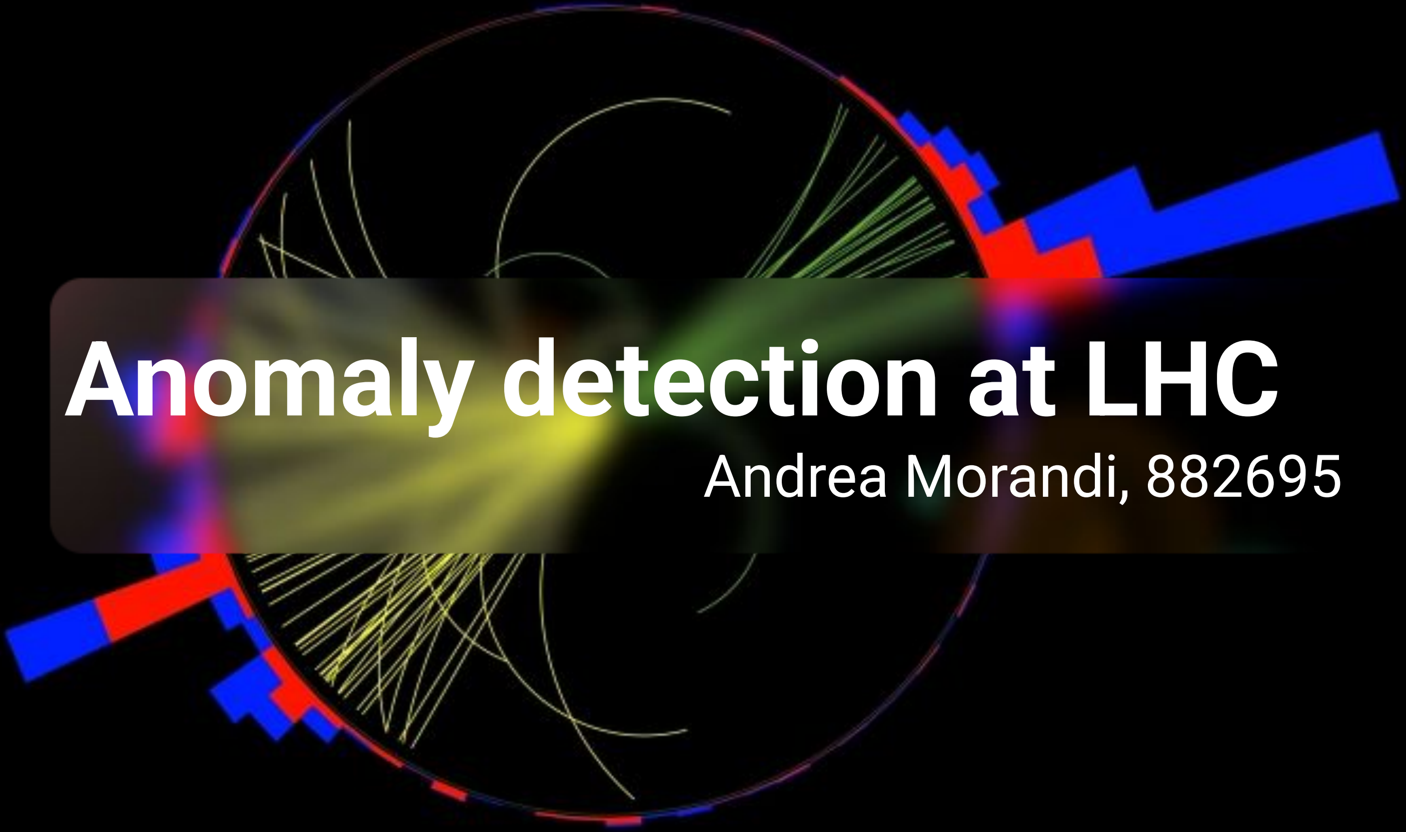
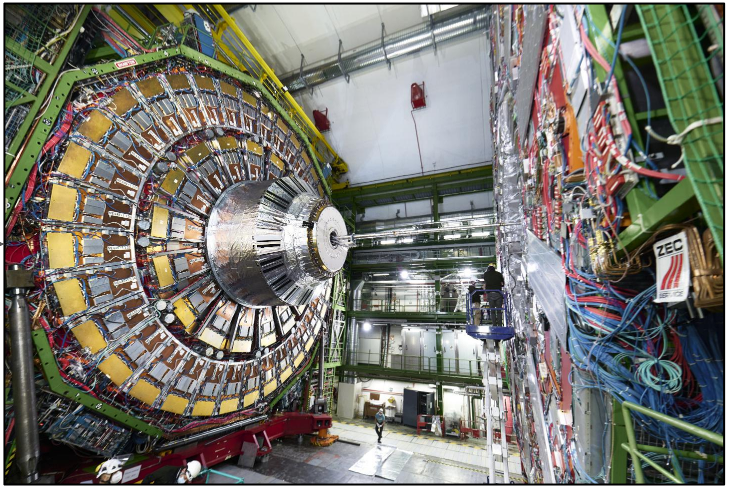


Anomaly detection at LHC

Andrea Morandi, 882695

Context: Large Hadron Collider

- At the Large Hadron Collider (LHC), **protons are collided every 25 ns at √s =13 TeV**
- After a collision, many particles are produced and detected by **particle detectors**
- We can group them into four categories: **electrons (e), muons (μ), hadronic jets, and neutrinos (ν).**
- **Neutrinos escape detectors** producing an energy unbalance → **missing transverse energy (MET)**

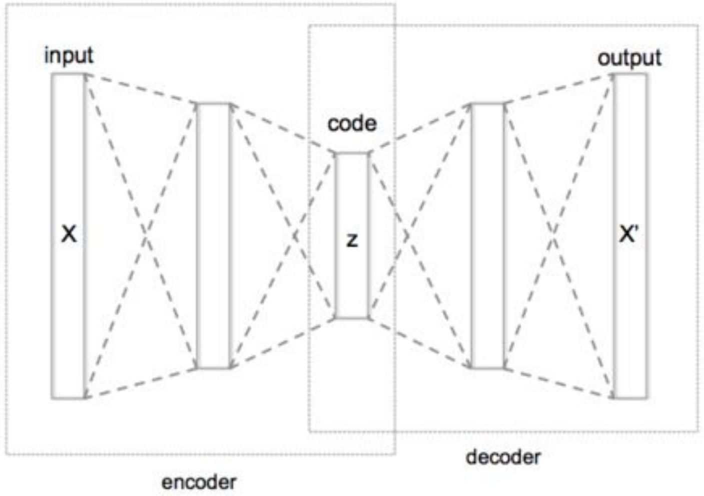


- Given the large number of events, it is not possible to collect them all
- A selection is made using **deterministic trigger algorithms** design to identify peculiar event topologies predicted by the Standard Model (SM) or a chosen subset of possible “new physics” theories (BSM)

- BSM events may not satisfy trigger conditions and therefore they could be missed.
- To overcome this limitation, **Machine Learning (ML) techniques can be use to search for BSM “in the form of anomaly detection”**

Machine Learning approach: Autoencoder

- Use an autoencoder trained to accurately reconstruct SM events, while failing to reconstruct BSM events.
- **Why an autoencoder?**
  Because it compresses input data into a lower-dimensional latent space, learning a **compact representation of the underlying structure of the data**.
- The **latent space can be interpreted as a manifold that captures the essential features and structure of SM events**. BSM events, which lie outside this manifold, are poorly reconstructed, leading to a higher reconstruction error.
- **The reconstruction error act as an anomaly score**.



Available datasets

Five datasets are considered in this analysis:

- **Simulated SM processes** (13,451,915 events): these data are used to train, in a supervised way, an algorithm able to recognize SM signals.
- $h^0 \rightarrow \tau\tau$ (340,544 events)
- $h \rightarrow \tau\nu$ (55,969 events)
- $A \rightarrow 4l$ (691,283 events)
- $LQ \rightarrow \tau b$ (760,262 events)

} **Simulated events never observed experimentally but theoretically hypothesized**, used to test whether the network can identify BSM data as “anomalies”

Where $l = e, \mu$; b is a quark that originates a hadronic jet; and τ is a heavy lepton that can decay in two main ways: $\tau \rightarrow l\nu$ or $\tau \rightarrow 2$ jets.

Knowing that the simulated SM processes involve the production of W and Z bosons, which decay as follows: $W \rightarrow l\nu$ and $Z \rightarrow ll$ (with $l = e, \mu, \tau$), or W, Z $\rightarrow$ 2 jets, it is expected that datasets such as $h^0 \rightarrow \tau\tau$ and $h \rightarrow \tau\nu$ will be more difficult to distinguish from SM events, as they resemble known processes more closely than the other two datasets.

Govorkova, E., Puljak, E., Aarrestad, T., Pierini, M., Woźniak, K. A., & Ngadiuba, J. (2021). *LHC physics dataset for unsupervised New Physics detection at 40 MHz*. Advances in Neural Information Processing Systems. https://arxiv.org/abs/2107.02157

Dataset structure

The input events are provided in HDF5 format as structured tables, where each row corresponds to a single particle within an event. Each **event can contain up to a maximum of 19 particles**, with their associated features:

- the **transverse momentum (p_T)**;
- the **pseudorapidity (η)**, which is related to the polar angle (θ) by: η=log(tan(θ/2));
- the **azimuthal angle (ϕ)**;
- the class, a number that identify the type of particle.

Events with fewer than 19 particles are zero-padded to maintain a consistent input shape, resulting in **sparse matrices**.

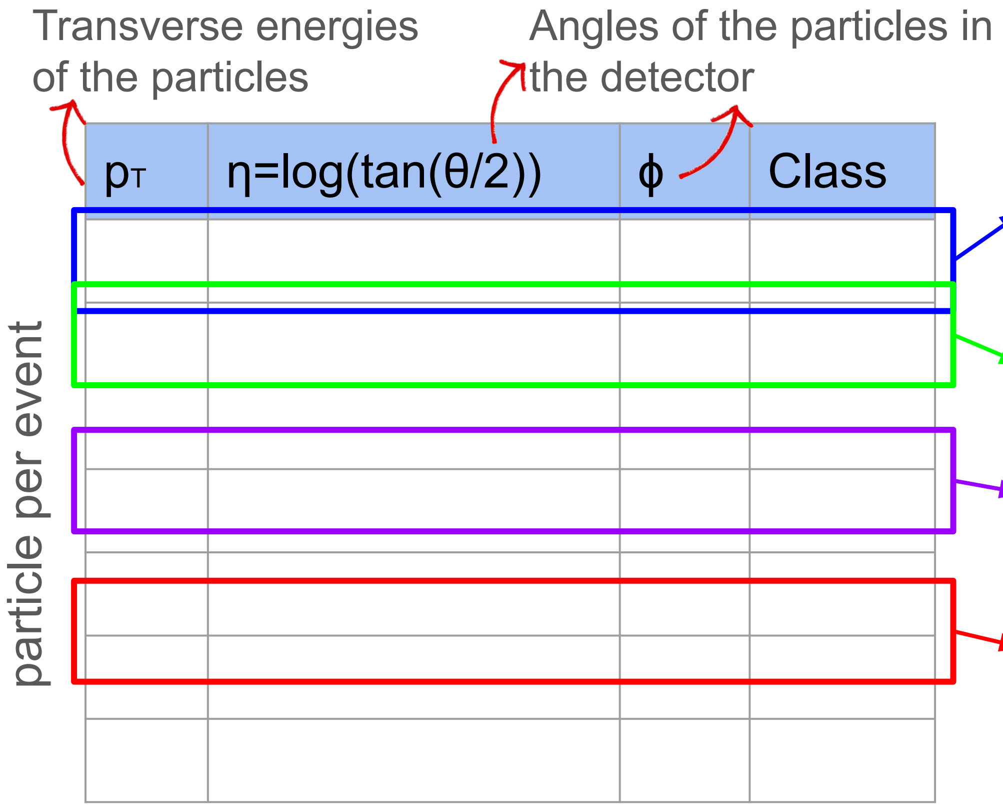
The features under study are on different scales, so the **data were normalized** to make them uniform. Each feature is normalized separately.

