1. Style Representation

Using the autoencoder network presented in this work, a natural question to ask is how style is represented in the encoder network. Therefore, a method to invert the networks layer activations is utilized. The activations of style images are inverted with different summary statistics and evaluated.

1.1. Inverting Feature Responses

To visualize, what a specific layer of the encoder network captures, the features at this layer are inverted with a method by Mahendran et al. [MV15].

Their idea is to firstly extract features from the input image $\mathbf{x}$. In a second step a spatial summary statistic is computed on the features to get a description of the image. Finally, a random noise image $\mathbf{n}$ is adjusted to match the summary statistic of $\mathbf{x}$. In this process the characteristics captured by the summary statistic are transferred from the input image to the noise image. The noise image is adjusted by performing gradient descent on it. The summary statistic computed on $\mathbf{x}$ and $\mathbf{n}$ is utilized as loss function. Figure 1.1 schematically shows the architecture of the network and how the feature responses are extracted.

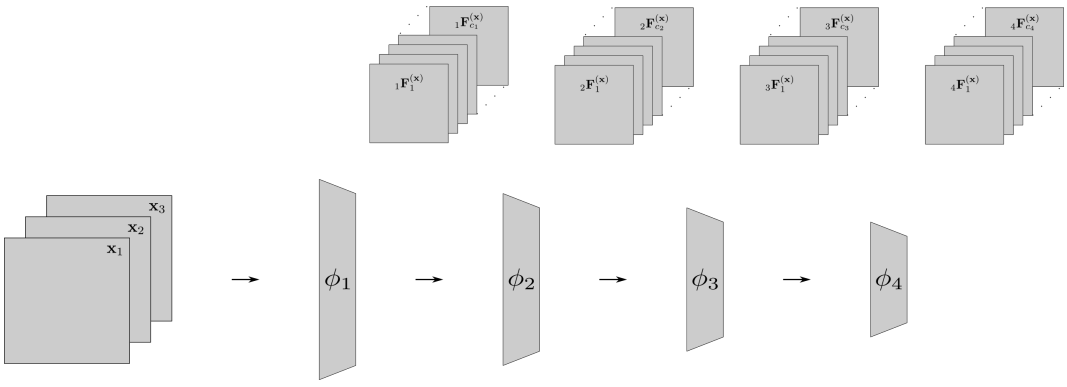

Figure 1.1.: Feature extraction from an encoder network. $\{\phi_i\}_{i=1}^{4}$ represent subsequent sets of network layers.

1.2. Style Loss

A style loss function is a summary statistic used to capture the style of an input image. The noise image is transformed according to this statistic. The choice of the style loss function heavily influences the style that is captured. Figure 1.2 shows a visualization of the loss.

1.2.1. Gram Matrix Loss

In their work on texture synthesis [GEB15] Gatys et al. introduce the first style loss function that is computed on the activations of a CNN. They argue, that a summary statistic that ignores spatial information is given by the correlations between feature maps. The Gram matrix ${}_l\mathbf{G} \in \mathbb{R}^{c_l \times c_l}$ represents the feature correlations at layer l up to a constant of proportionality.

The entry ${}_l\mathbf{G}_{ij}$ of the Gram matrix is defined as the inner product between feature map i and j at layer l.

$$ {}_l\mathbf{G}_{ij}^{(\mathbf{x})} \coloneqq \left\langle {}_l\mathbf{F}_i^{(\mathbf{x})}, {}_l\mathbf{F}_j^{(\mathbf{x})} \right\rangle \tag{1.1} $$

The Gram matrix loss given below measures the element-wise difference between the Gram matrix of each feature map of $\mathbf{s}$ and each corresponding feature map of $\mathbf{n}$.

$$ \mathcal{L}_{\text{Gram}}\left({}_l\mathbf{F}^{(\mathbf{s})}, {}_l\mathbf{F}^{(\mathbf{n})}\right) = \gamma \cdot \sum_{i=1}^{c_l} \sum_{j=1}^{c_l} \left({}_l\mathbf{G}_{ij}^{(\mathbf{s})} - {}_l\mathbf{G}_{ij}^{(\mathbf{n})}\right)^2 \tag{1.2} $$

$$ \gamma = \frac{1}{c^2 \cdot w^2 \cdot h^2} \tag{1.3} $$

1.2.2. Moment Loss

The moment loss is utilized as described in Section **??**.

$$ \mathcal{L}_{\text{mom}}\left({}_l\mathbf{F}^{(\mathbf{s})}, {}_l\mathbf{F}^{(\mathbf{n})}\right) = $$

$$ \frac{1}{c_l} \cdot \sum_{i=1}^{N} \left(\psi\left(\mathbb{M}_i\left({}_l\mathbf{F}^{(\mathbf{s})}\right)\right) \cdot \sum_{j=1}^{c_l} \left(\mathbb{M}_i\left({}_l\mathbf{F}_j^{(\mathbf{s})}\right) - \mathbb{M}_i\left({}_l\mathbf{F}_j^{(\mathbf{n})}\right)\right)^2 \right) \tag{1.4} $$

$$ \psi(\mathbf{x}) \coloneqq \min\left\{1, \frac{1}{\mathbb{M}_1(\mathbf{x})}\right\} \tag{1.5} $$

