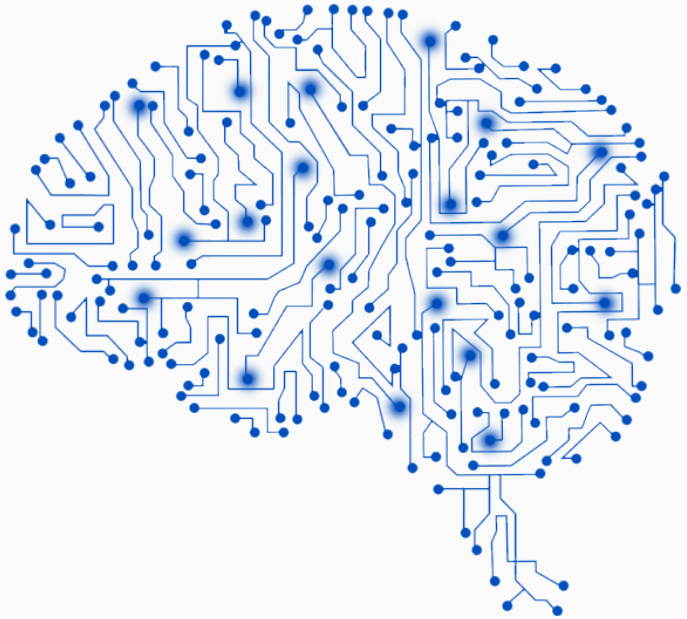
Master 2 Statistique & Data Science, Ingénierie Mathématique

Réseaux de neurones profonds pour l'apprentissage
Deep neural networks for machine learning

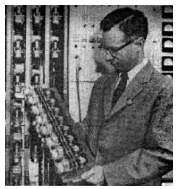
Course VI – Introduction to Artificial Neural Networks: Texture Synthesis and Style Transfer

Bruno Galerne

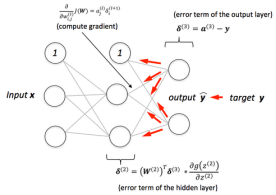
2023-2024



Timeline of (deep) learning



1958 Perceptron



1974 Backpropagation

Convolution Neural Networks for Handwritten Recognition

1998

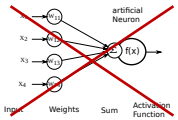
Google Brain Project on 16k Cores

2012

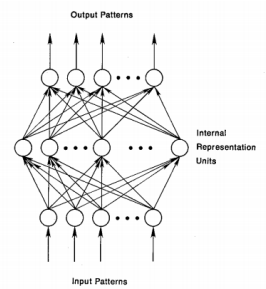
awkward silence (AI winter)

1969 Perceptrons book

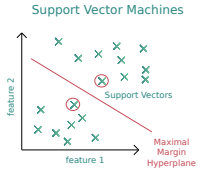
Perceptron criticized



~1980 Multilayer network



1995 SVM reigns



2006 Restricted Boltzmann Machine

2012 AlexNet wins ImageNet

IMAGENET

(Source: Lucas Masuch & Vincent Lepetit)

Texture synthesis and style transfer

What is a texture?

A minimal definition of a **texture** image is an "image containing repeated patterns" [Wei et al., 2009].

The family of patterns reflects a certain amount of randomness, depending on the nature of the texture.

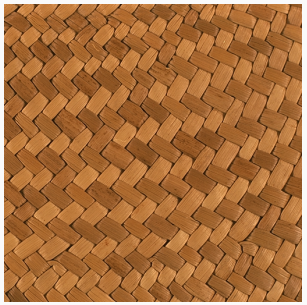
Two main subclasses:

- The **micro-textures**.



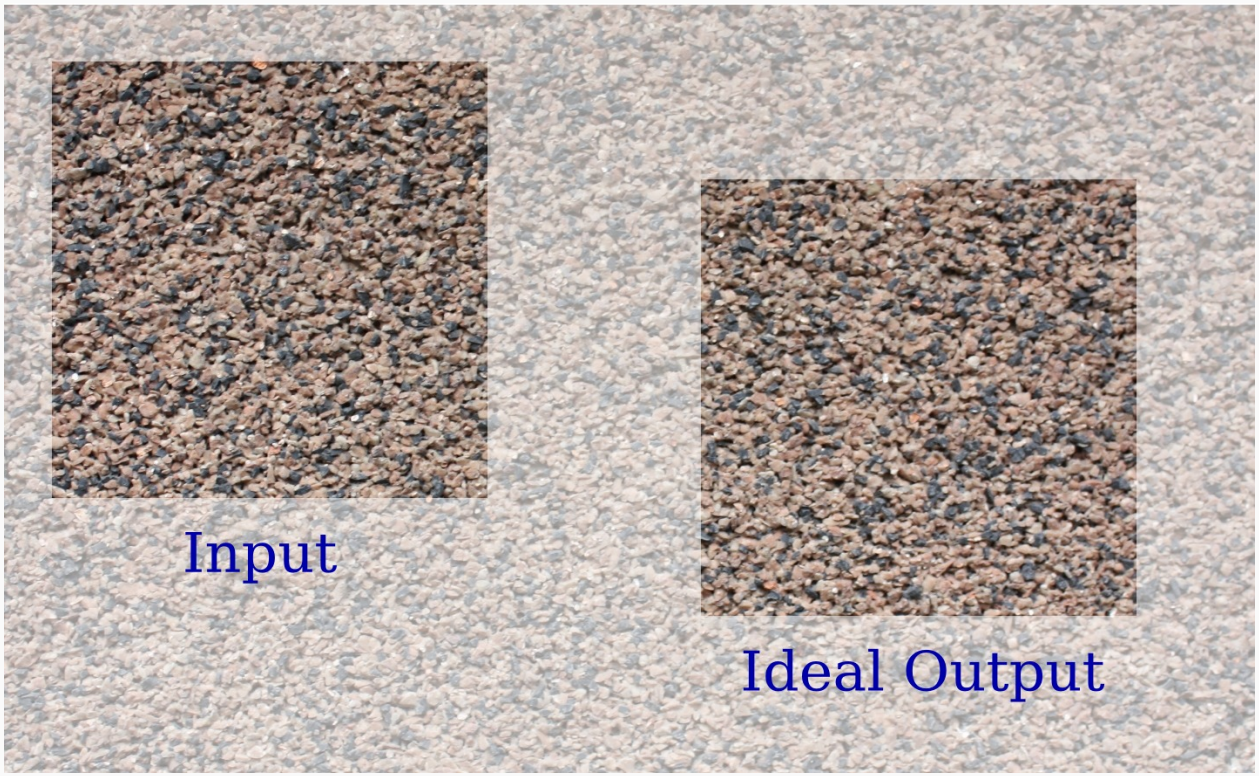
- The **macro-textures**, constitued of small but discernible objects.





Texture synthesis

Texture Synthesis: Given an input texture image, produce an output texture image being both visually similar to and pixel-wise different from the input texture.



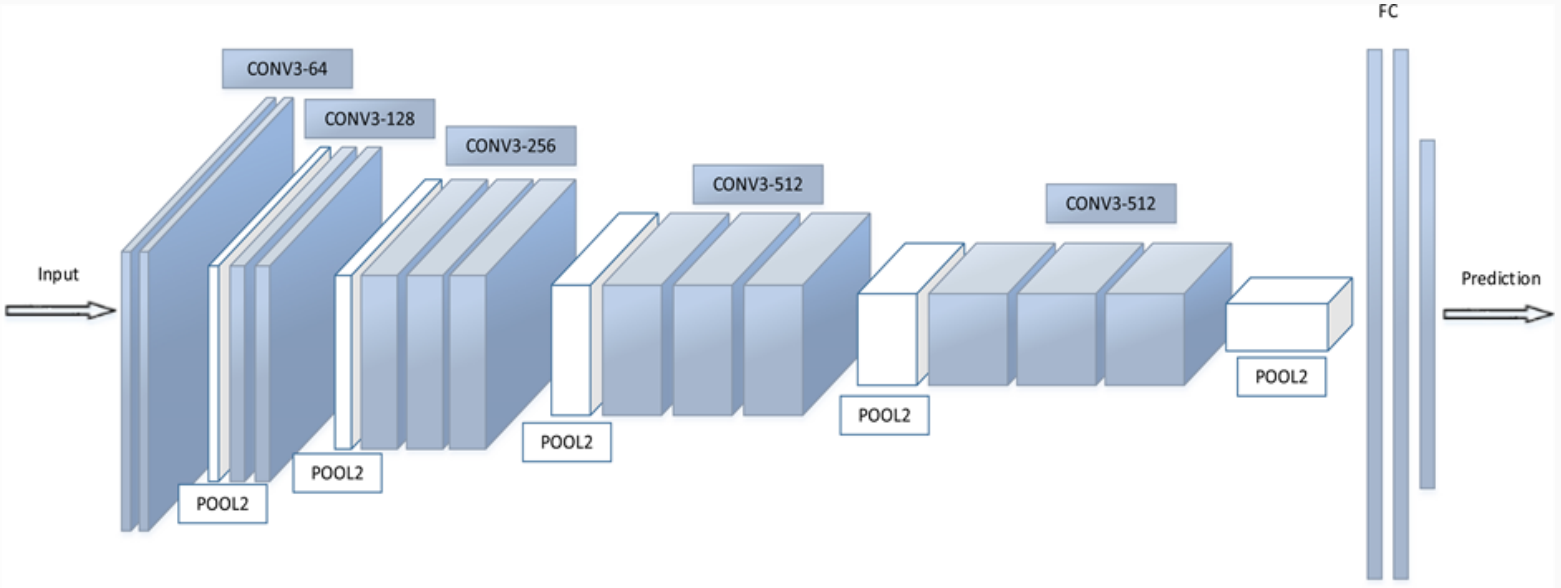
The output image should ideally be perceived as another part of the same large piece of homogeneous material the input texture is taken from.

