Department of Physics and Astronomy

University of Heidelberg

Master thesis

in Physics

submitted by

Fabian Sascha Keilbach

born in Heidelberg

2020

Super-resolving Jet Images

This Master thesis has been carried out by Fabian Sascha Keilbach

at the

Institute of Theoretical Physics

under the supervision of

Prof. Tilman Plehn

Abstract:

Jets are a substantial background in any LHC analysis so studying their substructure is essential in the search for new physics but these studies suffer from poor detector resolution in the forward region. The recent advent of deep generative models and their impressive performance on images offer a valuable opportunity to investigate if it is possible to enhance detector resolution through machine learning. We apply a Generative Adversarial Network to QCD- and Top-Jets and show that it is possible to super-resolve calorimeter images.

Zusammenfassung:

Jede LHC Analyse muss Hintergrundprozesse durch Jets beachten, daher ist ein genaues Verständniss der Substruktur bedeutend in der Suche nach neuer Physik. Dies wird jedoch durch eine geringe Detektorauflösung in Vorwärtsrichtung erschwert. Das Aufkommen generierender Modelle und deren beachtliche Leistung in der Anwendung auf Bilder ist eine willkommene Möglichkeit zu untersuchen ob es möglich ist die Detektorauflösung durch machinelles Lernen zu verbessern. Wir wenden ein Generatives Modell auf QCD- und Top-Daten an und zeigen, dass wir dadurch Kalorimeter Bilder super-auflösen können.

Contents

1 Introduction

The standard model of particle physics, including Einstein's theory of gravitation, is by far the best known description of all fundamental forces. Despite its success there remain open questions like the full mechanism of electroweak symmetry breaking, related to the hierarchy problem [1], the nature of dark matter [2] or the mechanism giving rise to neutrino masses [3]. Even questions purely within the standard model remain to be investigated, for example precision studies of the Higgs boson. Because these questions are so fundamental one needs to go to sufficient high energies to study them and only modern particle accelerators can achieve this, among them the LHC is leading.
Whatever the question is, jets play an important role at hadron colliders. Often they are responsible for dominating backgrounds and the level of precision achieved must be matched by the level these backgrounds can be understood and subtracted. This is especially difficult in the forward region where detector resolution is much lower than in the central region but often substantial amount of information could be derived in this regime due to the nature of collinear splittings. Traditionally cut based approaches have been employed as a first step for this task [4] but more and more alternative approaches manage to outperform them. Leading candidates include those based on machine learning. Often highly advanced techniques that have been developed in a pure machine learning setting, can be used for physics tasks, for example to speed up analyses [5].
Boosted Decision Trees have been employed for such tasks with great success [6] but modern computational resources allow for the application of deep neural networks that outperform those long standing approaches. Among them generative networks have been granted a great deal of attention leading to networks that can copy famous painters or compose music. In this thesis we will use these powerful generative models to investigate if we can take them a step further in physics applications and truly have them improve measurements. Specifically we will attempt to enhance jet image resolutions, driven by the aforementioned reality of low detector resolution in the forward region.
While we will not go as far as a real experimental setup would demand, we will show that a modified generative network can perform such an enhancement for the important cases of top- and qcd-jets as well as showing that these are not merely learned on a case by case basis but that the network can make use of underlying principles to enhance resolution of jet types it has not seen during training.
This thesis is organized as follows: In Section 2 we give a brief overview of jets at hadron colliders and introduce the relevant variables to describe calorimeter entries in terms of images. Section 3 gives an overview of the most basic machine learning

principles allowing a network to learn. In Section 4 we specify our task and introduce the generative model used in the analysis. Section 5 shows the principle ability of the model to enhance resolution in the simplest cases while Section 6 is dedicated towards an optimization of the model applied to top jets. In Section 7 we investigate difficulties when applied to qcd jets and in Section 8 we perform a detailed study of what we found to be an optimal setup. A summary of the results and an outlook for future work is given in Section 9.
Most of the relevant results of this thesis will be implicitly or explicitly part of a paper scheduled for submission [7]. In turn some of the interpretations of our results are derived from there. A substantial part of this work was done in close collaboration with Lukas Blecher and especially results and illustrations in the first sections have already been used in his bachelor thesis [8].

2 Jets at the LHC

2.1 Hadron Colliders

The Large Hadron Collider (LHC) is the most powerful operating machine for probing the standard model of particle physics as well as the search for beyond standard model effects [9]. Its status as a discovery machine relies on the unrivaled center of mass energy of $\sqrt{s} = 13$ TeV, scheduled to be increased to 14 TeV in the future [10]. To reach such energies one must control the energy loss due to synchrotron radiation which scales $\propto (\frac{E}{m})^4$ (here E is the beam energy and m the particle mass). This strong mass dependence suggest to collide heavy particles in order to achieve high center of mass energy, namely protons at the LHC which have over 1800 times the mass of the electron. However, protons are not fundamental particles but complicated objects with a lot of substructure. At the heart, protons are described by Quantum Chromodynamics (QCD) and its fundamental degrees of freedom the spin 1/2 quarks and spin 1 gluons. From the qcd point of view a proton consists of three valence quarks that constantly interact with each other through gluon exchange, inducing the appearance of sea-quarks. A crucial fact here is the asymptotic freedom of qcd, which means that the running of the strong coupling α_s [11] is such that the theory is perturbative in the UV regime but strongly coupled in the IR. This strong coupling at low energies means that the distribution of energy inside a proton, as described by parton distribution functions (pdf) [12], cannot be calculated from first principles but rather has to be fitted to measurements. This complicated internal structure leads to many complications as compared to the cleaner signals provided by lepton colliders [13], for example the center of mass energy of the underlying hard process ie. the scattering of partons inside the proton, can be any fraction of the collider center of mass energy. Therefore the total cross section of a given hadronic process is an intricate convolution of parton level sub-processes, usually encapsulated through the QCD factorization theorem [14].

Quarks and gluons cannot be directly observed. Partons created in a hard scattering process undergo different stages until they reach the detector. Up until energies of order GeV they lose energy by showering additional particles, mainly due to qcd interactions. This can formally be expressed through splitting kernels [15]. For the showering stage phenomenological effective describtions exists, however the following hadronization, ie. the combination of partons into color singlet hadrons, is completely non-perturbative. Those hadrons or their decay products are actually observed by particle detectors. In addition electroweak particles are also emitted during events but typically with smaller cross sections.

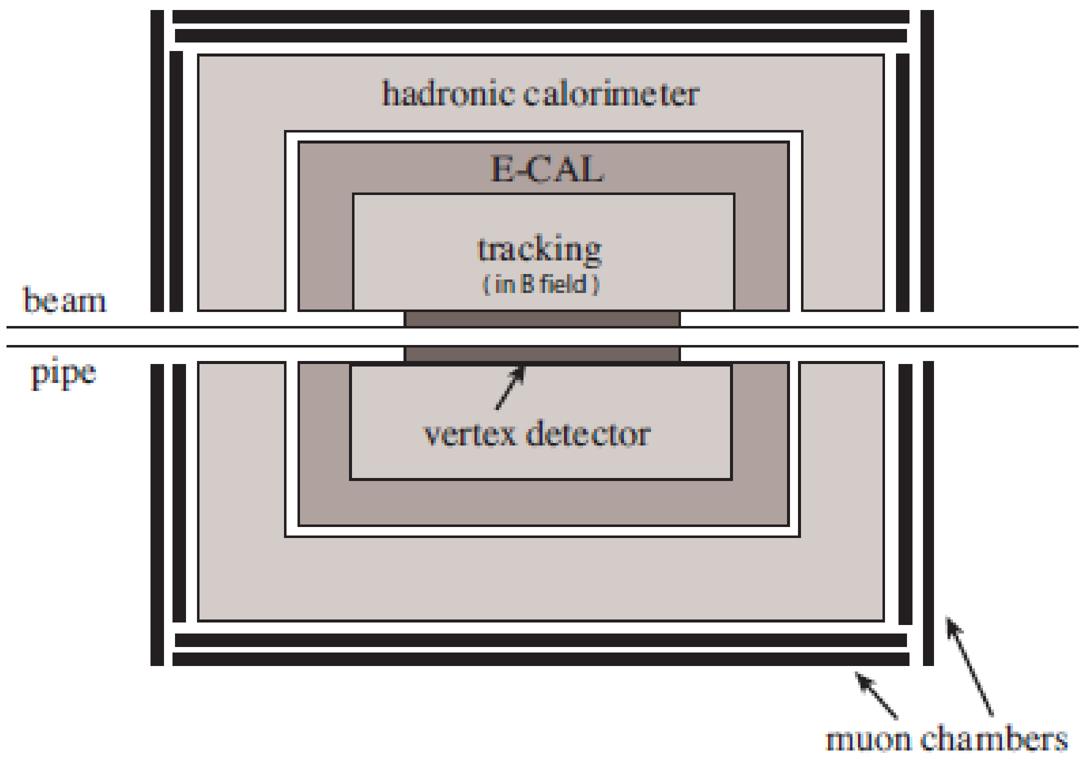

Figure 2.1: Schematic detector layout. Figure taken from Ref. [13]

The detection of particles is schematically shown in Figure 2.1: Particles leaving the interaction point enter the tracking system which is permeated by a strong magnetic field such that charged particle paths are bend while transversing the tracker. From the curvature of the tracks the momentum to charge ratio can be inferred. After the tracker comes the calorimeter system. In the electromagnetic calorimeter particles like leptons and photons deposit most of their energy in the form of an electromagnetic shower cascade whereas hadrons only deposit small amount of energy. This situation is reversed in the hadronic calorimeter where the hadrons deposit their energy in the form of hadronic showers. The shower information can be used to identify the particle and reconstruct its energy. After the hadronic calorimeter most particles have been stopped but muons can manage to advance into muon chambers since they are heavier than electrons but do not decay as quickly as the tau. The muon chamber is basically a coarser tracker and again the bending of the muon path is used for reconstruction. Figure 2.2 schematically shows the signature of some particles in the calorimeter system.

Since the trackers can only detect charged particles the calorimeter information is crucial in most events and it is desirable to have a high resolution in all detector regions. However at a functional level the calorimeter is divided into cells of given

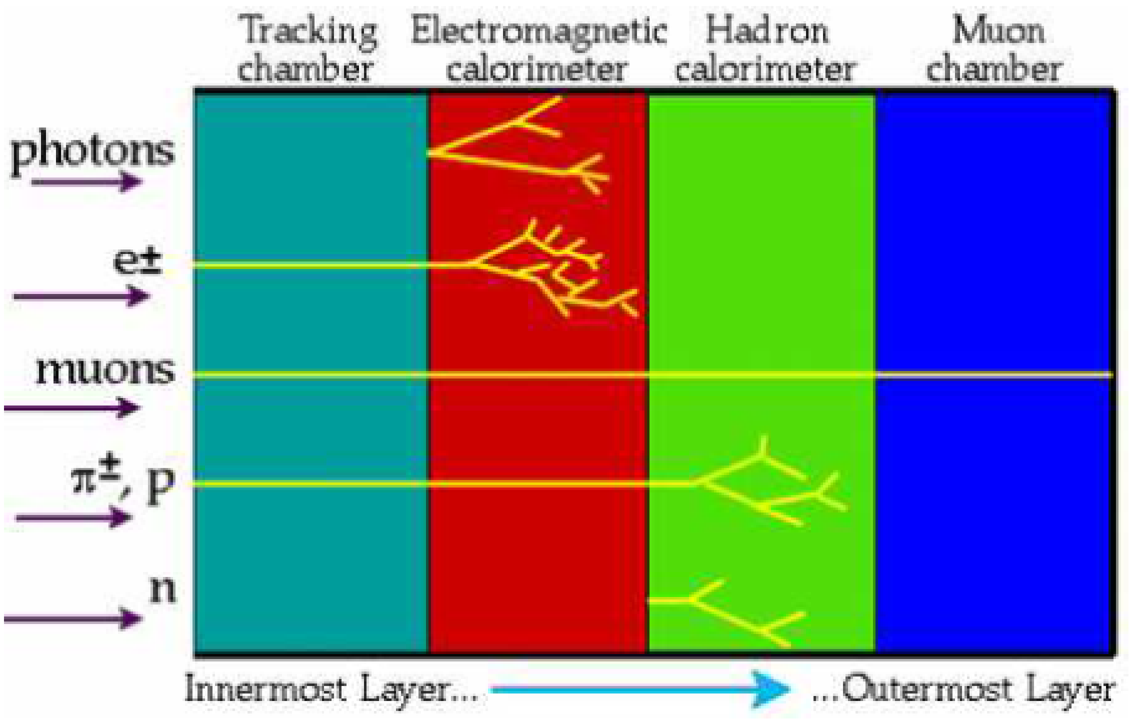

Figure 2.2: Signature of different particles in the calorimeter system. Figure taken from Ref. [13]

resolution and the detectors employed at LHC are such that that resolution is high perpendicular to the beam axis but becomes increasingly coarser. This is insofar unfortunate as the qcd splitting of partons possess a collinear divergence [16] which suggests that substantial amount of energy is showered into the soft and/or collinear regime, ie. into partons with low amount of transverse momentum

$$p_T = \sqrt{(p_x)^2 + (p_y)^2} \tag{2.1}$$

where p_x, p_y, p_z is the particles 3-momentum and at small angles Θ relative to the beam axis. Here we follow usual convention and define the beam axis to be in the z-direction.

Although the pp-collision takes place into the center-of-mass frame, the hard process does not necessarily do so since the energy fraction x of the partons in relation to the proton energy will in general be different for the interacting partons, leading to a boost in the beam direction. One would therefore like to describe the unfolded calorimeter plane in coordinates that are boost invariant or at least are additive under boosts. This is obviously true for the azimuthal angle around the beam pipe ϕ but not for the angle Θ. However it can be shown that the rapidity

$$y = \frac{1}{2} \ln \frac{E + p_z}{E - p_z} \tag{2.2}$$

is additive under boosts [13]. In the massless limit, which in this thesis we will always assume for particles reaching the detector, $E \approx |\vec{p}|$ and the rapidity is equal to the pseudo-rapidity

$$\eta = \ln \cot \frac{\theta}{2} \tag{2.3}$$

Thus the pseudo-rapidity, azimuthal angle and transverse momentum are an equivalent describtion of the 4-momentum for a massless particle, with the formula

$$p^\mu = (p_T \cosh \eta, p_T \sin \phi, p_T \cos \phi, p_T \sinh \eta) \tag{2.4}$$

and the description of calorimeter images in terms of (p_T, η, ϕ) allows us to view them like 1 channel 2d images. The central region of the calorimeter is usually taken to be the region $|\eta| < 2.5$ and possesses high spatial resolution, the forward region $2.5 < |\eta| < 5$ has low resolution.

2.2 Jets

Due to the collinear splittings a parton created in the hard process will show up in the detector as a spray of collimated meta-stable particles. Since we are interested in fundamental physics we would ideally be able to fully reconstruct the parton from which the spray originated but since showering is a stochastic process this is not possible in general. Instead, based on a distance metric, we cluster the collimated particles into an object called jet [17]. From a theoretical point it is surprisingly difficult to give a clear definition of jets, for example one has to keep in mind IR-safety [18].
In practice the most used jet algorithms are the GENERALIZED KT-ALGORITHMS [19; 20; 21; 22] :

1. Define the distances

$$d_{ij} = \min(p_{Ti}^{2l}, p_{Tj}^{2l}) \frac{\Delta R_{ij}{}^2}{R^2} \text{ (kT distance)} \tag{2.5}$$

$$d_{iB} = p_{Ti}^{2l} \text{ (beam distance)} \tag{2.6}$$

   for all particles, with

$$\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2} \tag{2.7}$$

   and R the jet-radius parameter.

2. Find the minimal distance d_{min}. If it is in the d_{ij} set combine particles i and j into one object with summed four momentum. If it is in the d_{iB} set remove particle i from the list and consider it a jet.

3. Repeat steps 1 and 2 until the lists are exhausted.

Typical values for the parameter l are 1 (kT algorithm), -1 (anti-kT algorithm) and 0 (Cambridge-Aachen algorithm).

2.2.1 Top Jets

The top, with a mass of roughly 172 GeV [23], is by far the heaviest among all quarks. This leads to the important observation that tops decay through electroweak interactions before they can hadronize. The dominating decay channel is $t \to b + W^+$ with the W further decaying into a quark anti-quark pair $q\bar{q'}$, the tree level Feynman diagram is depicted in Figure 2.3. Therefore the top displays a 3-prong signature as compared to the single-prongness of jets initiated by non-top quarks and for this reason we will refer to top jets and qcd jets in what follows, implicitly excluding the top from the latter.

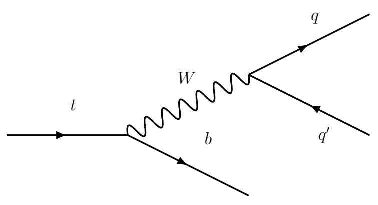

Figure 2.3: Purely hadronic top decay. Diagram created with [24].

3 Machine Learning and Neural Networks

Machine learning, in its broadest sense, is the field of automated learning by an artificial system through analyzing examples and making use of self-learning algorithms with the intent of enabling the system to solve a task despite not being explicitly programmed to do so. In a more narrow sense one wants the system to learn a mapping f between given inputs x and desired outputs y. The systems considered in this thesis are neural networks [25] which have become prime machine learning tools after the advent of powerful GPUs.

3.1 Fully Connected Neural Networks

The underlying mathematical foundation of neural networks are simple linear transformations, encapsulated in the basic units called perceptrons [26]. A perceptron takes an input x and transforms it according to

$$z = Wx + b \tag{3.1}$$

where W and b are trainable parameters of the perceptron denoted as weight and bias respectively. A neural network, at the most basic level, is simply a set of perceptrons. Conventionally the network is divided into an input layer, hidden layers and output layer. In a fully connected (or dense) neural network all perceptrons of layer l are connected (in the sense of receiving input) to all perceptrons of layer (l-1), see Figure 3.1, save for the input layer. The algorithm performed by a layer of perceptrons then becomes a linear transformation of vectors, so that Eq. 3.1 is altered to

$$\vec{z} = W\vec{x} + \vec{b} \tag{3.2}$$

where now W is a matrix and z, x and b are vectors.
A crucial step is the inclusion of *activation functions* ϕ for at least two reasons: So far the network as defined by Eq. 3.2 can only perform linear transformations but most tasks demand the network to learn non-linear mappings. Such nonlinearities are introduced by activation functions. Further, the output z will in general be a real number but often it is advantageous to map the real line into a finite interval, for example in classification tasks where then a final layer output can be interpreted as a probability. Common choices of activation functions are (leaky) ReLU [27], hyperbolic tangent or, historically, sigmoids. Thus Eq. 3.2 becomes

$$\phi(\vec{z}) = \phi(W\vec{x} + \vec{b}) \tag{3.3}$$

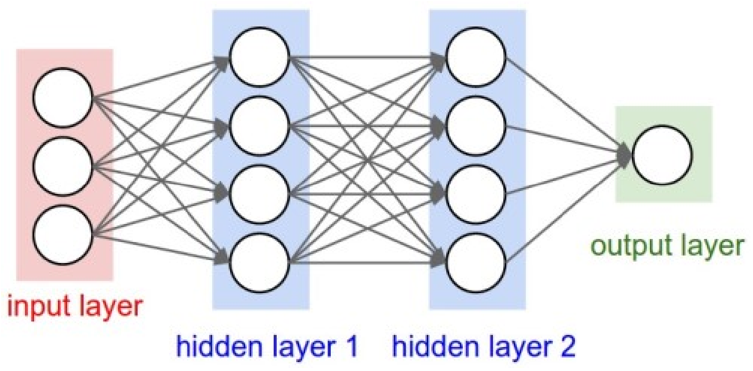

Figure 3.1: Fully connected neural network with 2 hidden layers. Figure taken from Ref. [28]

3.2 Convolutional Neural Networks

It can be shown that fully connected networks are universal function approximators [29] so in principal the dense network described in the previous section could learn any desired mapping f. In practice fully connected networks can only be employed for a limited range of tasks since computational resources and time is limited and the number of learnable parameters (ie. the weights and biases) rapidly scale with the number and dimension of hidden layers. Moreover, it might not even be desirable to connect every unit of a layer to each of the next layer since that might 'hide' relevant information by overloading the receiving perceptron, making it more difficult to filter out the relevant inputs. This is for example the case in arguably the most important subset of machine learning tasks: Image recognition, a field that benefits heavily from convolutional architectures. The design of convolutional layers is guided by two principles commonly found in images: locality and translation invariance [30]. Locality means that distinctive features in an image are encapsulated by clusters of neighboring pixels, for example facial features: To recognize an eyeball in the top left corner one does not need the pixels in the bottom right corner of the image. Translation invariance is the formalized notion that these local distinctive features can in general show up anywhere in the picture or appear more than once. Informally a convolutional layer is a $k \times k$ dimensional window that slides vertically and horizontally across the image and detects a feature, through the learned weights, within this $k \times k$ sub-image. Such a window is called a kernel and the result of a convolutional transformation is called feature map. A crucial step is that the weights are fixed during the sliding process (weight sharing) which assures that the same feature is looked for across the entire image. Convolutional layers are defined through a number of parameters, the most important ones are:

1. Number of Filters C: The number of different kernels employed. Each map has an independent set of weights and thus can extract different features.

