Attacking inverse problems with deep learning

Patrick Pérez

valeo.ai

AI & Physics, Pascal Institute, Saclay

21 March 2019

Inverse problems

High level formulation

- Forward process to "invert"

$$\mathbf{y} \to \mathbf{x} = g(\mathbf{y})$$

$$\hat{\mathbf{y}}(\mathbf{x}) \in g^{-1}(\mathbf{x})?$$

- Optimization-based inversion

$$\forall \mathbf{x},\ \hat{\mathbf{y}}(\mathbf{x}) \in \arg\min_{\mathbf{y}} \underbrace{\left(\text{Loss}(\mathbf{x}, g(\mathbf{y})) + \text{Prior}(\mathbf{y})\right)}_{E(\mathbf{x},\mathbf{y};\boldsymbol{\theta})}$$

- Model bricks: given (e.g., *physics*), engineered, and/or *learned*

Classic examples

Signal enhancement or completion

- Forward process: degradation and/or masking
- Prior: spatial/temporal regularity

$$\min_{\mathbf{y}} \left(\|\mathbf{x} - g(\mathbf{y})\|_2^2 + \mu \|\nabla \mathbf{y}\|_p^q \right) \text{ s.t. } \mathbf{y}|_{\partial D} = \mathbf{y}^*$$

Signal recovery

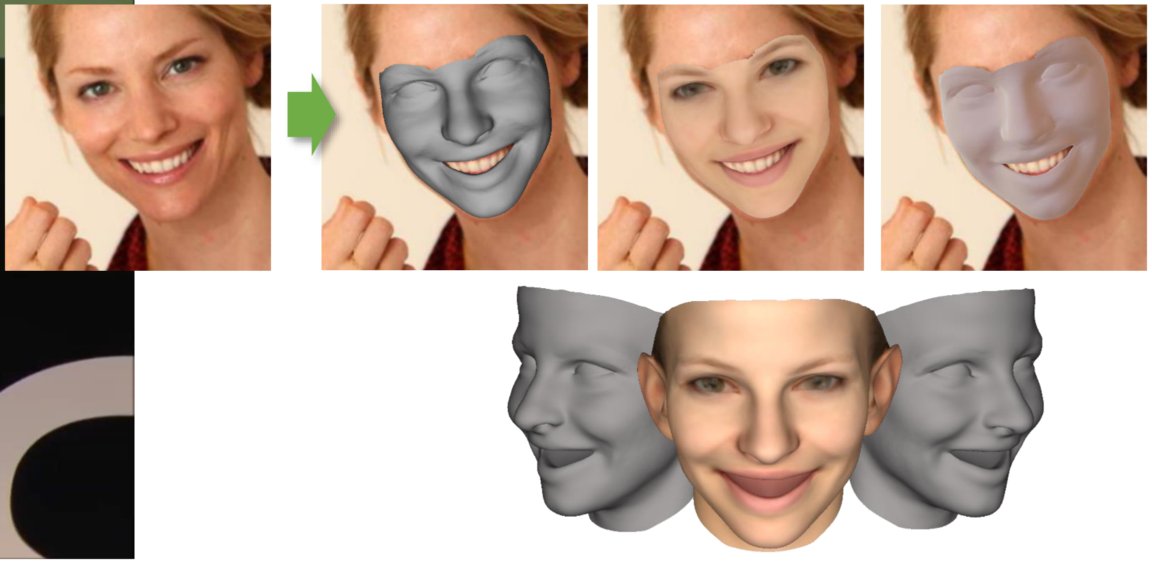
- Linear forward process: atom composition, random measurements
- Prior: sparsity

$$\min_{\mathbf{y}} \left(\|\mathbf{x} - A\mathbf{y}\|_2^2 + \mu \|\mathbf{y}\|_1 \right)$$

Less classic examples

Inverse graphics

- CGI: from scene model to photoreal images
- *Inverse rendering*: from real images to scene model (“graphic code”)
- Application: AR, VR, editing, retargeting, personalized models



[Tewari 2017]

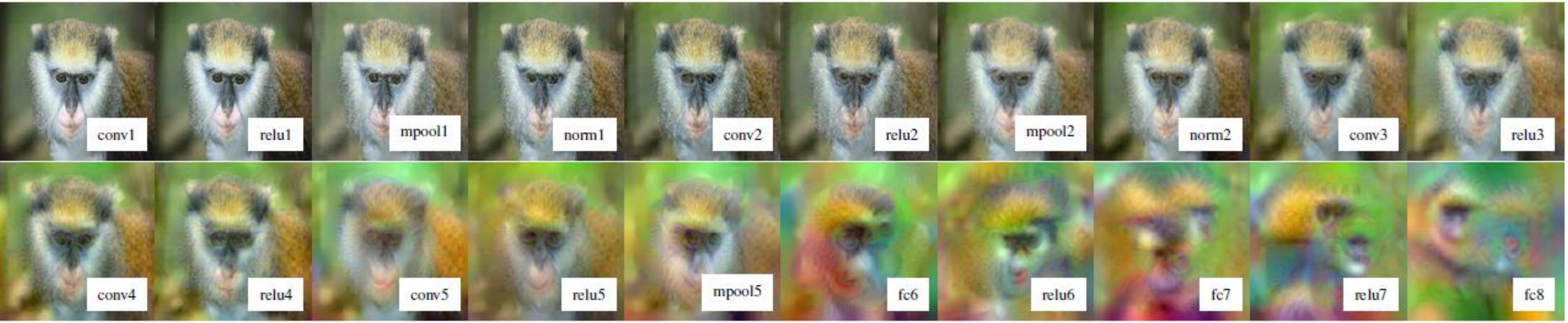
Less classic examples

"Neural" inverse problems

- Deep features: from signal to representations through feedforward neural net
- *Inverting*: from neural activations to NN input

