Mathematical foundations of Machine Learning 2024 – lesson 7

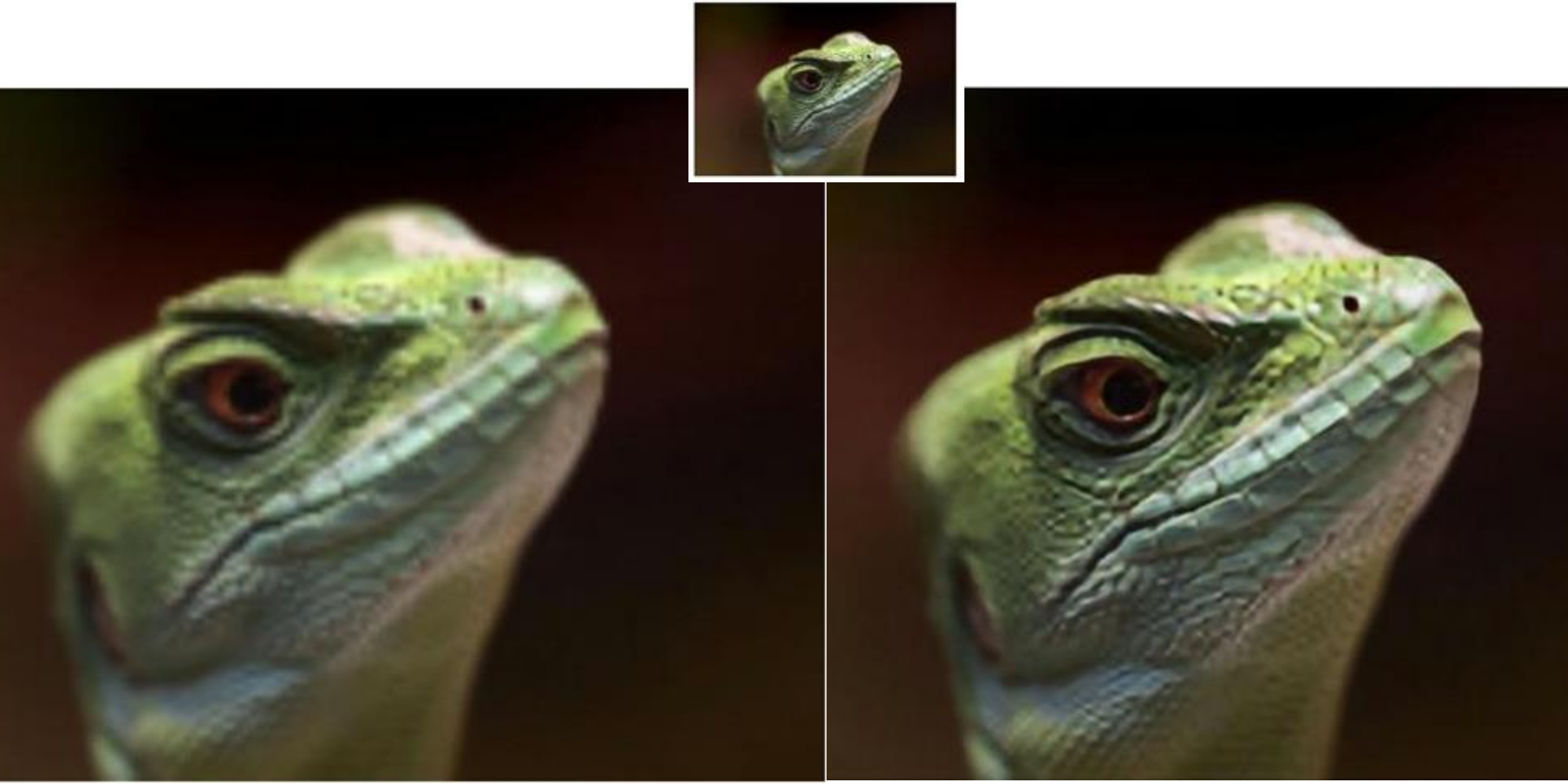
Shai Dekel

TEL AVIV UNIVERSITY

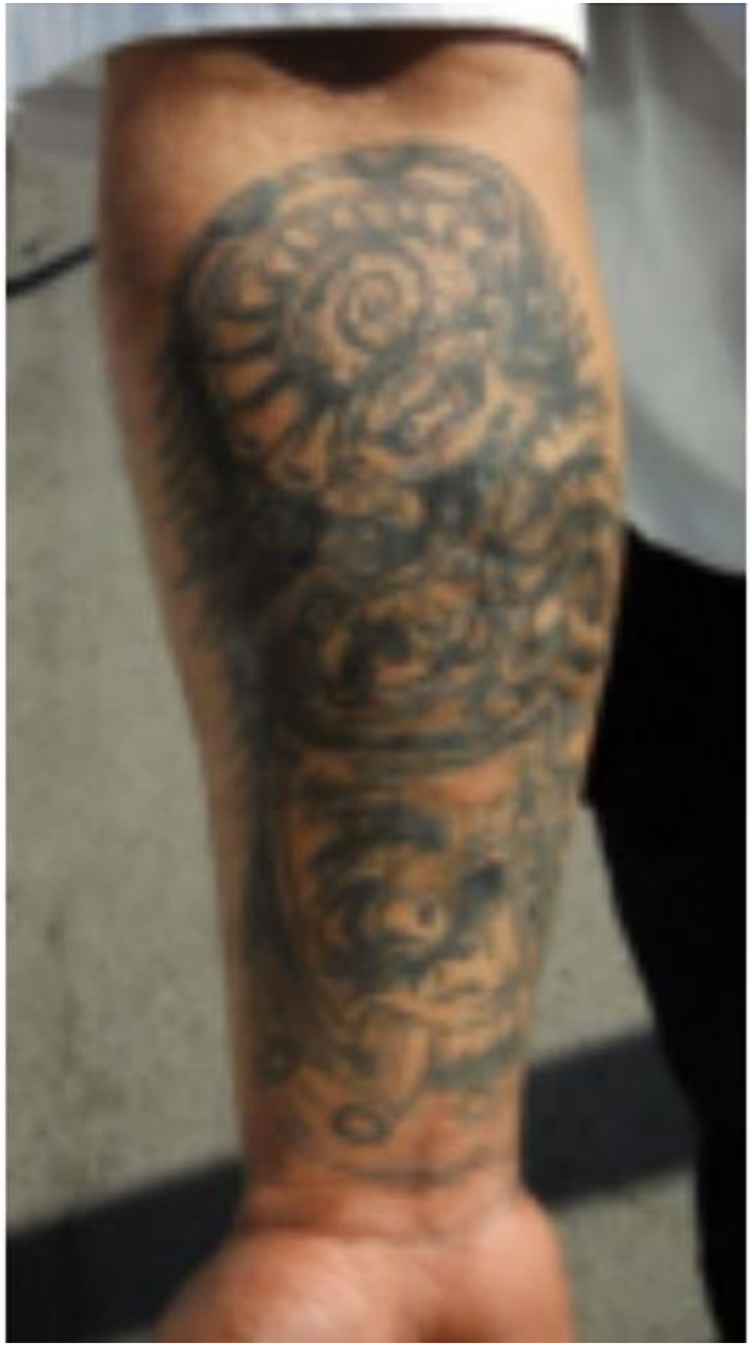
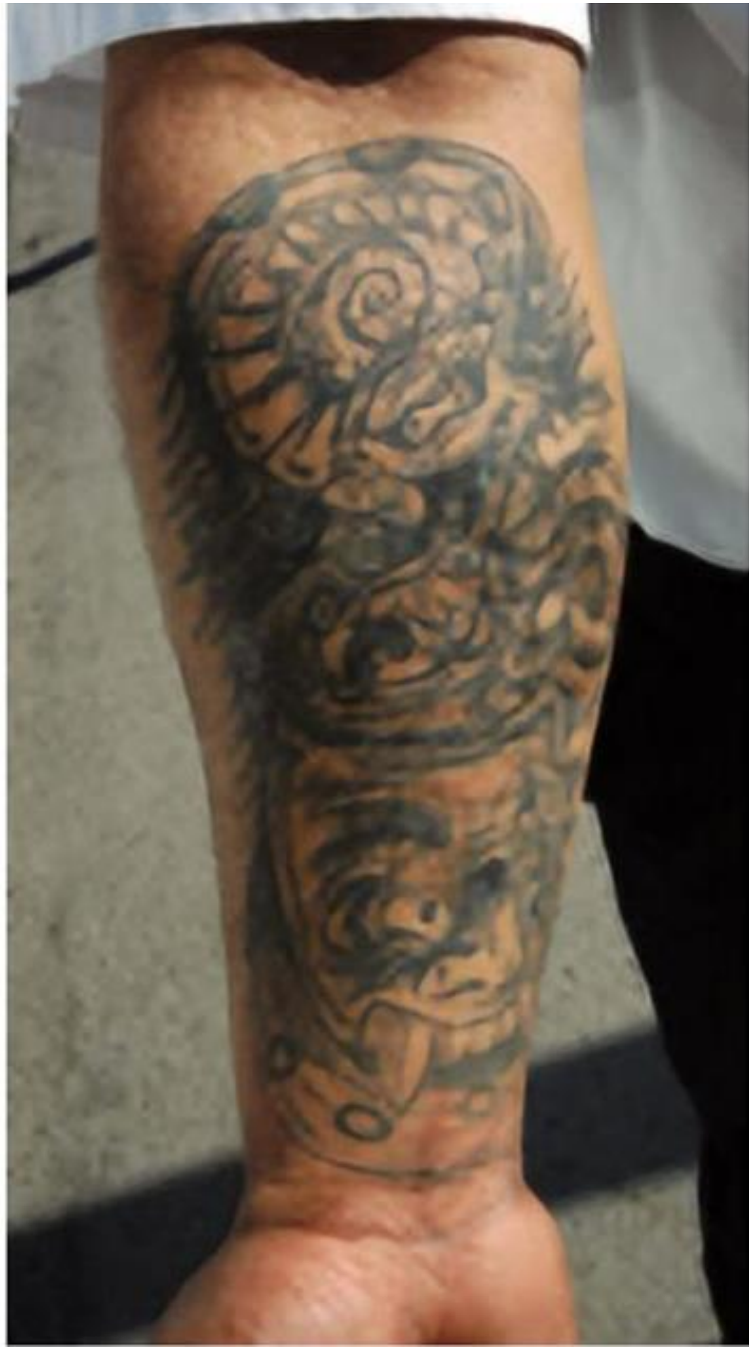
Computer vision/imagining applications of deep learning

AI-based image super resolution

(enlarging an image with best possible quality/sharpness)





Data Curation for Super Resolution

- Goal

Train a NN that takes as input a low-resolution image and outputs a high-resolution image (no blur, fine details).

- Training Dataset
  - 120,000 images
  - Ground truth – High resolution images
  - Input images – low resolution images produced from ground truth

Starting point (CVPR 2016)

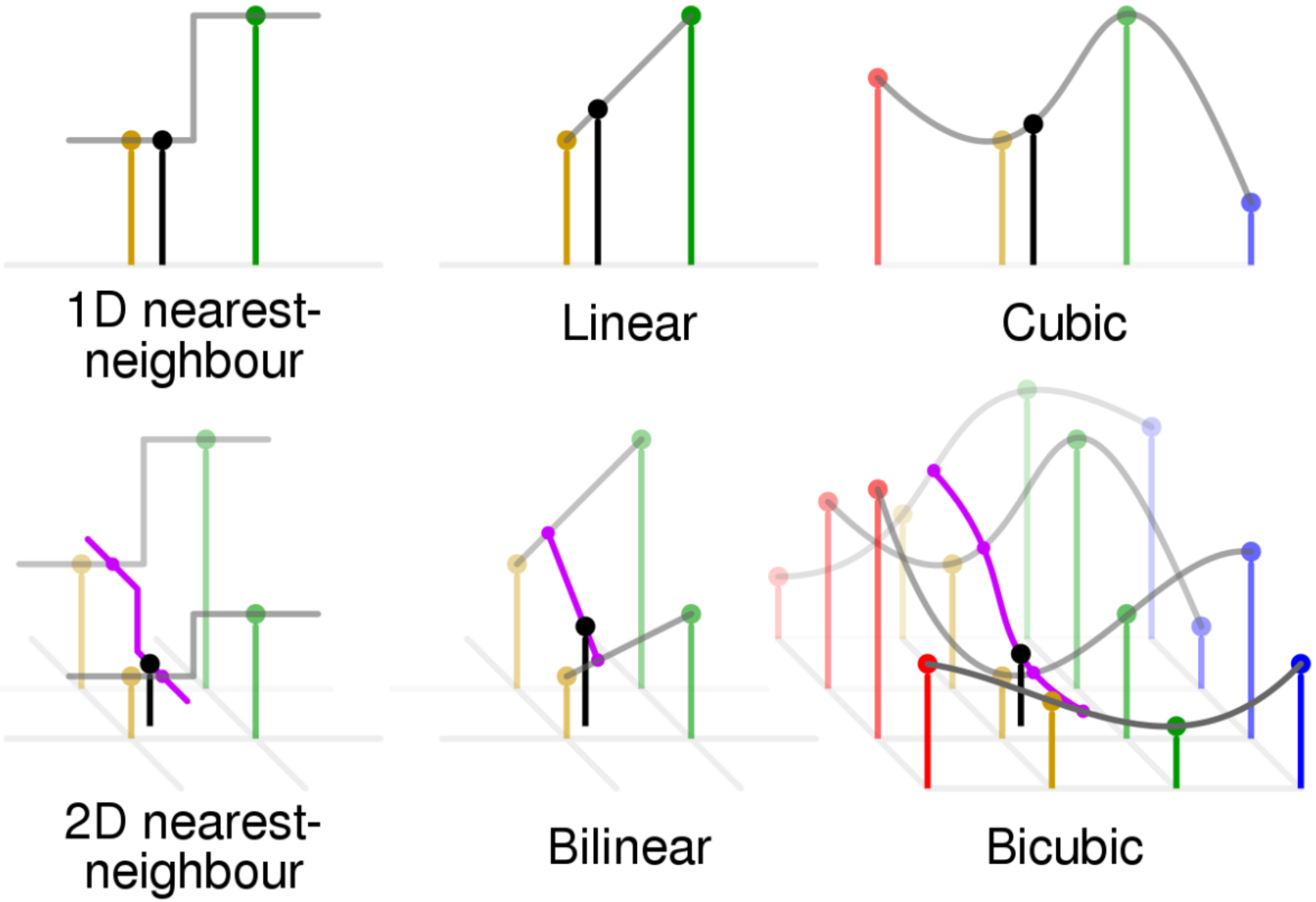
Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network