2. Kernel size k: The size of the window.

3. Stride: The amount of pixels the window is displaced each time it is moved across the image.

An example convolutional operation is depicted in Figure 3.2 where the input image is a 3×3 tensor, the kernel size is 2×2 and the stride is 1. The shaded pixels display the computation of the first output pixel:

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

So far we have implicitly described convolutional layers acting on one channel

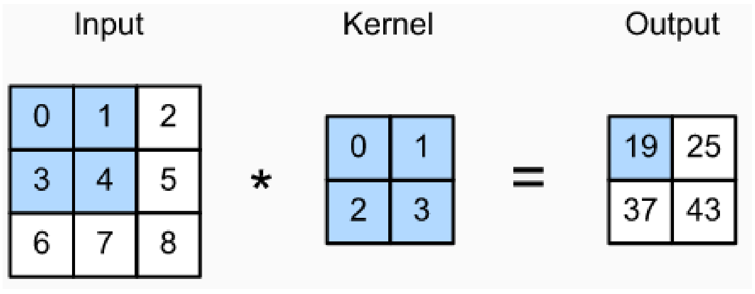

Figure 3.2: Example of a two dimensional convolution. Figure taken from Ref. [30]

two dimensional images but all of the above can straightforwardly be generalized to arbitrary numbers of channels and dimensions. In case of multi-channel inputs (which often arise in hidden layers) a given kernel runs over each channel separately and then adds up all resulting feature maps.

3.2.1 Pooling Layers

A special case of convolutional layers are pooling layers. These have fixed weights and are used to reduce image dimensionality and keeping only the most important information in a feature map. Max-pooling layers with a $k \times k$ kernel slide over the feature map and keep only the hardest pixel in the $k \times k$ sub-window. A very important pooling variant for our purpose is sum-pooling, where the values of the $k \times k$ pixels are summed up into one output pixel.

3.3 Loss Function and Training Procedure

If we want a neural network to solve a given task it has to first learn (or rather approximate) the implicitly desired mapping and then generalize to unseen data. The first part is called the training stage, the latter evaluation stage. Consequently, one usually uses at least two different data sets: A training set on which the model learns and an evaluation sets on which the model performance is measured.
In this thesis we will only make use of *supervised learning* in which every input x comes with an (abstract) label that corresponds to the desired output y. In this setting the model's learning progress can be expressed through a *loss function* L via:

$$L(y, y') = L(y, f(x|\theta)) \tag{3.4}$$

Here y is the truth label, $y' = f(x|\theta)$ is the model output and θ are the weights parametrizing the model. The loss function has to be chosen according to the task and model but in any case it is constructed such that the ideal mapping f corresponds to a global minimum of the loss function L.
Common choices of loss functions are L1 loss, L2 loss and Binary Cross Entropy:

$$L_1 = \sum_{i=1}^{n} |y_{truth} - y_{predicted}| \tag{3.5}$$

$$L_2 = \sum_{i=1}^{n} (y_{truth} - y_{predicted})^2 \tag{3.6}$$

$$L_{BCE} = \sum_{i=1}^{n} [y_{truth} \log(y_{predicted}) + (1 - y_{truth}) \log(1 - y_{predicted})] \tag{3.7}$$

where n denotes the number of samples. For simple tasks, like linear regression, it can be possible to derive the minimum of L analytically but in general this is not possible. One then has to use numerical approximation methods like *mini-batch stochastic gradient descent* [31]:
At the core this algorithm relies on a simple Taylor approximation under a small update of the weights $w \to w + \delta w$ (we now consider the loss L to be a function of the weights w and suppress additional labels)

$$L(w + \delta w) = L(w) + \delta\vec{w} \cdot \nabla_w L(w) + O((\vec{\delta w})^2) \tag{3.8}$$

The upshot is that this leads to a minimization of L if we chose the update δw to be proportional to minus the absolute value of the loss function's gradient so that our iterative update rule becomes

$$\vec{w} \leftarrow \vec{w} - \frac{\eta}{|B|} \sum_{i \in B} |\partial_w L(w)| \tag{3.9}$$

The computation of the gradient is done by a numerical algorithm commonly known as backpropagation [32].

In Eq. 3.9 we swiftly introduced two important notions: The learning rate η and the mini-batch B. η is a hyperparameter, ie. a parameter not learned but chosen before training that can be regarded as part of the model architecture, that determines how much the network moves per iteration in the loss function's manifold. If it is picked to be too big then the model might overshoot and into a region away from the minimum, if it is too small the model may never converge in feasible time. A mini-batch consists of a subset of available training data, using mini-batches speeds up training significantly but the batch size has to be big enough to reasonably approximate the average loss as compared to a computation over all training data. Based on this it seems a good idea to use a dynamic learning rate, bigger at the beginning of training and getting smaller as we approach a minimum. Such algorithms incorporating learning rate schedulers (and of course performing the gradient descent) are readily available, the one we will use throughout this thesis is ADAM [33].

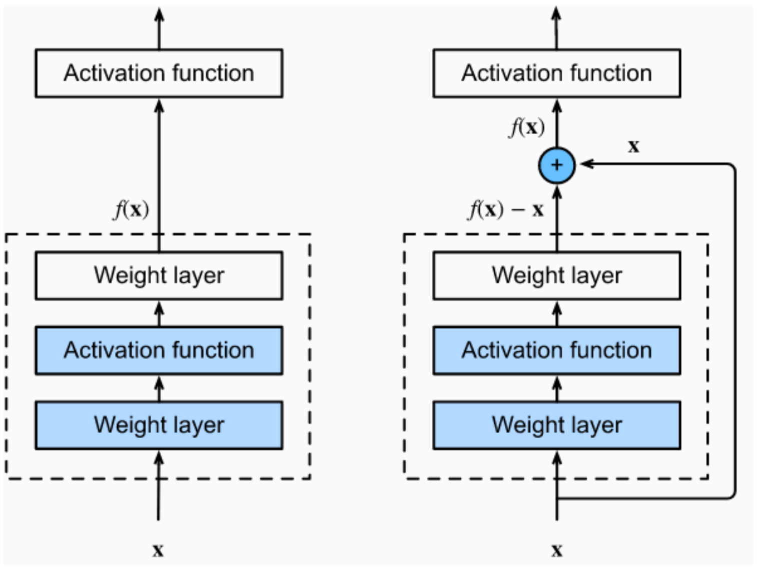

Figure 3.3: Right panel: Conventional layer block. The mapping $f(x)$ has to be learned by the weights. Right panel: Residual block using a skip connection. The dotted block only has to learn the deviation from the identity, moreover the identity mapping is easier to learn by pushing the weights towards zero. Figure taken from Ref. [30]

3.4 Residual Connections

Choosing the right network architecture is difficult. If the network is too deep training might take an unreasonable amount of time and might even lead to overfitting, that is the model picks up features present in the training set but not present in the underlying data distribution. Such a model will show excellent results during training but generalize very poorly as seen during evaluation. On the other hand, if the model is undercomplex it will lead to underfitting, meaning that the model does not have the necessary complexity to learn all relevant features and thus will not be as performant. Thus it would be ideal to have a guideline by which increasing model complexity will lead to a more expressive rather than just a different architecture. This is the rationale behind residual connections [30].
Let G be the function space accessible to a given model and g* the ideal mapping we desire to be learned. If g* is not in G then the best the model can do is find an optimal approximation $g_G^* \in G$. If we now increase the model complexity, usually by adding additional layers, it is a priori not clear that the model can perform better since the new function space G' will in general not be a subset of G. Residual connections prevent this from happening by guaranteeing $G \subset G'$ or in other words the identity mapping $g(x) = x$ is part of the function space induced by every additional layer.
Technically a residual connection is a block of layers (dense, convolutional, pooling, activation functions) followed by a skip connection, so let x be the input flowing into the block and f(x) the output without the skip connection. Then the total output is $f'(x) = f(x) + x$. This means that the residual block does not have to learn the desired mapping $f^*(x)$ but only the deviation from the identity and in fact the identity is a natural part of the function space so that residual connections truly increase the network's expressiveness. This is illustrated in Figure 3.3, the left panel shows a standard block of layers while on the right panel a skip connection is added. Historically the introduction of residual blocks has vastly improved deep learning and models like ResNet [34] achieved new performance heights.

4 Super Resolution

4.1 Generative Adversarial Networks

Generative models try to learn the underlying distribution of the training data and produce new output that is indistinguishable from the given inputs. Among them the class of GENERATIVE ADVERSARIAL NETWORKS (GANs) [35] are popular and powerful and the model structure used in our study. A GAN consists of two sub-networks, a generator and a discriminator that compete against each other, that is the generator produces fake simple, trying to fool the discriminator while the discriminator attempts to tell apart fake from truth samples. Ideally we manage through training that the discriminator cannot tell apart fake from truth input at all. Thus generator and discriminator compete in a minmax game.
At the heart of GANs is the adversarial loss

$$L_{adv} = \langle \log(D(x)) \rangle_x + \langle \log(1 - D(G(z))) \rangle_z \tag{4.1}$$

here D(x) is the discriminator's estimate of the probability that real data x is real, G(z) is the generator's output given noise z, D(G(z)) is the discriminator's estimate of the probability that a fake sample is real and $\langle \dots \rangle$ is the expectation value over probability distribution X or Z, in practice these will be approximated by sample means over each batch.
One major issue of GANs is that it can be difficult to find a reliable performance metric since, unlike for example in classification tasks, it is not straightforward to compare a given generator output to a desired label y. As an illustration we show an image with a checkerboard pattern in Figure 4.1, such repeating patterns often arise in generated images and one has to employ sophisticated methods to prevent them. For our physics application there are natural performance checks that we will describe in the following sections.

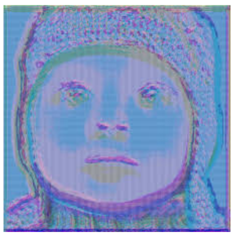
Figure 4.1: Example of a generated image suffering from checkerboard patterns. Figure taken from Ref. [36]

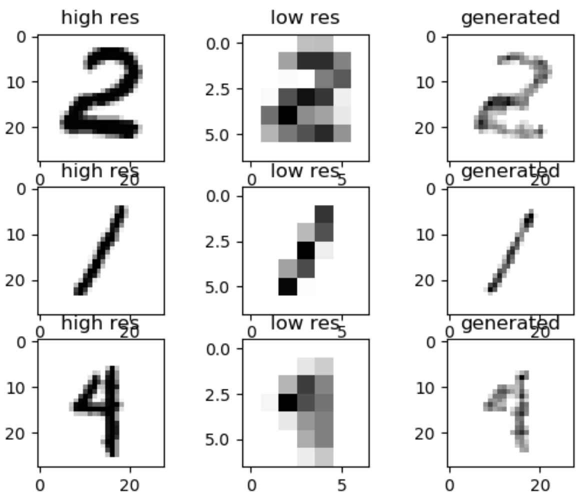

Figure 4.2: Toy example of generated images based on the Mnist dataset.

4.2 Single Image Super Resolution

Super Resolution [37] is a generative task where the model's input is a lower resolution (LR) image of a high resolution (HR) ground truth. The generator produces super resolved (SR) images that shall mimic the HR ground truth through an upsampling routine. Such a task has the inherent difficulty that usually there is not one but many acceptable SR outputs. Considering this it is questionable if a model can reliably produce good results on a image by image case or if it can only reproduce the HR space in a statistical sense, i.e. averaged over many images. However it also makes clear why a GAN setup is advisable as compared to a non adversarial upsampling procedure: Without the discriminator the upsampling routine had no guidance how to distribute the information present in the LR image in the larger SR image, the discriminator will enforce similarity to the HR image. A even more fundamental question is if the network can output more information in the SR image than the LR image to begin with. This again is ensured through the adversarial process since in the physics task we consider, to be described in the next section, the HR image will implicitly encode fundamental physical properties that the network can pick up on. In Figure 4.2 we show a toy usage of the GAN we will use in this thesis, upsampling of handwritten digits from the Mnist dataset [38]. The left panels show the ground truth, the middle panels the by a factor 4 downsampled low resolution input and the right panels the generated images. We trained the network for just a few hundred batches and already see a nice performance. The Mnist dataset is historically important but not suitable for performance benchmarks of modern deep networks.

4.3 Jet Image Dataset

Our task is to apply super resolution to Jet Images. Physically this is a very interesting challenge: Jets are part of virtually any process at the LHC [39] so understanding their substructure is crucial for most analyses. They provide a mean of stringently testing QCD and even in the search for new physics many background processes will produce substantial number of jets [40]. Given that beyond standard model signals are expected to have small rates, understanding and reducing QCD backgrounds is of immense importance. Experimentally the task of upsampling jet images is mandated by the fact that jets are seen in detectors as calorimeter entries and as described in section 2 the resolution in the forward region is much worse compared to the central region.
Since we are interested in probing the general capacity of neural networks to upsample jets we consider in this thesis only simple processes, namely $t\bar{t}$ and QCD dijet events. For our super resolving task we have to provide the network with paired HR and LR images. The data generation pipeline is as follows:

1. We use PYTHIA 8.240 [41] to create $t\bar{t}$ events. The tops are forced to decay to W and a bottom quark with the W subsequently decaying through a hadronic channel. Each top is required to have a transverse momentum p_T of at least 600 GeV. Analogously QCD events are created with the hard process now resulting in two quarks.

2. We pass these events to DELPHES 3.4.1 [42], using the standard ATLAS card with the sole modification of setting the calorimeter cell size to $2/160 = 0.0125$ for the whole calorimeter. This corresponds to 160 cells over a range $[-1, 1] \times [-1, 1]$ in the $\eta - \phi$ plane, the physical range captured in our images.

3. The resulting output is processed in the following manner on an event-by-event basis: First every calorimeter hit gets registered to a 2d histogram with a binsize of 0.0125. Then we apply sumpooling to this histogram as described at the end of section 3.2.1 to create a low resolution histogram. Now FASTJET [43] is ran on each set of entries (HR and LR) separately, using the anti-kt algorithm with a conesize of R = 0.8. Finally, from the reconstructed jets, we select the one with the highest p_T and apply a filter that lets it pass only if the jet p_T is in the range 550 GeV to 650 GeV and the absolute value of η is smaller than 2.0, in addition to that we compute the ΔR difference between the selected HR and LR jet to ensure the LR jet corresponds to the HR one, allowing a maximum discrepancy of 0.1,lastly we only accept jets with at least 15 constituents. We apply the same matching requirement to create LR image samples with downscaling factors of 2, 4, and 8, removing events that fail the requirement for any particular resolution from all samples. This way all jet samples are composed of the same events.

In this way we create image data in the $\eta - \phi$ plane with the transverse momentum p_T as entries. The images are further preprocessed by centering around the jet, that

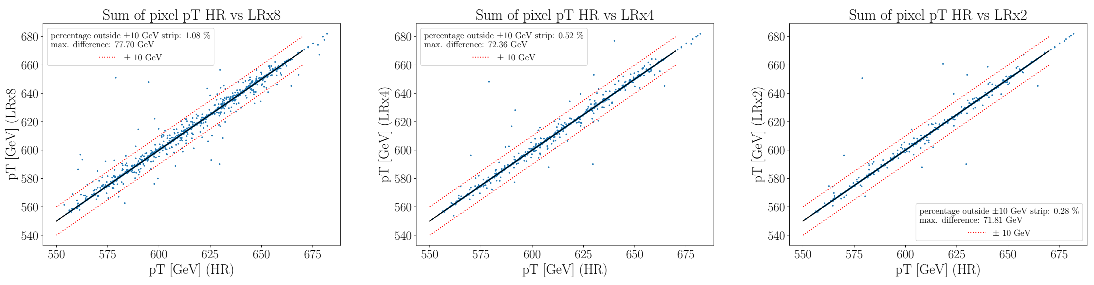

Figure 4.3: Jet image checks for multi factor top sample. Data points are blue dots, no mismatch occurs on the blue solid line. The dotted red lines correspond to a mismatch of ±10 GeV between HR and LR images.

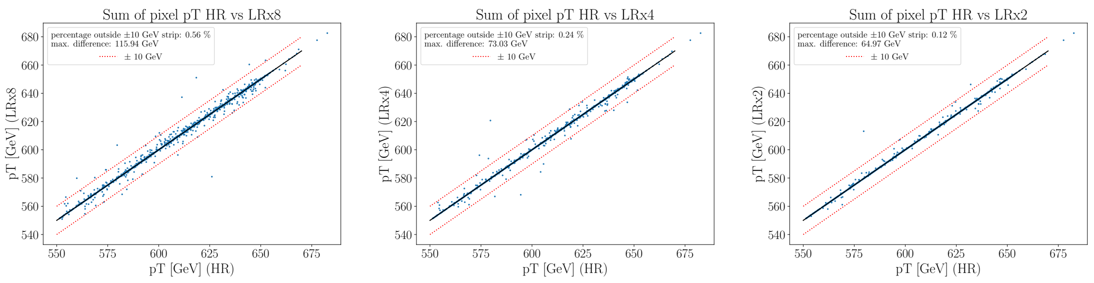

Figure 4.4: Jet image checks for multi factor qcd sample. Data points are blue dots, no mismatch occurs on the blue solid line. The dotted red lines correspond to a mismatch of ±10 GeV between HR and LR images.

is we shift the image so that the hardest pixel sits in the center.
In the third step it was crucial that the jet clustering is performed on the HR and LR entries separately because this prevents an information backdoor which means that no additional information present in the HR image can leak into the LR images. This comes at the price of an imperfect matching procedure so that in some cases the images are wrongly paired together. In practice this is not an issue as can be seen in Figure 4.3: Here we plot the HR p_T against the LR p_T, defined to be the sum over all pixels for the different downsampling factors of top jets. The individual entries are displayed as scattered dots, the solid blue line is the ideal bisecting line which means there would be no p_T difference. The dotted red lines represent a p_T mismatch of ±10 GeV. We see that only about 1% of events have a p_T mismatch larger than 10 GeV. Since our sample contains about 600.000 events we could in principal even remove these events from the dataset but given the sub-percent occurrence this was not done. Figure 4.4 shows that equivalent conclusions hold for the qcd sample. Figure 4.5 shows an example top jet image with the corresponding LR Image.

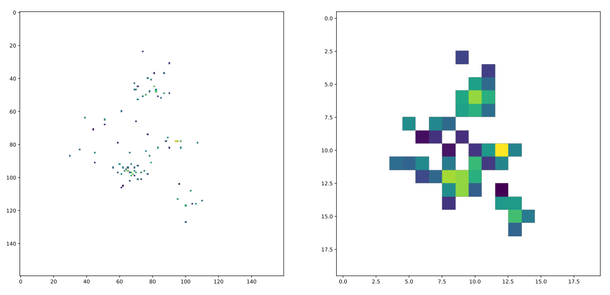

Figure 4.5: Example Top Jet Image and corresponding LR Image.

4.4 GAN Architecture

The starting point of our model is an state of the art architecture for image super resolution, the Enhanced Super Resolution GAN [44; 45; 46]. In contrast to regular images our jet images are extremely sparse, with of the order of 50 non-zero pixels. This makes it mandatory that we alter the basic architecture. In the following we describe the network architecture we ultimately employed.

4.4.1 Generator

The generator is the centerpiece of our network, correspondingly it is quite intricate. It is schematically displayed in the top part of Figure 4.6. Technically it is a fully convolutional residual network encapsulated through nested blocks. The basic block is the dense residual block (DRB): It is built out of consecutive convolutional layers with (3×3)-kernels, stride 1, padding 1, and 64 filters. The activation function is a leaky ReLU with $\alpha = 0.2$. These parameters for a convolutional layer have the property that they keep the spatial dimension fixed. This is insofar important as it allows us to easily use input images of different dimensions. Three dense residual blocks form a residual-in-residual dense block (RRDB), connected via residual connections. The hyperparameter β, indicated in the Figure is an alteration of the usual skip connection that allows us to relatively scale the importance of the block

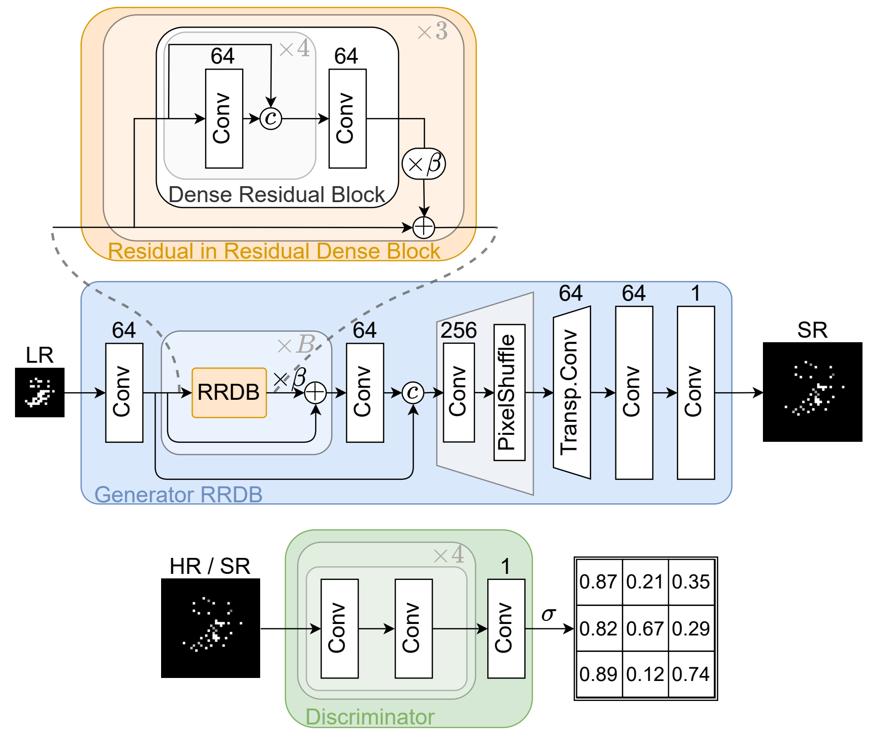

Figure 4.6: Modified generator from ESRGAN (upper) and discriminator modified from the SRGAN (lower). Figure by Lukas Blecher [8]

output $f(x)$ compared to the addition of the input x. The upsampling is performed in steps of factors f = 2 i.e. every upsampling step doubles the image dimension. In our application we will use at max 3 upsampling steps, corresponding to a maximal upsampling factor of f = 8. Our network can use two different algorithms for upsampling: pixel shuffle [47] and transposed convolution, each perform a factor 2 upsampling and in the case of several upsampling steps (corresponding to total factor 4 or 8 upsampling) we can freely choose which combinations to use. A study of the most performant upsample setup will be shown later. The default is to only use pixel shuffle, where upsampling is performed according to the formula

$$(C \times 4, H_{in}, W_{in}) \longrightarrow (C, H_{in} \times 2, W_{in} \times 2)$$

Here C is the number of channels and H, W the image height and width. An example of a transposed convolution is shown in Figure 4.7. In contrast to a standard convolutional operation the input window is now slid along the kernel, in a sense exchanging the role of the two.
The pixel shuffle has the advantage of encoding the full information from the feature maps. It simply redistributes the information by transforming a large number of channels, as usually arise after deep convolutions, into a set of feature maps with fewer channels but larger spatial dimensions. The transposed convolution takes into account local information though a trainable kernel. After learning meaningful

weights it can can help learning intricate, non-local patterns, which would be missed by a global pixel shuffle [48].

