Fusion of generative & discriminative models

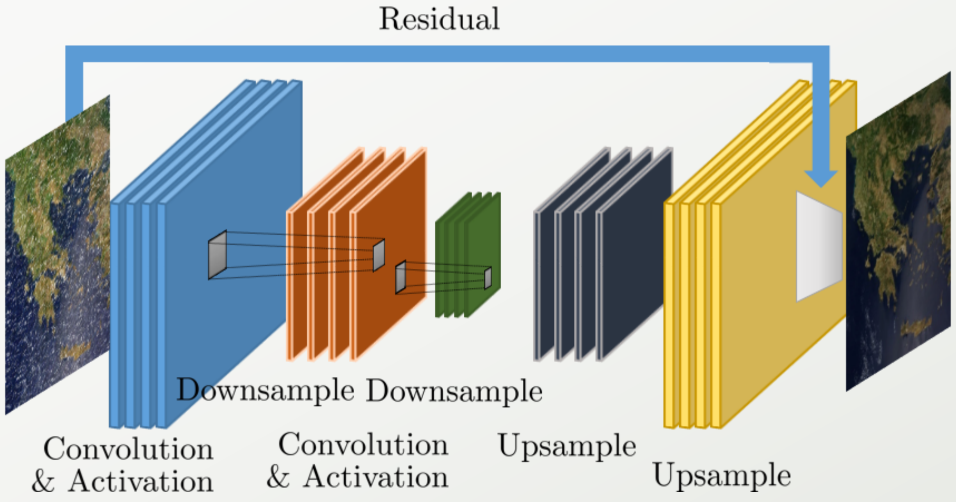
GREGORY TSAGKATAKIS

INSTITUTE OF COMPUTER SCIENCE - FORTH

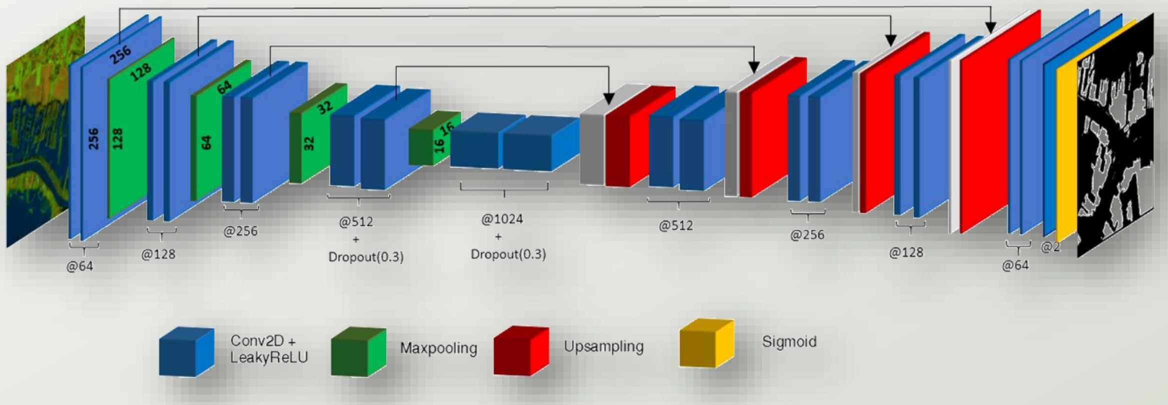
COMPUTER SCIENCE DEPARTMENT, UNIVERSITY OF CRETE

Challenges

Generative models

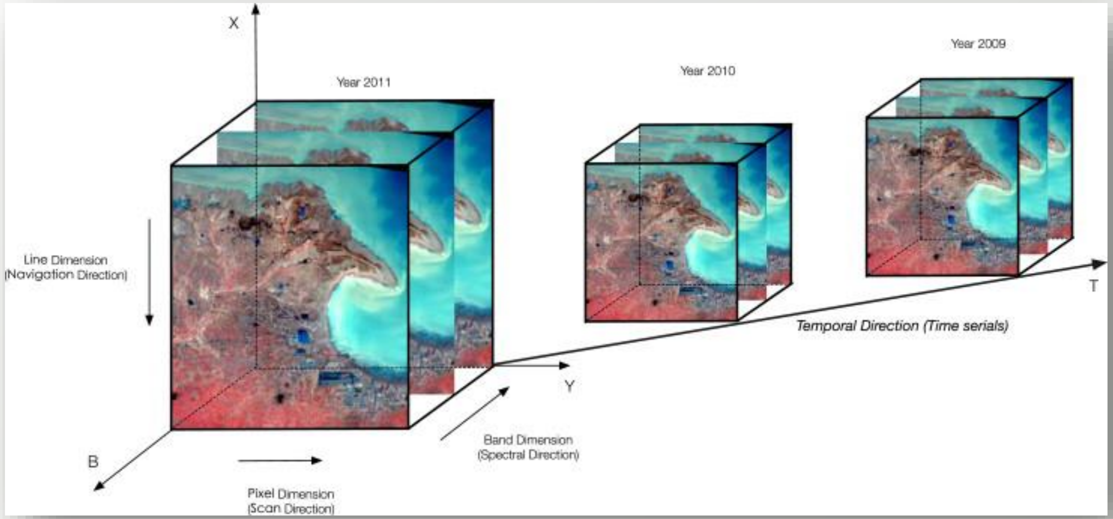


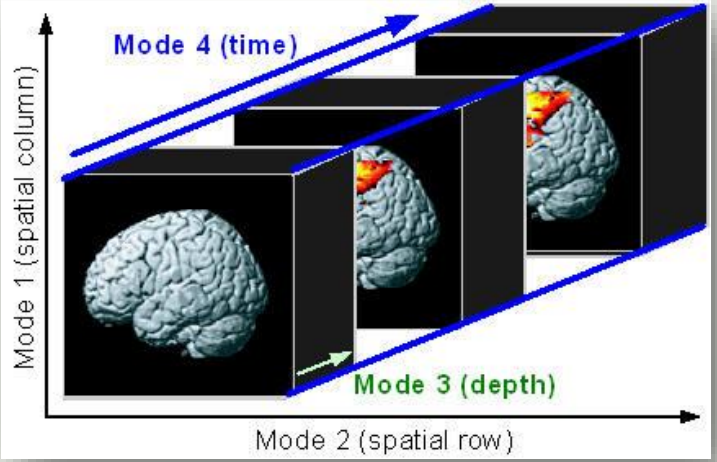
Discriminative models

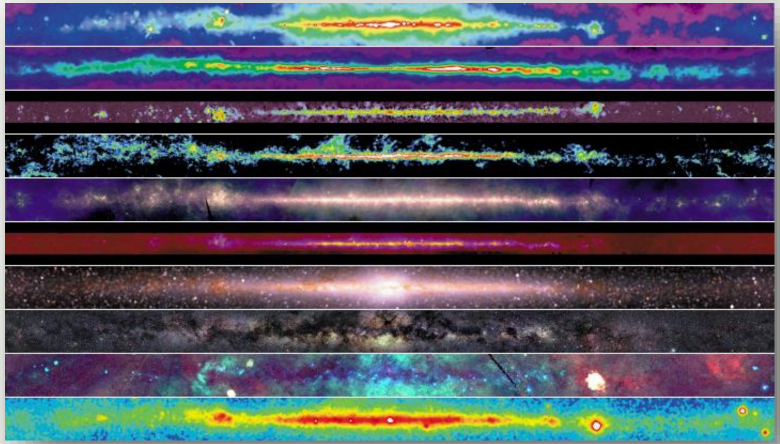


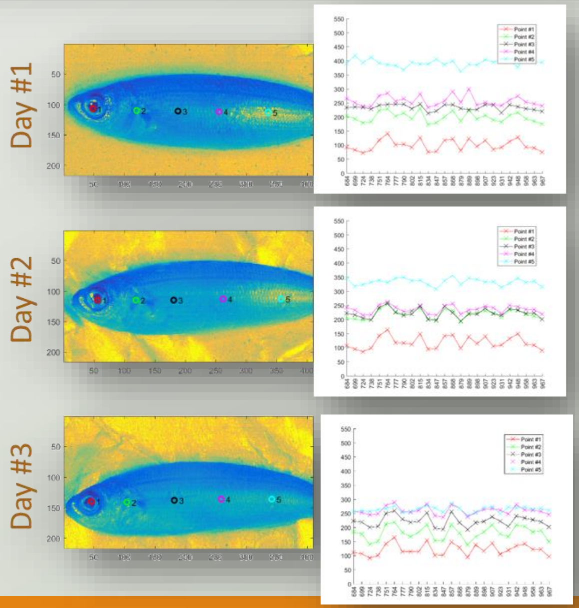
- High dimensionality of data
- Lack of proper “ground-truth”
- Class imbalance
- Hallucinations

High dimensional signals

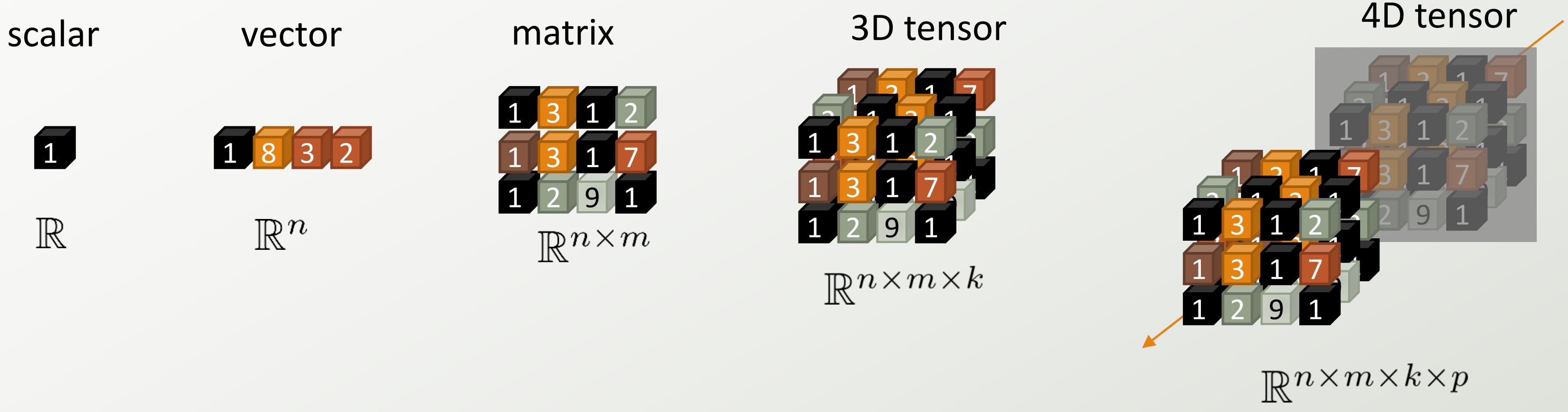


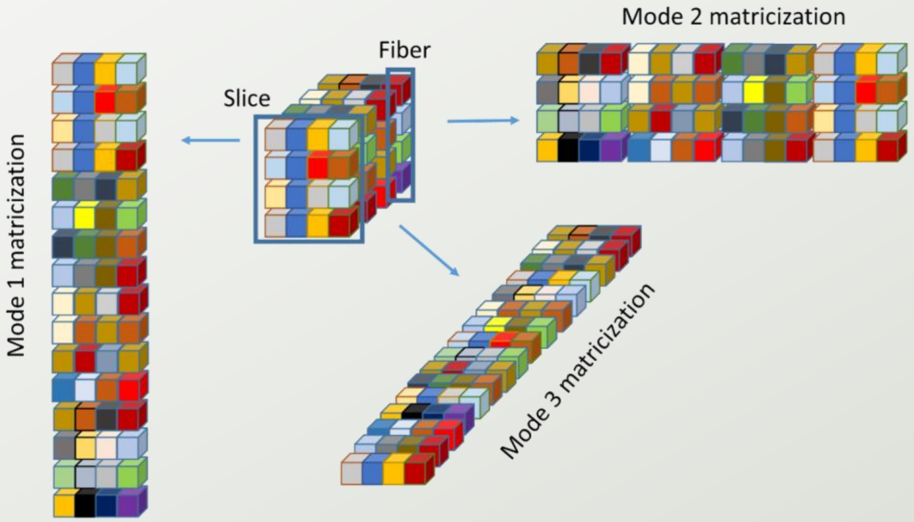






Tensor primer





Partial (Mode-1) Convolution

$\boldsymbol{X} \in \mathbb{R}^{I_1\times I_2}$

$\boldsymbol{Y} \in \mathbb{R}^{J_1\times J_2}$

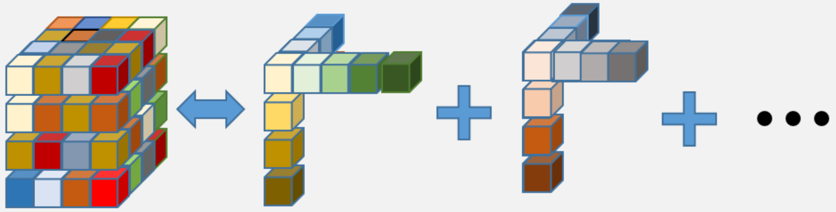
$$\boldsymbol{Z} = \boldsymbol{X} \boxdot_{(1)} \boldsymbol{Y}$$

$$\mathbf{Z}(:,k_2) = \mathbf{X}(:,i_2) \star \mathbf{Y}(:,j_2)$$

Matrix & Tensor Decomposition

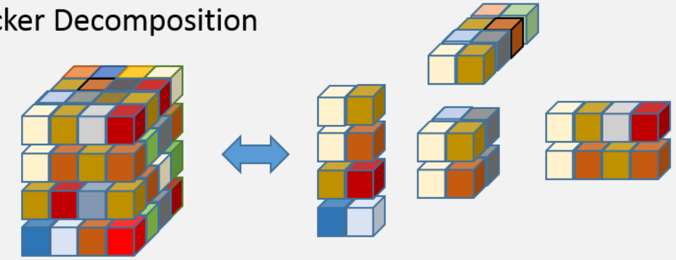
Assume a third-order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$

CANDECOMP/PARAFAC Decomposition



$$\mathcal{X} = \sum_{r=1}^{J} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

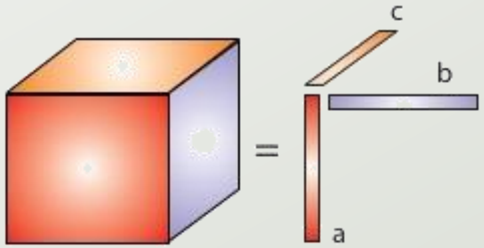
Tucker Decomposition



$$\mathcal{X} = \mathcal{G} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \times_3 \ldots \times_N \boldsymbol{A}^{(N)}$$

Tensor rank

The outer product of N vectors yields a *rank-1* N-way tensor.



