Department of Physics and Astronomy
Heidelberg University

Bachelor thesis in Physics
submitted by

Lukas Blecher

born in Ostfildern (Germany)

2020

Applying Super Resolution to Collider Events

This Bachelor thesis has been carried out by

Lukas Blecher

at the

Institute for Theoretical Physics

at

Heidelberg University

under the supervision of

Prof. Tilman Plehn

Abstract

In this thesis we use recent advances in deep learning to super resolve jet images. We show that models trained with the proposed pipeline can estimate a high resolution jet image from a singe low resolution input image. The generated images follow the physical distributions of the data set. Finally, we show that the model performs well even on previously unseen processes.

Zusammenfassung

In dieser Arbeit benutzen wir aktuelle Deep Learning Methoden um Jet Bilder höher aufzulösen. Wir zeigen, dass die Modelle, die auf dem hier vorgeschlagenen Weg trainiert werden in der Lage sind, aus einem niedrig aufgelösten Jetbild ein hochaufgelöstes zu erstellen. Die generierten Bilder haben die gleichen physikalischen Eigenschaften, wie die Bilder des zugrundeliegenden Datensatzes. Abschließend zeigen wir, dass das Modell auch auf Jetbildern von zuvor nicht gesehenen Prozessen funktioniert.

Contents

1. Introduction

The Large Hadron Collider (LHC) was built to check predictions of the Standard Model and potentially find new physics. The former was most notably achieved by the observation of the Higgs boson in 2012 [1] with the ATLAS detector [2]. In the LHC protons collide and produce new particles, e.g. top quarks. Top quarks themselves have a short lifespan and decay into other particles, which in return continue to hadronize. These jets can be tracked by the inner detector and are later observed in the calorimeter.
The inner detector covers a range of $|\eta| < 2.5$, while different calorimeters reach up to $|\eta| < 4.9$. In the region $3.1 < |\eta| < 4.9$ a different calorimeter is used, the so called *Forward Calorimeter* (FCAL) [3]. It has a lower resolution than the calorimeters in the central region, but since it is closer to the beam axis more events can be detected here. The question now is, whether it is possible to generate some useful high resolution data from the forward region.
With the rise of deep learning techniques, many super resolution models have been developed in the area of computer vision [4–12]. In this thesis we propose an adaptation to jet images for one of these models.

2. Physics of Jet Images

At the LHC, particles, e.g. protons, are accelerated to velocities close to the speed of light and collide with each other. The center of mass energy can reach up to $\sqrt{s} = 13\,\text{TeV}$ and will be increased to 14 TeV by 2021 [13]. During the collision, the quarks and gluons inside the protons interact with each other via the strong force and new particles are created. This interaction can be described with Quantum Chromodynamics (QCD). The process we are mainly focusing on is the hadronization of the top quark. It is the heaviest quark and thus has the shortest lifespan. Its electroweak decay can be seen in Fig. 2.1, where the top quark splits into a bottom quark and a W-boson which in return decays into two quarks itself. These quarks radiate gluons which produce more particles. The composition of these particles are called jets. If the proton collision only produces lighter quarks, all interactions can entirely be described with QCD, which is why we call the resulting jets QCD jets.

Around the collision point, multiple detectors are placed. Each focuses on a different particle type. The first section is the inner detector that acts as tracking system. Here the direction, momentum and charge of electrically charged particles is measured. The next part is the calorimeter setup which consists of the electromagnetic calorimeter, where electromagnetic showers are detected, and after that a hadronic calorimeter, where hadrons create hadronic showers. Only the muon spectrometer lies behind the calorimeters. The detectors have a cylindrical shape. The surface can be described by two parameters. The common choice are the azimuthal angle ϕ and the pseudorapidity

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right) , \tag{2.1}$$

which depends on the angle θ relative to the beam axis.

The entries of the resulting images are corresponding to the absolute value of the momentum component that is perpendicular to the beam axis, called transverse momentum p_T.

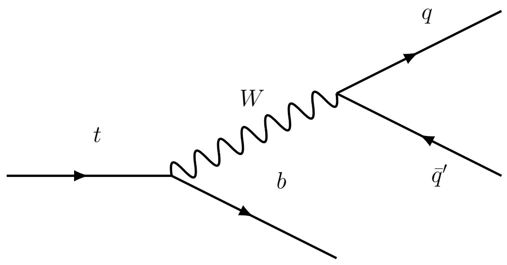


Figure 2.1. Feynman diagram of a top quark decay

3. Neural Networks

3.1. Fully Connected Networks and Activation Functions

The simplest neural network architecture is a fully connected network (FCnet), also called dense networks. The network is basically a composition of linear functions f_l and nonlinear activation functions ϕ_l called layers. A layer l can be described as follows:

$$z_l = \phi_l(\underbrace{f_l(z_{l-1})}_{\tilde{z}_{l-1}}) = \phi_l(W_l \cdot z_{l-1} + b_l) \tag{3.1}$$

Here z_{l-1} is the output of the layer $l-1$ and subsequently the input of the layer l. W_l and b_l are its weight matrix and bias vector respectively. $\tilde{z}_l$ is called the pre-activation of the layer and z_l is the activation of the layer. The final network is a composition of all layers. The nonlinear activation function ϕ plays a crucial role in the success of neural networks. Without the activation, the network would simply be a composition of linear functions, which in return is again a linear function. So the activation functions give the neural network its complexity and its power to approximate any mapping [14]. A popular choice for an activation function is the *Rectified Linear Unit* (ReLU) [15]:

$$\phi(t) = \max(0,\, t)\,. \tag{3.2}$$

The closely related leaky ReLU can also be seen often

$$\phi(t) = \begin{cases} t, & \text{for } t \geq 0 \\ \alpha\, t, & \text{for } t > 0 \end{cases} \tag{3.3}$$

with the parameter $\alpha > 0$. Traditionally, the sigmoid function was used as an activation function but due to its saturation on both ends, it now only serves as an output activation.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3.4}$$

3.2. Convolutional Neural Networks

A convolutional neural network (CNN) is a special case of the fully connected network with sparse weights and weight sharing. In Fig. 3.1 a convolutional layer is compared with a fully connected layer. They can be generalized into any amount of dimensions. Especially when applied to images, CNNs have an inherent advantage over FCnets. Neural networks are generally looking for patterns in the output of the last layer, also called feature map or feature vector for one dimensional networks. While fully connected networks are looking for target patterns in the whole feature map, convolutional networks are restricted to a

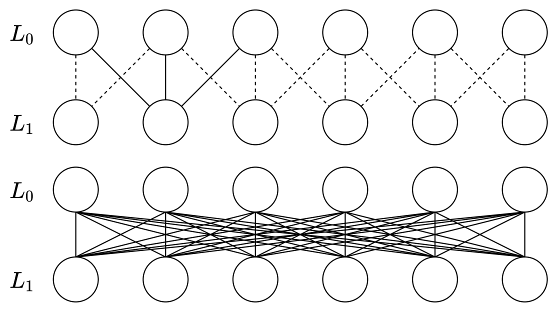


Figure 3.1. Comparison between a convolutional neural network (above) and a fully connected neural network (below). The CNN has only $3 + 1$ unique parameters while the FCnet has $6^2 + 6 = 42$ different weights and biases. The dotted lines symbolize weight sharing.

smaller window that is sliding over the image. This translation invariance fits perfectly with image-like data, since most of the time a target pattern is not filling the entire image, but can be found anywhere in the picture or appear multiple times.

The significantly lower parameter count per layer allows for deeper networks without the tendency for *overfitting* (see Fig. 3.1). Overfitting means that the model goes beyond generalizing the training data and memorizes the individual samples. It often happens when the model is too powerful, or in other words has too many trainable parameters.
A convolutional layer is parameterized by a number of values.

1. Kernel size: It describes how big the moving window is

2. Stride: Step size of the sliding window

3. Padding: Controls the amount of added zeros on each side of the input

4. Filters: The number of independent windows used in parallel.

In Fig. 3.1 the convolutional layer has a kernel size of 3, a stride of 1 and padding of 1. These parameters have the special property that they preserve the spatial dimension which is why it is often used in practice.

3.3. Loss Functions

The loss function, or cost function, defines the training target and is a measure of how well the model performs. It compares the prediction with the ground truth and reduces the difference to a single scalar number. During training (see Sec. 3.4) we aim to minimize the loss function as far as possible. It is common practice to calculate the loss for multiple training examples at once, a so called mini batch of size N. In regression problems, popular loss functions are the Mean Absolute Error (MAE) or Mean Squared Error (MSE), which

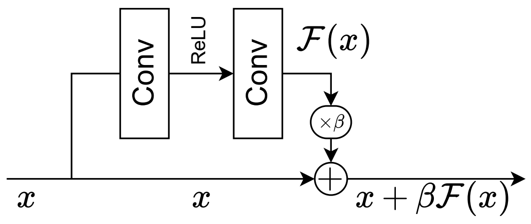


Figure 3.2. Residual building block

rely on the L_1 and L_2 metrics respectively. The loss functions will be called by their metric names in this thesis.

$$\mathrm{MAE}(a,\, b) = \frac{1}{N}\sum_{i}^{N} |a_i - b_i| \ , \tag{3.5}$$

where a and b are N-dimensional.

For classification problems with two classes, the binary cross entropy (BCE) function is used:

$$\mathrm{BCE}(a,\, b) = \frac{1}{N}\sum_{i}^{N} \left(b_i \ \log(a_i) + (1 - b_i) \ \log(1 - a_i)\right) \tag{3.6}$$

3.4. Training Process

In the case of supervised learning a training example consists of the input X and the ground truth y^*, also called target or label. The input is given to the network f with the parameters θ. Together they make a prediction $\hat{y} = f(X;\, \theta)$. Next, the performance is measured by a loss function $\mathrm{L}(\hat{y},\, y^*)$. All parameters are updated into the direction that minimizes the loss function. This is done by using the gradients $\nabla_\theta L$ that are calculated with the backpropagation algorithm [16]. An optimization algorithm, like *Adam* [17], then uses the gradients to compute the parameters θ_{t+1} for the following iteration.

3.5. Deeper Networks with the help of Residual Connections

He et al. introduced a *deep residual learning* framework [18] that allows for deeper networks by adding a skip connection. It partially prevents vanishing gradients because the gradients skip the nonlinear activation function. One building block can be seen in Fig. 3.2. The scaling parameter β is a hyperparameter of the model. Hyperparameters are model defining parameters which stay fixed during the entirety of the training process.
In this thesis *residual connection* is used whenever the output of a subsection of the model is added to its input, which is represented by $\bigoplus$. A *skip connection* is used when, instead of adding, the two objects are concatenated along the channel dimension, which is noted by $\textcircled{c}$.

