WWU
MÜNSTER

MLHEP19 Coopetition: WGAN-GP

Jan Honermann

wissen.leben

ALICE

Why WGAN-GP?

- Regular GAN difficult to train [5], one has to balance generator and discriminator.
- WGAN [1] actually encourages a more powerful critic, no vanishing gradients.
- One has to enforce a restraint to get a K-Lipschitz function [3], avoided weight clipping here.
- Chose to run with GP [3] since it was the first one to run well, so I wanted to see how far I could push it.

Hyperparameters

- RMSprop [7] was used in training the generator and critic.
- Momentum-based schemes were avoided, since it has been reported that they can to lead to unstable behavior during training WGAN's [1].
- Learning rate: 10^{-5}
- weight-decay: 10^{-5}
- Batch size: 128
- Noise dimension: 1000
- Training was performed on all 50250 training samples.

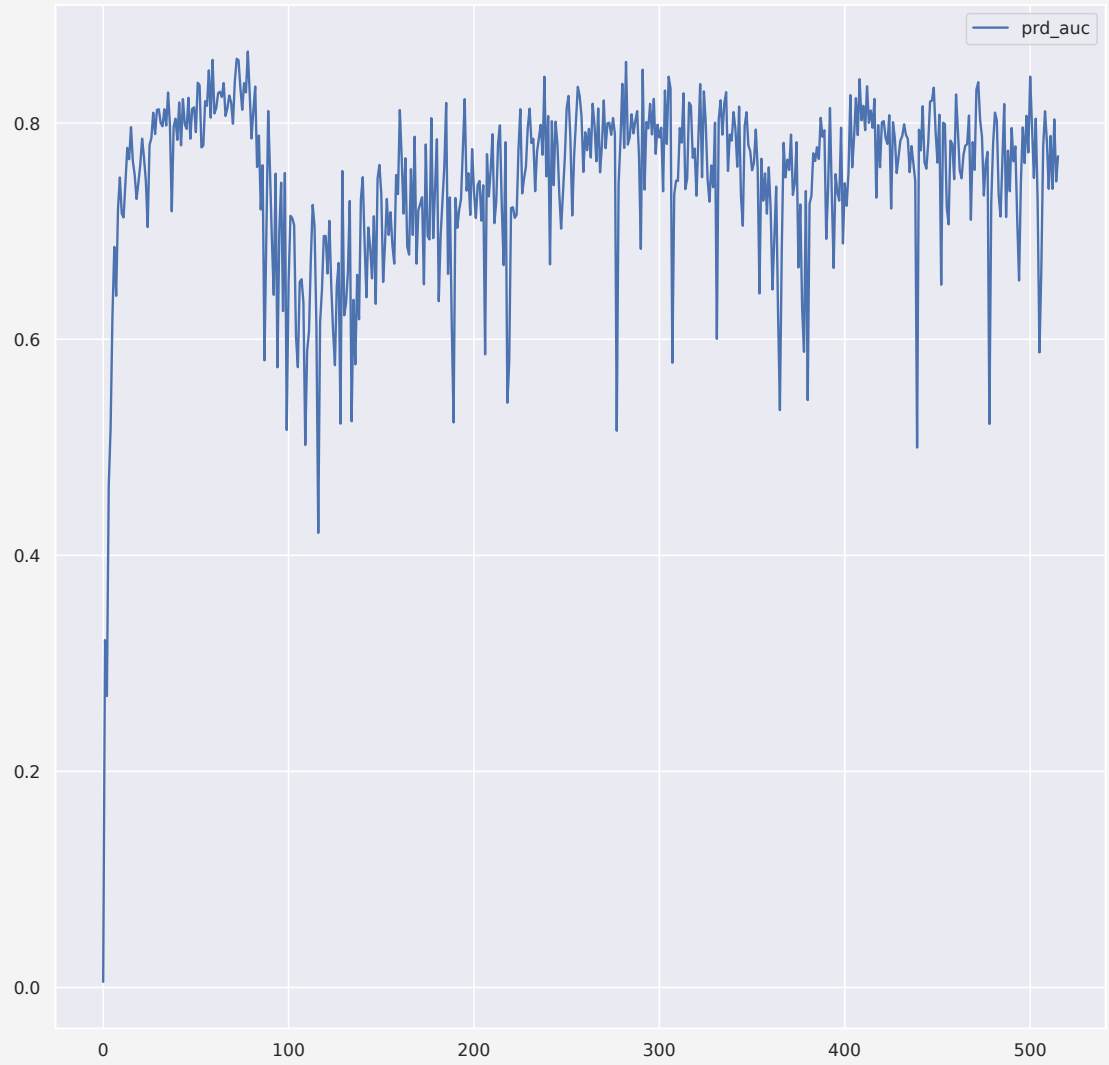
Additional regularisation

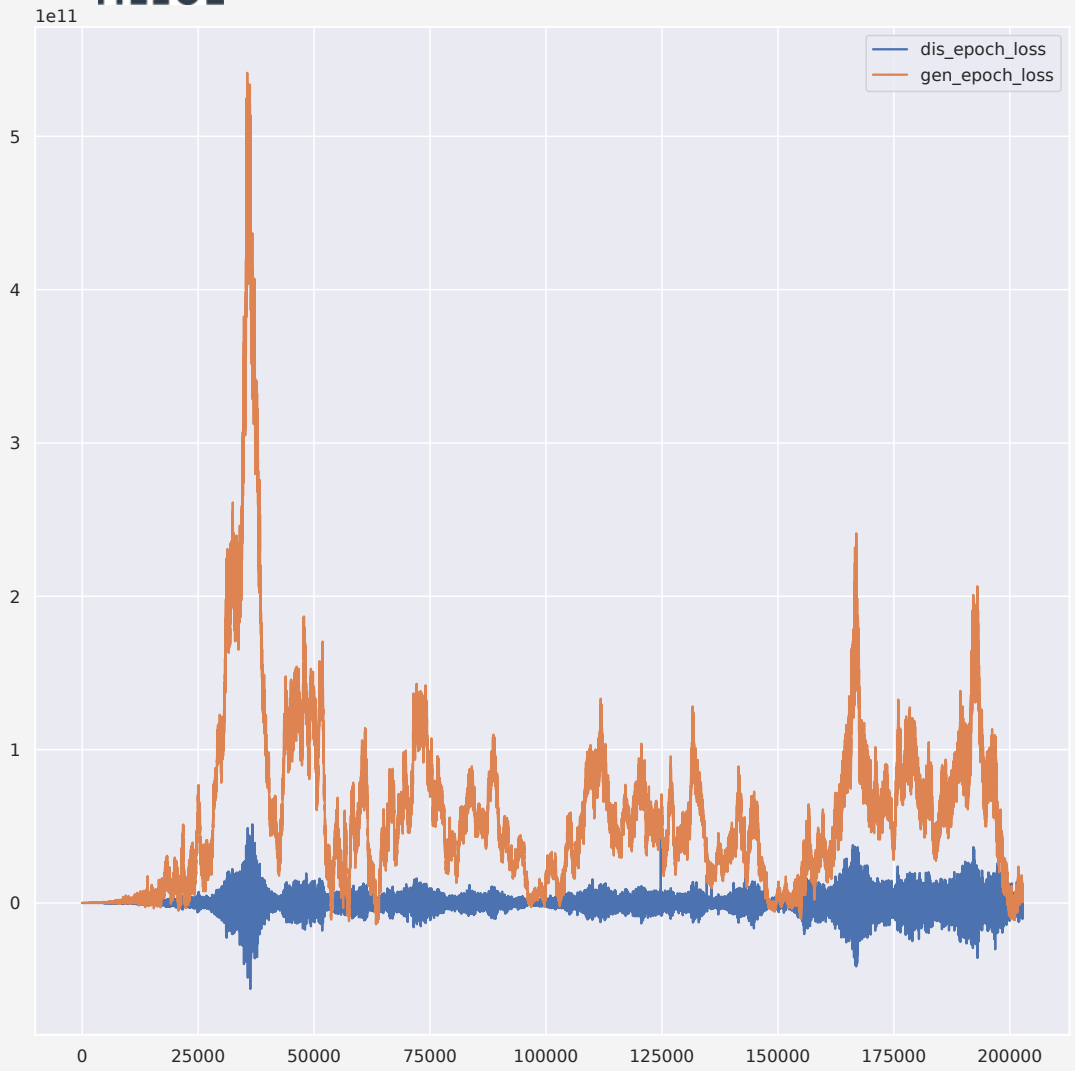
- Additional instance noise [6] was used on the images. Normal distribution with $\sigma = 0.01$.
- Weights were also manually initialized to a random distributions with bias equal to 0.001.
- Tried using small patches of random erasure, but saw no real benefit.
- Was unsure about other data augmentation methods (rotations e.g.) or normalisation since they are coupled to the initial particle origin and energy deposition. Also how these augmentations than relate to image generation.
- Batch normalisation shouldn't be used with WGAN-GP [3], at least in the critic. Layer normalisation [2] was used instead with additional dropout [4].