Gatys et al algorithm

References: L. Gatys, A. S. Ecker, and M. Bethge, *Texture synthesis using convolutional neural networks*, in Advances in Neural Information Processing Systems, 2015

Convolutional Neural Networks (CNN)

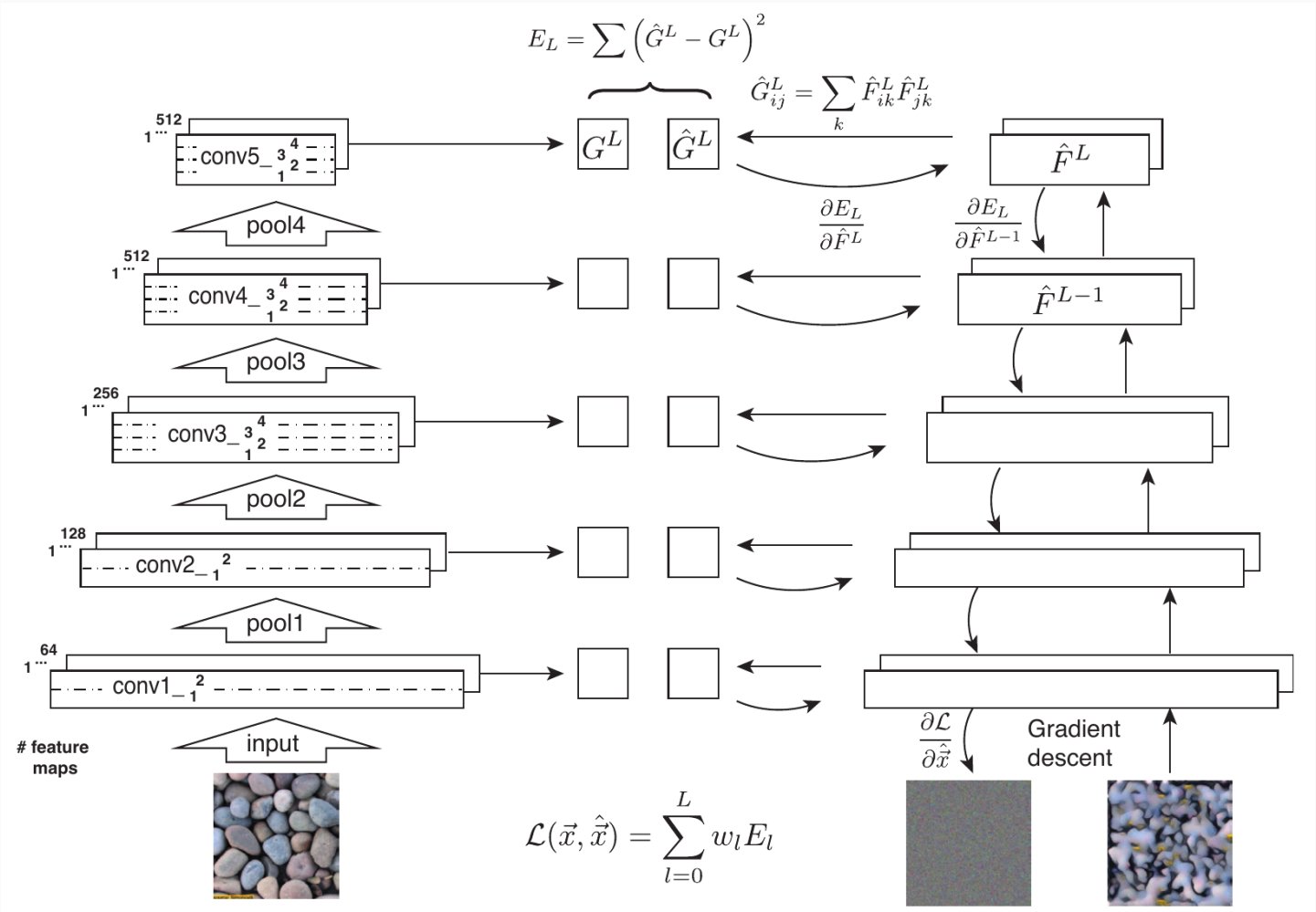
- Main idea: Use the feature layers of a trained deep CNN, namely VGG 19 [Simonyan and Zisserman, 2015], as statistics.
- VGG 19 was trained for image classification.
- It only uses 3×3 convolution kernels followed by RELU ($=$ positive part) and max-pooling.



- For texture analysis, we only use the "pool" layers.
- We do not use the last "fully connected" layers that perform classification.
- VGG 19 is understood as a multiscale nonlinear transform adapted to natural images.

Gatys et al algorithm

- Each feature layer is spatially averaged and the Gram matrix is formed for each layer.
- Texture synthesis consists in minimizing the sum of the Frobenius norm between Gram matrices.



Gatys et al algorithm

Given an example image u and a random initialization x_0, one optimizes the loss function

$$E(x) = \sum_{\text{for selected layers } L} w_L \left\| G^L(x) - G^L(u) \right\|_F^2$$

