Deep Unfolding Transformers for Sparse Recovery of Video

Brent De Weerdt, *Student Member, IEEE*, Yonina C. Eldar, *Fellow, IEEE*, and Nikos Deligiannis, *Member, IEEE*

***Abstract*—Deep unfolding models are designed by unrolling an optimization algorithm into a deep learning network. By incorporating domain knowledge from the optimization algorithm, they have shown faster convergence and higher performance compared to the original algorithm. We design an optimization problem for sequential signal recovery, which incorporates that the signals have a sparse representation in a dictionary and are correlated over time. A corresponding optimization algorithm is derived and unfolded into a deep unfolding Transformer encoder architecture, coined DUST. To show its improved reconstruction quality and flexibility in handling sequences of different lengths, we perform extensive experiments on video frame reconstruction from low-dimensional and/or noisy measurements, using several video datasets. We evaluate extensions to the base DUST model incorporating token normalization and multi-head attention, and compare our proposed networks with several deep unfolding recurrent neural networks (RNNs), generic unfolded and vanilla Transformers, and several video denoising models. The results show that our proposed Transformer architecture improves the reconstruction quality over state-of-the-art deep unfolding RNNs, existing Transformer networks, as well as state-of-the-art video denoising models, while significantly reducing the model size and computational cost of training and inference.**

***Index Terms*—Deep unfolding, transformer networks, sparse recovery, video compressed sensing, video denoising.**

I. Introduction

IN many imaging applications, one needs to reconstruct images, videos, or other data from low-dimensional and/or noisy measurements, for example in dynamic magnetic resonance imaging [1], radar target detection [2], compressive video sensing [3] and high-speed hyperspectral video acquisition [4]. In order to make reconstruction of the data possible, additional knowledge about the signal's properties is incorporated into the reconstruction algorithm. Sparse recovery uses the fact that the signals have a sparse representation with respect to a dictionary, such as natural images, which can often be represented by a sparse set of coefficients in the frequency domain.

There are many works on sparse recovery of temporal signal sequences, e.g., [5], [6], [7], [8]. In [5], they propose a method to dynamically determine the number of random measurements needed to reconstruct a signal, and [6] proposes the modified-CS approach to reconstruct sparse signals where part of the signal is known through prior knowledge or the previous reconstructed signal in a sequence. The authors of [7] and [8] model the temporal correlation between signals using an ℓ_1 or ℓ_2 norm, which results in ℓ_1-ℓ_1 or ℓ_1-ℓ_2 minimization problems. These optimization algorithms generally need a large number of iterations to converge, resulting in a high computational cost. This problem is exacerbated when the problem dimensionality increases. With the rise of deep neural networks (DNNs), many networks have been designed for recovery problems in image processing and medical imaging. Such DNNs offer improved reconstruction quality compared to iterative algorithms, while the computational cost is moved to the training phase, resulting in fast reconstruction times once the model is trained.

For the task of image compressed sensing (CS), [9], [10] proposed convolutional neural networks (CNNs) to reconstruct images, and [11] proposed a dual path CNN with added attention modules, one path for the image's structure and one focused on the texture. The authors in [12] designed a CNN and Transformer hybrid network, which aggregates the features from the CNN branch into the Transformer branch. For the recovery of videos, [13] designed CNNs to extract spatial features from compressed video frames and a recurrent neural network (RNN) to model the temporal correlations and reconstruct a sequence of frames. The authors of [14] introduced an RNN for single-pixel video reconstruction, and [15] proposed a Transformer network [16] that uses the spatial and temporal correlations to reconstruct high-speed video where groups of multiple frames are compressed into single measurements. The downside of DNNs is that they are black box models that lack interpretability and prior domain knowledge about the data at hand [17]. This lack of prior knowledge requires them to train on large amounts of data to learn efficient representations.

Recently, there have been efforts to combine the learning ability and fast reconstruction of DNNs with the domain

Manuscript received 24 July 2023; revised 21 December 2023; accepted 17 March 2024. Date of publication 25 March 2024; date of current version 9 April 2024. This work was supported by the FWO Flanders Ph.D. Fellowship Strategic Basic Research under Grant 1S44523N. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Samet Oymak. *(Corresponding author: Brent De Weerdt.)*

Brent De Weerdt and Nikos Deligiannis are with the Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel, B-1050 Brussels, Belgium, and also with the imec, B-3001 Leuven, Belgium (e-mail: brent.de.weerdt@vub.be; ndeligia@etrovub.be).

Yonina C. Eldar is with the Department of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 7610001, Israel (e-mail: yonina.eldar@weizmann.ac.il).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSP.2024.3381749, provided by the authors.