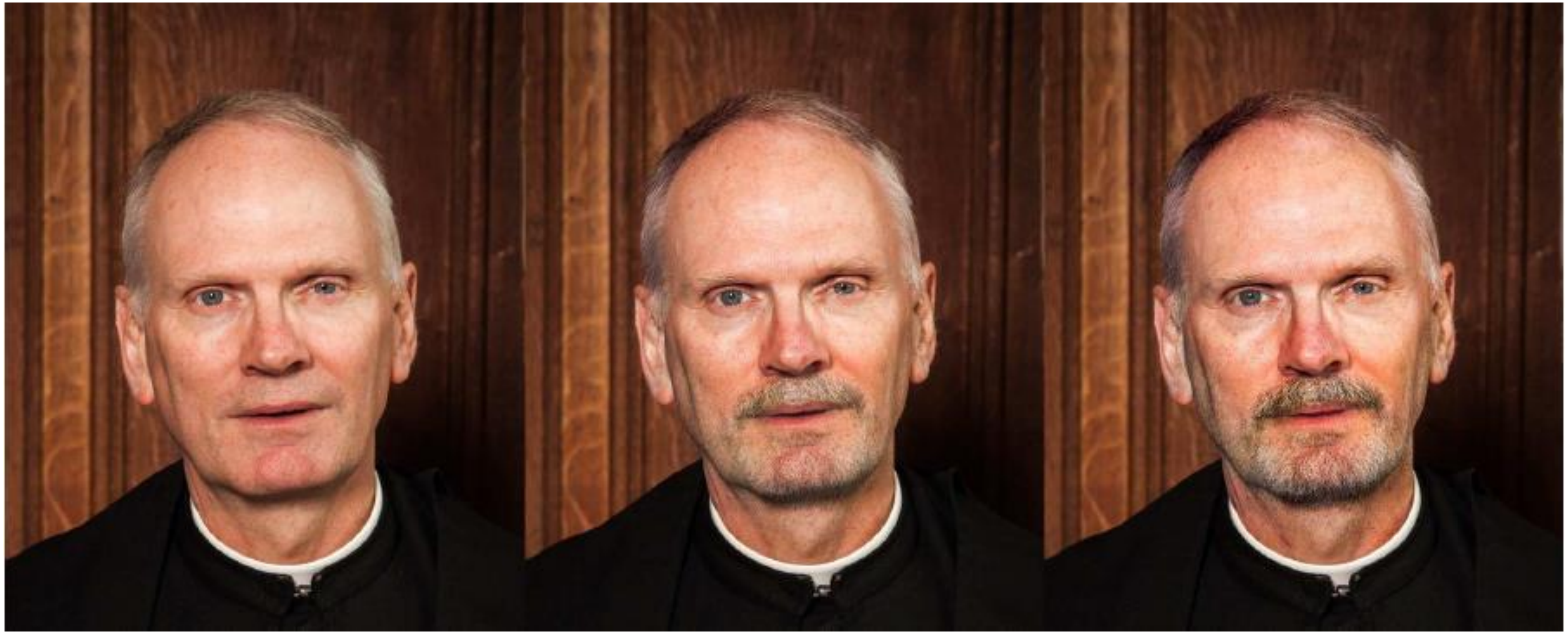
$$\min_{\mathbf{y}} \left(\|\mathbf{x} - \phi_{\ell_0}(\mathbf{y})\|_F^2 + \mu \|\nabla \mathbf{y}\|_1 \right)$$

- Application: visualization, inspection, editing in feature domain



Less classic examples

“Neural” inverse problems



[Upchich 2017]

Learning and inverse problems?

Learn the model, solve by optimization

- Forward process, prior and loss can be learned
- Examples: blind deconvolution, trained MRFs
- Inference with a classic solver, *iterative* and *generic*

$$\forall \mathbf{x}, \ \mathsf{Solver}(\mathbf{x}) = \mathrm{Iter}^{\infty}(\mathbf{x}; \mathbf{y}^0) \approx \hat{\mathbf{y}}(\mathbf{x})$$