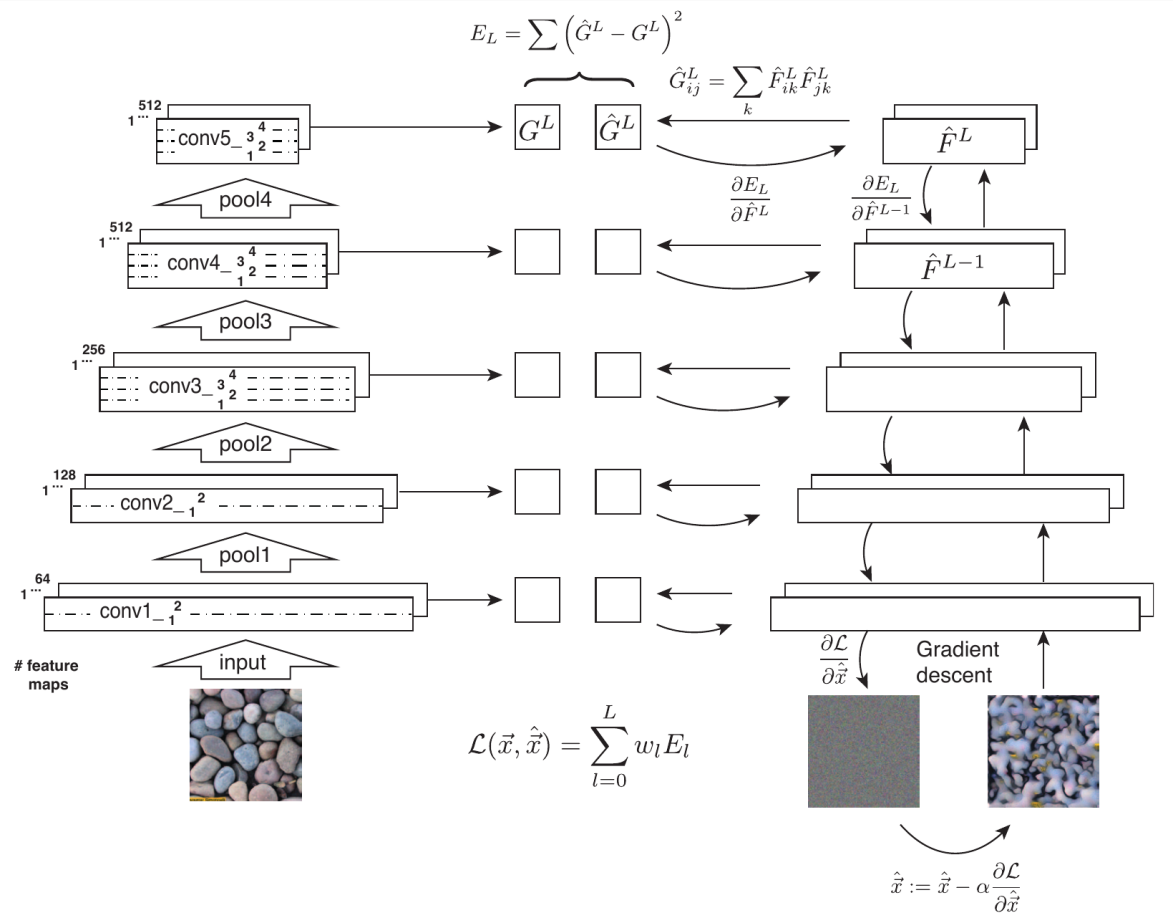
where

- w_L is a weight parameter for each layer
- $\| \cdot \|_F$ is the Frobenius norm
- for an image y and a layer index L, $G^L(y)$ denotes the **Gram matrix** of the VGG-19 features at layer L: if $V^L(y)$ is the feature response of y at layer L that has spatial size $w \times h$ and n channels,

$$G^L(y) = \frac{1}{wh} \sum_{k \in \{0,\dots,w-1\} \times \{0,\dots,h-1\}} V^L(y)_k V^L(y)_k^T \in \mathbb{R}^{n \times n}.$$

The Gram matrix is a spatial statistics of order 2 that contains mean and covariance information.

Gatys et al algorithm



- The gradient of the energy is computed using back-propagation routines.
- The authors use a quasi-Newton algorithm: L-BFGS that stands for Limited-memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm.
- It consists in using a low-rank approximation of the Hessian matrix for computing the descent direction.

Gatys et al algorithm

Given an example image u and a random initialization x_0, one optimizes the loss function

$$E(x) = \sum_{\text{for selected layers } L} w_L \left\| G^L(x) - G^L(u) \right\|_F^2$$

Pseudo-code:

Input: Input image u, set of selected VGG-19 layers $\mathcal{L}$ and associated layer weights parameter $\{w_L,\ L \in \mathcal{L}\}$

Output: Synthesized texture x

Apply VGG-19 to u and extract the layers $\{V_L(u)),\ L \in \mathcal{L}\}$

Compute the target Gram matrices $\{G_L(u) = \mathrm{Gram}(V_L(u)),\ L \in \mathcal{L}\}$.

Initialize x with some Gaussian white noise.

for N_{it} iterations **do**

Apply VGG-19 to x and extract the layers $\{V_L(x)),\ L \in \mathcal{L}\}$.

Compute the current Gram matrices of x:

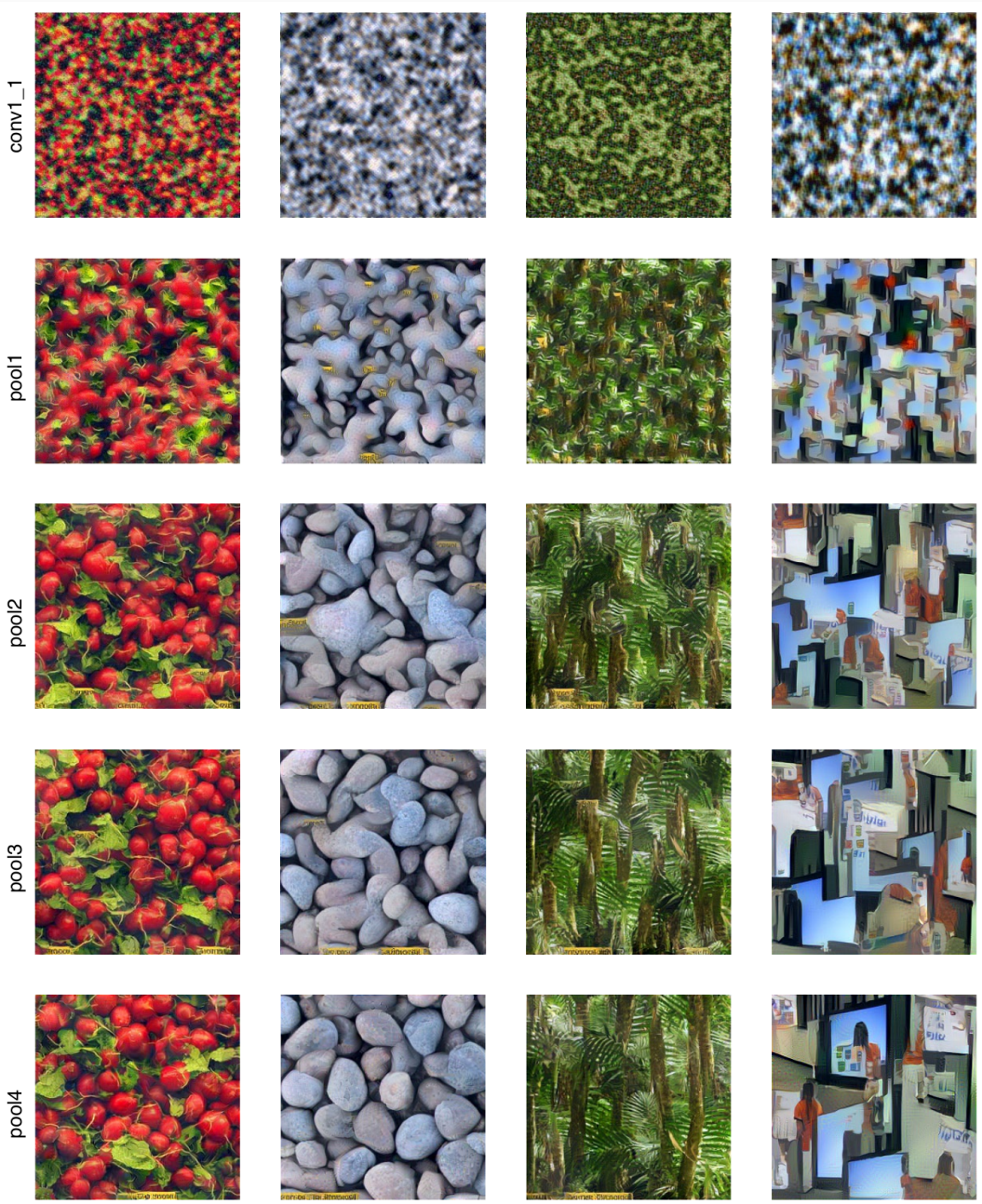
$\{G_L(x) = \mathrm{Gram}(V_L(x)),\ L \in \mathcal{L}\}$

Compute the loss $E(x)$ and its gradient $\nabla_x E(x)$ using backpropagation.