1.2.3. Maximum Mean Discrepancy Loss

Also, the MMD loss is utilized as described in Section **??**.

$$ \mathcal{L}_{\text{MMD}^2}\left({}_l\mathbf{F}^{(\mathbf{s})}, {}_l\mathbf{F}^{(\mathbf{n})}\right) = \sum_{i=1}^{c_l} \sum_{j=1}^{\lfloor \frac{h_l w_l}{2} \rfloor} h\left(\left({}_l\mathbf{F}_{i,2j-1}^{(\mathbf{s})}, {}_l\mathbf{F}_{i,2j-1}^{(\mathbf{n})}\right), \left({}_l\mathbf{F}_{i,2j}^{(\mathbf{s})}, {}_l\mathbf{F}_{i,2j}^{(\mathbf{n})}\right)\right) \tag{1.6} $$

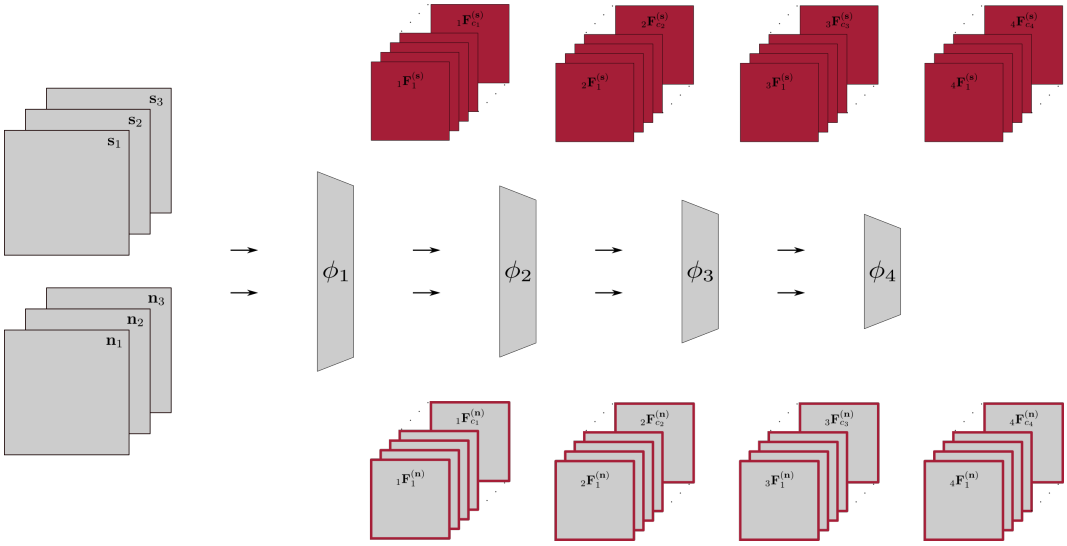

Figure 1.2.: Loss visualization of inverting style features. Parts that are colored red represent where the loss is computed on. Bordered red parts represent which weights are changed by gradient descent. $\{\phi_i\}_{i=1}^{4}$ represent subsequent sets of network layers.

1.3. Training

Every noise image $\mathbf{n}$ is trained by performing gradient descent on its pixels with a particular style loss function.

Each image is optimized with the Adam optimizer [KB15]. Adam is used with a learning rate of $\eta = 10^{-3}$ and $\beta_1 = 0.9, \beta_2 = 0.999$.

To get comparable results, each image starts with the same random seed 0 of the PyTorch framework [PGC+17].

The number of training iterations ι depends on the loss used as well as on the layer where features are inverted from. Table 1.1 lists the number of iterations.

l	$\mathcal{L}_{\text{Gram}}$	$\mathcal{L}_{\text{MMD}^2}$	$\mathcal{L}_{\text{mom}}$ $(N=1)$	$\mathcal{L}_{\text{mom}}$ $(N=2)$	$\mathcal{L}_{\text{mom}}$ $(N=3)$	$\mathcal{L}_{\text{mom}}$ $(N=4)$
`relu_1_1`	$100,000$	$100,000$	$20,000$	$40,000$	$60,000$	$100,000$
`relu_1_2`	$150,000$	$150,000$	$40,000$	$80,000$	$120,000$	$200,000$
`relu_2_1`	$200,000$	$200,000$	$60,000$	$120,000$	$180,000$	$300,000$
`relu_2_2`	$250,000$	$250,000$	$80,000$	$160,000$	$240,000$	$400,000$

Table 1.1.: Number of iterations ι the style representation visualizations are trained for.

Additionally, the optimization is stopped when the loss value defined by the style loss drops below $\theta = 10^{-5}$ since in this case the images already have the desired structure and are stylized sufficiently. If in the final iteration, the loss still is larger than θ each image is trained for another 1000 iterations. If necessary, this step is repeated up to 240 times which results in the maximal number of iterations $\iota_{\max} = \iota + 240 \cdot 1000$.

1.4. Experiments

Figure 1.3 shows the result images produced with Gram matrix loss. While the first layers of the networks capture the color of the input image, deeper layers focus in greater extend on the actual style patterns and structures. In particular, the deeper the layer the features are extracted from, the larger the style structures tend to get. This is explained by the size of the receptive fields that grows with the depth of the network.