Train a direct solver

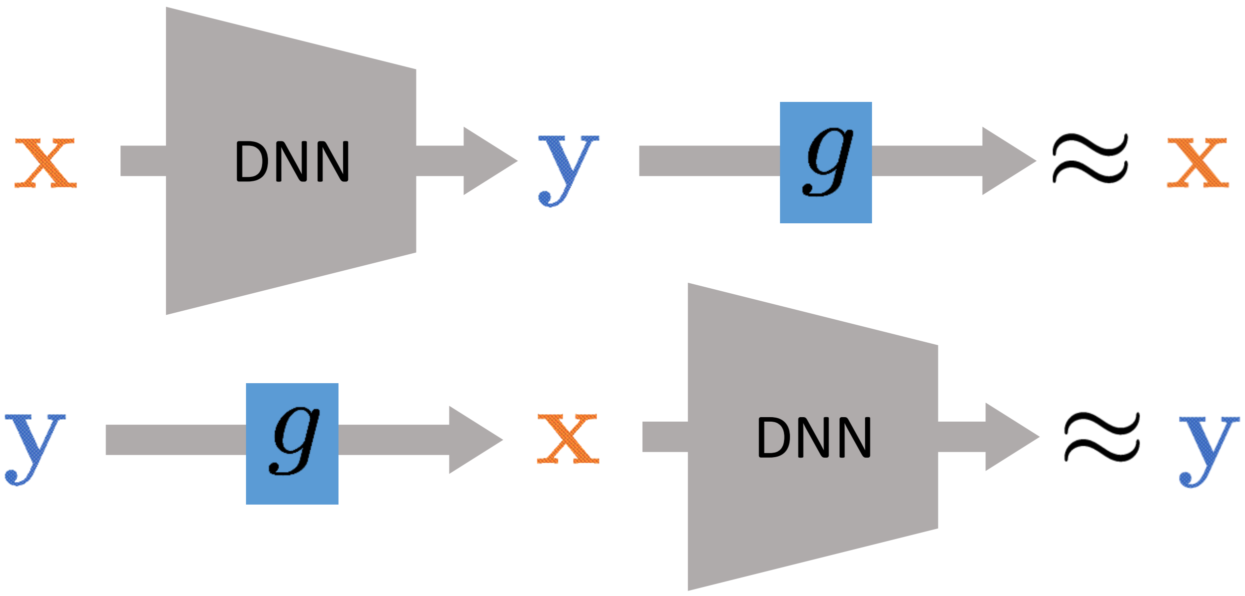
- From $\forall \mathbf{x}, \ \hat{\mathbf{y}}(\mathbf{x}) \in \arg\min_{\mathbf{y}} E(\mathbf{x}, \mathbf{y})$
- To $\forall \mathbf{x} \sim \mathbb{P}_X, \ f(\mathbf{x}; \mathcal{W}) \approx \hat{\mathbf{y}}(\mathbf{x})$

Neural inversion

Train a DNN to regress solution *for plausible inputs*

$$\forall \mathbf{x} \sim \mathbb{P}_X,\ \mathsf{DNN}(\mathbf{x}; \mathcal{W}) \approx \hat{\mathbf{y}}(\mathbf{x})$$

- Fixed complexity
- Possibly way faster
- Flexible in various ways
- Differentiable



Architecture?

Training?

Architectures

Your favorite DNN

- Exploit popular architectures, possibly pre-trained
- Popular convolutional and recurrent neural nets

Unrolling

- Mimic (possibly loosely) structure of iterative solver
- Each [non-linear ∘ linear] iteration becomes a trainable neural layer
- Fixed, smaller number of “iterations”
- But, way more freedom exploited through training

Training

Fully-supervised – Using Solver, reconstruction loss

$$\mathbf{x}^{(n)} \sim \mathbb{P}_X, \quad \min_{\mathcal{W}} \sum_{n=1}^{N} \|\text{Solver}(\mathbf{x}^{(n)}) - \text{DNN}(\mathbf{x}^{(n)}; \mathcal{W})\|_2^2$$

Self-supervised – Only “synthetic data”, reconstruction loss

$$\mathbf{y}^{(n)} \sim \mathbb{P}_Y, \quad \min_{\mathcal{W}} \sum_{n=1}^{N} \|\mathbf{y}^{(n)} - \text{DNN}(g(\mathbf{y}^{(n)}); \mathcal{W})\|_2^2$$

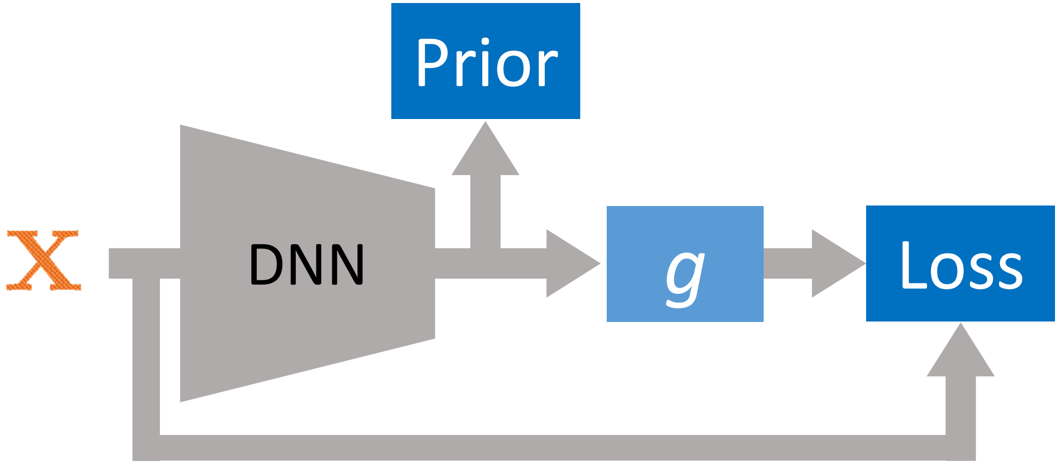
Unsupervised – Original objective function as training loss

$$\mathbf{x}^{(n)} \sim \mathbb{P}_X, \quad \min_{\mathcal{W}} \sum_{n=1}^{N} E(\mathbf{x}^{(n)}, \text{DNN}(\mathbf{x}^{(n)}; \mathcal{W}); \boldsymbol{\theta})$$