3.6. Generative Adversarial Networks

3.6.1. Basic Functionality

The concept of *Generative Adversarial Networks* (GANs) [19] is that two neural networks are competing with each other. The generator G is given the task to generate new samples that could belong to the training data set and the discriminator D has to decide if a given sample belongs to the real data set or if it was generated by the generator. The idea is that after successful training, the discriminator can no longer distinguish a real sample from a fake sample. For the generator, this means that he has learned to convincingly create real samples from the data set.
The loss function for the generator is

$$\mathrm{L}_G^{\mathrm{GAN}} = -\mathbb{E}_{X_f \sim \mathbb{Q}} \left[\log\Big(D(X_f) \Big) \right] . \tag{3.7}$$

Here $\mathbb{P}/\mathbb{Q}$ are the true/generated distributions respectively.
For the discriminator the loss function is as follows:

$$\mathrm{L}_D^{\mathrm{GAN}} = -\mathbb{E}_{X_f \sim \mathbb{Q}} \left[\log\Big(1 - D(X_f) \Big) \right] - \mathbb{E}_{X_r \sim \mathbb{P}} \left[\log(D(X_r)) \right] . \tag{3.8}$$

These loss functions can be constructed as the sum of two BCE loss terms (3.6) using the labels $b \in \{0,\, 1\}$.

3.6.2. Relativistic average GAN

In the standard GAN [19] the discriminator output $D(X)$ is a measure of how realistic the input X seems. In a relativistic average GAN [20] the discriminator tells us the probability of a fake sample being more realistic than real samples on average, and vice versa. The generator loss in the GAN setup, also called adversarial loss is as follows

$$\mathrm{L}_{\mathrm{adv}} = -\mathbb{E}_{X_f \sim \mathbb{Q}} \left[\log\Big(D_{\mathrm{Ra}}(X_f) \Big) \right] - \mathbb{E}_{X_r \sim \mathbb{P}} \left[\log(1 - D_{\mathrm{Ra}}(X_r)) \right] \tag{3.9}$$

with $D_{\mathrm{Ra}}(X_r) = \sigma\left(C(X_r) - \mathbb{E}_{X_f \sim \mathbb{Q}}\, C(X_f) \right)$, $D_{\mathrm{Ra}}(X_f) = \sigma\left(C(X_f) - \mathbb{E}_{X_r \sim \mathbb{P}}\, C(X_r) \right)$ where $C(x)$ is the unactivated output of the discriminator. The relativistic aspect of the setup comes from the way $D_{\mathrm{Ra}}(X)$ is computed. We can see that when comparing this loss to the standard adversarial loss (3.7), another term is added. In the standard GAN $D(X_r)$ is not dependent on the generator, which is why the term is not needed. However in the relativistic GAN $D_{\mathrm{Ra}}(X_r)$ does depend on X_f. As a consequence it has to be added to the target function.
The loss for the discriminator is the same as in (3.9) but with switched labels:

$$\mathrm{L}_{\mathrm{D}} = -\mathbb{E}_{X_f \sim \mathbb{Q}} \left[\log\Big(1 - D_{\mathrm{Ra}}(X_f) \Big) \right] - \mathbb{E}_{X_r \sim \mathbb{P}} \left[\log(D_{\mathrm{Ra}}(X_r)) \right] . \tag{3.10}$$

3.6.3. Gradient Penalty

The loss function of the generator or the discriminator depends on the parameters of one another. That means that after each iteration the training target changes. If that happens too quickly, the models cannot converge. That is why we use gradient penalty [20, 21] in our project. It prevents the discriminator from changing its weights too drastically in a single update by adding a constraint to the loss function that forces the norm of the gradient to be close to 1. It comes in the form of the following regularization term.

$$\mathrm{L_D} \to \mathrm{L_D} + \lambda_{\mathrm{reg}} \mathrm{L_{reg}} \tag{3.11}$$

with

$$\mathrm{L_{reg}} = \mathbb{E}_{\hat{X} \sim \mathbb{P}_{\hat{X}}} \left[\left(\left\| \nabla_{\hat{X}} C(\hat{X}) \right\|_2 - 1 \right)^2 \right] . \tag{3.12}$$

Here $\hat{X}$ is an interpolation between real and fake images
$\hat{X} = \varepsilon \, X_r + (1 - \varepsilon) \, X_f ,$ where $X_r \sim \mathbb{P}$, $X_f \sim \mathbb{Q}$, $\varepsilon \sim U[0, 1]$.

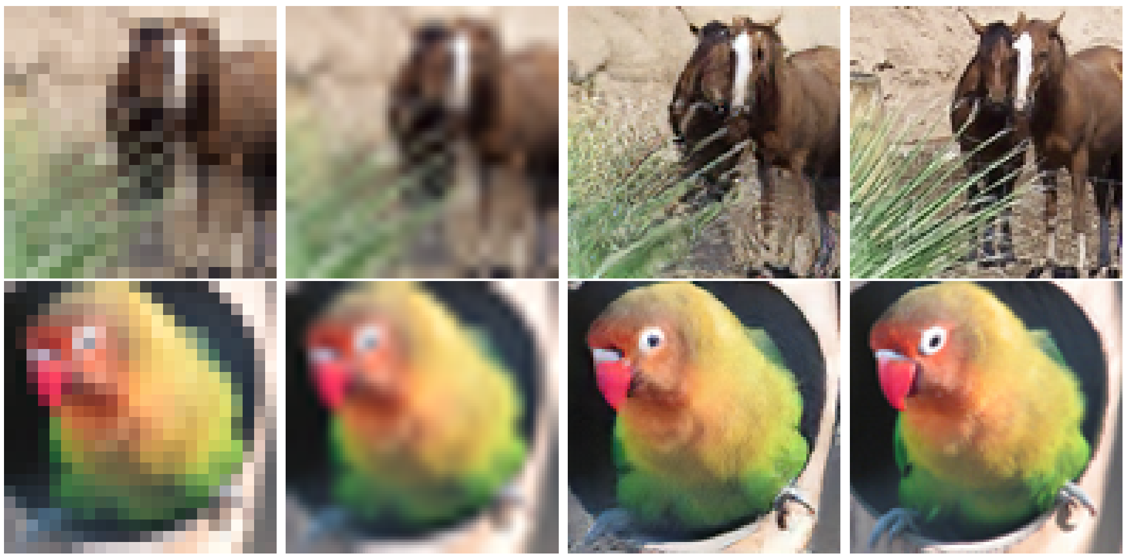
3.6.4. Markovian Discriminator

The *Markovian Discriminator* or also called *PatchGAN* [22, 23] differs from the standard discriminator in its output. While a traditional discriminator projects the input to a single number, the PatchGAN estimates the realness of multiple patches of the input image. Its output is a matrix with each value corresponding to a different section of the image.

4. Super Resolution on Jet Images

4.1. Super Resolution Networks

In the task of single image super resolution (SISR) the goal is to predict a high resolution (HR), super resolved (SR) version of a given low resolution (LR) image. It is an ill-posed problem because many HR images can correspond to a single LR image. The goal is to generate a meaningful SR image. Learning based approaches hallucinate details and texture that are consistent with the LR image but may look differently in the HR image. This can be seen in Fig. 4.1 where I applied super resolution to the *STL-10* [24] data set which consists of approximately 100.000 color images with a resolution of 96×96 pixels. In my experiment I chose an upsampling factor of $f = 4$, so the model learns to super resolve LR images of the size 24×24 pixels.
The results are shown in Fig. 4.1. The super resolved image 4.1c has better refined outlines, colors and sharper edges in comparison to the bicubic upsampled image 4.1b. The eye of the parrot for example is grey in the LR image, but the model reconstructs the correct color composition.



(a) LR **(b)** Bicubic **(c)** SR **(d)** HR

Figure 4.1. Example for super resolution on the STL-10 testset using the ESRGAN

4.2. Jet images as Datatype

The data set used in this thesis was originally created for top tagging [25]. It contains $t\bar{t}$-events and QCD dijets that were generated at a center of mass energy of $\sqrt{s} = 14\,\text{TeV}$ using *Pythia* [26], a Monte Carlo event generator. After that, a detector simulation, called *DELPHES* [27], simulates the ATLAS detector. The jet clustering was done by *FastJet*

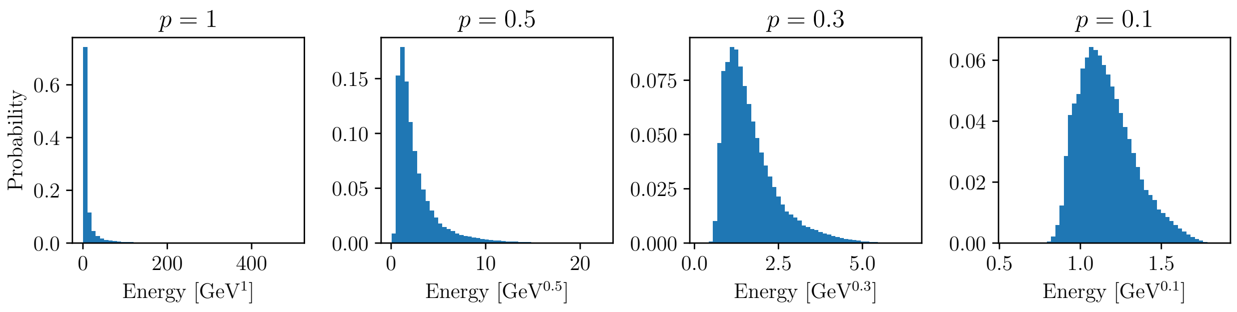


Figure 4.2. Energy distribution behaviour when raising it to different powers p

[28, 29] using the anti-k_T algorithm [30] with a jet radius $R = 0.8$. Next, a p_T cut was made. The jet had to fulfill $p_T \in [550, 650]$ GeV. The training data set only consisted of the top jets. At this point it is important to note that the events were required to fulfill $|\eta|_{\text{jet}} < 2$ which is well inside the central region of the ATLAS detector. The resulting data comes in the form of images that display top quark hadronizations, where the pixel values hold the transverse momentum p_T. The images are sparse, since a jet only has about 50 constituents. That corresponds to a sparsity of 99.80% for 160×160 pixel big pictures.
The few non-zero values themselves cover a big range. The transverse momentum can be as high as 500 GeV but most of the constituents still have momenta near zero. Usually, in machine learning the data is normalized in some way. There are multiple ways of normalizing jet images to be better suited for machine learning, e.g. dividing by the highest pixel value or setting the sum of the transverse momenta to 1. These transformations do not retain the information of the absolute momentum which may not be a problem for classification tasks but for our purposes this information is needed. In Fig. 4.2 the energy distribution of the data set is shown. When raising the images to a power $p \in (0, 1)$ in a pixel-wise fashion the distribution gets broader relative to its domain. The training is significantly improved because of this preprocessing step.
Since most super resolution models are based on paired LR/HR data, we need an artificial way of downsampling the HR image to get the LR version relatively quickly and if possible in a differentiable manner. In this project *Sum Pooling* (see Fig. 4.3) is used. It can be interpreted as a naive way of simulating a detector of lower resolution. This aspect of the training pipeline could be improved by a more sophisticated downsampling method or by using unpaired LR/HR images [31].