mean

$$z_i = \frac{x_i - \bar{x}}{\delta x}$$

standard deviation

Event structure


Transverse energies of the particles
Angles of the particles in the detector
p_T
$\eta=\log(\tan(\theta/2))$
ϕ
Class
particle per event

19x4 matrices

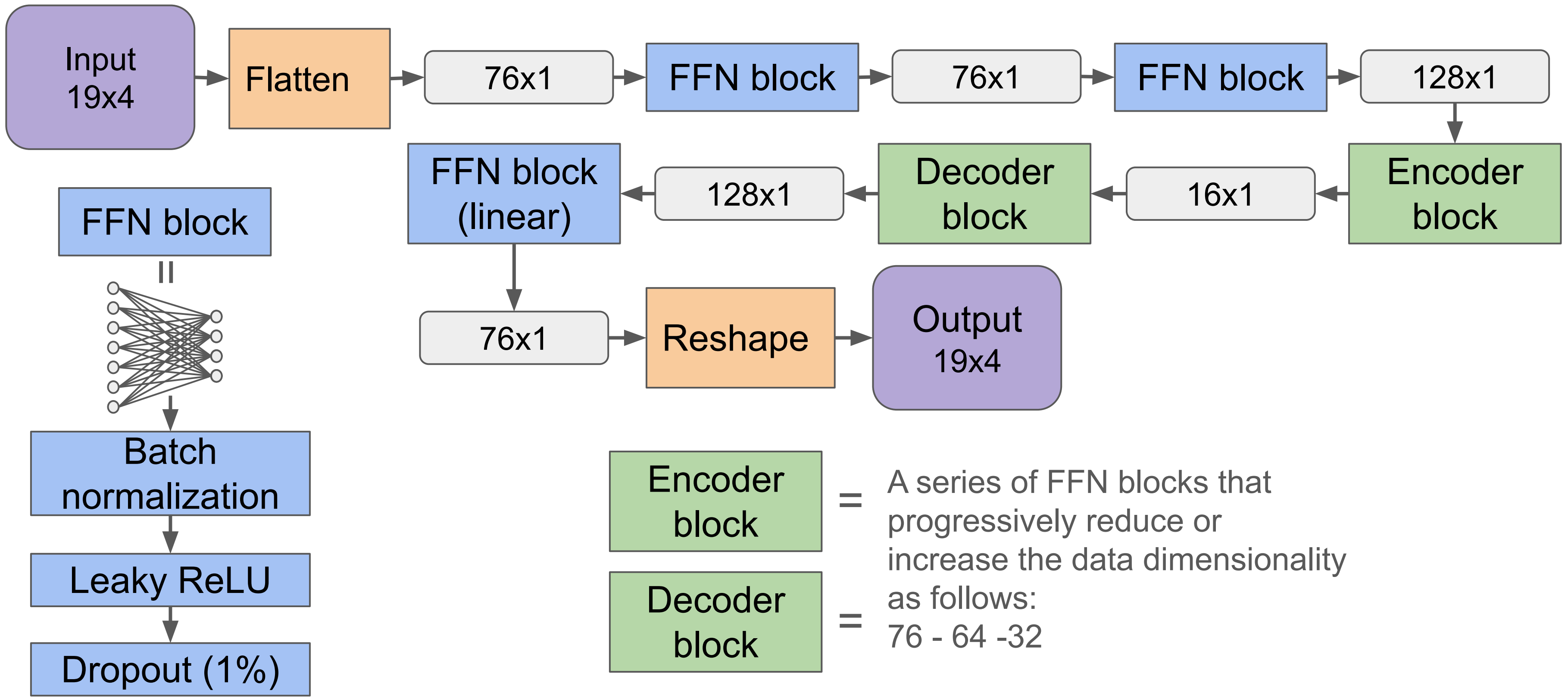
Line: 0, MET, Class id = 1

Line: 1-4, electrons, Class id = 2

Line: 5-8, muons, Class id = 3

Line: 9-18, hadronic jets, Class id: 4

Anomaly detection via FFN: Architecture



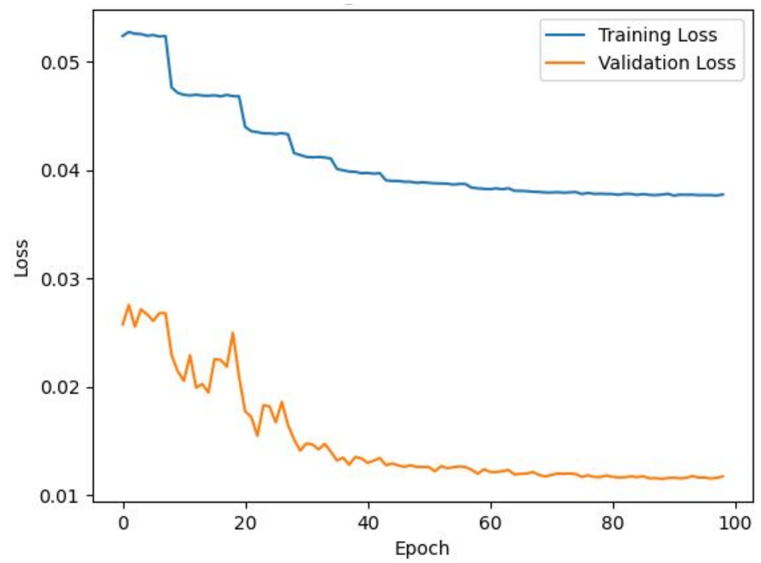
A series of FFN blocks that progressively reduce or increase the data dimensionality as follows:
76 - 64 -32

Anomaly detection via FFN: Training

- Loss function: **Mean Square Error (MSE)**.
- Regularizer: **L2 Regularization**, with weight α = 0.0001.

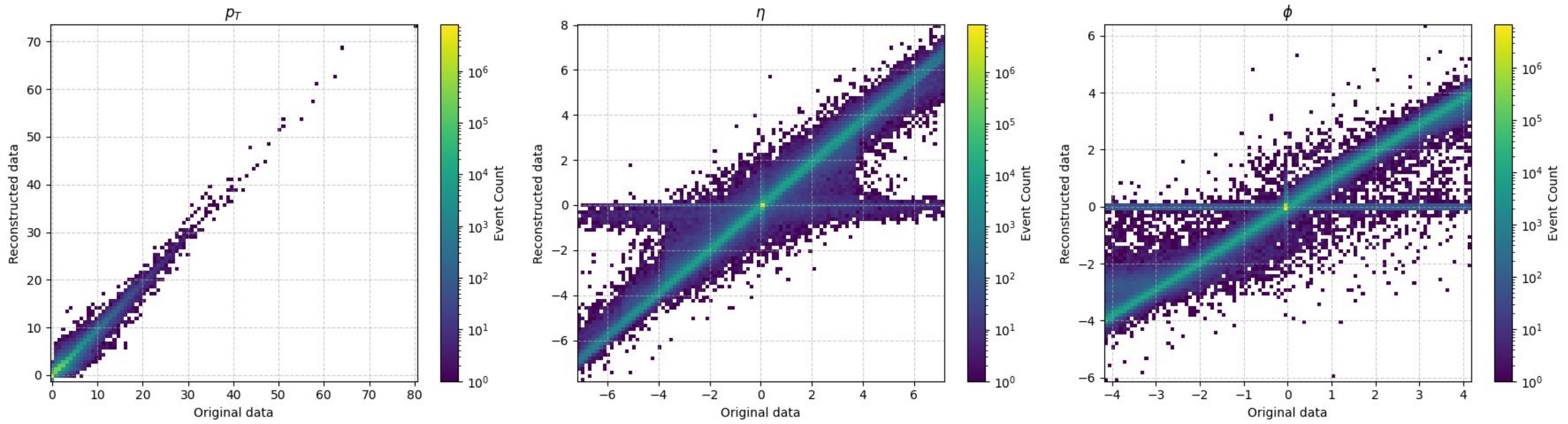
$$\mathcal{L}(\theta; X^{true}, X^{predict}) = \mathrm{MSE}(\theta; X^{true}, X^{predict}) + \alpha\Omega(\theta)$$

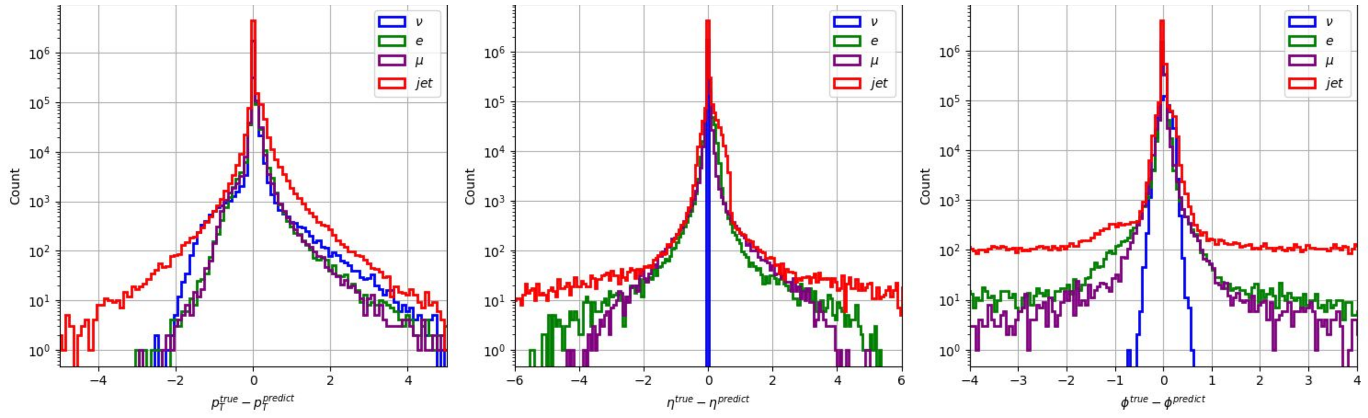
- Batch size: **256**, 2,72e-4 % of the training set.
- Optimizer: **Adam**, with initial learning rate equal to 0.001.
- Learning rate is halved after 5 epochs without a decrease in the validation loss.
- Early stop of the training after 10 epochs without a decrease in the validation loss.