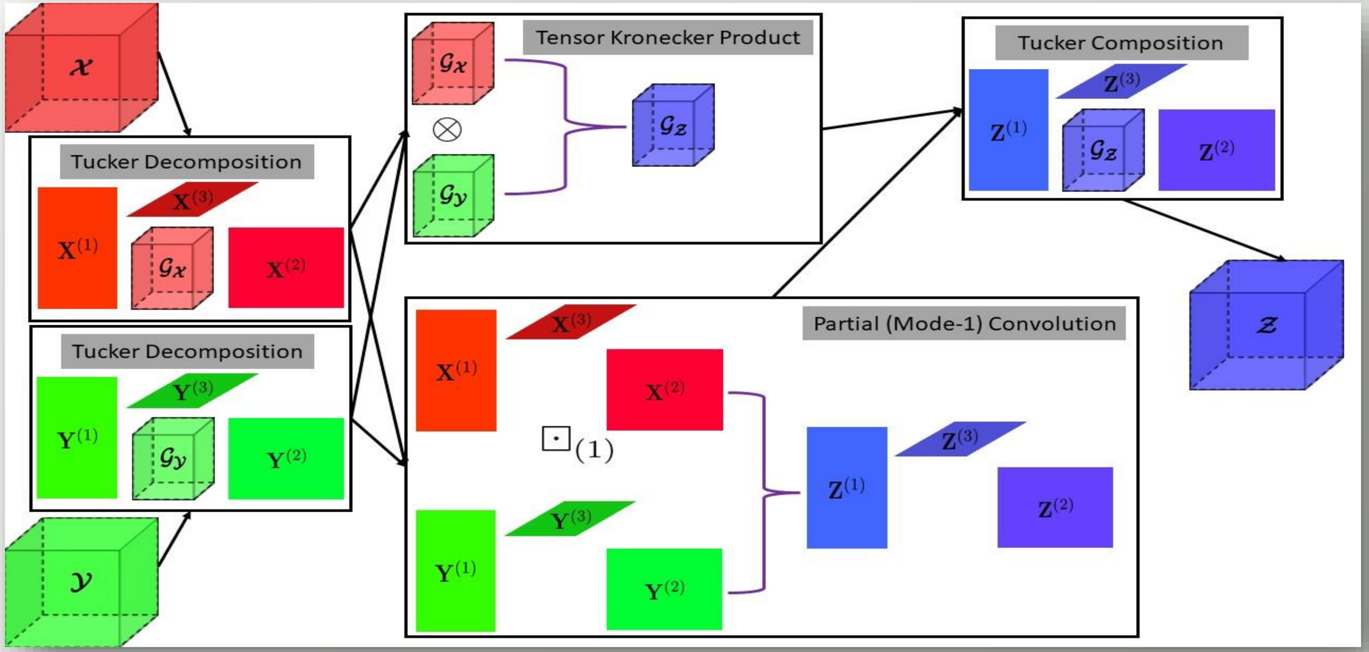
Kronecker Product

$\boldsymbol{X} \in \mathbb{R}^{I \times J}$

$\boldsymbol{Y} \in \mathbb{R}^{K \times L}$

$$\mathbf{Z} = \mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} x_{11}\mathbf{Y} & x_{12}\mathbf{Y} & \cdots & x_{1J}\mathbf{Y} \\ x_{21}\mathbf{Y} & x_{22}\mathbf{Y} & \cdots & x_{2J}\mathbf{Y} \\ \vdots & \vdots & \ddots & \vdots \\ x_{I1}\mathbf{Y} & x_{I2}\mathbf{Y} & \cdots & x_{IJ}\mathbf{Y} \end{bmatrix} \in \mathbb{R}^{IK \times JL}$$

Tensor Convolution



$$\mathcal{X} = \mathcal{G}_{\mathcal{X}} \times_1 \boldsymbol{X}^{(1)} \times_2 \boldsymbol{X}^{(2)} \times_3 \dots \times_N \boldsymbol{X}^{(N)}$$

$$\mathcal{Y} = \mathcal{G}_{\mathcal{Y}} \times_1 \boldsymbol{Y}^{(1)} \times_2 \boldsymbol{Y}^{(2)} \times_3 \dots \times_N \boldsymbol{Y}^{(N)}$$

$$\mathcal{Z} = conv(\mathcal{X}, \mathcal{Y}) \quad \begin{cases} \mathcal{G}_{\mathcal{Z}} = \mathcal{G}_{\mathcal{X}} \otimes \mathcal{G}_{\mathcal{Y}} \\ \begin{cases} \mathbf{Z}^{(n)} = \mathbf{X}^{(n)} \boxdot_{(1)} \mathbf{Y}^{(n)} \\ \mathbf{Z}^{(n)}(:, s_n) = \mathbf{X}^{(n)}(:, r_n) \star \mathbf{Y}^{(n)}(:, q_n) \\ s_n = \overline{r_n q_n} = 1, 2, \dots, R_n Q_n \end{cases} \end{cases}$$

Giannopoulos, M., Tsagkatakis, G. and Tsakalides, P., 4D U-Nets for multi-temporal remote sensing data classification. Remote Sensing, 2022

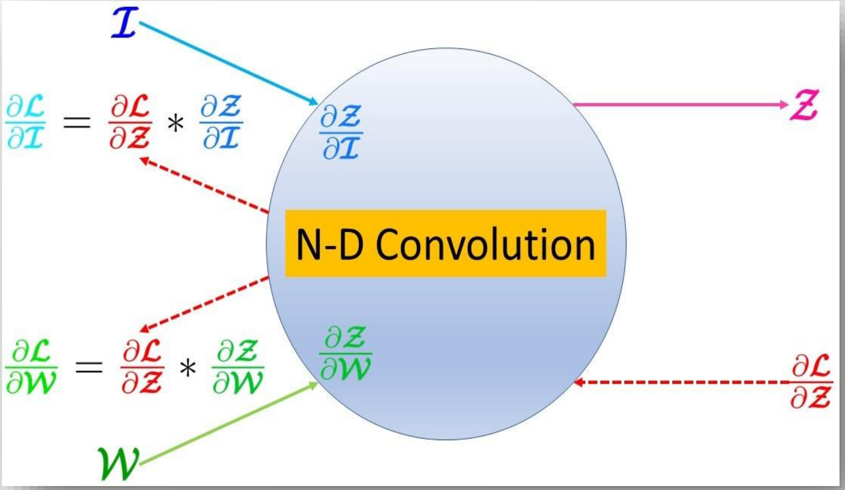
Backpropagation of 4D Convolution

$$\frac{\partial \mathcal{L}}{\partial \mathcal{I}} = conv\left(\frac{\partial \mathcal{L}}{\partial \mathcal{Z}}, \mathcal{W}(l)\right)$$

$$\frac{\partial \mathcal{L}}{\partial \mathcal{W}} = conv\left(\frac{\partial \mathcal{L}}{\partial \mathcal{Z}}, rot_N(\mathcal{I}(l-1))\right)$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{i_1}\sum_{i_2}\cdots\sum_{i_N}\left(\frac{\partial \mathcal{L}}{\partial \mathcal{Z}}\right)$$

Both forward and backward passes are convolutions!



Chain-Rule Computation

$\frac{\partial \mathcal{L}}{\partial \mathcal{Z}}$ → **Loss-gradient from the next layer**

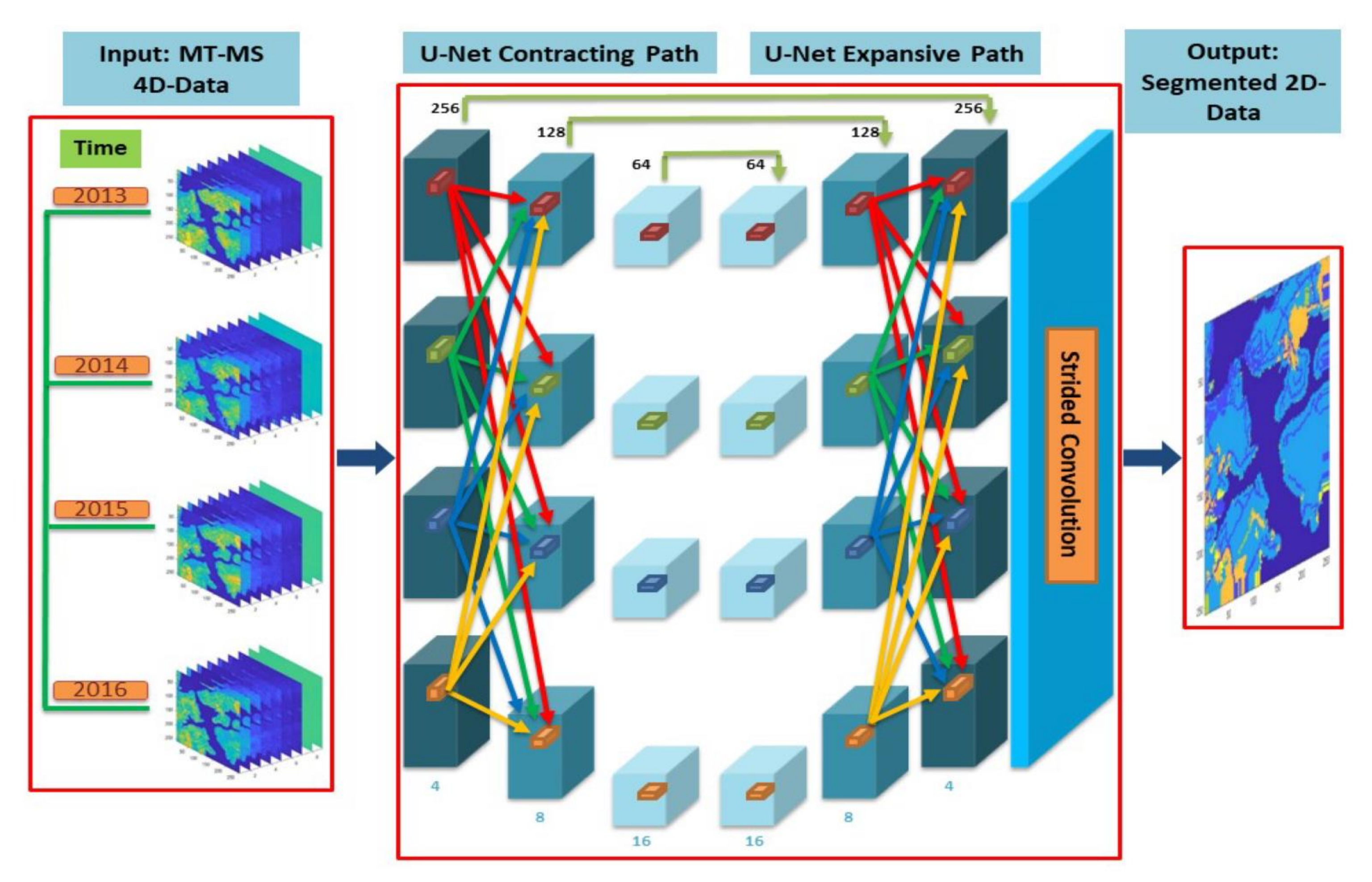
$\frac{\partial \mathcal{Z}}{\partial \mathcal{I}}, \frac{\partial \mathcal{Z}}{\partial \mathcal{W}}$ → **Local-gradients**

$\frac{\partial \mathcal{L}}{\partial \mathcal{I}}$ **Gradient w.r.t. Input** — Loss-gradient for the previous layer

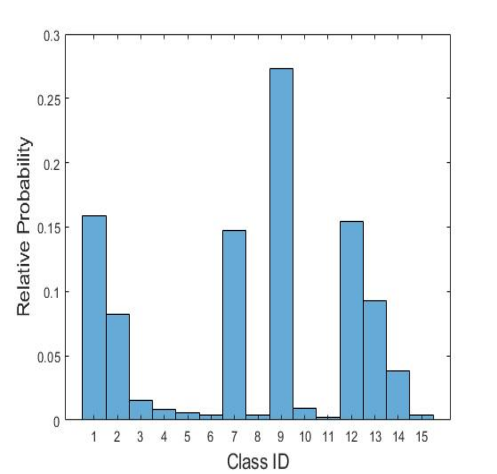
$\frac{\partial \mathcal{L}}{\partial \mathcal{W}}$ **Gradient w.r.t. Weights** — $\mathcal{W}_{New} = \mathcal{W}_{Old} - \mu \frac{\partial \mathcal{L}}{\partial \mathcal{W}}$

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \mathcal{I}} = \frac{\partial \mathcal{L}}{\partial \mathcal{Z}} * \frac{\partial \mathcal{Z}}{\partial \mathcal{I}} \\ \frac{\partial \mathcal{L}}{\partial \mathcal{W}} = \frac{\partial \mathcal{L}}{\partial \mathcal{Z}} * \frac{\partial \mathcal{Z}}{\partial \mathcal{W}} \end{cases}$$

