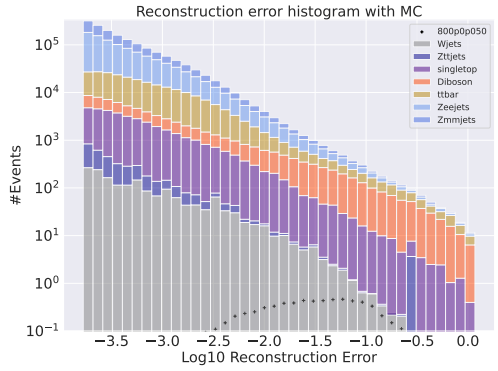


(b)

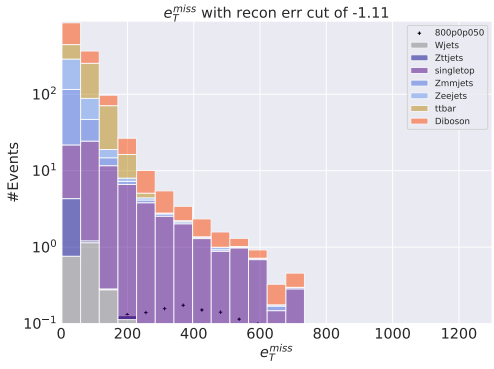


(c)

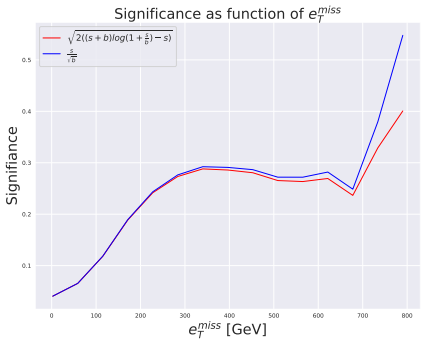
Figure 24: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a)

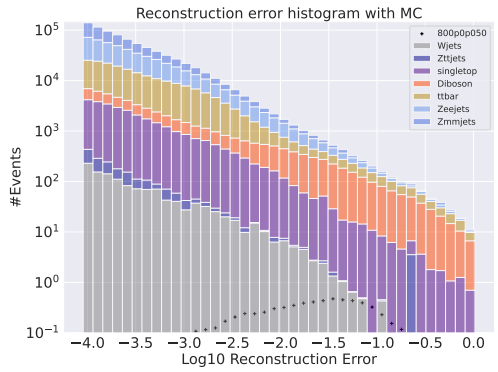


(b)

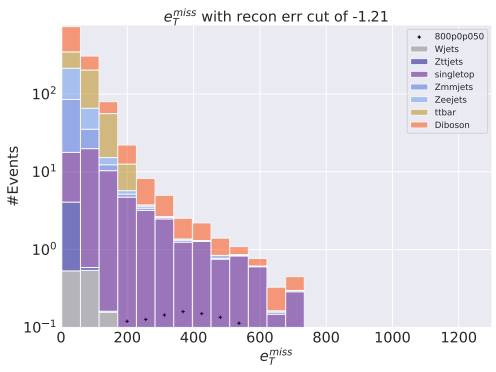


(c)

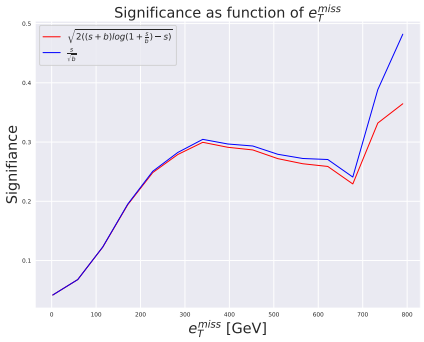
Figure 25: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 800p50 model is used.



(a)

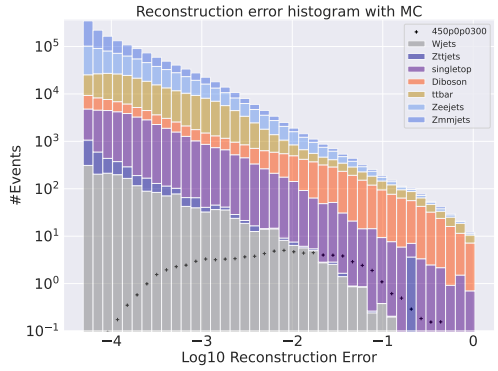


(b)

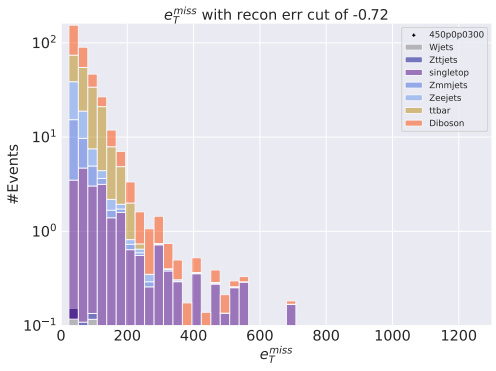


(c)

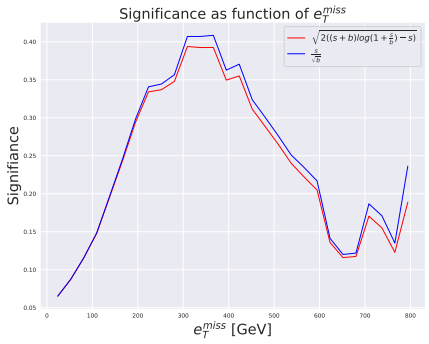
Figure 26: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 800p50 model is used.