Number of parameters	49,816
Time of training of an epoch (on a T4 GPU).	57-58 s

Anomaly detection via FFN: Reconstructing SM events

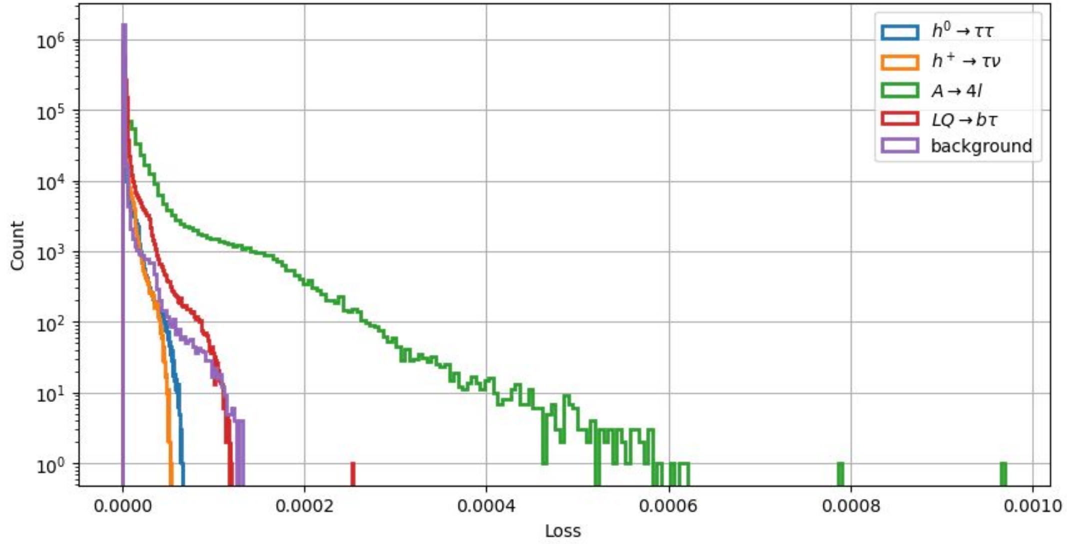




The results show that the **reconstructed data and the original data are not perfectly correlated**, particularly in the cases of η and ϕ, where the network often returns incorrect zero values.

Anomaly detection via FFN: Performance in BSM detection

- To evaluate the performance in the anomaly detection task, the **distance between the reconstructed and the true data** is computed for each particle.
- As shown in the previous slide, the network performs best in reconstructing the transverse momentum (p_T). Therefore, the **Mean Absolute Error** (MAE) is used for p_T to give more weight to outliers.
- For the angular coordinates, η and ϕ, the MSE is used.
- **This analysis is performed on both SM events** (background) **and BSM events**, allowing a comparison of reconstruction quality and anomaly detection capability.

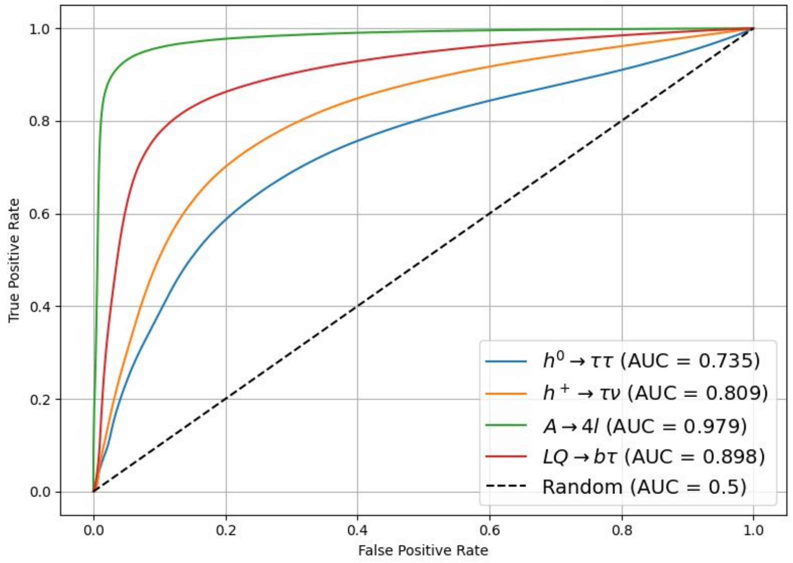


$$\mathcal{L}_j = \frac{1}{N_{event}}\left[|p_{Tj}^{true} - p_{Tj}^{predict}| + (\eta_j^{true} - \eta_j^{predict})^2 + \right.$$
$$\left. + (\phi_j^{true} - \phi_j^{predict})^2\right]$$

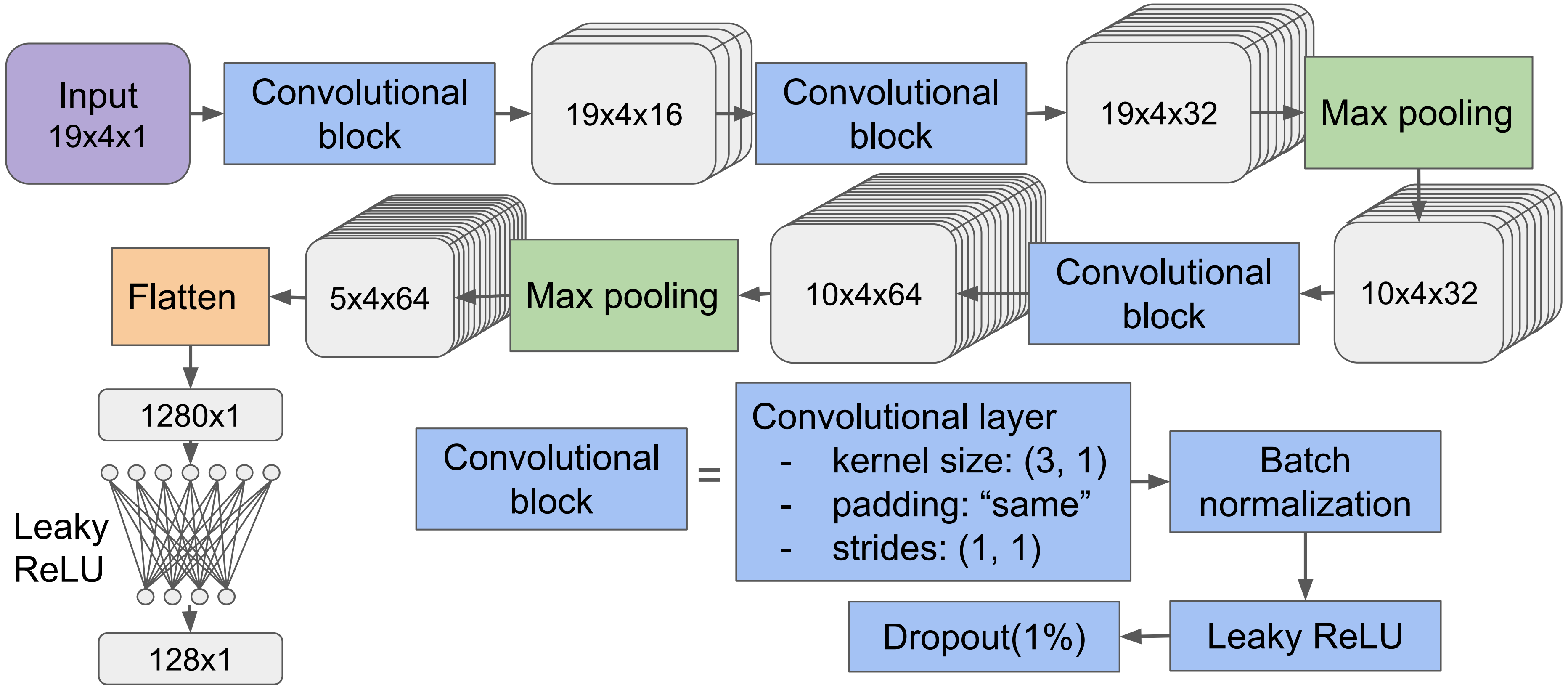
Anomaly detection via FFN: ROC curves

- To evaluate the model's performance, the **Receiver Operating Characteristic** (ROC) curve is used.
- The ROC curve plots the **True Positive Rate** (TPR) against the **False Positive Rate** (FPR).
- The ROC curve can be interpreted as a plot of the statistical power of a test (TPR) versus its **Type I error rate** (FPR).
- It is also possible to compute the **Area Under the Curve** (AUC). The closer the AUC value is to 1, the better the model's performance in distinguishing between normal and anomalous events.

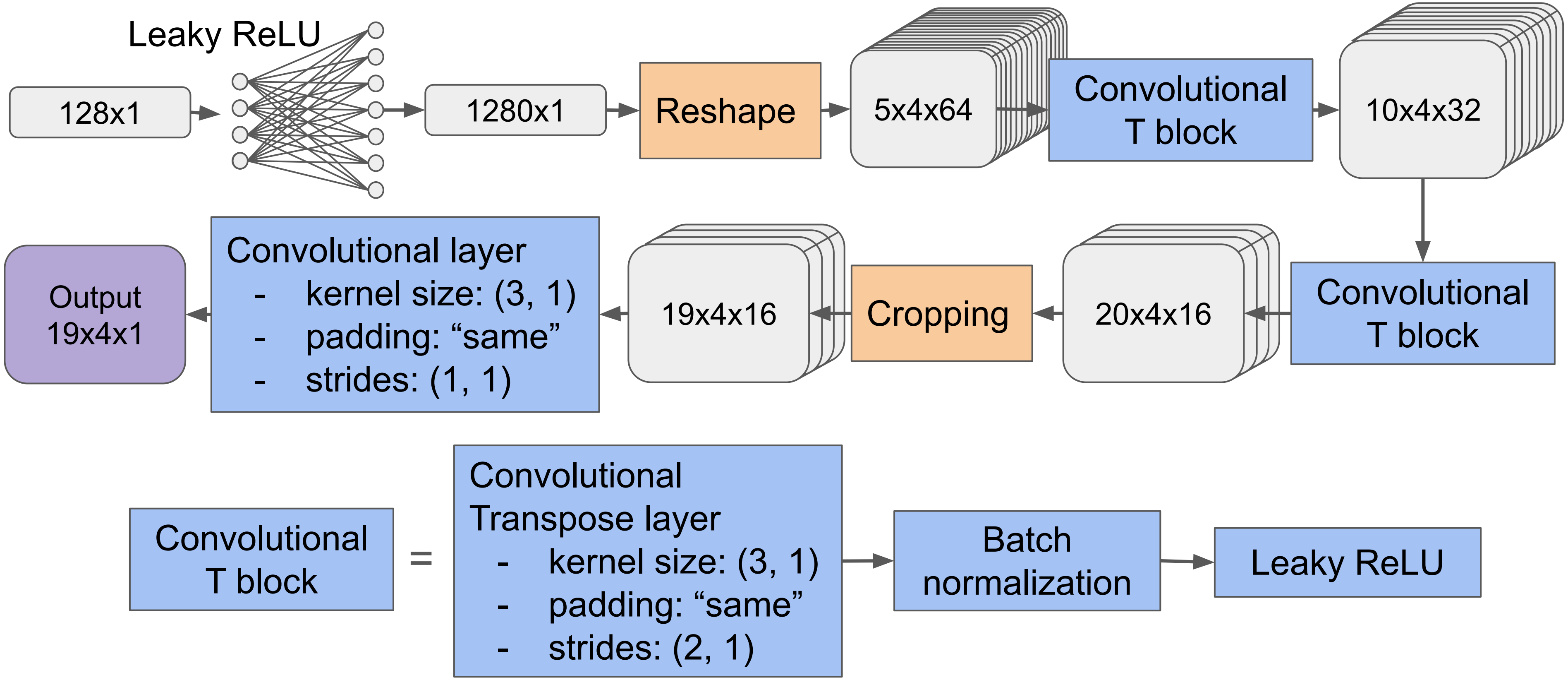
As expected the network reach better performance on the datasets A $\rightarrow$ 4l and LQ $\rightarrow$ τb.