4.4.2 Discriminator

The discriminator network is a convolutional network with leaky ReLU activations, shown in the bottom half of Figure 4.6. It uses blocks consisting of two convolutional layers with a (3×3)-kernel and padding 1, therefore also retaining spatial dimension but followed by a second layer halving the feature map dimensions through a strided convolution. We link four of those blocks and start with 64 filters, doubling the number of filters after each block. We cut off the network before the flattening and feed it into a fully connected layer, thereby switching to a Markovian discriminator. Effectively this means that instead of outputting a single number, our discriminator output is a matrix where each entry corresponds to a different patch in the image. Finally, we include the option of employing a second discriminator with exactly the same structure. In case this second discriminator is activated the full discriminator response is the sum of two discriminator networks. For the second discriminator we reset all weights after a fixed number of batches.
The original ESRGAN uses batch normalization layers to improve training, however we use a gradient penalty (see section 4.4.3) for stabilization and therefore remove batch normalization as pointed out in [49].

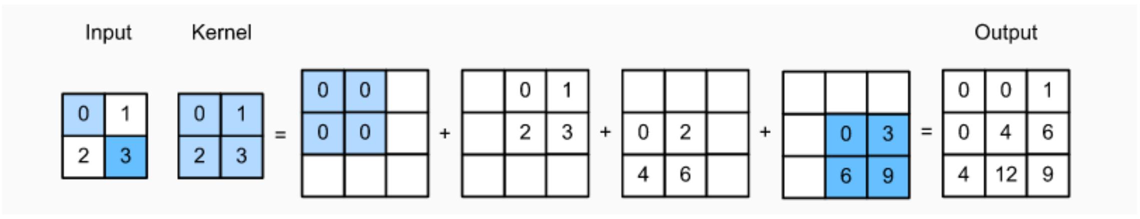

Figure 4.7: Transposed Convolution with 2x2-kernel and stride 1. Figure taken from Ref. [30].

4.4.3 Loss Function

Compared to a vanilla GAN the ESRGAN makes use of more complicated loss functions. In addition, having the task of super resolving sparse images in mind, we added some losses.

Generator Loss As a minimal loss we employ a L_1 loss calculated between the HR and SR image. This simply compares the image on a pixel-by-pixel basis. We prefer L_1 over L_2 since the latter would be especially sensitive to large differences in a given pixel but given the sparsity of our jet images we do not expect nor want the SR image to actually be the same as the HR image.

$$L_{\text{HR}} = L_1\left(\text{SR, HR}\right) \tag{4.2}$$

As a consistency check we can analogously measure how similar the LR image is to the downsampled SR image which we will denote as LR$_{gen}$, we want to point out that this image is not a generator output but rather just the sumpooled version of the SR image. Thus we add an LR loss

$$L_{\text{LR}} = L_1\left(\text{LR}_{gen}, \text{SR}\right) \ . \tag{4.3}$$

The adversarial loss is changed compared to Eq. 4.1, instead we use a loss proposed in RELATIVISTIC AVERAGE GAN models [46]:

$$\begin{aligned} L_{\text{adv}} = -&\langle \log D\rangle_G - \langle \log(1-D)\rangle_T \\ \text{with} \quad D_T =&\sigma\left(C_T - \langle C\rangle_G\right) \\ D_G =&\sigma\left(C_G - \langle C\rangle_T\right) \end{aligned} \tag{4.4}$$

where σ is a sigmoid classifier function and C is the unactivated discriminator output. G and T refer to the generated and truth distribution, respectively. Compared to a standard GAN loss adversarial loss we have an additional term because D_T depends on the generated data G. Compared to standard adversarial losses, here the truth and generated samples both enter in D_G or D_T, intuitively a standard adversarial loss only considers how real the generated samples appear but the relativistic average adversarial loss compares how generated samples appear more realistic than real samples on average and the other way round.

When we up-sample the LR-jet image by a factor f, we need to distribute each LR pixel energy over $f \times f$ SR pixels. These $f \times f$ pixels define a patch, and we encourage the network to spread the LR pixel energy such that the number of active pixels corresponds to the HR truth. This defines the additional loss term

$$L_{\text{patch}} = L_2\left(\text{patch(SR), patch(HR)}\right) \tag{4.5}$$

It is important to notice that in this loss we do not consider the actual value of the given pixel but simply add up the number of non-zero pixels.

The combined generator loss is then

$$L_G = \left(\lambda_{\text{HR}}\, L_{\text{HR}} + \lambda_{\text{LR}}\, L_{\text{LR}} + \lambda_{\text{adv}}\, L_{\text{adv}} + \lambda_{\text{patch}}\, L_{\text{patch}}\right) \ , \tag{4.6}$$

where $\lambda_{\ldots}$ are the respective hyperparameters that regulate the relative importance of the loss terms.

Discriminator Loss The discriminator loss corresponds to the adversarial loss of the generator but with switched labels,

$$L_{D'} = -\langle \log(1 - D)\rangle_G - \langle \log D\rangle_T \ . \tag{4.7}$$

this naturally incorporates the different tasks: While the generator tries to make the generated samples look real, the discriminator attempts to mitigate such a conclusion.
In an adversarial setting it is important that both the generator and discriminator perform on comparable levels: If the generator becomes too strong then the discriminator will be unable to provide relevant feedback and thus the generator may hit a plateau before it is actually optimized. Conversely, if the discriminator is too strong then the generator also will stop improving because this time the discriminator feedback will be too definite in the sense of not being swayed by little changes in the generator weights but such a small step improvement was the basis of our gradient descent algorithm. Since upsampling sparse images is difficult we expect the generator to perform bad at the beginning stages of training, thus we want to regularize the discriminator by adding a regularization term to the loss. Simply making the discriminator weaker is not a good option because then we might run in the first issue of a too weak discriminator at later stages. A suitable regularization is gradient penalty [49]:

$$\mathrm{L}_{reg} = \langle (\|\nabla_{X'} C(X')\|_2 - 1)^2 \rangle. \tag{4.8}$$

Here X' is a randomly weighted average between a real and generated sample:

$$X' = \epsilon X_T + (1 - \epsilon) X_G, \tag{4.9}$$

and C(X') is the unactivated discriminator output. The gradient penalty penalizes discriminator outputs that are close to 0 or 1, or in other words makes it less attractive for the discriminator to output very definite predictions.

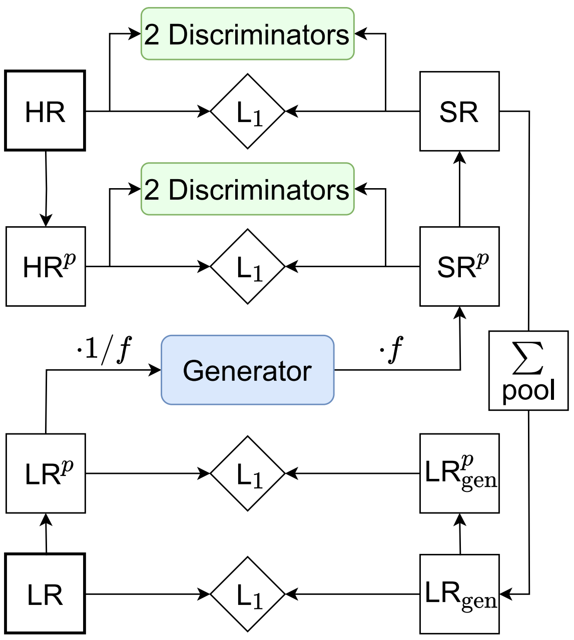

Figure 4.8: Training process for jet images. The generator and discriminators are illustrated in Fig. 4.6. Figure by Lukas Blecher [8]

4.5 Training and Evaluation

Our training procedure can be seen in Figure 4.8. To begin with we have the paired HR and LR ground truth images. Both of them are pixel-wise raised to a power p. Further the LR image is divided by a factor 1/f where f is the upsampling factor and then passed through the generator. The output is the SR image raised to a power of p. We undo the scaling by multiplying with k as well as taking the p$^{\text{th}}$ root to get to the SR image, however we retain the power raised SR image for further loss computations. Through sumpooling the standard and power SR image with a factor f we finally get the LR$_{gen}$ and LR$^p_{gen}$ images, so that we have a set of four images (HR, LR, SR, LR$_{gen}$) twice: Once with a standard power of 1 and once with a power p. These are then used to compute the various loss functions described in the previous section. It is important to note that both sets have their own discriminator, this must not be confused with the aforementioned maximum setup involving a second set of discriminators. In such a maximal setup we will have in total 4 discriminators, of which two are reset at fixed intervals. The total generator loss is therefore a sum over standard and power losses of Eq. 4.6

$$L_G = \sum_{s\in\{\text{std},\text{pow}\}} \lambda_s \left(\lambda_{\text{HR}}\, L_{\text{HR}} + \lambda_{\text{LR}}\, L_{\text{LR}} + \lambda_{\text{adv}}\, L_{\text{adv}} + \lambda_{\text{patch}}\, L_{\text{patch}}\right) . \tag{4.10}$$

The reason for this power setup is our choice of physical process: As previously explained the jet evolution starts with a set of partons created in the hard process.

Due to our high p_T requirement these will undergo a few splittings with the resulting daughter partons still carrying high p_T and therefore they will show up in our images as the hardest pixels. As we enter the showering stage the soft qcd-mediated splittings will lead to a number of low p_T partons corresponding to soft entries in the image. For the machine learning task this means that the p_T distributions will be sharply peaked near zero but still covering a wide range with a few entries close to 500 GeV. Raising the distribution to a power p will widen the distribution and help training. This is exemplified for some powers in Figure 4.9, there and in the rest of this thesis we will use transverse momentum and energy to mean the same. We can even give a purely physical argument for why the dual setup is thought to perform better: The images are dominated by the few (usually up to 5) hardest pixels. Thus the network can increase its performance best by focusing on these hard pixels, neglecting the soft noise. By broadening the distribution we mitigate the dominance of the hard pixel and therefore it is beneficial for the network to learn to get also softer pixels right.

Evaluation As previously mentioned it can be difficult to find good performance metrics for generative models, the lowest loss does not always correspond to a satisfying generated image. For standard images like faces powerful metrics like peak signal-to-noise ratio exist but these are not applicable for our sparse jet images. To circumvent this we make use of *early stopping* [50] and physical variables. Early stopping means that we evaluate our model on a validation set after a fixed number of batches and save the model's weights whenever it achieves a better evaluation score than the previously saved weights. Evaluation is based on the distribution of n$^{\text{th}}$-hardest pixel distributions compared between the HR and SR datasets, the patch image described in the previous section and a mean image, taken to be a 2d histogram of the sum of activity in each pixel. Finally we test how well our models generalize by evaluating it on an independent evaluation set. Such an evaluation is shown for a model upsampling by a factor of 8 in Figure 4.10.

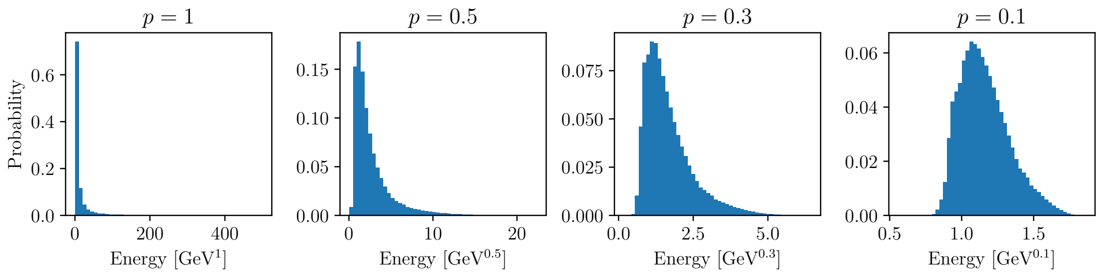

Figure 4.9: p_T distributions raised to different powers of p. We clearly see that the distributions get broader with smaller power values.

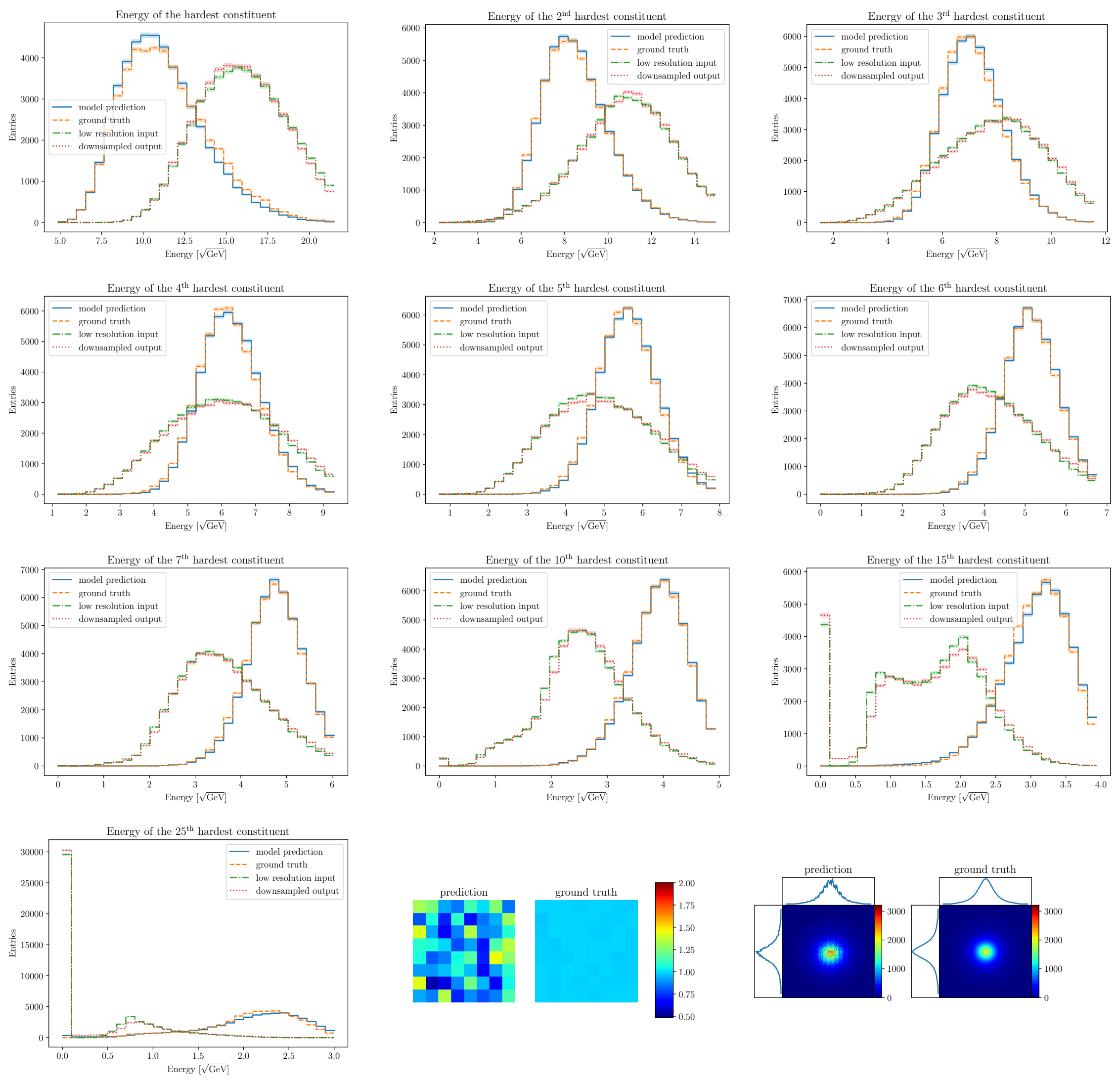

Figure 4.10: Example evaluation of a factor 8 upsampling model. In addition to the ordered pixels (with a square-root scaling), we show the patch image and mean histogram. Each entry of the patch image has been divided by the ground truth mean value. The left and top panel of the 2d histogram show the projected 1d distributions. The zero-bin in energy collects jets with too few entries.

5 Baseline Model

In the analysis performed in this chapter we will make use of a simplified dataset: It contains only top jets and the HR and LR images are not uncorrelated, instead the LR image is simply obtained by performing sumpooling. This can lead to an information backdoor but since the goal is to establish a well performing baseline model for upsampling factors f of 2, 4 and 8 this is acceptable. This also mandates the usage of a minimal architecture: We only use the L_{HR}, L_{LR} and L_{adv} losses, no second discriminator set and all upsampling layers are pixel shuffle. The corresponding hyperparameters of our baseline are listed in Table 5.1. We will present the performance of this model for the different upsampling factors, discuss its shortcomings and based on that perform a careful optimization in the next chapter. There we will also take up the matter of qcd jets.
Throughout we will use the Adam optimizer with a learning rate $\eta = 0.0001$ and momentum $\beta_1 = 0.5$, $\beta_2 = 0.9$ [51]. Our models train for approximately 100.000 batches, the evaluation set contains 50.000 images for good statistics. Our HR images have 40×40, 80×80 and 160×160 pixels for the respective factors 2, 4 and 8.

#RRDB	batch size	β	rescaling 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	0.01

Table 5.1: Hyperparameters for our baseline model; β is the residual scale factor defined in Fig. 4.6.

5.1 2x Upsampling

It is natural to begin with the minimal upsampling factor 2. The evaluation results are depicted in Figure 5.1. We clearly see that the model is capable of learning all n$^{\text{th}}$ hardest pixel distributions very well. It is remarkable that the distributions for the 10th, 20th and 30th hardest pixel are approximated as well as the one's for the 4 hardest pixels. This suggests that the model has picked up on the pattern of soft qcd-mediated splittings. It further is noteworthy that the biggest difference between HR and SR distribution, albeit by a small margin, appears for the 5th hardest pixel. This however is explainable in terms of physics because the region of 5th to 10th hardest pixel is neither entirely in the hard process regime nor the soft noise regime, therefore it is the most difficult to learn. Later we will see this effect more pronounced. In the bottom panel of Figure 5.1 we see that the model also manages to populate the right number of pixels in the SR space as well as assign them the correct energy, at least on an average level. Together this leads to the conclusion that the total energy across the image is conserved during the generative process. What is not perfect is the distribution of the energy of a single LR pixel into the corresponding 2×2 SR pixels. This is captured in the bottom-left panel, the patch image. We want to reiterate again that in a way all information about the super resolving performance is implicitly present in the patch image: Given the LR image, the only real task of the network is to distribute the energy of all LR pixels in the right way across the 2×2 corresponding pixels in the SR image and obviously this distributions will be uniform for the HR images, as it is. The model however has a tendency to put too much energy into the bottom right pixel, although for the case of $2\times$ upsampling the relative imbalance as can be read off from the colorbar is of the order of 50000/47000.

5.2 4x Upsampling

The case of $4\times$ upsampling is shown in Figure 5.2. Similar conclusions hold as in the factor 2 case. Again the soft noise regime is equally well approximated by the super resolved image as the hard regime. The most significant difference arises in the bottom panel: The model now fills too many pixels compared to the HR ground truth (bottom left) but on average puts less energy into those pixels (bottom center). This however again suggests that the total energy of the image is preserved. In the patch image we can now clearly see that the distribution of energy is not uniform for the SR image, some pixels are overpopulated while others are underpopulated. Still, the ordered pixel distributions are so well described that we will move on to the more challenging factor 8 upsampling and attempt further improvements for this case.

5.3 8x Upsampling

A first obvious attempt to improve the patch image is to activate the corresponding loss L_{patch}. We found a value for the corresponding hyperparameter λ_{patch} of 0.1 to work best, so the updated list of hyperparameters is captured in Table 5.2.