(a)

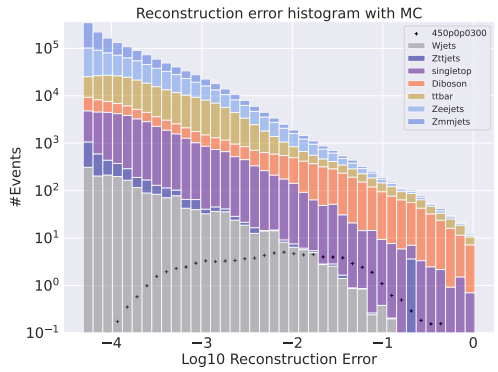


(b)

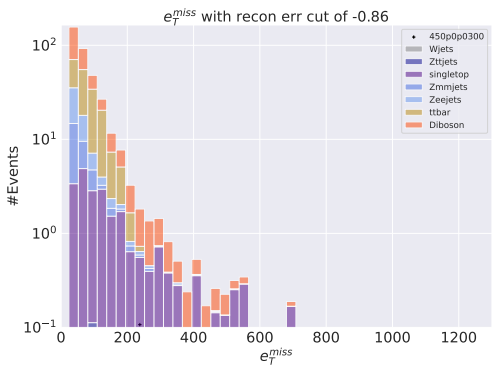


(c)

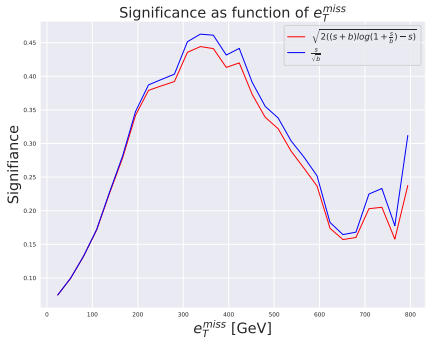
Figure 27: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a)

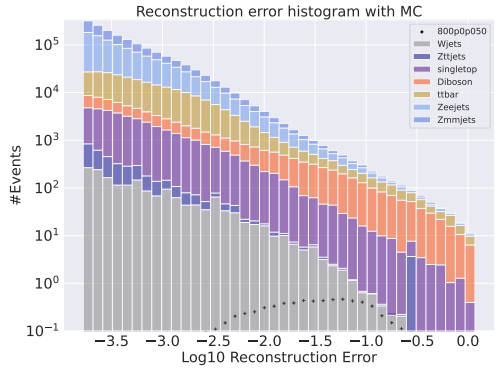


(b)

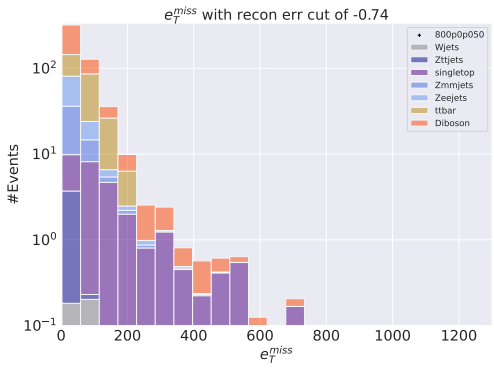


(c)

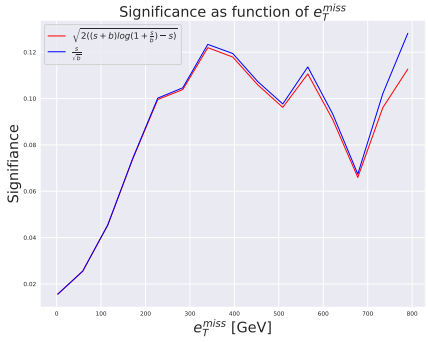
Figure 28: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a)

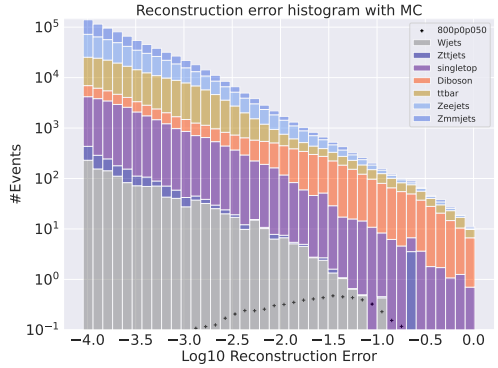


(b)

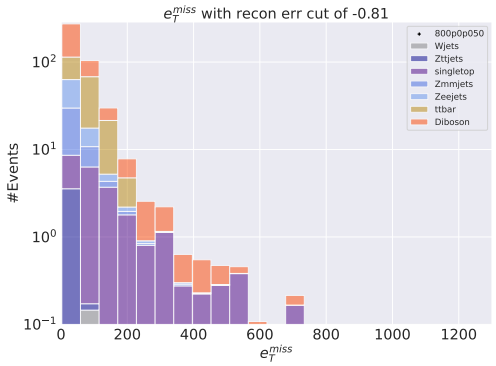


(c)

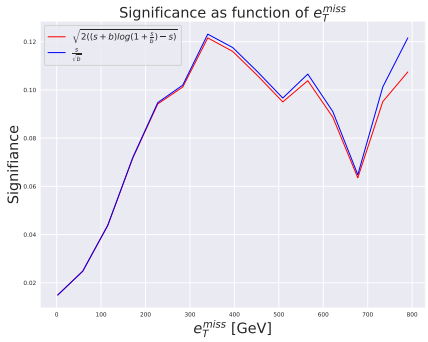
Figure 29: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 800p50 model is used.