4.3. Network Architecture

The model used in this thesis is mainly based on the *Enhanced SRGAN* (ESRGAN) [10] (see Fig. 4.4) which is an improved version of the *Super Resolution GAN* (SRGAN) [11]. It has two *Markovian Relativistic average Discriminators with Gradient Penalty* (RaGAN-GP) [20, 22].

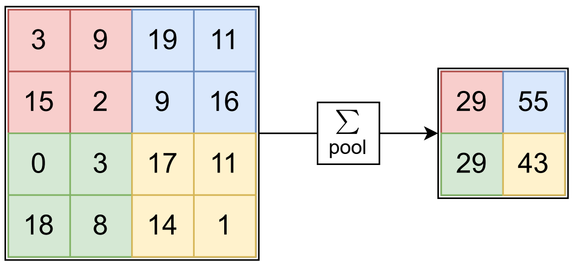


Figure 4.3. Sum Pooling example

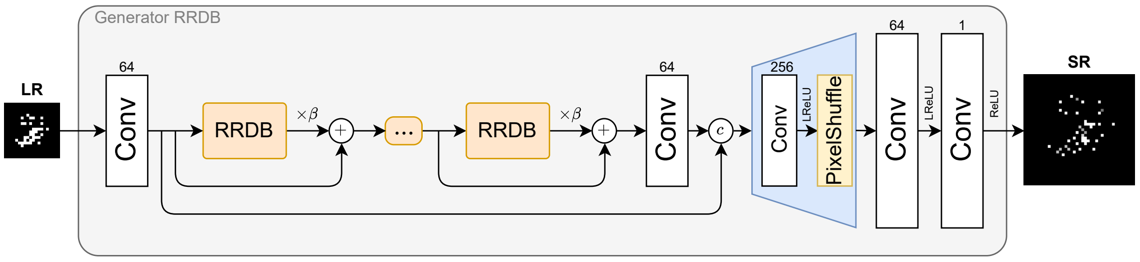


Figure 4.4. Generator from ESRGAN. The "Residual in Residual Dense Block" is shown in Fig. 4.5.

4.3.1. Generator

The generator is the model we are ultimately interested in. It takes a LR image and converts it into a SR image. The generator is a deep residual fully convolutional network. The main building block of the generator are the *Residual in Residual Dense Block* (RRDB) [10] which in return consists of 3 *Dense Residual Blocks* (DRB) [32] that are connected with each other via residual connections. The DRB is built using a number of consecutive convolutional layers, all of which have a 3×3 kernel size, stride 1, padding 1 and 64 filters. The activation function is a LeakyReLU with $\alpha = 0.2$. The particularity of the DRB is that a layer receives the input of all other layers as addition to the output of the previous layer. This structure fuses all the feature maps inside of the block and aims to extract the most information in that way. The amount of RRDBs B is a hyperparameter and determines how deep the neural network gets. All the convolutions used in the generator preserve the spatial dimensions of the input image, but since we want to end up with a bigger picture in the end, some sort of upsampling is needed. This is done by one or more *Pixel Shuffle*-Layers [33]. Each doubles the spatial dimensions by spreading out the feature maps. In the HR feature space there are two additional convolutional layers, one of which only scales the output by a fixed value.

4.3.2. Discriminator

The discriminator network is fairly simple in comparison to the generator. It is a simple feed forward convolutional network with LeakyReLU activations. Every discriminator block consists of two convolutional layers with a 3×3 kernel and padding 1. After each

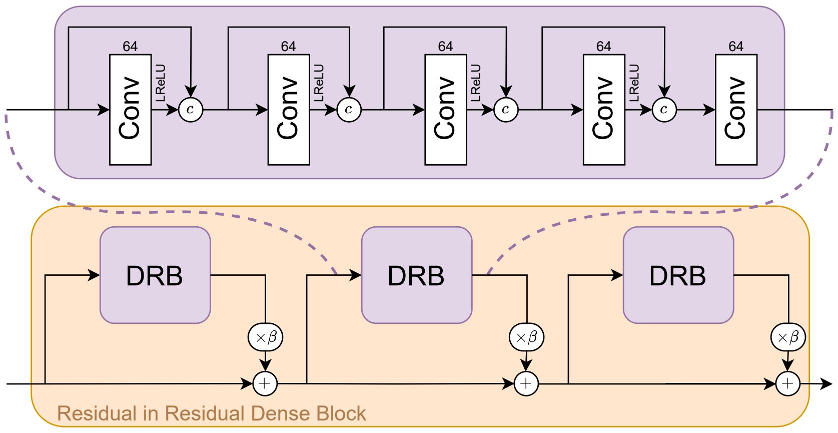


Figure 4.5. The Dense Residual Block (DRB) and its integration into the Residual in Residual Dense Block (RRDB)

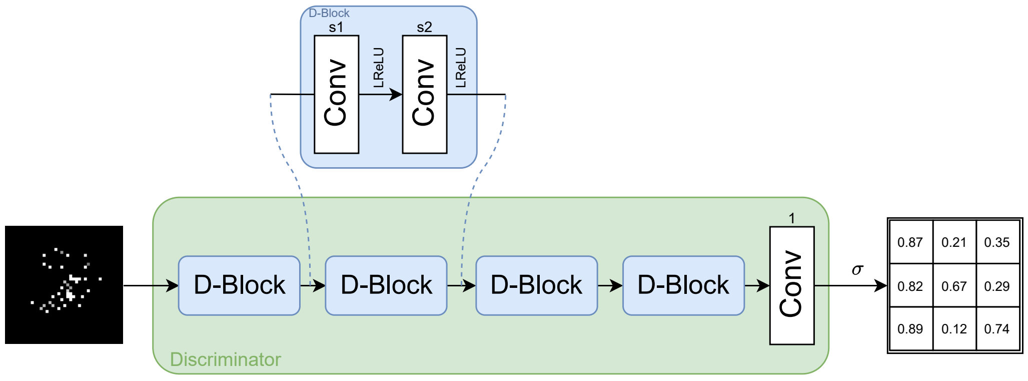


Figure 4.6. The structure of the Markovian discriminator

block the amount of filters is doubled. The first convolution of each block conserves the spatial dimensions of the input while the second layer halves it because it is a strided convolution. We used 4 blocks and started with 64 filters. So far this is the same discriminator structure as Ledig et al. used in the SRGAN [11]. We deviated from the proposed structure by removing the batch normalization layers to help the model converge. This was recommended by Gulrajani et al. [21] when using gradient penalty. We also cut off the network before it was flattened and fed into a fully connected layer. That way we employed a Markovian discriminator in its place.

4.4. Loss Function

SRGAN and ESRGAN work great with images like in ImageNet [34] or STL-10 [24] but they also had some excess functionalities e.g. the perceptual loss that aims to make the resulting images visually more pleasing. It is a combination of the adversarial loss from the Discriminator and a content loss that compares feature maps of a pre-trained image classification network. The latter is redundant for our purposes and only the adversarial loss remains. Additionally, we want the high resolution image to look like the ground truth, so we add a L_1 loss between SR and HR. L_1 prevents blurring in comparison to the L_2 loss.

$$L_{HR} = L_1\,(\mathrm{SR},\ \mathrm{HR}) \tag{4.1}$$

Since the high resolution jet image should correspond to its low resolution counterpart, we added another loss term that compares the model input with the downsampled model output on a pixel by pixel basis.

$$L_{LR} = L_1\left(\sum_{\mathrm{pool}}(\mathrm{SR}),\ \mathrm{LR}\right) \tag{4.2}$$

Additionally, there is the GAN loss (3.9) from chapter 3.6.
The final loss function for the generator can be written as a weighted sum of the standard loss and the power loss, which in return are a weighted sum of the HR, LR and adversarial loss.

$$L_{tot} = \sum_{s\in\{\mathrm{std},\,\mathrm{pow}\}} \lambda_s\left(\lambda_{HR}\,L_{HR} + \lambda_{LR}\,L_{LR} + \lambda_{adv}\,L_{adv}\right) \tag{4.3}$$

The discriminator loss only consists of the GAN loss (3.10) with gradient penalty (3.11).

4.5. Training on Jet Images

The process of a training iteration can be seen in Fig. 4.7. The starting point is the high resolution ground truth jet image HR from which the low resolution input image LR is computed via sum pooling. Next, the jet image it is raised to the power of p as a pixel-wise operation and scaled by a factor k afterwards. We chose $p = 0.3$, any lower and we had to deal with numerical instabilities, and $k = 1/f$ with the upscaling factor f. This results in a flatter energy distribution, as explained before, which in return is easier to learn for the model. The data is then fed into the *Generator RRDB* and the output is divided by the factor k from above. The result is supposed to be the super resolved image raised to the power p: SR^p. This intermediate result is saved for the computation of the loss later on. But since the model is supposed to approximate the high resolution ground truth in the end, we take the p^{th} root of the intermediate result $\mathrm{SR} = \sqrt[p]{\mathrm{SR}^p}$ to get the final result.
Every loss is computed once for the standard jet images, the standard loss, and once for the images raised to the p^{th} power, the power loss. The first contribution comes from the corresponding discriminator. Both, the standard loss and the power loss have their own discriminators. Next the SR images are pooled to get their LR version and compared with the real LR images as described in (4.2).
The hyperparameters we used can be seen in Tab. 4.1. Adam was used for the optimization with $\beta_1 = 0.5$, as suggested by Radford et al. [35], and $\beta_2 = 0.9$. The learning rate was set to $\lambda = 0.0001$. The training of a model took 50k-100k iterations.