Unsupervised training

Unsupervised – Original objective function as training loss

$$\mathbf{x}^{(n)} \sim \mathbb{P}_X, \quad \min_{\mathcal{W}} \sum_{n=1}^{N} \mathsf{Loss}\Big(\mathbf{x}^{(n)}, g \circ \mathsf{DNN}(\mathbf{x}^{(n)}, \mathcal{W})\Big) + \mathsf{Prior}\Big(\mathsf{DNN}(\mathbf{x}^{(n)}, \mathcal{W})\Big)$$



Unrolling example

Sparse coding: LISTA [Gregor 2010]

$$\min_{\mathbf{y}} \left(0.5\|\mathbf{x} - A\mathbf{y}\|_2^2 + \mu\|\mathbf{y}\|_1\right)$$

Iterative solver: Iterative Soft Thersholding Alg. (ISTA)

$$\text{Iter}(\mathbf{y}) = \sigma_{2\mu\alpha}\left((\text{Id} - \alpha A^\top A)\mathbf{y} + \alpha A^\top \mathbf{x}\right)$$

Learned neural layer (residual block and skip connection)

$$f_k(\mathbf{y}) = \sigma\left((\text{Id} - V_k)\mathbf{y} + W_k^\top \mathbf{x}\right), \ \mathcal{W} = \{(W_k, V_k)\}_k$$

- Fully supervised training
- The deeper, the better the approximation
- Modest speed-up

Unsupervised neural inversion

Personalized 3D face model: Tewari *et al.* 2017

Encoder-decoder with differentiable rendering layer

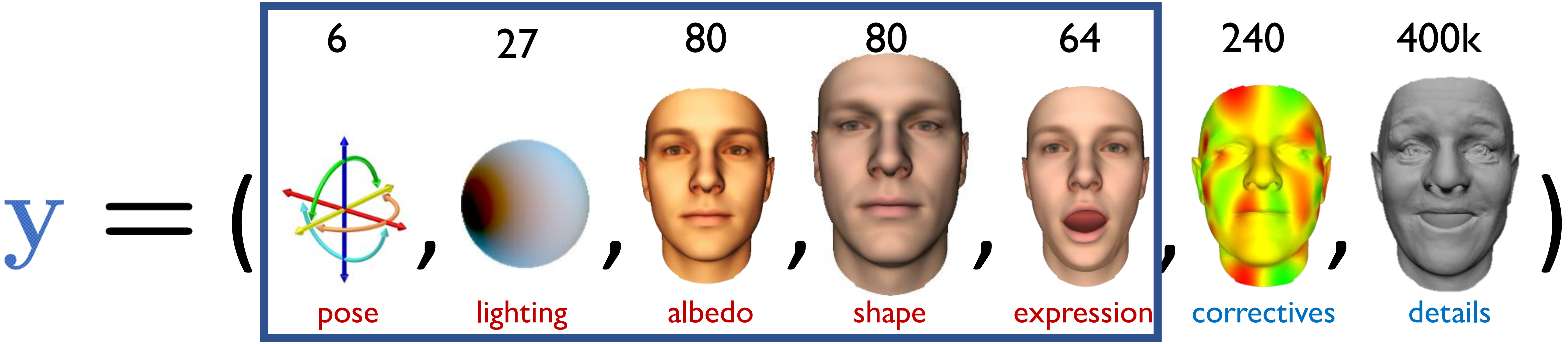
Artistic style transfer: Ulyanov *et al*. 2016, Johnson *et al*. 2016

Encoder-decoder with “perceptual loss”

Flexible style transfer: Puy & Pérez 2019

Unrolling descent, run time flexibility

Photo-real face rendering



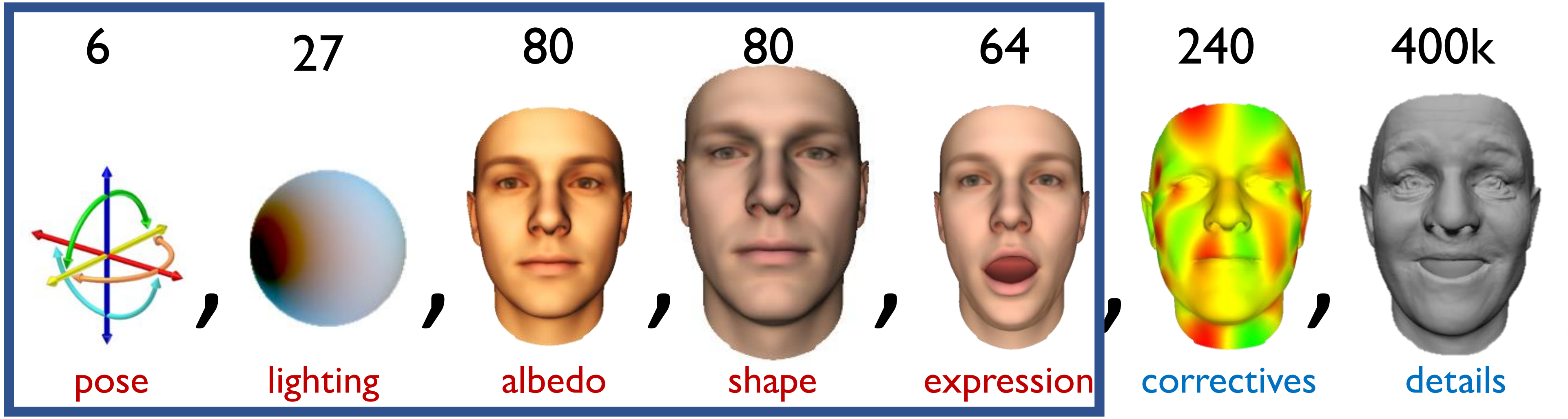
$$\mathbf{y} \in \mathbb{R}^{257} \rightarrow g(\mathbf{y}) =$$