(a)



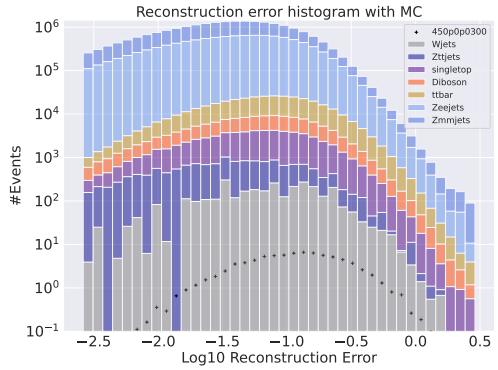
(b)



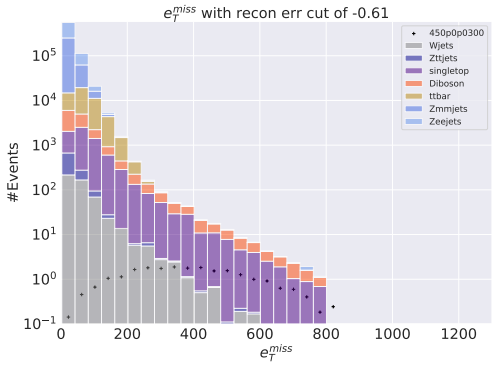
(c)

Figure 30: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 800p50 model is used.

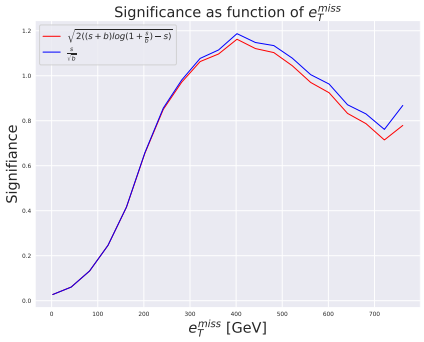
Variational autoencoder output



(a)

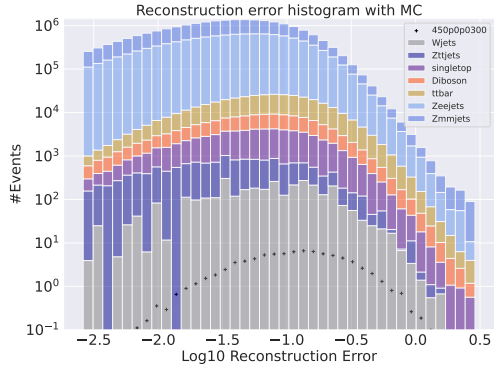


(b)

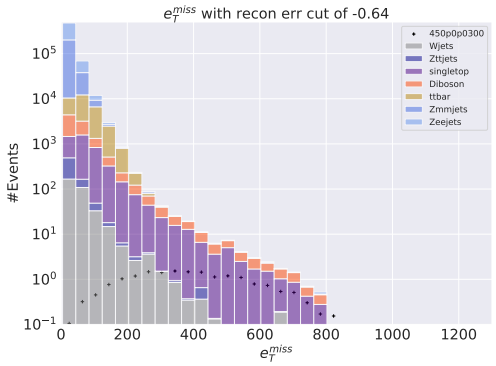


(c)

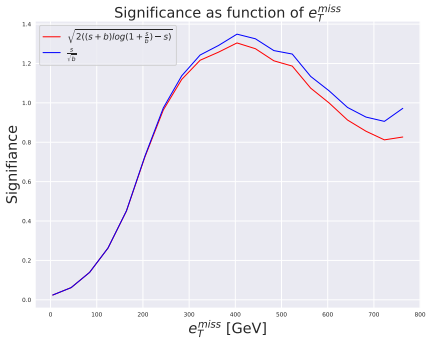
Figure 31: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a)

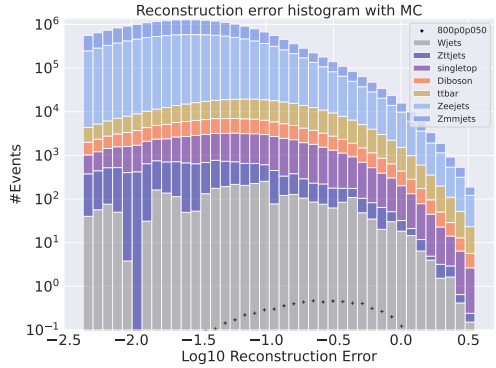


(b)

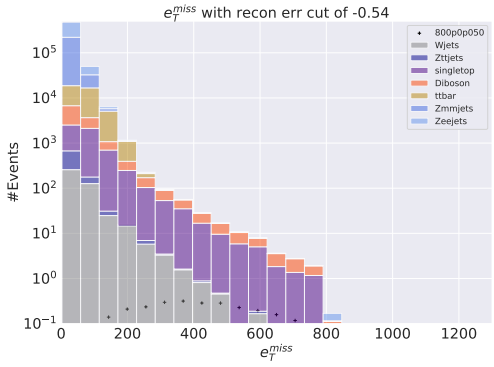


(c)

