Permutation invariance and uncertainty in multitemporal image super-resolution

Diego Valsesia, Enrico Magli
Politecnico di Torino

MULTI-IMAGE SUPER-RESOLUTION

- **Multiple images** of a scene (temporal series):
  - SR is possible thanks to sub-pixel misregistration
- Tremendous research interest since the ESA Proba-V challenge
- **DeepSUM** [1]: dynamic registration filters
- **HighRes-Net** [2]: recursive fusion
- **RAMS** [3]: feature attention
- **Burst computational photography** [4]

[1] Molini, Valsesia, Fracastoro, Magli, "*DeepSUM: Deep neural network for Super-resolution of Unregistered Multitemporal images*", IEEE TGRS 2020
[2] Arefin et al. "*Multi-Image Super-Resolution for Remote Sensing using Deep Recurrent Networks*", CVPRW 2020
[3] Salvetti, Mazzia, Khaliq, Chiaberge, "*Multi-Image Super Resolution of Remotely Sensed Images Using Residual Attention Deep Neural Networks*", MDPI Remote Sensing 2020
[4] Bhat, Danelljan, Van Gool, Timofte, "*Deep burst super-resolution*", CVPR 2021

A NEGLEGTED ISSUE…

- All state-of-the-art methods neglect an important issue:

The temporal ordering of the LR images does not matter

- **Full Invariance** to temporal permutation is needed
- Non-invariant models: averaging multiple permutations improves results → expensive!
- Building a model that is **invariant** allows for **richer representations**
  - no parameters are wasted learning correlation patterns exploiting ordering

INVARIANCE AND EQUIVARIANCE

- **Invariance**

$$f(g \circ x) = f(x)$$

(output does not change when input is permuted)

- **Equivariance**

$$f(g \circ x) = g \circ f(x)$$

(output has the same permutation as input)

EXAMPLE: CONV2D IS NOT INVARIANT

- Example: stack the **temporal** images as **channels**
  - (B, Nx, Ny, T)
- Using conv2D to process the tensor is NOT invariant to temporal permutation

$$h^{before} = w_1 x_1 + w_2 x_2 + \cdots + w_T x_T$$

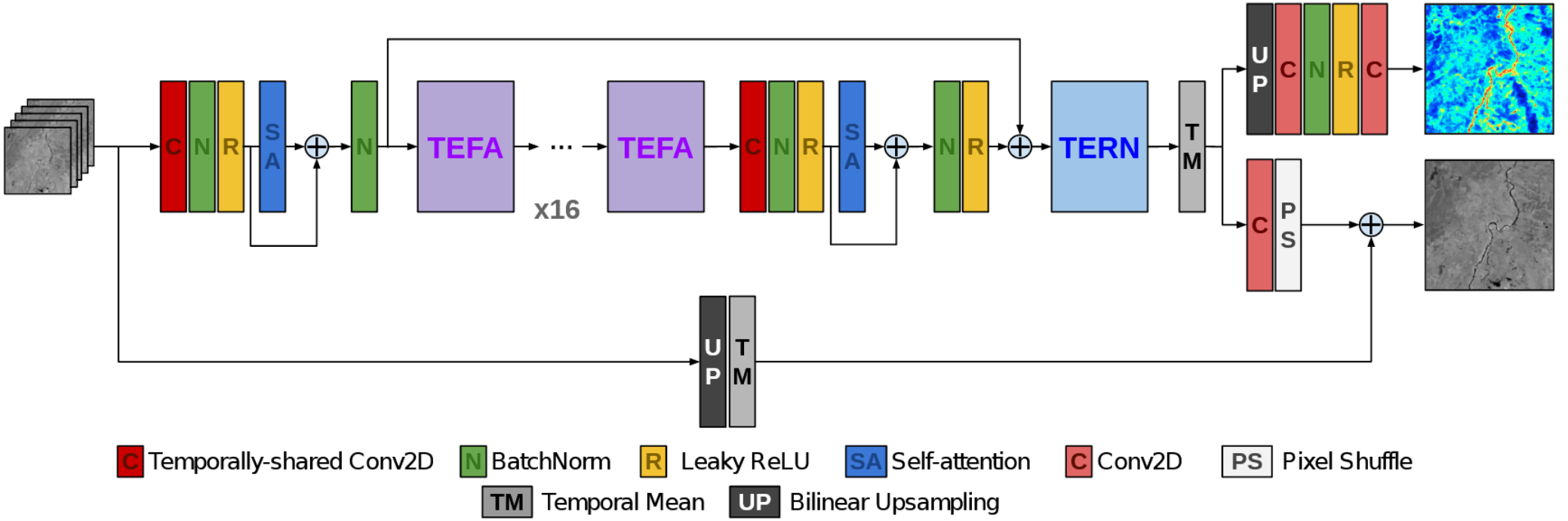
$$h^{after} = w_1 x_3 + w_2 x_1 + \cdots + w_T x_5$$

HOW DO YOU BUILD AN INVARIANT MODEL?