Wenzhe Shi[1], Jose Caballero[1], Ferenc Huszár[1], Johannes Totz[1], Andrew P. Aitken[1], Rob Bishop[1], Daniel Rueckert[2], Zehan Wang[1]
[1]Magic Pony Technology [2]Imperial College London
[1]{wenzhe, jose, ferenc, johannes, andy, rob, zehan}@magicpony.technology
[2]D.Rueckert@imperial.ac.uk

Linear & Bicubic interpolation

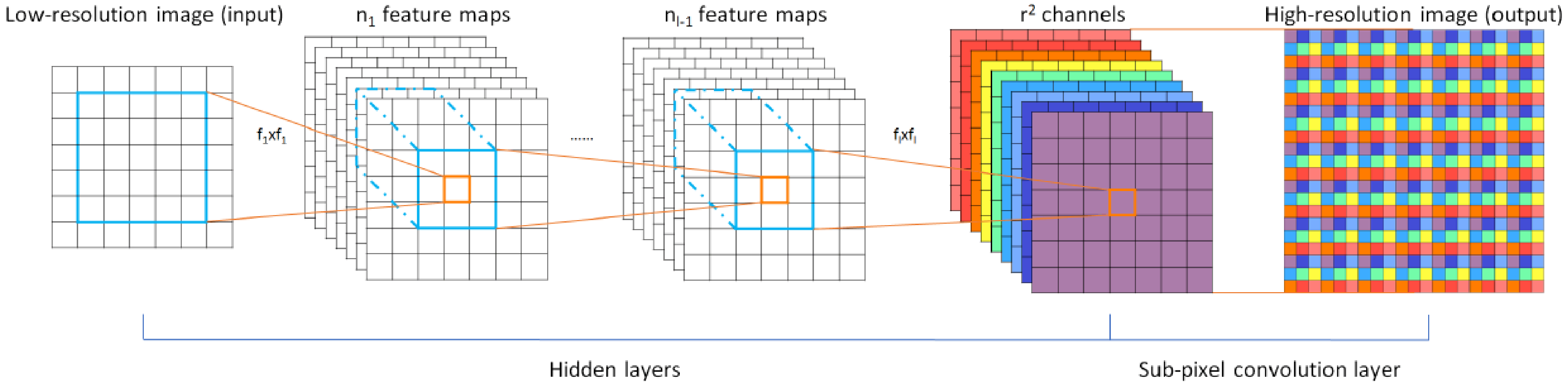
$$p_1(x_1, x_2) = \sum_{i_1=0}^{1}\sum_{i_2=0}^{1} a_{i_1,i_2} x_1^{i_1} x_2^{i_2}$$

$$p_3(x_1, x_2) = \sum_{i_1=0}^{3}\sum_{i_2=0}^{3} a_{i_1,i_2} x_1^{i_1} x_2^{i_2}$$

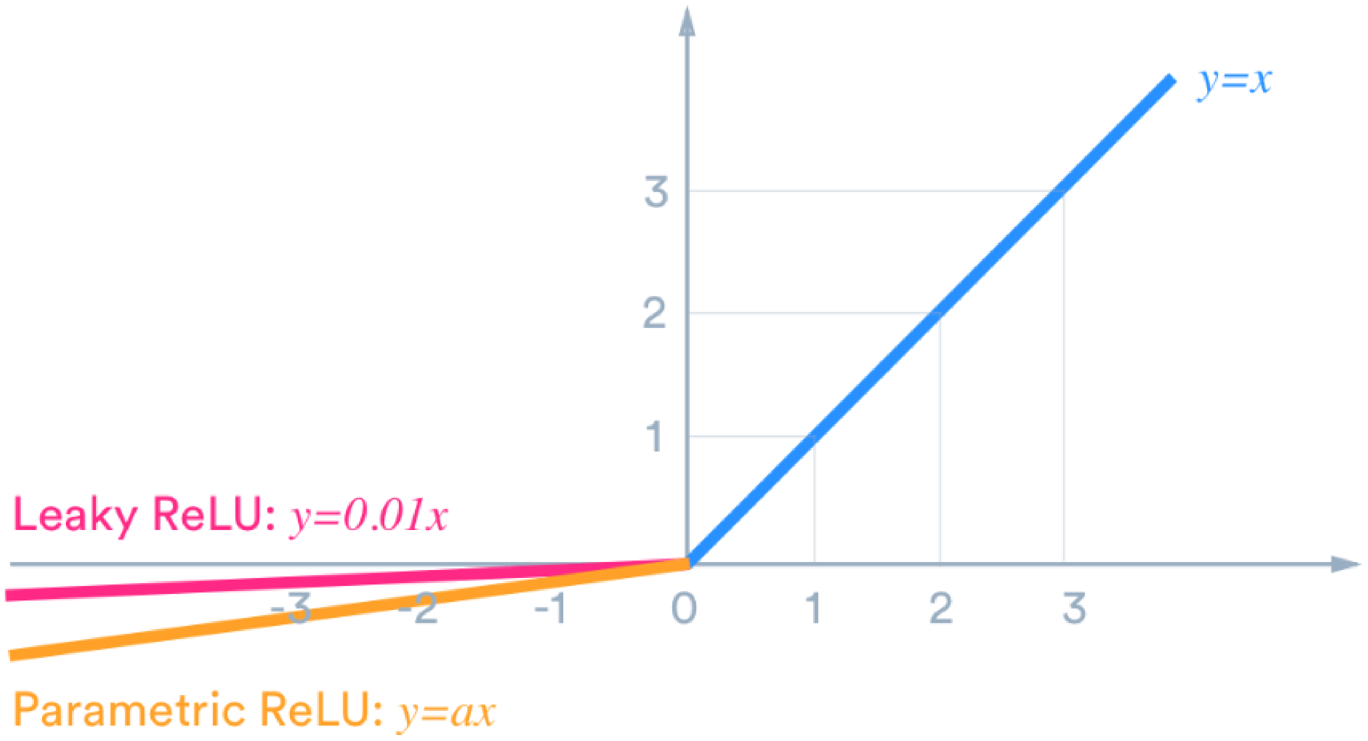


Super-resolution network architecture

- Key architectural concept – advanced nonlinear ‘interpolation’ through learning.
- Loss function – combination of ‘pixel-to-pixel’ MSE loss & AI-based “visual loss”
- JPEG images – Additional preprocessing network for de-blocking



Another non-linearity: Leaky ReLU



Leaky ReLU

Python code for x4 network

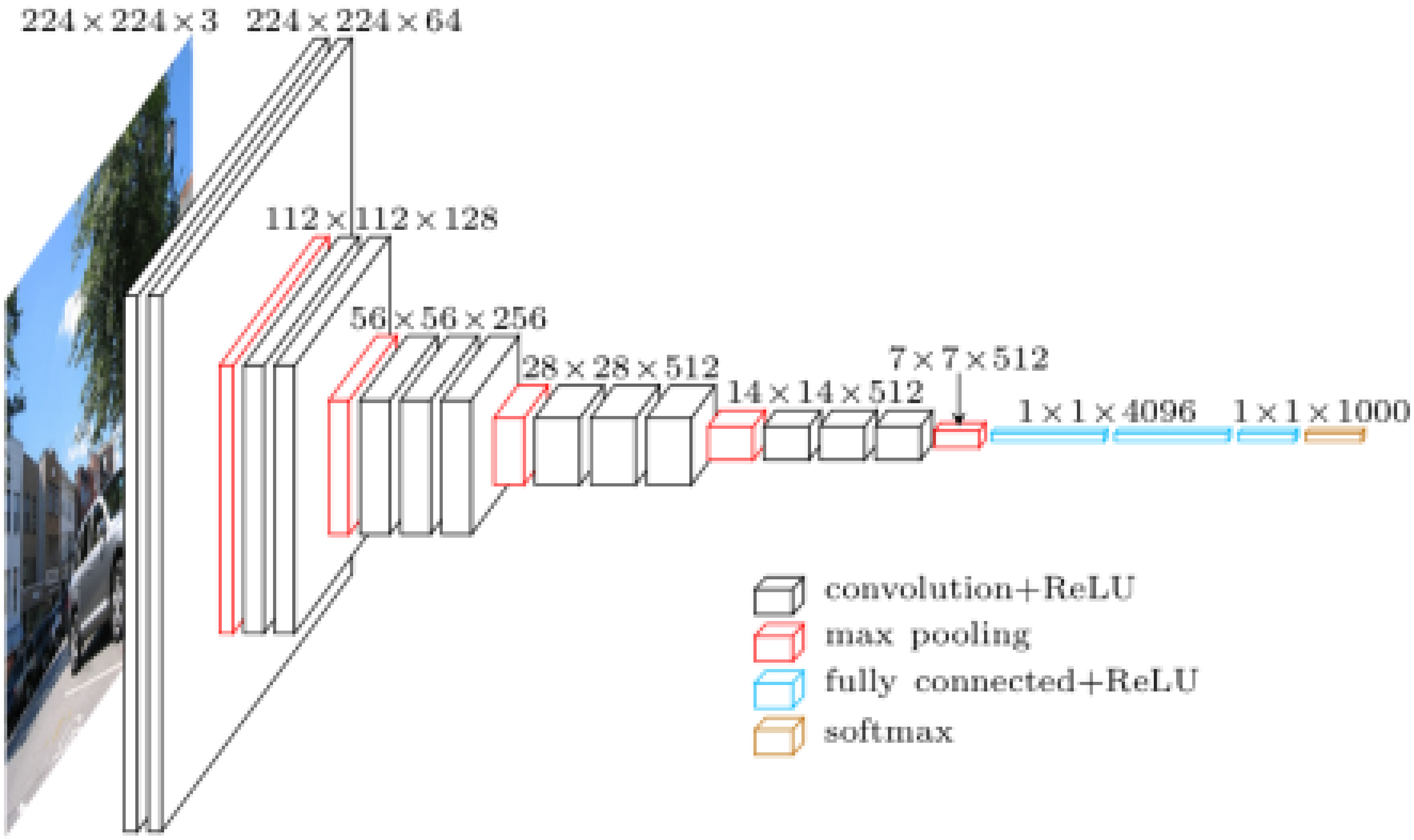
```
def net_subpixel(input_shape, r=4, activation='sigmoid', num_channels=3, **kwargs):
    input = Input(shape=input_shape, name='input')
    l1 = Conv2D(64, (5, 5),
                strides=(1, 1),
                padding='same',
                activation=None,
                name='conv1')(input)
    l1 = LeakyReLU(0.2)(l1)
    l2 = Conv2D(64, (5, 5),
                strides=(1, 1),
                padding='same',
                activation=None, name='conv2')(l1)
    l2 = LeakyReLU(0.2)(l2)
    l3 = Conv2D(int(num_channels * r * r), (5, 5), activation=None, padding='same',
                name='subp3')(l2)

    out = Activation(activation, name='gen_out')(l3)
    # if activation == 'relu':
    #     out = Lambda(lambda_clip)(out)

    out = Lambda(reshape_subpixel, arguments={'r': r, 'num_channels': num_channels})(out)

    model = Model(inputs=input, outputs=out)
```

Pre-trained VGG Net (2015)



VGG content Loss

- Select a layer j (e.g. $j=3$). It has a 'feature map' of dimension

$$\underset{\text{\# Feature maps}}{C_j} \times \underset{\text{Feature maps height}}{H_j} \times \underset{\text{Feature maps width}}{W_j}$$

- Let $\phi_j(x)$ be the 'feature map' activation of any input image x $j_1 \le j \le j_2$
- The content loss uses pre-determined shallow layers

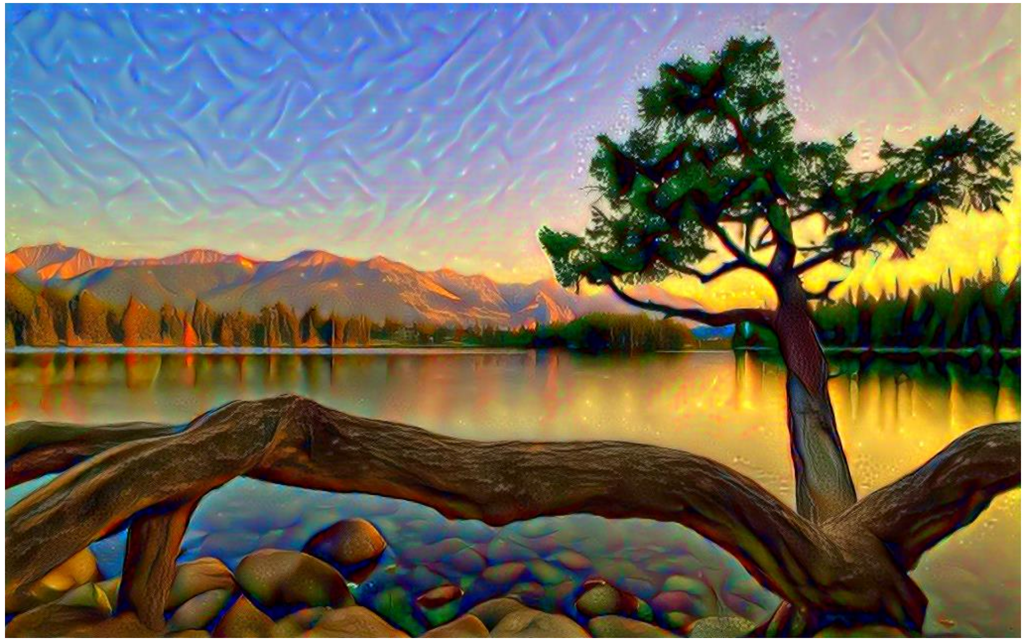
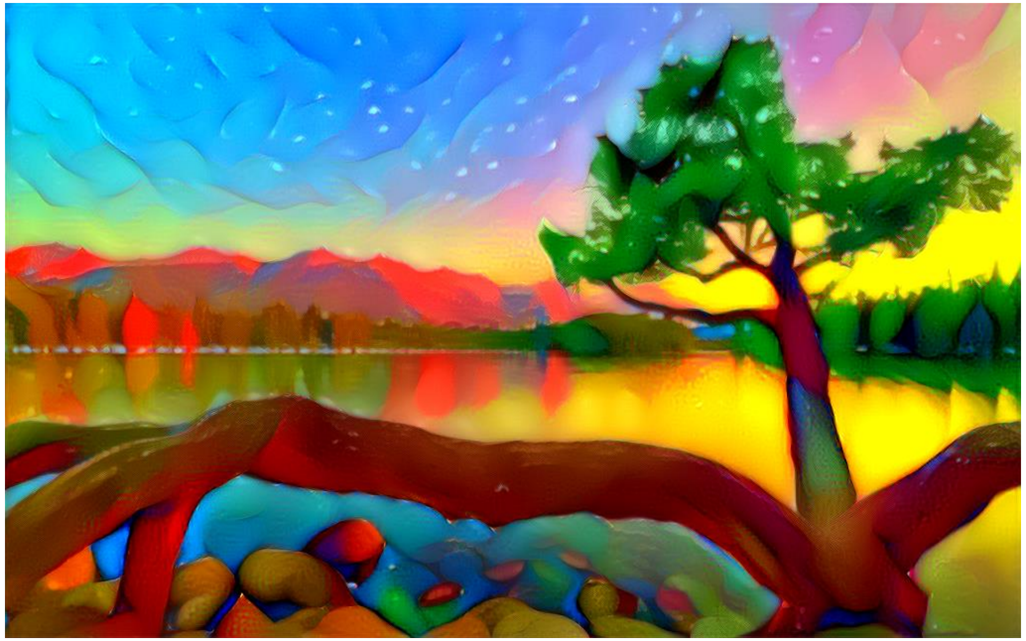
$$Loss_{\text{content}}(x,\hat{x}) := \sum_{j=j_1}^{j_2} \sum_{(c,h,w)_j} \left(\phi_j(x)(c,h,w) - \phi_j(\hat{x})(c,h,w)\right)^2$$