[Garrido 2016]

Invert rendering

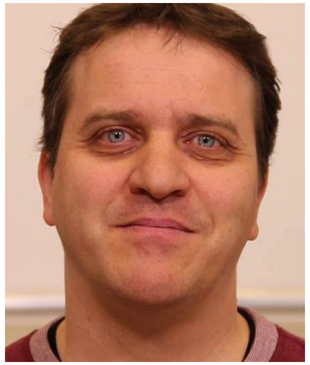
to obtain animatable personnalized 3D rig

$\mathbf{y} = ($



$)$

$\mathbf{x} =$

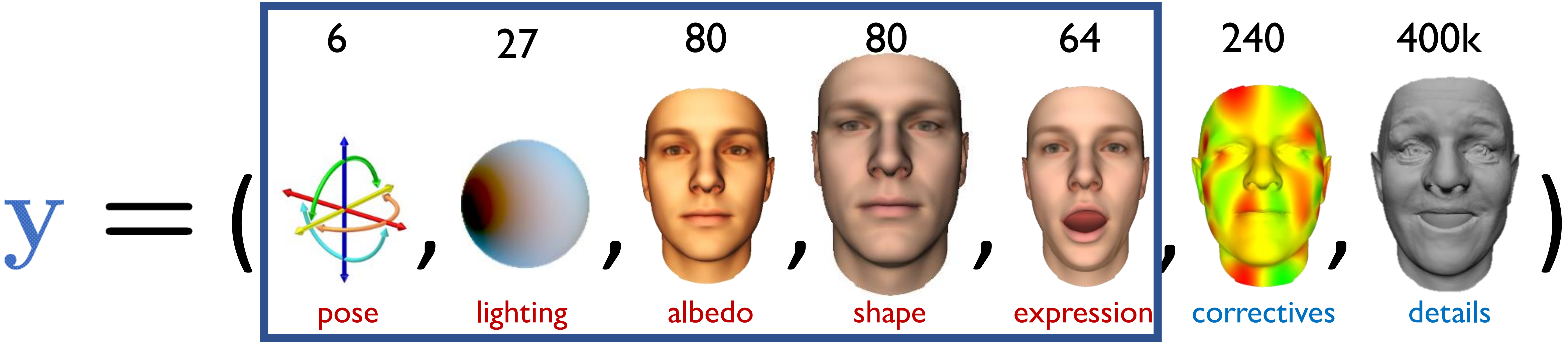


$\rightarrow \mathbf{y} \in \mathbb{R}^{257}$?

[Garrido 2016]