- **Step 1**: build a neural network with only **equivariant** layers
  - If temporal ordering is permuted, all feature maps are exactly the same but also permuted

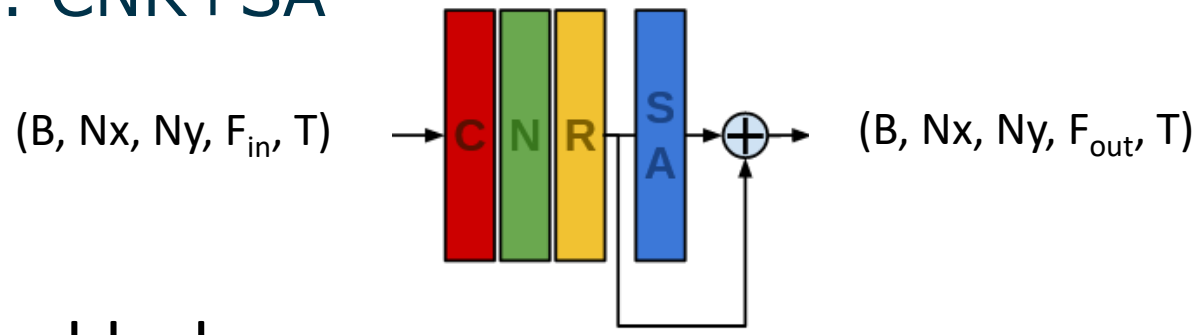
- **Step 2**: place an aggregation operation that is **invariant** to permutation (e.g. temporal average)

PIUnet MODEL



- Inspired by SoTA architectures using attention
- Backbone is redesigned to only use **equivariant layers**
- **Temporal Mean** is used to transform equivariance into invariance

BUILDING BLOCKS: CNR+SA



- **CNR-SA** building block:
  - C: spatial convolution, **shared** across time
  - N: batch normalization
  - R: leaky ReLU
  - SA: residual **temporal self-attention**

Mix space ← Equivariant to temporal permutation!

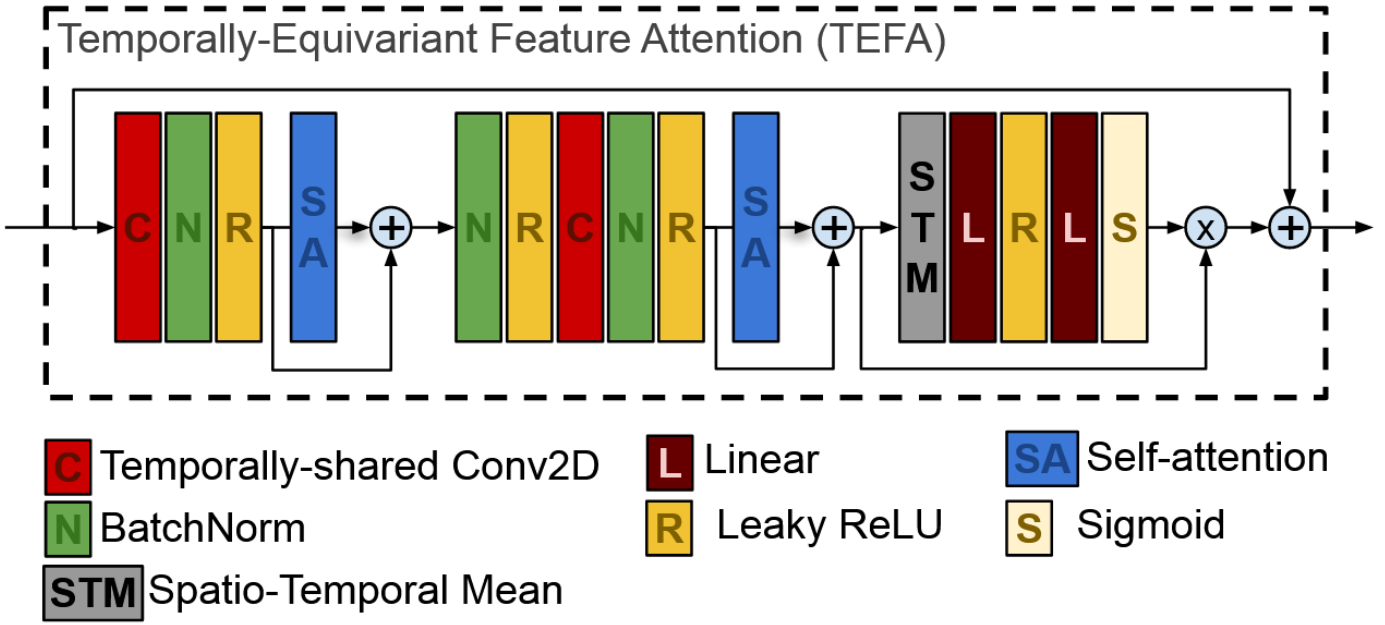
Mix time

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_k, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_v$$

$$\mathbf{Y} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{T}}\right) \cdot \mathbf{V} = \mathbf{A}\mathbf{V}$$