$$Loss_{\text{MSE}}(x,\hat{x}) := \frac{1}{\#\text{pixels}} \sum_i (x_i - \hat{x}_i)^2$$

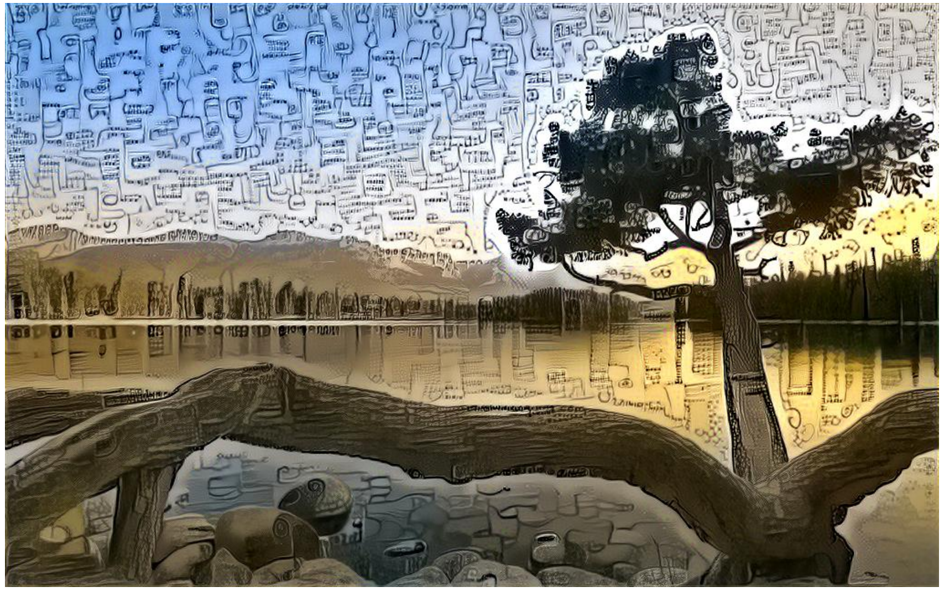
$$Loss(x_{\text{Net Out}}, x_{\text{GT}}) := Loss_{\text{MSE}}(x_{\text{Net Out}}, x_{\text{GT}}) + \lambda Loss_{\text{content}}(x_{\text{Net Out}}, x_{\text{GT}})$$

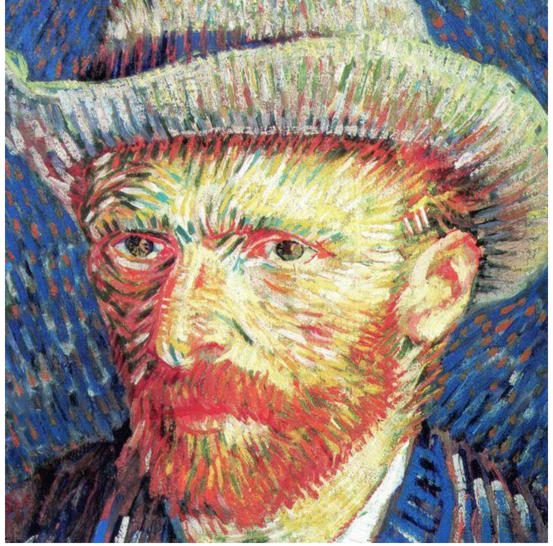
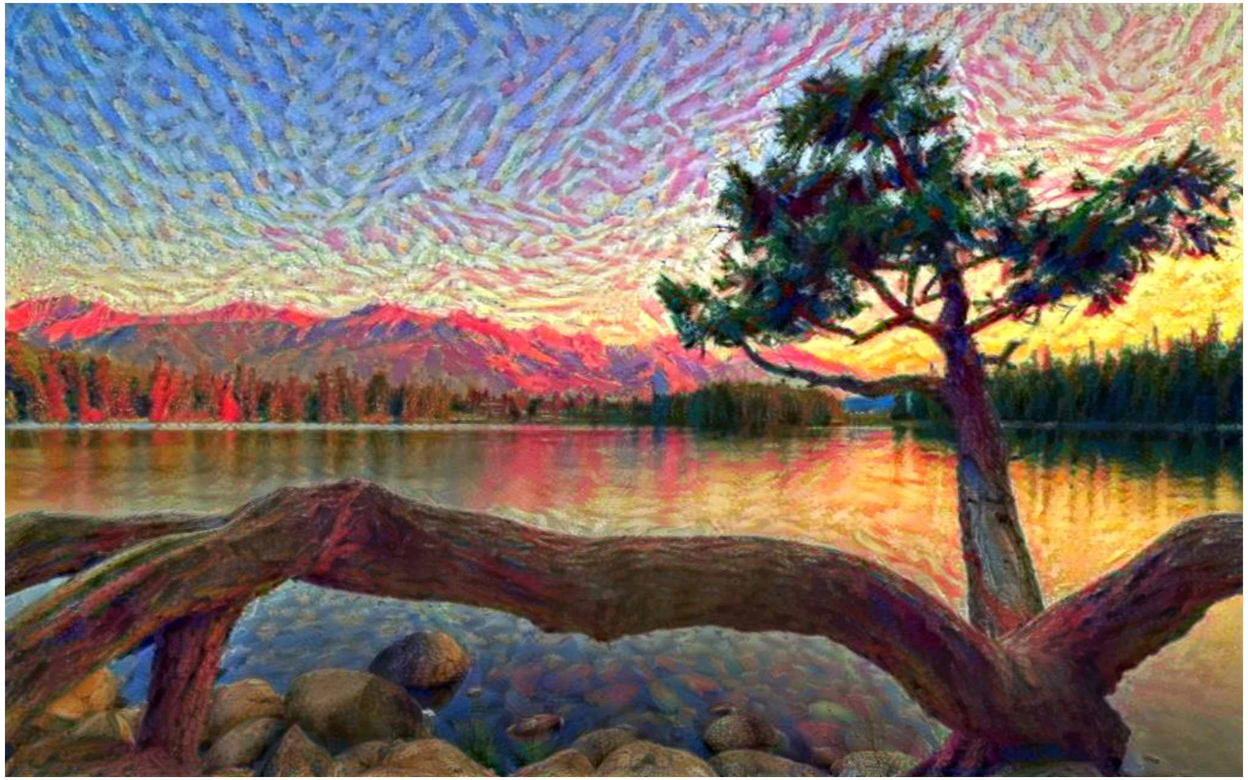
Style Transfer

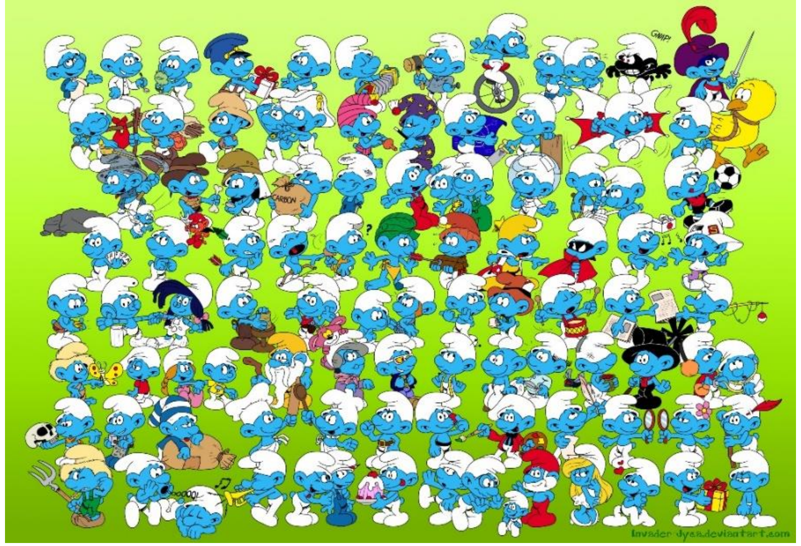


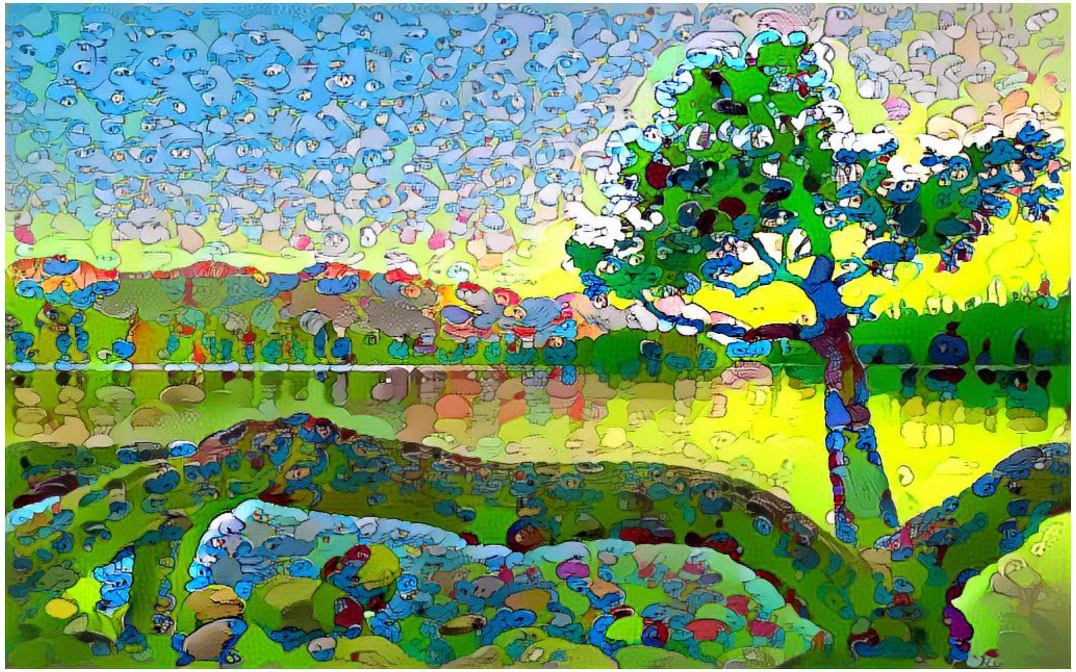



Etsy
POLER
PICA
IKEA
APRISA
Brunch
TARGET
COUCH
MEGA MILLIONS
CHURCH
jetBlue
Miso Soup
ZUCCHINI MUFFINS
SPARKLE CARDIGAN
Croque Madame
I LOVE TV
Kiehls
OMSI
KEEP IT CLEAN
HOT DOG
Gradekeeper
Old Fashioned
GLOVES
FAT ROLL





Starting point (CVPR 2016)