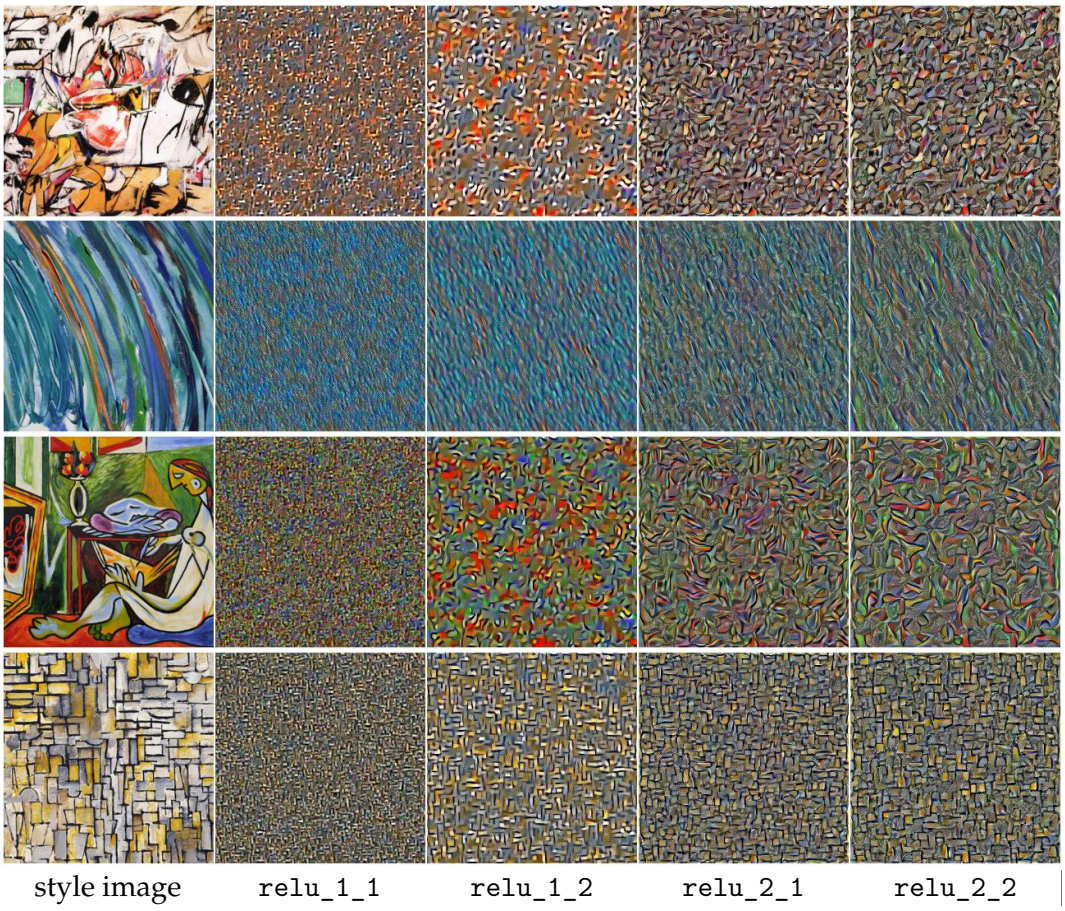

Figure 1.3.: Visualization of style images passing through a VGG network with Gram matrix loss.

Figure A.1, Figure A.2 and Figure A.3 show images with $N \in \{1, 2, 3, 4\}$ moments aligned at layers $l \in R$. There clearly is an improvement in style by aligning more moments. In particular, the improvement in style is given by the increasing size of the style structures and the degree of similarity between the colormaps of the images.

$$R = \{\texttt{relu\_1\_1}, \texttt{relu\_1\_2}, \texttt{relu\_2\_1}, \texttt{relu\_2\_2}, \} \tag{1.7}$$

Comparing the Gram matrix loss to the moment loss as visualized in the figures, it can be detected that the moment loss captures slightly larger style structures by approximately the same loss magnitude. Also, the colormaps of the style images seem to be better preserved by using the moment loss.

Empirically, the Gram matrix loss converges with about the same number of iterations as the moment loss for $N = 2$. Additionally, the Gram matrix loss converges almost every time to a good minimum which is visible by an artefact-free result image. In contrary, the moment loss for $N > 2$ moments often has difficulties in finding a good minimum and converges towards a local minimum which is noisy and does not reflect the style structure appropriately. Furthermore, for $N > 2$ the moment loss needs a larger number of iterations than the Gram matrix loss. Both can be explained by the much higher complexity of the moment loss function with each additional moment that is minimized.

Besides the increased time the generation task takes, the number and size of artifacts which stay in the produced images also increase with the number of moments aligned. On the one hand, this can be explained by the increasing complexity of the loss function as stated above. On the other hand, the artifacts also depend on the initial random seed used. By changing the seed, the artifacts appear with different sizes and at different locations. Thus, one can conclude that the loss landscape of the moment loss prevents the Adam optimizer from finding the optimal solution. Using an optimizer that also uses the second derivative like L-BFGS [MN11], does not bring any benefit but instead produces worse images.

Figure 1.7 shows result images produced with MMD loss. Evidently, the loss only reconstructs the style image and does not capture any style structure. This is explained by the fact that the MMD loss minimizes the error between all moments of the noise feature map and the style feature map. Since all moments of the noise feature map are aligned to those of the style feature map the style image gets reconstructed.