Digital Object Identifier 10.1109/TSP.2024.3381749

knowledge encapsulated in optimization algorithms [18]. Particularly, the approach of deep unfolding aims to unroll existing optimization algorithms into their individual iterations, and map operations to neural network layers that are equivalent or similar in nature [19]. The computations performed by this unfolded neural network are close to the original algorithm, but its parameters can be learned and improved through training on data. In the context of sparse recovery, examples include LISTA [20], a Learned version of the Iterative Soft Thresholding Algorithm (ISTA) [21], the unfolded version of the Alternating Direction Method of Multipliers (ADMM), called ADMM-Net [22], and the learned Approximate Message Passing (AMP) network [23].

For compressive sensing, several recent works designed deep unfolding based neural networks for the reconstruction of images from low-dimensional measurements, such as ISTA-Net [24] and its successor ISTA-Net++ [25], which are based on ISTA, AMP-Net [26], and ADMM-DAD [27], which are unfoldings of AMP and ADMM, respectively. The authors of [28] used the Neumann series expansion to design Neumann networks that can be used for various inverse problems in imaging, and the study in [29] proposed a hybrid Transformer-CNN model for compressed sensing that incorporates ISTA-inspired elements.

For sparse recovery of sequential data, particularly video reconstruction, the number of deep unfolding networks is limited. In [30], ISTA is modified into a sequential version, coined SISTA, which is then unfolded into a recurrent neural network SISTA-RNN. The authors of [31] and [32] start from an ℓ_1-ℓ_1 minimization problem and a reweighted version, respectively, where the derived optimization algorithms are then unfolded into respectively the ℓ_1-ℓ_1-RNN and reweighted-RNN model.

Recently, the work in [33] designed an energy function and corresponding optimization algorithm that unfolds into a (simplified) Transformer encoder network. The Transformer architecture was introduced in [16], and has achieved state-of-the-art results in language and vision tasks [34], [35], [36], [37], [38], [39], [40], [41]. This neural network architecture consists of several Transformer blocks stacked on top of each other, where each block takes as input a sequence of vectors (tokens), employs a self-attention operation on the tokens to exchange context information between them, then applies the same nonlinear transformation on each token in parallel, and passes the transformed sequence to the next block or the output. While the authors of [33] empirically showed that their unfolded Transformer network converges towards the solution of the corresponding optimization problem, no experiments were conducted in which the network is trained and evaluated on any practical regression or classification task. We even show that without some modification, this network does not perform well at all on the sparse recovery tasks. Furthermore, the generic unfolded Transformer does not incorporate priors specific to sparse recovery, resulting in a lower reconstruction performance compared to e.g., deep unfolding RNNs, which do take into account this prior knowledge. To the best of our knowledge, our work is the first to propose a deep unfolding Transformer architecture specifically designed for sparse recovery problems.

In this paper, we propose an energy function for sparse recovery of sequential signals, by incorporating priors on 1) the sparsity of signals in a dictionary and 2) the correlation between signals over the entire sequence. Building upon the framework in [33], we derive a new optimization algorithm to minimize our energy function and unfold it into a Transformer network coined DUST. Our deep unfolding Transformer model has a modified self-attention mechanism, different linear projections, and a different activation function compared to the model in [33], making it tailored to the task of sequential sparse recovery. We extend our initial work in [42] with additional modifications to the architecture, namely the normalization of attention tokens to improve training convergence and the introduction of multi-head attention. Moreover, we provide an extensive evaluation of our proposed architectures on (noisy) compressed sensing, video denoising, and the reconstruction of longer sequences and compare them to state-of-the-art deep unfolding RNNs, the unfolded Transformer in [33] and a vanilla Vision Transformer network. Furthermore, we compare our networks with multiple state-of-the-art video denoising models. The reconstruction performance in compressed sensing and denoising is evaluated on multiple video datasets, showing our DUST network improves reconstruction quality, while significantly reducing the computational cost and number of parameters at the same time. The specific contributions of this work are:

- We design an energy function, derive a corresponding optimization algorithm and unfold it into a Transformer network, named DUST. It differs from [33] by incorporating the priors necessary for sparse recovery of sequences. The Transformer also operates very differently compared to the deep unfolding RNNs. RNNs reconstruct a sequence in a recurrent fashion, focusing only on the current and previous signal in the sequence, while the Transformer can handle long-range correlations throughout the full sequence to improve reconstruction quality.
- We present two extensions to our basic unfolded Transformer architecture, namely the normalization of attention keys/queries, which significantly improves the performance of the model, and the design of a multi-head attention version, called MH-DUST.
- We provide extensive experiments for the evaluation of DUST and MH-DUST on the task of video frame reconstruction from low-dimensional and/or noisy measurements, using multiple real-world video datasets. The results show that our proposed models outperform state-of-the-art deep unfolding RNNs, generic Transformer networks, and state-of-the-art video denoising models in reconstruction quality, number of parameters, and computational cost for training and inference.