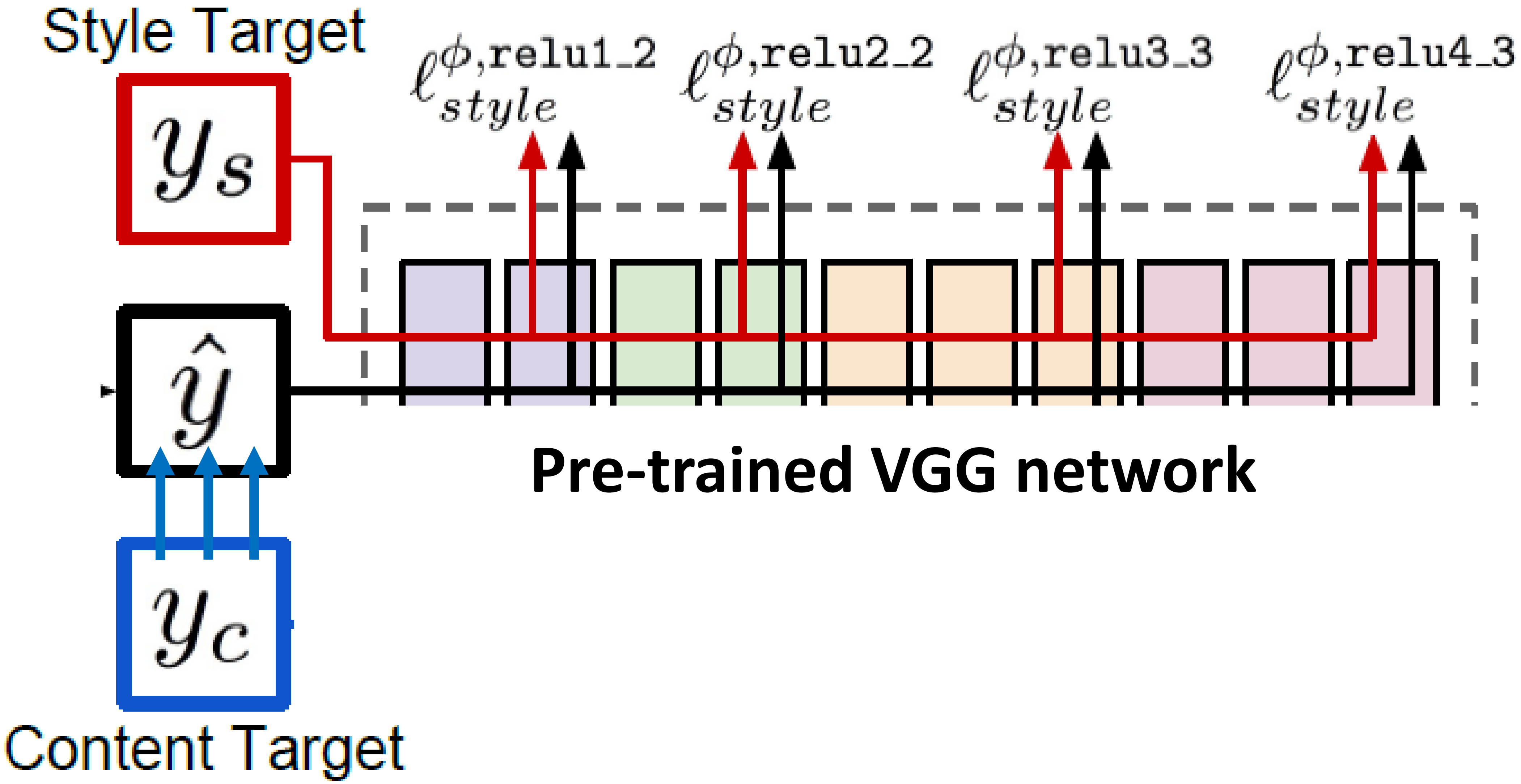
Image Style Transfer Using Convolutional Neural Networks

Leon A. Gatys
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Graduate School of Neural Information Processing, University of Tübingen, Germany
leon.gatys@bethgelab.org

Alexander S. Ecker
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Max Planck Institute for Biological Cybernetics, Tübingen, Germany
Baylor College of Medicine, Houston, TX, USA

Matthias Bethge
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Max Planck Institute for Biological Cybernetics, Tübingen, Germany

Style Transfer Architecture



VGG Style Loss

- Select a layer j (e.g. $j=3$). It has a 'feature map' of dimension

$$C_j \times H_j \times W_j$$

- Let $\phi_j(x)$ be the 'feature map' activation of any input image x
- We compute the Gram matrix of the feature pairs (c,c')

$$G_j(x)(c,c') := \frac{1}{H_j \times W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)(c,h,w)\phi_j(x)(c',h,w)$$

- The style loss function with a reference style image y_s is then

$$Loss_{\text{style}}(y_s,\hat{y}) := \sum_{j=j_1}^{j_2} \underset{\text{layer weights}}{w_j} \underbrace{\left\| G_j(y_s) - G_j(\hat{y}) \right\|_F^2}_{\text{Matrix distance}} \qquad \|A\|_F := \sqrt{\sum_{k_1,k_2} a_{k_1,k_2}^2}$$

Fast near-real time style transfer

- For an input content image y_c, initialize $\hat{y}^{(0)} = y_c$

- Run a few ‘uncontrollable’ iterations towards minimization of style loss.

$$Loss_{\text{style}}\left(y_s, \hat{y}\right) := \sum_{j=j_1}^{j_2} \left\| G_j\left(y_s\right) - G_j\left(\hat{y}\right) \right\|_F^2$$

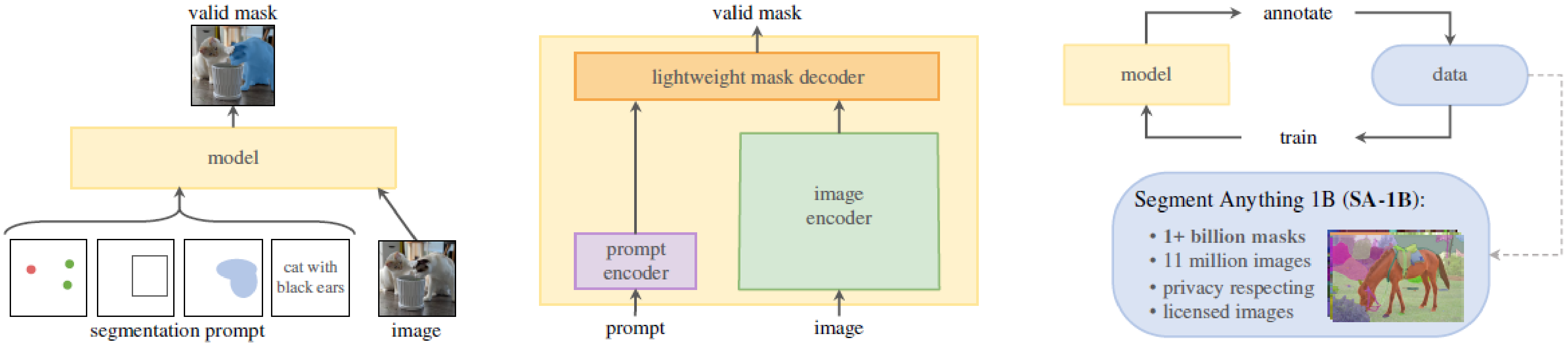
$$\hat{y}^{(k+1)} = \hat{y}^{(k)} + \eta \nabla Loss_{\text{style}}\Big|_{\hat{y}} \left(y_s, \hat{y}^{(k)}\right)$$

AI-based portrait segmentation

Segment Anything

Alexander Kirillov[1,2,4] Eric Mintun[2] Nikhila Ravi[1,2] Hanzi Mao[2] Chloe Rolland[3] Laura Gustafson[3]
Tete Xiao[3] Spencer Whitehead Alexander C. Berg Wan-Yen Lo Piotr Dollár[4] Ross Girshick[4]
[1]project lead [2]joint first author [3]equal contribution [4]directional lead
Meta AI Research, FAIR

5 Apr 2023



200-300 masks

Portrait segmentation - Examples

Portrait segmentation - Examples

Data Curation for Portrait segmentation

- Manual “Ground truth” segmentation - crowdsourcing & designers
- Training set - 30,000 manually segmented portraits
- Testing set – 2,000
- Quality metric - IoU (Intersection / Union) of ground truth & segmentation

Pyramid Scene Parsing Network

Hengshuang Zhao[1] Jianping Shi[2] Xiaojuan Qi[1] Xiaogang Wang[1] Jiaya Jia[1]
[1]The Chinese University of Hong Kong [2]SenseTime Group Limited
{hszhao, xjqi, leojia}@cse.cuhk.edu.hk, xgwang@ee.cuhk.edu.hk, shijianping@sensetime.com

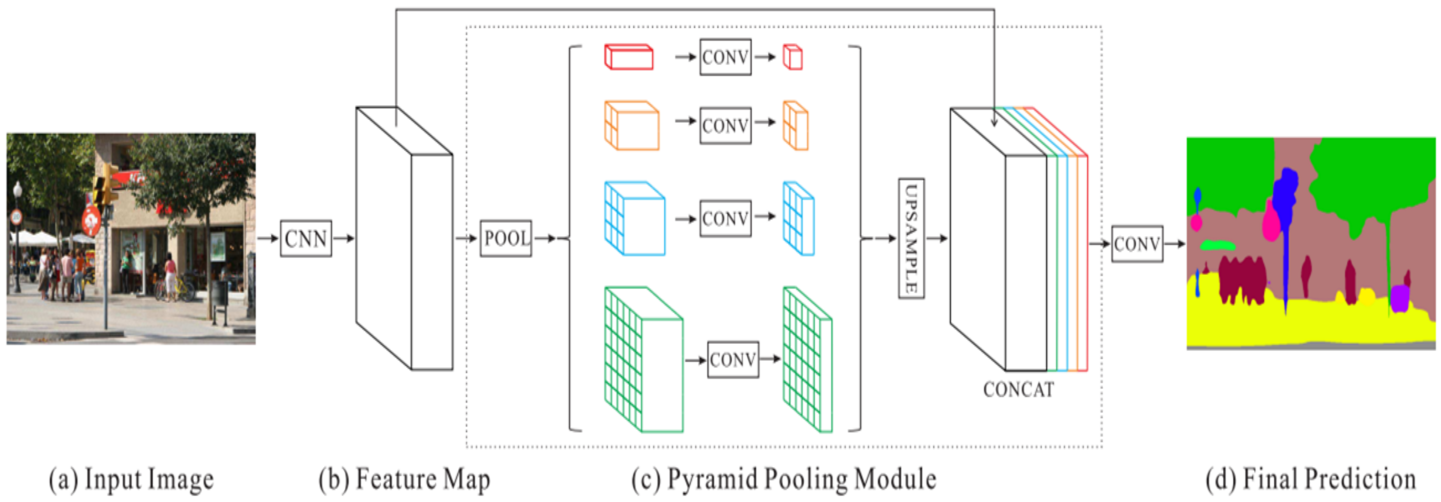
Abstract

Scene parsing is challenging for unrestricted open vocabulary and diverse scenes. In this paper, we exploit the capability of global context information by different-region-based context aggregation through our pyramid pooling module together with the proposed pyramid scene parsing network (PSPNet). Our global prior representation is effective to produce good quality results on the scene parsing task, while PSPNet provides a superior framework for pixel-level prediction. The proposed approach achieves state-of-the-art performance on various datasets. It came first in ImageNet scene parsing challenge 2016, PASCAL VOC 2012 benchmark and Cityscapes benchmark. A single PSPNet yields the new record of mIoU accuracy 85.4% on PASCAL VOC 2012 and accuracy 80.2% on Cityscapes.

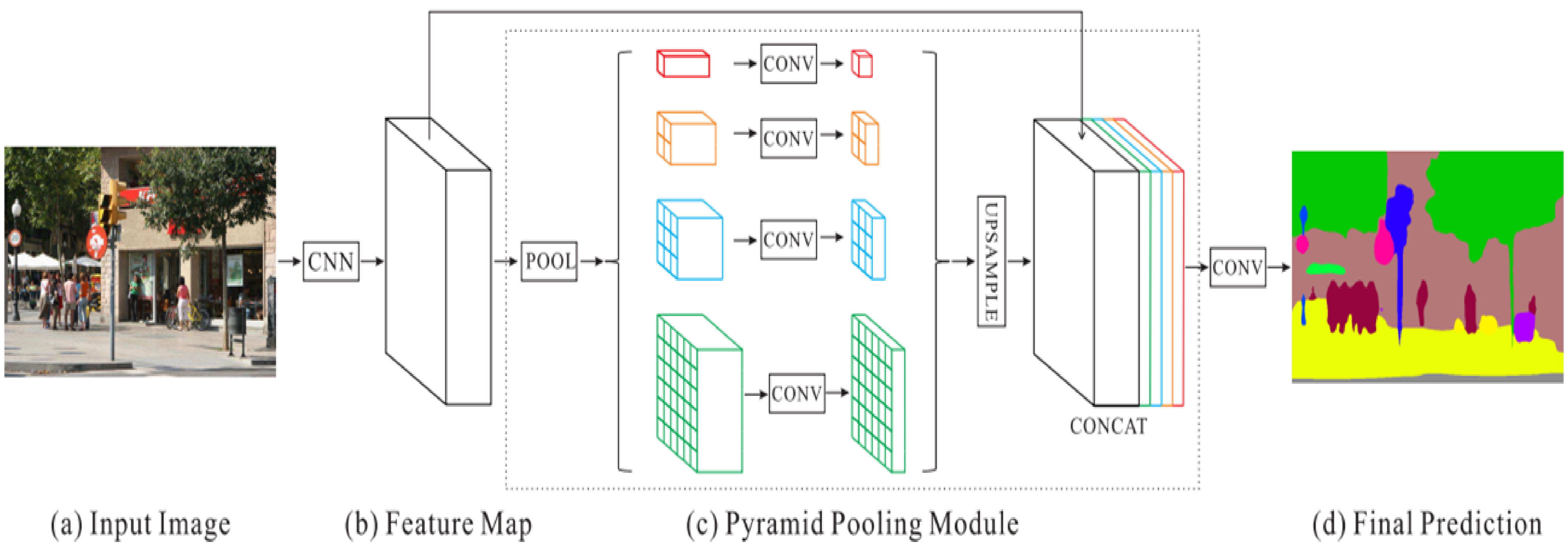
(a) Image (b) Ground Truth

Figure 1. Illustration of complex scenes in ADE20K dataset.

Portrait segmentation pipeline



The Pyramid Scene Parsing Network (PSPNet)



Loss functions for segmentation

Let $p = (p_1, \ldots, p_n)$ be the ground truth labels of the segmentation. This means that $p_i \in \{0,1\}$.

Let $\tilde{p} = (\tilde{p}_1, \ldots, \tilde{p}_n)$ be an approximation (e.g., as generated through a deep learning network) of the segmentation, with $0 \leq \tilde{p}_i \leq 1$. One can ensure that the network outputs 'probability pixels' by applying at the last layer at each pixel the logistic function

$$\sigma(t) := \frac{1}{1+e^{-t}}.$$

Negative log-likelihood as a loss for image segmentation

We can define a loss per image

$$-\sum_{i=1}^{n} p_i \log \tilde{p}_i + (1-p_i)\log(1-\tilde{p}_i)$$

Jaccard loss for segmentation

It is clear that in unbalanced cases, where there are more ground truth background pixels than ground truth object pixels, the standard negative log-likelihood loss is potentially biased.