Anomaly detection via CNN: Encoder

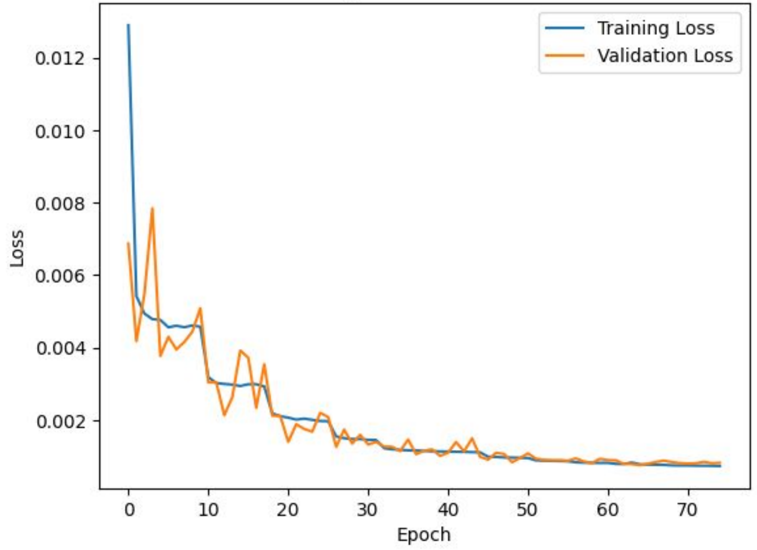


Anomaly detection via CNN: Decoder



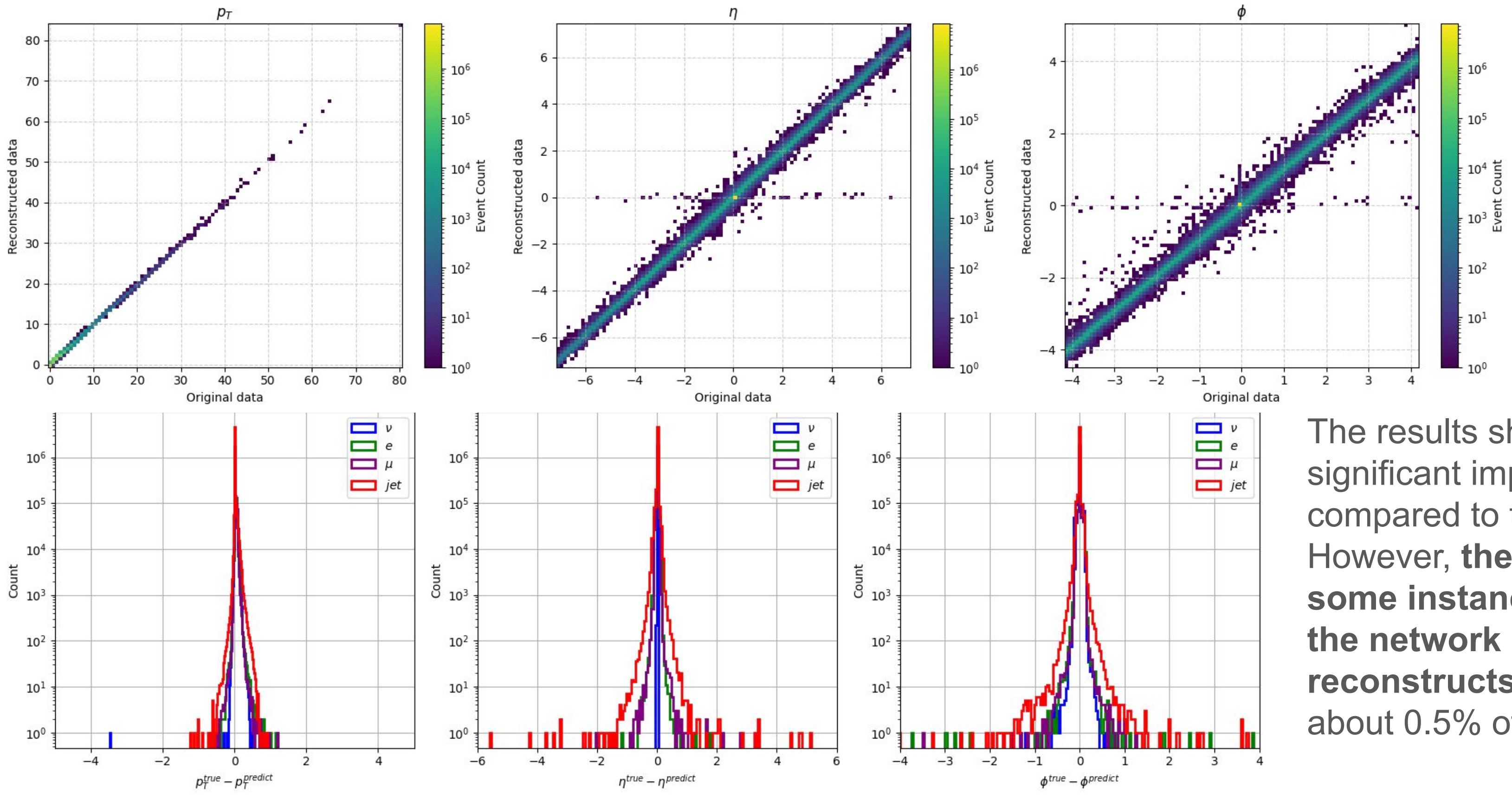
Anomaly detection via CNN: Training

- Loss function: **Mean Square Error (MSE)**.
- Regularizer: **L2 Regularization**, with weight α = 0.0001.
- Batch size: **512**, 5,44e-4 % of the training set.
- Optimizer: **Adam**, with initial learning rate equal to 0.001.
- Learning rate is halved after 5 epochs without a decrease in the validation loss.
- Early stop of the training after 10 epochs without a decrease in the validation loss.



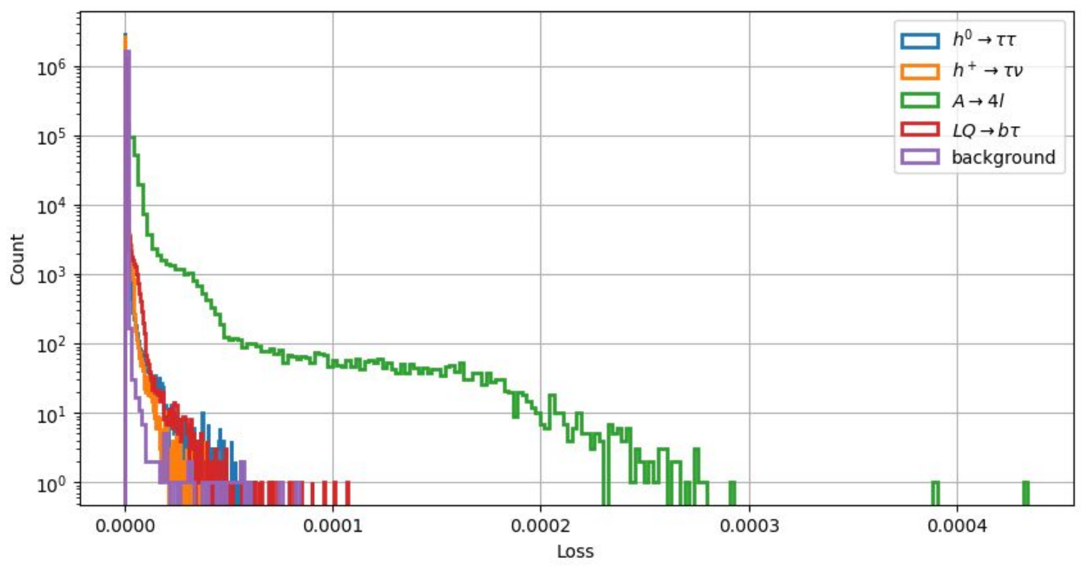
Number of parameters	345,345
Time of training of an epoch (on a T4 GPU).	106-108 s

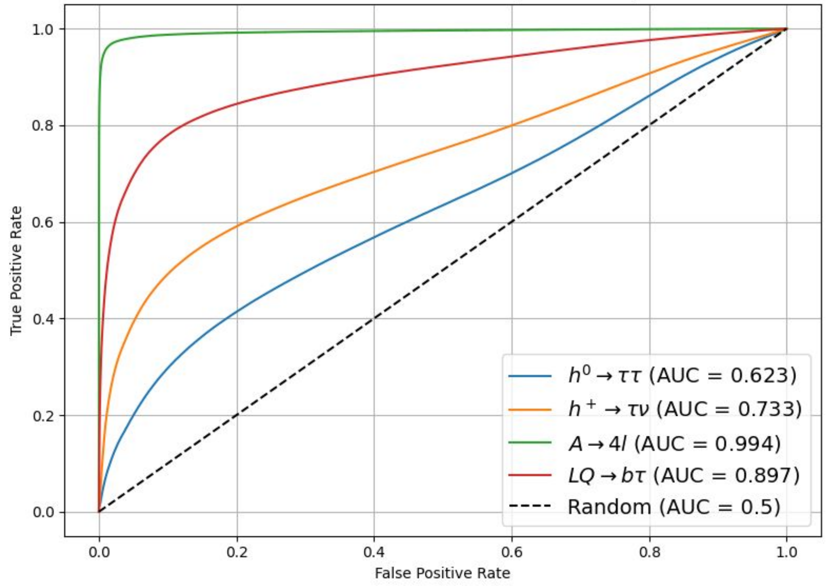
Anomaly detection via CNN: Reconstructing SM events



The results show a significant improvement compared to the FFN case. However, **there are still some instances where the network incorrectly reconstructs zero values**, about 0.5% of the cases.