#RRDB	batch size	β	rescaling p	λ_{reg}	λ_{std}	λ_{pow}	λ_{HR}	λ_{LR}	λ_{adv}	λ_{patch}
15	15	0.1	0.3	0.001	0.2	1	1	0.1	0.01	0.1

Table 5.2: Hyperparameters for our baseline model including the patch loss.

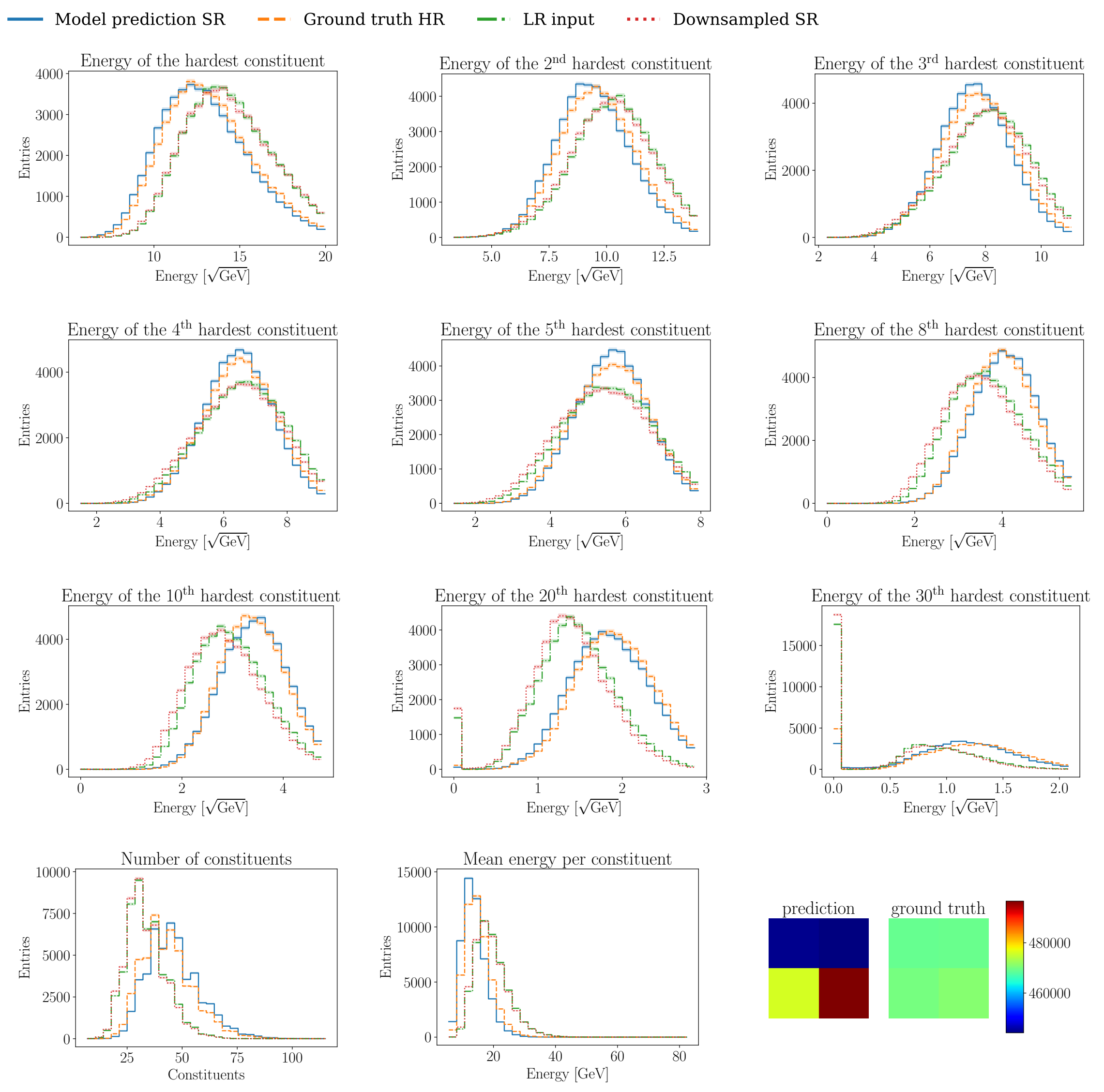

Figure 5.1: Evaluation of the baseline model with 2x upsampling. In addition to the ordered pixels (with a square-root scaling), we show the patch image, the number of non-empty pixels and the average energy per pixel. The zero-bin in energy collects jets with too few entries. Legend by Lukas Blecher [8]

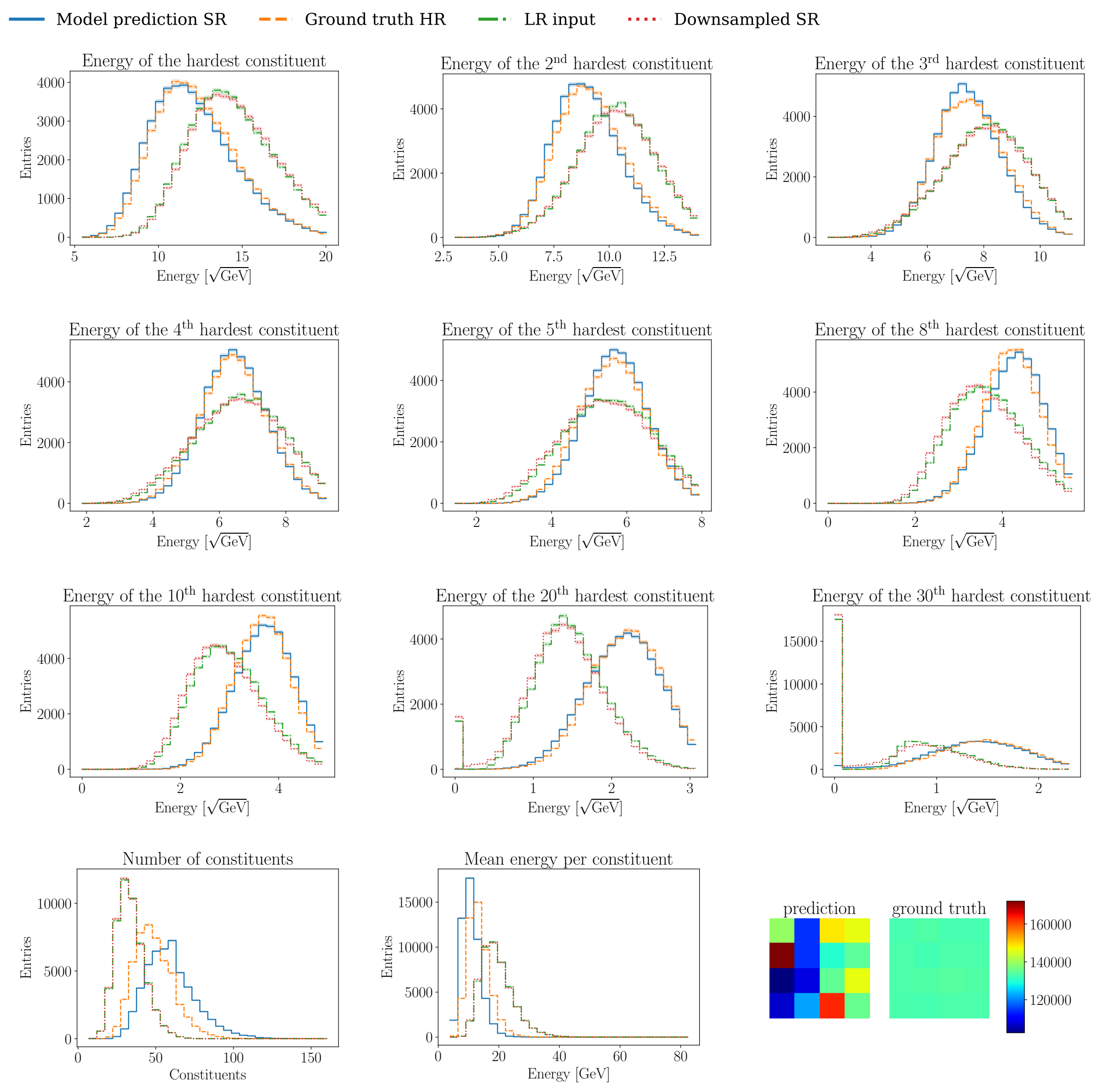

Figure 5.2: Evaluation of the baseline model with 4x upsampling. In addition to the ordered pixels (with a square-root scaling), we show the patch image, the number of non-empty pixels and the average energy per pixel. The zero-bin in energy collects jets with too few entries. Legend by Lukas Blecher [8]

The model evaluation based on these hyperparameters is shown in Figure 5.3. Compared to the 2x and 4x case we now see clear deviations of the prediction compared to the HR ground truth. The SR distributions are systematically shifted to the right, i.e. the SR pixels are given too much energy. As before, the patch image features on overactive pixel, meaning that the model still prefers to distribute most of the energy of a LR pixel into a single SR pixel so we have to conclude that inclusion of the patch loss L_{patch} alone cannot fix this model behavior. This is also seen in the mean image (bottom-right) featuring characteristic repeating patterns.

At this point two comments are in order: First, we can see that in all cases the relation of the LR image to the $\mathrm{LR}_{\mathrm{gen}}$ is the same as the relation between the HR and SR image. This comes as no surprise because so far the LR image is just the sumpooled version of the HR image. For tops, with their innate three-prong decay structure, the active pixels cover a relative big area of the available image space, as can be seen in the relative broad distribution of the mean image. This means that sumpooling a subwindow containing a given n^{th} hardest pixel will usually not include other hard pixels so that the n^{th} hardest HR pixel will be pooled to the n^{th} hardest LR pixel. Secondly we already pointed out the importance of the patch image. We might therefore ask if our early stopping evaluation, dominated by the orderd pixel distributions, really manages to capture the point of best model performance. This suggests to change up the early stopping in such a way that the performance during training is solely given by the patch image. Such a modification was performed for the baseline model and the resulting final evaluation is shown in Figure 5.4. The result is somewhat inconclusive: While the shift in the SR pixel distributions seems to have been attenuated this might just be an effect of initial seeding. We will investigate this briefly in the next section. The patch and mean image in Figure 5.4 cannot be said to having improved compared to Figure 5.3 but it is still impressive that model performance has not deteriorated which corroborates our statement that most of the relevant information is captured in the patch image.

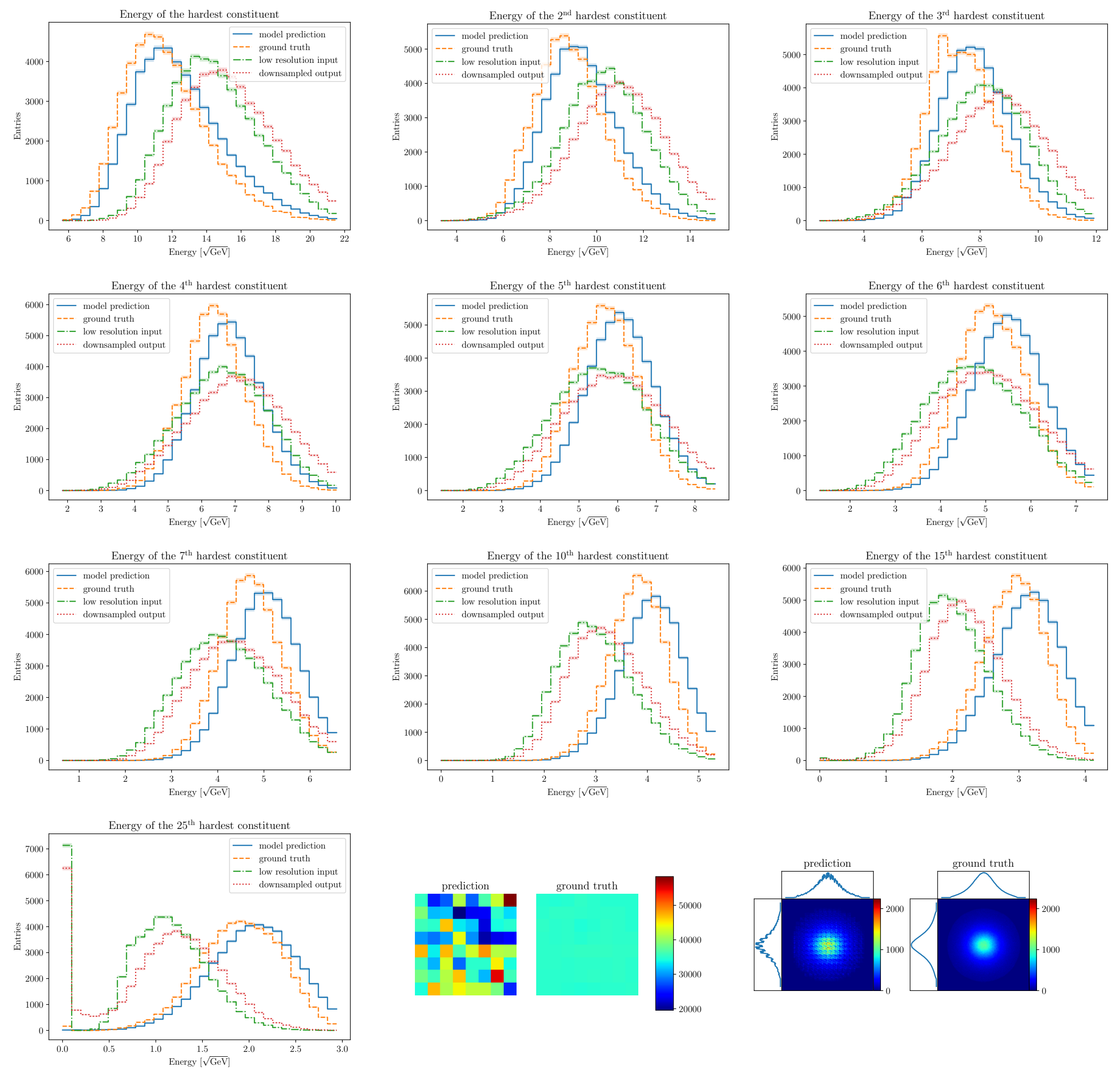

Figure 5.3: Evaluation of the baseline model with 8x upsampling. In addition to the ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

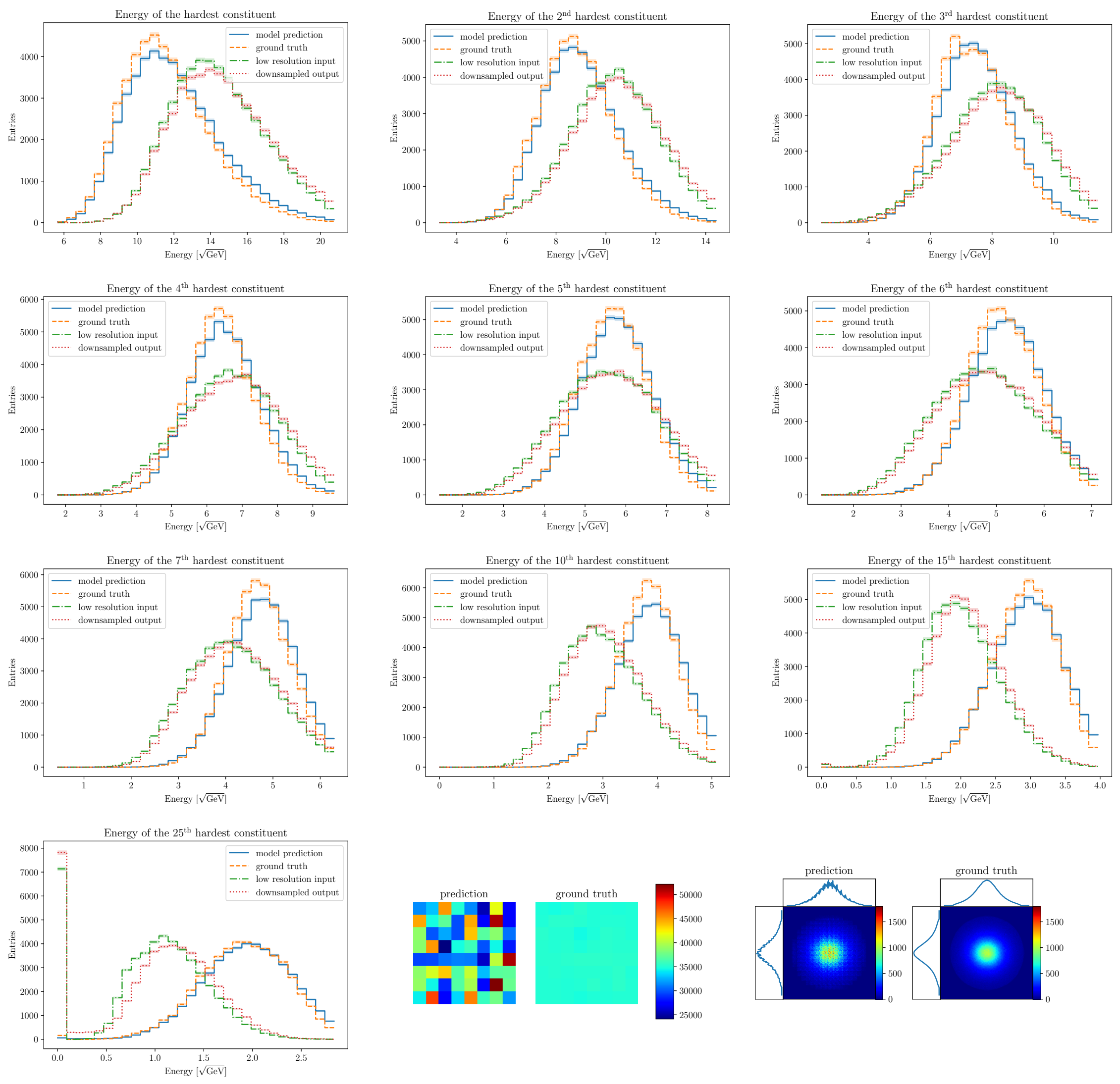

Figure 5.4: Evaluation of the baseline model with 8x upsampling where the best model point during training was solely determined by the patch image. In addition to the ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

6 Top Optimization

Previously we saw that our baseline model can approximate the ordered pixel distributions arguably well but struggles to distribute the energy of a single LR pixel in the SR image space. This leads to repeating patterns in the mean image and makes it obvious that our super resolved images are so far imperfect. In this section we will try to improve on this. This means it is also time to drop the simplified dataset and use LR jets that do not benefit from an information backdoor according to the pipeline outlined in section 4.3. As a first step it is mandatory to check how this affects model performance. Thus we have retrained our baseline model with the same hyperparameters as described in table 5.2, again using only the patch image in the early stopping evaluation, on the new dataset. The evaluation results are displayed in Figure 6.1 and are very appealing: Compared to Figure 5.4 model performance is at least equal. This suggests that either there has not been any information backdoor to begin with or the model never made use of it.
As outlined our task is now to remove the artifacts in the mean image or correspondingly lead the network to distribute energy more uniformly in the patch image. Before we change up our model we will investigate two related questions: How much artifacts does our network implement into a generated image irrespective of the given LR image and how do the results depend on the seeding of the network. The first question will qualitatively and briefly be touched on by performing the *blank image test*, the latter by training and averaging several networks with the same hyperparameters.

Blank Image Test The blank image test is a very basic sanity check often performed with generative models. One simply feeds a blank image into the network. Ideally the resulting generated image is also blank. The outcome for our baseline model can be seen in the top panel of Figure 6.2. The SR image is indeed blank so our model passes the test. As a further cross-check we added 150 pixels with random gaussian noise to the blank image in the bottom left panel of Figure 6.2 and fed this also through the network, the result being shown in the bottom right panel. There we see that the SR image does not correspond to the HR image but the difference is reasonable. Together these tests give us confidence that the appearance of artifacts is not inherently ingrained into the model.

Initial Seeding Machine learning algorithms often make use of random numbers but since these are produced by pseudo-random number generators [52] the entire 'randomness' is captured into the seeding, meaning that given the same initial seed, the sequence of random numbers produced will also be the same. We have seen that

the super resolved patch image tends to feature one very active pixel in which most of the energy is distributed but the location of this pixel within the $f \times f$ window changes. On one hand this is simply a reflection of symmetry: We know that there is no preferred direction in the image so this is no surprise but it also tells us that we can assume that the effect will be mitigated when we average over sufficiently many initial seeds. We trained 10 separate iterations of the baseline model for approximately 30.000 batches. In Figure 6.3 we plot the root mean square per pixel over the 10 resulting patch images together with the ground truth for comparison. We also include example patch images of two instances in Figure 6.4. The colorbar has been normalized to 2x the ground truth mean value at the upper end and 0.5x of it at the lower end. Further, the numbers displayed in each pixel show the actual count divided by the ground truth mean i.e. how hard the pixel is relative to the mean value.
We see that the averaged patch image is much more uniform compared to the results in the previous section. As expected the position of the very hard pixel is essentially random and thus washed out in an average. However from Figure 6.4 we also see that single instance performance can drastically vary, with the left panel instance almost comparable to the averaged image and the right panel featuring multiple overpopulated pixels. What we can learn from this is that the influence of the seed seems to be much larger than the concrete choice of hyperparameters, suggesting the conclusion that another parameter search will most likely not lead to stable performance improvements. Instead we are lead into the direction of changing up our model architecture.