Deconvolution vs. upscale

- First iterations of the generator used deconvolution[1]. Managed a 50% validation score with this approach.
- Replaced the deconvolution layers with upscale layers and regular convolution layers in the generator. This method can avoid the formation of checker board artefacts.
- Got up to 68% with this approach. Failed to save the exact version that achieved this score, but scored between 50% and my high-score consistently.
- Also tried different padding methods (reflection padding e.g.) in the convolution layers, but observed no notable benefit here.

[1] https://distill.pub/2016/deconv-checkerboard/

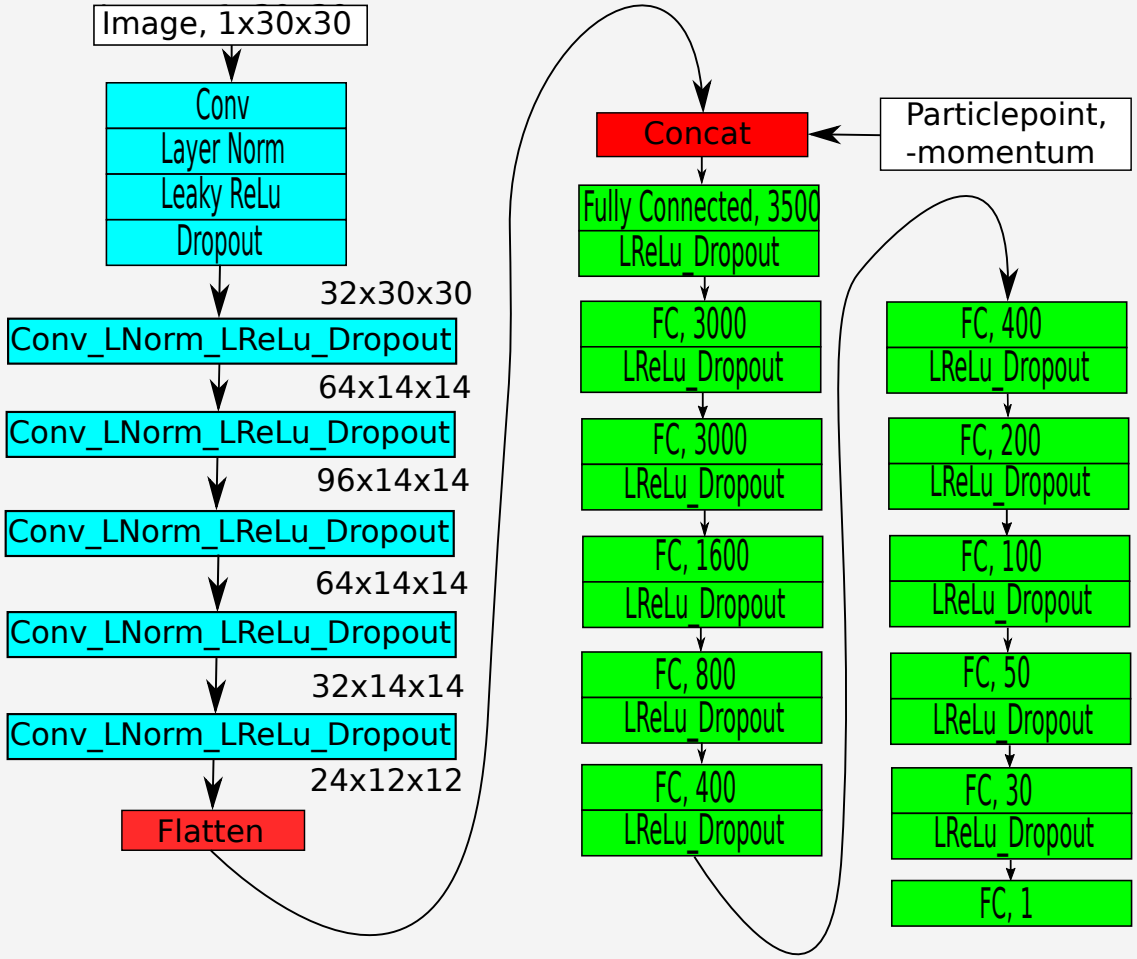

prd_auc
0.0
0.2
0.4
0.6
0.8
0
100
200
300
400
500


1e11
dis_epoch_loss
gen_epoch_loss
5
4
3
2
1
0
0
25000
50000
75000
100000
125000
150000
175000
200000