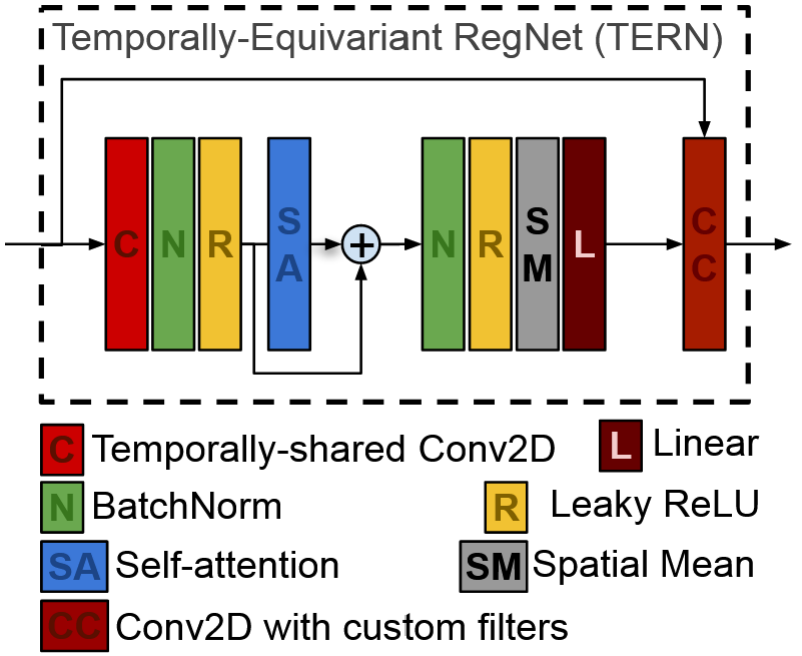
← Equivariant to temporal permutation!

BUILDING BLOCKS: TEFA



- **TEFA**: Temporally-Equivariant Feature Attention
  - Use attention mechanism to extract space-time features
  - Made equivariant by CNR-SA
  - Backbone of the model (16 TEFA blocks)

BUILDING BLOCKS: TERN



- **TERN**: Temporally-Equivariant Registration Network
  - Expanded from DeepSUM and made equivariant
  - Dynamically compute and apply filters to finetune pixel registration
    - Network computes a filter kernel as function of input and then applies it to input (**adaptive filter**)

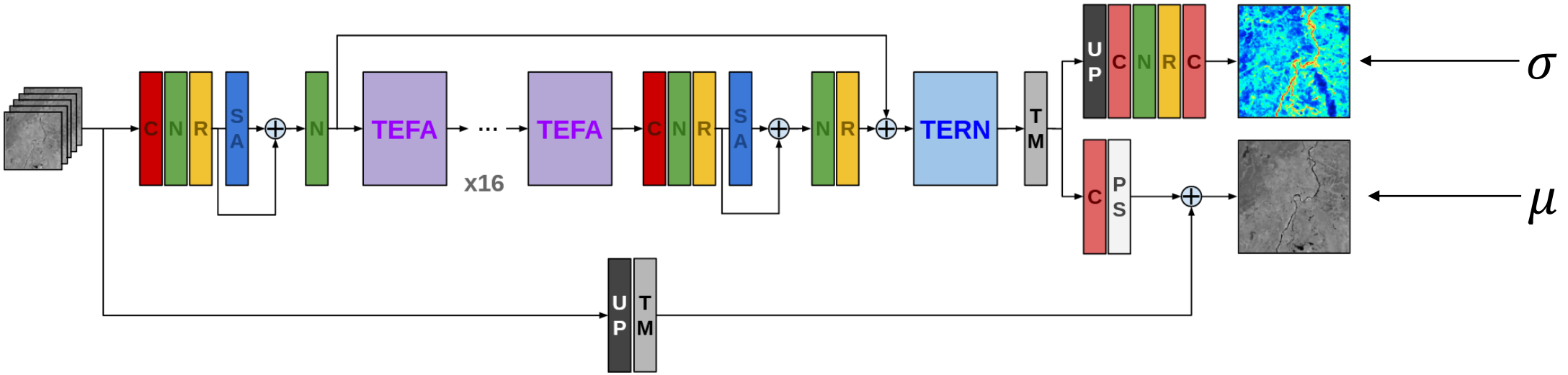
UNCERTAINTY ESTIMATION

- **Explainable AI**:
  - It is important to understand what the model is doing and how reliable is the output

Can I trust the super-resolved image when an area had a lot of temporal variation?

UNCERTAINTY ESTIMATION

- We want to estimate the **aleatoric uncertainty** of the SR image
- This uncertainty is due to the quality of the input data:
  - temporal variation increases uncertainty
- Model **output** is not just the SR pixel but the **SR pixel distribution**
  - **Two heads: mean value, standard deviation**



SR PIXEL MODEL & TRAINING

- We choose the SR pixel to have a **Laplace distribution**

$$p(x_i) = \frac{1}{2\beta_i} \exp\left(-\frac{|x_i - \mu_i|}{\beta_i}\right)$$

- Training minimizes the **negative log likelihood**

$$L = -\frac{1}{NB} \sum_{b,i} \log p(x_i)$$

$$= \frac{1}{NB} \sum_{b} \left[\sum_{i} \left(\delta_i^{(b)} + e^{-\delta_i^{(b)}} |x_i^{\mathrm{HR}(b)} - \mu_i^{(b)}| \right) \right]$$

This is similar to the L1 loss but weighted by the uncertainty!