B	N	β	p	λ_{reg}	λ_{std}	λ_{pow}	λ_{HR}	λ_{LR}	λ_{adv}
15	15	0.1	0.3	0.002	0.2	1	1	0.1	0.05

Table 4.1. Hyperparameters we found to be optimal

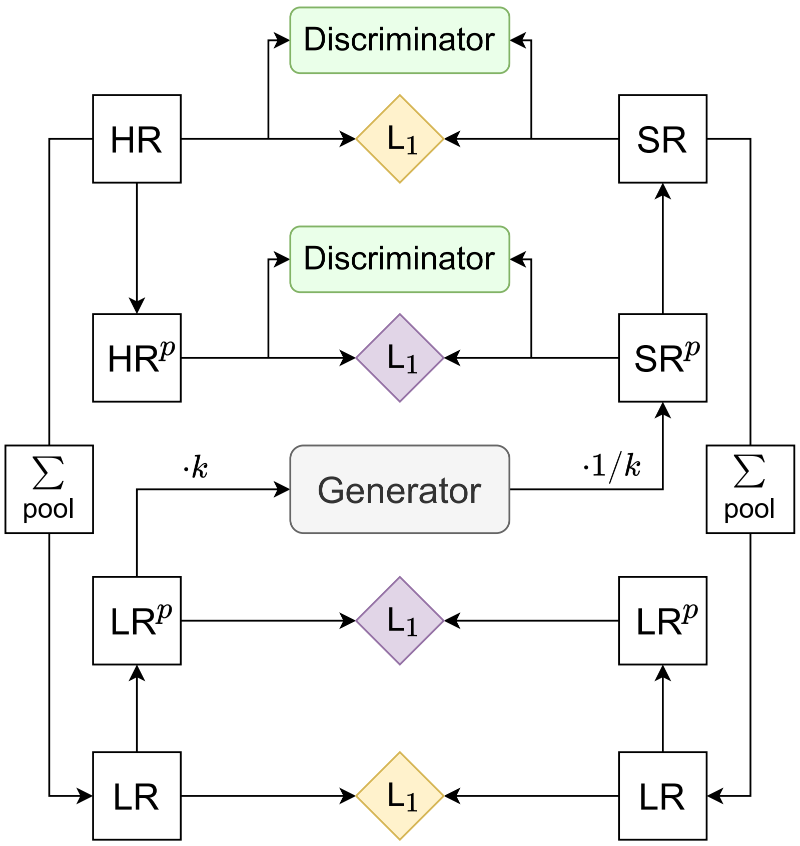


Figure 4.7. Training process for jet images. The generator can be seen in Fig. 4.4

5. Evaluation

We want the super resolved images to be indistinguishable from the ground truth jet images. That means the images have to show the same physical properties, and they need to look like real jet images.

5.1. Evaluation Methods

5.1.1. Metrics

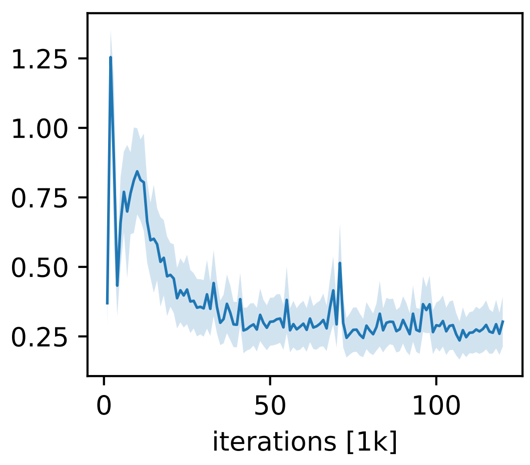
In the standard super resolution task metrics like the *Peak Signal to Noise Ratio* (PSNR) or *structural similarity* (SSIM) are used to compare different models and approaches. These metrics are meaningless in the context of jet images.
A natural metric is the L_1 metric which is part of the loss function. It can be used as a measure of how close the generated super resolution image is to the actual ground truth image. But as stated above, we aim to produce meaningful high resolution images rather than to reproduce the original image. Komiske et al. developed a metric for collider events called Energy mover's distance (EMD) [36], similar to the Earth mover's distance, that can measure how similar two jet images are with respect to the energy of the constituents. This is a much better metric for our purposes than a pixel-wise comparison. In practice the difference is not as pronounced as expected. Looking at the validation results during the training of a model in Fig. 5.1, we can see the HR L_1 metric as well as the EMD metric. Both show the same tendencies at the same validation points but the L_1 metric is a little bit smoother while the EMD metric is more sensitive to the models parameter changes. Nevertheless, it seems like the L_1 metric is a comparable measure to the EMD metric, when tested on many samples.
The best results for the EMD metric for all super resolution factors, evaluated on the test set, can be found in Tab. 5.1.

5.1.2. Observables

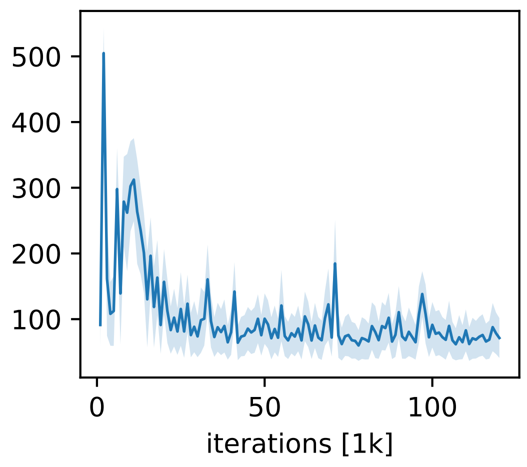
We want our model to be able to reproduce the physics of the high resolution jet image. That's why we look at physical observables like energy of the n^{th} constituent. As reference for the HR/SR-distributions, we additionally plot both LR distributions. That way we

f	2	4	8	16
EMD [GeV]	31.40 ± 10.22	45.46 ± 14.13	67.90 ± 28.85	80.49 ± 27.13

Table 5.1. EMD metrics for different upsampling factors f. Uncertainties are of statistical nature. Information on the image dimensions can be found in Sec. 5.2.

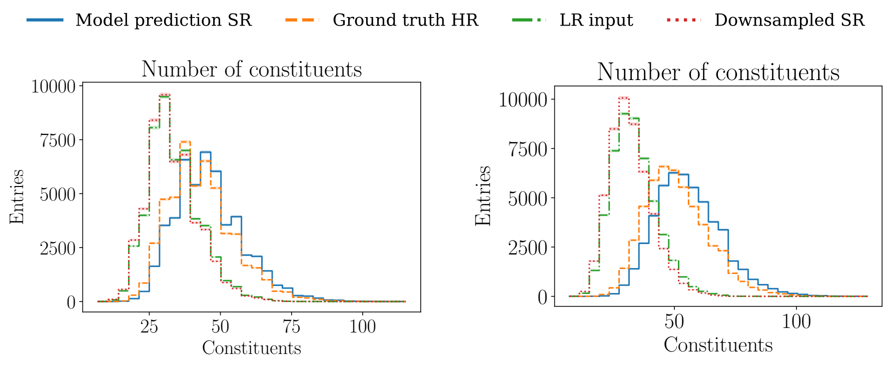


(a) HR L_1



(b) EMD

Figure 5.1. Validation metrics during the training of a $8\times$ model



(a) $2\times$ upsampling **(b)** $8\times$ upsampling

Figure 5.2. Number of constituents

can see how different the HR and LR distributions are, i.e. how much work had to be done by the model, and how good the model approximates the ground truth. We mainly focused on the transverse momentum distributions of the single constituents.
A few of these distributions can be seen in Fig. 5.5. The model can approximate the correct distributions for a wide range of constituents.
Additionally, we compare the number of constituents for SR and HR in Fig. 5.2.

5.1.3. Event-Event Evaluation

When looking at Fig. 5.5 one might get the impression that our model can reconstruct the HR image with great precision. However, this is mostly a statistical effect. The model learns these distributions very well, but on an event by event basis it struggles to assign the correct energy to a constituent. This was to be expected because of the ill-posed nature of the problem. The correlation of the hardest constituent in the HR image and

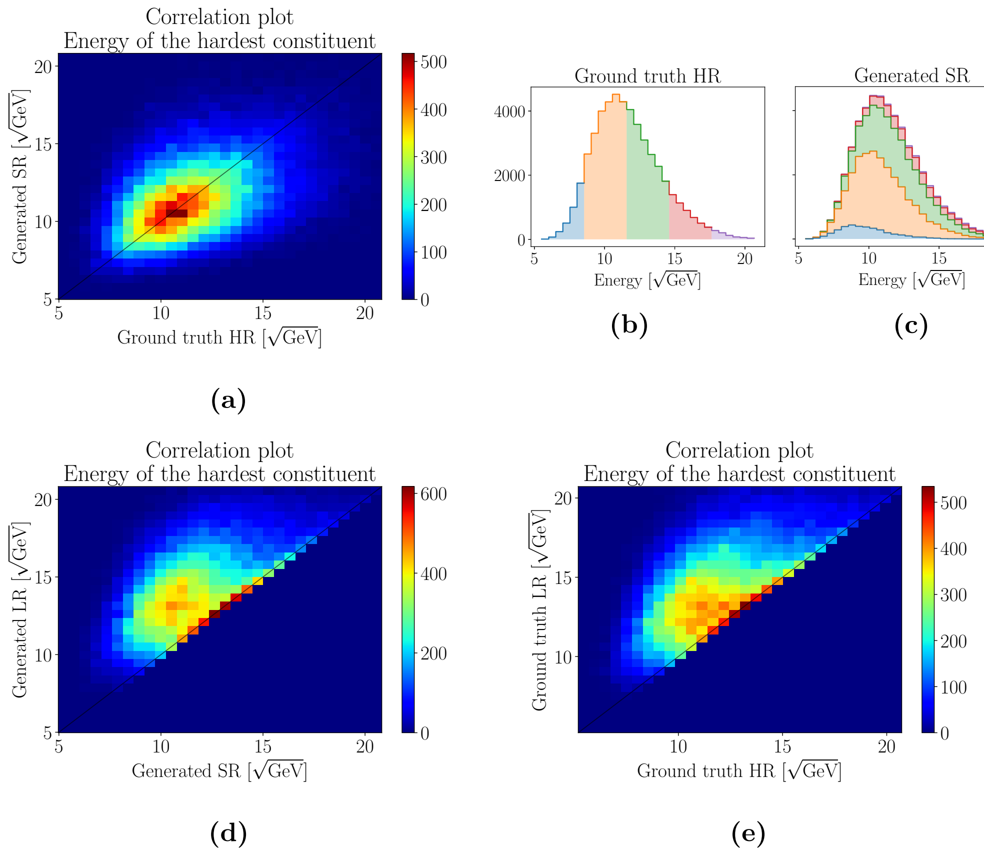


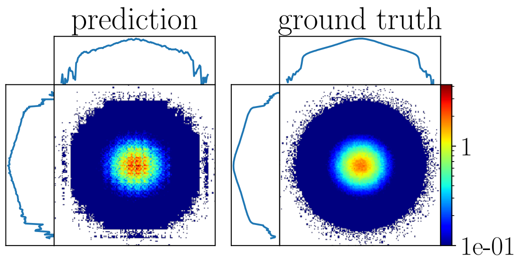
Figure 5.3. Correlation between the energies of the hardest constituent in the HR image and in the SR image

in the SR image can be seen in Fig. 5.3. Especially in Fig. 5.3a the energy spread can be seen. If the model could reconstruct the original HR image, all entries would lie on the bisecting line.