The remainder of the paper is organized as follows: Section II describes background and related work, Section III presents our base DUST model for sparse recovery of signal sequences and several extensions. Section IV presents experiments on video sparse recovery and video denoising, and Section V concludes the paper.

II. Background and Related Work

We seek to recover a sequence of signals $\mathbf{s}_t \in \mathbb{R}^n$, with $t = 1, \ldots, T$ from a series of compressive and/or noisy measurements:

$$\mathbf{x}_t = \mathbf{A}\mathbf{s}_t + \boldsymbol{\epsilon}_t, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}(m \ll n)$ is the measurement matrix or sensing matrix and $\boldsymbol{\epsilon}_t \in \mathbb{R}^m$ is additive noise. Since this is an underdetermined system of equations, we need additional constraints in order to recover the original signals. We assume that the signals $\mathbf{s}_t$ have a sparse representation $\mathbf{h}_t \in \mathbb{R}^d$ in a dictionary $\mathbf{D} \in \mathbb{R}^{n \times d}$, that is, $\mathbf{s}_t = \mathbf{D}\mathbf{h}_t$. Furthermore, the correlation of signals over time is used to further improve reconstruction performance.

A. Learned Iterative Soft Thresholding Algorithm (LISTA)

When no temporal correlation is considered, the signals $\mathbf{s}_t$ can be recovered separately, by minimizing the following energy function with respect to $\mathbf{h}_t$, for each time step t [43]:

$$E(\mathbf{x}_t, \mathbf{h}_t) = \frac{1}{2}\|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t\|_2^2 + \lambda_1\|\mathbf{h}_t\|_1, \tag{2}$$

where $\|\cdot\|_p$ is the ℓ_p norm and λ_1 is a regularization parameter. A popular algorithm for this optimization problem is ISTA [21], which consists of the following iterations:

$$\mathbf{h}_t^{(k+1)} = \phi_{\frac{\lambda_1}{c}}\left(\mathbf{h}_t^{(k)} + \frac{1}{c}\mathbf{D}^T\mathbf{A}^T\left(\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t^{(k)}\right)\right), \tag{3}$$

where $\phi_\gamma(u) = \text{sign}(u)\max(0, |u| - \gamma)$ is the soft thresholding function and c is an upper bound on the Lipschitz constant of the gradient of $\frac{1}{2}\|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t\|_2^2$. From the solution $\mathbf{h}_t^*$, we obtain the reconstructed signal by multiplication with the dictionary $\mathbf{D}$, that is, $\mathbf{s}_t^* = \mathbf{D}\mathbf{h}_t^*$.

The iterations of ISTA can be unrolled into a feedforward neural network called LISTA [20], such that each layer $l = 1, \ldots, L$, corresponds to one iteration of ISTA and has the form:

$$\mathbf{h}_t^{(l+1)} = \phi_{\frac{\lambda_1}{c}}\left(\mathbf{U}\mathbf{h}_t^{(l)} + \mathbf{V}\mathbf{x}_t\right). \tag{4}$$

LISTA performs better than ISTA, and with a much smaller number of iterations (layers).

B. Deep Unfolding RNNs

The deep unfolding RNNs proposed in [30], [31], [32] use the reconstruction of the previous signal in time $\mathbf{s}_{t-1}$, or its sparse representation $\mathbf{h}_{t-1}$, to aid in recovery of $\mathbf{s}_t$. They start from the following optimization problem:

$$\min_{\mathbf{h}_t} E(\mathbf{x}_t, \mathbf{h}_t) + \lambda_2 C\left(\mathbf{h}_t, \mathbf{h}_{t-1}\right), \quad \forall t, \tag{5}$$