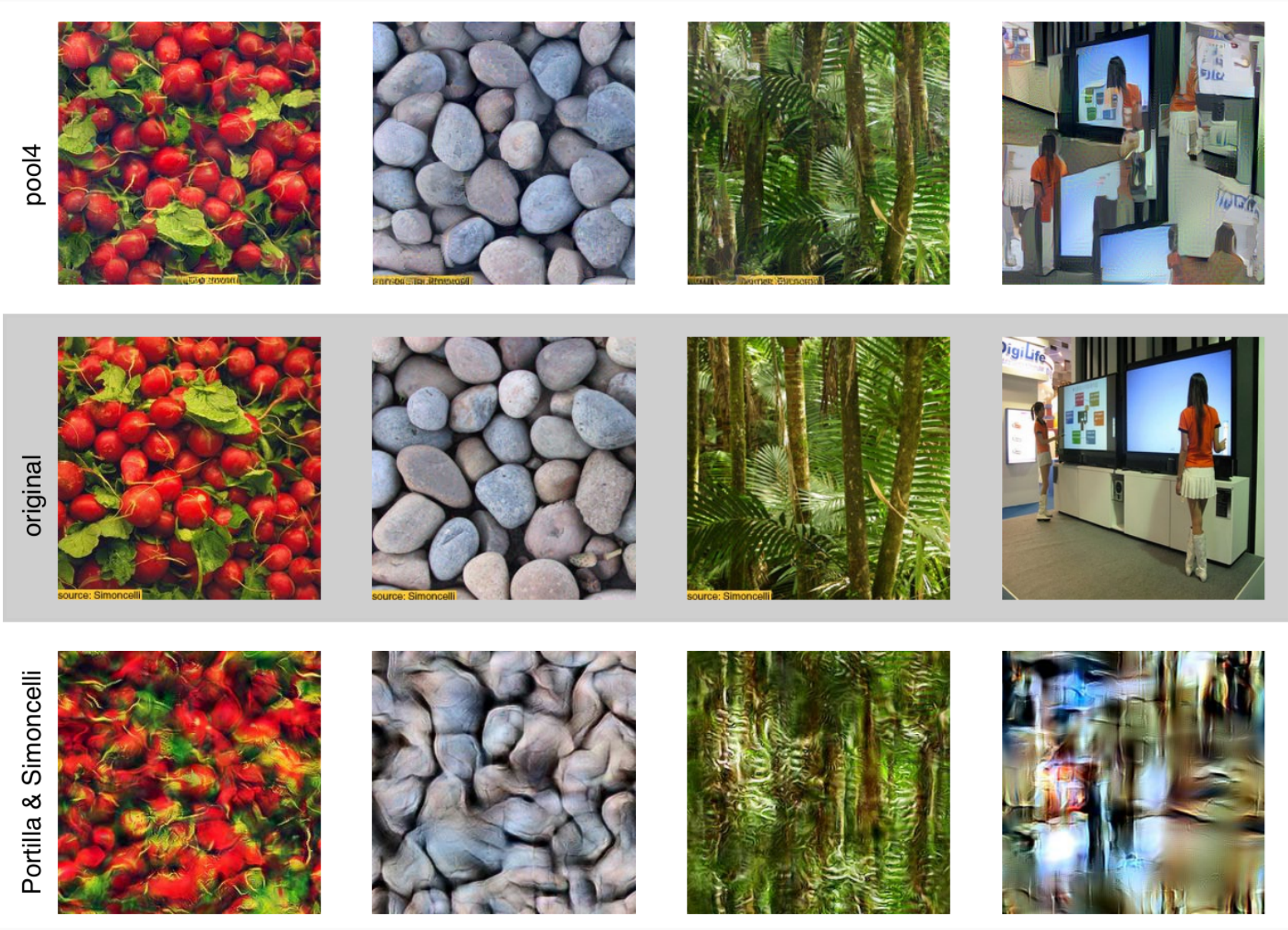
$x \leftarrow \text{L-BFGS-step}(x, E(x), \nabla_x E(x))$.

end for

Gatys et al algorithm: Depth influence



Gatys et al algorithm: Results



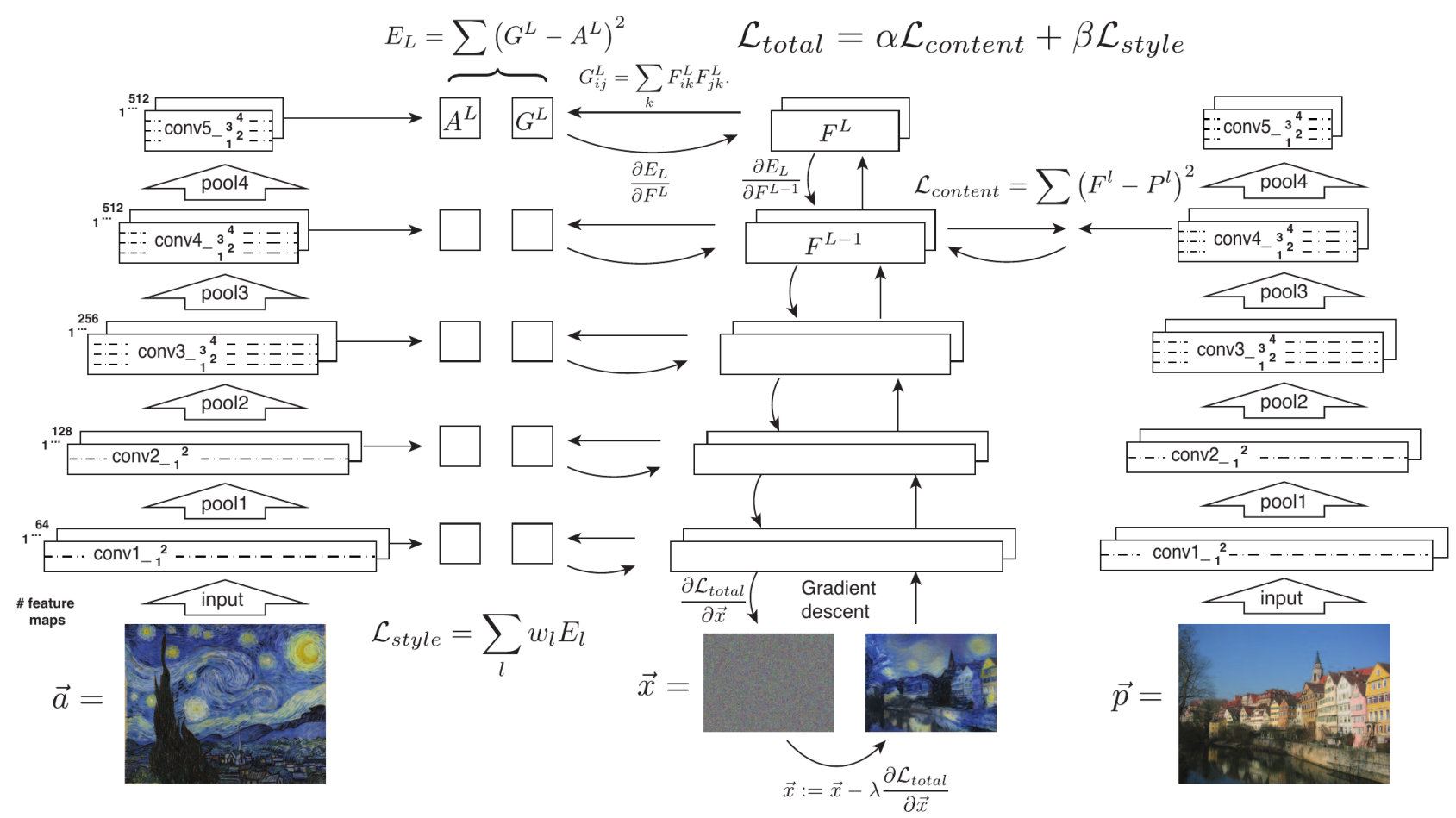
- MORE RESULTS

Gatys et al algorithm

- This algorithm is the current state of the art.
- The computational cost is really high (even with high-end GPUs it takes minutes).
- A lot of improvements have been proposed, eg by adding term to the energy or by adding correlation between layers.
- Extension for style transfer with equally impressive results, and maybe more impact.

Gatys et al for style transfer

Reference: [Gatys et al., 2016]



Gatys et al for style transfer

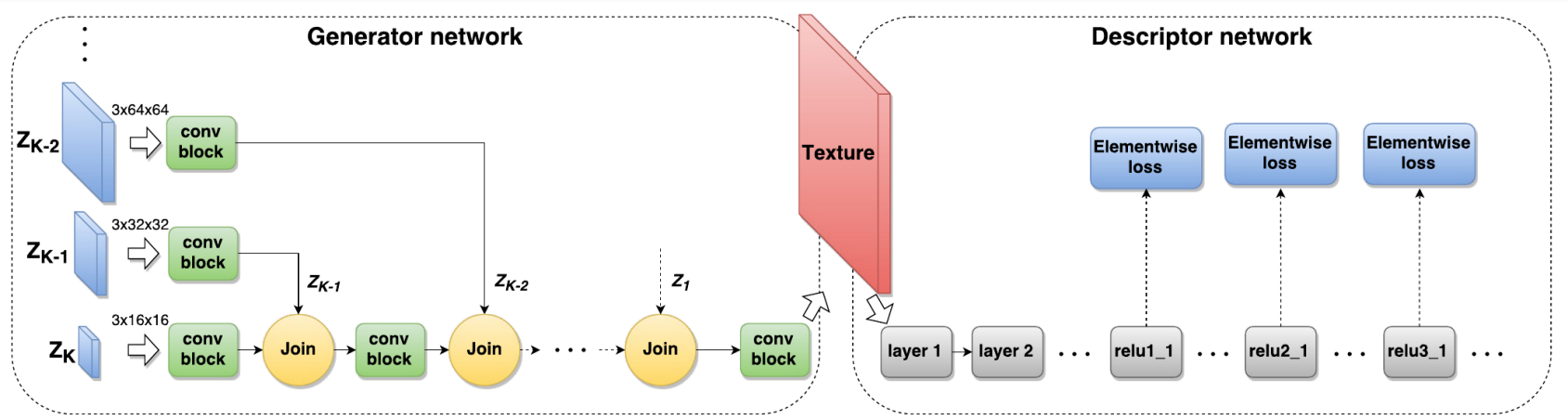
Reference: [Gatys et al., 2016]

Gatys et al for texture synthesis and style transfer

- Very nice and clean PyTorch implementation:
  `https://github.com/leongatys/PytorchNeuralStyleTransfer`
- Today: Practice session based in this code.
- **Very slow** on CPU and computationally demanding with high-end GPU (and memory consuming, e.g. 8 GB of memory for a 1024×1024 image). See practice session.
- Regarding texture modeling, the **number of parameters is huge**: Textures are described by the Gram matrices and the number of elements in the Gram matrices totals 850k. That is 1000 times more than Portilla-Simoncelli !