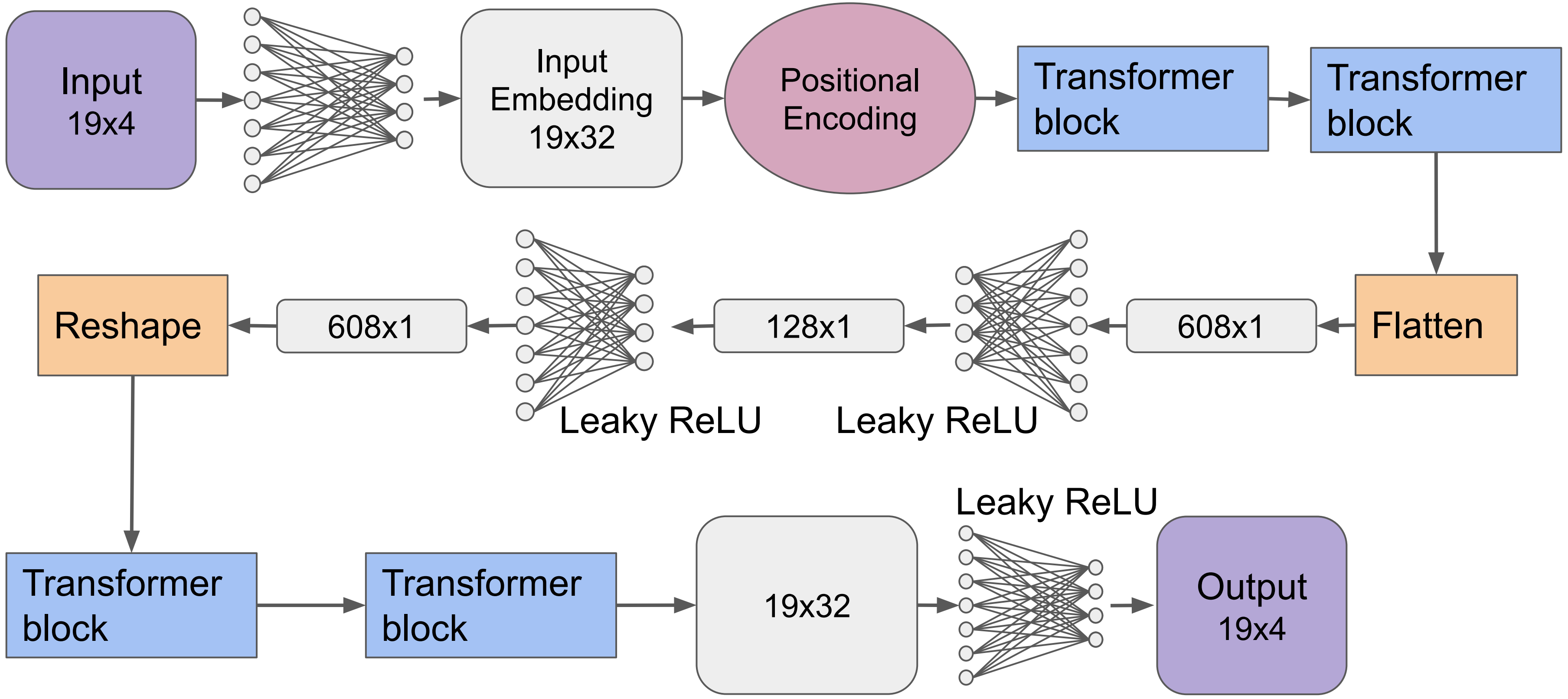
Anomaly detection via CNN: Performance in BSM detection



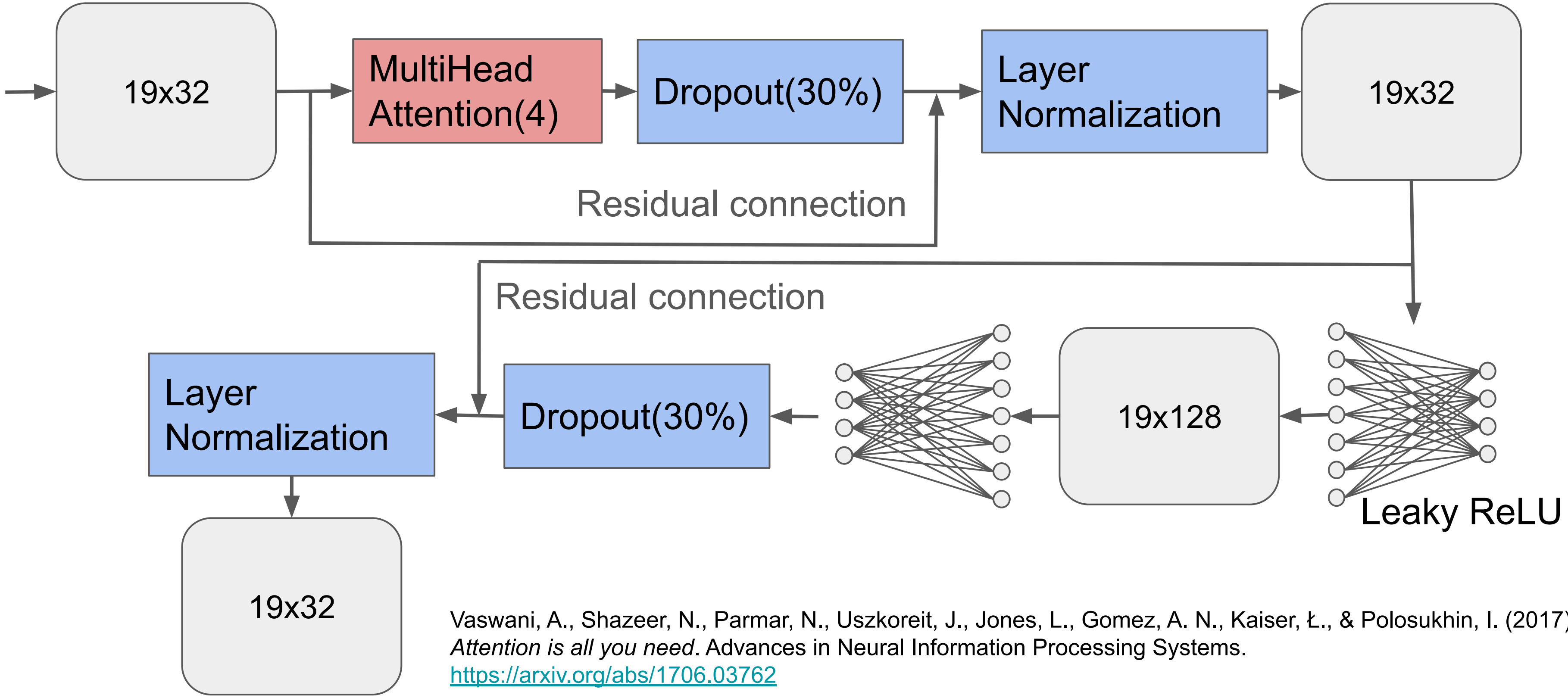


It is used the same approach described for the FFN in slide 8 and 9.
Although the reconstruction of SM signals shows better performance compared to the FFN case, **there is no significant improvement in anomaly detection performance**.
Probably this is linked to the fact that the data are sparse matrices.

Anomaly detection via Transformer: Architecture



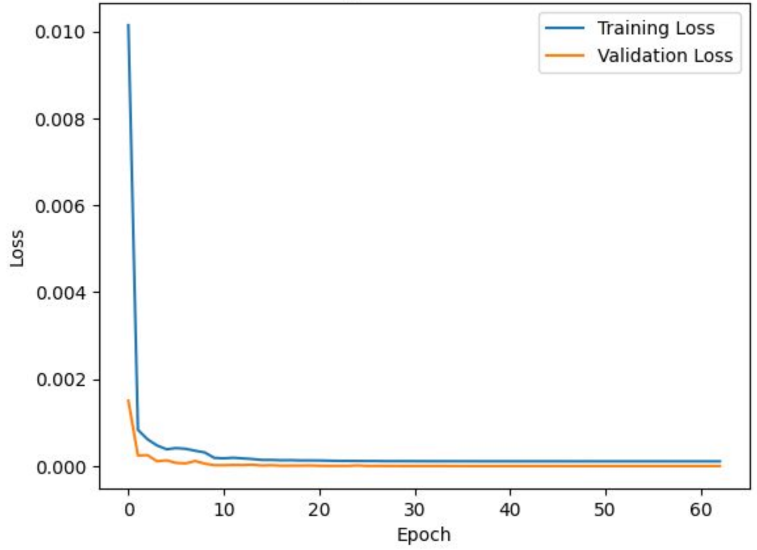
Anomaly detection via Transformer: Transformer block



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. Advances in Neural Information Processing Systems. https://arxiv.org/abs/1706.03762

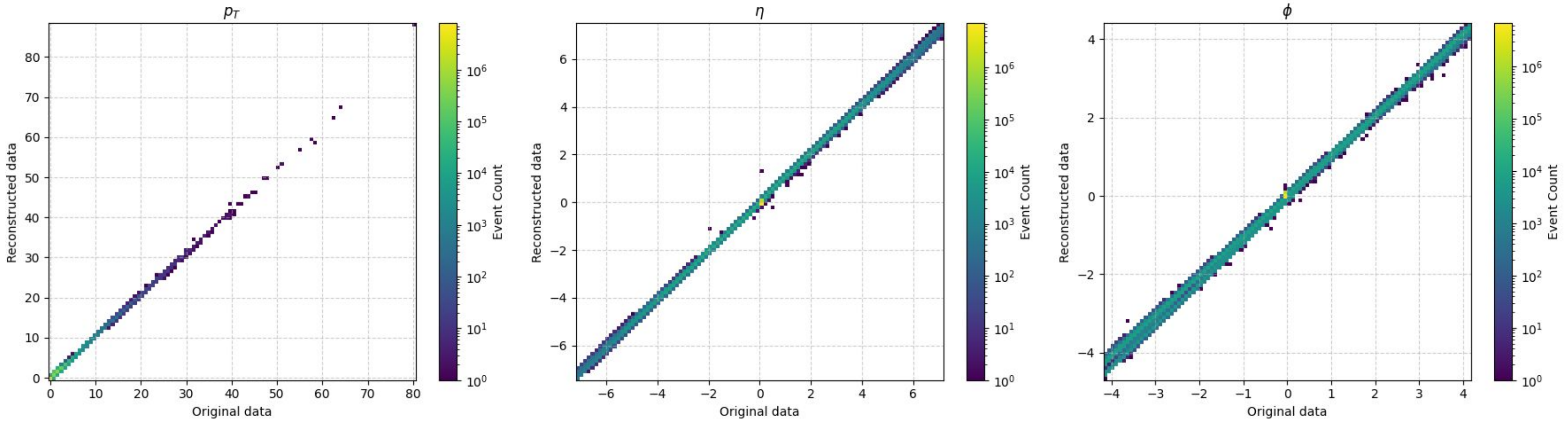
Anomaly detection via Transformer: Training

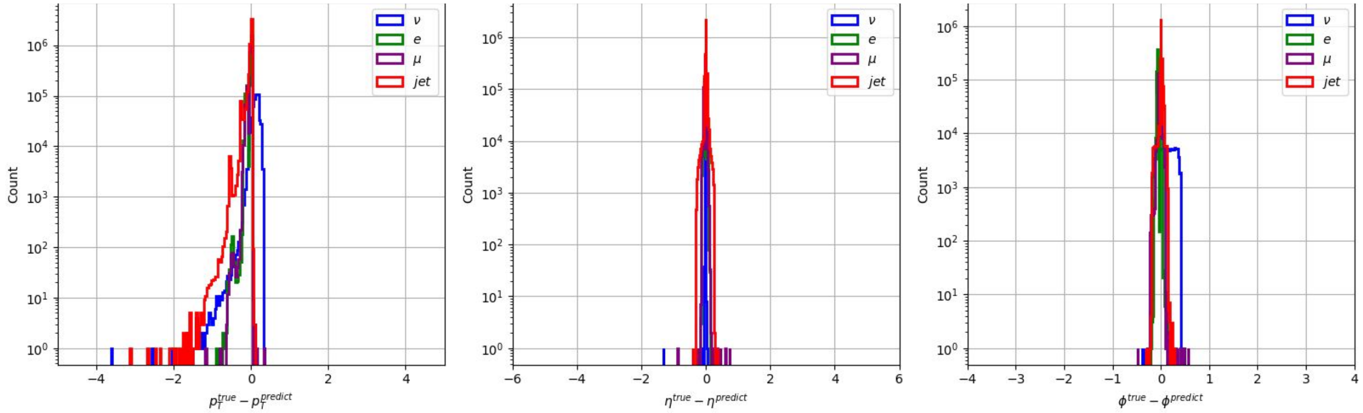
- Loss function: **Mean Square Error (MSE)**.
- Regularizer: **L2 Regularization**, with weight α = 0.0001.
- Batch size: **512**, 5,44e-4 % of the training set.
- Optimizer: **Adam**, with initial learning rate equal to 0.001.
- Learning rate is halved after 5 epochs without a decrease in the validation loss.
- Early stop of the training after 10 epochs without a decrease in the validation loss.