($\delta = \log \beta$ is used for numerical stability)

EXPERIMENTAL RESULTS

- **Proba-V dataset** (T=9)
- Note: (ens.) methods use temporal self-ensembling

TABLE I: Quantitative performance - cPSNR (dB) and cSSIM

	Bicubic	IBP [17]	BTV [22]	DUF [27]	HighResNet [31]	DeepSUM (ens.) [29]	DeepSUM++ (ens.) [30]	RAMS [32]	RAMS [32] (ens.)	**PIUnet**
NIR cPSNR	45.44	45.96	45.93	47.06	47.55	47.84	47.93	48.23	48.51	**48.72**
NIR cSSIM	0.9771	0.9778	0.9794	0.9842	0.9855	0.9858	0.9862	0.9875	0.9880	**0.9883**
RED cPSNR	47.34	48.21	48.12	49.36	49.75	50.00	50.08	50.17	50.44	**50.62**
RED cSSIM	0.9840	0.9865	0.9861	0.9842	0.9904	0.9908	0.9912	0.9913	0.9917	**0.9921**

- PIUnet beats SoTA and also beats self-ensembling

SELF-ENSEMBLING IS EXPENSIVE

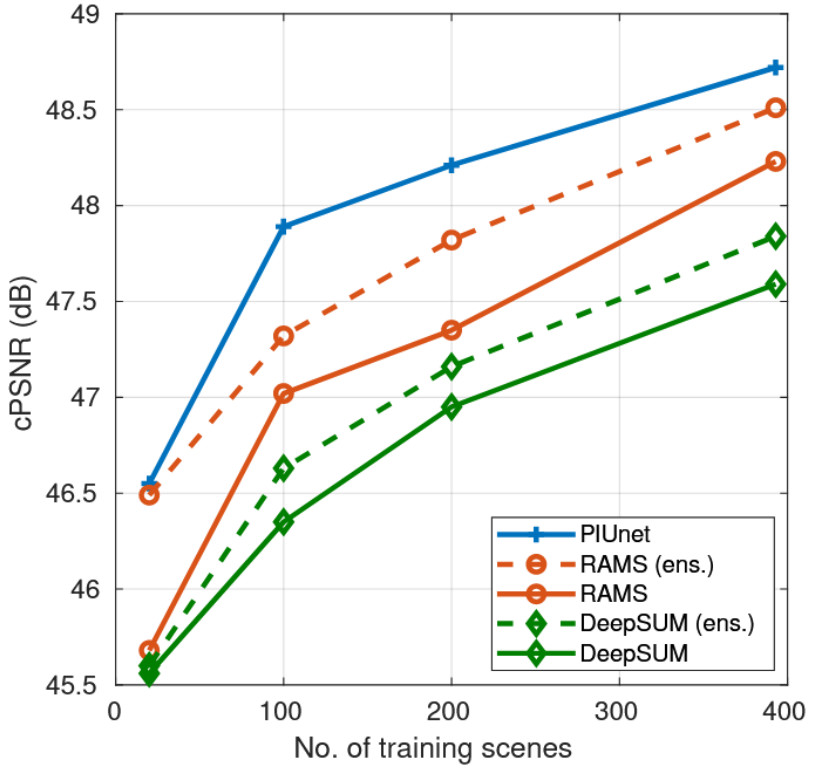
- Averaging the results of multiple permutations is **expensive**
- PIUnet does **not** do that (you always get the same result for every permutation)
  - You are capturing the **richness** of the self-ensemble **directly in the model**

TABLE III: Computational complexity

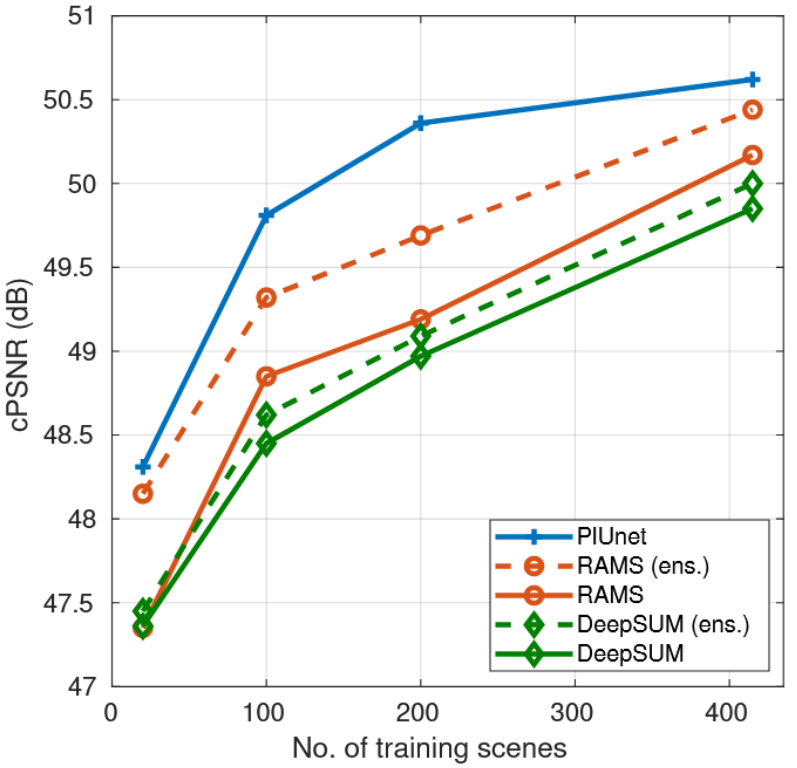
	runtime	memory
DeepSUM	484 ms	5420 MB
RAMS	102 ms	1250 MB
RAMS (ens.)	1642 ms / 1075 ms	1250 MB / 5340 MB
PIUnet	181 ms	1280 MB

IMPROVED DATA EFFICIENCY

- We do not waste training data to learn operations depending on temporal ordering
  - Increased data efficiency → DeepSUM performance with just 25% of the data!