Note that in other works that actually manage to capture the style structure with a MMD loss like [LWJH17], the loss is defined in a different way. In those works, the MMD loss aligns the distributions of feature maps element-wise for every element between all the feature maps. It would not fit into the concept of this work to align the distributions of all feature maps element-wise since the goal is to align corresponding pairs of feature maps.

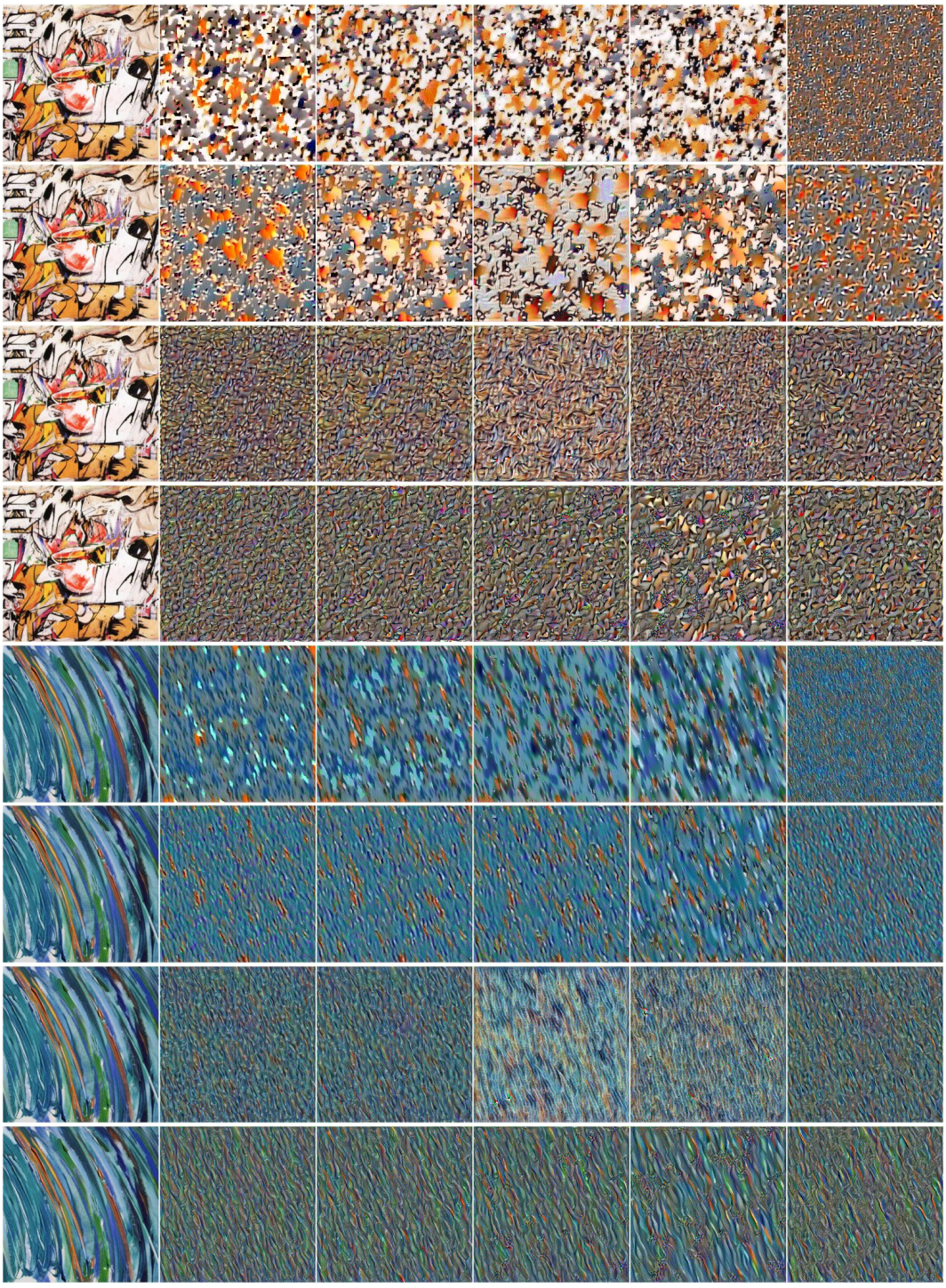

Figure 1.4.: Visualization of style images at layer `relu_1_1`, `relu_1_2`, `relu_2_1`, `relu_2_2` from top to bottom. Note how style structures get larger by aligning more moments.