Land cover classification

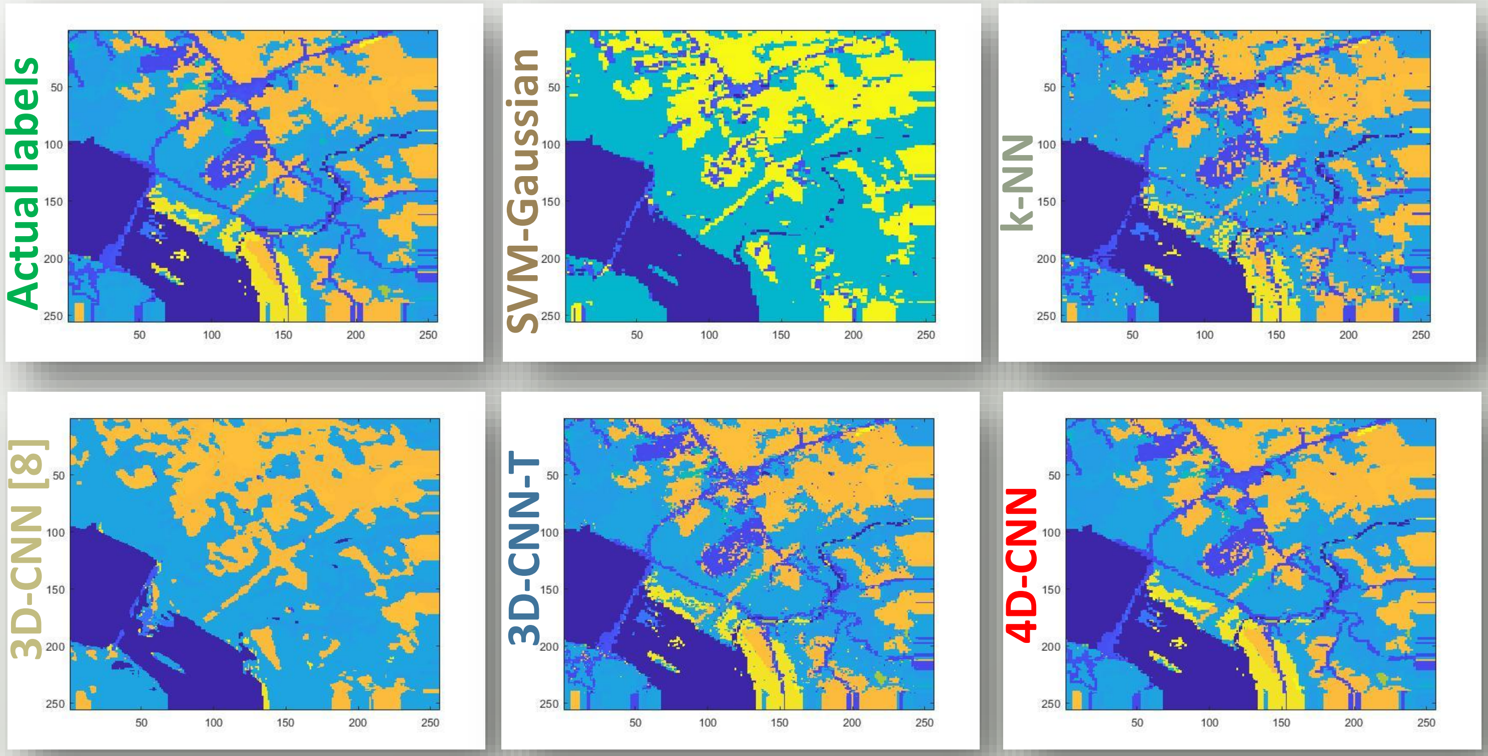


Class ID	Class Value	Class Name
1	11	Open Water
2	21	Developed, Open Space
3	22	Developed, Low Intensity
4	23	Developed, Medium Intensity
5	24	Developed, High Intensity
6	31	Barren Land (Rock/Sand/Clay)
7	41	Deciduous Forest
8	42	Evergreen Forest
9	43	Mixed Forest
10	52	Shrub/Scrub
11	71	Grassland/Herbaceous
12	81	Pasture/Hay
13	82	Cultivated Crops
14	90	Woody Wetlands
15	95	Emergent Herbaceous Wetlands



Results on 4D classification

Model	Accuracy	F1-Score	Time
k-NN	0.74	0.57	1
SVM-Gaussian	0.59	0.49	2
2D-CNN	0.60	0.36	4
3D-CNN [8]	0.70	0.50	6
3D-CNN-T	0.80	0.65	14
3D-CNN-S	0.79	0.65	27
4D-CNN	**0.89**	**0.77**	**107**



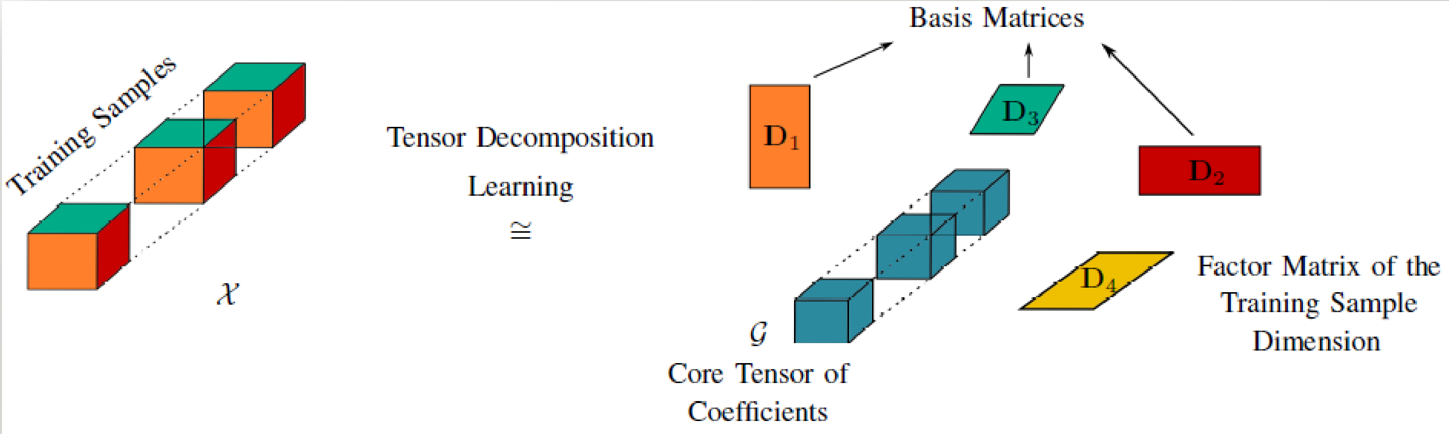
Tensor Decomposition Learning

Learn a basis for each mode, $\boldsymbol{D}_n \in \mathbb{R}^{I_n \times R_n}$ for $n = 1, .., N$, from S training samples $\mathcal{X} = (\mathcal{X}^1, ..., \mathcal{X}^S) \in \mathbb{R}^{I_1 \times \cdots \times I_N \times S}$ such that

$$\min_{\mathcal{G}, \boldsymbol{D}_1, .., \boldsymbol{D}_{N+1}} \frac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_N \boldsymbol{D}_N \times_{N+1} \boldsymbol{D}_{N+1}\|_F^2 + \lambda \|\boldsymbol{A}\|_*$$

$$\text{subject to } \boldsymbol{A} = \boldsymbol{D}_{N+1} \text{ and } \boldsymbol{D}_n^T \cdot \boldsymbol{D}_n = \boldsymbol{I}_{R_n}, \; n = 1, .., N$$

where $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N \times S}$, $\boldsymbol{D}_{N+1} \in \mathbb{R}^{S \times S}$ and $\boldsymbol{A} \in \mathbb{R}^{S \times S}$, by applying ADMM.



Aidini, A., Tsagkatakis, G., and Tsakalides, P., “Tensor decomposition learning for compression of multidimensional signals.” *IEEE Journal of Selected Topics in Signal Processing*, *15*(3), 2021

Algorithm (1/2)

We apply ADMM by minimizing the Lagrangian function

$$\mathcal{L}(\mathcal{G}, \boldsymbol{D}_1, .., \boldsymbol{D}_N, \boldsymbol{D}_{N+1}, \boldsymbol{A}, \boldsymbol{Y}) = \frac{1}{2}\|\mathcal{X} - \mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_{N+1} \boldsymbol{D}_{N+1}\|_F^2 + \lambda\|\boldsymbol{A}\|_* + \boldsymbol{Y} \cdot (\boldsymbol{A} - \boldsymbol{D}_{N+1}) + \frac{p}{2}\|\boldsymbol{A} - \boldsymbol{D}_{N+1}\|_F^2,$$

where $\boldsymbol{Y} \in \mathbb{R}^{S \times S}$ is the Lagrange multiplier matrix, while $p > 0$ denote the step size parameter. We optimize each variable alternatively while fixing the others

1. Auxilary variable $\boldsymbol{A}$: $\nabla_{\boldsymbol{A}}\mathcal{L} = 0 \Rightarrow$

$$\begin{aligned} \hat{\boldsymbol{A}} &\leftarrow \boldsymbol{D}_{N+1} - \frac{\boldsymbol{Y}}{p} \\ (\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}) &\leftarrow \mathrm{svd}(\hat{\boldsymbol{A}}) \\ \boldsymbol{s} &\leftarrow B_\lambda(\mathrm{diag}(\boldsymbol{S})) \\ \boldsymbol{A} &\leftarrow \boldsymbol{U} \cdot \mathrm{diag}(\boldsymbol{s}) \cdot \boldsymbol{V}^T \end{aligned}$$

Algorithm (2/2)

2. Basis of each mode $\boldsymbol{D}_n,\ n = 1, .., N$: $\quad \nabla_{\boldsymbol{D}_n} \mathcal{L} = 0 \Rightarrow$

$$\begin{aligned} \hat{\boldsymbol{D}}_n &\leftarrow (\boldsymbol{X}_{(n)} \boldsymbol{C}^T_{n_{(n)}}) \cdot (\boldsymbol{C}_{n_{(n)}} \boldsymbol{C}^T_{n_{(n)}})^{-1}, \\ (\boldsymbol{Q}, \boldsymbol{R}) &\leftarrow QR(\hat{\boldsymbol{D}}_n) \\ \boldsymbol{D}_n &\leftarrow \boldsymbol{Q}(:, 1 : R_n) \end{aligned}$$

where $\mathcal{C}_n = \mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_{n-1} \boldsymbol{D}_{n-1} \times_{n+1} \boldsymbol{D}_{n+1} \times_{n+2} \cdots \times_{N+1} \boldsymbol{D}_{N+1}$.

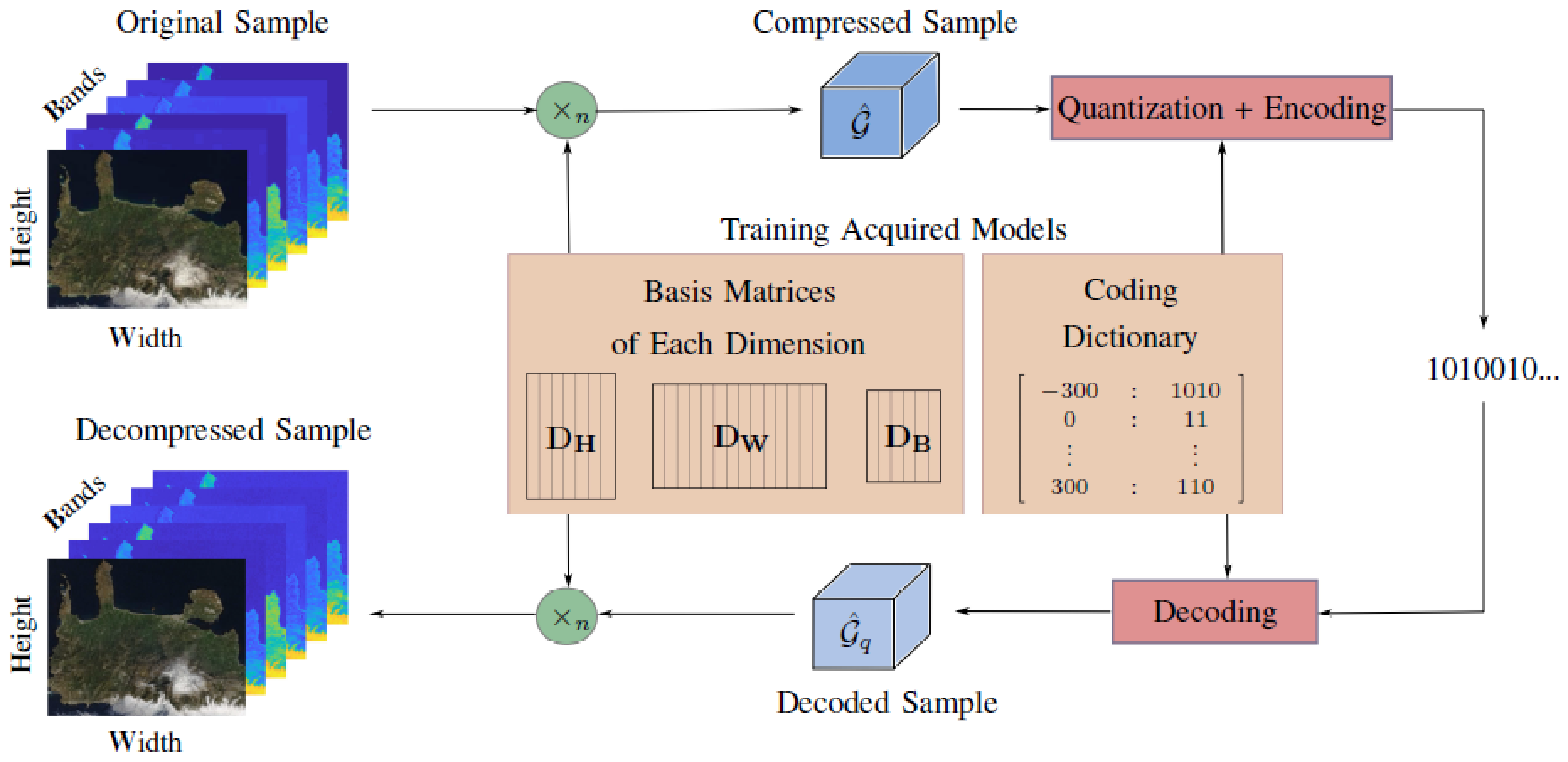
3. Factor matrix along the sample-variable $\boldsymbol{D}_{N+1}$: $\nabla_{\boldsymbol{D}_{N+1}} \mathcal{L} = 0 \Rightarrow$

$$\boldsymbol{D}_{N+1} \leftarrow (\boldsymbol{X}_{(N+1)} \boldsymbol{C}^T_{N+1_{(N+1)}} + \boldsymbol{Y} + p\boldsymbol{A}) \cdot (\boldsymbol{C}_{N+1_{(N+1)}} \boldsymbol{C}^T_{N+1_{(N+1)}} + p\boldsymbol{I}_S)^{-1}$$