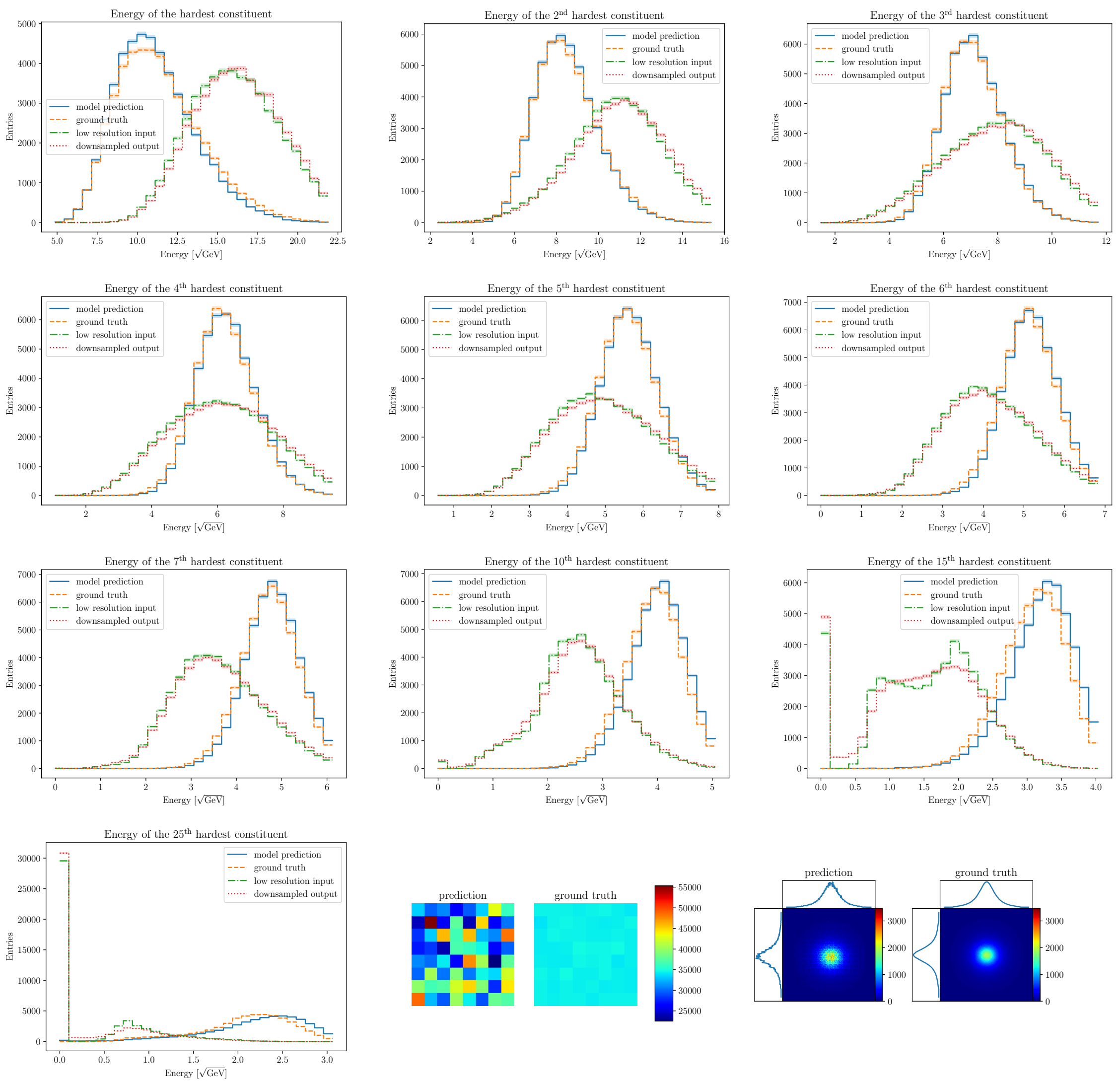

Figure 6.1: Evaluation of the baseline model with 8x upsampling where the best model point during training was solely determined by the patch image. Model trained on a dataset without information backdoor. In addition to the ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

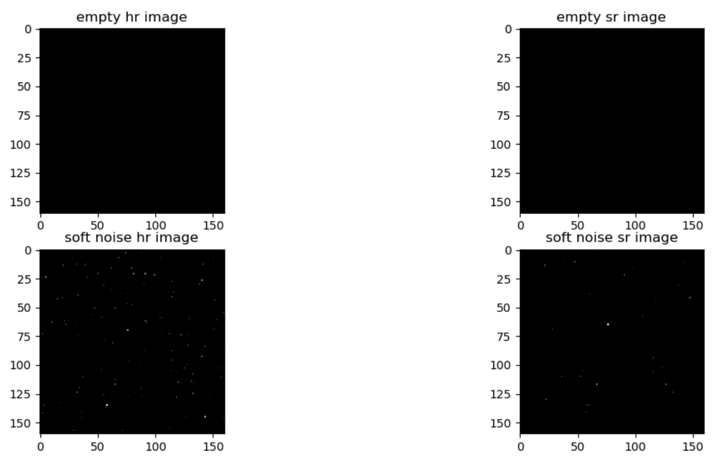

Figure 6.2: Upsampling of an empty image (top), soft noise as cross-check (bottom).

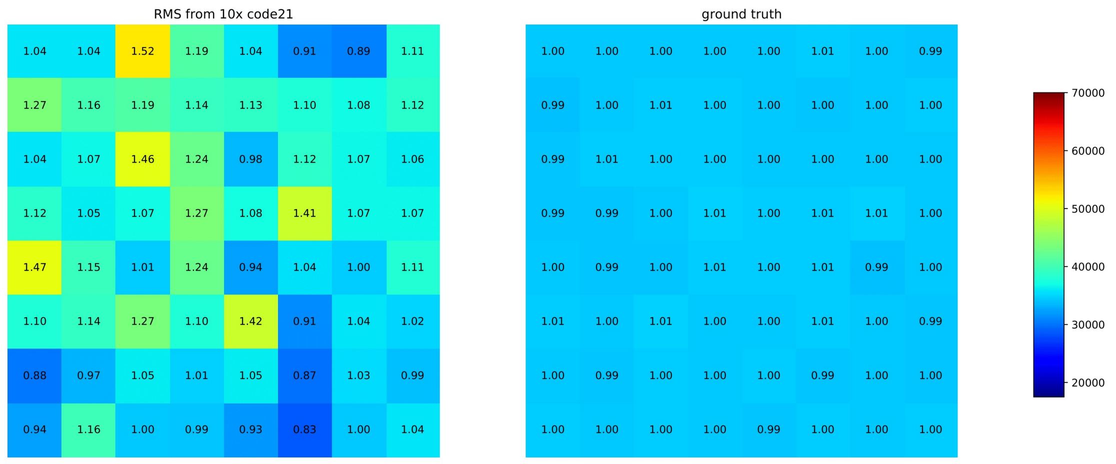

Figure 6.3: Root mean square per pixel over 10 iterations of the baseline (left) and ground truth (right). The values in each pixel are explained in the text.

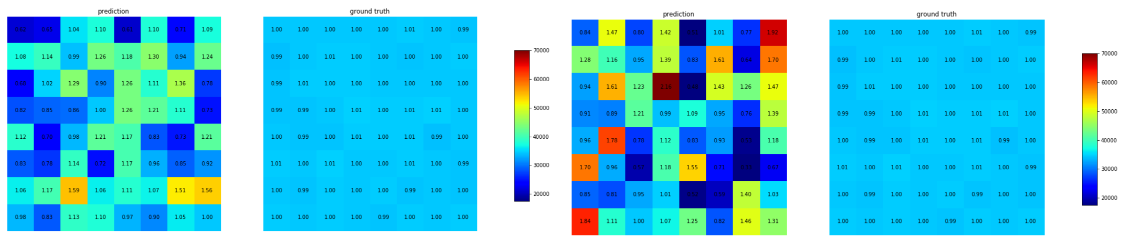

Figure 6.4: Example of two of the ten instances of the baseline, trained with different initial seeds. The values in each pixel are explained in the text.

6.1 Modified Generator

We will now start to make full use of the generator capabilities as described in Section 4.4.1 and depicted in Figure 4.6. As a first step we experiment with different upsampling routines, that is different combinations of pixel shuffle and transposed convolution layers. The possible advantages of each have been mentioned in Section 4.4.1. In addition to that we allow for one further extension, namely the inclusion of additional RRDB blocks (see Figure 4.6) after upsampling. The reason for this is that so far all relevant convolutional layers are before the upsampling and it might be interesting to see if further processing of the image afterwards can improve performance. In practice the addition of RRDB blocks after upsampling is computationally expensive [53] because now all convolutions have to operate on the 160×160 feature maps so we can only allow for one additional RRDB block. Since we will exclusively work with an upsampling factor $\mathrm{f} = 8 = 2^3$ we will always have 3 upsampling layers. The standard option used so far was exclusively pixel shuffle. We will test three alternatives:

1. Alternating between pixel shuffle and transposed convolutions, so for the $\mathrm{f} = 8$ case the upsampling sequence is [pixel shuffle, transposed convolution, pixel shuffle].

2. Using exclusively transposed convolutions.

3. Using exclusively pixel shuffle (default mode) and adding one RRDB block after the upsampling layers.

All tests are based on the baseline model with a pure patch image early stopping, the relevant comparison is therefore Figure 6.1. The results are shown in the Figures 6.5, 6.6, 6.7. We can derive some conclusions from these: First of all in all cases the distribution of n^{th} hardest pixels is at least as good as in the default case Figure 6.1, suggesting again that these distributions can be learned in a stable way for the top data. Concerning the patch and mean image, the most blatant observation arises in the case with additional RRDB block. The overpopulated pixel in the patch image has become even harder and correspondingly the repeating patterns in the mean image are even more pronounced. We note that the same poor performance can be seen in any other combination including RRDB blocks after upsampling and therefore discard this option.
In contrast to that the other two alternatives, alternating the upsampling and using transposed convolutions exclusively, show substantial progress: The overactive pixel in the patch image is now much less pronounced compared to the ground truth average and overall the patch image starts to look more uniform. In the mean jet image we still see some fluctuations but these now appear less repetitive. We conclude that both approaches yield valuable improvements.

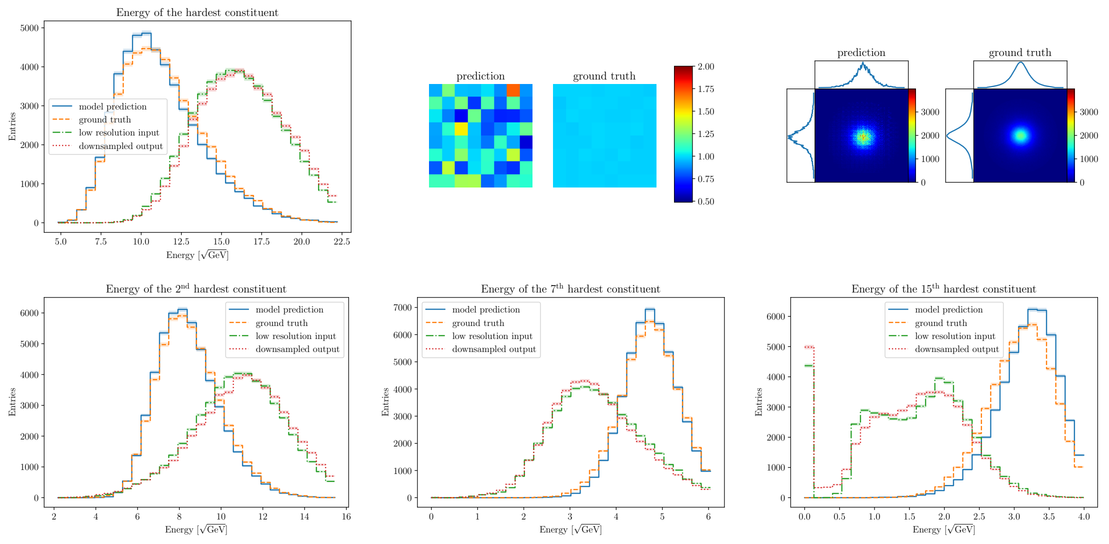
Figure 6.5: Evaluation with alternating upsampling. In addition to some ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

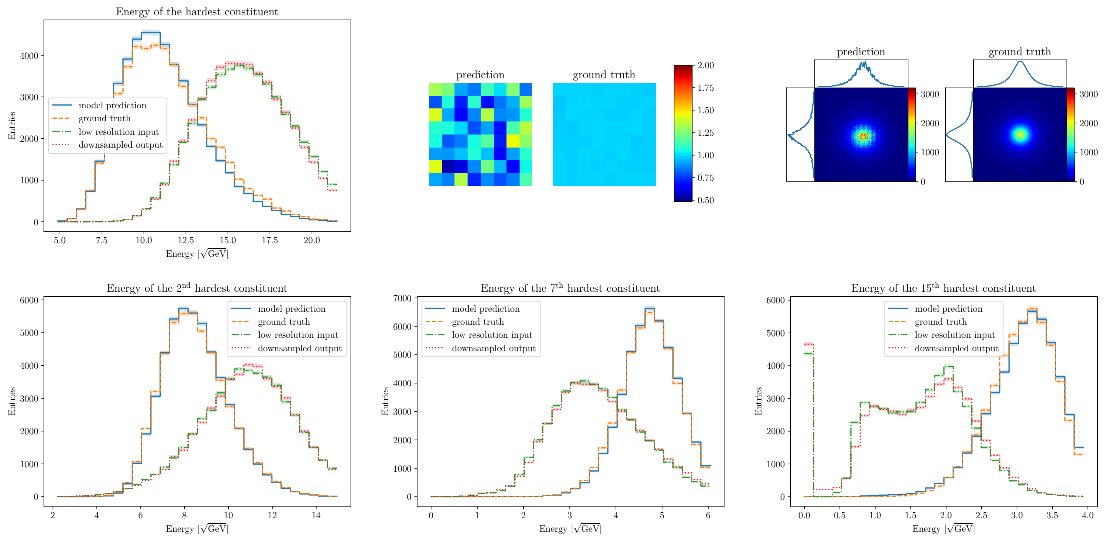
Figure 6.6: Evaluation with exclusively transposed convolution upsampling. In addition to some ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

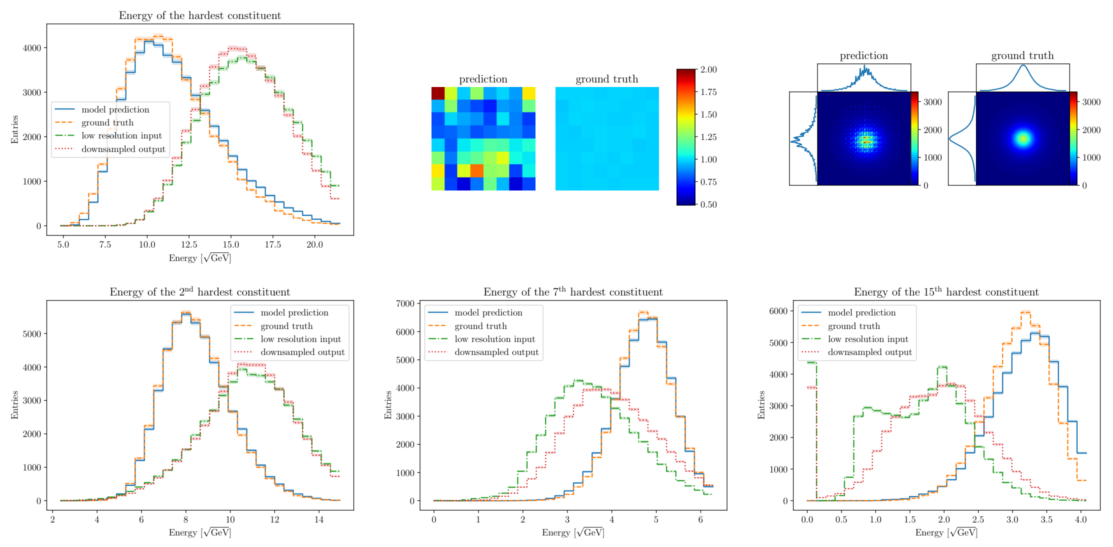

Figure 6.7: Evaluation with additional RRDB block. In addition to some ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

6.2 Second Discriminator

What is left before we make use of the full power of the network is activating the second set of discriminators, one for the standard images and one for the power raised images. As described in Section 4.4.2 these will have the same architecture as the default discriminators but their weights are reset after a fixed number of training batches [54]. Unfortunately we do not know a solid, compelling theoretical argument for this setup but we can motivate this a little bit by noting that the total discriminator response is the average of the individual responses, therefore, after resetting, the second discriminator response will be very poor in the sense of not being able to discriminate between real and generated images. This in turn implies that even a well performing generator has to make weight changes again in order to decrease the adversarial loss, thereby breaking up the pattern of putting most energy into a single SR pixel. The tests performed are similar to those of the previous sections, the relevant changes are summarized in Table 6.1.

Upsampling Procedure	Reset Interval
Pixel Shuffle	15000
Alternating	5000
Alternating	20000
Transposed Convolutions	15000

Table 6.1: Different setups with second discriminator.

The evaluations are shown in Figures 6.8, 6.9, 6.10, 6.11.

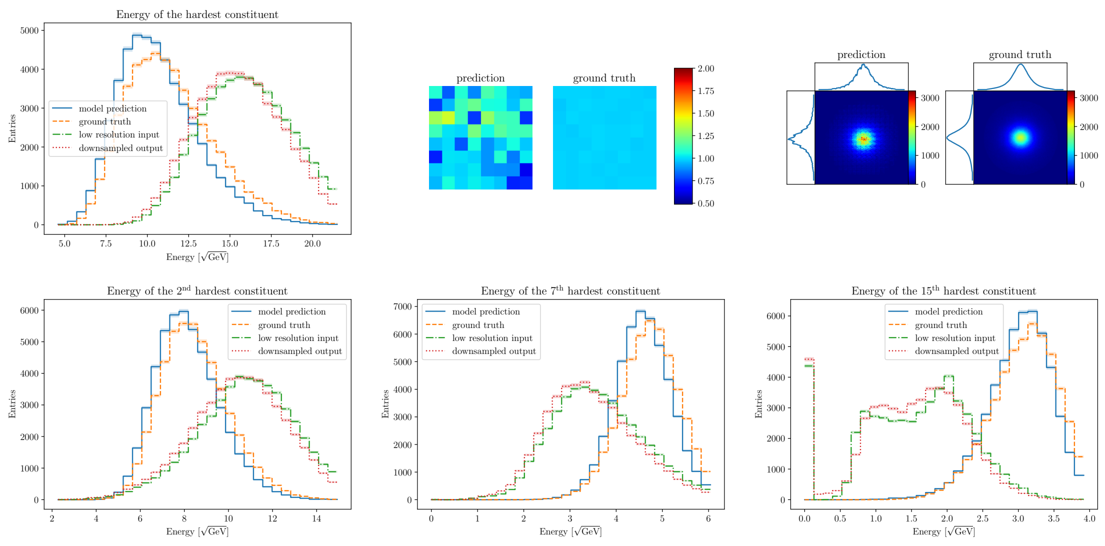

Figure 6.8: Evaluation with reset interval 15000 and pixel shuffle upsampling. In addition to some ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

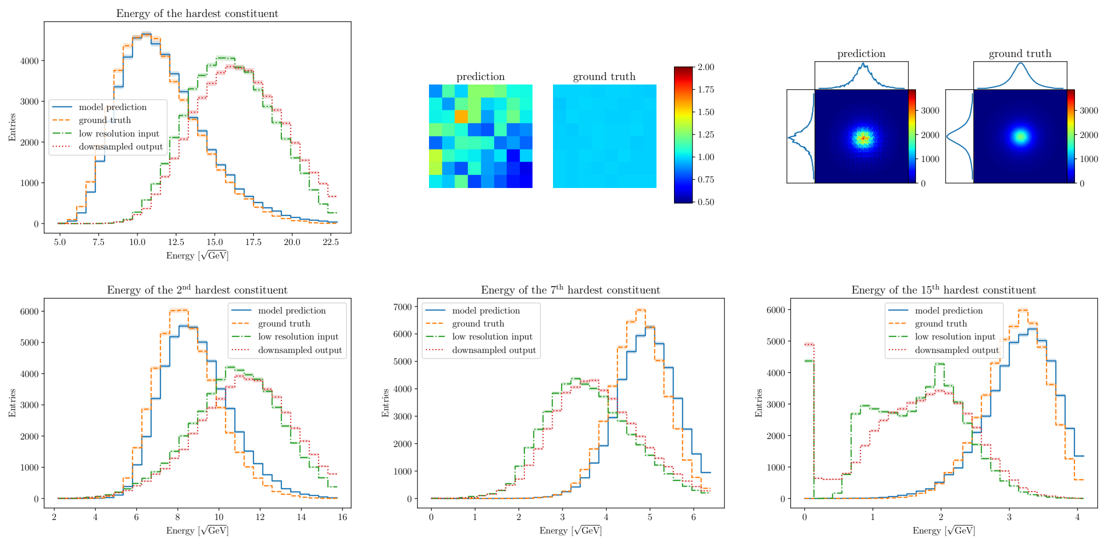

Figure 6.9: Evaluation with reset interval 5000 and alternating upsampling. In addition to some ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