Number of parameters	257,796
Time of training of an epoch (on a T4 GPU).	326-330 s

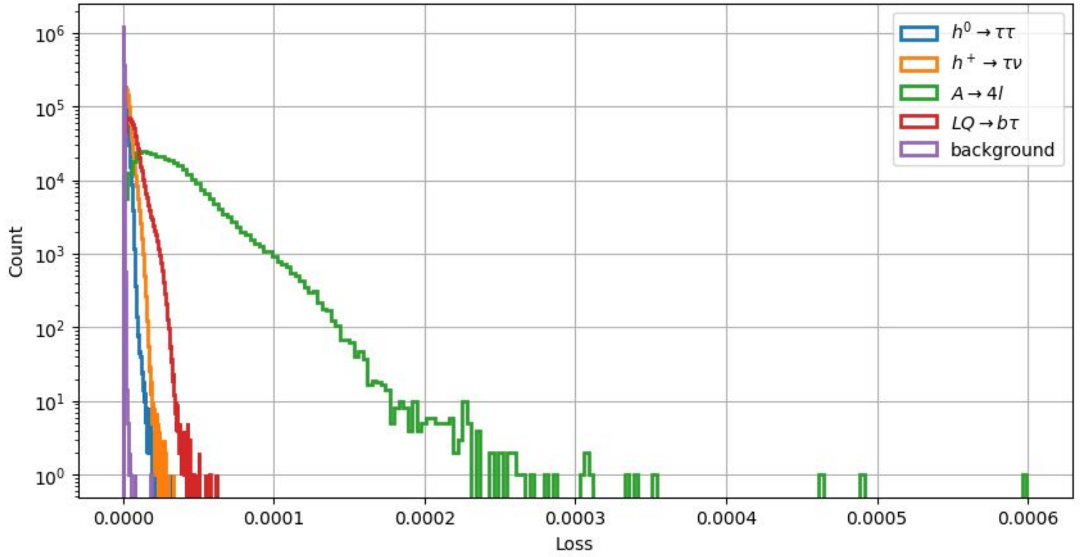
Anomaly detection via Transformer: Reconstructing SM events

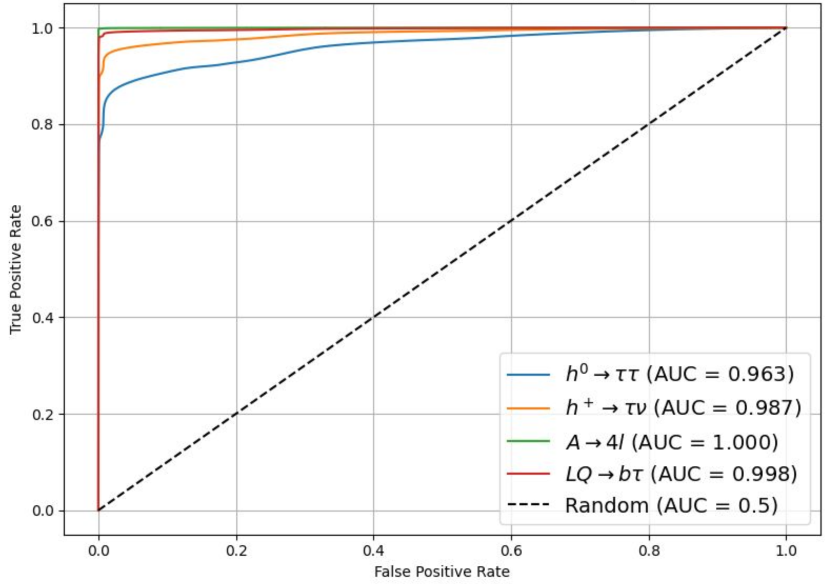




The results show a significant improvement compared to the other networks, the data incorrectly reconstructs to zero are completely disappear.

Anomaly detection via Transformer: Performance in BSM detection





It is used the same approach described for the FFN in slide 8 and 9.
In this case also **the performance in the anomaly detection task show a significant improvement**.
This results show how the attention mechanism is able to gives more weight to the non zero-rows and to understand the patterns in the data, linked to the different particles that, through their decay, produce the observed events.

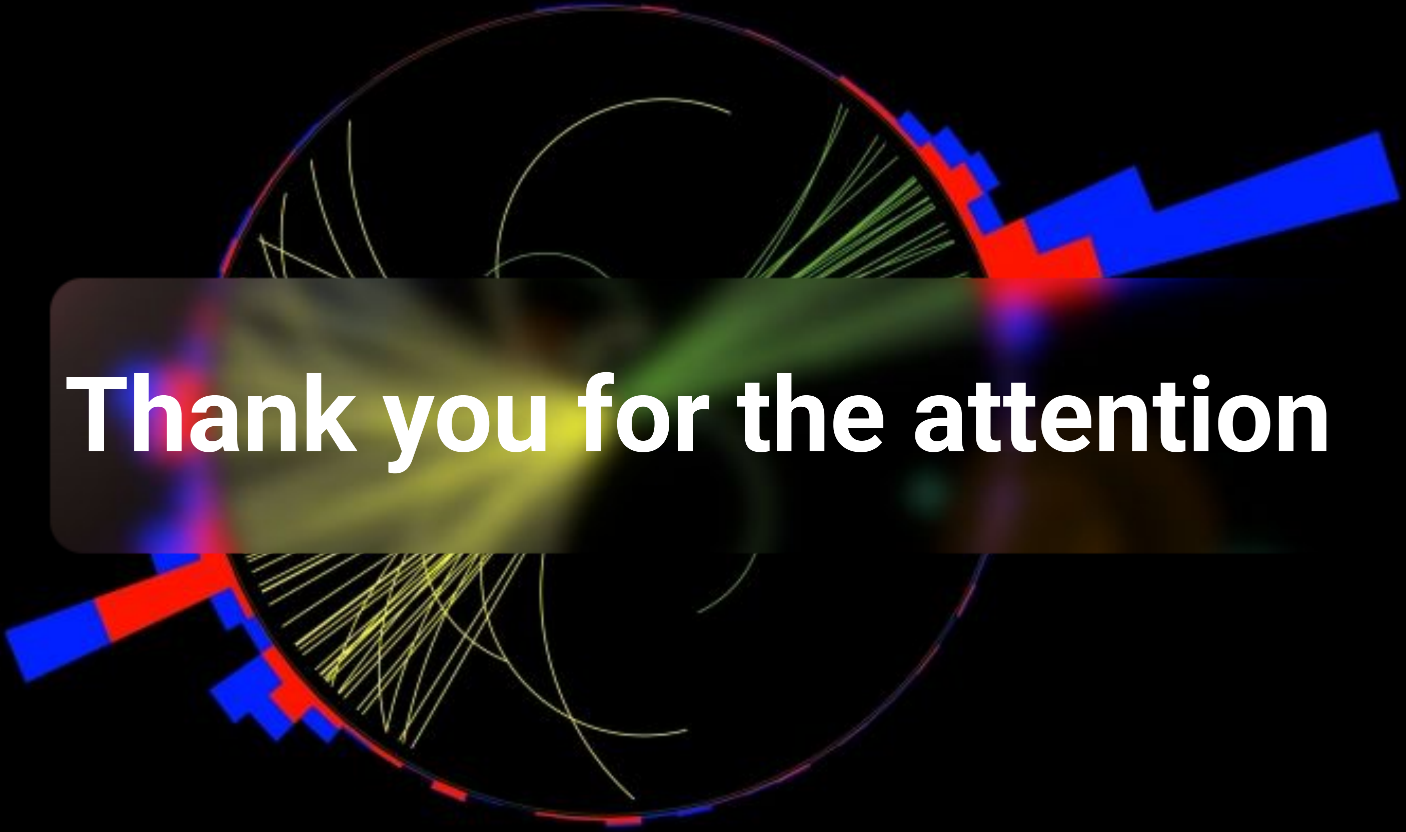
Conclusions

- In this project, three different autoencoders are used to solve an **anomaly detection task**.
- Autoencoders are **trained on** known collision events predicted by the **SM**.

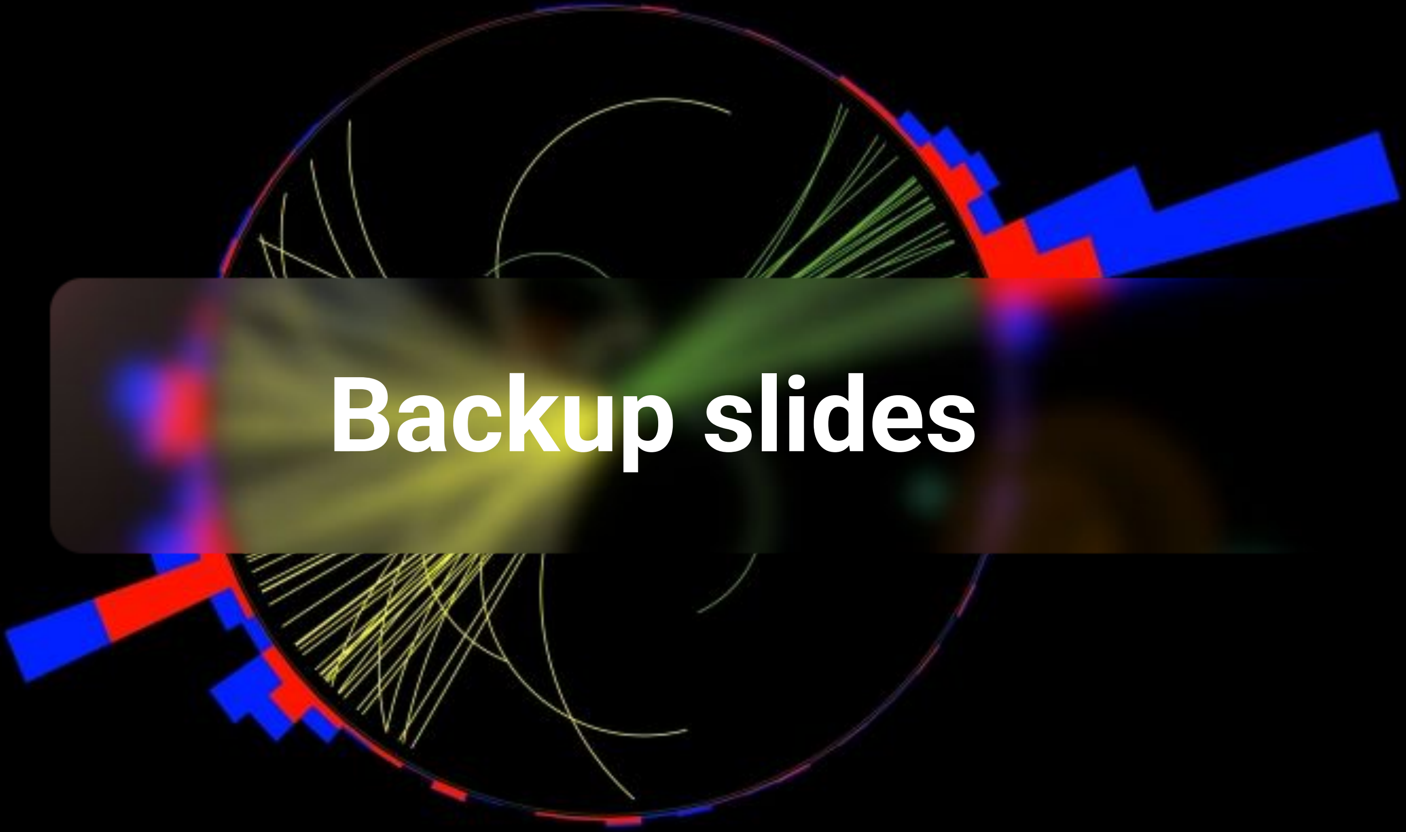
- **The networks learning performance** i.e. capability to reconstruct SM events is investigated.
  - Transformer performs better than CNN, that is followed by a feed-forward NN (FFN).