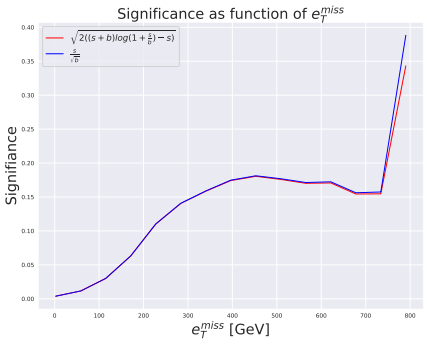
Figure 32: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a)

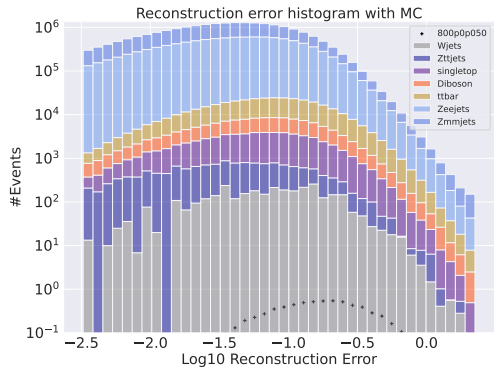


(b)

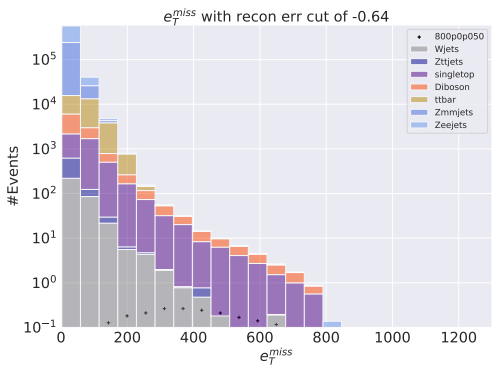


(c)

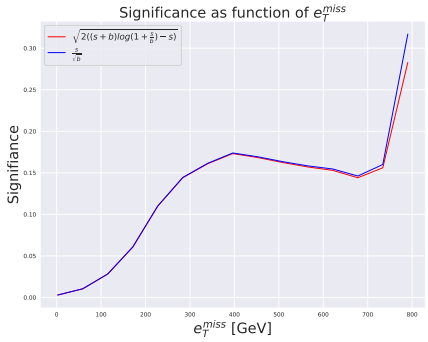
Figure 33: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 800p50 model is used.



(a)

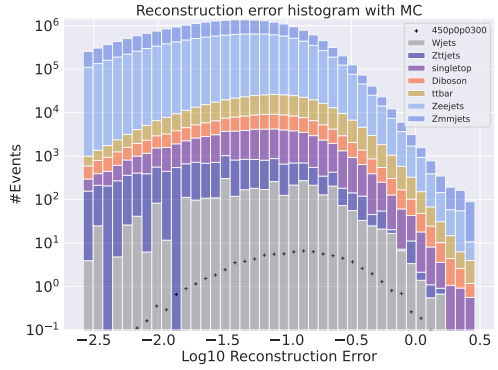


(b)

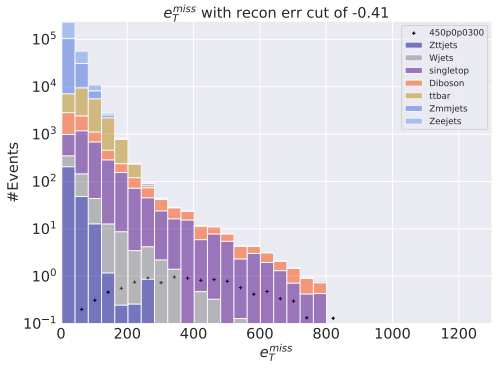


(c)

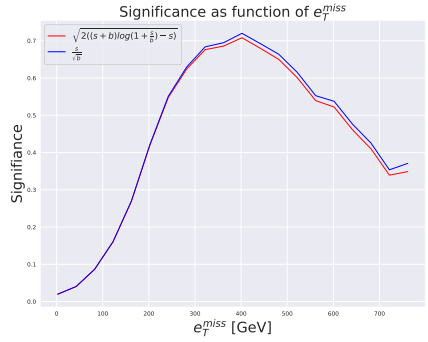
Figure 34: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 800p50 model is used.



(a)