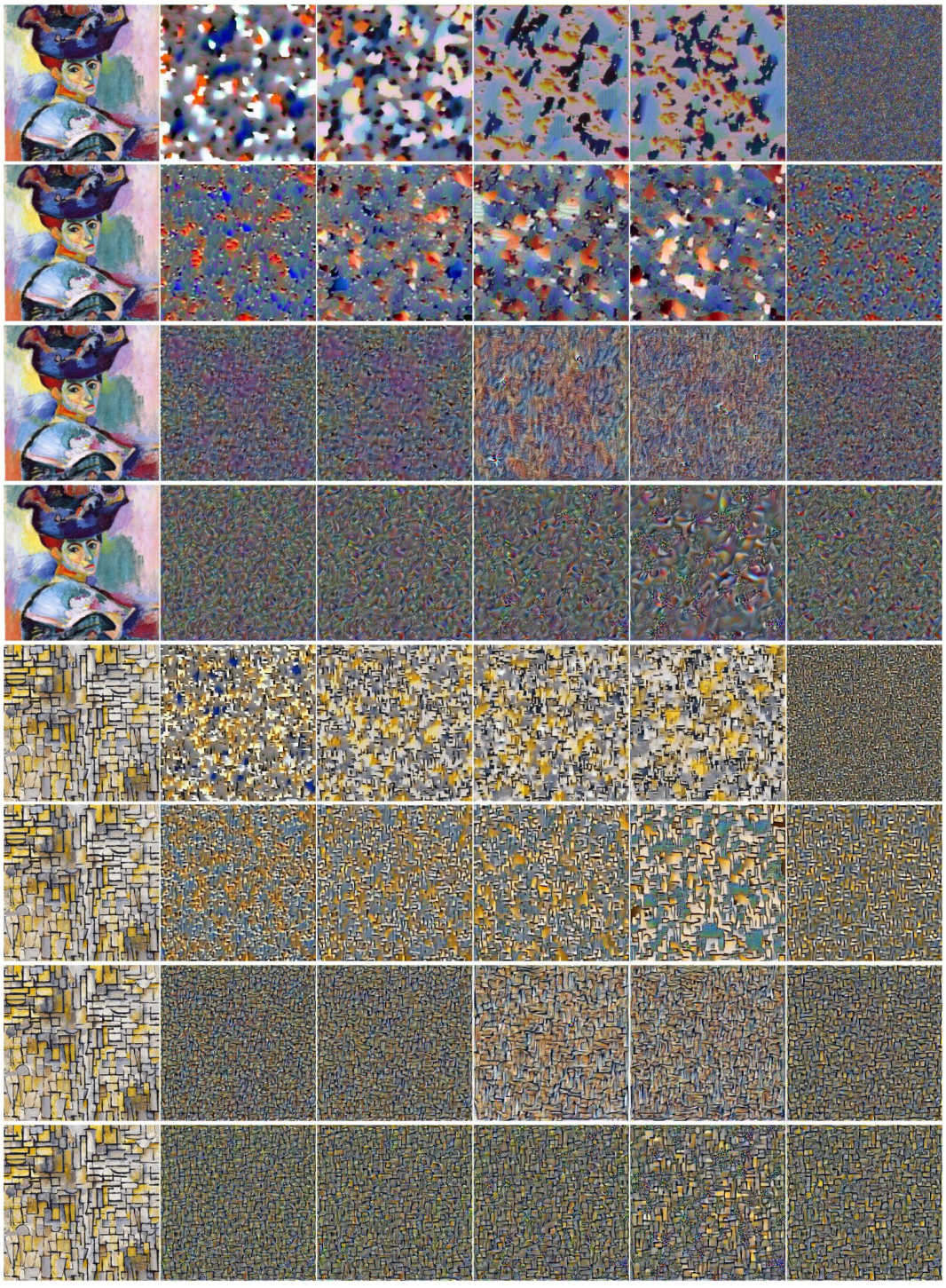

Figure 1.5.: Visualization of style images at layer `relu_1_1`, `relu_1_2`, `relu_2_1`, `relu_2_2` from top to bottom. Note how style structures get larger by aligning more moments.