One would like to train a DL segmentation network and measure/maximize performance using Intersection Over Union (IoU) loss, which is also called the **Jaccard loss**

$$0 \le \frac{|A \cap B|}{|A \cup B|} \le 1 .$$

The Jaccard index in our special is

$$J(p, \tilde{p}) = \frac{\#\{i : p_i = 1 \text{ and } \tilde{p}_i \ge 0.5\}}{\#\{i : p_i = 1 \text{ or } \tilde{p}_i \ge 0.5\}} \ .$$

Since the Jaccard index is not differentiable, we look for a surrogate. First note that

$$\frac{|A\cap B|}{|A\cup B|}=\frac{|A\cap B|}{|A|+|B|-|A\cap B|},$$

Let

$$\langle p,\tilde{p}\rangle=\sum_{i=1}^{n}p_i\tilde{p}_i\,,\qquad |p|_1=\sum_{i=1}^{n}p_i\,,\qquad |\tilde{p}|_1=\sum_{i=1}^{n}\tilde{p}_i\,.$$

We define a differentiable ‘loss’ function that we wish to maximize

$$\tilde{J}(p,\tilde{p}):=\frac{\langle p,\tilde{p}\rangle}{|p|_1+|\tilde{p}|_1-\langle p,\tilde{p}\rangle}\,.$$

Note that in the special case where $\tilde{p}_i\in\{0,1\}$, $i=1,\ldots,n$, we get

$$\tilde{J}(p,\tilde{p})=J(p,\tilde{p})\,.$$

The log-likelihood and Jaccard are typically combined with a weight.

Auto - correction

BEFORE
(go to next slide for after)

AFTER

Phase Retrieval

(example for a machine learning approach to inverse problems)

* S. Dekel and L. Gugel, PR-DAD: Phase Retrieval Using Deep Auto-Decoders, *7th International Conference on Frontiers of Signal Processing (ICFSP)*, 2022, pp. 165-172

The Fourier transform

$\mathcal{F}$ 

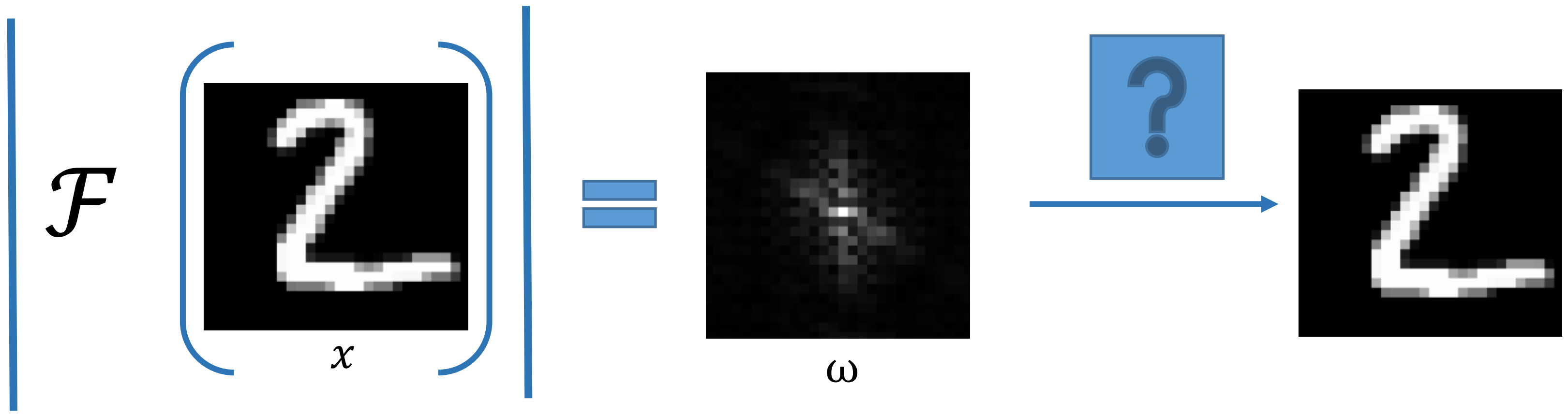
- The Fourier transform is an invertible transform to a frequency representation

- Continuous form $\mathcal{F}(f)(w) = \hat{f}(w) := \int_{\mathbb{R}^n} f(y) e^{-i\langle y,w \rangle} dy, \quad w \in \mathbb{R}^n.$

- Discrete univariate form for input $\{f_j\}_{j=0}^{N-1}$ $\quad \hat{f}_k = \sum_{j=0}^{N-1} f_j e^{-\frac{i2\pi}{N}kj}$

- Note that the coefficients are complex, even if the samples are real
- Euler form of a complex $z = re^{i\theta}, \quad r \in \mathbb{R}_+$ is the magnitude.

Phase Retrieval: problem formulation



- $x \in \mathbb{R}^{n \times n}$, where $\mathcal{F}(x) = \omega\, e^{-i\varphi}$, ω magnitude and φ phase of Fourier transform
- PR is an ill-posed inverse problem: recover x from ω

How critical is the phase?

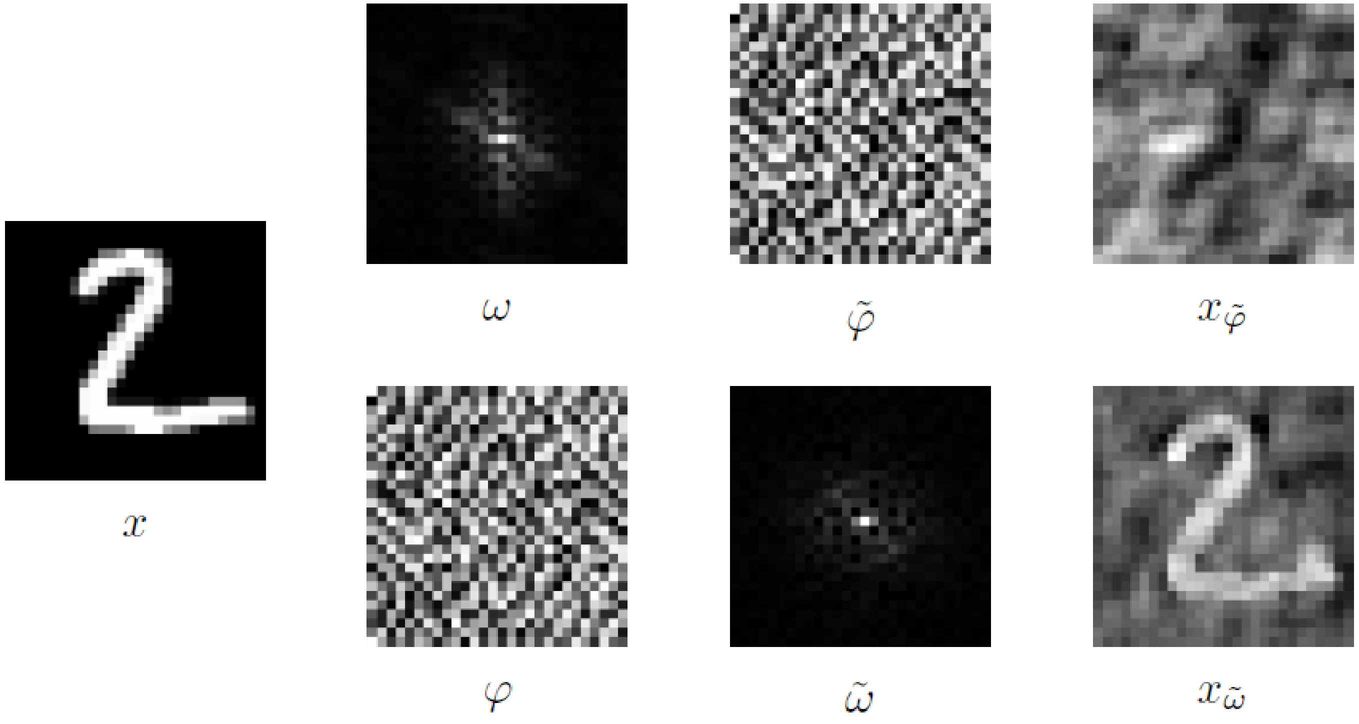
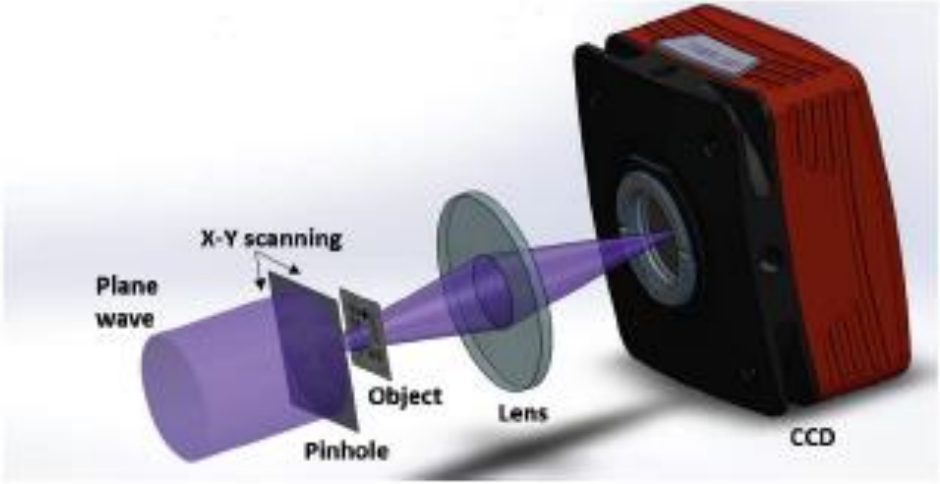
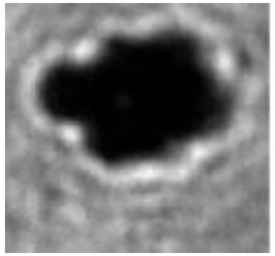


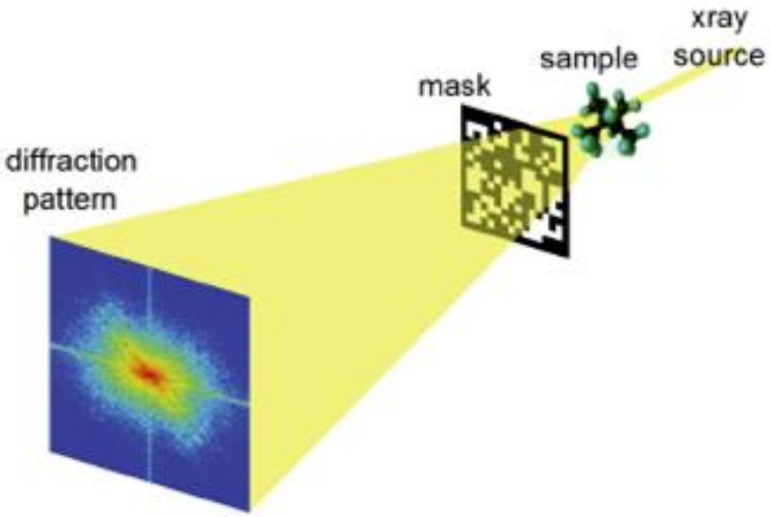
Fig. 1. Most information about the image is contained in the phase, which can be demonstrated by exchanging the phase with a random phase. For comparison we also exchange the magnitude with a random magnitude. Original image x, original magnitude ω, random phase $\tilde{\varphi}$, image obtained by combining the original magnitude and the random phase $x_{\tilde{\varphi}}$, original phase φ, random magnitude $\tilde{\omega}$, image obtained by combining the original phase and the random magnitude $x_{\tilde{\omega}}$.

Use cases in imaging

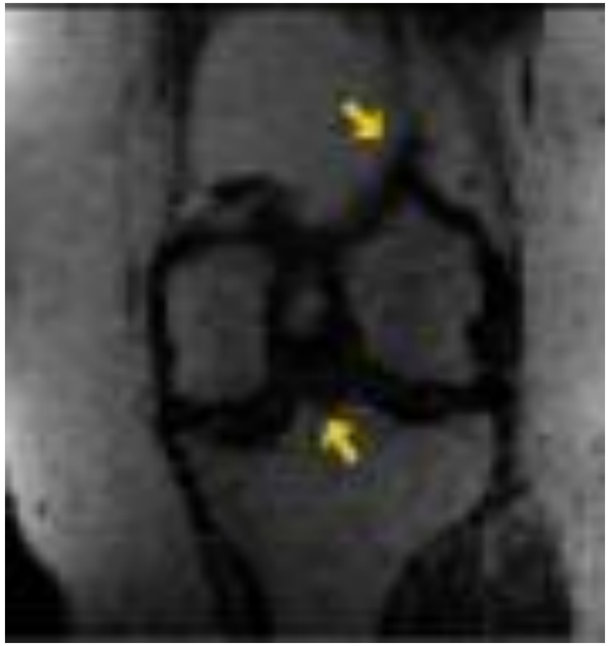
Electro-microscopy images







Fast MRI



Alternating projection-based algorithm

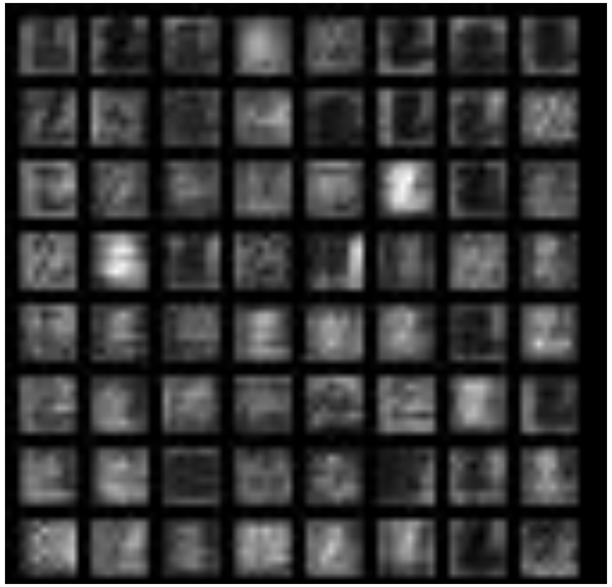
- F_M - DFT matrix ($M \times M$) , P_N - zero padding matrix (N$\times M$) for $x \in \mathbb{C}^N$
- Input: DFT magnitude ω, random initialization of x_0

Loop

- $\hat{x}_i = F_M P_N x_i$
- Compute current phase $u_i = \frac{\hat{x}_i}{|\hat{x}_i|}$
- Replace magnitude by ground truth magnitude $z_i = F_M^T (\omega \circ u_i)$, where $\circ$ is the element-wise product
- Enforce known constraints on solution $x_i = \mathcal{H}(z_i)$. For example, projection onto real non-negative values
- $i = i+1$