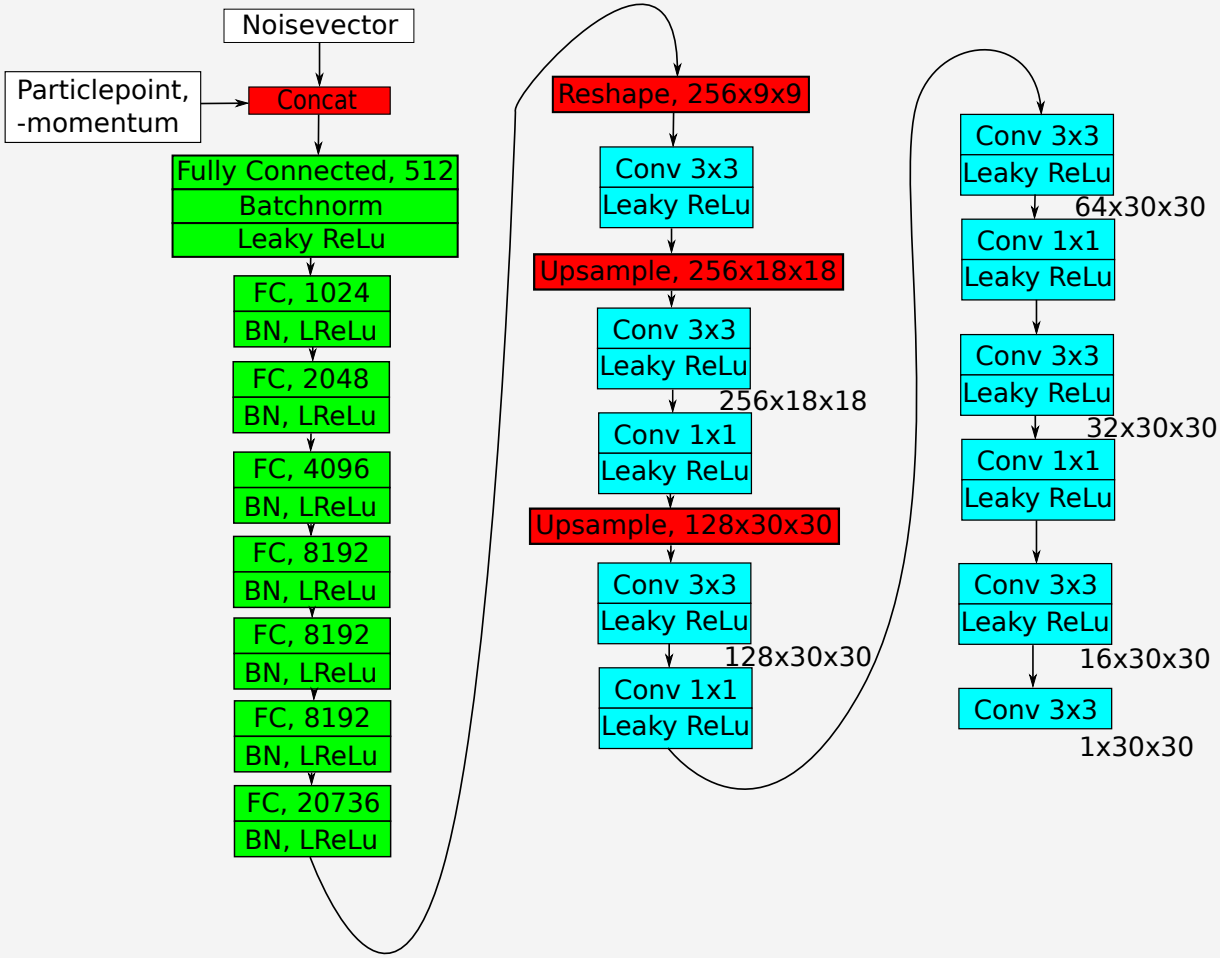
Some remarks

- From experience the prd-auc score is a better metric for validation score than the loss.
- Training with all training samples takes some time, approximately 20 minutes per epoch.
- The previous graphs were from a four week long training, while i was away.
- Training was done on a GTX 1080, the training takes up about 8Gb of VRAM.