4. Core tensor of coefficients $\mathcal{G}$: $\quad \nabla_{\mathcal{G}} \mathcal{L} = 0 \Rightarrow \mathcal{G} \leftarrow \mathcal{X} \times_1 \boldsymbol{D}_1^T \times_2 \cdots \times_N \boldsymbol{D}_N^T \times_{N+1} \boldsymbol{D}_{N+1}^{-1}$

5. Lagrange multiplier matrix $\boldsymbol{Y}$: $\boldsymbol{Y}^{(k+1)} \leftarrow \boldsymbol{Y}^{(k)} + p \cdot (\boldsymbol{A} - \boldsymbol{D}_{N+1})$, where $p = 0.01$ in our setup

Compression and Decompression



$$\textbf{Compression:} \quad \hat{\mathcal{G}} = \hat{\mathcal{X}} \times_1 \boldsymbol{D}_1^T \times_2 \cdots \times_N \boldsymbol{D}_N^T$$

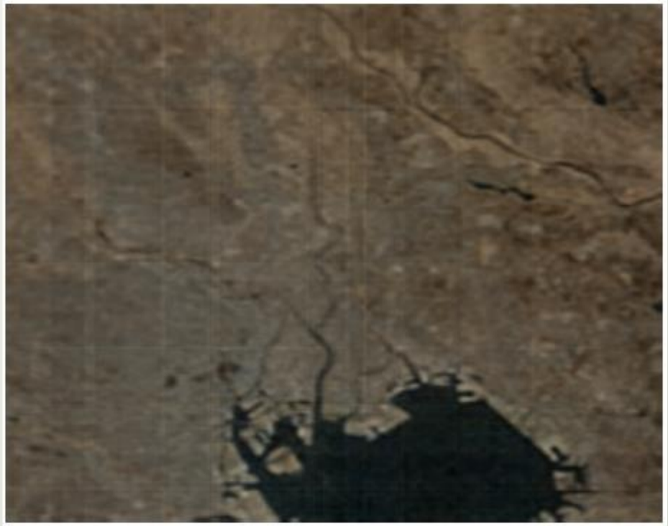
$$\textbf{Decompression:} \quad \hat{\mathcal{X}} \approx \hat{\mathcal{G}}_q \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_N \boldsymbol{D}_N, \; \hat{\mathcal{G}}_q = \mathcal{Q}(\hat{\mathcal{G}})$$

Training:
200 multispectral images over Chania, Barcelona, New York, Sydney

Testing:
100 multispectral images over Buenos Aires and Tokyo


(a) Original image

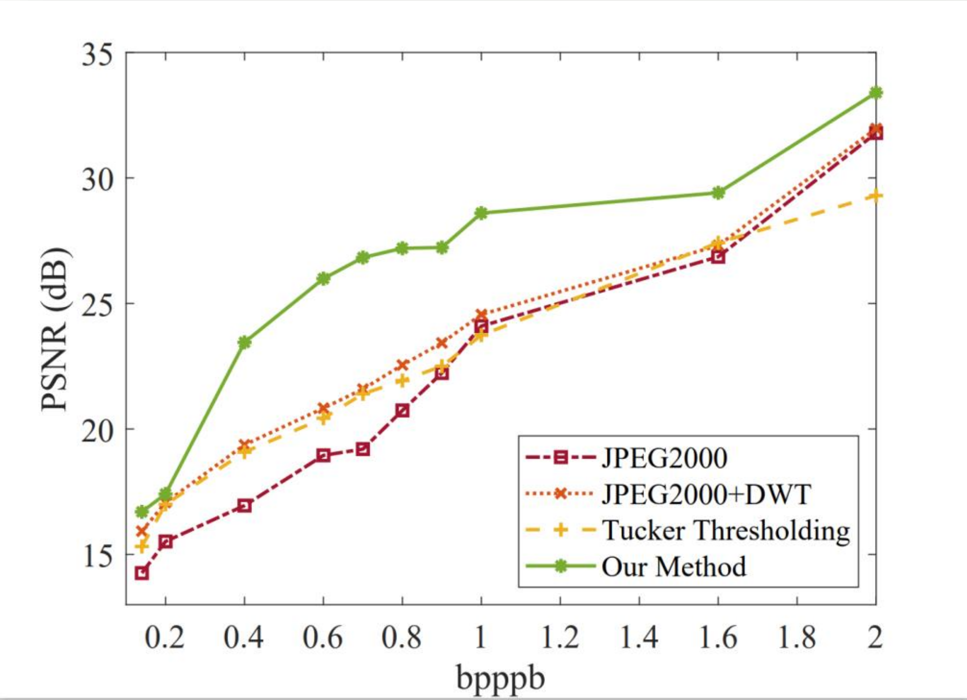

(b) JPEG2000+DWT
PNSR: 23.85dB

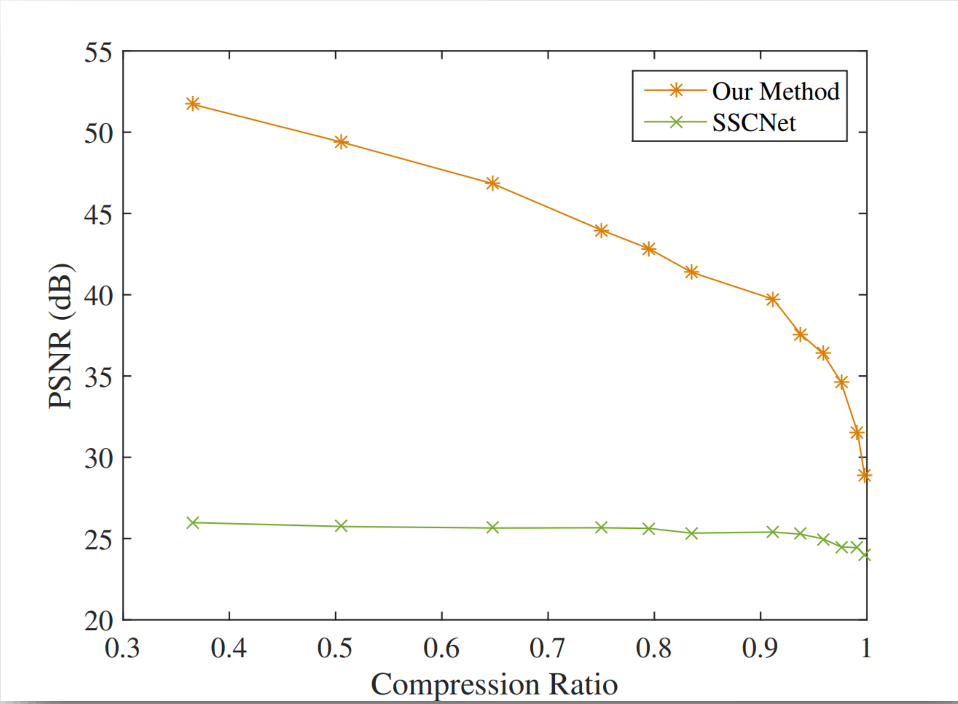

(c) Tucker Thresholding
PNSR: 24.39dB


(d) Proposed approach
PNSR: 27.02dB

Comparison

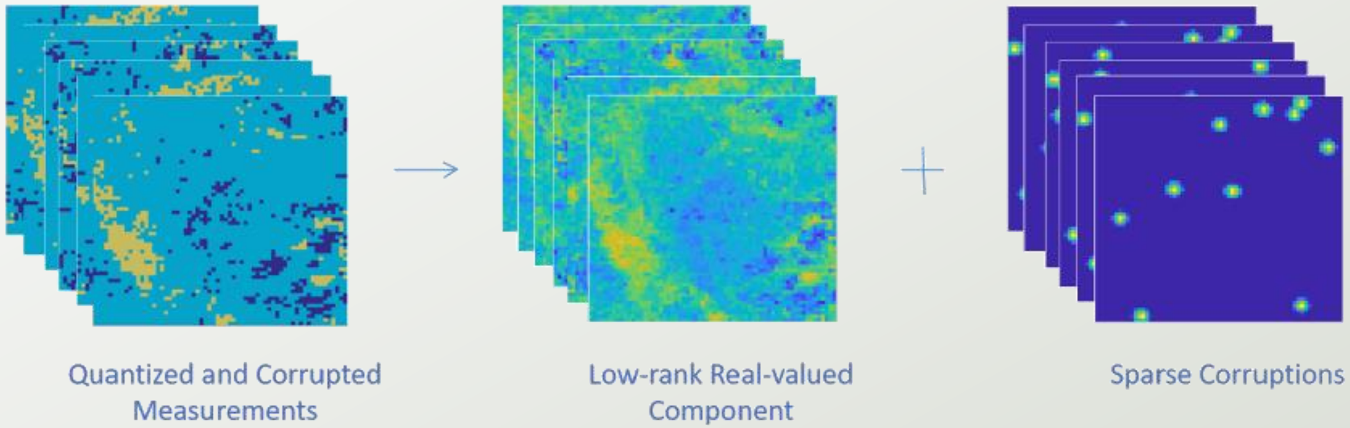
Data: Satellite multispectral image patches (64 × 64 × 5) Training: 2400 patches, Testing: 1200 patches (from different regions)





Tensor Robust Principal Component Analysis

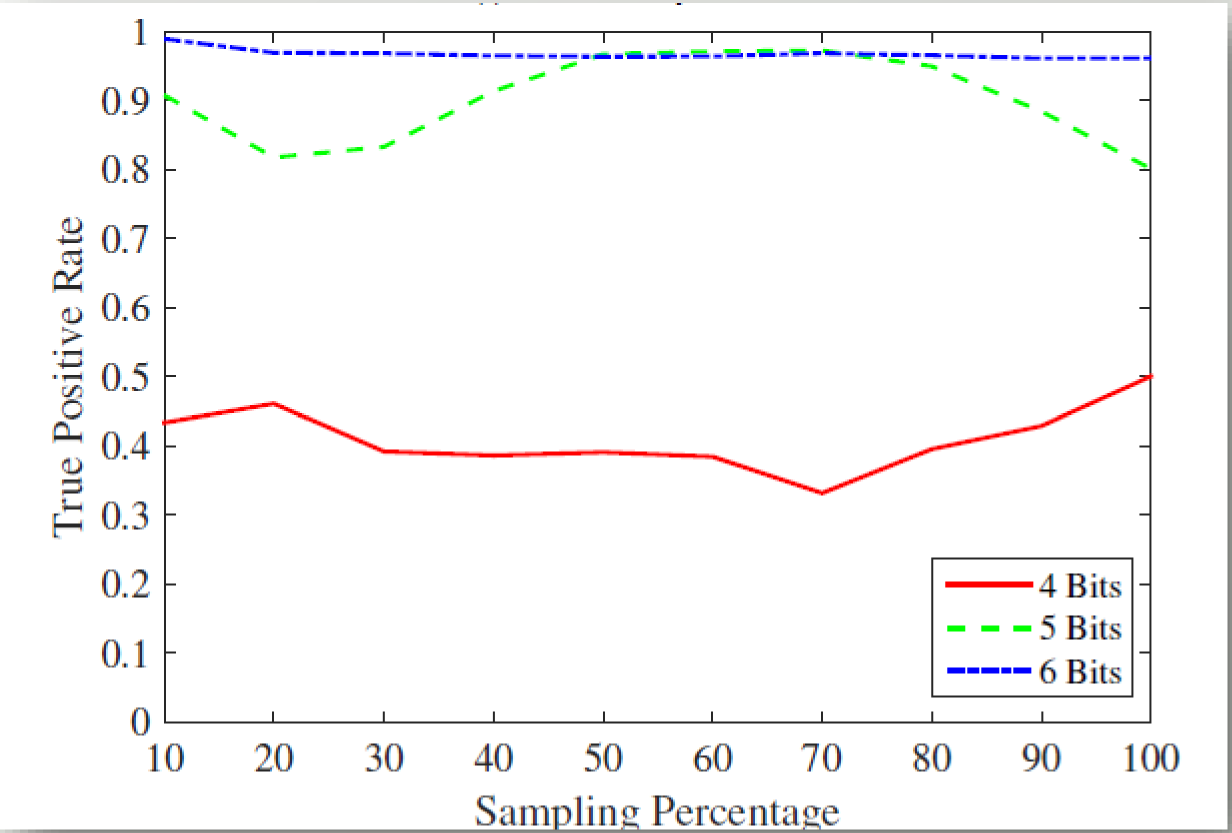
Problem: Recovery of a tensor from partial quantized and sparsely corrupted measurements.