Generative networks for texture synthesis

- A workaround for speeding up synthesis is to train generative forward networks to mimic Gatys algorithm, as proposed by [Ulyanov et al., 2016] (badly coined Texture Networks).
- The generator is trained to produce images with low Gatys loss (self-supervised training).
- Then synthesis is fast thanks to the feedforward architecture.



Generative networks for texture synthesis

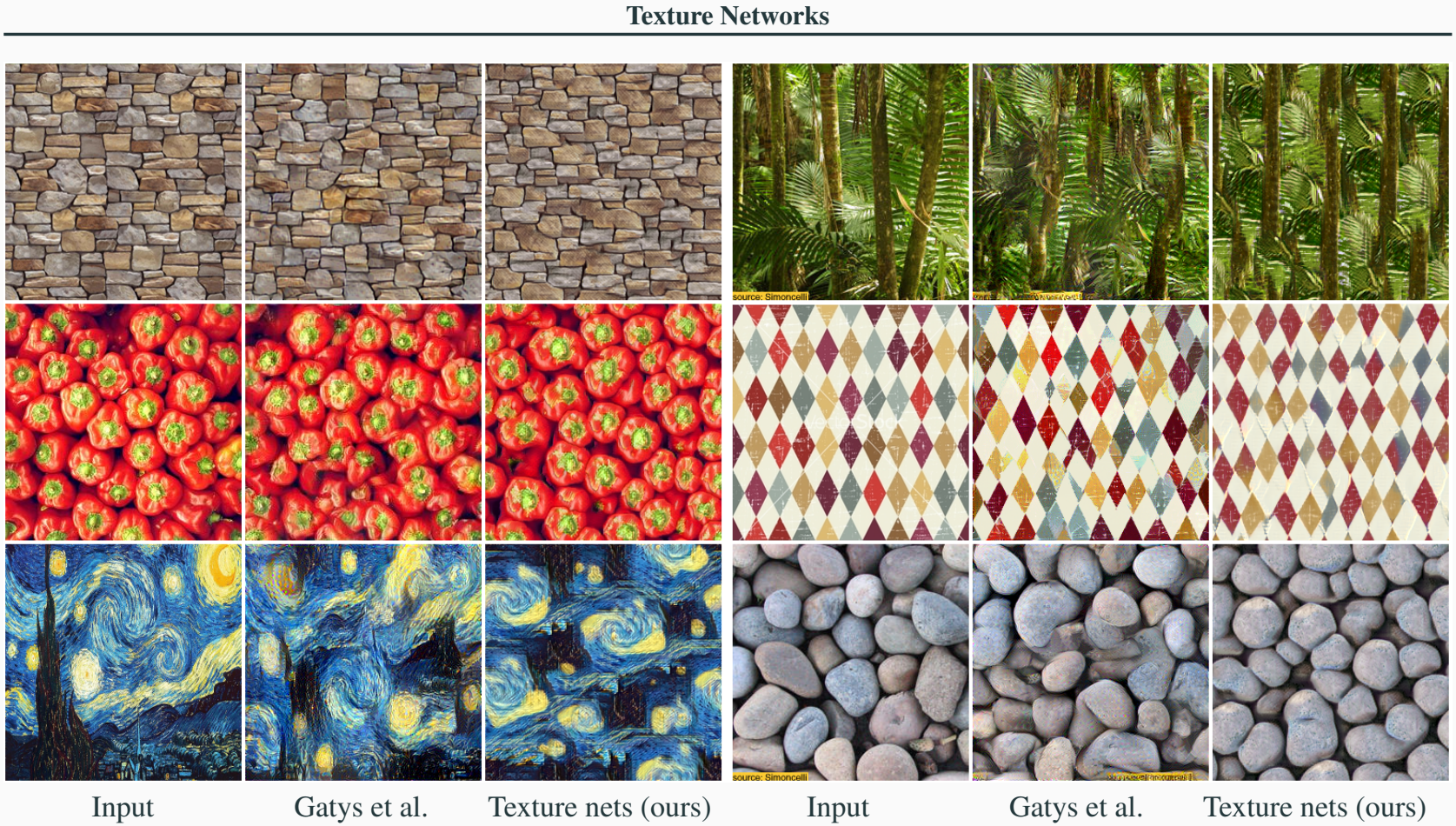
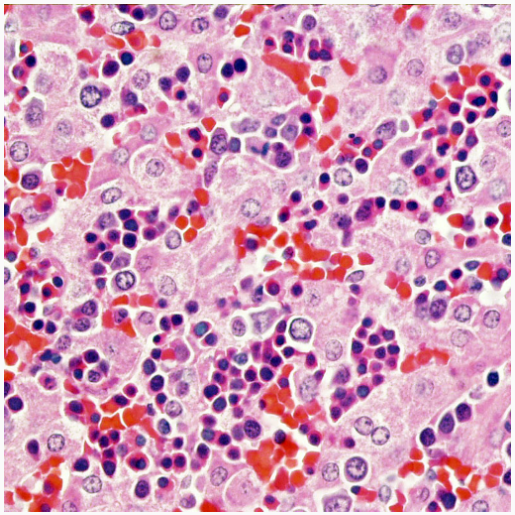


Figure 1. Texture networks proposed in this work are feed-forward architectures capable of learning to synthesize complex textures based on a single training example. The perceptual quality of the feed-forwardly generated textures is similar to the results of the closely related method suggested in (Gatys et al., 2015a), which use slow optimization process.

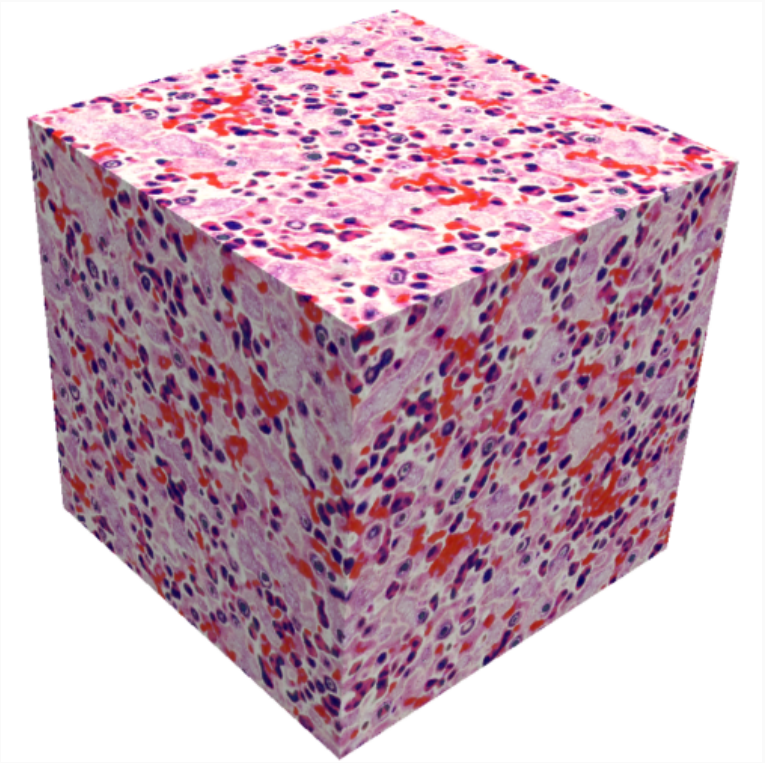
- However texture quality is not as good, and a network has to be trained for each new image.

On Demand Solid Texture Synthesis Using Deep 3D Networks

- Work with Jorge Gutierrez (PhD), Julien Rabin and Thomas Hurtut [Gutierrez et al., 2020].
- Training of generative networks for 3D textures (called solid textures) where the Gatys approach is infeasible.