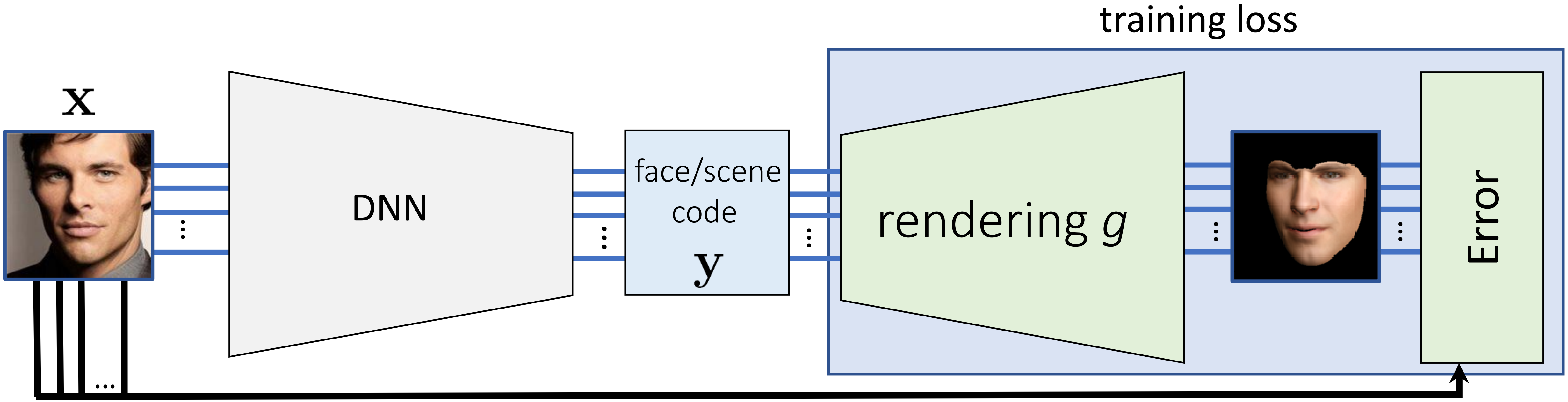
Invert rendering

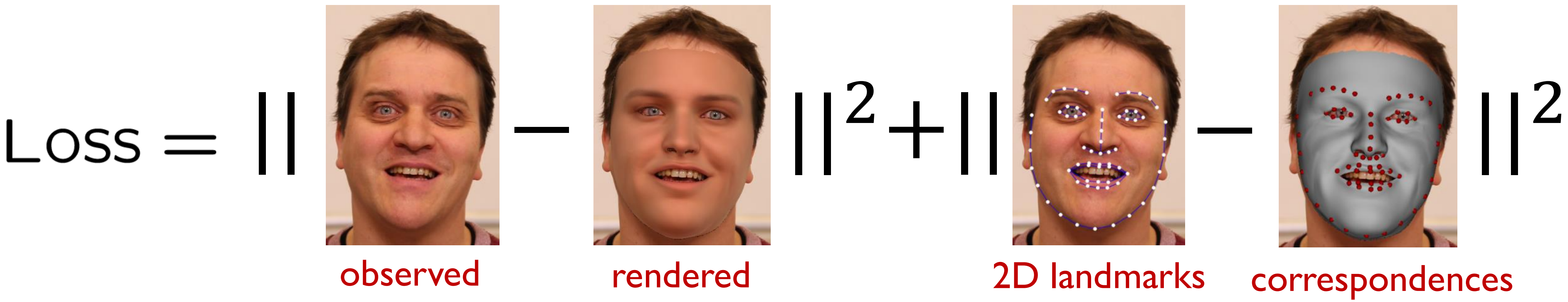
To obtain animatable personnalized 3D rig



[Garrido 2016]

Fast DNN solver



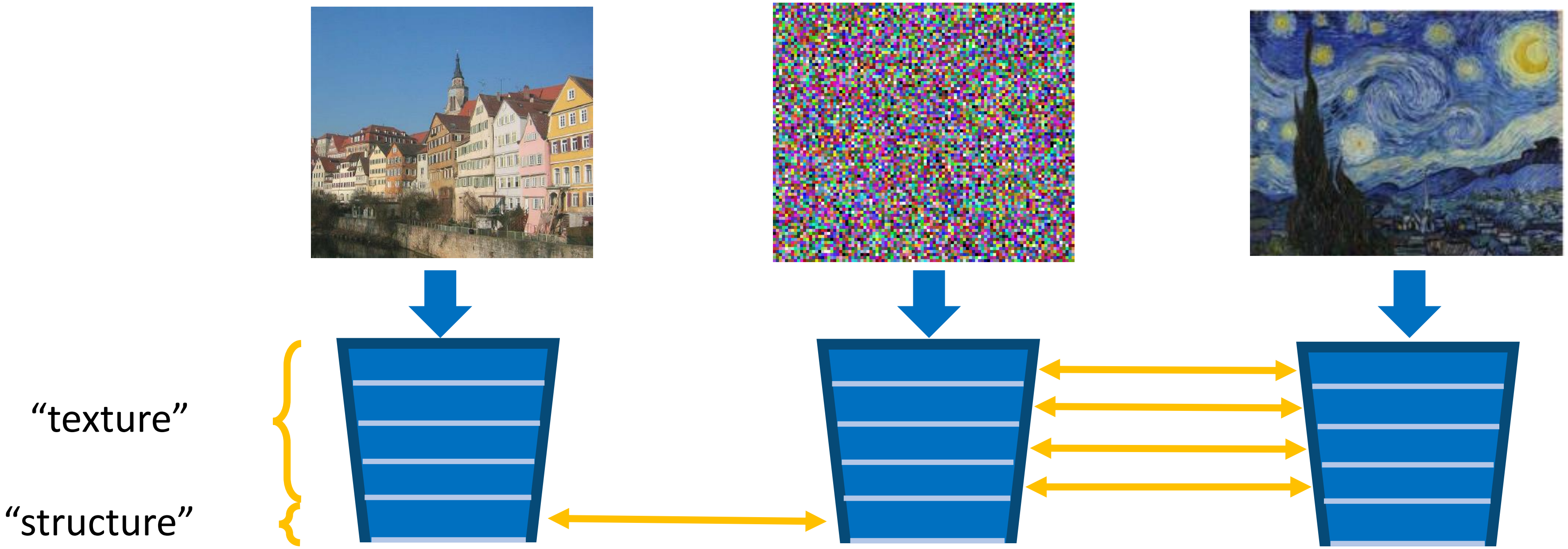


[Tewari 2017-2018]

“Artistic” Style transfer

retain structure, imitate texture

$$\min_{\mathbf{y}} \left(\|\mathbf{x} - \phi_{\ell_0}(\mathbf{y})\|_{\mathsf{F}}^2 + \lambda \sum_{\ell \in \mathcal{L}_{\mathsf{sty}}} \|G_\ell - \phi_\ell(\mathbf{y})^\top \phi_\ell(\mathbf{y})\|_{\mathsf{F}}^2 \right)$$