A. Aidini, G. Tsagkatakis, and P. Tsakalides. "Quantized tensor robust principal component analysis." IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020

Anomaly detection

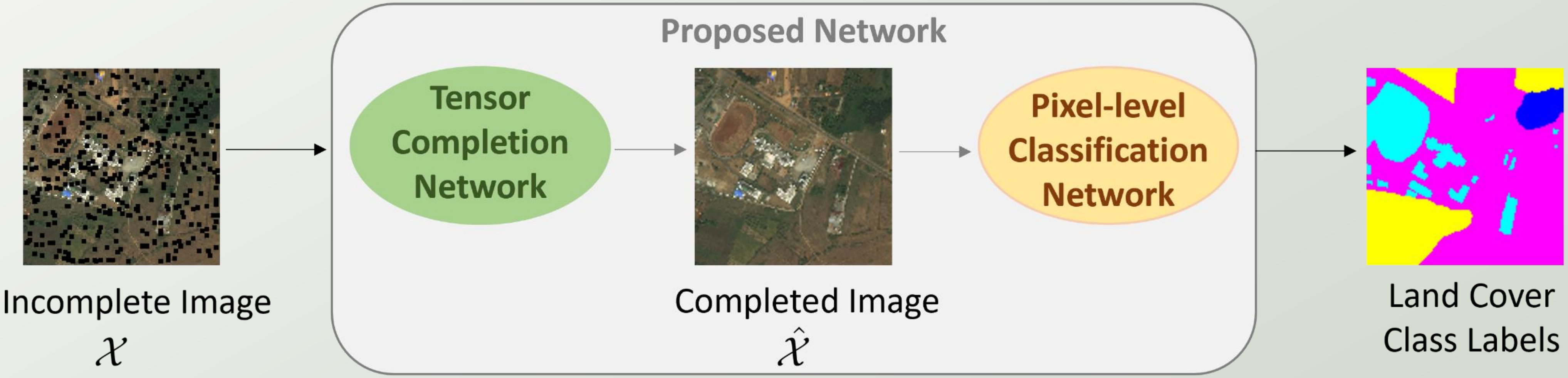
Data: Satellite (MODIS) image time-series of the land surface temperature over Brazil (8 bpppb - 22 days of July and August 2019)



Tensor-based Neural Networks

Deep learning formulation of tensor models:

- Exploit the benefits of both tensor analysis and deep learning techniques
- Create a tensor completion network
- Combine the tensor network with other popular networks
- Perform two tasks simultaneously



Baseline model

$$\min \tfrac{1}{2}\|\mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_N \boldsymbol{D}_N - \mathcal{Z}\|_F^2$$

$$\text{s.t. } \mathcal{P}_\Omega(\mathcal{Z}) = \mathcal{P}_\Omega(\mathcal{X}) \text{ and } \mathbf{D}_n^T \cdot \boldsymbol{D}_n = \boldsymbol{I}_{R_n}, n = 1, .., N$$

Lagrange function:

$$L(\mathcal{G}, \mathbf{D}_1, .., \mathbf{D}_N, \mathcal{Z}, \mathcal{Y}) = \frac{1}{2}\|\mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_N \boldsymbol{D}_N - \mathcal{Z}\|_F^2 - <\mathcal{Y}, \mathcal{P}_\Omega(\mathcal{Z}) - \mathcal{P}_\Omega(\mathcal{X})>$$

At each iteration l, we update:

$$\begin{aligned}
\boldsymbol{D}_n &= \mathrm{QR}(\boldsymbol{Z}_{(n)}^{l-1} \cdot \boldsymbol{C}_{n_{(n)}}^{-1}) \text{ where } \mathcal{C}_n = \mathcal{G} \times_{i=1, i \neq n}^{N} \boldsymbol{D}_i \\
\mathcal{G} &= \mathcal{Z}^{l-1} \times_1 \boldsymbol{D}_1^T \times_2 \cdots \times_N \boldsymbol{D}_N^T \\
\mathcal{Z}^l &= \mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_N \boldsymbol{D}_N + P_\Omega(\mathcal{X}) - P_\Omega(\mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_N \boldsymbol{D}_N)
\end{aligned}$$

"Low-rank tensor completion via tucker decompositions", Shi, Jiarong, et al., J. Comput. Inf. Syst, (2015)

Low Rank Tensor Completion – Net (LRTC-Net)

Trainable Parameters: Factor matrices $\boldsymbol{D}_n, n = 1, .., N$ (the same for all layers)

At each **layer** l we update:

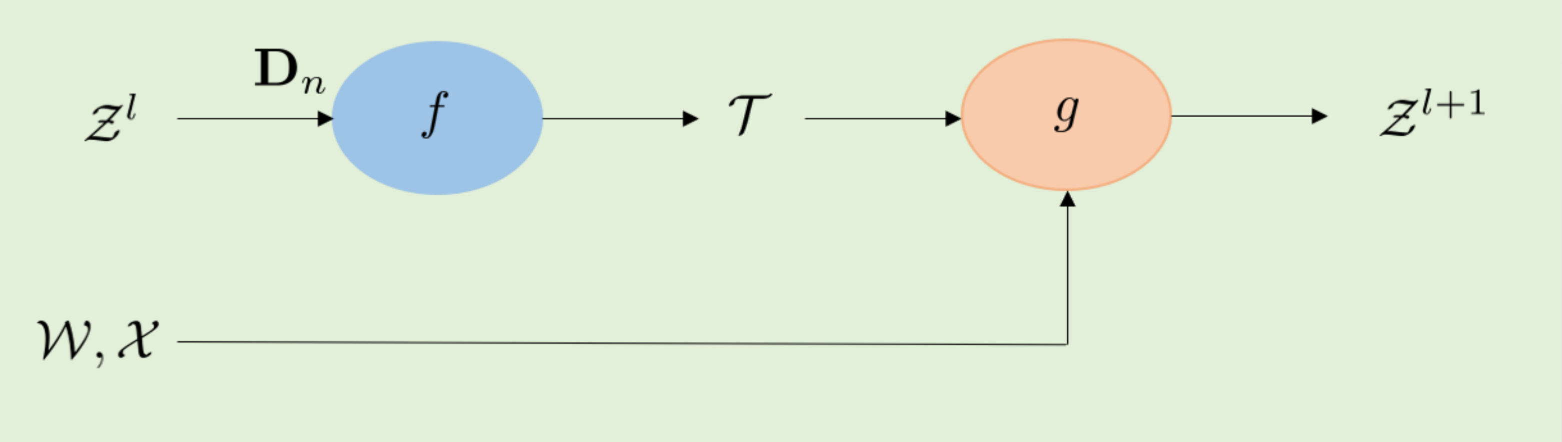
$$\begin{aligned}\mathcal{G} &= \mathcal{Z}^{l-1} \times_1 \boldsymbol{D}_1^T \times_2 \cdots \times_N \boldsymbol{D}_N^T \\ \mathcal{Z}^l &= \mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_N \boldsymbol{D}_N + \mathcal{P}_\Omega(\mathcal{X}) - \mathcal{P}_\Omega(\mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_N \boldsymbol{D}_N)\end{aligned}$$

Loss function

$$Loss = \mathrm{MSE}(\mathcal{G} \times_1 \boldsymbol{D}_1 \times_2 \cdots \times_N \boldsymbol{D}_N, \mathcal{Z}^L)$$

where L is the number of layers.

LRTC-Net layer



$$f(\mathcal{Z}^l, \boldsymbol{D}_n) = \mathcal{Z}^l \times_1 (\boldsymbol{D}_1 \cdot \boldsymbol{D}_1^T) \times_2 \cdots \times_N (\boldsymbol{D}_N \cdot \boldsymbol{D}_N^T)$$

$$g(\mathcal{T}, \mathcal{W}, \mathcal{X}) = \mathcal{T} + \mathcal{W} * \mathcal{X} - \mathcal{W} * \mathcal{T}$$

$$\mathcal{W}_{w_1,..,w_N} = \begin{cases} 1, & (w_1, .., w_N) \in \Omega \\ 0, & (w_1, .., w_N) \notin \Omega \end{cases}$$

DynamicEarthNet dataset

Images from the Fusion Monitoring product from Planet Labs:

- Resolution 3m, 1024 x 1024 , cloud free, 730 days
- 4 spectral bands (RGB + near-infrared), 7 classes

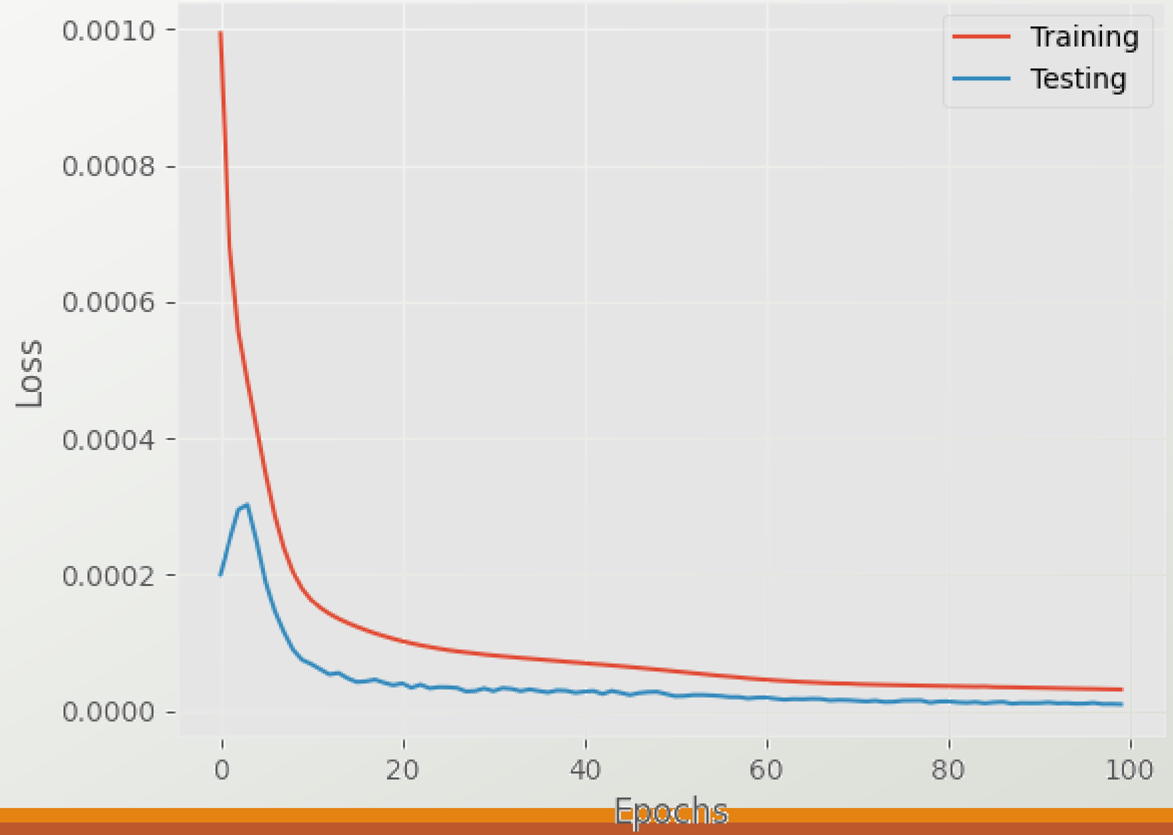
32 x 32 patches of 55 areas $\rightarrow$ 56320 image time series

Toker, Aysim, et al. "DynamicEarthNet: Daily Multi-Spectral Satellite Dataset for Semantic Change Segmentation.“, IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

LRTC-Net - Results

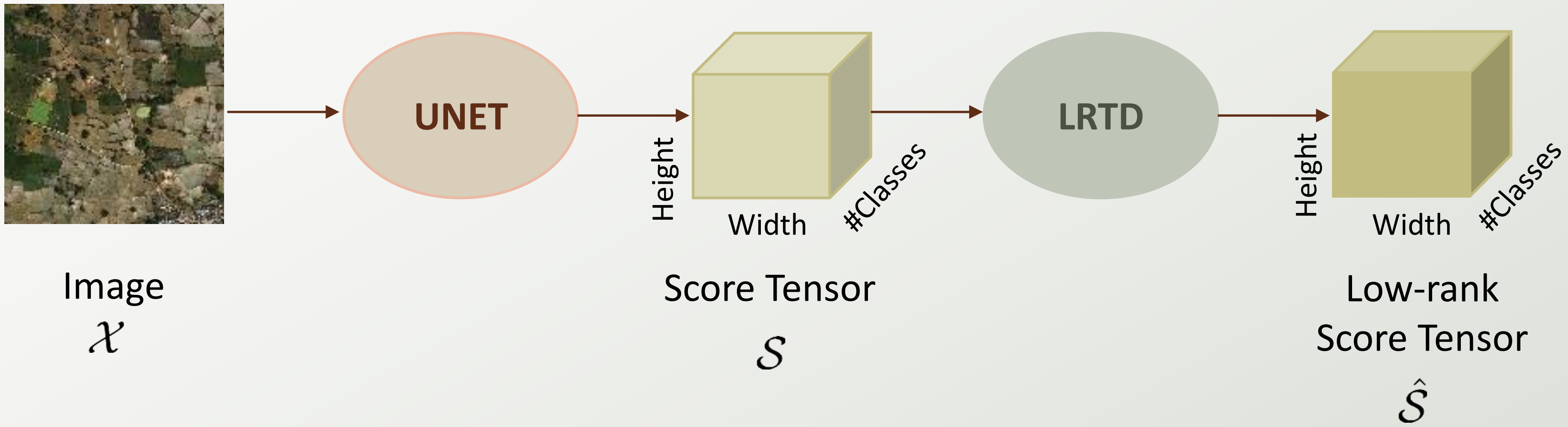
Data: 2000 image times series of size $32 \times 32 \times 3 \times 100$, 80% for training