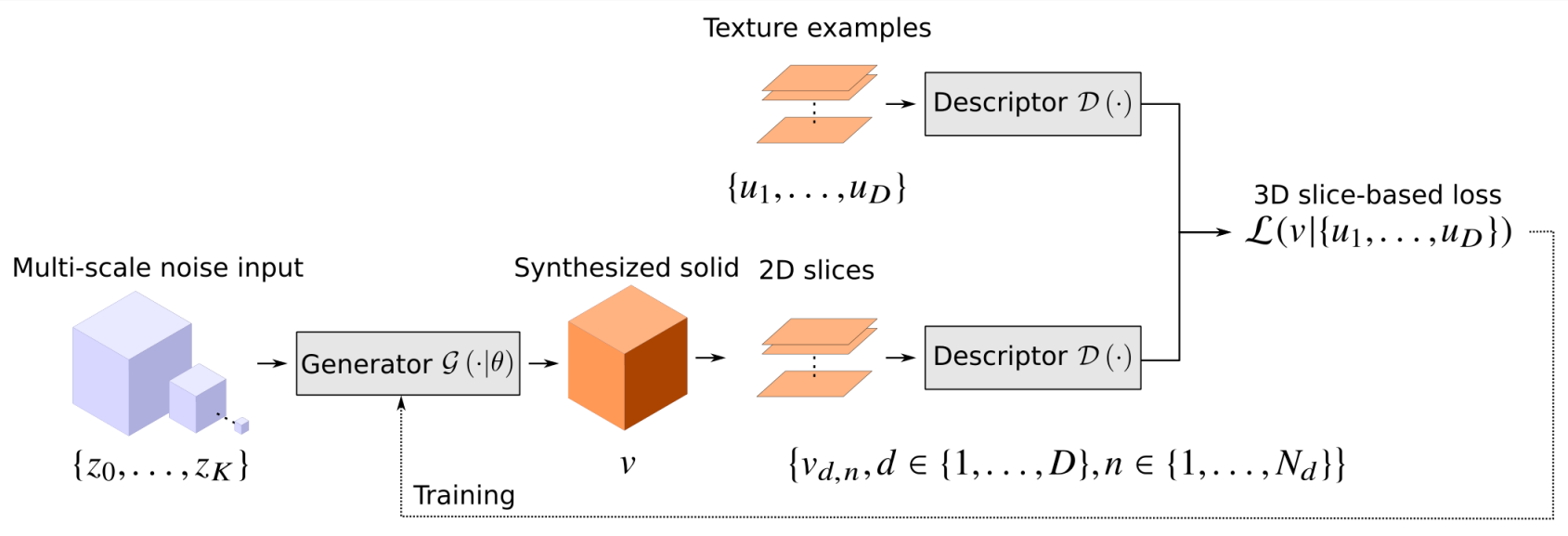


Input



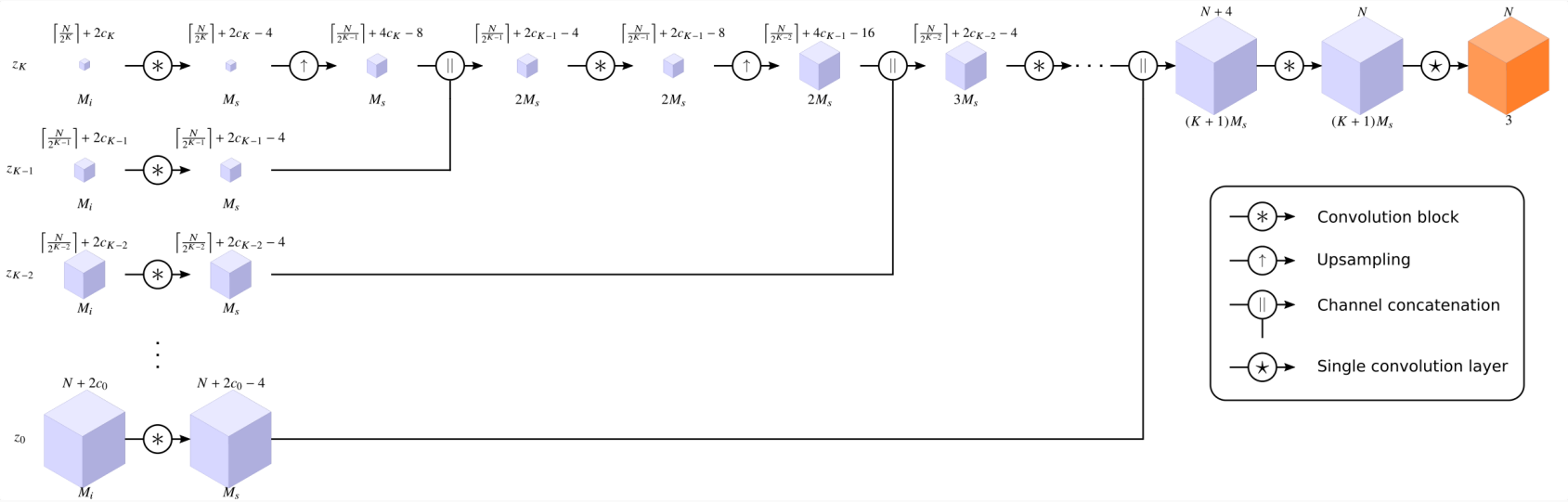
Output

Training framework for the proposed CNN Generator network:



- The generator $\mathcal{G}(\cdot|\theta)$ with parameters θ processes a multi-scale noise input Z to produce a solid texture v
- The loss $\mathcal{L}$ compares, for each direction d, the feature statistics induced by the example u_d in the layers of the pre-trained Descriptor network $\mathcal{D}(\cdot)$ (loss of Gatys et al).

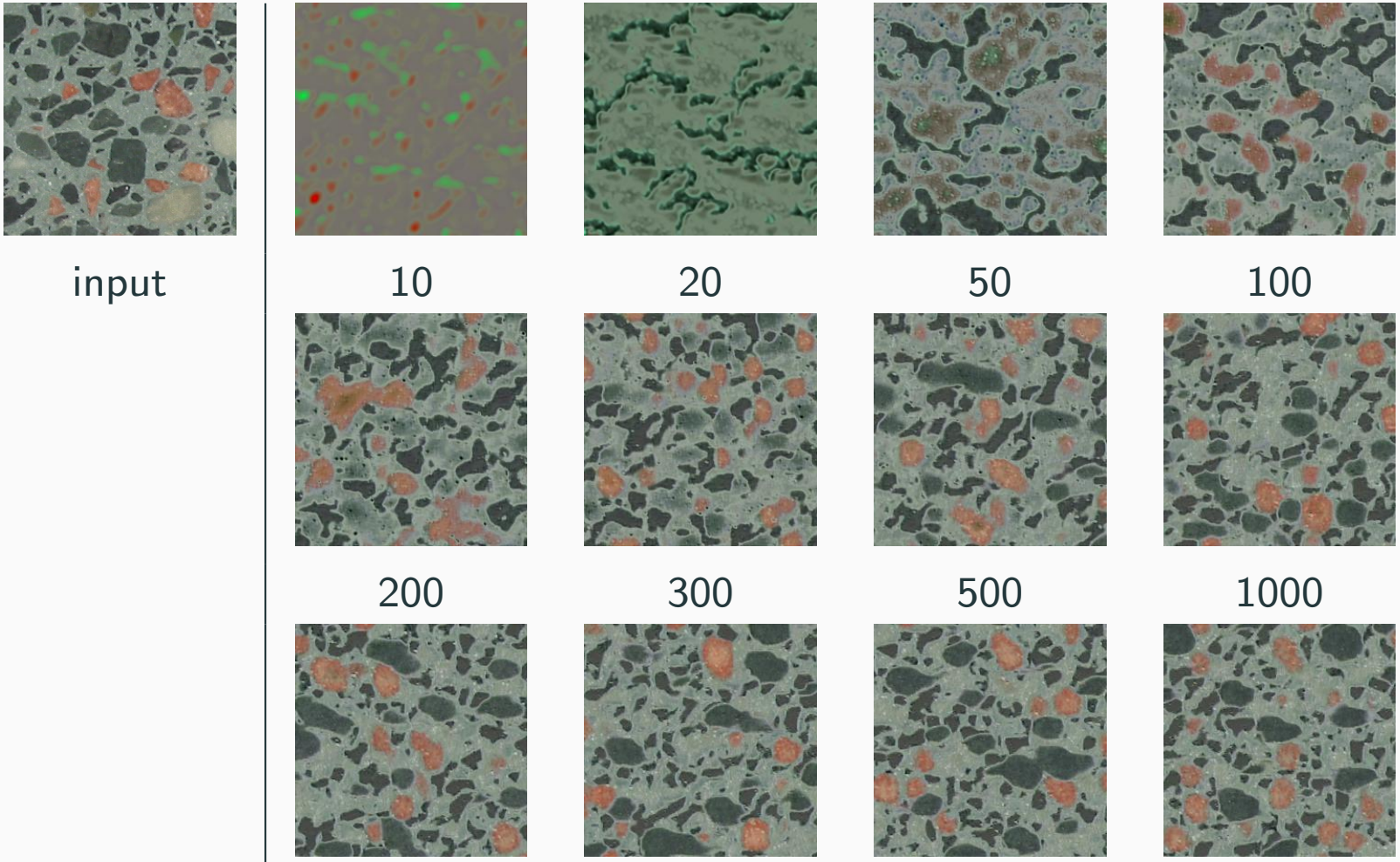
Schematic of the Generator's architecture:



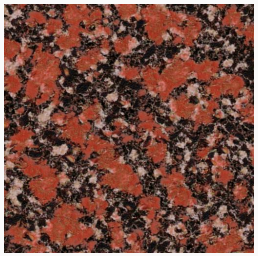
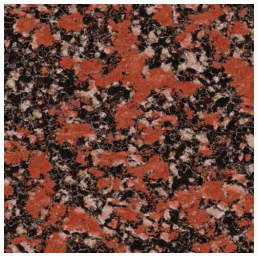
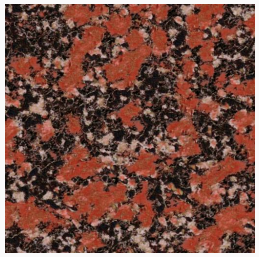
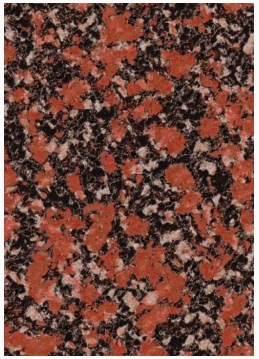
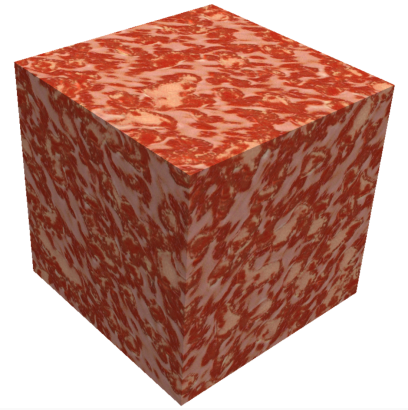
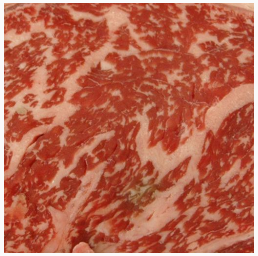
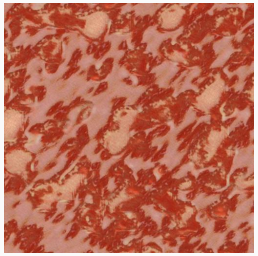
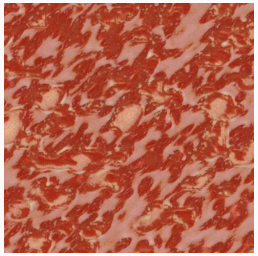
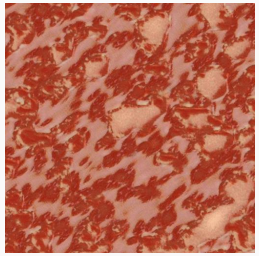
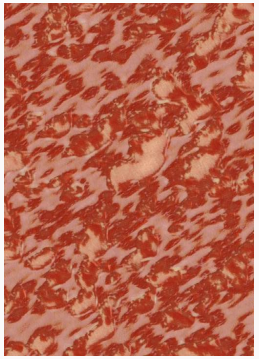
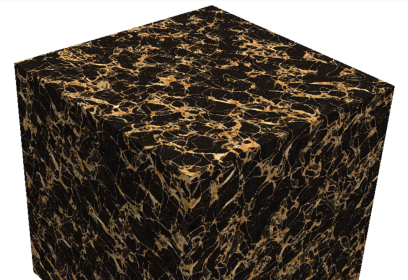
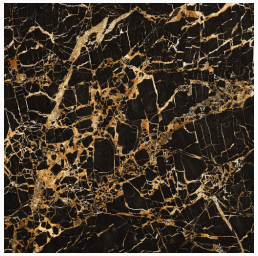
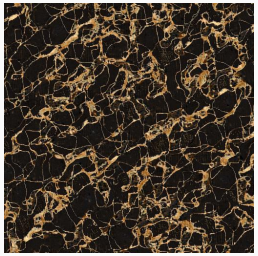
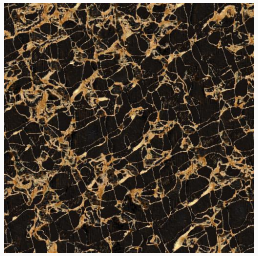
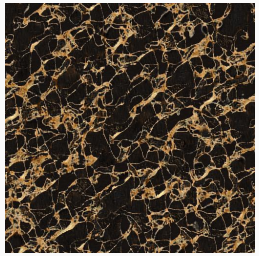
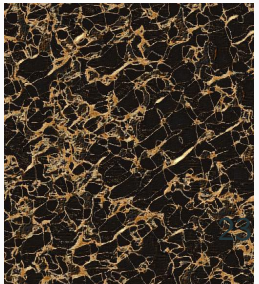
- Processes a set of noise inputs $Z = \{z_0, \ldots, z_K\}$ at $K + 1$ different scales using convolution operations and non-linear activations
- The information at different scales is combined using upsampling and channel concatenation (similar to the right part of a U-net).

Training: Find the parameters θ for a given texture

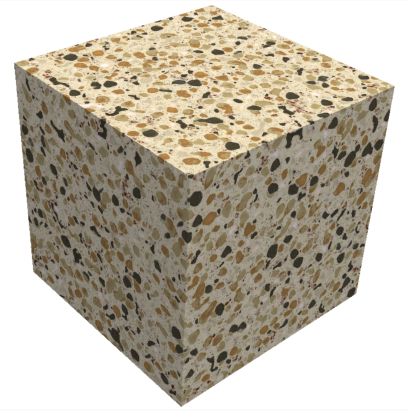
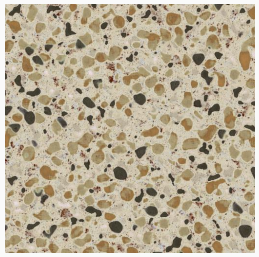
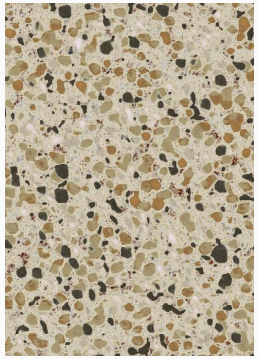
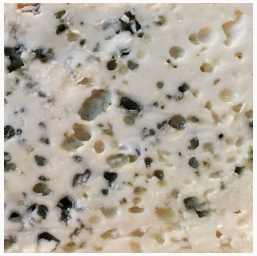
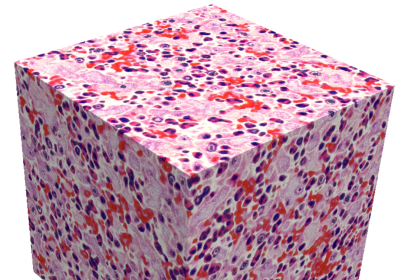
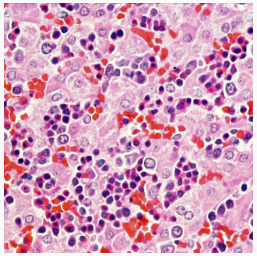
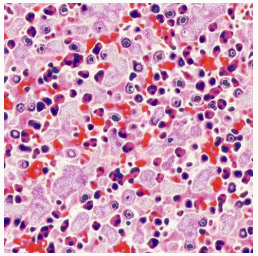
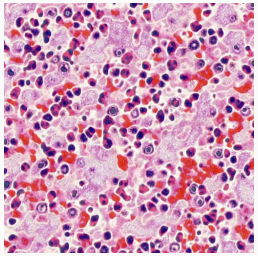
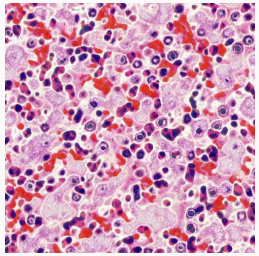
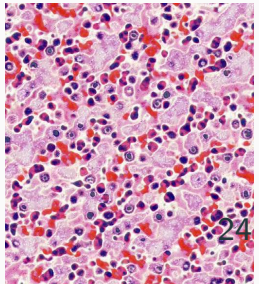
- Minimize Gatys' loss for each slice
- Exploit invariance by translation to generate batches of width one voxel only
- Minimization using 3000 iterations of Adam algorithm