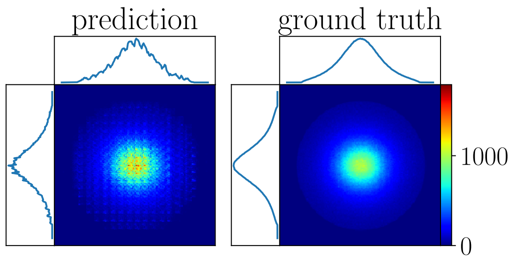
In Fig. 5.3b and Fig. 5.3c the energy range of the ground truth axis from Fig. 5.3a is split into 5 sections. Every event from one section has the same color in both plots. The distributions in Fig. 5.3c are additive. We can see that the model often gets the energy of the hardest constituent wrong. In Fig. 5.3d and Fig. 5.3e the correlations of the high resolution and low resolution images are shown, once for the ground truth and once for the generated images. Because both correlation plots look very similar, we conclude that the model learns how to generate jet images that follow the distributions of the training data, although it rarely assigns the correct energy. That means that the spread is a property of the data set and does not originate from an inadequate model.

5.1.4. Visual Quality

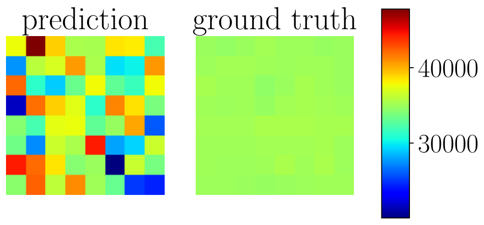
The images generated by our model can follow the physical distributions very closely but the model still can deposit most of the energy in the same HR-pixel every time. In order to detect such artifacts we introduce three new plots.



(a) Average jet image [GeV]



(b) HR constituent histogram



(c) HR patch histogram

Figure 5.4. Visual evaluation for $8\times$ super resolution.

The first plot is simply the average jet image, as it can be seen in Fig. 5.4a. Here we can see regularities, especially at the edges and in the middle of the prediction. Very similar is the second plot, where we look at the histogram of the constituents. For every pixel in the high resolution image, the amount of constituents is counted once for the predicted SR image and once for the HR image (see Fig. 5.4b). To be classified as constituent, the activation needs to be higher than 0.1 GeV. For the last plot we subdivide the high resolution image into $f \times f$ big patches and count the constituents in those segments (see Fig. 5.4c). Each patch corresponds to one pixel in the LR space. Here, the repeated structure from the first two plots is isolated. We can see the habits of the model the best. We can see that the model approximates to the ground truth distributions but there still is room for improvement. Especially in Fig. 5.4c the artifacts of the model are visible. While the ground truth distribution is almost uniform, the prediction clearly favors some pixels over others.

5.2. f times Super Resolution

In this section the final model will be evaluated using the introduced methods.
The ground truth image size for our data has a resolution of 160×160 pixels. At this resolution almost every constituent is in a single pixel. The natural first step in the super resolution task was to test the model on a two times upscaling factor. The ground truth images were scaled down to 40×40 pixels, so that the LR images have a size of 20×20 pixels.
In App. A.1.1 a few observables, the visual quality plots as well as a correlation plot can

be found. The model can approximate the ground truth distributions well for all energy scales. However, when we look at the patch histogram, we can see that the bottom right pixel is favored by the model. The relative difference in the entries is only 5%, but it could be better because these artifacts can be seen in the average jet image aswell.
For $4\times$ and $8\times$ super resolution, the LR image size stays the same but as it can be seen in App. A.1.2, for $4\times$ upscaling, and in Figures 5.1.4, 5.2b, 5.3 and 5.5, for $8\times$ upsampling, the model still manages to learn the distributions. However, the model also has the same shortcomings in distributing the energy evenly across all pixels.
For $16\times$ super resolution the low resolution images had a size of only 10×10 pixels. Here the model still approximates the first few p_T distributions, as shown in App. A.1.3, but the average jet image, as well as the number of constituents were not estimated correctly. The relative difference in the patch histogram counts is $> 50\%$. We conclude that 16-fold super resolution is not possible in those regards with our current model.
It is important to note that for each of these upsampling steps, a new model was trained from scratch, since our model structure does not allow for progressive upsampling.
In summary, it can be said that the model can approximate the observables very well for all factors. However, the visual quality is not perfect yet, even for small upsampling factors. We can see great differences in the patch histogram and it is getting worse for higher factors.

5.3. QCD Jets

In reality more processes are happening than just the top hadronization, so we also tested the model on QCD background jets, without fine tuning it first. The evaluation can be seen in the Appendix A.2. The model almost performs as good on QCD data as it does on top jets, even though the energy distributions and jet images are different. We see that the model is not only limited to the top jet process but can operate in a wider range of processes.

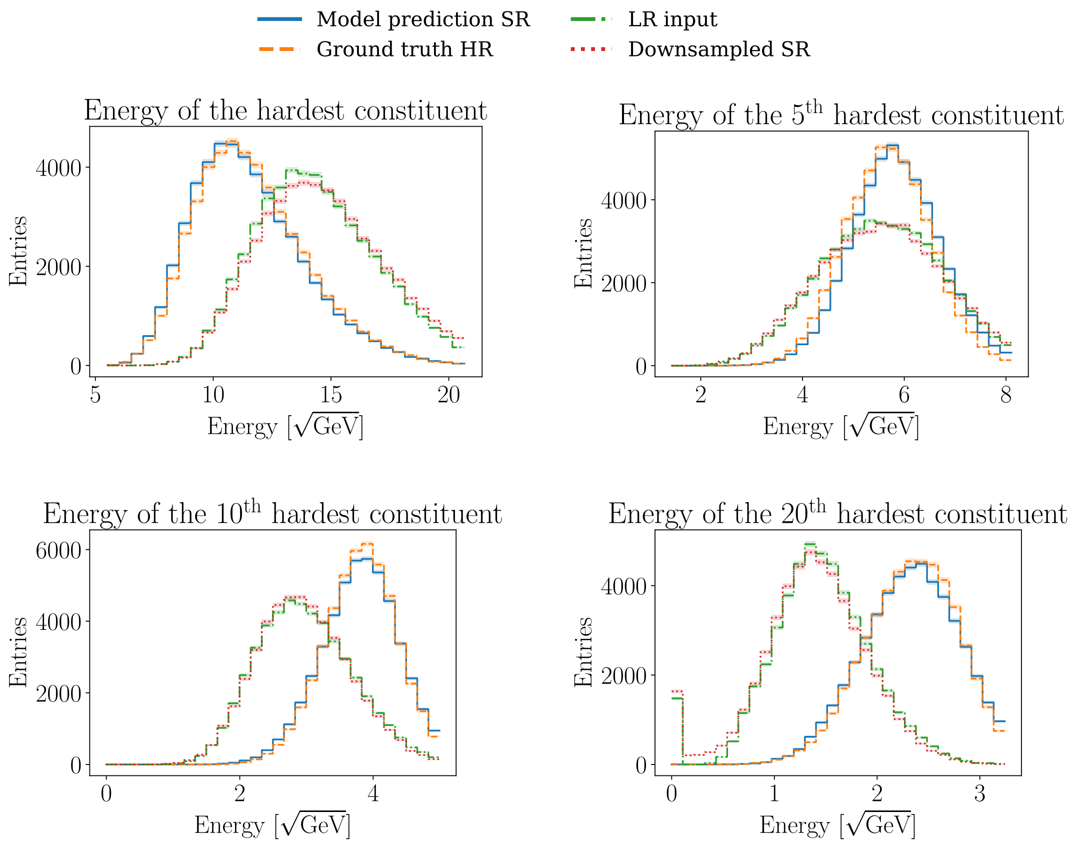


Figure 5.5. Energy distributions of the n^{th} hardest constituent computed over 50k test images for eightfold super resolution. The blue line is the model prediction which should approximate the ground truth in orange (dashed), the green line (dash dotted) is the LR distribution which should be the same as the downsampled SR output in red (dotted). On the x-axis the transverse momenta are plotted in $\sqrt{\text{GeV}}$ in order to get a wider distribution.

6. Experiments and Improvements

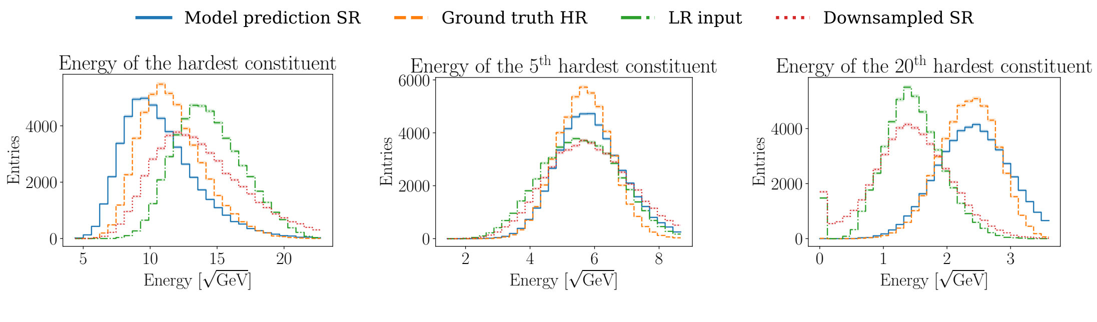
The model described in this thesis is based on models proposed for computer vision [10, 11]. The changes we made and their effect on the performance will be discussed in this chapter.