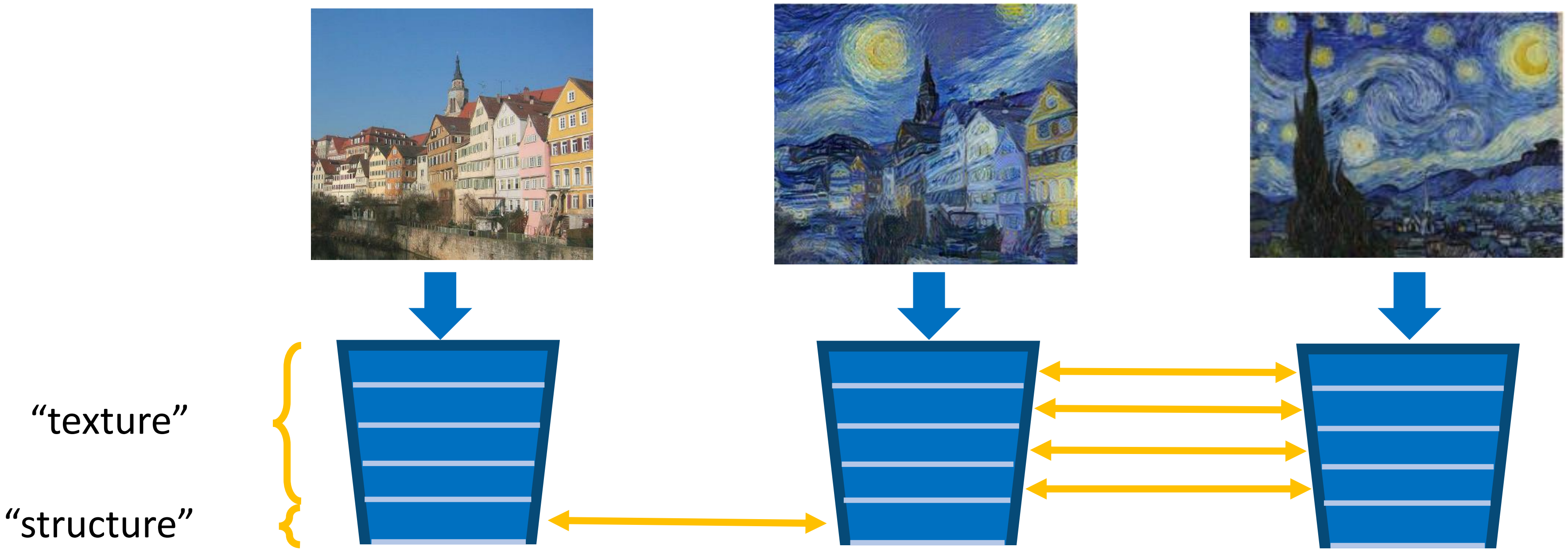


[Gatys 2015-2016]

“Artistic” Style transfer

retain structure, imitate texture

$$\min_{\mathbf{y}} \left(\|\mathbf{x} - \phi_{\ell_0}(\mathbf{y})\|_{\mathsf{F}}^2 + \lambda \sum_{\ell \in \mathcal{L}_{\mathsf{sty}}} \|G_\ell - \phi_\ell(\mathbf{y})^\top \phi_\ell(\mathbf{y})\|_{\mathsf{F}}^2 \right)$$



[Gatys 2015-2016]

Fast artistic style transfer

[Ulyanov 2016, Johnson 2016] and successors

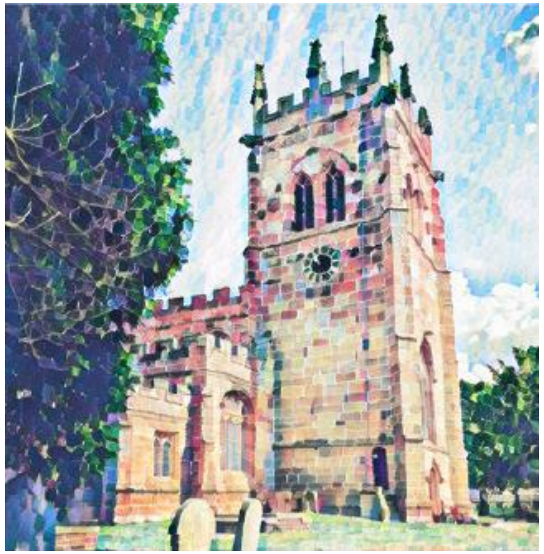
- Convolutional encoder-decoder architectures
- Unsupervised training *for specified paintings*

[Ulyanov 2017]

Fast artistic style transfer

[Ulyanov 2016, Johnson 2016] and successors

- Convolutional encoder-decoder architectures
- Unsupervised training *for specified paintings*

[Ulyanov 2017]

Fast flexible style transfer [Puy 2019]

Unrolling (part of) gradient descent

$$E(\mathbf{x},\mathbf{y}) = \|\mathbf{x} - \phi_{\ell_0}(\mathbf{y})\|_F^2 + \mu\|\nabla\mathbf{y}\|_1 + \sum_{\ell\in\mathcal{L}_{\text{sty}}} \lambda_\ell \|G_\ell - \phi_\ell(\mathbf{y})^\top \phi_\ell(\mathbf{y})\|_F^2$$

- One layer mimics one step $\mathbf{y} \leftarrow \mathbf{y} - \alpha\nabla E_{\text{sty}}(\mathbf{x},\mathbf{y})$

$$\mathbf{y}_k = \mathbf{y}_{k-1} - f_k\Big(\mathbf{x}, \mathbf{y}_{k-1}; \mathcal{W}_k, \underbrace{\{\lambda_\ell, G_\ell\}_{\ell\in\mathcal{L}_{\text{sty}}}}_{\text{modifiable at } \textit{run time}}\Big)$$

modifiable at *run time*

- Unsupervised training

Runtime restructuring

Choose style, mix styles, tune stylization intensity or scale

Runtime restructuring

Add new regularizers via proximal operator, e.g. for photorealism

$$\mathbf{y}_k = \mathbf{y}_{k-1} - \mathsf{Prox}_{\Omega}\left[f_k(\mathbf{x}, \mathbf{y}_{k-1})\right]$$