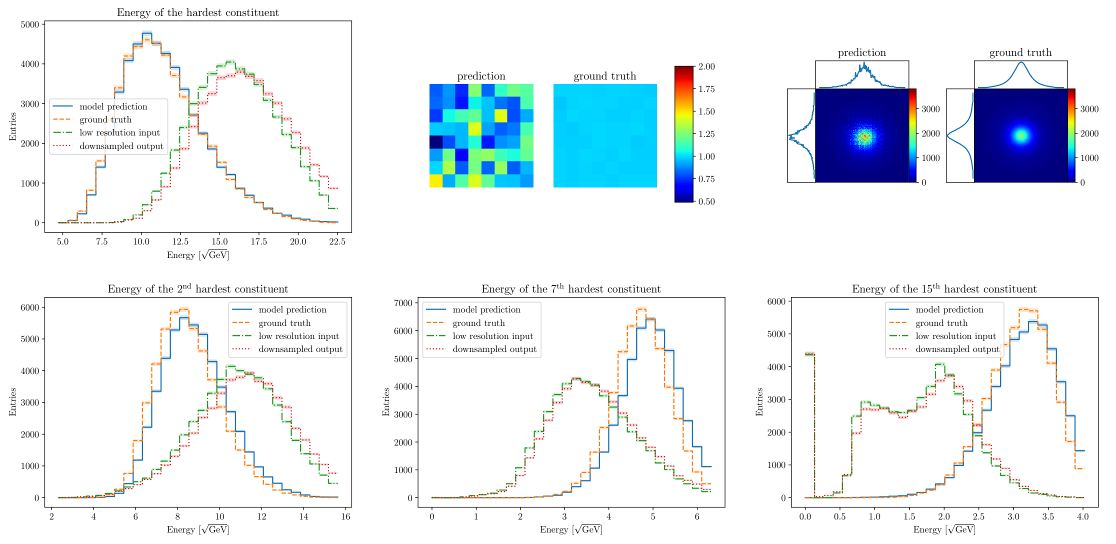
Figure 6.10: Evaluation with reset interval 20000 and alternating upsampling. In addition to some ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

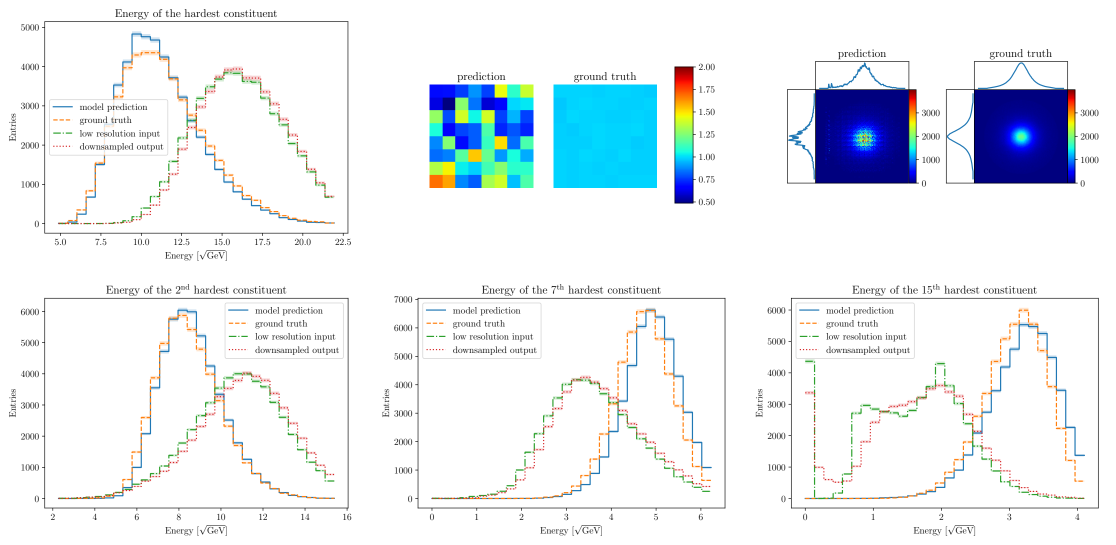
Figure 6.11: Evaluation with reset interval 15000 and transposed convolution upsampling. In addition to some ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

Again we see that in all cases the ordered pixel distributions are well approximated so the relevant changes are once more contained in the patch image and mean jet image. While the patch image in Figure 6.11 is comparable but not improved compared to the results in the previous section, the ones in Figures 6.8, 6.9 and 6.10 have become even more uniform than before, leading to the conclusion that the first three variations in Table 6.1 are highly performant.

From the evaluation plots it may appear that the first option, pixel shuffle upsampling with a second discriminator, performs exceptionally well but these results have to be verified by training the same setup multiple times, in accordance to our discussion of seeding effects. Upon doing so we found that on average the second variation, alternating between pixel shuffle and transposed convolution with a second discriminator reset interval of 20000 batches works best and therefore this is the setup we will adopt. We note however that there is no real performance drop in the reset range from 15000 to 30000 batches. The full set of hyperparameters for our optimized top model are summarized in Table 6.2

#RRDB	batch size	β	rescaling p	λ_{reg}	λ_{std}	λ_{pow}	λ_{HR}	λ_{LR}	λ_{adv}	λ_{patch}	reset
15	15	0.1	0.3	0.001	0.2	1	1	0.1	0.01	0.1	20k

Table 6.2: Hyperparameters for our optimized top model including the second discriminator reset interval. The upsampling procedure is alternating between pixel shuffle and transposed convolutions.

7 Upsampling QCD Jets

Having achieved impressive performance on the top dataset we will now try to upsample qcd jets. Based on what was said in Section 2.2.1 we know that these will behave differently than the tops. Instead of a three-prong structure we now expect pixel entries to arise solely from collinear and/or soft splittings which in the jet images will manifest in an increased activity near the center and a more narrow spread in the $\eta - \phi$ plane. From a machine learning perspective this will probably be a more difficult task since there will be less information in the LR image, most of it encapsulated in the hardest pixel. In spite of that we might hope that our network setup and modifications from the previous sections have been general enough to lead to a good performance. We trained a model with parameters as in Table 6.2 on qcd jets and show the evaluation on a qcd jet testset in Figure 7.1. From this we can derive a number of conclusions: As conjectured the activity is sharply peaked around the center as can be seen in the jet mean image, so much that even the ground truth patch image is not entirely uniform anymore. The SR patch image does not feature any unreasonable overactive pixel, even less so than in the top case, but this is supposedly just a reflection of the easier distribution of activity in the HR image. The n^{th}-hardest pixel distributions for the SR image are well described and approximate the HR distributions, albeit not as well as in the top case. What blatantly sticks out however is that LR_{gen} distributions are shifted and slightly distorted compared to the LR distributions, at least for the leading four pixels. This is in stark contrast to the top sample where the relative distance of the SR to HR distributions was completely mirrored by the LR_{gen} to LR distance. The fact that the SR distributions are closer to the ground truth than the LR_{gen} counterparts seems disturbing but we have to keep in mind that the LR_{gen} image arises through sumpooling the SR image in contrast to the ground truth were HR and LR are independent, therefore we have to be careful what we compare: A priori it is not given that the n^{th}-hardest LR_{gen} pixel results from the n^{th}-hardest SR pixel, the details about what constitutes the first is captured by the exact distribution of energy in the SR pixels. Because the qcd activity is more peaked around the center, getting this distribution right is more difficult than in the top case and apparently the network is not getting it right so far. This demands a closer investigation of how the LR_{gen} distributions come about.

To be specific we are interested which pixels in the SR image are clustered into a given hard pixel in the LR_{gen} image by the sumpooling. To this end we perform the sumpool manually for a few images in the training set, keeping track of which SR pixels constitute a given LR_{gen} pixel. An example is shown in Figure 7.2. Note that there we also sumpooled the HR image (top-left) so the corresponding LR image (bottom-left) is not the same as $\mathrm{LR}_{\mathrm{gen}}$. What we find is that the hardest pixel in the pooled SR image (bottom-right) is made up by the three hardest pixel in the SR image (top-right) in addition to some very soft pixels. In contrast, the hardest pixel in the pooled HR image just includes the hardest, 4th, 7th and 10th hardest HR pixels.

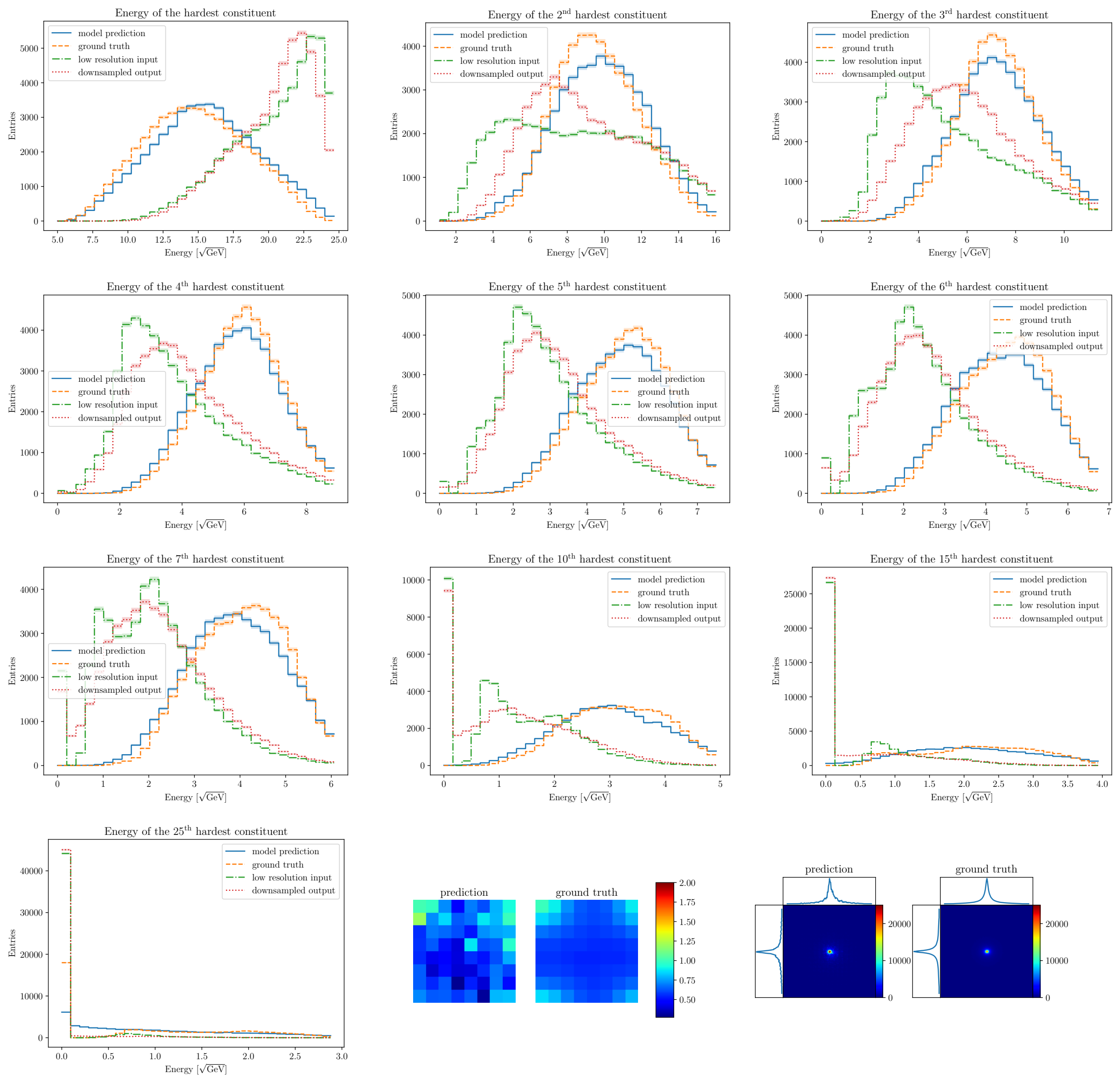

Figure 7.1: Evaluation of the top-optimized model on qcd jets. In addition to the ordered pixels (with a square-root scaling), we show the patch image and the mean image. The zero-bin in energy collects jets with too few entries.

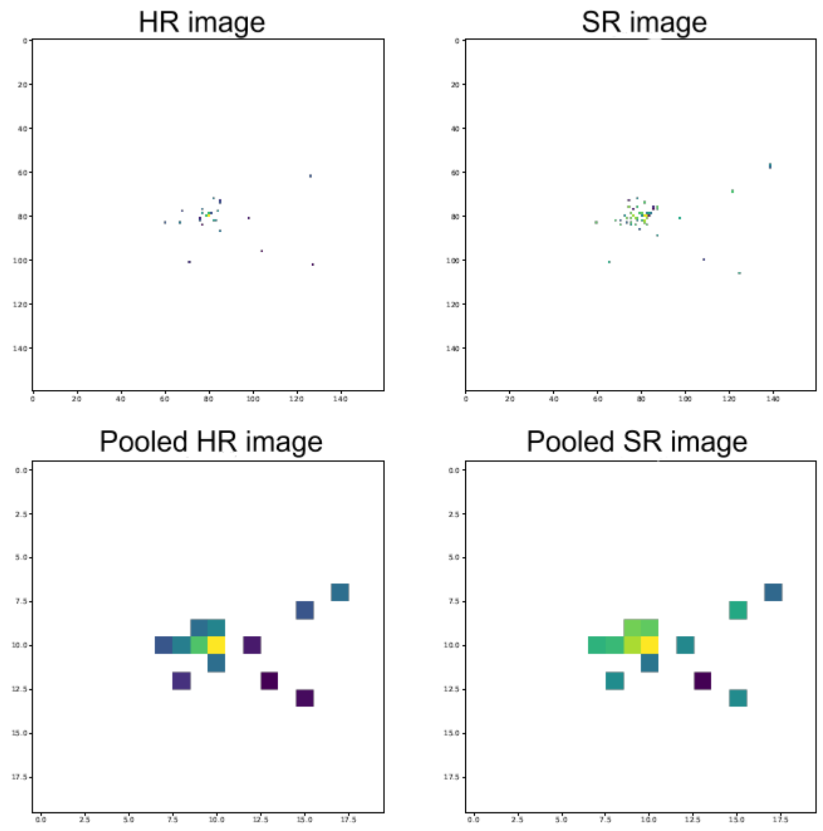

Figure 7.2: Example of a manually sumpooled qcd jet image.

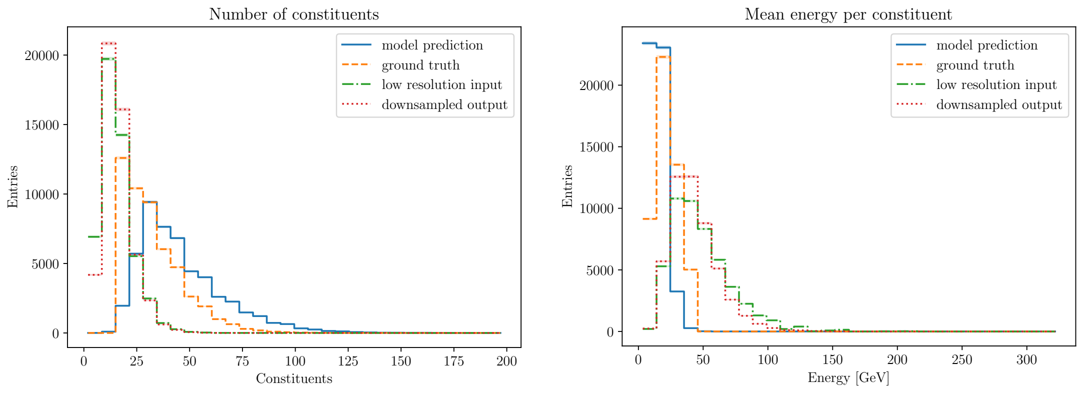

Figure 7.3: Number of non-empty pixels (left) and average energy per pixel (right) for the model trained on qcd jets.

Repeating this analysis for some more jet images we can conclude the following:

1. For qcd jets all the hard pixels are very close to the center. This leads to the first few hardest pixels in the SR image to all be sumpooled into the hardest pixel of the $\mathrm{LR_{gen}}$ image.
2. As seen in the left plot of Figure 7.3 the LR and $\mathrm{LR_{gen}}$ images have essentially the same number of non-empty pixels but the SR image has more non-zero pixels on average than its HR counterpart, thus these additional non-zero entries in the SR image must be included in the 8×8 pixels that constitute each $\mathrm{LR_{gen}}$ pixel via sumpooling. On the other hand, as seen in the right plot of Figure 7.3, the SR pixels are slightly softer than their HR counterparts so that the total energy in the SR and HR image is approximately the same. This is in accordance to what we already found earlier for top jets.

Based on this we are lead to believe that the first few hardest pixels in the $\mathrm{LR_{gen}}$ image already include all hard pixels in the SR image since these are all near the center, in addition to very soft pixels that are ubiquitous in the SR image but more likely to be distributed near the center. This explains why the leading $\mathrm{LR_{gen}}$ pixels are systematically harder than the leading LR pixels. The main difficulty for the network is therefore to disentangle the complicated distribution of energy around the center.

Having made these observations we may ask if physics can guide us from a model optimized on top jets towards a model working well on qcd jets. To summarize the challenges from qcd: In the top jet dataset hard pixels are relatively well separated, therefore learning the high resolution distributions well automatically leads to well approximated downsampled distributions. On the other hand, this also implies that the jet image as a whole is more spread out than its qcd equivalent, something we have noted earlier. This leads to a more difficult topology of the jet as a whole. In the qcd case the situation is different: As the manual downsampling analysis showed there is a much more complicated relation between the HR and LR distributions and our model so far only learned to approximate the former well. This is not actually so surprising because up to now the model did not have an actual impetus to care about the LR distributions due to us only including the HR distributions in the early stopping evaluation (see Section 4.5). Thus a first reasonable change is to include also the downsampled distributions in the evaluation, an unnecessary step for the top jets but important for qcd ones because the latter are all distributed close to the center where disentangling them is much more challenging. Further we might ask if our model really needs to be so deep. We know that qcd jets are determined by the collinear splittings and therefore this is what the network really needs to learn when improving on the LR information. An overcomplex model might have difficulties to backpropagate this information, leading to very well approximations for whatever we choose to explicitly evaluate but failing to see the bigger picture. On the other hand the model also must not be undercomplex so a reasonable compromise seems to reduce the number of RRDB blocks from 15 to 10. This will be our final proposal for the model and the dictating hyperparameters are summarized in Table 7.1. Extensively testing this model setup is the task performed in the next section.

#RRDB	batch size	β	rescaling	λ_{reg}	λ_{std}	λ_{pow}	λ_{HR}	λ_{LR}	λ_{adv}	λ_{patch}	reset
10	15	0.1	0.3	0.001	0.2	1	1	0.1	0.01	0.1	20k

Table 7.1: Hyperparameters for our final model proposal. The relevant changes are reducing the number of RRDBs and including the LR distributions in the early stopping evaluation.

8 Cross Check Evaluation

So far we have considered top and qcd jets in isolation but our overarching goal is to achieve a model setup that can perform well on both types of jets. To show that the model proposed in the last section does live up to this task we will now first train and evaluate it on qcd jets, then we will take a step back and show that also the upsampling of top jets is performed well. Finally we will analyse a cross-check by evaluating the model trained on qcd data using top jets and vice versa.

8.1 High Level Jet Observables

Up to this point we have treated the model's output like we would any image, describing its performance entirely in terms of the individual pixel entries and a few collective distributions (the patch- and jet mean image) but since we know that our images arose from jets it is natural to include typical jet observables in the evaluation, probing if the network can reproduce what we know on physical grounds even though this has not been explicitly ingrained into the performance metric during training. The high level jet observables we will consider in what follows are [55; 56; 57; 58]:

$$\begin{aligned} m_{\text{jet}} &= \left(\sum_i p_i^\mu\right)^2 & w_{\text{pf}} &= \frac{\sum_i p_{\text{T},i}\Delta R_{i,\text{jet}}}{\sum_i p_{\text{T},i}} \\ C_{0.2} &= \frac{\sum_{i,j} p_{\text{T},i}p_{\text{T},j}(\Delta R_{i,j})^{0.2}}{(\sum_i p_{\text{T},i})^2} & \tau_N &= \frac{\sum_k p_{\text{T},k}\min(\Delta R_{1,k},...,\Delta R_{N,k})}{\sum_k p_{\text{T},k}R_0}\ . \end{aligned} \tag{8.1}$$

The variables appearing on the right side of these equations have been introduced in Section 2. The sum is taken over all jet constituents. The invariant mass m_{jet} will be very different for top and qcd jets: We expect the distribution of the former to peak around the top mass ≈172 GeV while for the latter, arising from massless partons, we expect detector effects to have the distribution peak around 20 to 50 GeV.

The *girth* w_{pf} will be dominated by the hard pixels and essentially describes their placement relative to the jet axis. $C_{0.2}$ is the two-point energy correlator.

τ_N is the N-subjettiness, a measure of the likelihood that the jet arose from N subjets. In practice we will use the N-subjettiness ratios τ_2/τ_1 and τ_3/τ_2 because these can discriminate between 2- and 3-prong jets which we expect to be important based on the 3-prongness of top jets.