6.1. Multi-Power Learning

With two separate loss functions which both receive almost the same input, our training setup is rather unconventional. In order to justify both, we train two new models, each with one loss turned off. We expect the model to mainly focuses on the hardest constituents, when only training on the standard loss, since their influence in the L_1 loss is the greatest. Without the standard loss, the hardest constituents should lose importance while the softer particles gain some.
First, we investigate the effect of only training with the power loss. A few observables can be seen in Fig. 6.1. The model assigns too little energy to the hardest constituent, seen in Fig. 6.1a. Otherwise, the distributions align but the counts are not equal.
The results for the model which was only trained on the standard loss can be seen in Fig 6.2. The model performs better for the harder constituents but for the softer ones, the distributions differ from the ground truth. This is in agreement with what we expected. Both losses together compensate for their weaknesses and the generator is able to approximate the distributions for the hard and for the soft constituents.

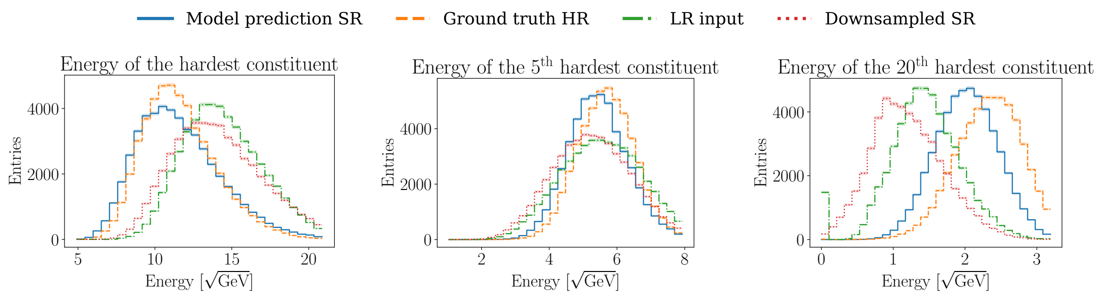
6.2. Gradient Penalty

SRGAN and ESRGAN have no gradient penalty incorporated into their discriminator loss, but we found it to yield much better results for the image quality. It also improved the stability of the training. Before implementing gradient penalty the discriminator could easily distinguish the fake sample from the real sample. But the generator could not



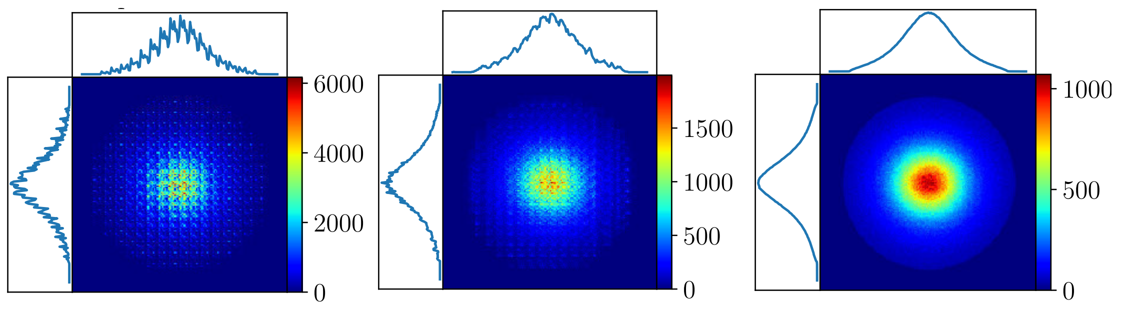
(a) (b) (c)

Figure 6.1. Energy distributions of a model trained only with the power loss



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

Figure 6.2. Energy distributions of a model trained with only the standard loss



(a) 8× average image without gradient penalty **(b)** 8× average image with gradient penalty **(c)** Ground truth average image

Figure 6.3. Comparison between models with gradient penalty (b) and without (a) to the ground truth (c)

improve the quality of super resolved images because the discriminator weights changed too rapidly, and with it the adversarial target changed. The generator learned that not every single one of the HR-pixels have non-zero entries so the model tends to choose the same few pixels to carry most of the energy for every image. After adding gradient penalty, both the adversarial loss for the generator and the discriminator loss saturated. As a consequence the images had fewer artifacts. This can be seen in Fig. 6.3 where the histograms of the average jet image are shown, as explained in Sec. 5.1.4.
Even though the jet images did not follow the average distribution before gradient penalty, the observable distributions did. The same model that produced the average image in Fig. 6.3a was able to approximate the distribution for the n^{th} constituent, seen in Fig. 6.4.

6.3. Conditional Discriminator

A discriminator is called conditional if it is presented with not only the real or fake sample but also with the generator input [22, 37]. In our case that would be the LR image. The idea is that the discriminator learns what a realistic LR image is to a corresponding HR image and not only focuses on the realism of the HR image. This in addition to the LR loss should, in theory, improve the quality of the SR image. The observables are approximated very well but the visual quality of the images stays the same. However, that does not mean

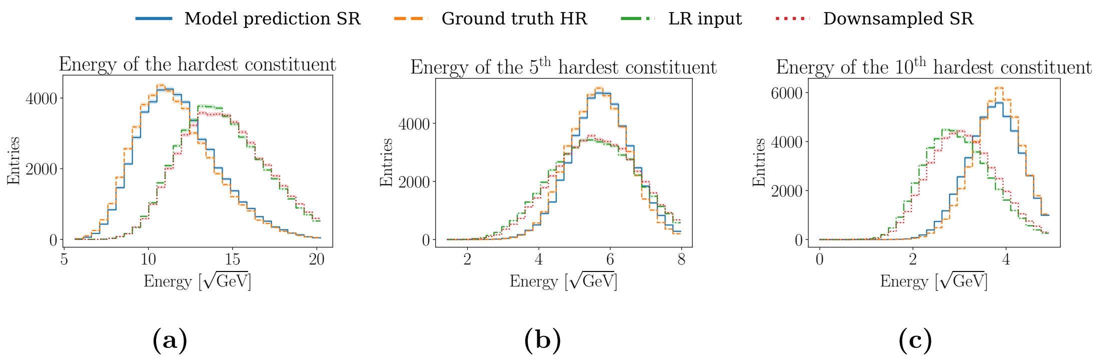
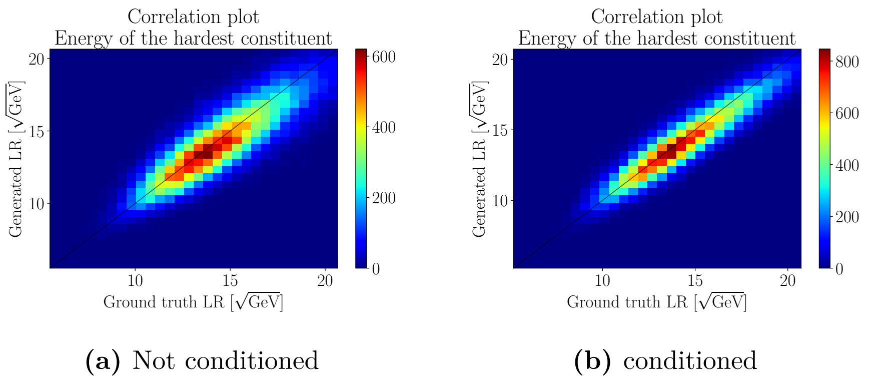


Figure 6.4. Energy distributions of a model without gradient penalty

that the conditional discriminator did not improve some aspect of the performance. When we compare the correlation plots for ground truth LR and downsampled SR, we notice that the spread is much smaller when using a conditional discriminator (see Fig. 6.5). So the condition does improve the predicted SR image in the regard that it fits better to the underlying LR image.



(a) Not conditioned **(b)** conditioned

Figure 6.5. Effect of the conditional discriminator on the downsampled SR image

7. Conclusion

Our modified model of the ESRGAN can successfully estimate a meaningful super resolved image from its pooled low resolution version even for big upsampling factors. The predicted distributions approximate the ground truth well, which is due to the proposed training pipeline that helps the model to focus on all energy scales. The introduction of gradient penalty helped stabilizing the training and reduced the number of artifacts, however some artifacts in the SR image remain. Maybe the generator structure was not adapted enough for our purposes and additional computation in the high resolution space could solve the problem.
The model seems to generalize beyond its training data to some extent. It can successfully super resolve QCD background events in the same p_T range, even though it was only trained on top jets. Additionally, one could explore the idea of a perceptual loss function in the terms of the physical application, by minimizing the feature map difference between SR and HR, calculated from a physical classification network.
The EMD metric is better suited for jet images, especially when we have to work with a small sample size. That is why I would like to replace the high resolution L_1 loss with an implementation of the EMD metric. Also potentially interesting would be the idea of progressive image super resolution like in LapSRN [8] or ProSR [12], or as already mentioned, to explore the possibility of using unpaired data.
Another point is that the model used in this work is deterministic. The same input image will always get super resolved to the same SR image. One could incorporate some variational component into the forward pass and check if it solves the average jet image distribution problem.
The work in this thesis serves as a proof of concept. We have not yet tested or fine tuned the model on unseen FCAL data, which was the motivation of the project.