Runtime restructuring

Add new regularizers via proximal operator, e.g. for photorealism

$$\mathbf{y}_k = \mathbf{y}_{k-1} - \mathsf{Prox}_\Omega\Big[f_k(\mathbf{x}, \mathbf{y}_{k-1})\Big]$$

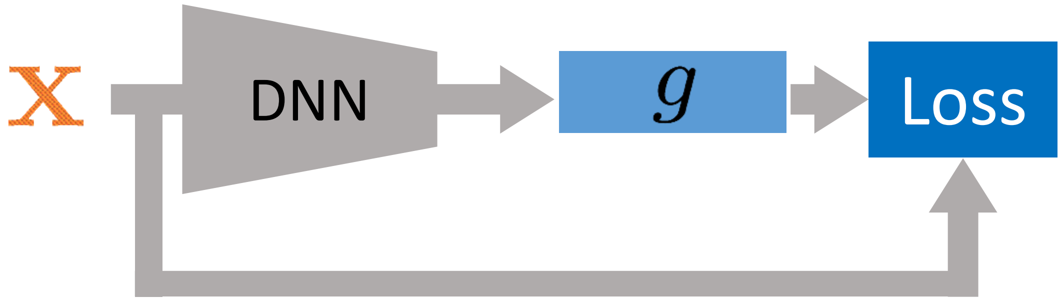
Runtime restructuring

Add new regularizers via proximal operator, e.g. for photorealism

$$\mathbf{y}_k = \mathbf{y}_{k-1} - \mathsf{Prox}_{\Omega}\Big[f_k(\mathbf{x}, \mathbf{y}_{k-1})\Big]$$

Perspectives

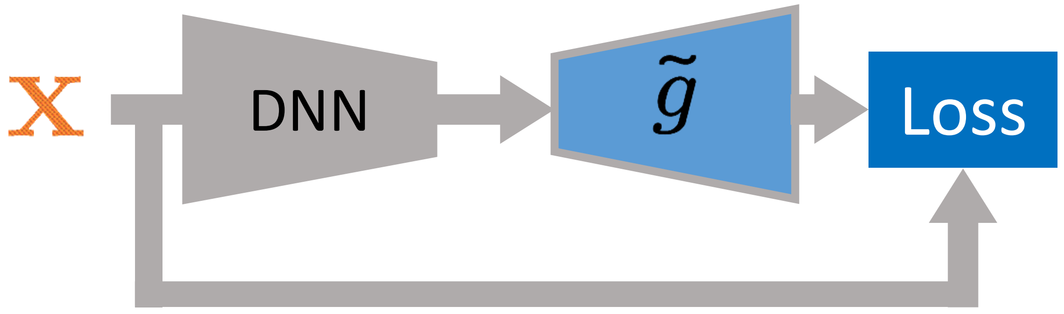
- Encoder-decoder view of unsupervised inversion: make forward process (partly) trainable, e.g. [Tewari 2018]



- *Invertible* NN [Ardizzone 2019] trained to mimic *g*, inversion for free
- Inversion of state-of-art GAN?
- Predict multiple relevant pre-images?

Perspectives

- Encoder-decoder view of unsupervised inversion: make forward process (partly) trainable, e.g. [Tewari 2018]



- *Invertible* NN [Ardizzone 2019] trained to mimic *g*, inversion for free
- Inversion of state-of-art GAN?
- Predict multiple relevant pre-images?

Conclusion

Neural solvers for inverse problems

- fast, specialized, differentiable, possibly unsupervised, flexible
- can go beyond original model by learning
- applies to other optimization-based/variational problems

References

[Ardizzone 2019] L. Ardizzone, J. Kruse, C. Rother, U. Köthe. *Analyzing inverse problems with invertible neural networks*. ICLR 2019

[Garrido 2016] P Garrido, M Zollhöfer, D Casas, L Valgaerts, K Varanasi, P Pérez, Ch. Theobalt. *Reconstruction of personalized 3D face rigs from monocular video*. ACM TOG, 2016

[Gatys 2016] L. Gatys, A. Ecker, M. Bethge. *Image style transfer using convolutional neural networks*. CVPR 2016

[Gregor 2010] K. Gregor, Y. LeCun Y. *Learning fast approximations of sparse coding*. ICML 2010

[Johnson 2016] J. Johnson, A. Alahi, L. Fei-Fei. *Perceptual losses for real-time style transfer and super-resolution*. ECCV 2016

[Krizhevsky 2012] A. Krizhevsky, I. Sutskever, G. Hinton. *Imagenet classification with deep convolutional neural networks*. NIPS 2012

[Mahendran 2015] A. Mahendran, A. Vedaldi. *Understanding deep image representations by inverting them*. ICCV 2015

[Puy 2019] G. Puy and P. Pérez. *A flexible convolutional solver with application to photorealistic style transfer*. CVPR 2019

[Tewari 2017] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Pérez, Ch. Theobalt. *MoFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction*. ICCV 2017

[Tewari 20178 A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, Ch. Theobalt. *Self-supervised multi-level face model learning for monocular reconstruction at over 250 Hz*. CVPR 2018

[Tompson 2017] J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin. *Accelerating Eulerian fluid simulation with convolutional network*. ICML 2017

[Ulyanov 2016] D. Ulyanov, V. Lebedev, A. Vedaldi, V. Lempitsky. TexturenNetworks: Feed-forward synthesis of textures and stylized Images. ICML 2016