- **The networks** capability to **identify BSM events** as anomails is investigated.
  - Transformer performance significantly better than both CNN and FFN.

Datasets	$h^0 \rightarrow \tau\tau$	$h \rightarrow \tau\nu$	$A \rightarrow 4l$	$LQ \rightarrow \tau b$
AUC (FFN)	0.735	0.809	0.979	0.898
AUC (CNN)	0.623	0.733	0.994	0.897
AUC (Transformer)	0.963	0.987	1.000	0.998



Thank you for the attention



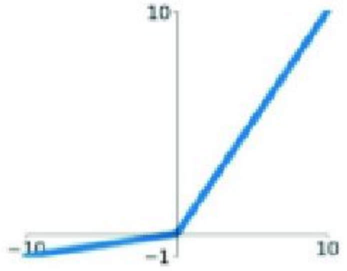
Backup slides

Leaky ReLU vs ReLU

Leaky RELU

Leaky ReLU

$\mathbf{max}\,(\mathbf{0.1}x, x)$

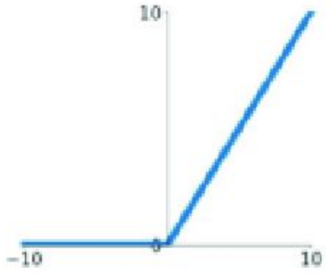


This value can change, in the networks is set to be equal to **0.3**

ReLU

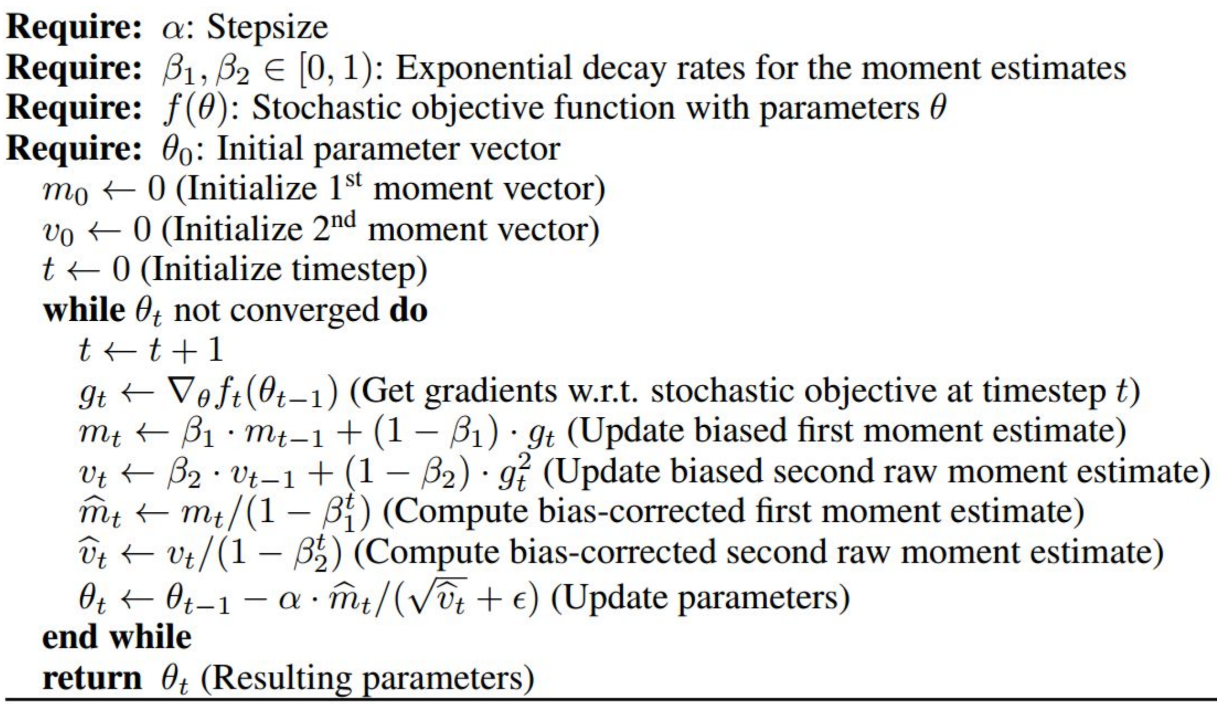
ReLU

$\mathbf{max}\,(\mathbf{0}, x)$



Better results are obtained using Leaky ReLU instead of ReLU as activation function in the network.
This is likely due to the nature of the data, which, after normalization, are distributed around zero, including in the negative range, and using the ReLU activation function, all the negative data are set to be equal to 0.