On Demand Solid Texture Synthesis Using Deep 3D Networks

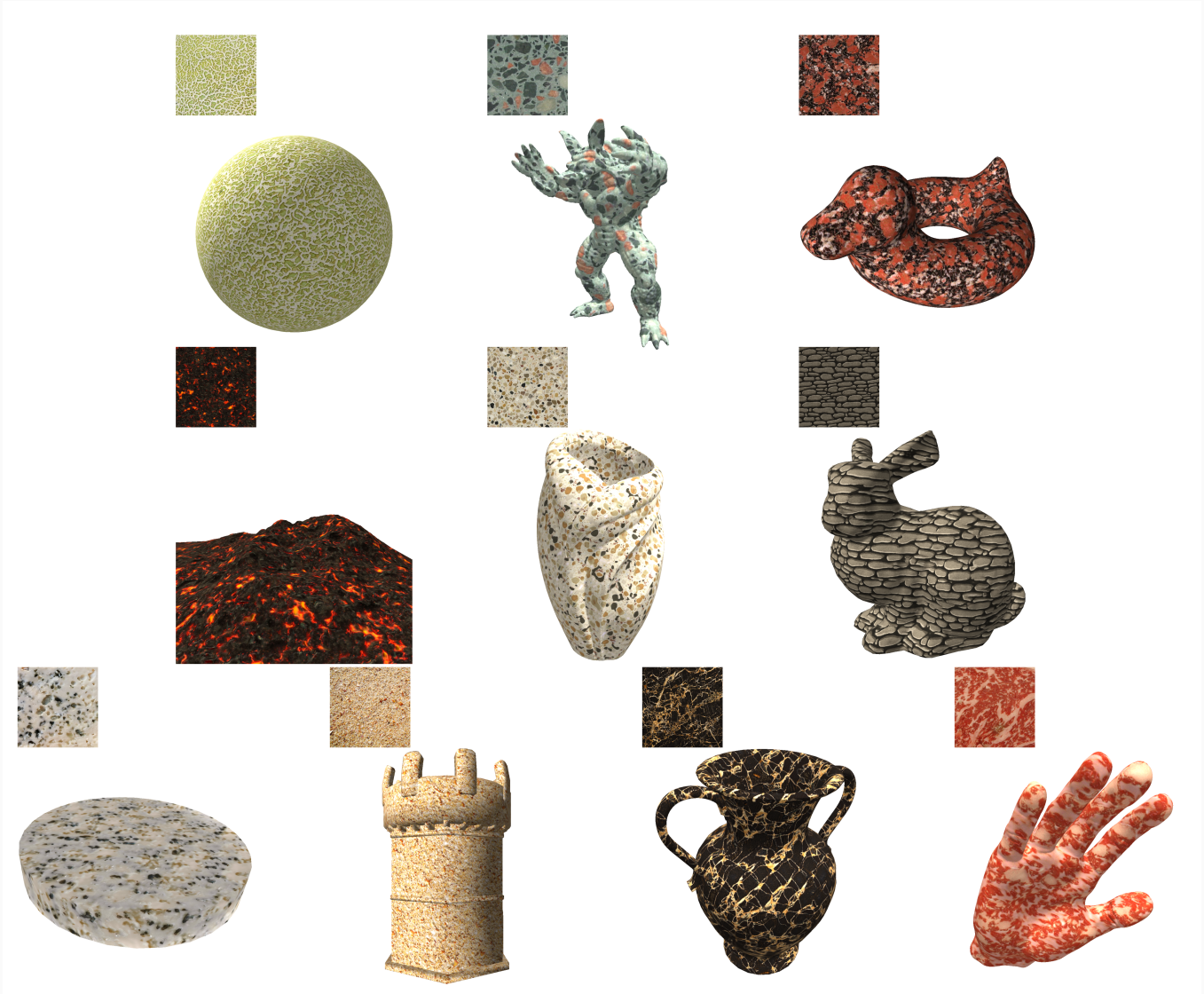
	Generated volume v	Examples $u_1 = u_2 = u_3$	Generated slices $v_{1,\frac{N_1}{2}}$	$v_{2,\frac{N_2}{2}}$	$v_{3,\frac{N_3}{2}}$	oblique (45°)
granite						
beef						
marble						

On Demand Solid Texture Synthesis Using Deep 3D Networks

	Generated volume v	Examples $u_1 = u_2 = u_3$	Generated slices $v_{1,\frac{N_1}{2}}$	$v_{2,\frac{N_2}{2}}$	$v_{3,\frac{N_3}{2}}$	oblique $(45°)$
pebble						
cheese						
histology						

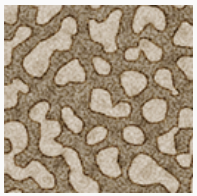
On Demand Solid Texture Synthesis Using Deep 3D Networks

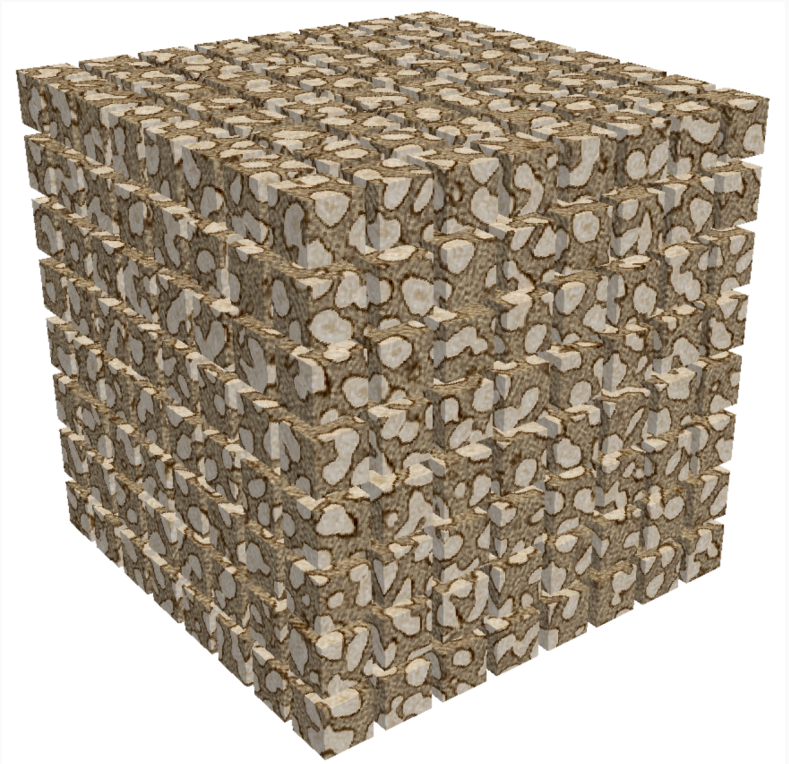
- Solid textures can be used to apply textures on surfaces without parametrization.



On Demand Solid Texture Synthesis Using Deep 3D Networks

- Fast synthesis thanks to the feed forward network (1 sec. for 256^3)
- On demand synthesis using a pseudo random number generator seed with spatial coordinates





- Training and synthesis with high resolution images without memory issues thanks to the single slice strategy.

Generative networks for texture synthesis

- Other contributions propose generative networks for **universal** style transfer/texture synthesis.
- **Universal** means that a single network is adapted to all images.
- This still relies in approximating the Gatys procedure with some approximation for a faster style transfer: auto-encoders to invert VGG19 and imposing Gram matrices.

References i


Gatys, L. A., Ecker, A. S., and Bethge, M. (2016).
Image style transfer using convolutional neural networks.
In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423.


Gutierrez, J., Rabin, J., Galerne, B., and Hurtut, T. (2020).
On demand solid texture synthesis using deep 3d networks.
Computer Graphics Forum, 39(1):511–530.


Simonyan, K. and Zisserman, A. (2015).
Very deep convolutional networks for large-scale image recognition.
In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

References ii


Ulyanov, D., Lebedev, V., Vedaldi, A., and Lempitsky, V. (2016).
Texture networks: Feed-forward synthesis of textures and stylized images.
In *ICML*, pages 1349–1357.


Wei, L.-Y., Lefebvre, S., Kwatra, V., and Turk, G. (2009).
State of the art in example-based texture synthesis.
In *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association.

Questions?

Slides from Charles Deledalle