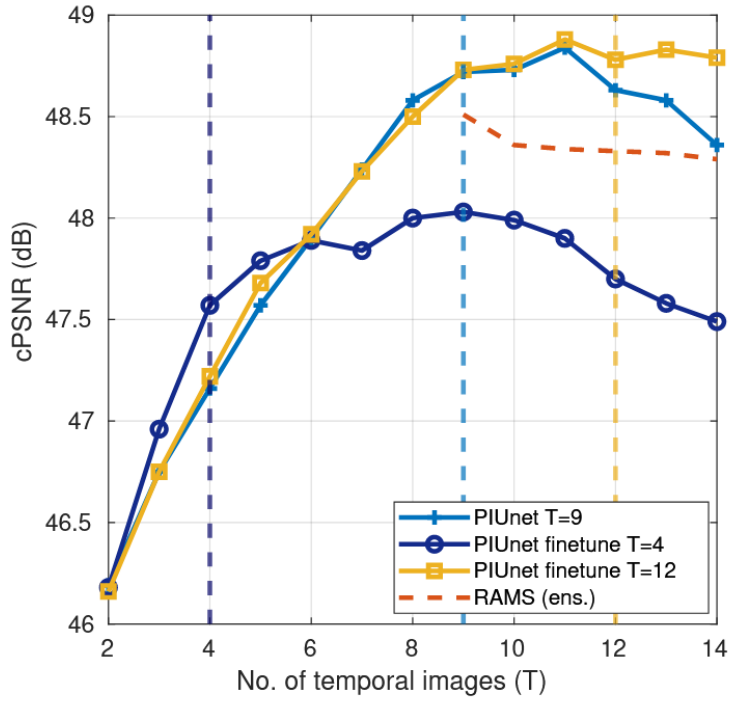
(a) NIR



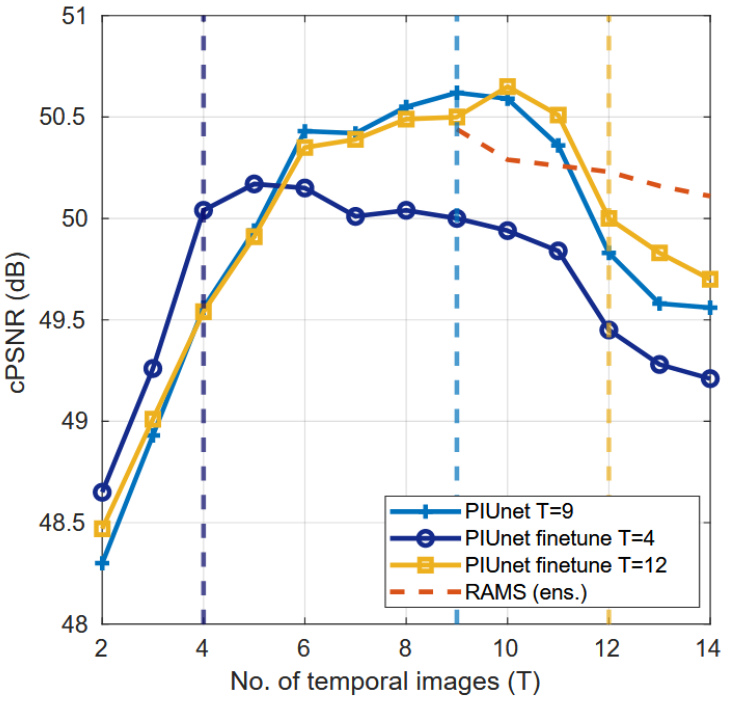
(b) RED

PROCESS ANY NUMBER OF LR IMAGES

- Most models are rigid: you can only use a fixed number of images
  - This is hard-coded in how the architecture is made (strided convs, etc.)
- PIUnet can process **any number** of LR images