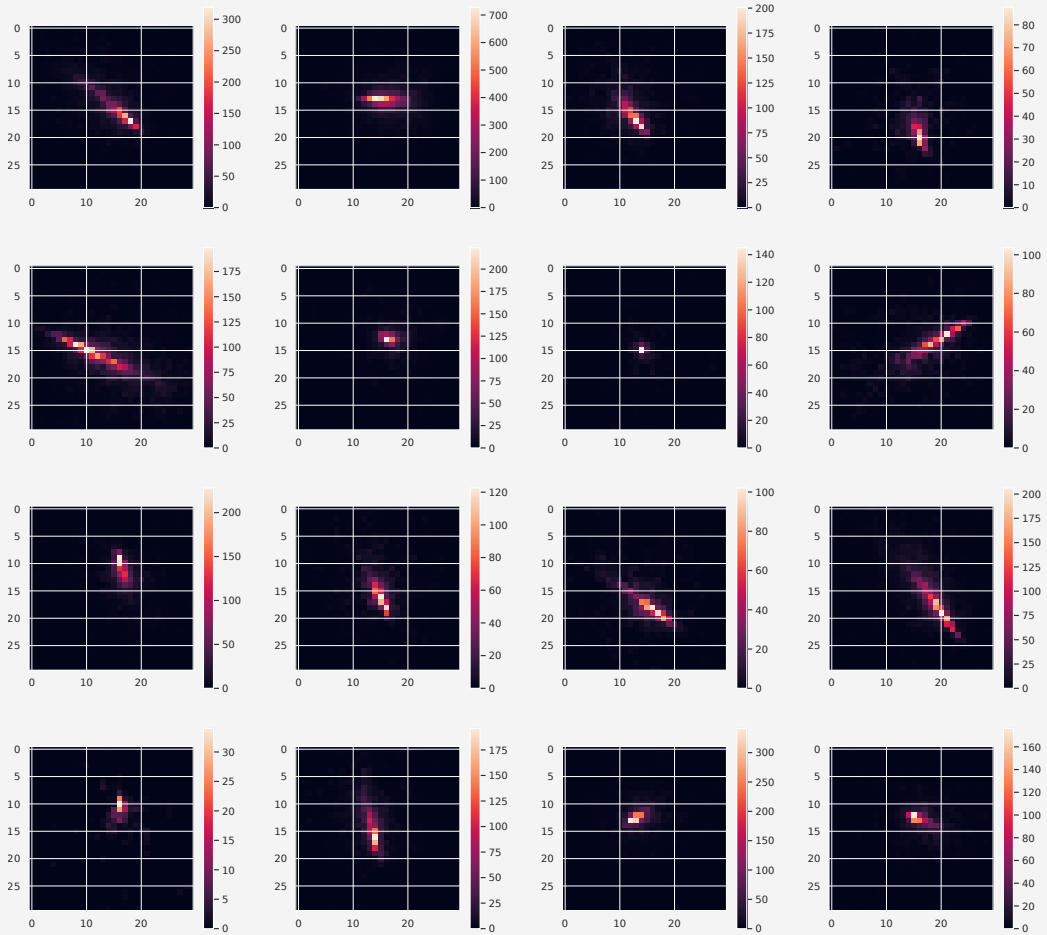
Discriminator



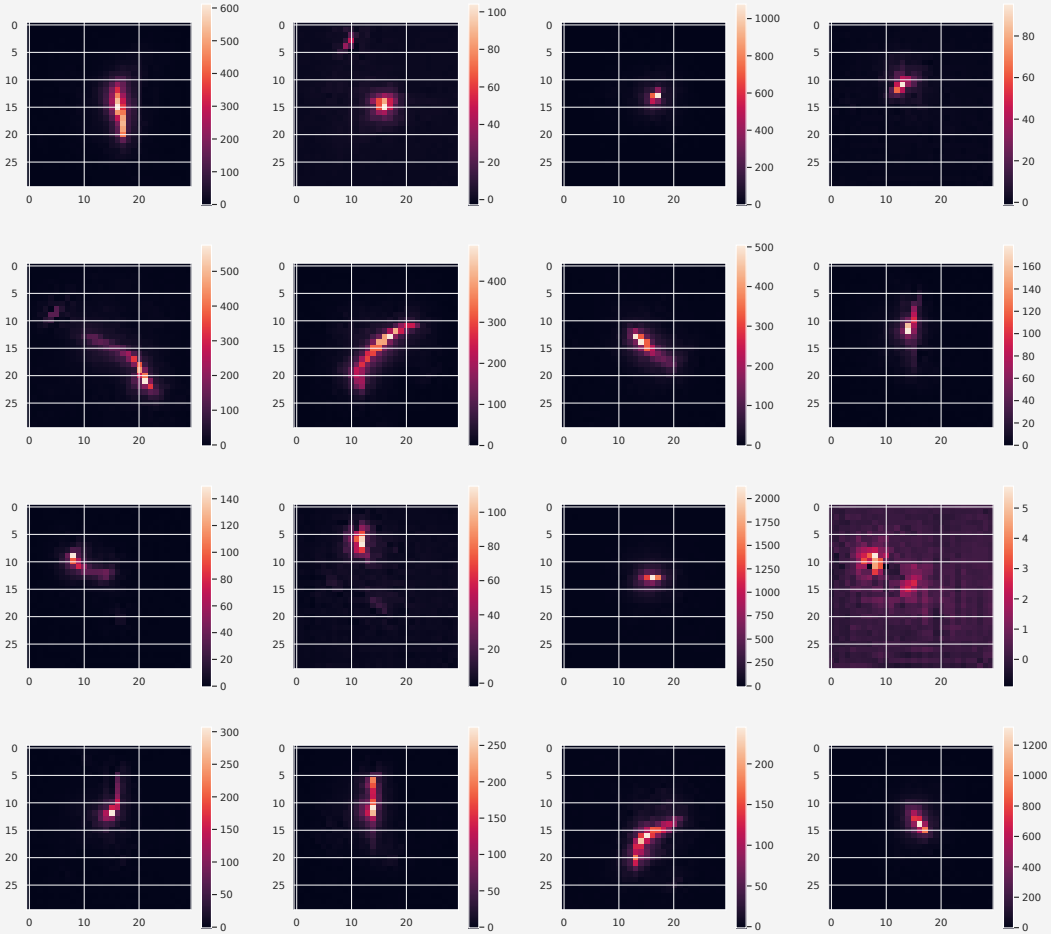
Generator


Noisevector
Particlepoint, -momentum
Concat
Fully Connected, 512
Batchnorm
Leaky ReLu
FC, 1024
BN, LReLu
FC, 2048
BN, LReLu
FC, 4096
BN, LReLu
FC, 8192
BN, LReLu
FC, 8192
BN, LReLu
FC, 8192
BN, LReLu
FC, 20736
BN, LReLu
Reshape, 256x9x9
Conv 3x3
Leaky ReLu
Upsample, 256x18x18
Conv 3x3
Leaky ReLu
256x18x18
Conv 1x1
Leaky ReLu
Upsample, 128x30x30
Conv 3x3
Leaky ReLu
128x30x30
Conv 1x1
Leaky ReLu
Conv 3x3
Leaky ReLu
64x30x30
Conv 1x1
Leaky ReLu
Conv 3x3
Leaky ReLu
32x30x30
Conv 1x1
Leaky ReLu
Conv 3x3
Leaky ReLu
16x30x30
Conv 3x3
1x30x30