A. Appendix

A.1. f times Super Resolution - Evaluation Plots

A.1.1. 2x Super Resolution

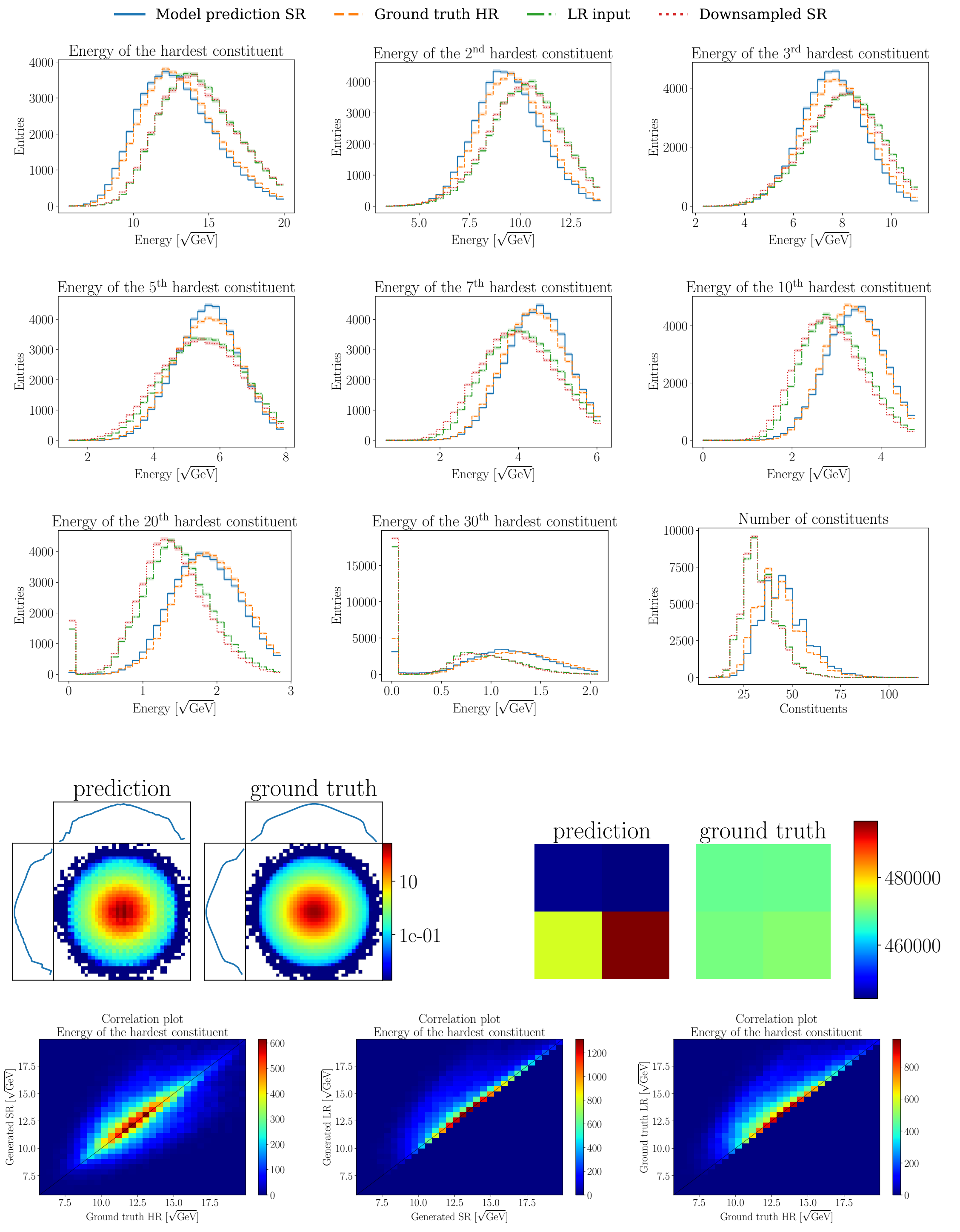


Figure A.1. Evaluation of a model trained for 2× super resolution on the top jet test set

A.1.2. 4x Super Resolution

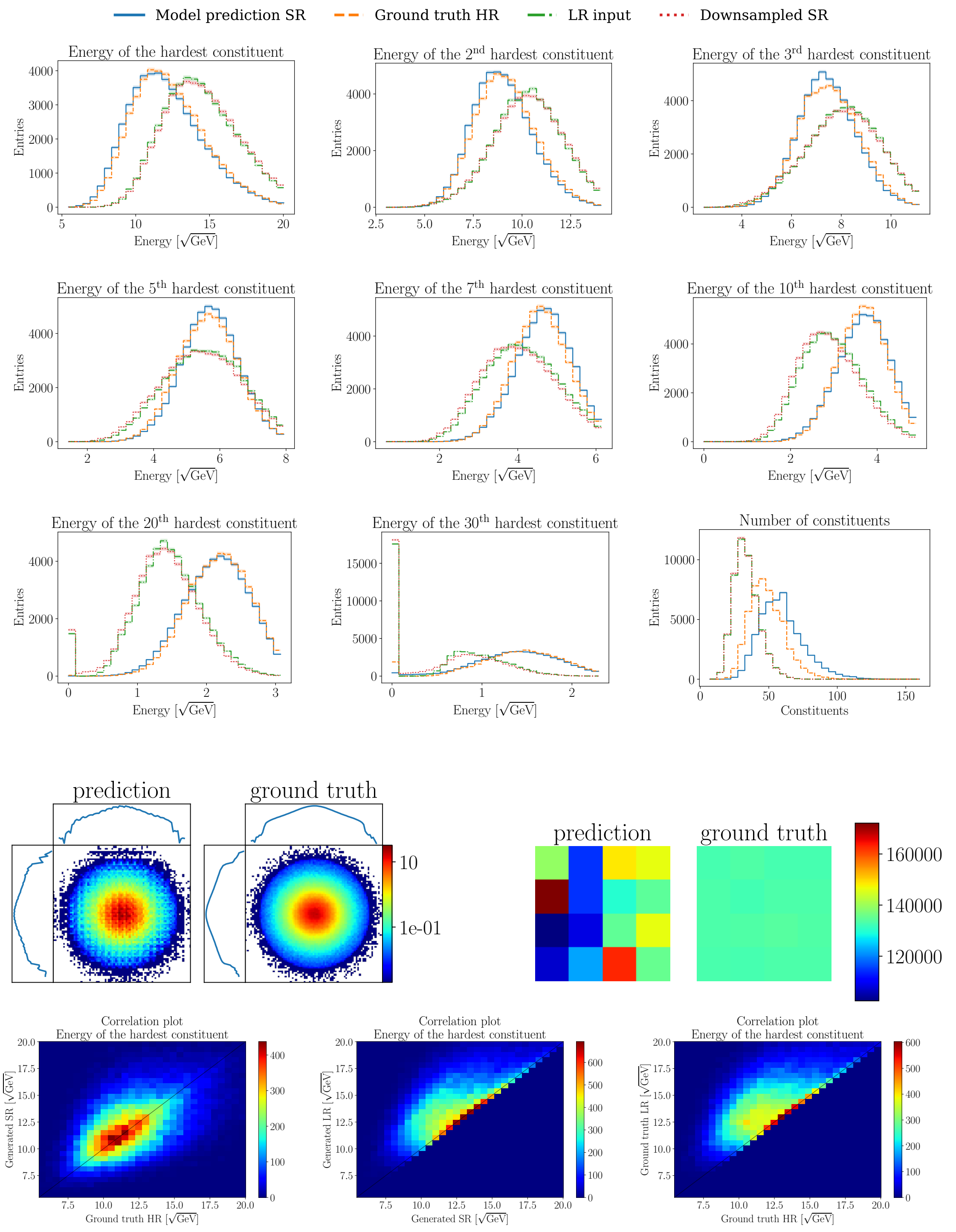


Figure A.2. Evaluation of a model trained for 4× super resolution on the top jet test set

A.1.3. 16x Super Resolution

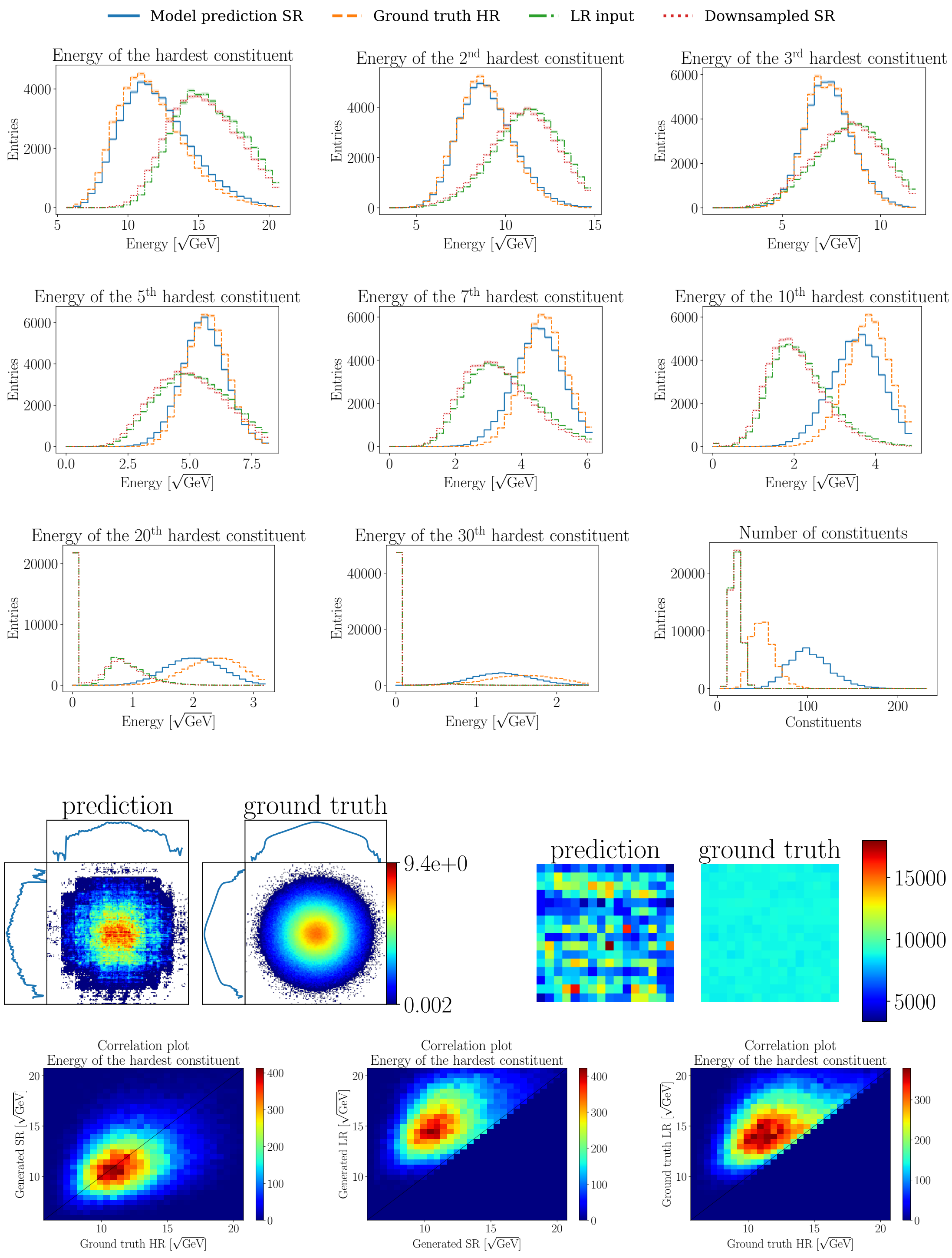


Figure A.3. Evaluation of a model trained for 16× super resolution on the top jet test set