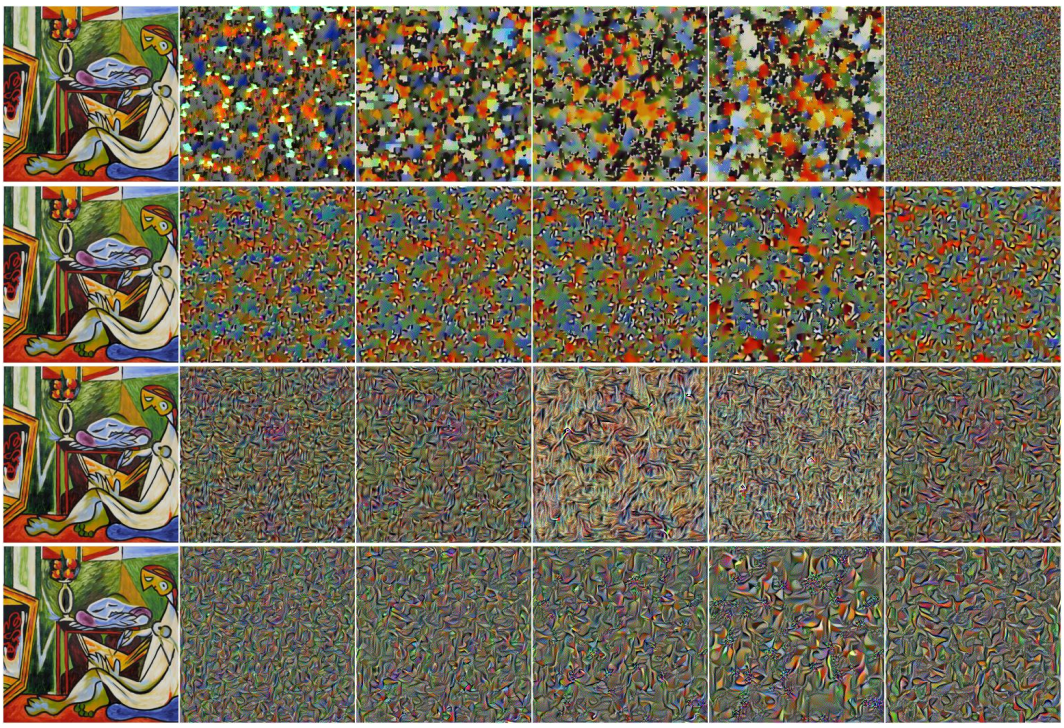

Figure 1.6.: Visualization of style images at layer `relu_1_1`, `relu_1_2`, `relu_2_1`, `relu_2_2` from top to bottom. Note how style structures get larger by aligning more moments.

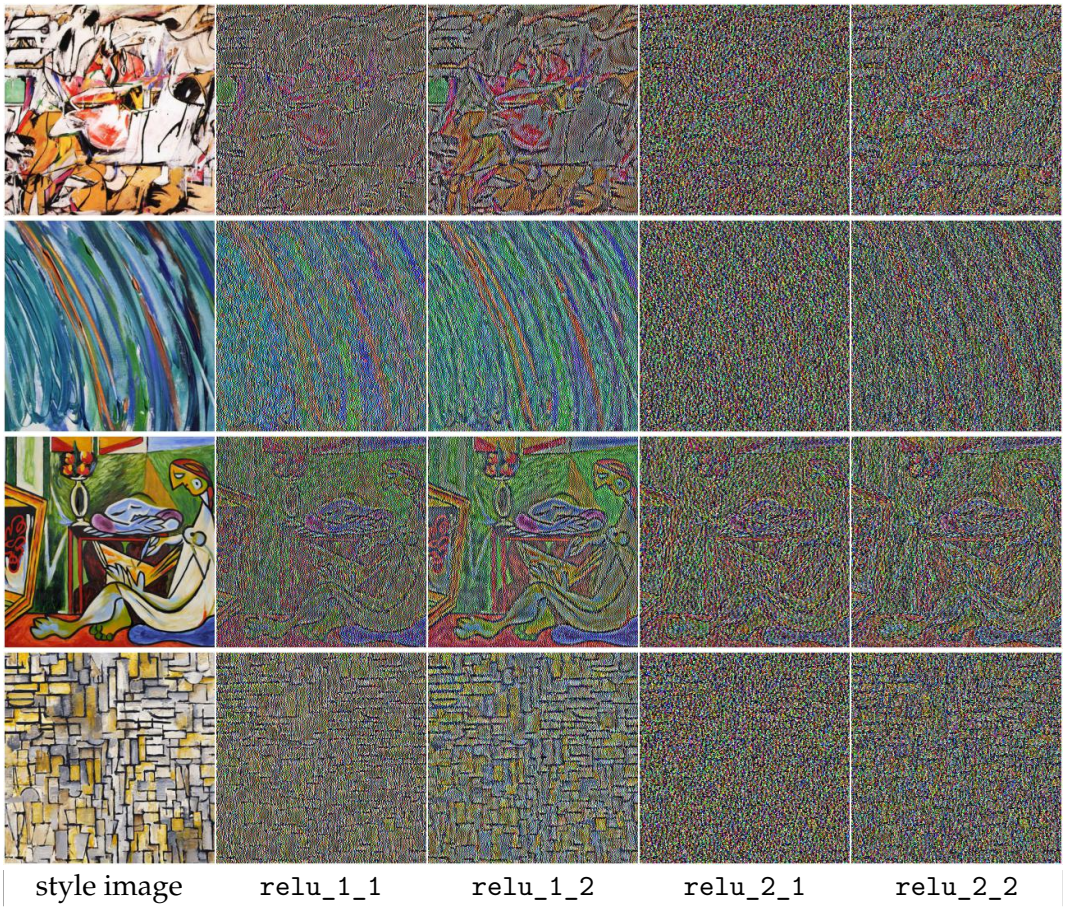

Figure 1.7.: Visualization of style images passing through a VGG network with MMD loss.

A. Style Representation

This appendix gives further information on style representation presented in Chapter 1.

A.1. Number of Training Iterations

A.2. More Image Examples

Figure A.1, Figure A.2 and Figure A.3 show style images passing through a VGG network. From top to bottom the images show the responses of different style losses at layers `relu_1_1`, `relu_1_2`, `relu_2_1` and `relu_2_2`.