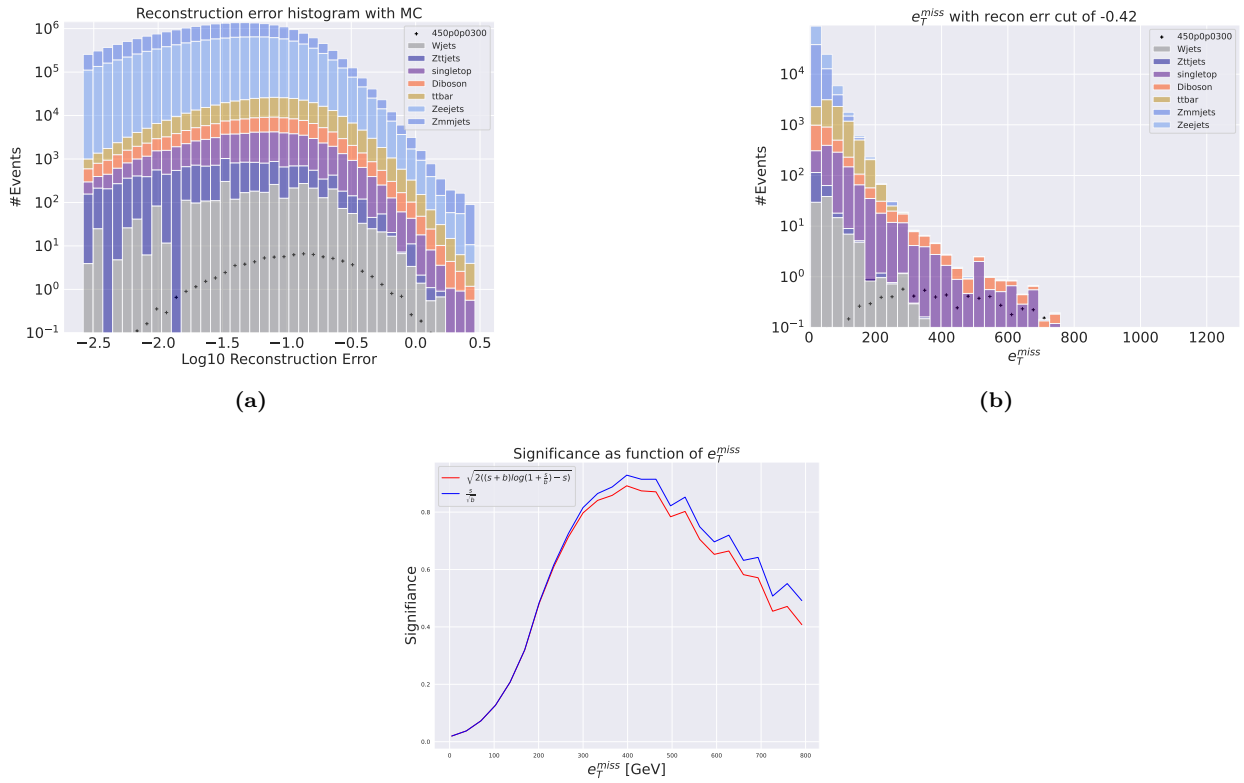


(b)



(c)

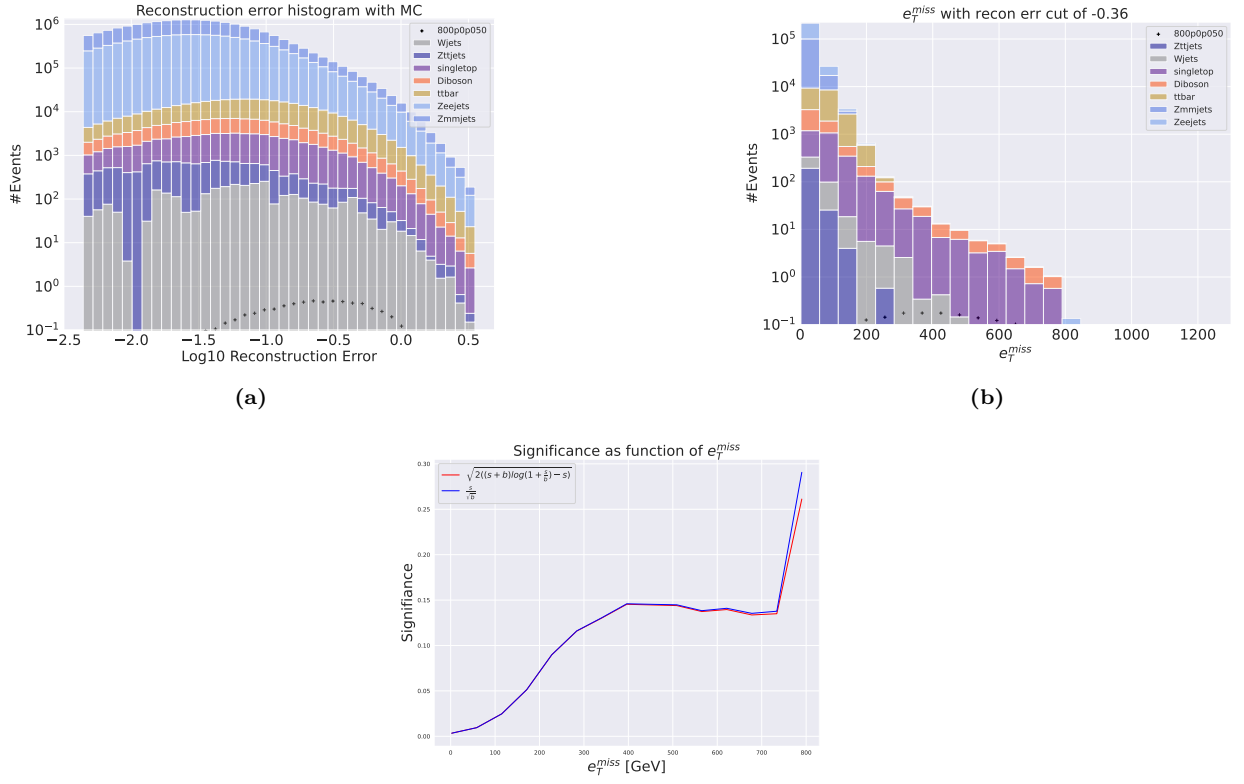
Figure 35: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



(a) **(b)**

(c)

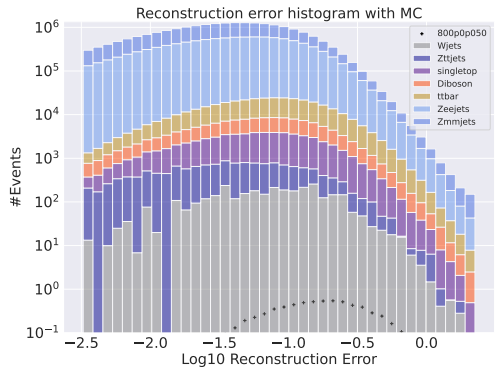
Figure 36: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 450p300 model is used.