(a) NIR



(b) RED

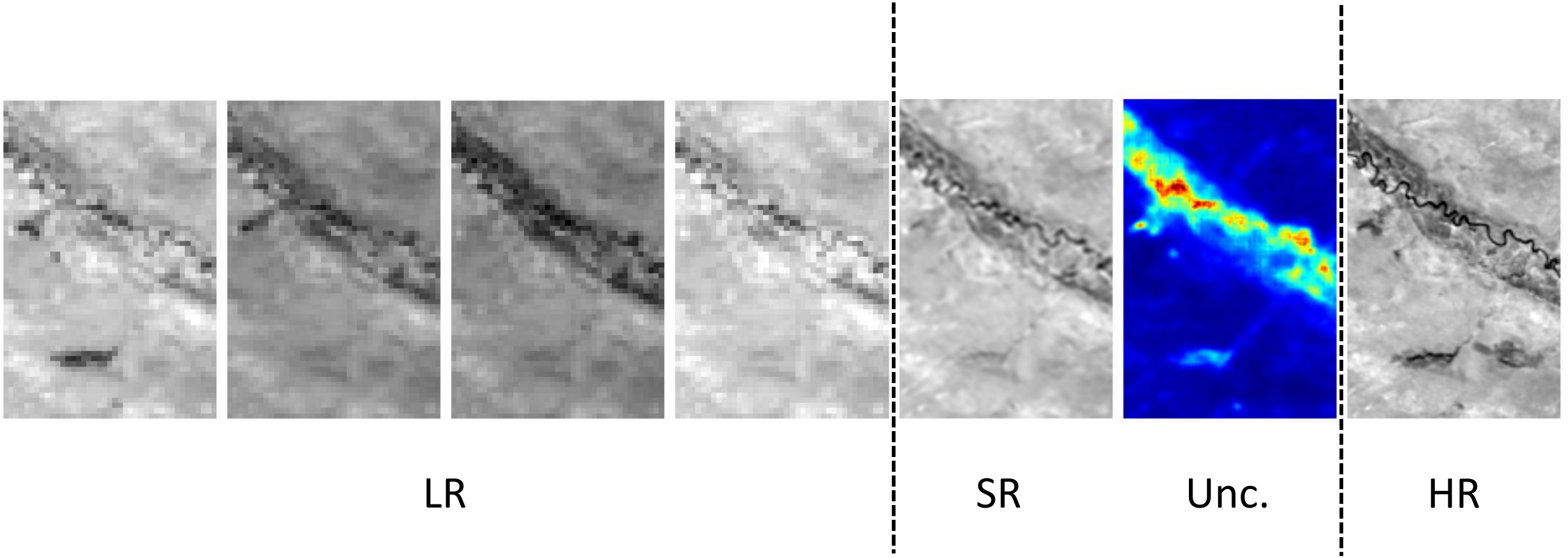
UNCERTAINTY ALSO BOOSTS PERFORMANCE

- Estimating uncertainty provides useful explainability…
- …but also increases model **performance**!
- Uncertainty **regularizes** overconfident predictions

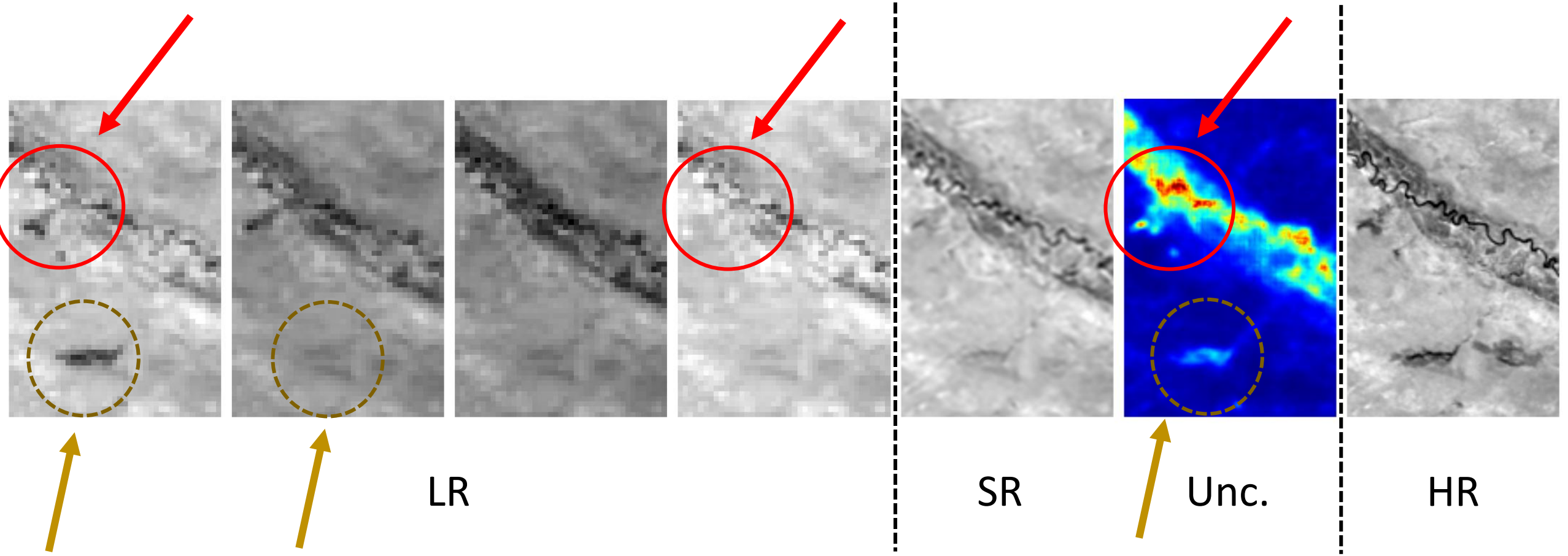
TABLE IV: Training loss comparison (cPSNR)