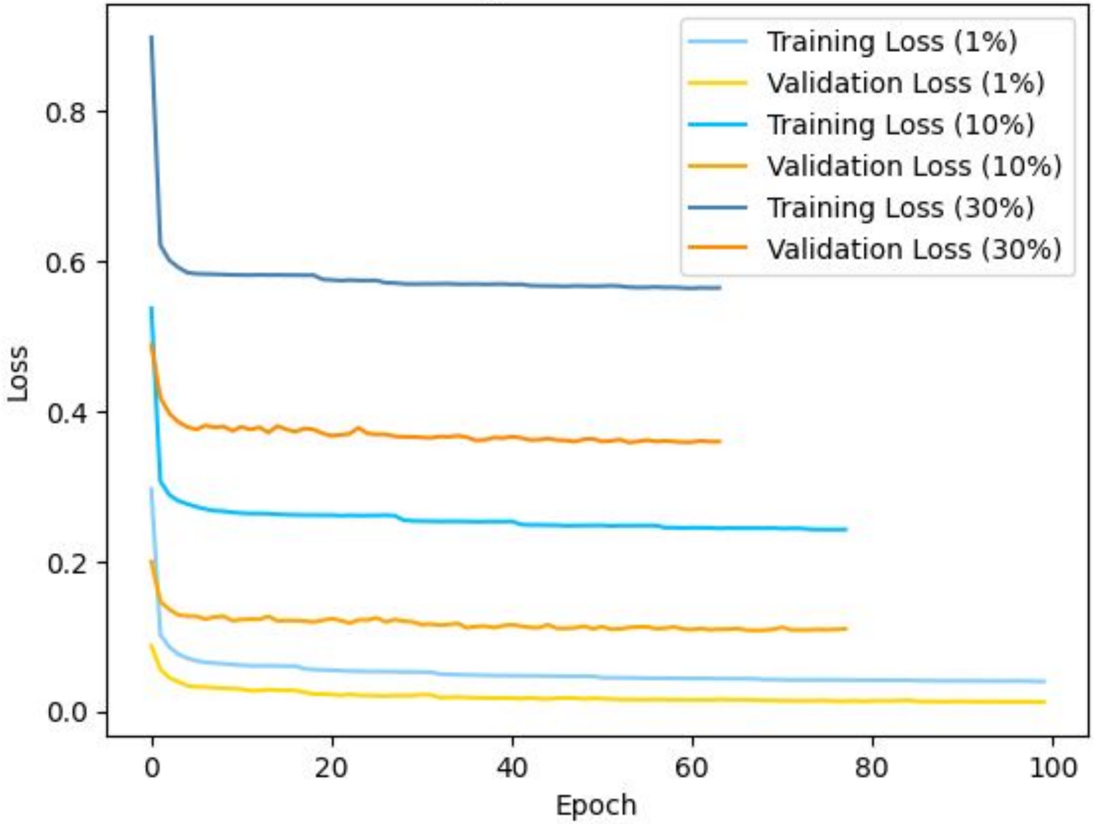
Adam Optimizer



```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1412.6980

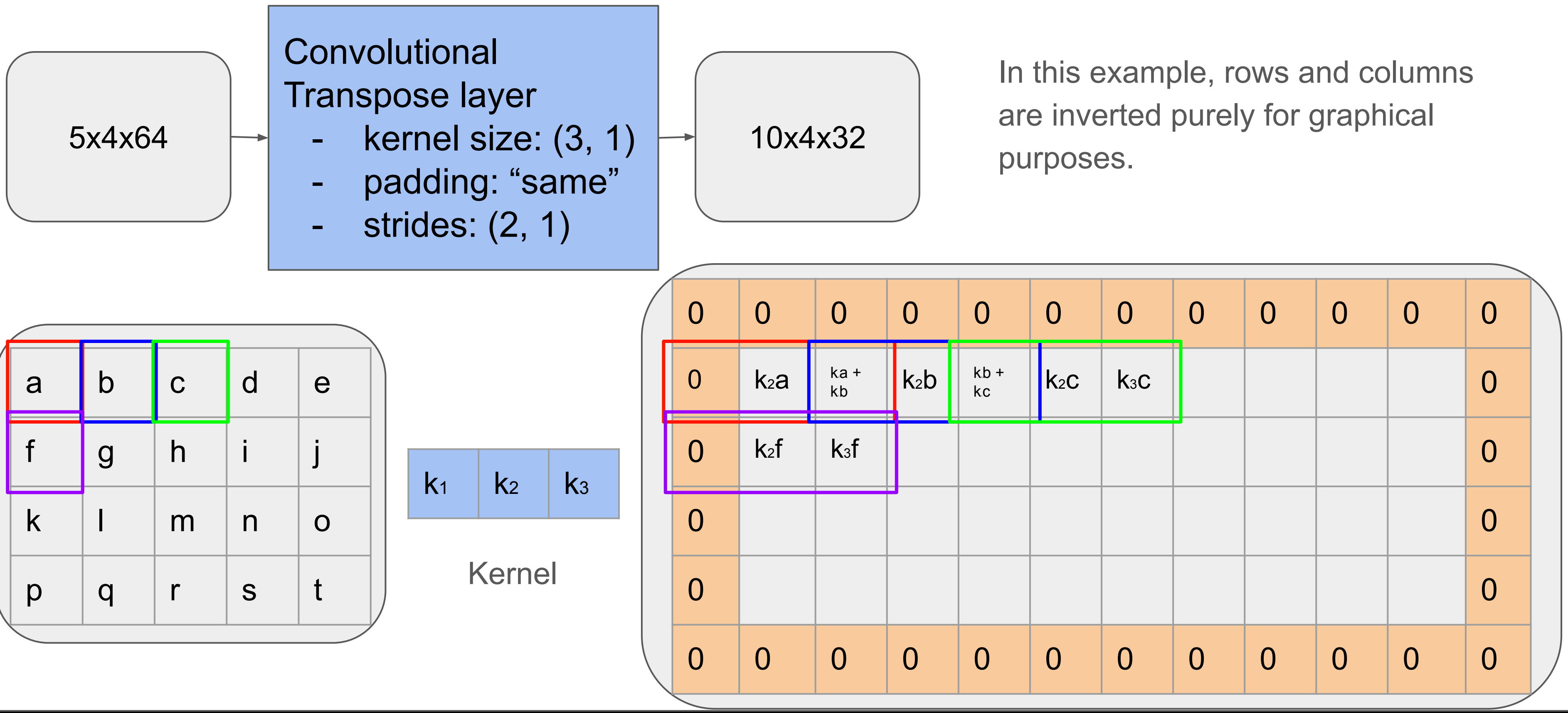
Anomaly detection via FFN: Training with different Dropout



This plot is made training the algorithm only on a part of the dataset, in particular the first 1,000,000 events are used.

It can be observed that, in all cases, the **validation loss is lower than the training loss, and the gap between them increases as the dropout rate increases**.

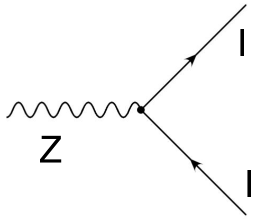
Convolutional Transpose layer

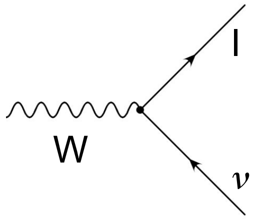


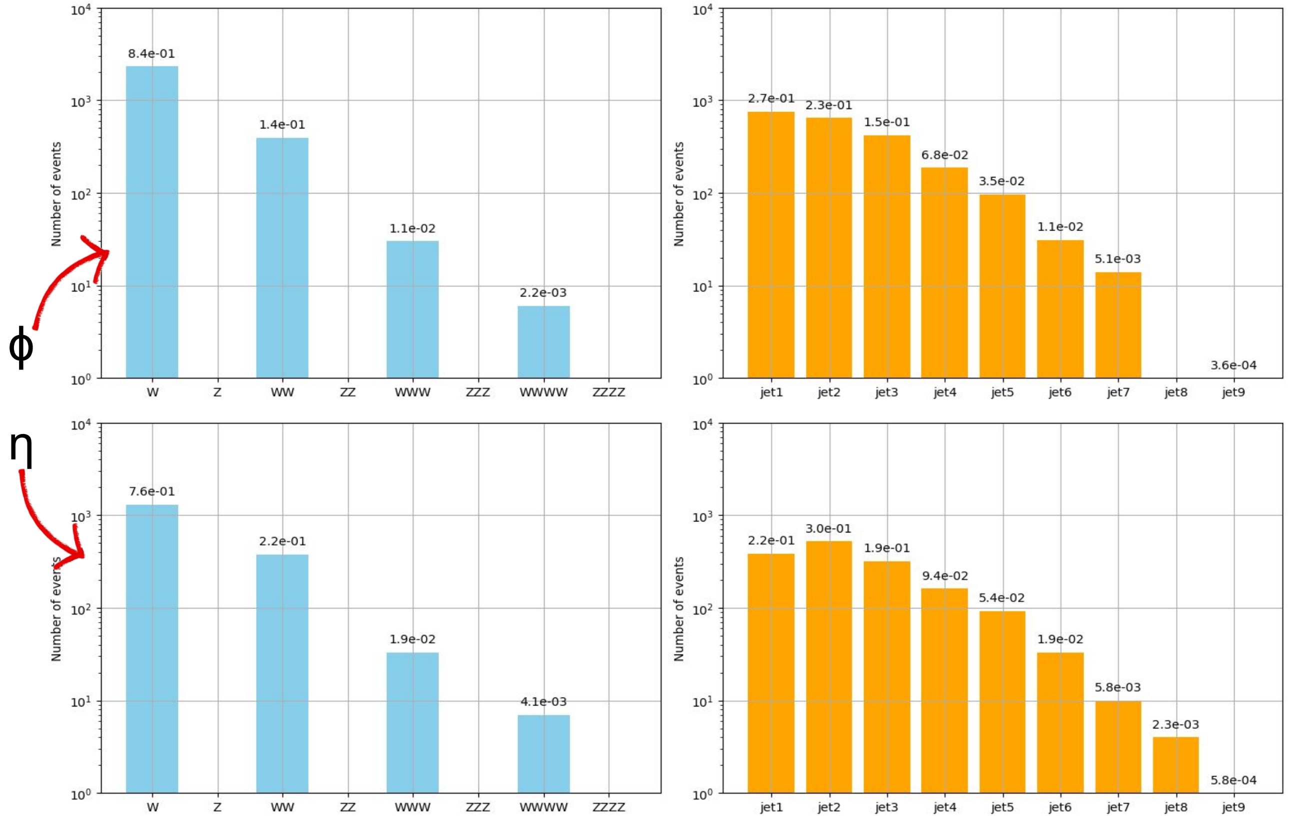
In this example, rows and columns are inverted purely for graphical purposes.

Anomaly detection via CNN: False zero in predicted data

By analyzing the events that the CNN incorrectly reconstructs as zeros (approximately **0.5%** of the cases), it was observed that they all originate from events involving the **decay of one or more W bosons**, in which one or more leptons and one or more neutrinos are produced, or events involving the **decay of one or two Z bosons in two τ.**





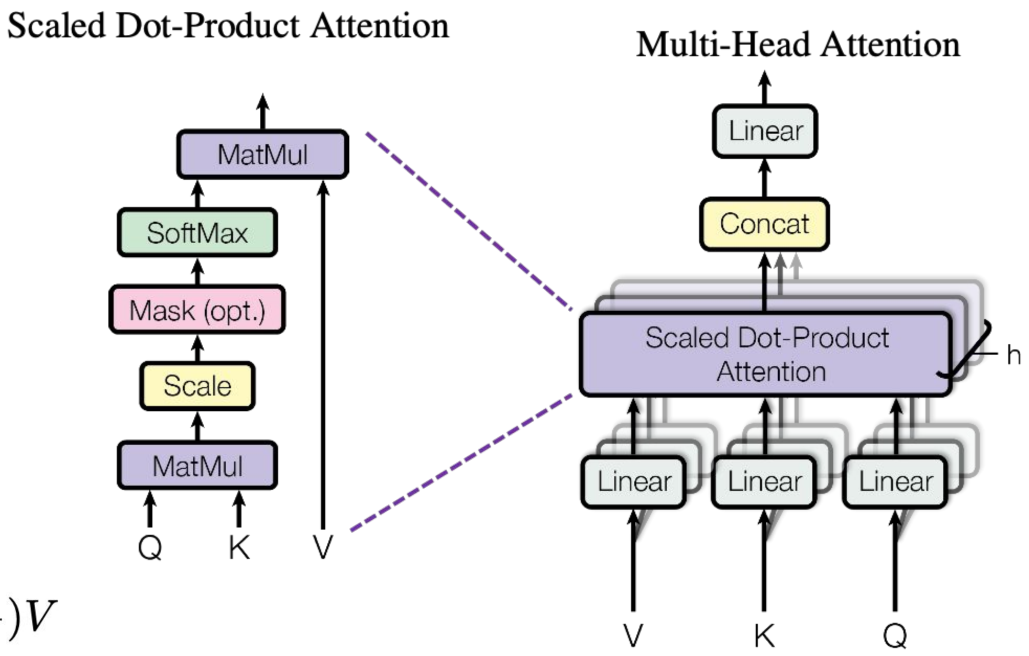


MultiHead Attention layer

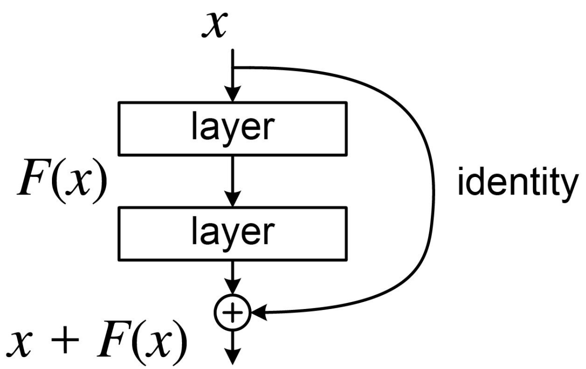
- The **query** represents what we are looking for.
- The **key** represents possible keywords that we want to compare with the query.
- The **value** represents the real content we obtain.

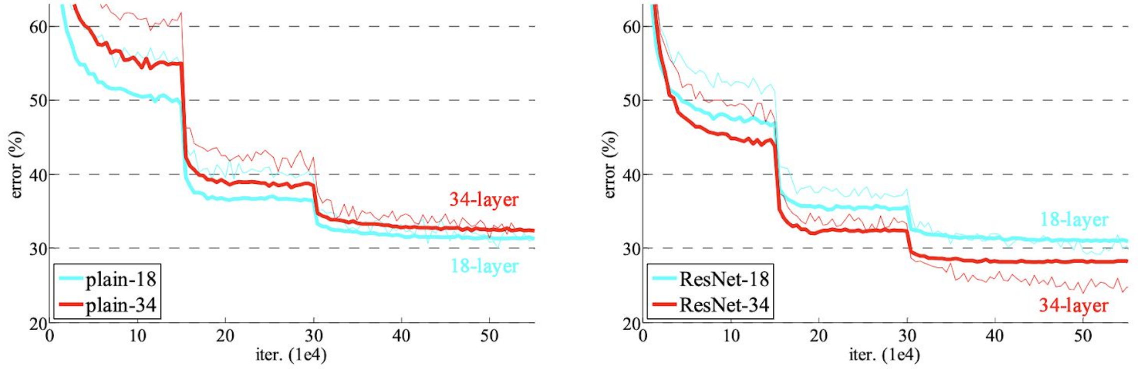
Q = K = V

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Residual connection





- Residual connections allow gradients to flow more easily during backpropagation, helping to **avoid the vanishing gradient problem**.
- Residual connections directly pass information from earlier to later layers, **reducing the risk of losing important details** and improving the quality of the learned representations.
- The residual structure encourages the network to learn only the necessary transformations, allowing it to **approximate identity mappings when needed**.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://arxiv.org/abs/1512.03385

Batch Normalization vs Layer Normalization

Batch Normalization

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \quad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

Batch size

- Depends on batch size.
- Requires maintaining moving averages of mean/variance.

Layer Normalization

$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i, \quad \sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2$$

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

Number of features

- Apply the same normalization independently per sample.
- Does not use batch statistics.