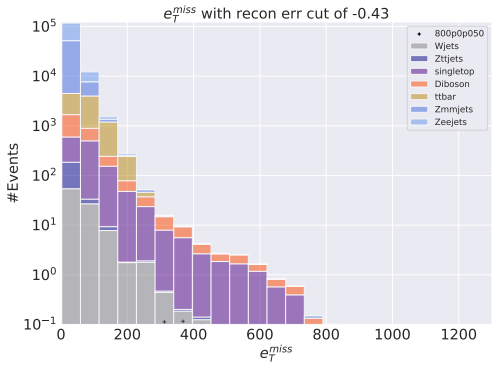
(a) **(b)**

(c)

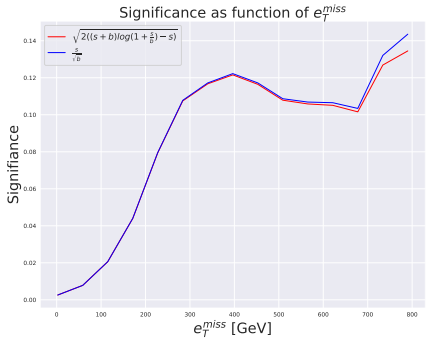
Figure 37: Reconstruction error, e_T^{miss} signal region, m_{ll} signal region and significance as function of e_T^{miss} for the deep regular autoencoder. Here the SUSY 800p50 model is used.



(a)



(b)



(c)

Figure 38: Reconstruction error, e_T^{miss} signal region, m_{lll} signal region and significance as function of e_T^{miss} for the shallow regular autoencoder. Here the SUSY 800p50 model is used.

Appendix B

B.1 Algorithmic implementation

The RMM as a set of features were a bit tricky to implement, and one way to implement it was to use a dictionary containing the names of features already in the RDataFrame set. RDataFrame allows for custom c++ functions to be run on the entire dataframe, and because almost all the features in the RMM use the kinematical variables for each particle they were needed to be accessed. Each element in the dictionary contains a number ID corresponding to the column it belongs to, a name, the 4 kinematic variables p_T, η, ϕ and E, and the rank within its particle type. Thus, ele_0 is the first electron, and its rank is 0.

```
rmm_structure = {
    1: ["ljet_0", "jetPt[ljet]", "jetEta[ljet]", "jetPhi[ljet]", "jetM[ljet]", 0,],
    2: [ "ljet_1", "jetPt[ljet]", "jetEta[ljet]", "jetPhi[ljet]", "jetM[ljet]", 1,],
    3: [ "ljet_2", "jetPt[ljet]", "jetEta[ljet]", "jetPhi[ljet]", "jetM[ljet]", 2,],
    4: [ "ljet_3", "jetPt[ljet]", "jetEta[ljet]", "jetPhi[ljet]", "jetM[ljet]", 3,],
    5: [ "ljet_4", "jetPt[ljet]", "jetEta[ljet]", "jetPhi[ljet]", "jetM[ljet]", 4,],
    6: [ "ljet_5", "jetPt[ljet]", "jetEta[ljet]", "jetPhi[ljet]", "jetM[ljet]", 5,],
    7: [ "bjet_0", "jetPt[bjet77]", "jetEta[bjet77]", "jetPhi[bjet77]", "jetM[bjet77]", 0,],
    8: [ "bjet_1", "jetPt[bjet77]", "jetEta[bjet77]", "jetPhi[bjet77]", "jetM[bjet77]", 1,],
    9: [ "bjet_2", "jetPt[bjet77]", "jetEta[bjet77]", "jetPhi[bjet77]", "jetM[bjet77]", 2,],
    10: [ "bjet_3", "jetPt[bjet77]", "jetEta[bjet77]", "jetPhi[bjet77]", "jetM[bjet77]", 3,],
    11: [ "bjet_4", "jetPt[bjet77]", "jetEta[bjet77]", "jetPhi[bjet77]", "jetM[bjet77]", 4,],
    12: [ "bjet_5", "jetPt[bjet77]", "jetEta[bjet77]", "jetPhi[bjet77]", "jetM[bjet77]", 5,],
    13: [ "ele_0", "lepPt[ele_SG]", "lepEta[ele_SG]", "lepPhi[ele_SG]", "lepM[ele_SG]", 0,],
    14: [ "ele_1", "lepPt[ele_SG]", "lepEta[ele_SG]", "lepPhi[ele_SG]", "lepM[ele_SG]", 1,],
    15: [ "ele_2", "lepPt[ele_SG]", "lepEta[ele_SG]", "lepPhi[ele_SG]", "lepM[ele_SG]", 2,],
    16: [ "ele_3", "lepPt[ele_SG]", "lepEta[ele_SG]", "lepPhi[ele_SG]", "lepM[ele_SG]", 3,],
    17: [ "ele_4", "lepPt[ele_SG]", "lepEta[ele_SG]", "lepPhi[ele_SG]", "lepM[ele_SG]", 4,],
    18: [ "muo_0", "lepPt[muo_SG]", "lepEta[muo_SG]", "lepPhi[muo_SG]", "lepM[muo_SG]", 0,],
    19: [ "muo_1", "lepPt[muo_SG]", "lepEta[muo_SG]", "lepPhi[muo_SG]", "lepM[muo_SG]", 1,],
    20: [ "muo_2", "lepPt[muo_SG]", "lepEta[muo_SG]", "lepPhi[muo_SG]", "lepM[muo_SG]", 2,],
    21: [ "muo_3", "lepPt[muo_SG]", "lepEta[muo_SG]", "lepPhi[muo_SG]", "lepM[muo_SG]", 3,],
    22: [ "muo_4", "lepPt[muo_SG]", "lepEta[muo_SG]", "lepPhi[muo_SG]", "lepM[muo_SG]", 4,],
}
```