	L1 loss	NLL loss
NIR	48.41 dB	**48.72** dB
RED	50.53 dB	**50.62** dB

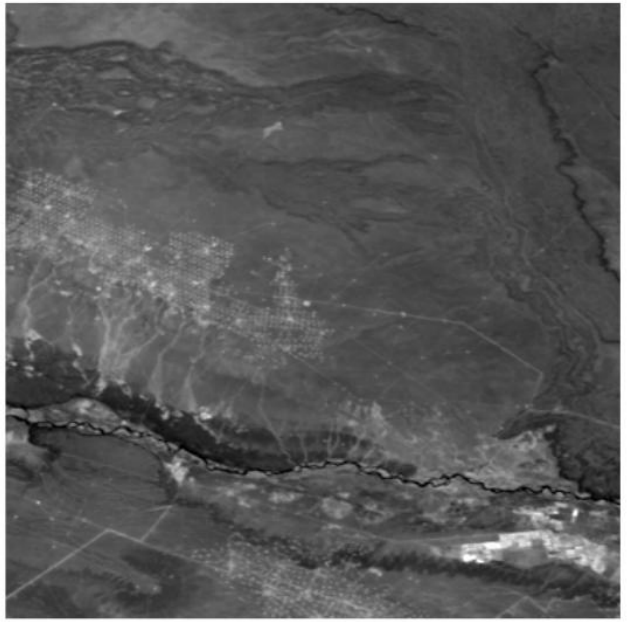
UNCERTAINTY MAP CORRELATES WITH TEMPORAL VARIATION



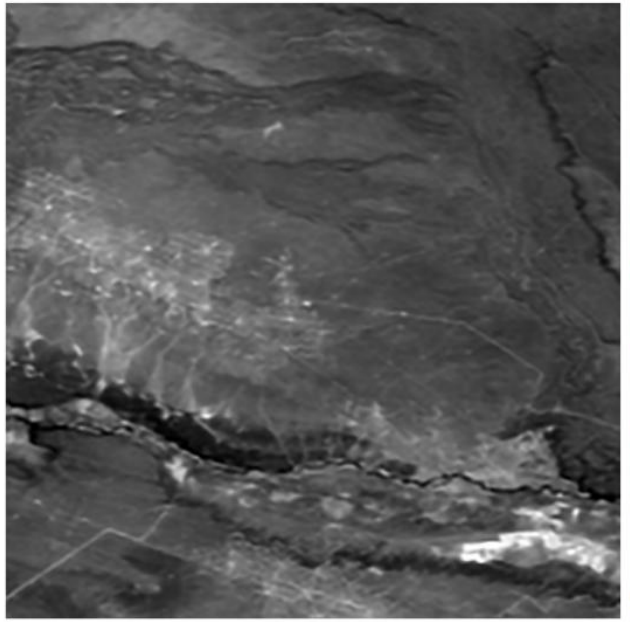
UNCERTAINTY MAP CORRELATES WITH TEMPORAL VARIATION



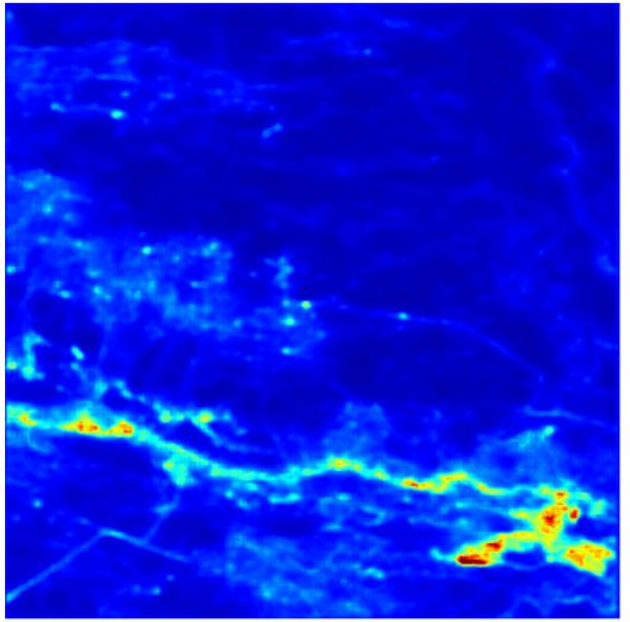
UNCERTAINTY MAP CORRELATES WITH TRUE ERROR



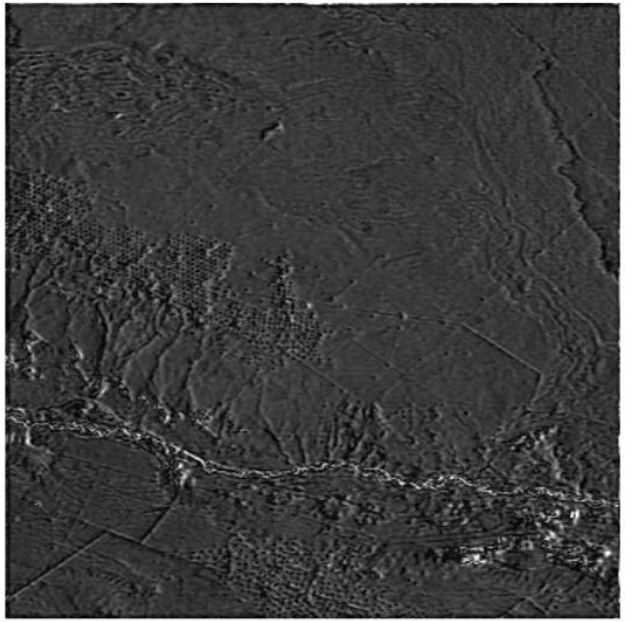
HR



SR



Unc.