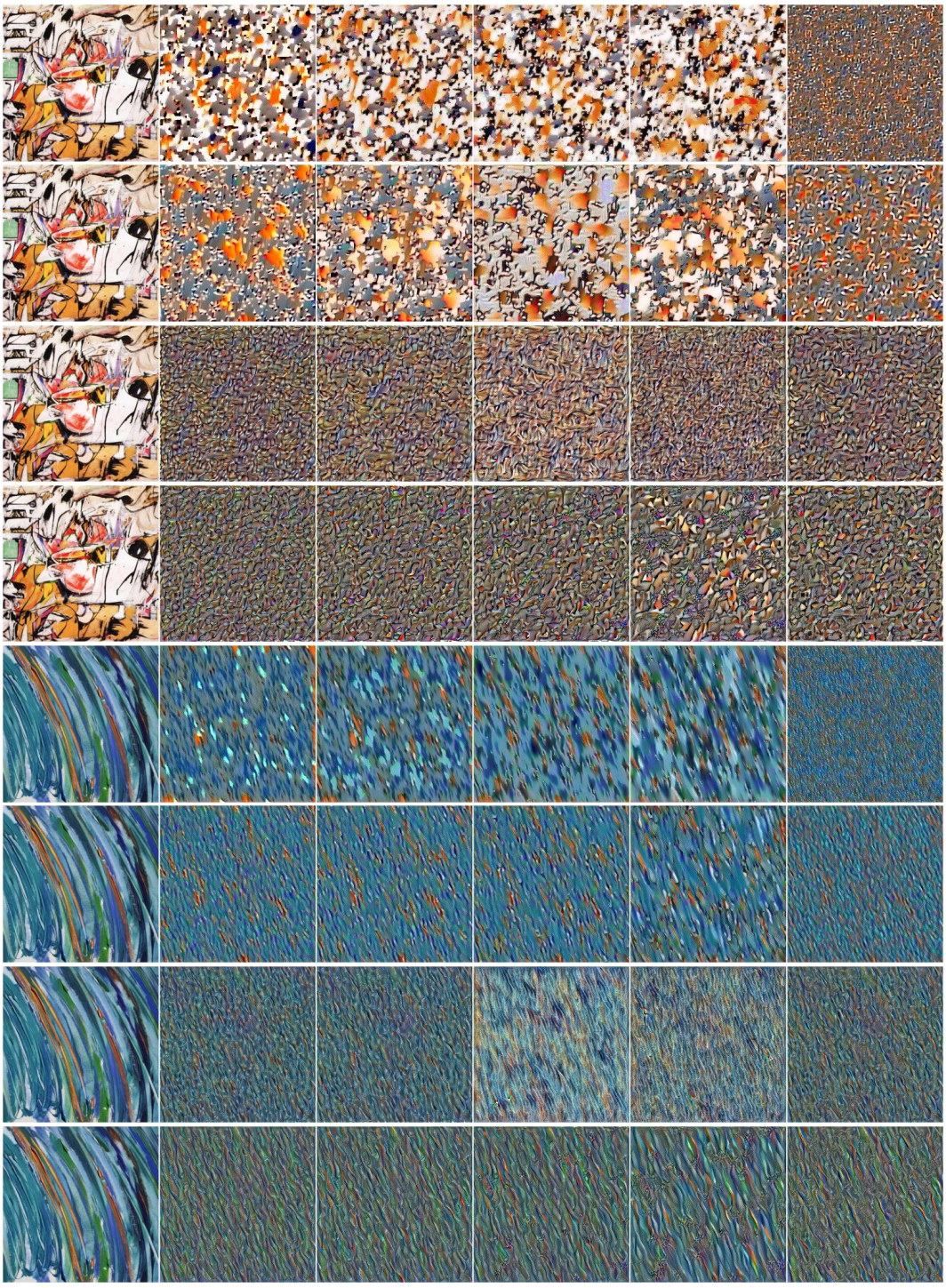

Figure A.1.: Visualization of style images at layer `relu_1_1`, `relu_1_2`, `relu_2_1`, `relu_2_2` from top to bottom.