A.2. Evaluation on QCD jets (8×)

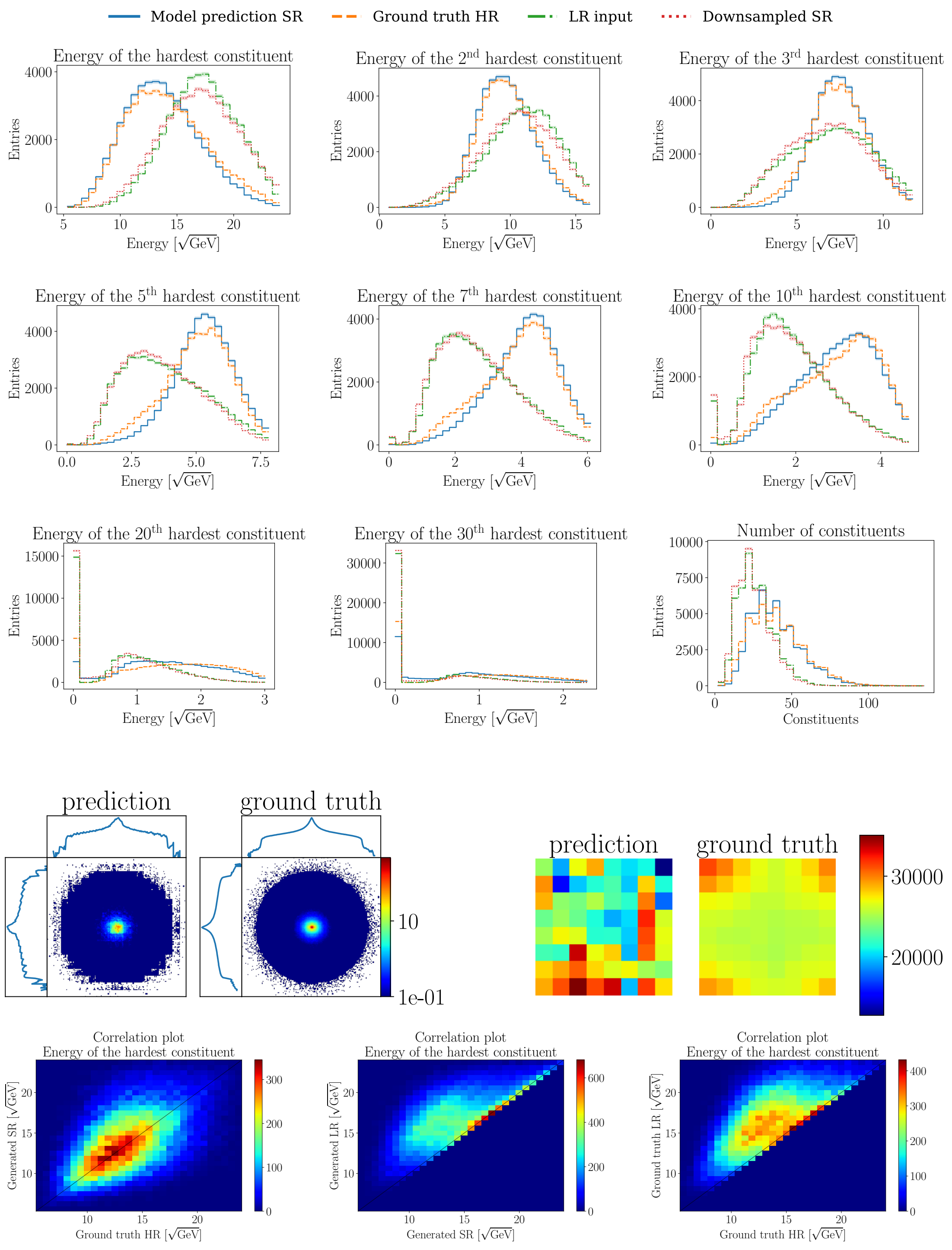


Figure A.4. Evaluation on QCD background of a model trained on top jets with 8× upsampling

References

[1] G. Aad et al. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: *Physics Letters B* 716.1 (2012), pp. 1–29 (cit. on p. 1).

[2] The ATLAS Collaboration et al. “The ATLAS Experiment at the CERN Large Hadron Collider”. In: *Journal of Instrumentation* 3.08 (2008), S08003–S08003 (cit. on p. 1).

[3] Valerio Rossetti. *Performance of the ATLAS Calorimeters and Commissioning for LHC Run-2*. [Online; accessed 23-February-2020]. 2015 (cit. on p. 1).

[4] Chao Dong et al. “Learning a Deep Convolutional Network for Image Super-Resolution”. In: *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014, pp. 184–199 (cit. on p. 1).

[5] Chao Dong et al. *Image Super-Resolution Using Deep Convolutional Networks*. 2014. arXiv: `1501.00092 [cs.CV]` (cit. on p. 1).

[6] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. *Accurate Image Super-Resolution Using Very Deep Convolutional Networks*. 2015. arXiv: `1511.04587 [cs.CV]` (cit. on p. 1).

[7] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. *Deeply-Recursive Convolutional Network for Image Super-Resolution*. 2015. arXiv: `1511.04491 [cs.CV]` (cit. on p. 1).

[8] Wei-Sheng Lai et al. *Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution*. 2017. arXiv: `1704.03915 [cs.CV]` (cit. on pp. 1, 27).

[9] Bee Lim et al. *Enhanced Deep Residual Networks for Single Image Super-Resolution*. 2017. arXiv: `1707.02921 [cs.CV]` (cit. on p. 1).

[10] Xintao Wang et al. “ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks”. In: *CoRR* abs/1809.00219 (2018). arXiv: `1809.00219` (cit. on pp. 1, 12, 13, 23).

[11] Christian Ledig et al. “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”. In: *CoRR* abs/1609.04802 (2016). arXiv: `1609.04802` (cit. on pp. 1, 12, 14, 23).

[12] Yifan Wang et al. *A Fully Progressive Approach to Single-Image Super-Resolution*. 2018. arXiv: `1804.02900 [cs.CV]` (cit. on pp. 1, 27).

[13] Simone Marzani, Gregory Soyez, and Michael Spannowsky. “Looking Inside Jets”. In: *Lecture Notes in Physics* (2019) (cit. on p. 3).

[14] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural Networks* 4.2 (1991), pp. 251–257 (cit. on p. 5).

[15] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 315–323 (cit. on p. 5).

[16] Henry J Kelley. “Gradient theory of optimal flight paths”. In: *Ars Journal* 30.10 (1960), pp. 947–954 (cit. on p. 7).

[17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: `1412.6980 [cs.LG]` (cit. on p. 7).

[18] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385` (cit. on p. 7).

[19] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: `1406.2661 [stat.ML]` (cit. on p. 8).

[20] Alexia Jolicoeur-Martineau. “The relativistic discriminator: a key element missing from standard GAN”. In: *CoRR* abs/1807.00734 (2018). arXiv: `1807.00734` (cit. on pp. 8, 9, 12).

[21] Ishaan Gulrajani et al. “Improved Training of Wasserstein GANs”. In: *CoRR* abs/1704.00028 (2017). arXiv: `1704.00028` (cit. on pp. 9, 14).

[22] Phillip Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *CoRR* abs/1611.07004 (2016). arXiv: `1611.07004` (cit. on pp. 9, 12, 24).

[23] Chuan Li and Michael Wand. “Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks”. In: *CoRR* abs/1604.04382 (2016). arXiv: `1604.04382` (cit. on p. 9).

[24] Andrew Y. Ng Adam Coates Honglak Lee. “An Analysis of Single Layer Networks in Unsupervised Feature Learning”. In: (2011) (cit. on pp. 11, 14).

[25] Anja Butter et al. “Deep-learned Top Tagging with a Lorentz Layer”. In: *SciPost Physics* 5.3 (2018) (cit. on p. 11).

[26] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. “A brief introduction to PYTHIA 8.1”. In: *Computer Physics Communications* 178.11 (2008), pp. 852–867 (cit. on p. 11).

[27] J. de Favereau et al. “DELPHES 3: a modular framework for fast simulation of a generic collider experiment”. In: *Journal of High Energy Physics* 2014.2 (2014) (cit. on p. 11).

[28] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. “FastJet user manual”. In: *The European Physical Journal C* 72.3 (2012) (cit. on p. 12).

[29] Matteo Cacciari and Gavin P. Salam. *Dispelling the N^3 myth for the Kt jet-finder*. 2005. arXiv: `hep-ph/0512210 [hep-ph]` (cit. on p. 12).

[30] Matteo Cacciari, Gavin P Salam, and Gregory Soyez. "The anti-ktjet clustering algorithm". In: *Journal of High Energy Physics* 2008.04 (2008), pp. 063–063 (cit. on p. 12).

[31] Yuan Yuan et al. *Unsupervised Image Super-Resolution using Cycle-in-Cycle Generative Adversarial Networks.* 2018. arXiv: `1809.00437 [cs.CV]` (cit. on p. 12).

[32] Yulun Zhang et al. *Residual Dense Network for Image Restoration.* 2018. arXiv: `1812.10477 [cs.CV]` (cit. on p. 13).

[33] Wenzhe Shi et al. *Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network.* 2016. arXiv: `1609.05158 [cs.CV]` (cit. on p. 13).

[34] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09.* 2009 (cit. on p. 14).

[35] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.* 2015. arXiv: `1511.06434 [cs.LG]` (cit. on p. 15).

[36] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. "Metric Space of Collider Events". In: *Physical Review Letters* 123.4 (2019) (cit. on p. 17).

[37] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets.* 2014. arXiv: `1411.1784 [cs.LG]` (cit. on p. 24).

List of Figures

Acknowledgements

I would like to thank Tilman Plehn for the opportunity to work on this interesting topic and his advice. Furthermore, I would like to thank Anja Butter and Jennifer Thompson for helpful discussions and pointing Fabian and me in the right direction, and Christopher Lüken-Winkels for proofreading this thesis.
Finally, I thank the whole group for their kindness and the nice working atmosphere.

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 02.03.2020

Lukas Blecher