The dictionary is then used in the nested for loop below. The loop is partitioned into several scenarios. Firstly, the first element in the matrix is the e_T^{miss}. Once that is set, the loop has three cases to check, the upper triangle, the lower triangle and the diagonal. The upper triangle is related to masses, and the lower triangle is related to longitudal properties. Using this, the number ID helps ID which particle(s) to use and where to put the properties calculated based on them.

```
for row in range(N_row):
    if row == 0:
        # Calculate e_T^miss and m_T for all objects
        for column in range(N_col):
            if column == 0:
                # Set e_T_miss
                df[k] = df[k].Define("e_T_miss", "met_Et")
            else:
                # Set m_T for all particles
                name, pt,eta,phi,m,index = rmm_structure[column]
                df[k] = df[k].Define(
                    f"m_T_{name}", f"getM_T({pt},{eta},{phi},{m},{index})"
                )
    else:
        # Calculate rest of matrix
        for column in range(N_col):
            if column == 0:
                # Set h_L for all particles
                name, pt,eta,phi,m,index = rmm_structure[row]

                df[k] = df[k].Define(
                    f"h_L_{name}", f"geth_L({pt},{eta},{phi},{m},{index})"
                )
                elif column == row:
                name, pt,eta,phi,m,index = rmm_structure[column]
                if index == 0:
                    # If particle is the first of its type, calculate e_T of particle
                    df[k] = df[k].Define(
                        f"e_T_{name}", f"getET_part({pt},{m},{index})"
                    )
                else:
                    # If particle is not the first of its type, calculate the difference in e_T
                    df[k] = df[k].Define(
                        f"delta_e_t_{name}", f"delta_e_T({pt},{m},{index})"
                    )


            elif column > row:
                # For invariant mass
                # Particle 1
                name1, pt1,eta1,phi1,m1,index1 = rmm_structure[row]

                # Particle 2
                name2,pt2,eta2,phi2,m2,index2 = rmm_structure[column]

                histo_name = f"m_{name1}_{name2}"
                df[k] = df[k].Define(
                    histo_name,
                    f"getM({pt1},{eta1}, {phi1}, {m1}, {pt2}, {eta2}, {phi2}, {m2}, {index1}, {
    index2})",
                )
            elif row > column:
                # For h longitudal stuff
                # Particle 1
                name1, pt1,eta1,phi1,m1,index1 = rmm_structure[row]

                # Particle 2
                name2,pt2,eta2,phi2,m2,index2 = rmm_structure[column]

                histo_name = f"h_{name1}_{name2}"
                df[k] = df[k].Define(
                    f"{histo_name}",
                    f"geth({pt1},{eta1}, {phi1}, {m1},  {pt2}, {eta2}, {phi2}, {m2},  {index1}, {
    index2})",
                )
```

Bibliography

[1] L. Weng, *From autoencoder to beta-vae*, *lilianweng.github.io* (2018) .

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

[3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan et al., *Pytorch: An imperative style, high-performance deep learning library*, 2019.

[4] V. Chandola, A. Banerjee and V. Kumar, *Anomaly detection: A survey*, *ACM Comput. Surv.* **41** (07, 2009) .

[5] T. Chen and C. Guestrin, *XGBoost*, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, aug, 2016. DOI.

[6] C. Cortes and V. Vapnik, *Support-vector networks*, *Machine learning* **20** (1995) 273–297.

[7] S. Frette, W. Hirst and M. M. Jensen, *A computational analysis of a dense feed forward neural network for regression and classification type problems in comparison to regression methods*, .

[8] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016.

[9] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. 10.48550/ARXIV.1412.6980.

[10] D. Rumelhart, G. Hinton and R. Williams, *Learning representations by back-propagating errors*, *Nature* (1986) .

[11] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2013. 10.48550/ARXIV.1312.6114.

[12] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh and A. Talwalkar, *Hyperband: A novel bandit-based approach to hyperparameter optimization*, .

[13] K. Jamieson and A. Talwalkar, *Non-stochastic best arm identification and hyperparameter optimization*, 2015. 10.48550/ARXIV.1502.07943.