LRTC-Net: 20 layers, rank 10% of the original dimensions, batch size 16, learning rate 0.0001, 100 epochs, 20% missing random values



Testing recovery error (NRMSE): 0.0134

Label recovery (semi-supervised learning)



Loss function:

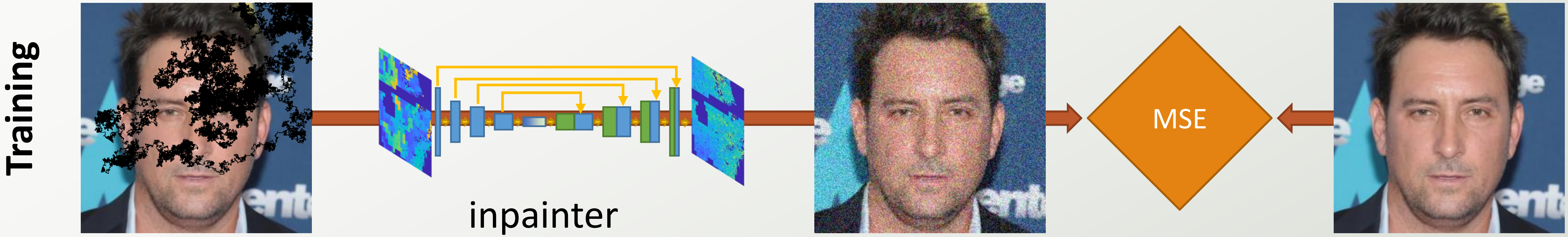
$$L = -\mathcal{Y}\log(f(g_\theta(\mathcal{X}))) + \frac{\lambda}{2}\|\mathcal{S} - \mathcal{G}\times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N\|_F^2 = CrossEntropy(\mathcal{S},\mathcal{Y}) + MSE(\mathcal{S},\hat{\mathcal{S}})$$

where f is the softmax function, g_θ indicates the UNET model with parameters θ, $\hat{\mathcal{S}} = \mathcal{G} \times_1 \mathbf{D}_1 \times_2 \cdots \times_N \mathbf{D}_N$, and $\lambda > 0$ controls the two terms of the loss function.

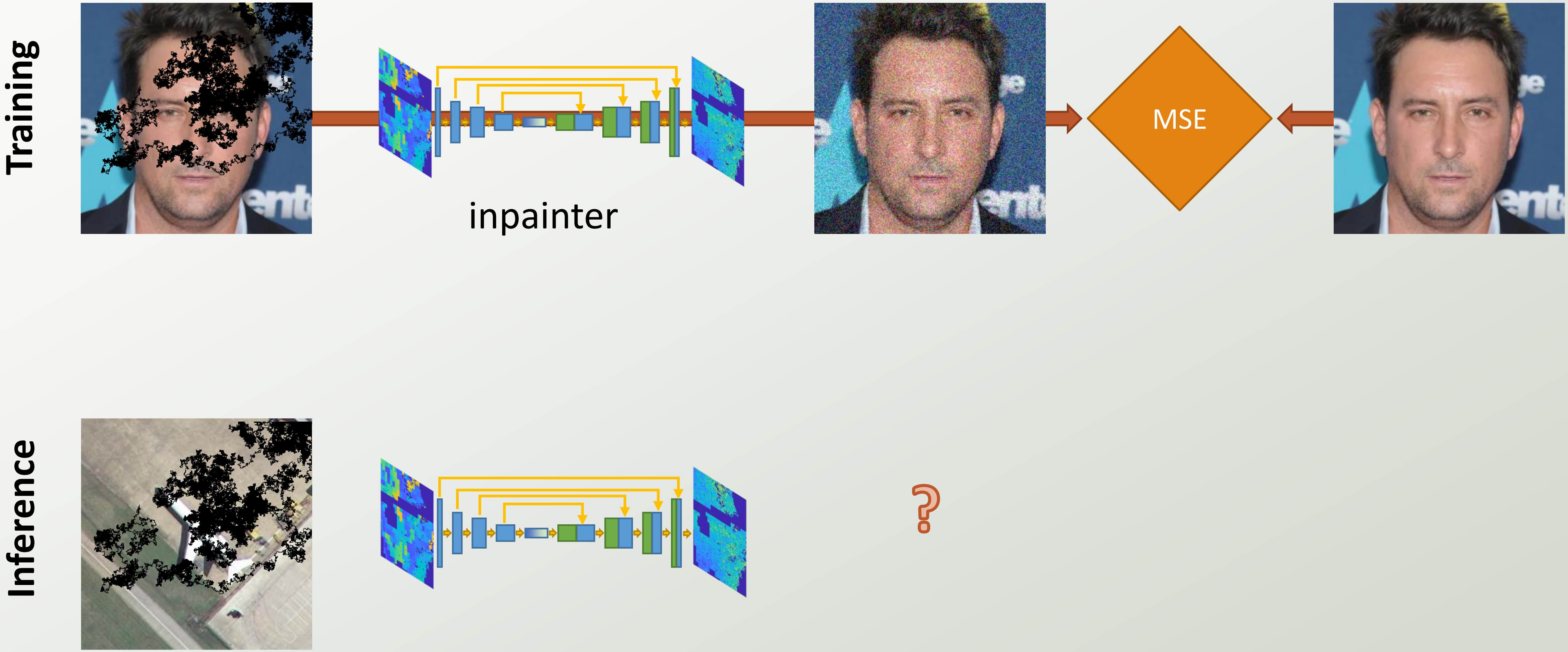
LRTD – Layer

1. No parameters, Tucker decomposition (iteratively) -> error on the update of the factor matrices
2. Parameters: Factor matrices, Estimation of Core tensor for each batch

Detection of Hallucinations

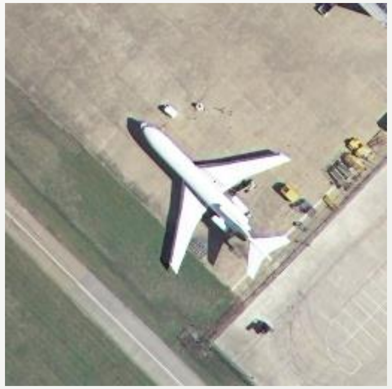


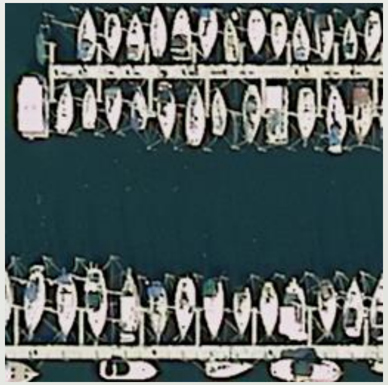
Detection of Hallucinations



Datasets

UC Merced land use consists of satellite images such as:



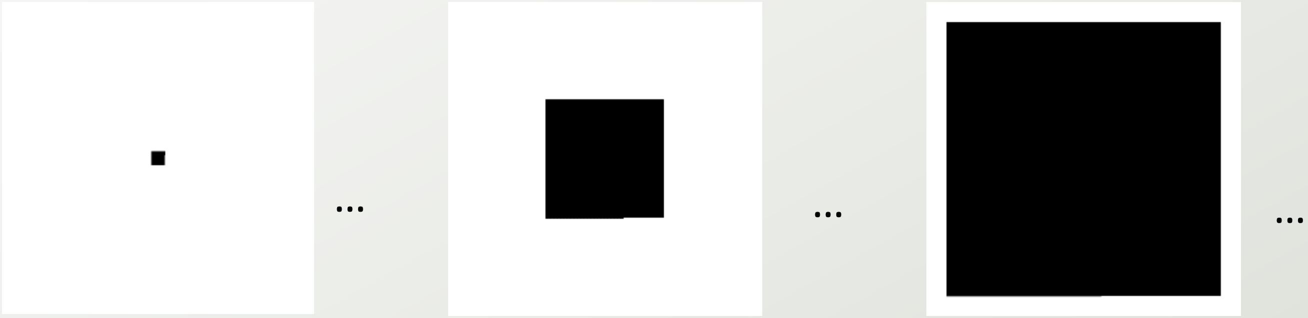



Celeba consists of celebrity faces such as:




Mask types

Rectangular masks



Random masks

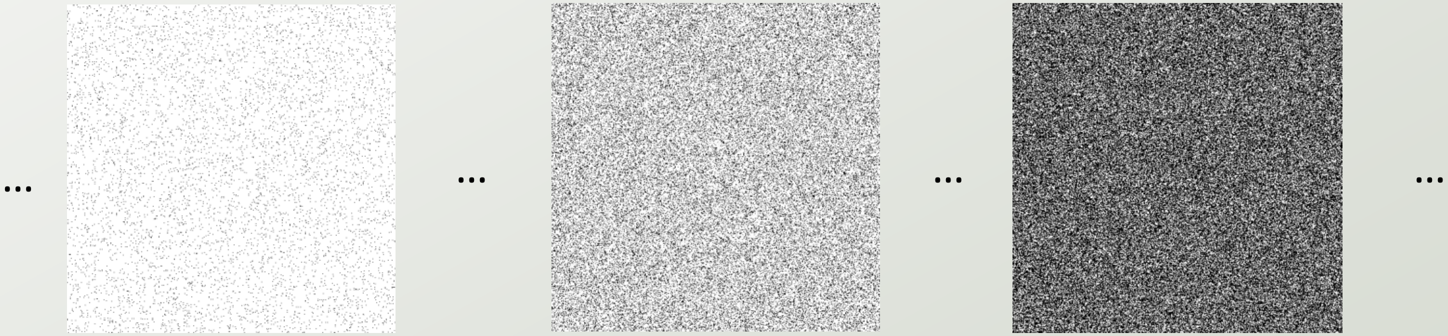
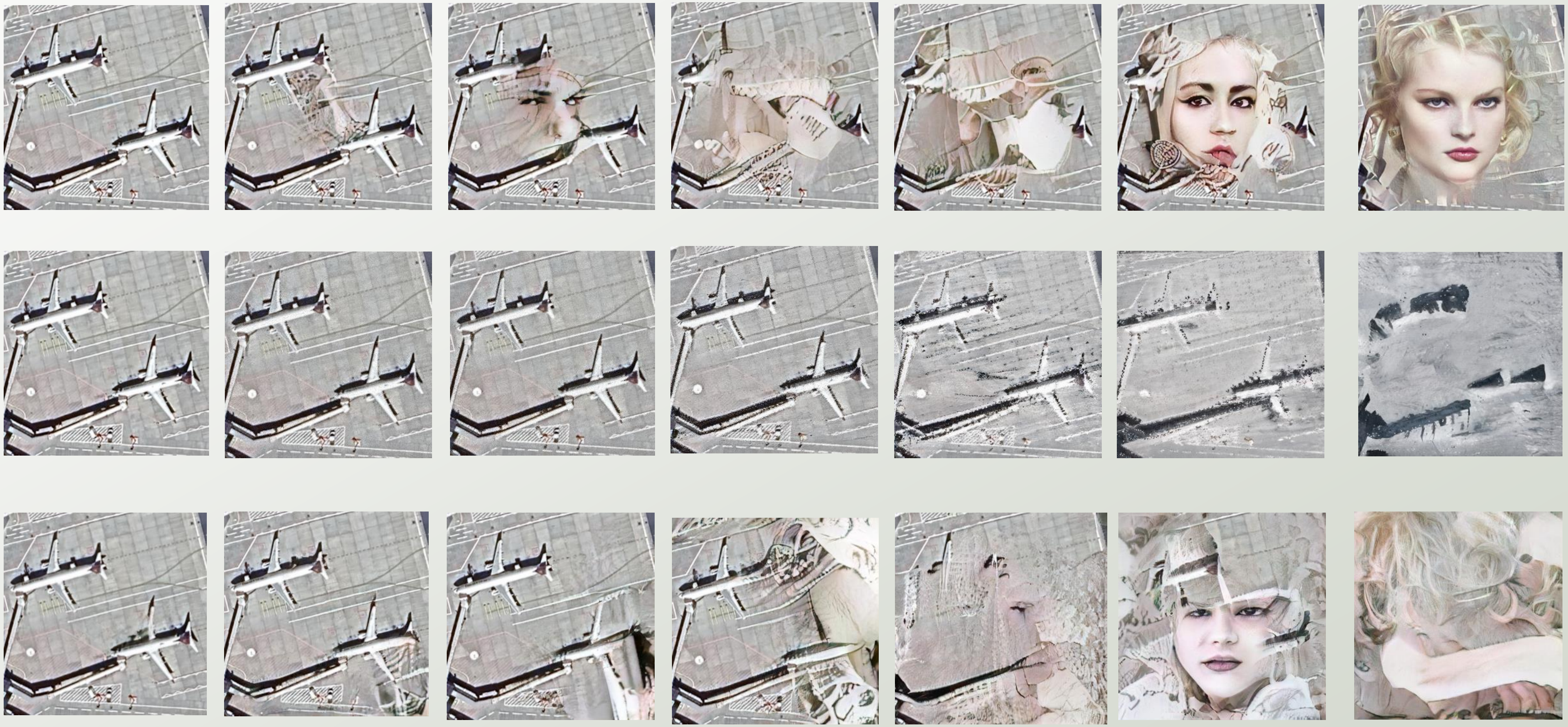