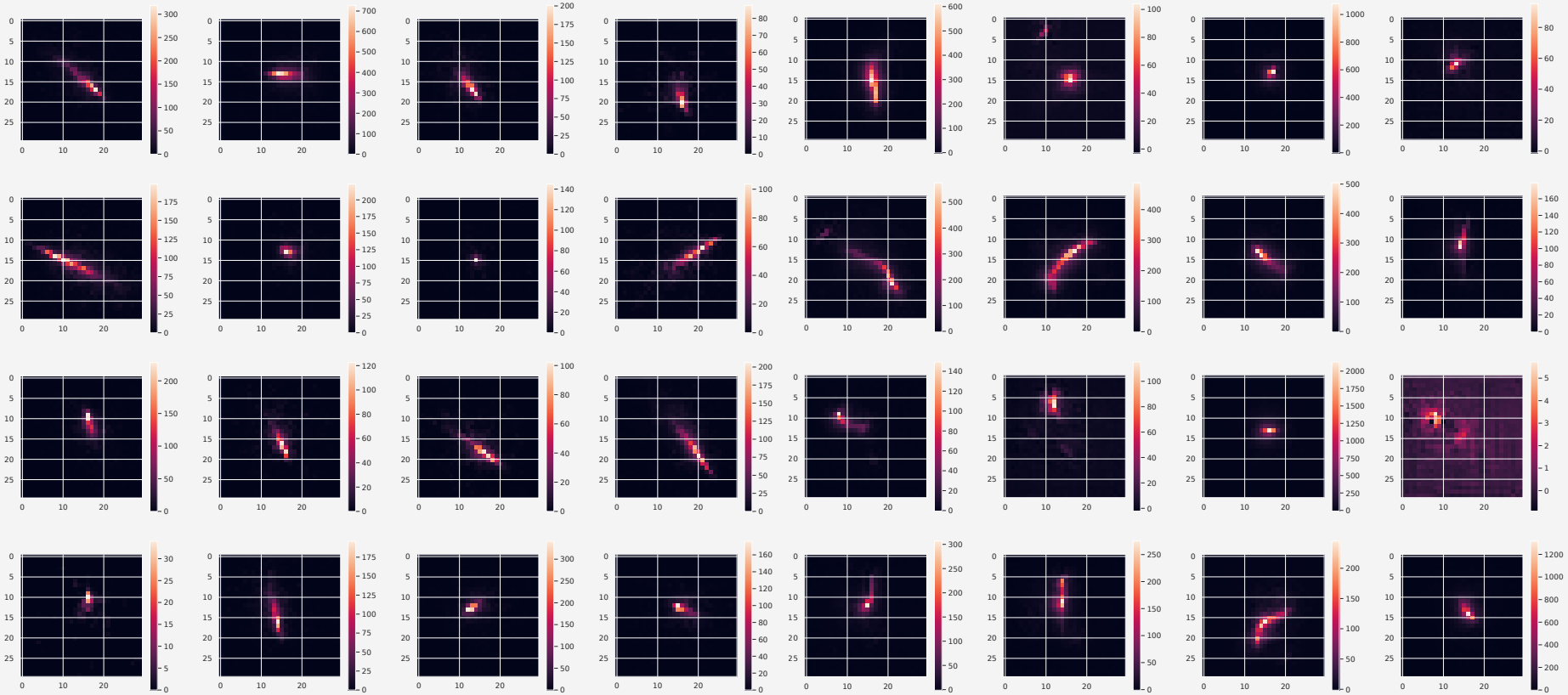
Real energydeposit



Generated energydeposits



Direct comparision



Summary

1. One should be careful about saving iterations of ones model and weights. In case the optimal observed performance comes from a intermediate version.
2. Tried to stick with one approach that worked and pushing it to the maximum.
3. Validation scores between $50 - 65\%$ were achieved.
4. Upscale coupled with convolution layers works better than deconvolution layers.

Bibliography I

Martin Arjovsky, Soumith Chintala, and Léon Bottou.
Wasserstein gan, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton.
Layer normalization, 2016.

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

Bibliography II


Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov.
Improving neural networks by preventing co-adaptation of feature detectors, 2012.


Lars Mescheder, Andreas Geiger, and Sebastian Nowozin.
Which training methods for gans do actually converge?, 2018.


Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár.
Amortised map inference for image super-resolution, 2016.

Bibliography III


T. Tieleman and G. Hinton.
Neural networks for machine learning - lecture 6.5, 2012.

Thank you for your attention!

Notebook can be found here

`https://github.com/jhonerma/MLHEP19_Competition`