[14] T. Fawcett, *An introduction to roc analysis*, *Pattern Recognition Letters* **27** (2006) 861–874.

[15] M. Bugge, *Discovery and exclusion in high energy physics*, 2022.

[16] A. Pich, *The Standard Model of Electroweak Interactions; rev. version*, .

[17] M. E. Peskin and D. V. Schroeder, *An Introduction to Quantum Field Theory*. Westview Press, 1995.

[18] M. Thomson, *Modern particle physics*. Cambridge University Press, New York, 2013.

[19] S. Weinberg, *The Quantum Theory of Fields*, vol. 1. Cambridge University Press, 1995, 10.1017/CBO9781139644167.

[20] K. Kumericki, *Feynman diagrams for beginners*, 2016. 10.48550/ARXIV.1602.04182.

[21] M. Aker, A. Beglarian, J. Behrens, A. Berlev, U. Besserer, B. Bieringer et al., *Direct neutrino-mass measurement with sub-electronvolt sensitivity*, *Nature Physics* **18** (Feb, 2022) 160–166.

[22] E. Gramstad, *Searches for Supersymmetry in Di-Lepton Final States with the ATLAS Detector at $\sqrt{s}$ = 7 TeV*, 2013.

[23] ATLAS collaboration, A. Airapetian, V. Grabsky, H. Hakopian, A. Vartapetian, B. Dick, F. Fares et al., *ATLAS detector and physics performance: Technical Design Report, 1*. Technical design report. ATLAS. CERN, Geneva, 1999.

[24] D. Green, *High p_T Physics at Hadron Colliders*. Cambridge University Press, 2004.

[25] ATLAS collaboration, S. Mehlhase, *ATLAS detector slice (and particle visualisations)*, .

[26] S.-M. Wang, *ATLAS Computing and Data Preparation. ATLAS Induction Day + Software Tutorial*, .

[27] C. Bernius, *ATLAS Trigger and Data Acquisition. ATLAS Induction Day + Software Tutorial*, .

[28] T. P. S. Gillam, *Identifying fake leptons in ATLAS while hunting SUSY in 8 TeV proton-proton collisions*. PhD thesis, Cambridge U., 2015. 10.17863/CAM.16619.

[29] E. Manca and E. Guiraud, *Using RDataFrame, ROOT's declarative analysis tool, in a CMS physics study. Using RDataFrame, ROOT's declarative analysis tool, in a CMS physics study*, .

[30] The pandas development team, *pandas-dev/pandas: Pandas*, Feb., 2020. 10.5281/zenodo.3509134.

[31] The HDF Group, *Hierarchical Data Format, version 5*, 1997-2022.

[32] S. Chekanov, *Imaging particle collision data for event classification using machine learning*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **931** (jul, 2019) 92–99.

[33] S. V. Chekanov, *Machine learning using rapidity-mass matrices for event classification problems in HEP*, *Universe* **7** (jan, 2021) 19.

[34] R. Santos, M. Nguyen, J. Webster, S. Ryu, J. Adelman, S. Chekanov et al., *Machine learning techniques in searches for $t\bar{t}h$ in the $h \to b\bar{b}$ decay channel*, *Journal of Instrumentation* **12** (apr, 2017) P04014–P04014.

[35] R. S. Geiger, D. Cope, J. Ip, M. Lotosh, A. Shah, J. Weng et al., *"garbage in, garbage out" revisited: What do machine learning application papers report about human-labeled training data?*, *CoRR* **abs/2107.02278** (2021) , [2107.02278].

[36] M. Aaboud, , G. Aad, B. Abbott, J. Abdallah, O. Abdinov et al., *Performance of the ATLAS trigger system in 2015*, *The European Physical Journal C* **77** (may, 2017) .

[37] G. Aad, B. Abbott, D. Abbott, A. A. Abud, K. Abeling, D. Abhayasinghe et al., *Performance of the ATLAS muon triggers in run 2*, *Journal of Instrumentation* **15** (sep, 2020) P09015–P09015.

[38] G. Aad, , B. Abbott, D. C. Abbott, A. A. Abud, K. Abeling et al., *Performance of electron and photon triggers in ATLAS during LHC run 2*, *The European Physical Journal C* **80** (jan, 2020) .

[39] F. Chollet et al., *Keras*, 2015.

[40] Plotly Technologies Inc., *Collaborative data science*, 2015.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., *Scikit-learn: Machine learning in Python*, *Journal of Machine Learning Research* **12** (2011) 2825–2830.

[42] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau et al., *Array programming with NumPy*, *Nature* **585** (Sept., 2020) 357–362.

[43] R. Vink, S. de Gooijer, A. Beedie, J. van Zundert, G. Hulselmans, C. Grinstead et al., *pola-rs/polars: Python polars 0.16.14*, Mar., 2023. 10.5281/zenodo.7744139.

[44] A. Asperti and M. Trentin, *Balancing reconstruction error and kullback-leibler divergence in variational autoencoders*, 2020. 10.48550/ARXIV.2002.07514.

[45] LHC, *Lhc nominal luminosity projection*, 2013.

[46] R. Okuta, Y. Unno, D. Nishino, S. Hido and C. Loomis, *Cupy: A numpy-compatible library for nvidia gpu calculations*, in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

[47] ATLAS Collaboration collaboration, *Review of the 13 TeV ATLAS Open Data release*, tech. rep., CERN, Geneva, Jan, 2020.