Image Inpainting with masks

Rectangular masks

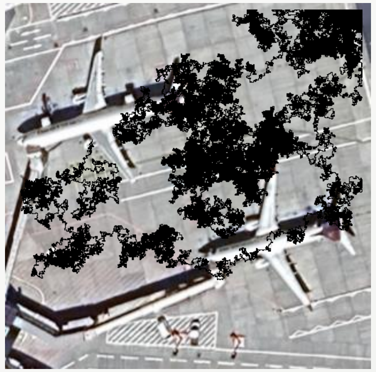
Random masks

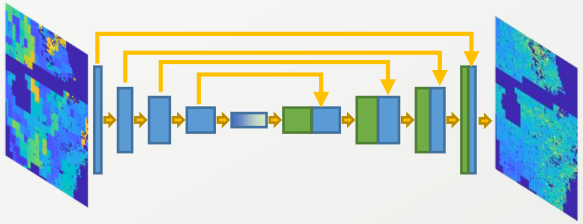
Irregular centered masks



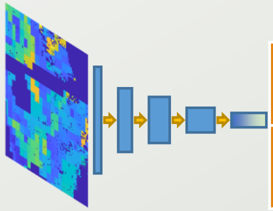
Percent missing 10% 20% 30% 40% 50% 60% 70%

Approach





inpainter



Discriminator

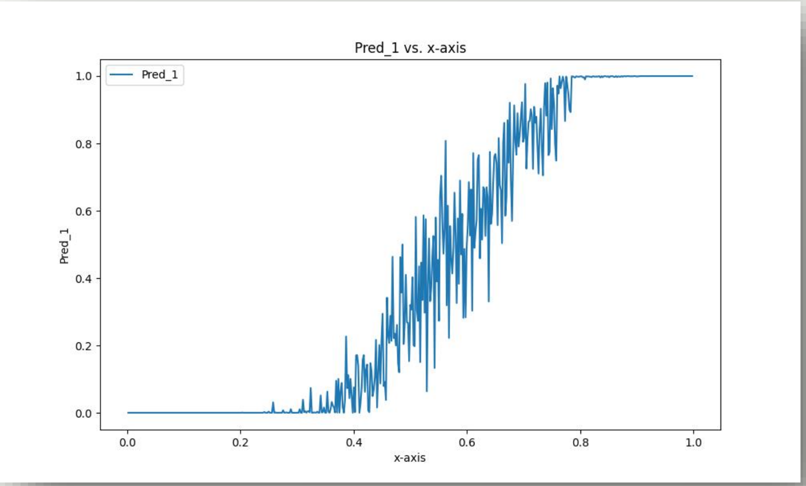
In distribution	0.2
Out of distribution	0.8

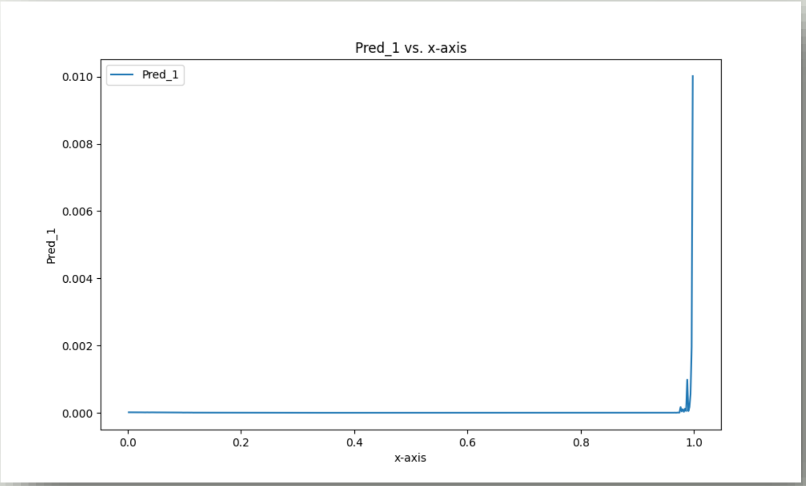
Initial results - Detection

Rectangular mask

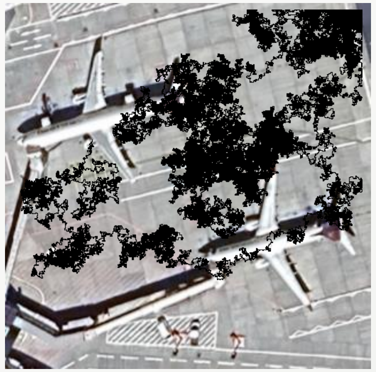
Random Mask

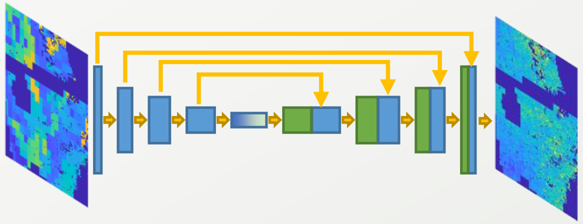
"Probability" of image being celebA



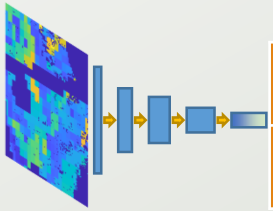


Approach



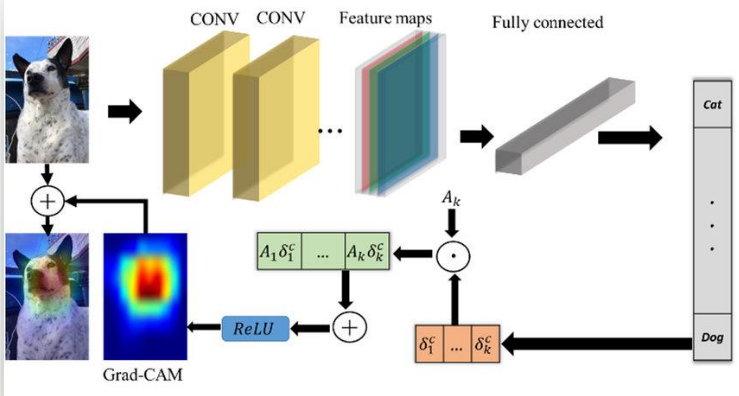


inpainter

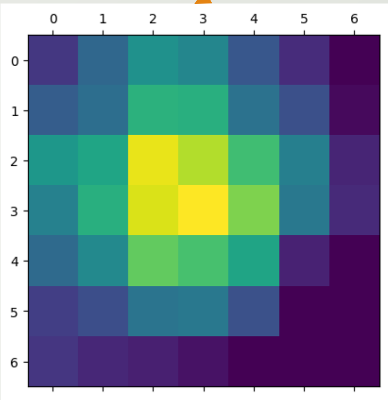


In distribution	0.2
Out of distribution	0.8

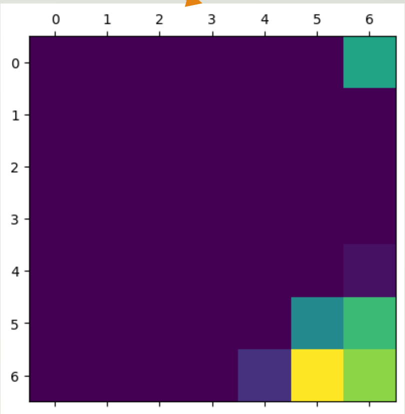
Discriminator



GradCAM



Heatmap for CelebA

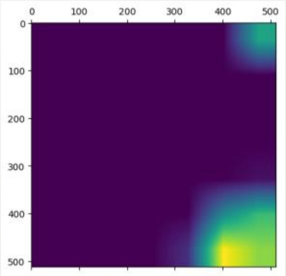


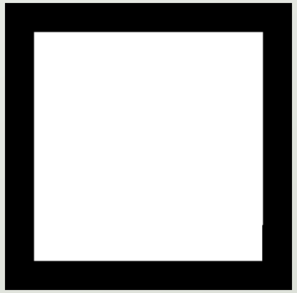
Heatmap for Land Use

Metrics

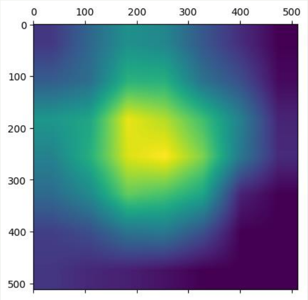
Upscaled heatmaps | Thresholded heatmaps | Modified mask

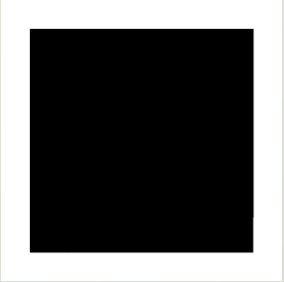
Heatmap for land use





Heatmap for celebA



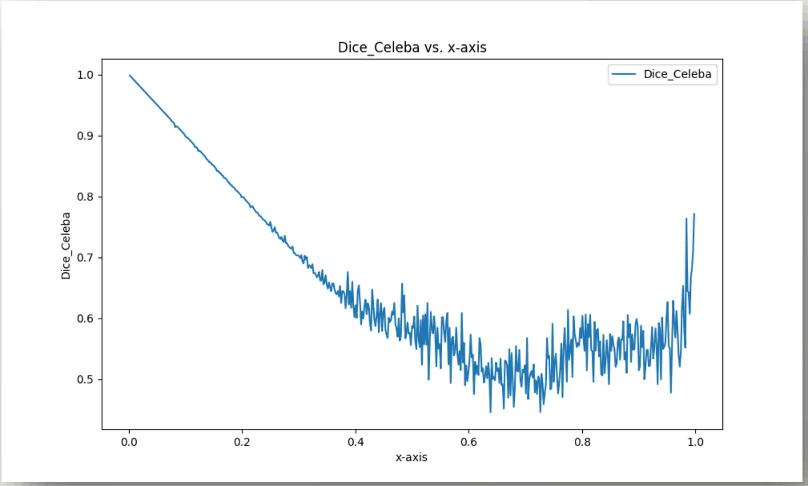


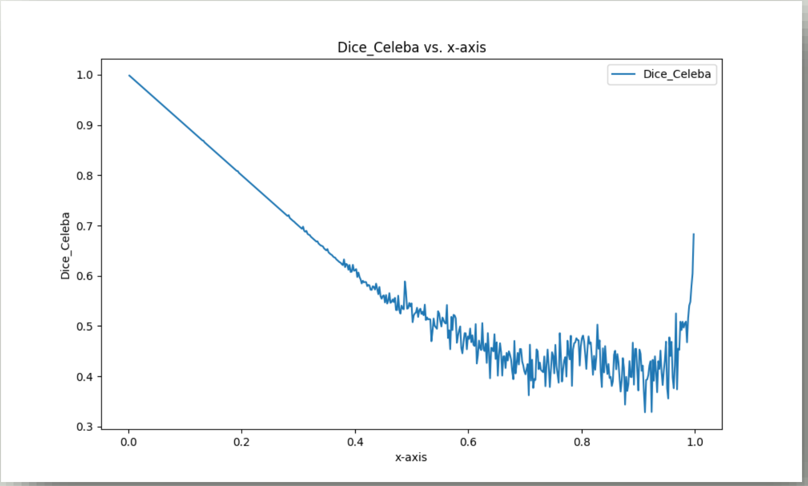
Initial results - Localization

Rectangular mask

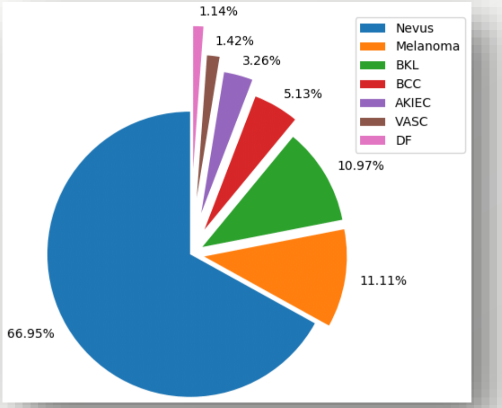
Random Mask

CelebA heatmap

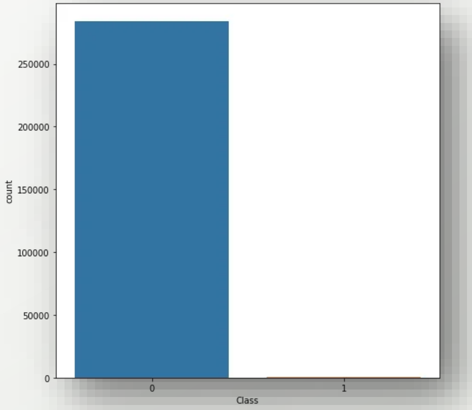




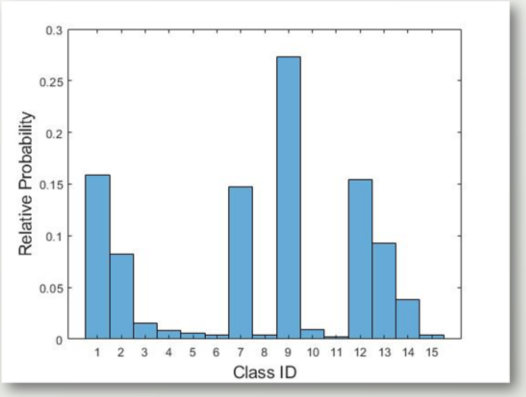
Challenge of imbalanced datasets



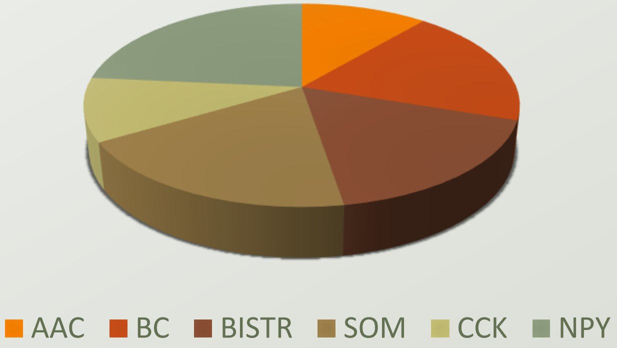
multi-source dermatoscopic images of 7 skin lesions