Before evaluating a model with these jet variables it is first interesting to see how

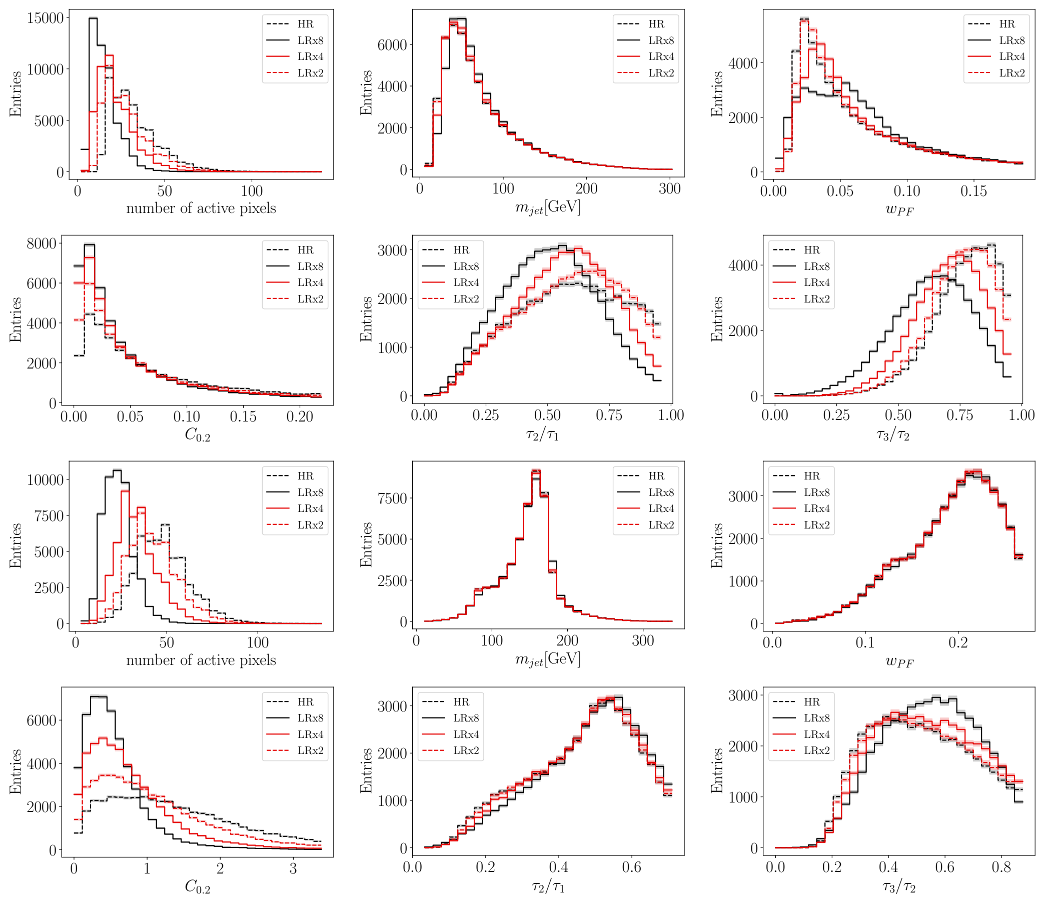
Figure 8.1: Distribution of the number of active pixels and high-level observables m_{jet}, τ_2/τ_1, τ_3/τ_2, w_{pf} for the ground truth HR and different ground truth LR images. Qcd jet results shown in top rows, top jet results in the bottom rows.

they inherently differ between different resolutions. To this end we now make use of all the different low resolution data, namely LRx2, LRx4 and LRx8 for respective factors of 2, 4 and 8 between LR and HR image. In Figure 8.1 we plot the jet variable distributions as well as the number of non-empty pixels for these ground truth images evaluated on qcd jets (upper part of the Figure) and top jets (lower part of the Figure).

As expected the jet mass is essentially invariant across the different resolutions and peaks around 170 Gev for tops and around 50 GeV for qcd jets. In both cases the different resolutions interpolate between the number of active pixels, highest for the HR image and lowest for the factor 8 downsampled one. Further we see that the τ_2/τ_1 ratio is the same across all resolutions for the top jets, this reflects their innate three-prong nature, while we see a difference between resolutions for τ_3/τ_2. In contrast to this both subjettiness ratios differ with resolution in the qcd case. Lastly the energy correlator $C_{0.2}$ is essentially the same for all qcd resolutions but clearly interpolates between resolutions for tops while the girth w_{pf} this is reversed between tops and qcd jets. We can attribute this to our earlier findings, the more widespread leading pixels for tops allowing the energy correlator to differ when downsampling to a lower resolution as slowly more and more hard pixels are put together while the leading pixels in qcd images are already so near the center that downsampling does not change much.

8.2 QCD Evaluation

Having familiarized with the high level jet observables we will now present the results when evaluated for the model SR images. In Figure 8.2 we show the evaluation of the model proposed in Table 7.1 when trained and evaluated on qcd jet images. We see that the jet mean image and patch image are reasonable uniform and free of too pronounced repeating patters. In obvious contrast to our first qcd jet evaluation, Figure 7.1, the ordered pixel distributions are now very well approximated for HR and LR, at least for the leading pixels and the soft noise regime. We take this as a conformation of the changes proposed at the end of the last section. The biggest discrepancy between LR and LR_{gen} distribution arises for the 7^{th}- and 10^{th}-hardest pixel but we already expected such a behavior earlier because this regime interpolates between hard pixels and soft noise and therefore was conjectured to be the most difficult to learn. All the high-level jet observables are well approximated by the SR and LR_{gen} distributions, especially the invariant mass is clearly learned. Further the N-subjettiness ratio τ_3/τ_2 is approximated slightly worse by the SR image compared to HR than the τ_2/τ_1 ratio. This suggests that the network is not generating enough splittings of the hard parton. All together we see a clear improvement compared to the qcd evaluation based on the top-optimized model but we now have to investigate how well the current model still works for top jets.

8.3 Top Evaluation

The optimal top jet evaluation was shown for the ordered pixels and the patch and jet mean image in Figure 6.10. To conclude that our proposed hyperparameter changes in Table 7.1 really improve the big picture the performance of the new model compared to this optimized model must at least not be worse than the performance of the optimized top model trained on qcd jets in Figure 7.1. The top-trained model evaluated on top jets is shown in Figure 8.3. We see that the first four leading pixel distributions are described extremely well by both SR and LR_{gen}. These correspond to the well separated hard partons arising in the top decay and are therefore easy to learn even with the less complex network. Again, starting from the 7^{th}-hardest pixel we see clear deviations of the SR distribution compared to the HR one. This suggests that this cross-over regime is similar for both top and qcd jets and indeed the soft pixel distributions start to resemble their qcd equivalents. One difference is that even for soft pixels the number of zero entries is substantially lower than for qcd

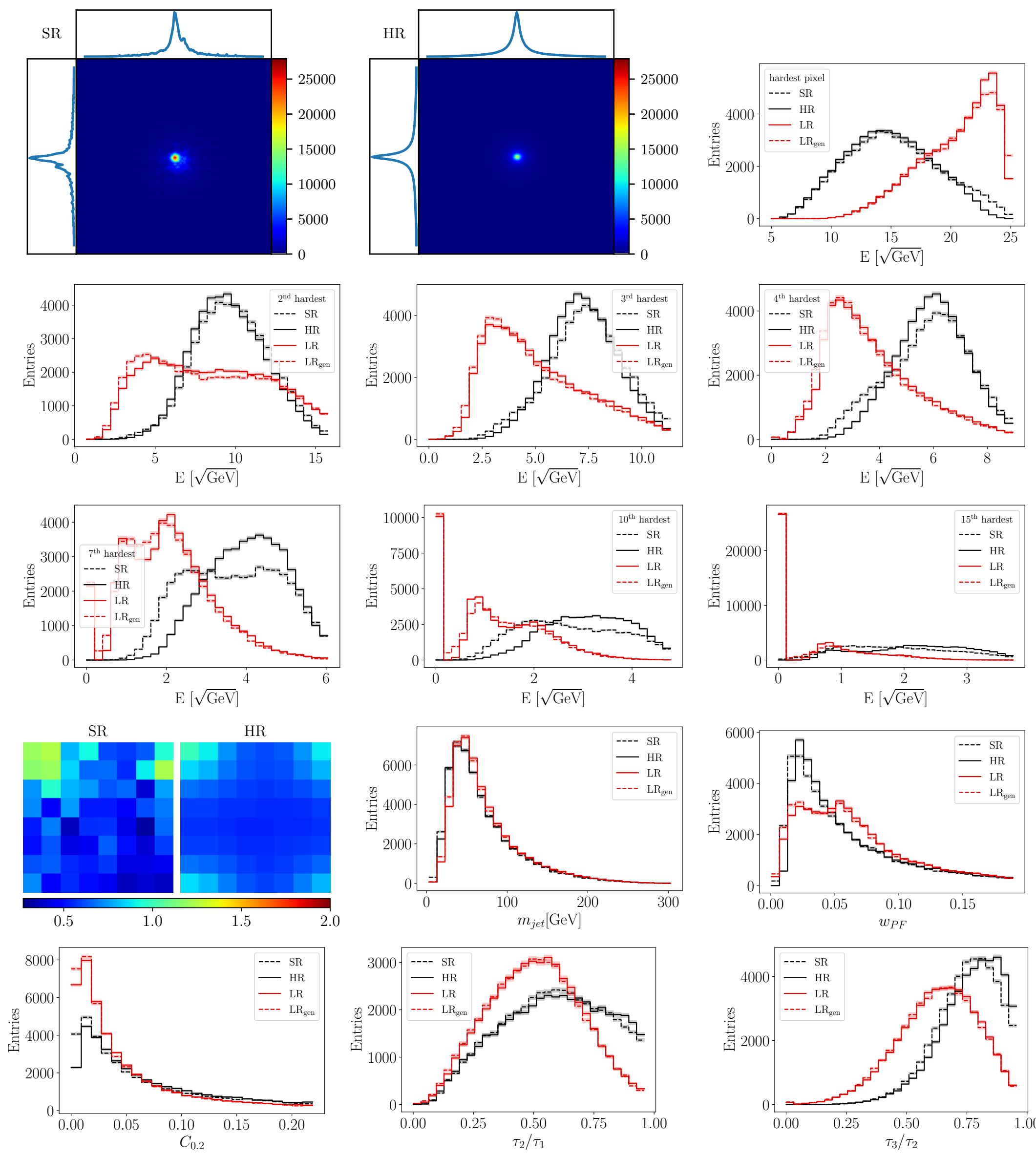
Figure 8.2: Evaluation of a network trained on QCD-jets and evaluated on QCD-jets. We show square root scaled ordered pixel distribution, mean and patch image as well as the high-level jet observables. The zero-bin in energy collects jets with too few entries.

images, indicating that there is more structure left to be learned in top jet images and this can be used by the network to approximate the soft pixel distributions slightly better than for qcd jets. The ordered pixel distributions are arguably approximated as well as in the optimal model evaluation however the mean jet image features a more pronounced repeating pattern than in Figure 6.10. This however is no surprise because in fact diminishing this pattern was the sole purpose of the optimized model. As for qcd jets all high level observables are very well approximated. There is a minimal shift of the jet mass in the SR distribution compared to HR. In contrast to qcd jets the N-subjettiness ratios are equally well learned. Again this can be attributed to the three-prongness of top jets, resulting in three well separated hard partons. Overall the performance is at best minimally worse than for the optimized model and therefore we can conclude that our final model changes indeed lead to an increased performance when top and qcd jets are not regarded in isolation anymore.

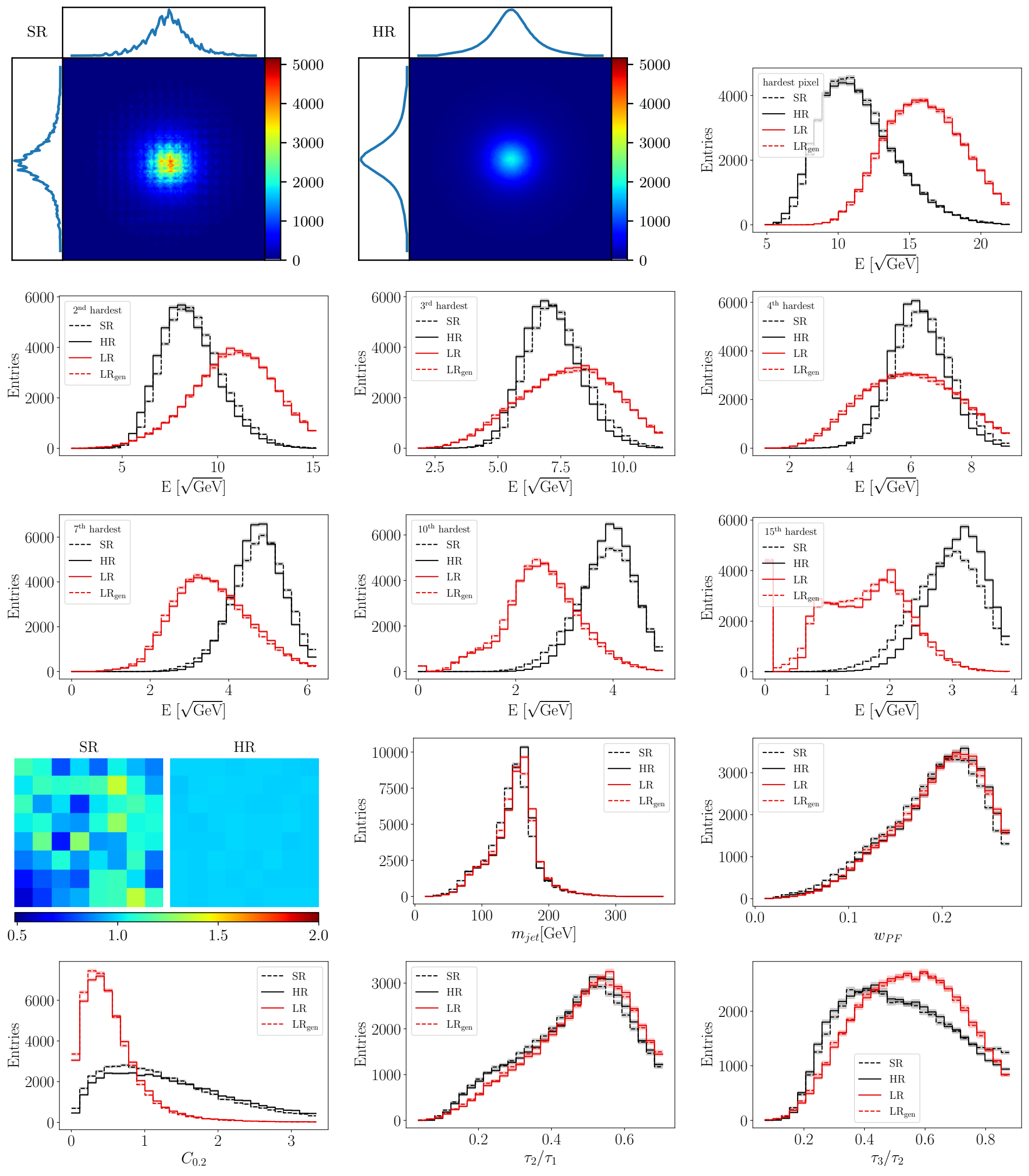

Figure 8.3: Evaluation of a network trained on top-jets and evaluated on top-jets. We show square root scaled ordered pixel distribution, mean and patch image as well as the high-level jet observables. The zero-bin in energy collects jets with too few entries.

8.4 Cross-Evaluation

So far we have seen that we could find model parameters that can learn to upsample top jets and qcd jets but in a pointed remark we could ask if the network has really learned some physics or merely learned to upsample images. After all, in any slightly realistic application the network would have to work on jets regardless from which process they arose. To qualitatively check our model dependence on the dataset we will now train the model on either top or qcd jets but perform the evaluation on the respective other dataset. The results for a model trained on qcd jets but evaluated on top jets can be seen in Figure 8.4. The results are promising: The ordered pixel distributions are still learned very well albeit with larger deviations than in the top trained model. The jet mean image features some stronger noise and the repeating pattern is clearly visible in the top-left panel but in the patch image we see that the most populated SR pixel is just 1.5 times harder than the average HR pixel which shows that the significant improvements made in Section 6 are still present. The shift in the jet mass between SR and HR has become a little bigger but overall the jet observables are still well approximated. The discrepancy between the SR and HR τ_3/τ_2-ratio has increased whereas the τ_2/τ_1-ratio has remained stable. This is due to the network being trained on qcd jets and therefore not conditioned to create a three-prong signature.
The last case, a model trained on tops but evaluated on qcd jets, is displayed in Figure 8.5. Based on all our previous conclusions we expect this to be the most challenging case. The simple, sharply peaked jet mean image is again learned well and correspondingly the patch image is still quite uniform but the SR distributions of the leading pixels show significant deviations compared to the HR case while the soft noise distributions, starting with the 10^{th}-hardest pixel are accurately approximated. This is in accordance to what we had to say earlier about the difficulties of upsampling qcd jets: The model trained on tops has not learned to disentangle the intricate activity in the center of qcd images, especially it never learned to regard LR distributions during training since these are intimately related to the HR distributions for tops. Interestingly the deviations in the leading pixels are now between SR and HR, completely reversed to the deviations between LR and LR_{gen} that we started with in Section 7. In the high-level jet observables we now see the biggest deviation in the τ_2/τ_1 ratio between SR and HR distribution. This again arises through the three-prong structure of the training top dataset, conditioning the network on a higher subjet count.

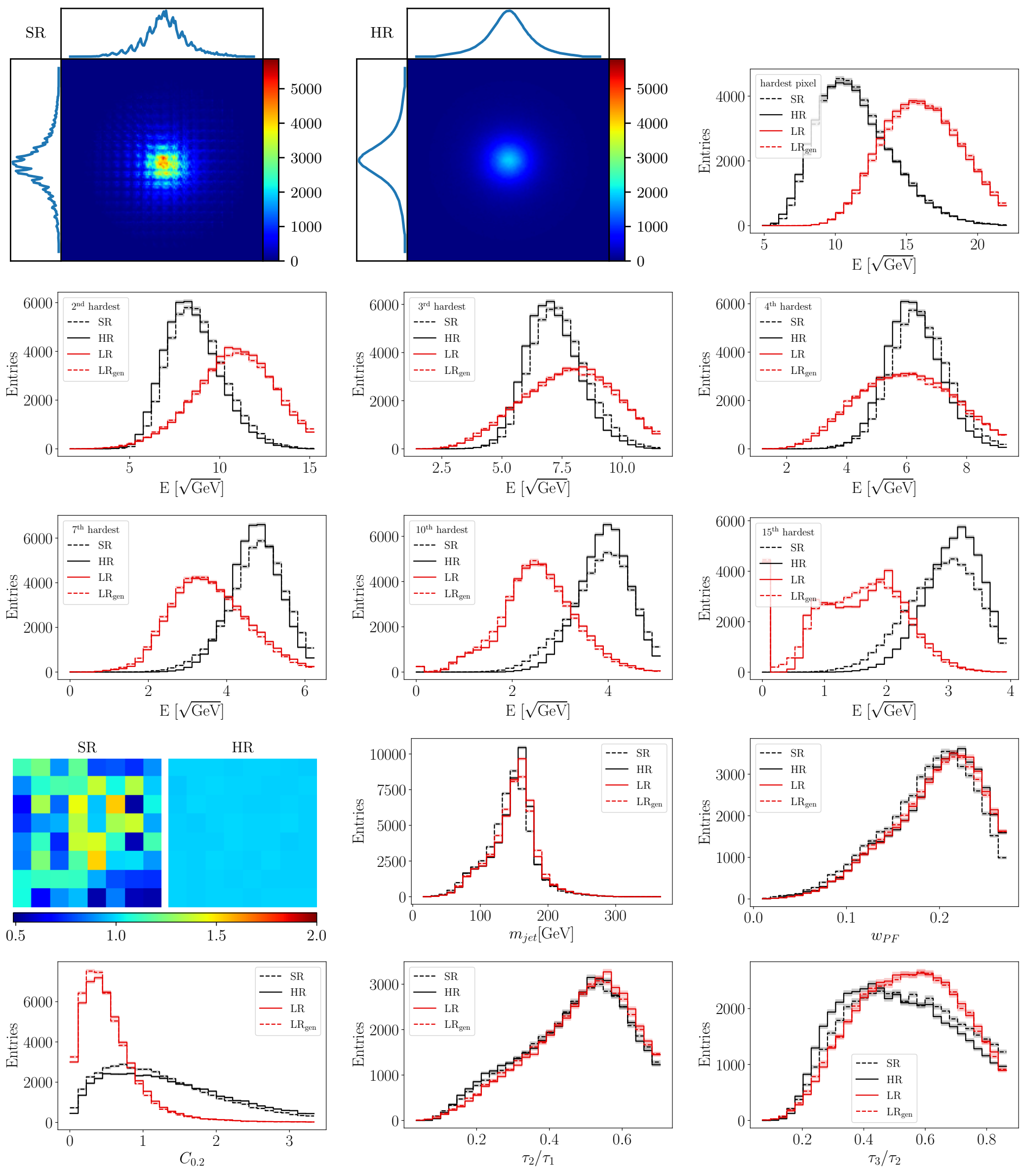

Figure 8.4: Cross-Evaluation of a network trained on qcd-jets and evaluated on top-jets. We show square root scaled ordered pixel distribution, mean and patch image as well as the high-level jet observables. The zero-bin in energy collects jets with too few entries.