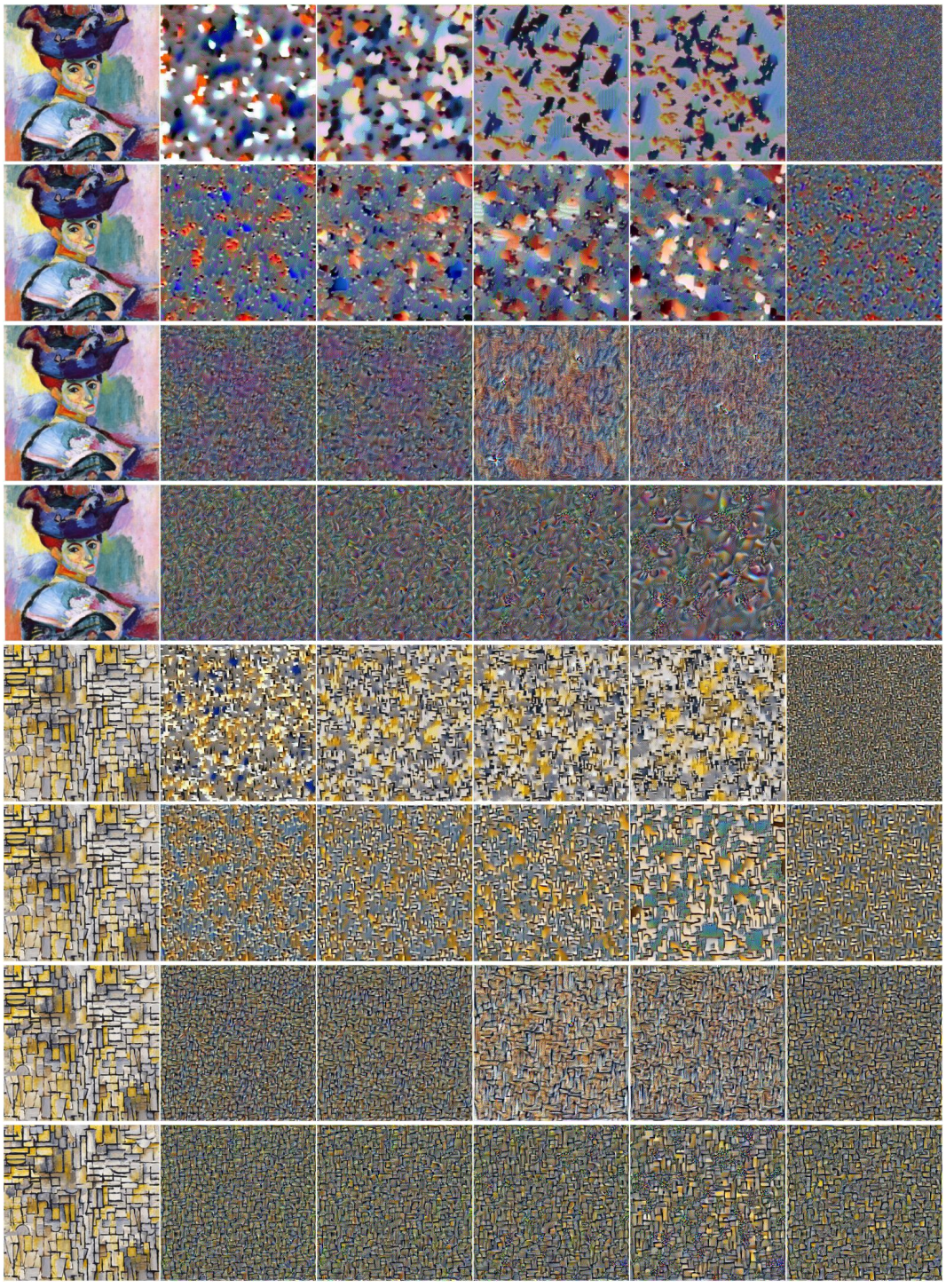

Figure A.2.: Visualization of style images at layer `relu_1_1`, `relu_1_2`, `relu_2_1`, `relu_2_2` from top to bottom.

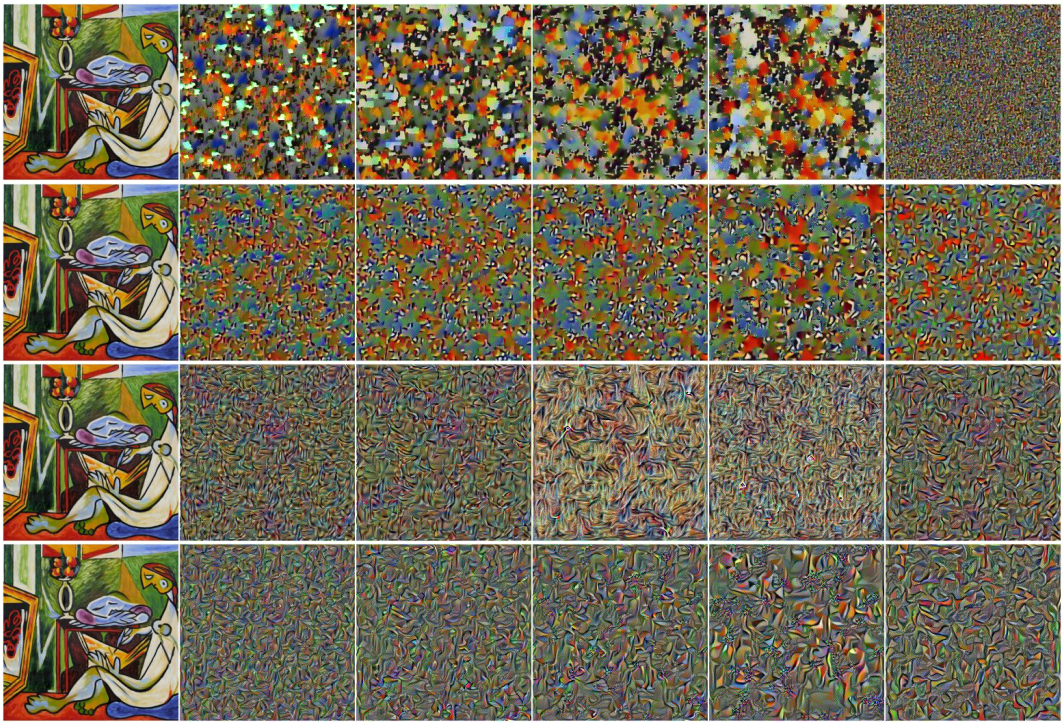

Figure A.3.: Visualization of style images at layer `relu_1_1`, `relu_1_2`, `relu_2_1`, `relu_2_2` from top to bottom.

Bibliography

[GEB15] L. Gatys, A. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2015.

[KB15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[LWJH17] Y. Li, N. Wang, J.Liu, and X. Hou. Demystifying neural style transfer. In *International Joint Conference on Artificial Intelligence*, 2017.

[MN11] J. Morales and J. Nocedal. Remark on "algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound constrained optimization". *Association for Computing Machinery Trans. Math. Softw.*, 2011.

[MV15] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[PGC+17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, and L. Antigaand A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.