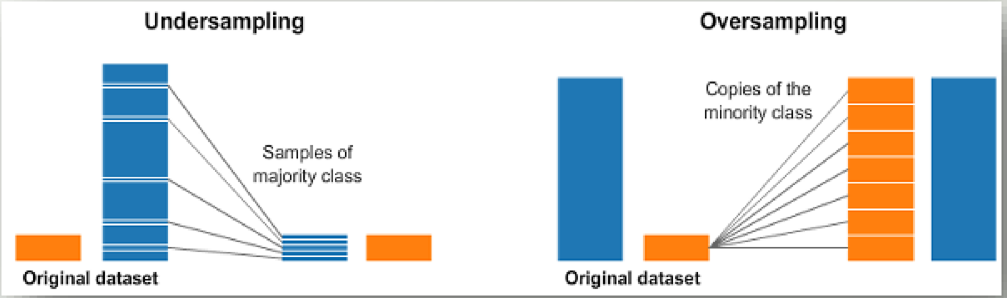
Fraudulent transactions

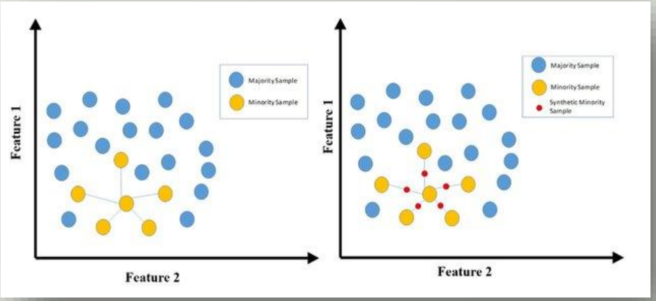


Land Cover \ Land Use

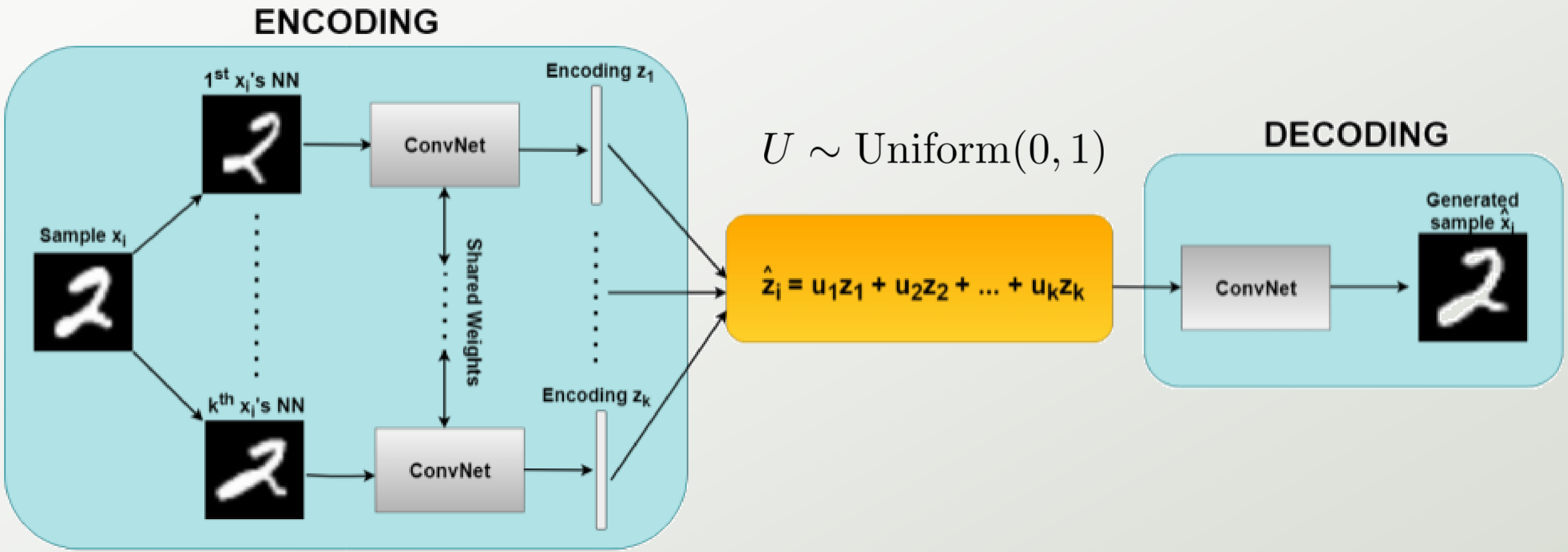


Neuron classification





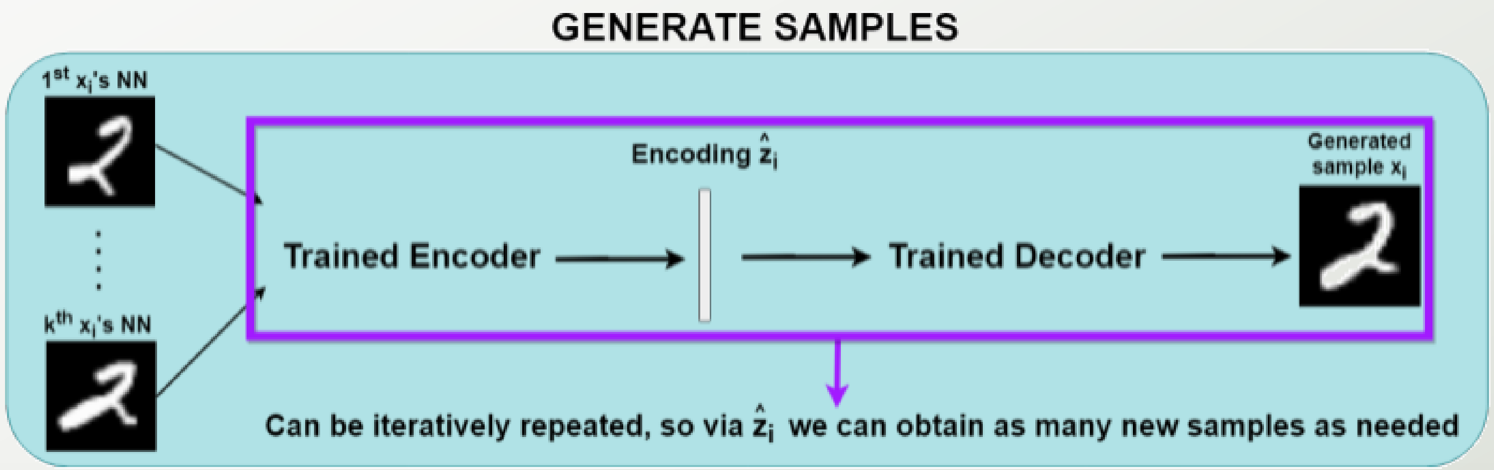
GENDA



$$L = \frac{1}{M}\sum_{i=1}^{M}(x_i - \hat{x}_i)^2 = \frac{1}{M}\sum_{i=1}^{M}(x_i - d(e(N(x_i))))^2$$

Troullinou, E., Tsagkatakis, G., Losonczy, A., Poirazi, P., & Tsakalides, P. A Generative Neighborhood-based Deep Autoencoder for Robust Imbalanced Classification. *IEEE Transactions on Artificial Intelligence,* 2023.

Results

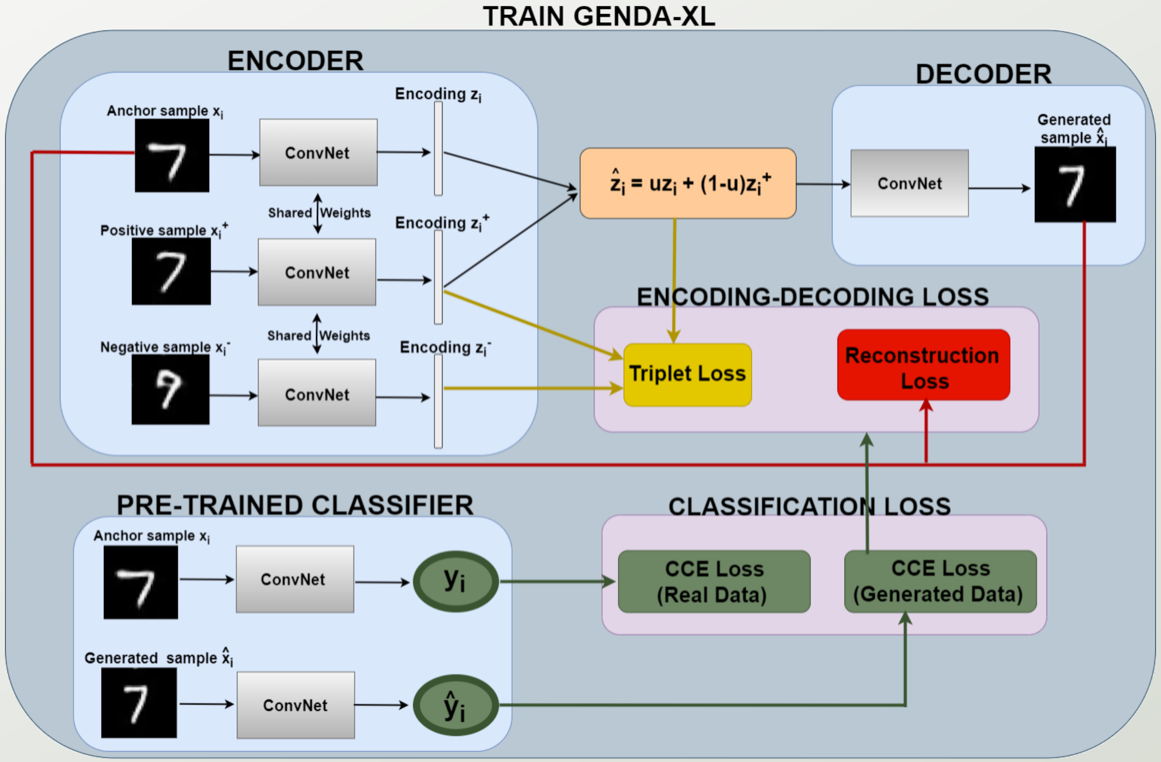


	HAR			TwoLeadECG			Ca^{2+} Imaging		
Method	ACSA	F1-Score	Precision	ACSA	F1-Score	Precision	ACSA	F1-Score	Precision
Baseline	0.605	0.536	0.555	0.5	0.342	0.26	0.65	0.674	0.714
SMOTE	0.731	0.682	0.652	0.81	0.823	0.815	0.77	0.78	0.792
TimeGAN	0.713	0.67	0.643	0.735	0.716	0.693	0.697	0.674	0.654
GENDA	**0.877**	**0.878**	**0.883**	**0.829**	**0.838**	**0.817**	**0.787**	**0.797**	**0.809**

	HAR		
Rebalancing Approach	ACSA	F1-Score	Precision
Oversampling	0.642	0.645	0.65
Undersampling	0.53	0.52	0.525

GENDA-XL

$$\min \sum_{i=1}^{M} \underbrace{\max\{\|\hat{z}_i - z_{i+}\|^2 - \|\hat{z}_i - z_{i-}\|^2 + m, 0\}}_{\text{Triplet Loss}} + \underbrace{\frac{1}{M}(x_i - \hat{x}_i)^2}_{\text{MSE}} - \underbrace{\sum_{j=1}^{D} t_{ij} \log \hat{y}_{ij}}_{\text{Cross-Entropy}}$$



E. Troullinou, G. Tsagkatakis, A. Losonczy, P. Poirazi, and P. Tsakalides, "A Generative Neighborhood-Based Deep Autoencoder with an Extended Loss Function for Robust Imbalanced Classification," in Proc. 57th Annual Asilomar Conference on Signals, Systems and Computers, 2023.

Performance

Method	MNIST			Fashion-MNIST		
	ACSA	F1-Score	Precision	ACSA	F1-Score	Precision
Baseline	0.579	0.563	0.54	0.499	0.475	0.454
SMOTE	0.895	0.894	0.883	0.738	0.708	0.712
DGC	0.948	0.947	0.911	0.836	0.831	0.781
BAGAN-GP	0.863	0.85	0.841	0.731	0.729	0.69
GENDA	0.925	0.922	0.926	0.811	0.801	0.794
GENDA-XL	**0.952**	**0.95**	**0.95**	**0.84**	**0.828**	**0.817**

Method	HAR			TwoLeadECG			Ca^{2+} Imaging		
	ACSA	F1-Score	Precision	ACSA	F1-Score	Precision	ACSA	F1-Score	Precision
Baseline	0.605	0.536	0.5	0.5	0.342	0.26	0.65	0.674	0.714
SMOTE	0.731	0.682	0.652	0.81	0.823	0.815	0.77	0.78	0.792
TimeGAN	0.713	0.67	0.643	0.735	0.716	0.693	0.697	0.674	0.654
GENDA	0.877	0.878	0.883	0.829	0.838	0.817	0.787	0.797	0.809
GENDA-XL	**0.919**	**0.918**	**0.919**	**0.938**	**0.934**	**0.927**	**0.834**	**0.846**	**0.859**

Open topics

Specify 1-2 “grand challenges”

- Classification vs Inverse problems
- Identify ML aspects (robustness, imbalance, recovery, etc.)

Specify and generate TITAN datasets

- Access to Simulations or Observations
- Dimensions, Characteristics, Storage, Open-access

Define performance metrics

- State-of-the-art solutions (codes & papers)
- SotA datasets


Thank you