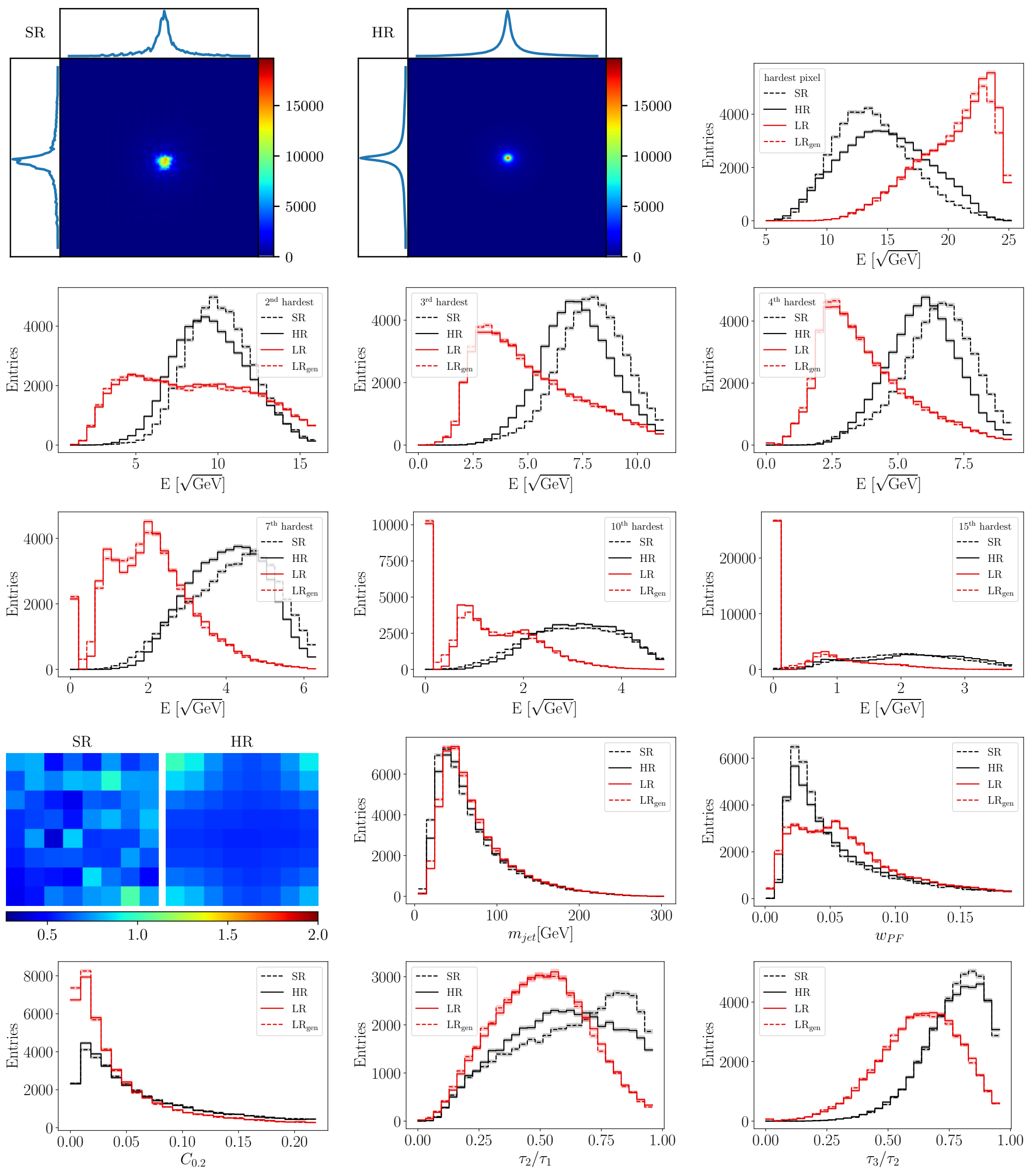

Figure 8.5: Cross-Evaluation of a network trained on top-jets and evaluated on qcd-jets. We show square root scaled ordered pixel distribution, mean and patch image as well as the high-level jet observables. The zero-bin in energy collects jets with too few entries.

8.5 Single Event Performance

We have seen that our model can upsample top and qcd jets in a way that well approximates the ground truth distributions but all of this is essentially a statistical statement. It is natural to ask what happens when we compare SR and HR on an event-by-event basis. In some sense this is doomed to fail from the beginning: While in the dataset every HR image comes with a paired LR image, such a pairing is in itself not unique, or in other words given a LR image there are multiple HR images it could have possibly arisen from through the downsampling procedure. Despite that we might investigate if certain characteristic properties, like the energy of the leading pixels, can be retained on a single event case. To this end we compare how the two hardest pixel energies are transferred from HR to SR in Figure 8.6 for the top sample and in Figure 8.7 for the qcd sample. The top panels show the HR energy of a given event versus the SR energy of of the same event for the two leading pixels. The bisecting line corresponds to the pixel energy being the same in HR and SR. In both, top and qcd case, we see a substantial spread of activity away from the bisector, indicating that there is often an imbalance in energy. In the bottom panels we divide the distribution of energy into 5 distinct slices for the HR samples (bottom-left) and show how a value in a given slice is mapped by the model into the SR distribution (bottom-right). Again for both cases and both leading pixels we see that these distributions are significantly broadened for the SR images, further showing that on a single event basis the energy of the leading pixels is not retained in the SR image.

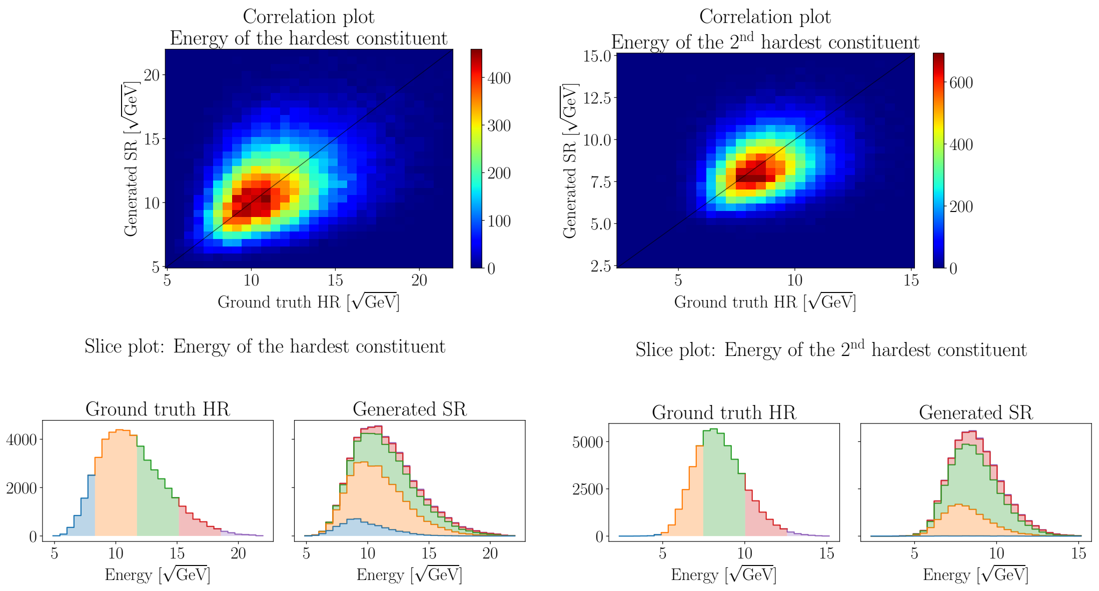

Figure 8.6: Correlation between two leading pixel energies in SR and HR images on an event-by-event base (top panel) and mapping of a certain energy range from HR to SR (bottom panel) for top-jet sample.

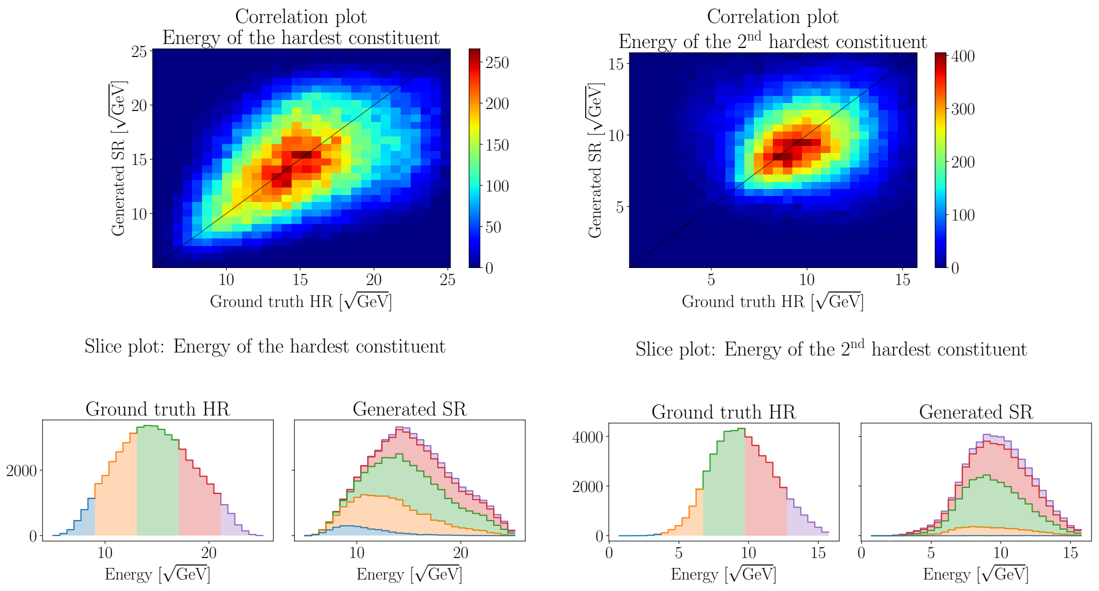

Figure 8.7: Correlation between two leading pixel energies in SR and HR images on an event-by-event base (top panel) and mapping of a certain energy range from HR to SR (bottom panel) for qcd-jet sample.

8.6 Conclusion and Outlook

We applied the task of single image super resolution to sparse jet images, describing what modifications are necessary starting from a Generative Adversarial Model built to yield optimal performance on regular, machine learning benchmark image sets.
We have showed that our model can reliably upsample top-jet images up to a factor 8, correctly reproducing low-level and high-level distributions and described the necessary modifications to reach similar performance on the more difficult qcd-jet data. Importantly we then demonstrated that our model is not dependant on any specific type of jet but can generalize from one sample to the other.
The work done so far was mostly proof-of-concept, much more has to be done until real experiments could benefit from it. Open questions and ideas for future work are:

- At what upsampling factor does the model fail? Due to our data pipeline any new resolution has to be created from scratch so that it was not possible to test the next step, 16x Upsampling.
- We have seen that some of the jet observables interpolate between different resolutions. Can we in turn make use of multiple resolutions during training, e.g. comparing the SR output after every upsample layer to the corresponding resolution ground truth? This could especially help to disentangle the complicated qcd patterns.
- Can we completely drop any separation between qcd and top data and instead simply train and evaluate on a large mixed sample? How does the model interpolate between the two extremes in such a case?
- In this thesis we were only concerned with performance, not computational cost or training time constraints. Is the immense complexity of the network really necessary to yield satisfying results? By how much does performance really decrease when lowering model complexity and what is the ideal thin model?

Some of these questions are currently investigated by our collaborators [7] and we look forward to what can be achieved in this direction.

Acknowledgments

First of all i would like to thank Lukas Blecher for providing the first functioning code, speeding up my first contact with deep learning and providing valuable work even after finishing his own thesis. I thank Tilman Plehn, Anja Butter and Jennifer Thompson for always offering support and useful ideas, especially when hyperparameter searches became frustrating, and broadening my insight for physics. Further i thank everyone who contributed to our (soon to be submitted) paper. Lastly i am grateful to the entire Pheno group for providing a fun and interesting working environment.

A Lists

A.1 List of Figures

A.2 List of Tables

Bibliography

[1] Fred Jegerlehner. The hierarchy problem of the electroweak standard model revisited, 2013, 1305.6652.

[2] Lars Bergström. Dark matter candidates. *New Journal of Physics*, 11(10): 105006, Oct 2009. ISSN 1367-2630. URL http://dx.doi.org/10.1088/1367-2630/11/10/105006.

[3] Christian Weinheimer and Kai Zuber. Neutrino masses. *Annalen der Physik*, 525(8-9):565–575, Aug 2013. ISSN 0003-3804. URL http://dx.doi.org/10.1002/andp.201300063.

[4] Erik Gerwick, Tilman Plehn, and Steffen Schumann. Understanding jet scaling and jet vetos in higgs searches. *Physical Review Letters*, 108(3), Jan 2012. ISSN 1079-7114. URL http://dx.doi.org/10.1103/PhysRevLett.108.032003.

[5] Benjamin Nachman. A guide for deploying deep learning in lhc searches: How to achieve optimality and account for uncertainty. *SciPost Physics*, 8(6), Jun 2020. ISSN 2542-4653. URL http://dx.doi.org/10.21468/SciPostPhys.8.6.090.

[6] Byron P. Roe, Hai-Jun Yang, Ji Zhu, Yong Liu, Ion Stancu, and Gordon McGregor. Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 543(2-3):577–584, May 2005. ISSN 0168-9002. URL http://dx.doi.org/10.1016/j.nima.2004.12.018.

[7] Pierre Baldi, Lukas Blecher, Anja Butter, Julian Collado, Jessica N. Howard, Fabian Keilbach, Tilman Plehn, Gregor Kasieczka, and Daniel Whiteson. How to gan higher jet resolution, TBA.

[8] Lukas Blecher. Applying super resolution to collider events, 2020. URL https://www.thphys.uni-heidelberg.de/~plehn/includes/theses/blecher_b.pdf.

[9] Gautam Bhattacharyya. Electroweak symmetry breaking and beyond the standard model physics — a review. *Pramana*, 72(1):37–54, Jan 2009. ISSN 0973-7111. URL http://dx.doi.org/10.1007/s12043-009-0004-0.

[10] Simone Marzani, Gregory Soyez, and Michael Spannowsky. Looking inside jets. *Lecture Notes in Physics*, 2019. ISSN 1616-6361. URL http://dx.doi.org/10.1007/978-3-030-15709-8.

[11] Alexandre Deur, Stanley J. Brodsky, and Guy F. de Teramond. The QCD Running Coupling. *Nucl. Phys.*, 90:1, 2016, 1604.08082.

[12] Ringaile Placakyte. Parton distribution functions, 2011, 1111.5452.

[13] TAO HAN. Collider phenomenology: Basic knowledge and techniques. *Physics In D 4 Tasi 2004*, Jul 2006. URL http://dx.doi.org/10.1142/9789812773579_0008.

[14] John C. Collins, Davison E. Soper, and George F. Sterman. Factorization of Hard Processes in QCD. *Adv. Ser. Direct. High Energy Phys.*, 5:1–91, 1989, hep-ph/0409313.

[15] Stefano Catani and Massimiliano Grazzini. Collinear factorization and splitting functions for next-to-next-to-leading order qcd calculations. *Physics Letters B*, 446(2):143–152, Jan 1999. ISSN 0370-2693. URL http://dx.doi.org/10.1016/S0370-2693(98)01513-5.

[16] Christopher Frye, Holmfridur Hannesdottir, Nisarga Paul, Matthew D. Schwartz, and Kai Yan. Infrared finiteness and forward scattering. *Phys. Rev. D*, 99:056015, Mar 2019. URL https://link.aps.org/doi/10.1103/PhysRevD.99.056015.

[17] George Sterman and Steven Weinberg. Jets from quantum chromodynamics. *Phys. Rev. Lett.*, 39:1436–1439, Dec 1977. URL https://link.aps.org/doi/10.1103/PhysRevLett.39.1436.

[18] A. Banfi, G.P. Salam, and G. Zanderighi. Infrared-safe definition of jet flavour. *The European Physical Journal C*, 47(1):113–124, May 2006. ISSN 1434-6052. URL http://dx.doi.org/10.1140/epjc/s2006-02552-4.

[19] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. Fastjet user manual. *The European Physical Journal C*, 72(3), Mar 2012. ISSN 1434-6052. URL http://dx.doi.org/10.1140/epjc/s10052-012-1896-2.

[20] S. Catani, Yuri L. Dokshitzer, M.H. Seymour, and B.R. Webber. Longitudinally invariant K_t clustering algorithms for hadron hadron collisions. *Nucl. Phys. B*, 406:187–224, 1993.

[21] Yu.L Dokshitzer, G.D Leder, S Moretti, and B.R Webber. Better jet clustering algorithms. *Journal of High Energy Physics*, 1997(08):001–001, Aug 1997. ISSN 1029-8479. URL http://dx.doi.org/10.1088/1126-6708/1997/08/001.

[22] Matteo Cacciari, Gavin P Salam, and Gregory Soyez. The anti-ktjet clustering algorithm. *Journal of High Energy Physics*, 2008(04):063–063, Apr 2008. ISSN 1029-8479. URL http://dx.doi.org/10.1088/1126-6708/2008/04/063.

[23] André H. Hoang. The top mass: Interpretation and theoretical uncertainties, 2014, 1412.3649.

[24] Thorsten Ohl. Drawing feynman diagrams with fx340-1 and metafont. *Computer Physics Communications*, 90(2-3):340–354, Oct 1995. ISSN 0010-4655. URL http://dx.doi.org/10.1016/0010-4655(95)90137-S.

[25] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[26] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 0033-295X. URL http://dx.doi.org/10.1037/h0042519.

[27] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings. URL http://proceedings.mlr.press/v15/glorot11a.html.

[28] Cs231n: Convolutional neural networks for visual recognition. URL https://cs231n.github.io/neural-networks-1/.

[29] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257, 1991. ISSN 0893-6080. URL http://www.sciencedirect.com/science/article/pii/089360809190009T.

[30] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2020. https://d2l.ai.

[31] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017, 1609.04747.

[32] HENRY J. KELLEY. Gradient theory of optimal flight paths. *ARS Journal*, 30 (10):947–954, 1960, https://doi.org/10.2514/8.5282. URL https://doi.org/10.2514/8.5282.

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017, 1412.6980.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015, 1512.03385.

[35] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014, 1406.2661.

[36] Andrew Aitken, Christian Ledig, Lucas Theis, Jose Caballero, Zehan Wang, and Wenzhe Shi. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize, 2017, 1707.02937.

[37] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks, 2015, 1501.00092.

[38] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

[39] Roman Kogler, Benjamin Nachman, Alexander Schmidt, Lily Asquith, Emma Winkels, Mario Campanelli, Chris Delitzsch, Philip Harris, Andreas Hinzmann, Deepak Kar, and et al. Jet substructure at the large hadron collider. *Reviews of Modern Physics*, 91(4), Dec 2019. ISSN 1539-0756. URL `http://dx.doi.org/10.1103/RevModPhys.91.045003`.

[40] Jessie Shelton. Tasi lectures on jet substructure, 2013, 1302.0260.

[41] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, and Peter Z. Skands. An introduction to pythia 8.2. *Computer Physics Communications*, 191:159–177, Jun 2015. ISSN 0010-4655. URL `http://dx.doi.org/10.1016/j.cpc.2015.01.024`.

[42] J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. Delphes 3: a modular framework for fast simulation of a generic collider experiment. *Journal of High Energy Physics*, 2014(2), Feb 2014. ISSN 1029-8479. URL `http://dx.doi.org/10.1007/JHEP02(2014)057`.

[43] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. Fastjet user manual. *The European Physical Journal C*, 72(3), Mar 2012. ISSN 1434-6052. URL `http://dx.doi.org/10.1140/epjc/s10052-012-1896-2`.

[44] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018, 1809.00219.

[45] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017, 1609.04802.

[46] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan, 2018, 1807.00734.

[47] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and

video super-resolution using an efficient sub-pixel convolutional neural network, 2016, 1609.05158.

[48] Wenzhe Shi, Jose Caballero, Lucas Theis, Ferenc Huszar, Andrew Aitken, Christian Ledig, and Zehan Wang. Is the deconvolution layer the same as a convolutional layer?, 2016, 1609.07009.

[49] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017, 1704.00028.

[50] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks, 2019, 1903.11680.

[51] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016, 1511.06434.

[52] Pytorch framework for cryptographically secure random number generation, torchcsprng, now available. URL https://pytorch.org/blog/torchcsprng-release-blog/.

[53] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. pages 5353–5360, 06 2015.

[54] Yasin Yazıcı, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in gan training, 2019, 1806.04498.

[55] Jason Gallicchio, John Huth, Michael Kagan, Matthew D. Schwartz, Kevin Black, and Brock Tweedie. Multivariate discrimination and the Higgs + W/Z search. *JHEP*, 04:069, 2011, 1010.3698.

[56] Andrew J. Larkoski, Gavin P. Salam, and Jesse Thaler. Energy Correlation Functions for Jet Substructure. *JHEP*, 06:108, 2013, 1305.0007.

[57] Jesse Thaler and Ken Van Tilburg. Identifying Boosted Objects with N-subjettiness. *JHEP*, 03:015, 2011, 1011.2268.

[58] Gregor Kasieczka, Nicholas Kiefer, Tilman Plehn, and Jennifer M. Thompson. Quark-Gluon Tagging: Machine Learning vs Detector. *SciPost Phys.*, 6(6):069, 2019, 1812.09223.

Erklärung:

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

Heidelberg, den 18.12.2020 .... [illegible]