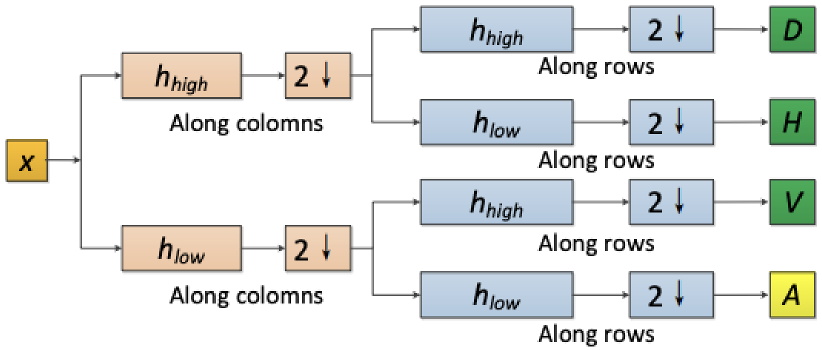
PR-DAD – Finding a sparse encoding space of the image class

- Our algorithm is based on finding good sparse representations for the given image class.
- Sparsity of a representation – $T(x) = \{T_j(x)\}, \quad \|T(x)\|_{l_1} = \sum_j |T_j(x)|$
- The l_1 norm "approximates" the l_0 that counts non-zero elements.
- In our work we show two options/architectures for T: fixed transform, trained encoder



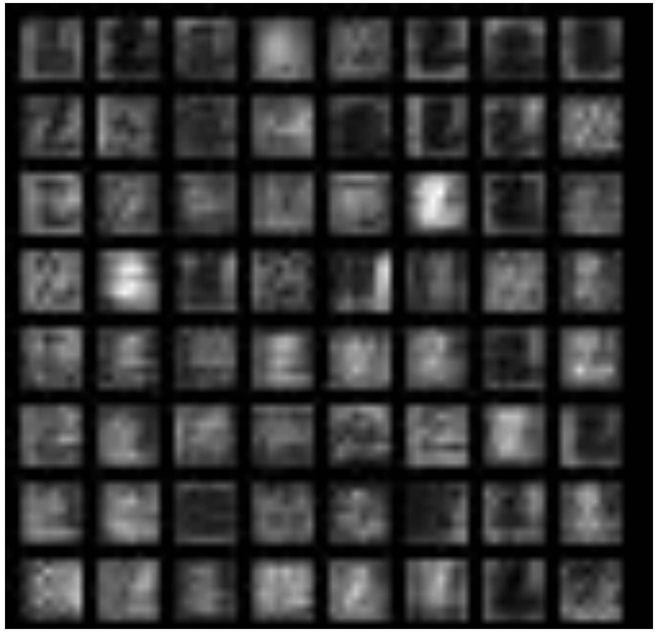
PR-DAD I: Phase Retrieval Using un-supervised Haar wavelet packet transform

- Encoding space – coefficients of a <u>fixed(!)</u> wavelet packet transform.
- We assume the image class is <u>sparse(!)</u> in this representation.
- Projections onto representation are linear.
- The decoder is then the inverse wavelet packet transform → plugged as a component of the PR network.

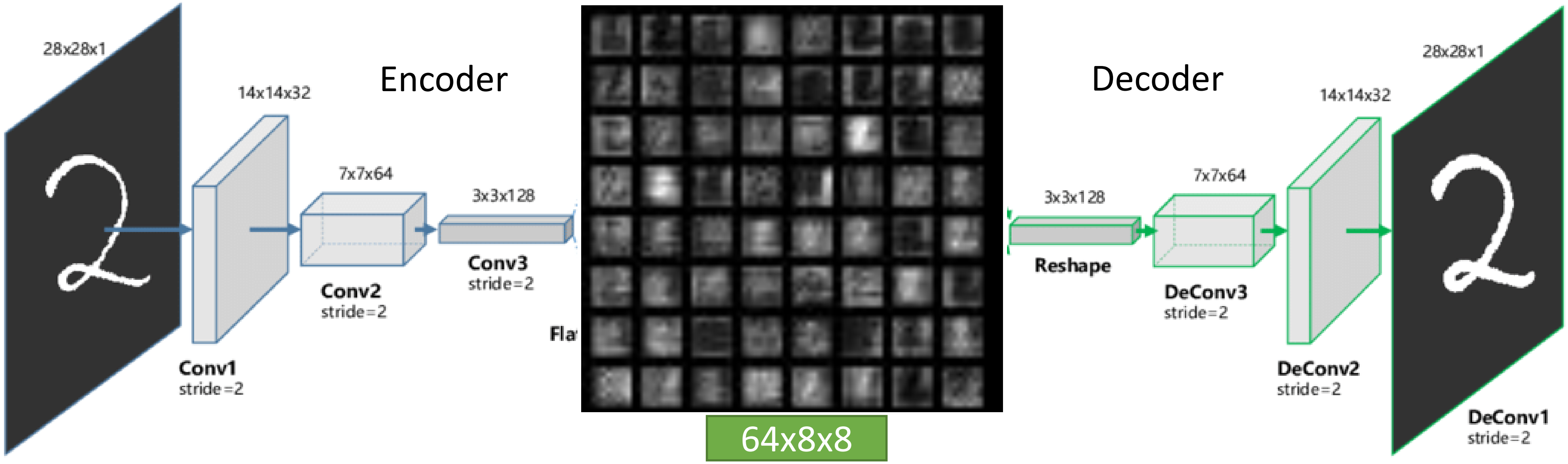


PR-DAD II: Phase Retrieval Using Deep Auto-Decoder

- Encoder – decoder architecture first “learns” the image class from existing ground truth images.
- We use augmentation or add synthetic data if not enough ground truth images are available.
- Encoding space – over-sampled representation composed of low resolution components.
- We train(!) the encoding representation to be sparse.
- Projections onto representation are non linear using an NN encoder.
- The decoder is then then plugged as part of the PR network.



PR-DAD II: Phase Retrieval Using Deep Auto-Decoder



- **ConvDoubleBlock**
  - **Conv2D**: kernel=3x3, stride=1, padding=replicate/zeros
  - **2DBatchNormalization**
  - **Non-linear activation:** ReLu/Prelu
  - ...repeat once more

- **DownConvBlock:**
  - ConvDoubleBlock
  - Average Pooling: kernel=2x2

Encoder:

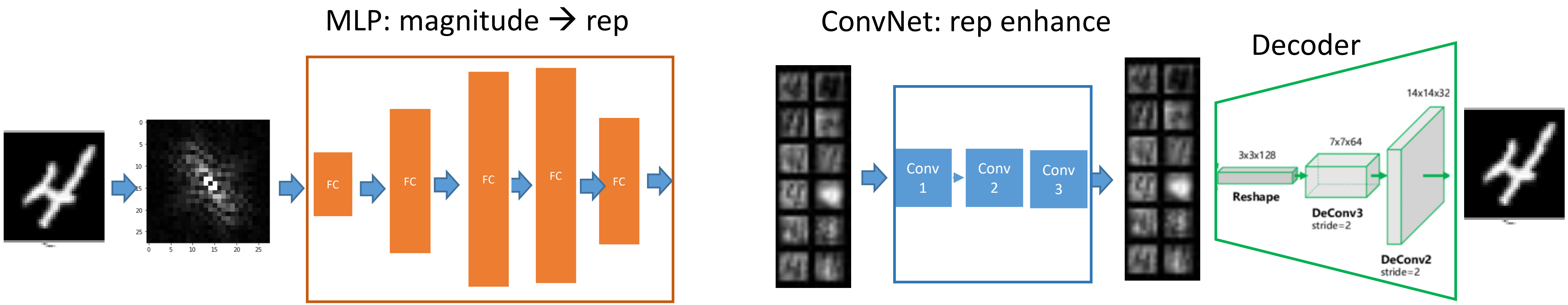
- DownConvBlock #1
- DownConvBlock #2
- DownConvBlock #3

- **UpConvBlock:**
  - UpSampling: bilinear interpolation, scale x2
  - ConvDoubleBlock

Decoder:

- UpConvBlock #1
- UpConvBlock #2
- UpConvBlock #3

Phase Retrieval network



Fully connected block:

- Linear transform with bias Ax_{in}+B
- 2D Batch normalization
- Non linear activation: PReLU

$$\mathrm{PReLU}_a(x) := \begin{cases} x, & x \geq 0, \\ ax, & x < 0. \end{cases}$$

$$\mathrm{PReLU}_0(x) = \mathrm{ReLU}(x)$$

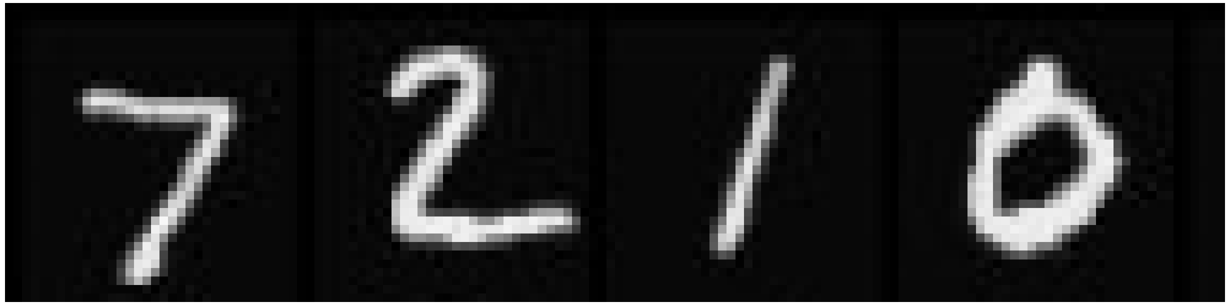
MLP (Multi Layer Perceptron):

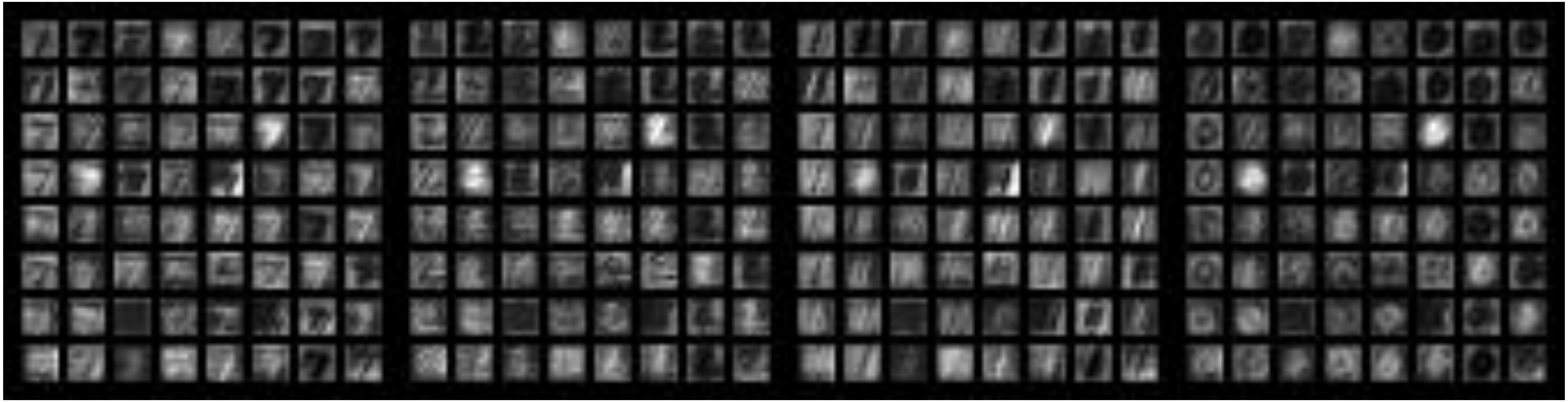
- Fully connected block #1: scale factor x2
- Fully connected block #2: scale factor x2
- Fully connected block #3: output flatten features map size

ConvNet:

- Double Conv block #1
- Double Conv block #2
- Double Conv block #3

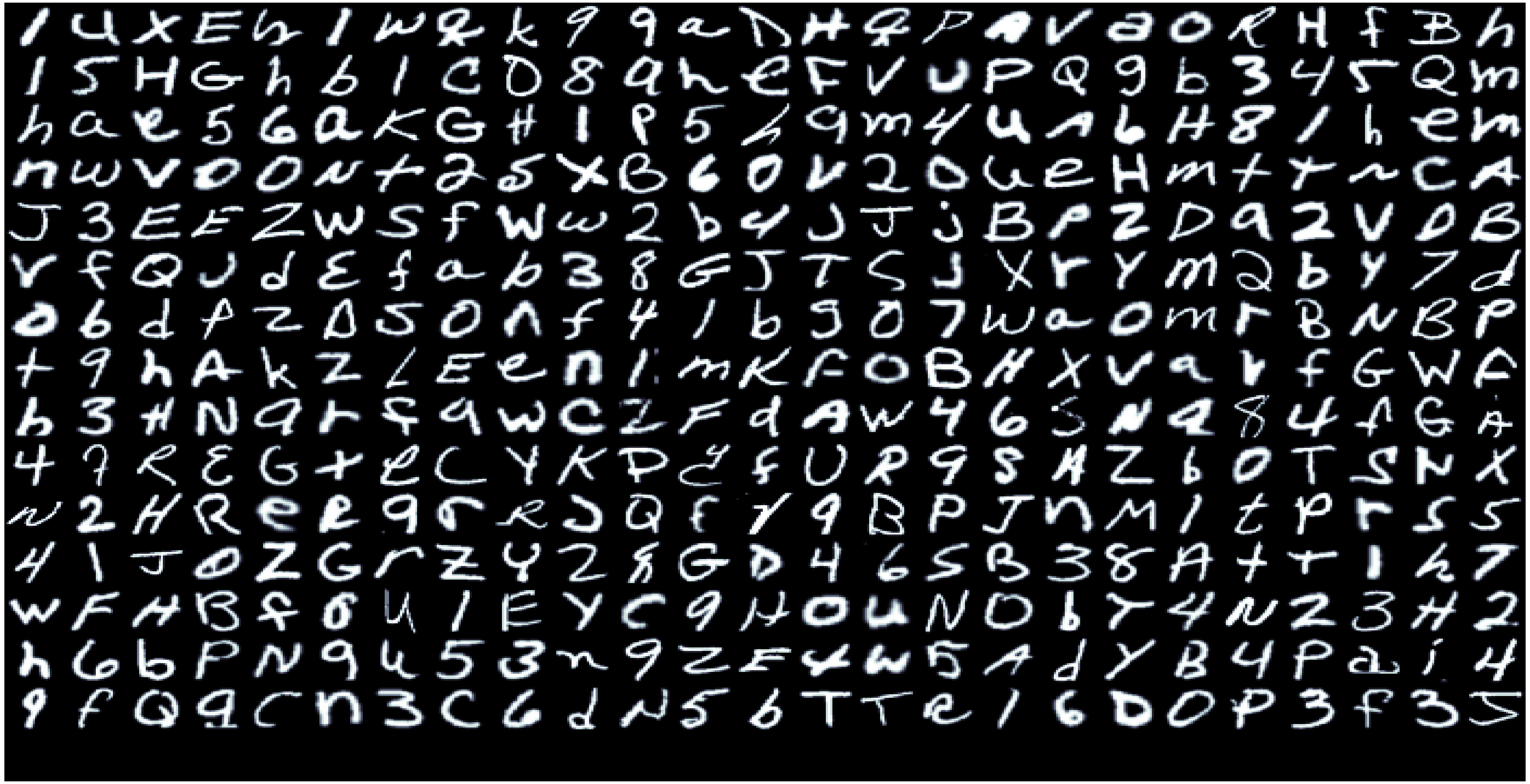
MNIST encoder features





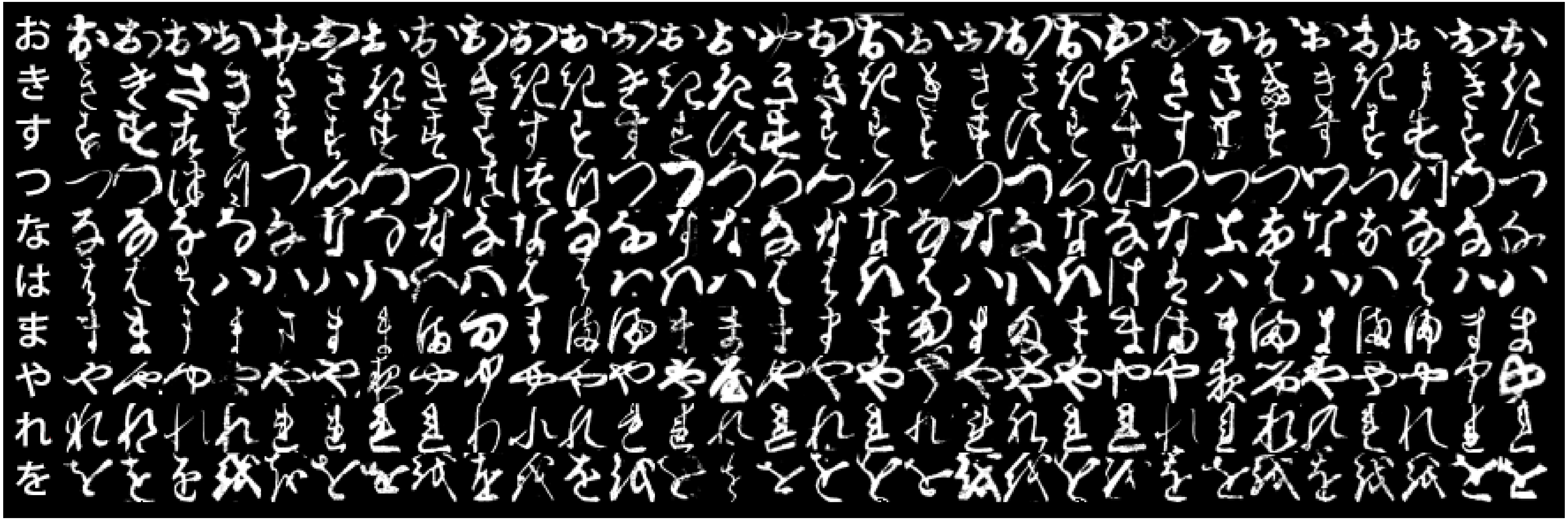
EMNIST

EMNIST Balanced:
28x28 gray scale 131K images , 47 classes, train: 112.8 K, test: 18.8K



KMNIST

Kuzushiji-MNIST is a drop-in replacement for the MNIST dataset 28x28 grayscale, 70,000 images 10 classes, based on **Kuzushiji Dataset** created by National Institute of Japanese Literature



Fashion MNIST

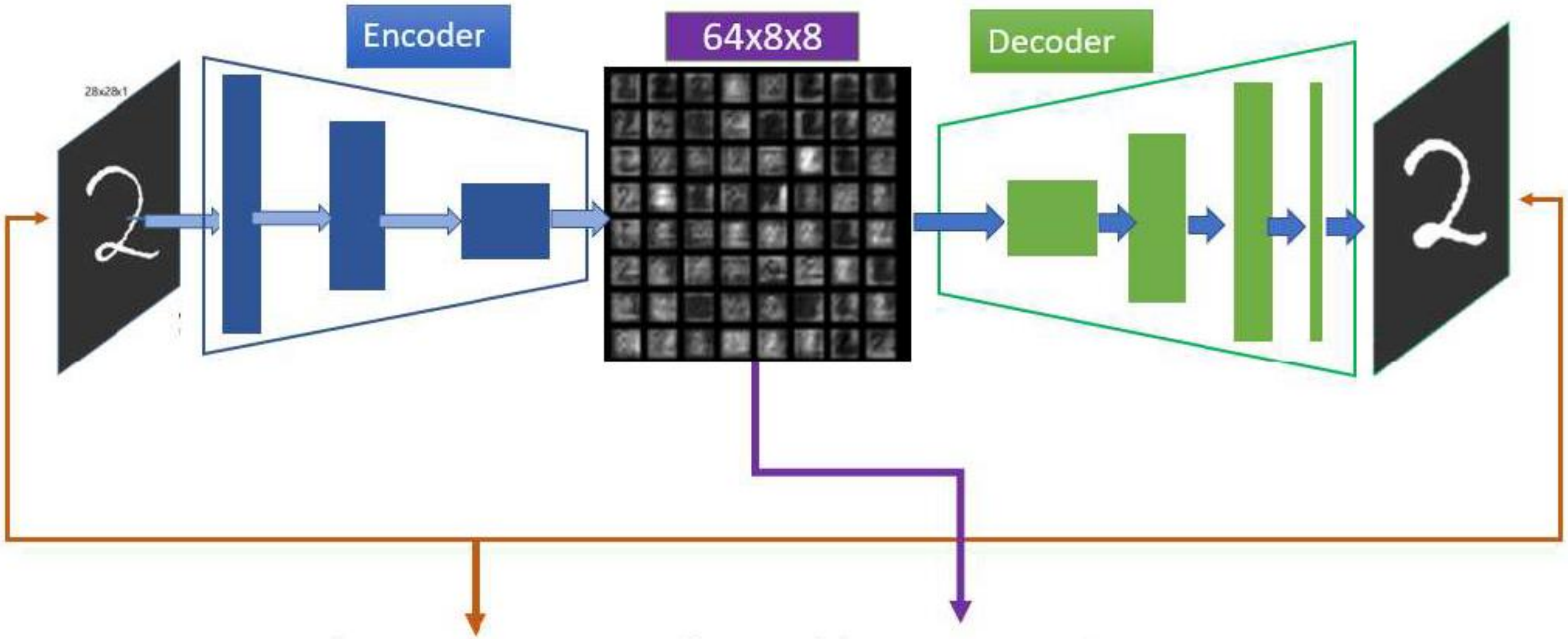
Dataset of Zalando's article 27x27 grayscale images—consisting of a training set of 60K and a test set of 10K.

PR-DAD: data preparation

- MNIST datasets: up-scaling to 32x32
- CelebA facial images:
  - centre cropping to 120x120
  - Up scaling 64x64
  - Map from RGB to Grayscale
- Normalization to range [0, 1]
- Random Augmentation, prob in (0.25, 0.5):
  - Geometric transforms
    - Vertical/Horizontal flipping
    - Free Rotation in range $(-\theta, \theta)$
    - Scaling
  - Color transforms:
    - Sharpness
    - Gaussian Blur
    - Gamma Transform