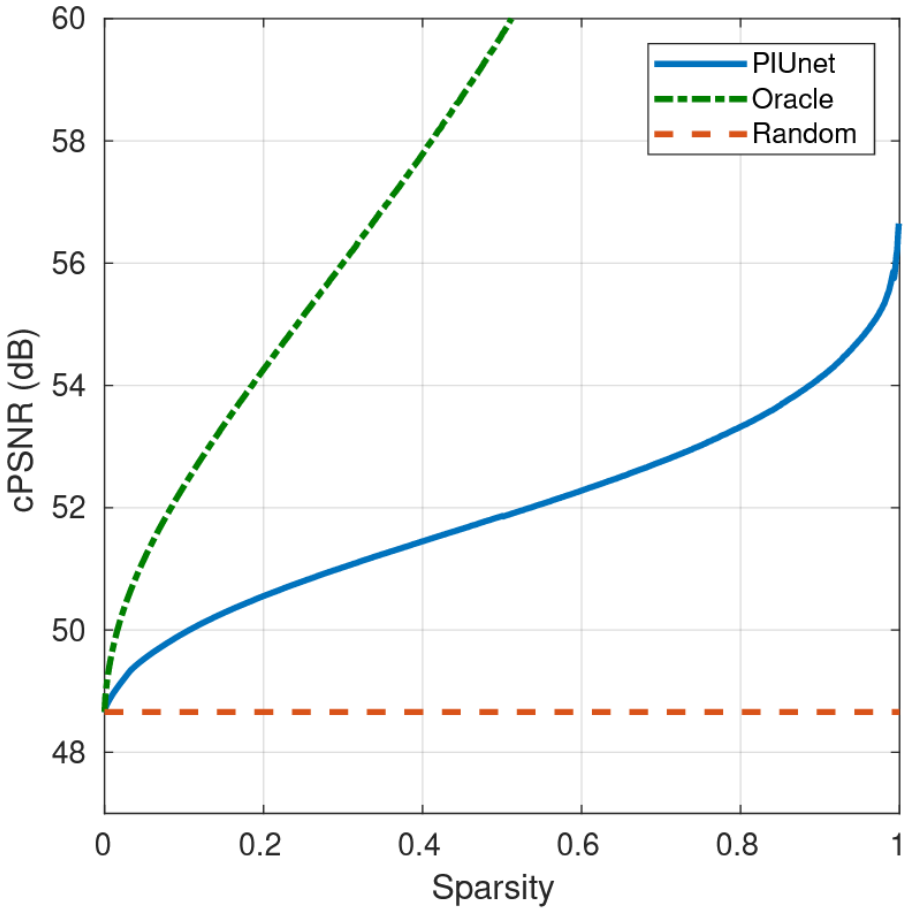


Absolute error

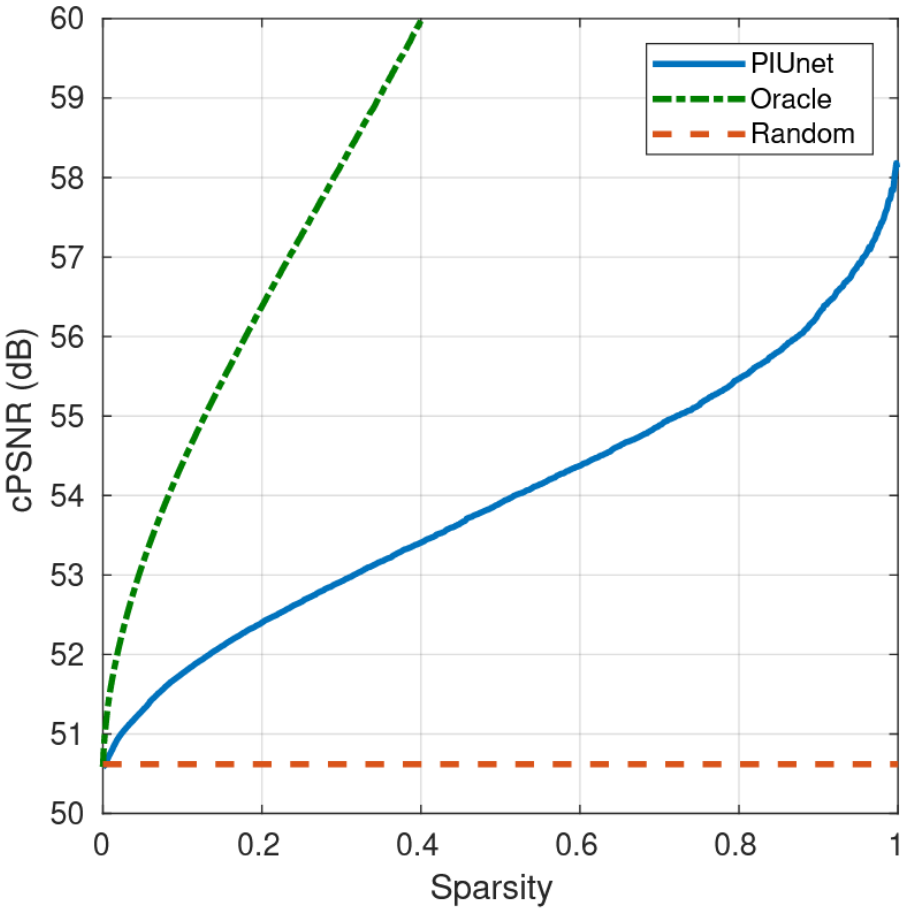
UNCERTAINTY MAP CORRELATES WITH TRUE ERROR

- How do you **quantitatively** check if uncertainty estimation is good?
- **Sparsification curve**:
  - Sort pixels by decreasing uncertainty
  - Remove one pixel at a time
  - If uncertainty correlates with error, PSNR of surviving pixels increases
- Compare methods of uncertainty estimation by comparing sparsification curves

UNCERTAINTY MAP CORRELATES WITH TRUE ERROR



(a) NIR



(b) RED

CONTACTS

diego.valsesia@polito.it

github.com/diegovalsesia

www.ipl.polito.it

@IPLPolito

Discover more about super-resolution, permutation invariance and SR uncertainty estimation:
https://arxiv.org/abs/2105.12409
Valsesia, Magli, «*Permutation invariance and uncertainty in multitemporal image super-resolution*»
Code (soon): https://github.com/diegovalsesia/piunet