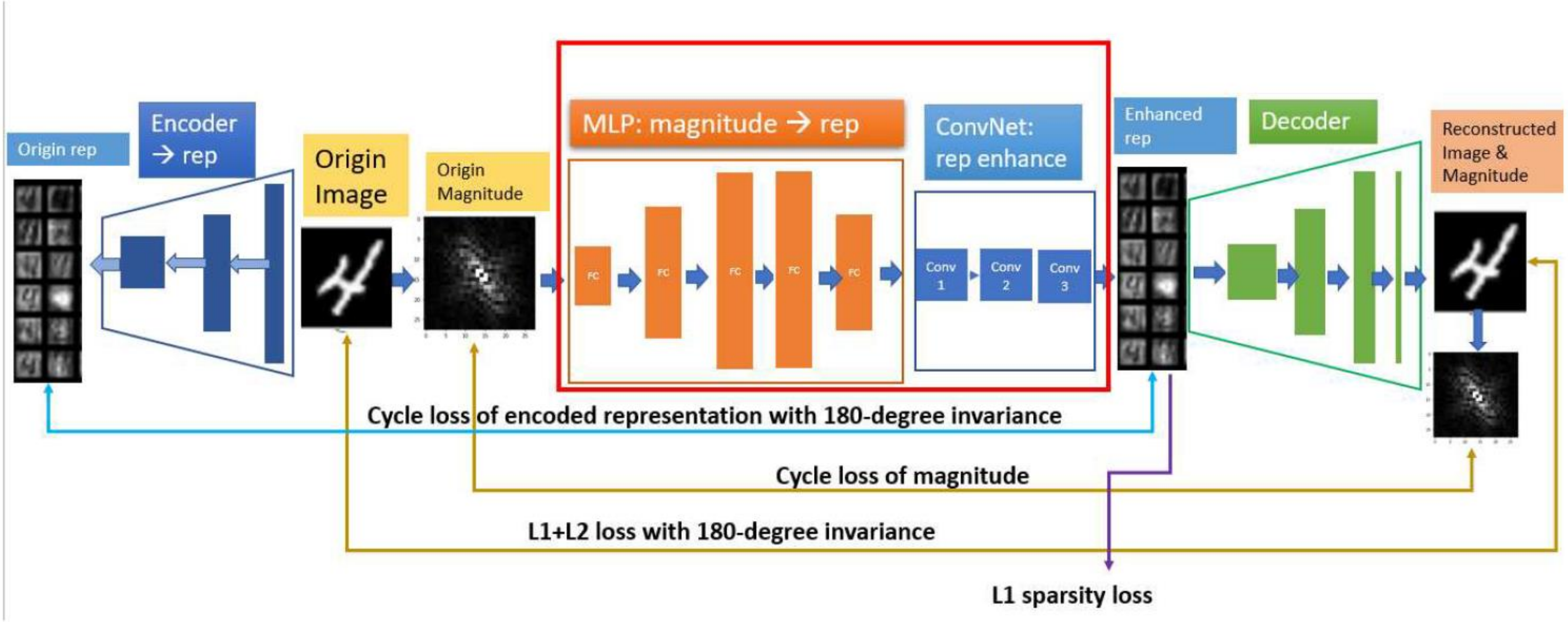
Encoder-decoder loss functions

- L2 loss, rotation 180 invariant
- L1 sparsity of encoded representation

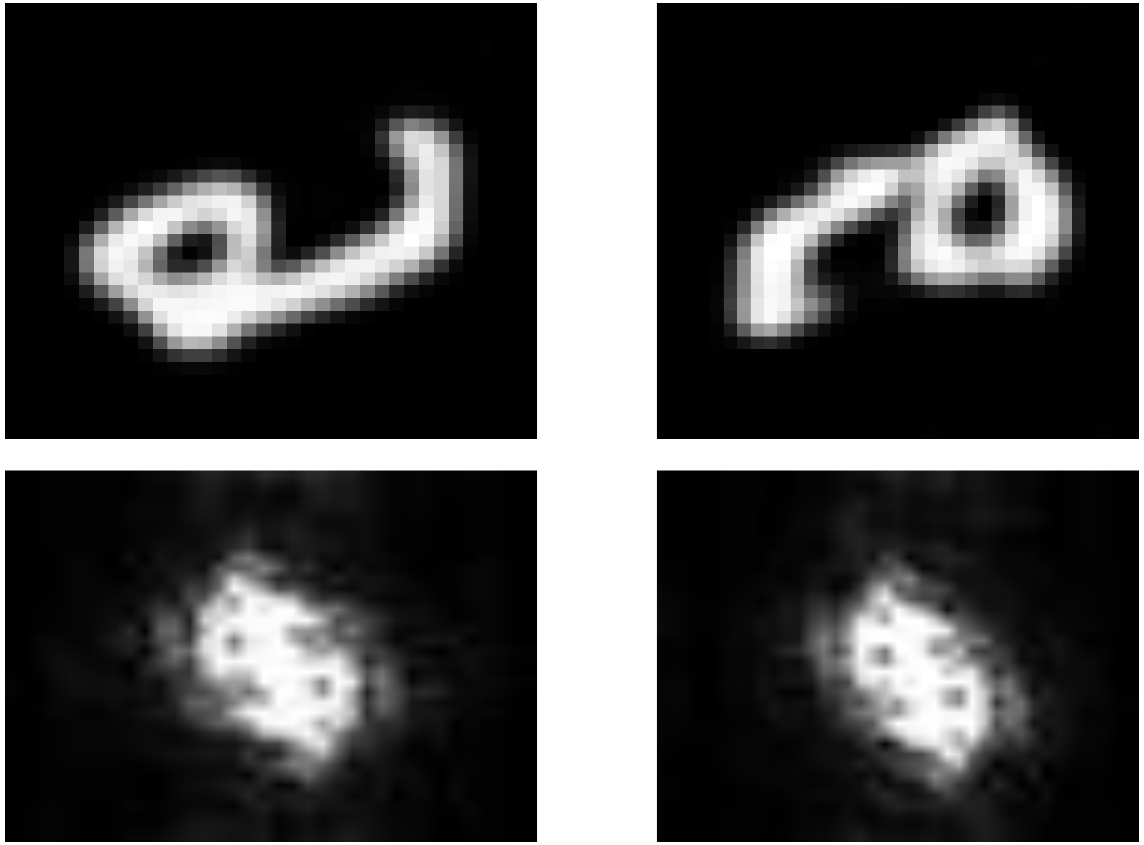


$$|Orig - Recon|_2 + \lambda|Features|_1$$

PR-DAD: Loss functions

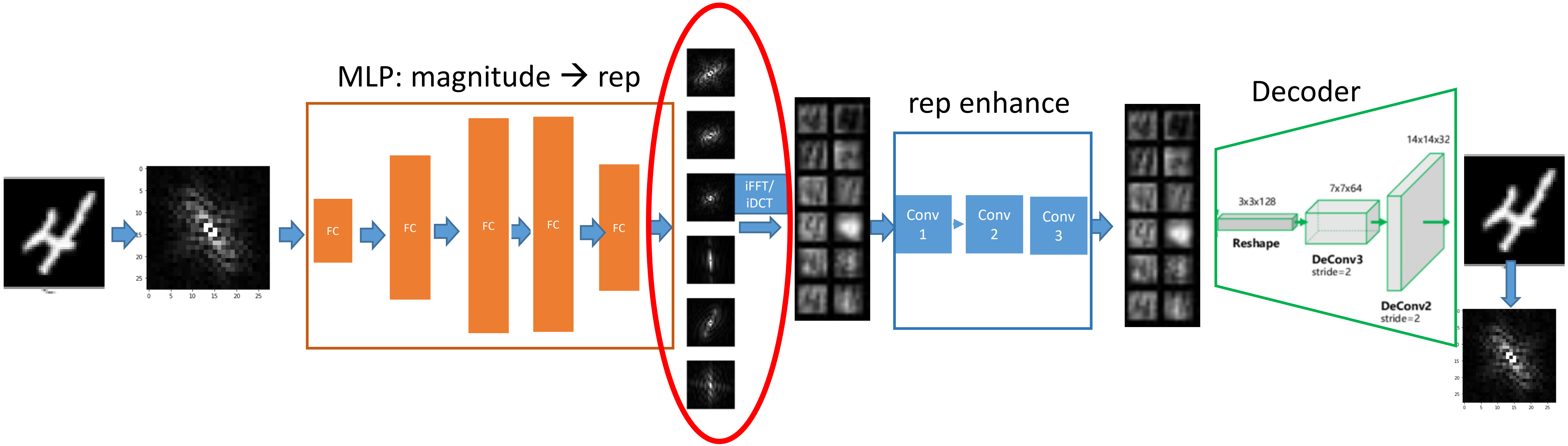


Loss function - 180 rotation invariance



$$loss(x, y) = min(\|x - y\| \quad , \|x - R_{180}(y)\| \quad)$$

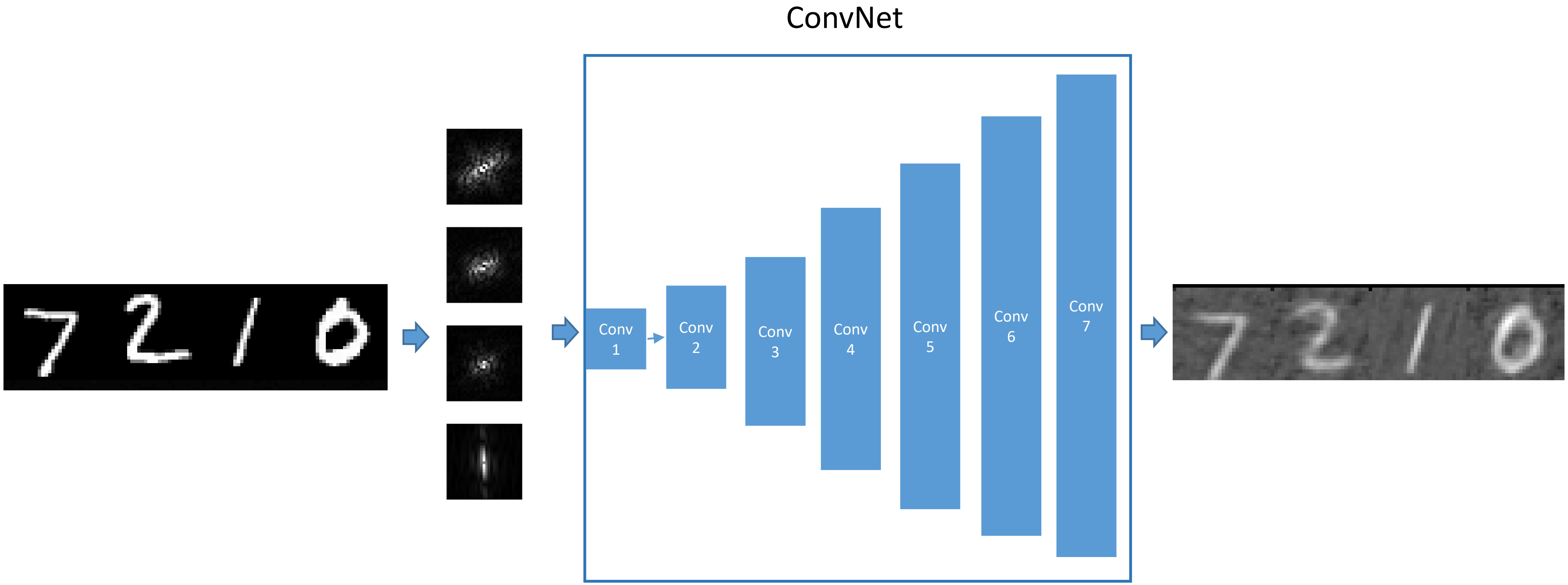
Ablation study: adding iFFT/iDCT component



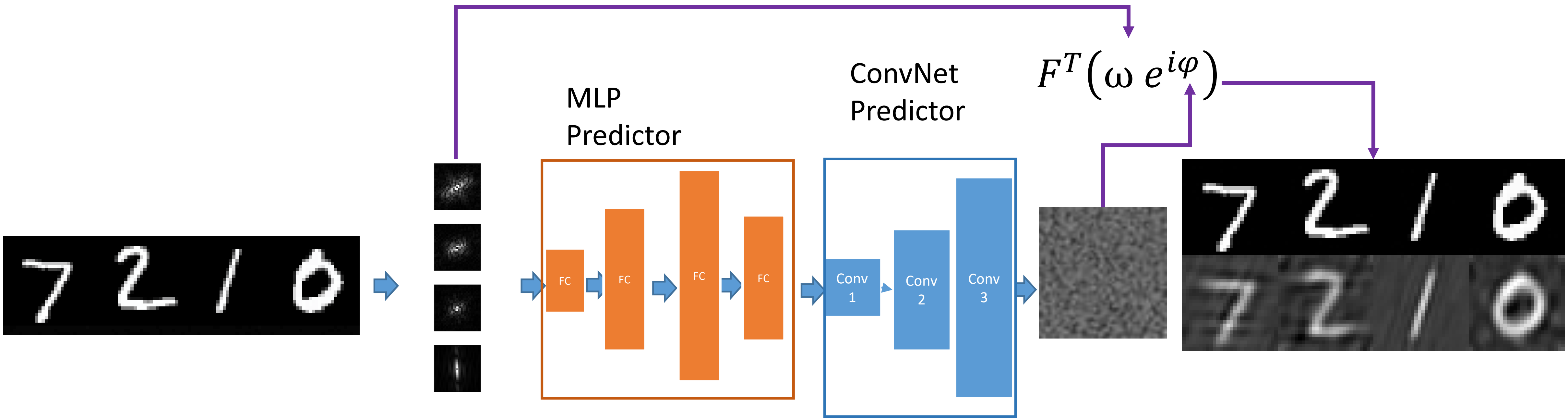
- Initially it made sense to add
- Covered/learnt by MLP subnet

Ablation study: using only conv nets

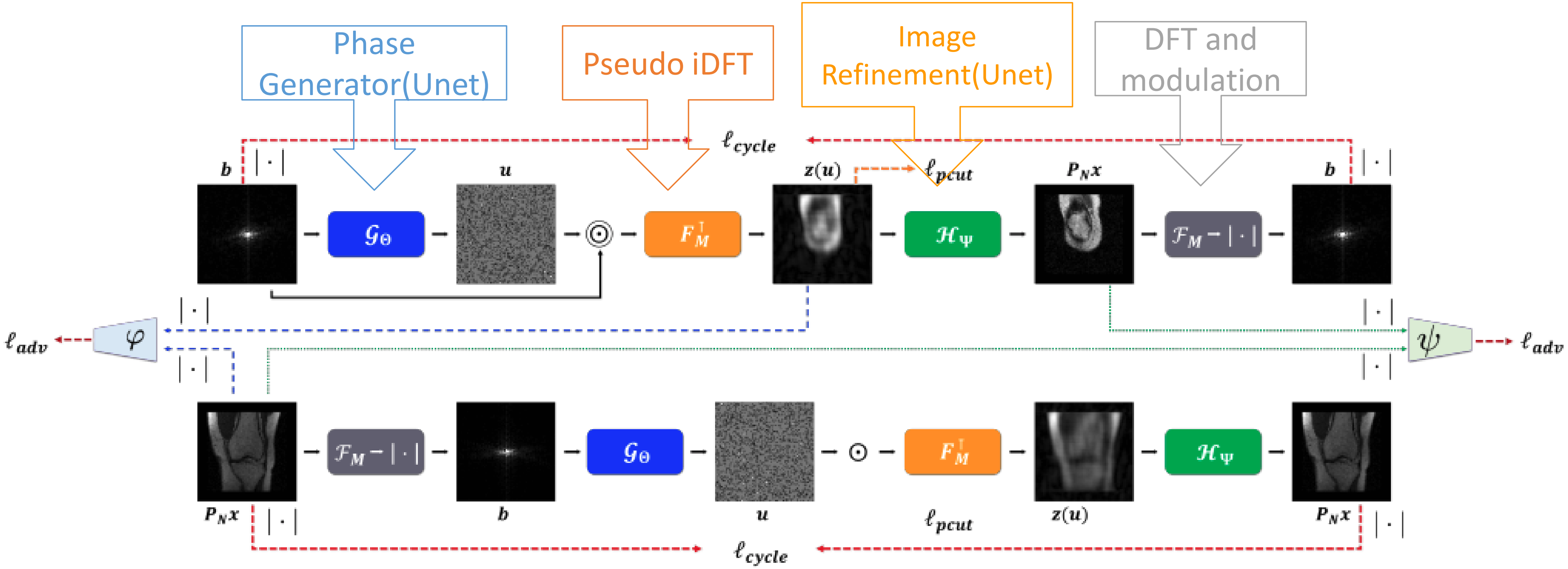
Convolutions do not make sense in the Fourier domain !



Ablation study: Actual phase recovery



Alternative architecture: DeepPhaseCut



PR-DAD: Evaluation metrics

180 rotation invariant metrics

- MSE – L2
- MAE – L1
- Similarity loss

$$SSIM = \frac{(2\mu_{\tilde{\boldsymbol{x}}}\mu_{\boldsymbol{x}} + c_1)(2\sigma_{\tilde{\boldsymbol{x}}\boldsymbol{x}} + c_2)}{(\mu_{\tilde{\boldsymbol{x}}}^2 + \mu_{\boldsymbol{x}}^2 + c_1)(\sigma_{\tilde{\boldsymbol{x}}}^2 + \sigma_{\boldsymbol{x}}^2 + c_2)}$$

- PSNR

$$PSNR = 20 \cdot \log_{10}\left(\frac{MAX_{\boldsymbol{x}}}{\sqrt{MSE(\tilde{\boldsymbol{x}}, \boldsymbol{x})}}\right),$$

TABLE II

QUANTITATIVE COMPARISON ON THE MNIST DATASET

Model	MSE	MAE	SSIM	PSNR
PRCGAN [16]	0.0168	0.0399	0.8449	-
CPR [15]	0.0123	**0.037**	0.8756	-
PR-DAD Haar Packet	0.0106	0.0381	**0.8815**	39.4861
PR-DAD auto encoder-decoder	**0.0100**	0.0398	0.8799	**40.0208**

TABLE III

QUANTITATIVE COMPARISON ON THE EMNIST DATASET

Model	MSE	MAE	SSIM	PSNR
PRCGAN [16]	0.0239	0.0601	0.8082	-
CPR [15]	0.0144	0.0501	0.8700	-
PR-DAD Haar Packet	0.0119	0.0475	0.8710	38.4744
PR-DAD auto encoder-decoder	**0.0108**	**0.0422**	**0.8879**	**39.2972**

TABLE IV

QUANTITATIVE COMPARISON ON THE KMNIST DATASET

Model	MSE	MAE	SSIM	PSNR
PRCGAN [16]	0.0651	0.1166	0.5711	-
CPR [15]	0.0433	0.1034	0.6624	-
PR-DAD Haar Packet	0.0383	0.1027	0.6365	28.3249
PR-DAD auto encoder-decoder	**0.0380**	**0.0957**	**0.6605**	**28.4031**

TABLE V

QUANTITATIVE COMPARISON ON THE FASHION-MNIST DATASET

Model	MSE	MAE	SSIM	PSNR
PRCGAN [16]	0.0151	0.0572	0.7749	-
CPR [15]	0.0113	0.0497	0.8092	-
PR-DAD Haar Packet	**0.0078**	0.0471	0.8186	**42.1862**
PR-DAD auto encoder-decoder	0.0081	**0.0442**	**0.8242**	41.811

TABLE VI
QUANTITATIVE COMPARISON ON THE CROPPED CELEBA 64×64 DATASET

Model	**MSE**	**MAE**	**SSIM**	**PSNR**
PRCGAN [16]	0.0138	0.0804	0.6779	n/a
HIO [6]	n/a	n/a	0.472	19.573
PhaseCut [17]	n/a	n/a	0.7600	25.3600
On-RED [18]	n/a	n/a	0.4940	19.7960
PrDeep [13]	n/a	n/a	0.7380	26.0579
DeepPhaseCut [4]	n/a	n/a	0.8540	27.1190
PR-DAD	**0.0025**	**0.0340**	**0.8815**	**51.9661**

Fig. 7. Top - cropped CelebA original images, bottom - cropped celebA recovered images