where E is defined in (2), C defines the correlation between the two time steps and λ_2 is another regularization parameter. (5) is solved consecutively for each time step t, using the reconstruction from the previous step, which results in a RNN architecture when unfolded. Depending on C, one obtains different optimization algorithms, which unfold into the different deep unfolding RNN architectures. When $C\left(\mathbf{h}_t, \mathbf{h}_{t-1}\right) = \frac{1}{2}\|\mathbf{D}\mathbf{h}_t - \mathbf{F}\mathbf{D}\mathbf{h}_{t-1}\|_2^2$, where the matrix $\mathbf{F}$ models the correlation between consecutive signals, the problem can be solved using a sequential version of ISTA, referred to as sequential ISTA (SISTA), or its unfolded version SISTA-RNN [30]. On the other hand, when $C\left(\mathbf{h}_t, \mathbf{h}_{t-1}\right) = \|\mathbf{h}_t - \mathbf{G}\mathbf{h}_{t-1}\|_1$, where the matrix $\mathbf{G}$ models the temporal correlation, the reconstruction becomes an ℓ_1-ℓ_1 optimization problem, which unfolds into the ℓ_1-ℓ_1-RNN [31]. Additional weighting parameters can be inserted into this ℓ_1-ℓ_1 optimization problem, which unfolds into the reweighted-RNN [32].

C. Transformers and Deep Unfolding

In recent years, Transformer models have achieved state-of-the-art results in language modeling [16], [34], [35], computer vision [36], [37], [38], and image processing tasks [40], [41] (see [39] for an overview in vision). Their key idea is splitting the input data into independent tokens and processing them in parallel to model long-range relationships. This is in contrast to RNNs, which process data sequentially. In Transformers, the input tokens can be word embeddings for language modeling or image patches for image or video processing. In this work (and also in [33]), only the encoder part of the Transformer is considered. Specifically, we focus on obtaining the core Transformer encoder operations in our unfolded network, namely, 1) the self-attention module, which lets tokens extract context from each other, and 2) the nonlinear transform that is applied to each token in parallel, generally implemented by a one- or two-layer fully connected network (FCN). Whereas a Transformer block is also interleaved with normalization layers and residual connections, the core operation of a vanilla Transformer encoder block is described by:

$$\mathbf{Z}^{(k+1)} = \mathbf{Y}^{(k)}\,\text{softmax}\left(\mathbf{Y}^{(k)T}\mathbf{W}_K^{(k)T}\mathbf{W}_Q^{(k)}\mathbf{Y}^{(k)}\right), \tag{6}$$

$$\mathbf{y}_n^{(k+1)} = \text{FCN}\left(\mathbf{z}_n^{(k+1)}\right) \quad \forall n, \tag{7}$$

where the tokens $\mathbf{y}_1^{(k)}, \ldots, \mathbf{y}_N^{(k)}$ are concatenated into the matrix $\mathbf{Y}^{(k)}$, transformed into $\mathbf{Z}^{(k)}$ by the self-attention module with learnable matrices $\mathbf{W}_K^{(k)}$ and $\mathbf{W}_Q^{(k)}$, and then the columns $\mathbf{z}_n^{(k)}$ are further processed into the output tokens of the block $\mathbf{y}_1^{(k+1)}, \ldots, \mathbf{y}_N^{(k+1)}$. The softmax function transforms the values $u_1, \ldots, u_N$ of each column in a matrix to a probability distribution of N possible outcomes with probability $\frac{\exp(u_n)}{\sum_{j=1}^N \exp(u_j)}$. Many of these blocks are then stacked on top of each other to form a full Transformer encoder model.

The authors of [33] designed the first deep unfolding based Transformer, which consists of a softmax self-attention operation and a nonlinear transform as in (6)-(7), by starting from the following optimization problem:

$$\min_{\mathbf{Y}}\left(\sum_{i,j} - \exp\left(-\frac{1}{2}\|\mathbf{W}_a\mathbf{y}_i - \mathbf{W}_a\mathbf{y}_j\|_2^2\right) + \frac{1}{2}\|\mathbf{W}_a\mathbf{Y}\|_{\mathcal{F}}^2\right) + \left(\frac{1}{2}\,\text{Tr}\left(\mathbf{Y}^T\mathbf{W}_b\mathbf{Y}\right) + \frac{1}{2}\|\mathbf{Y}\|_{\mathcal{F}}^2 + \varphi(\mathbf{Y})\right), \tag{8}$$

where $\mathbf{Y}$ is a matrix with the vectors $\mathbf{y}_1, \ldots, \mathbf{y}_N$ as columns, $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm, $\text{Tr}(\cdot)$ is the trace of a matrix, $\varphi(u)$

is the indicator function, whose value is $+\infty$ for $u < 0$ and 0 otherwise, and $\mathbf{W}_a$ and $\mathbf{W}_b$ are arbitrary weight matrices. They proved that this total energy can be minimized by iterations of an algorithm alternating between a step that decreases the value of the first part of (8) and a proximal gradient descent step on the second part of (8). When unfolded, this first part will result in a weighted softmax self-attention layer, while the second part unfolds into a linear projection with ReLU activation as follows:

$$\mathbf{Z}^{(k+1)} = \mathbf{Y}^{(k)} \operatorname{softmax}_\beta \left(\mathbf{Y}^{(k)T} \mathbf{W}_a^{(k)T} \mathbf{W}_a^{(k)} \mathbf{Y}^{(k)} \right), \quad (9)$$

$$\mathbf{y}_n^{(k+1)} = \operatorname{ReLU} \left(\mathbf{W}_b^{(k)} \mathbf{z}_n^{(k+1)} \right) \quad \forall n, \quad (10)$$

where $\mathbf{W}_a^{(k)}$ and $\mathbf{W}_b^{(k)}$ are learnable matrices, $\operatorname{ReLU}(u) = \max(0, u)$ and $\operatorname{softmax}_\beta$ is a weighted softmax function with probabilities $\frac{\beta_n \exp(u_n)}{\sum_{j=1}^N \beta_j \exp(u_j)}$, where $\beta_i = \exp\left(-\frac{1}{2}\|\mathbf{y}_i\|_2^2\right)$. We can see the correspondence between (6)-(7) and (9)-(10). The sequence of (9) and (10) together form one block of the unfolded Transformer, which can be stacked into a network of L such blocks.

It was shown in [33] that the network architecture is able to decrease the Transformer energy in (8) throughout its layers, demonstrating that the conditions for convergence of the model likely hold in many practical settings. However, no experiments were conducted in which the unfolded Transformer is trained and tested on any regression or classification task. In fact, we show in Section IV-D that without our modification to (9), the network is not able to train well at all on the sparse recovery task. Additionally, the model in [33] does not incorporate the priors specific to sparse recovery that are embedded in, for example, the deep unfolding RNNs [30], [31], [32], allowing them to obtain a significantly better reconstruction than the generic unfolded Transformer of [33]. Therefore, in this work we explore unfolded Transformer architectures specific to sparse recovery.

III. Deep Unfolding Sparse Transformer (DUST)

In this section we propose our optimization problem for the sparse recovery of sequential signals, the derivation of the corresponding optimization algorithm, and the unfolding of the algorithm into a deep unfolding Transformer network, coined DUST. Additionally, we describe extensions to the base model that incorporate token normalization and multi-head attention in order to improve the reconstruction performance.

A. DUST Optimization Problem

In the deep unfolding RNNs discussed in Section II-B, only the correlation between pairs of consecutive signals is considered. When modeling a video with a static background, for example, measurements from *all* frames are useful to recover the background and could improve reconstruction quality compared to looking at only two frames at a time. The same can be argued for videos with recurring events or temporary occlusions, where similar signals in the video are not necessarily adjacent. The technique of using similar image patches to aid the reconstruction process has been used before in traditional optimization algorithms [44], [45] and deep learning models [46]. In this work, we take all signals in a sequence, e.g., all image patches from the same spatial location in a video clip, and want to extract information from the most similar patches through an attention mechanism.

In order to find similar signals in the sequence and reconstruct them simultaneously, we aim to extend the term $\frac{1}{2}\|\mathbf{D}\mathbf{h}_t - \mathbf{F}\mathbf{D}\mathbf{h}_{t-1}\|_2^2$ in SISTA [30] to model correlations over the whole sequence, in the form of $\sum_{t,\tau} \frac{1}{2}\|\mathbf{F}\mathbf{D}\mathbf{h}_t - \mathbf{F}\mathbf{D}\mathbf{h}_\tau\|_2^2$. The indices t and τ span the whole signal sequence, from time step 1 to T, such that we compare each possible pair of signals in the sequence. Initial experiments showed that using $\mathbf{F}$ does not improve performance and therefore we set it to the identity matrix. This is further elaborated on in Appendix B. In order to diminish the influence of uncorrelated signals, we introduce an exponential function: $\sum_{t,\tau} - \exp\left(-\frac{1}{2}\|\mathbf{D}\mathbf{h}_t - \mathbf{D}\mathbf{h}_\tau\|_2^2\right)$. When performing gradient descent on this expression, the gradient will be near zero for dissimilar signals while we improve the estimation of correlated signals. We also add a constraint on the ℓ_2 norm of each signal, i.e., $\sum_t \frac{1}{2}\|\mathbf{D}\mathbf{h}_t\|_2^2$, to obtain the following temporal correlation function:

$$C_{\mathbf{D}}(\mathbf{h}_1, \ldots, \mathbf{h}_T) = \sum_t \sum_\tau - \exp\left(-\frac{1}{2}\|\mathbf{D}\mathbf{h}_t - \mathbf{D}\mathbf{h}_\tau\|_2^2\right) + \sum_t \frac{1}{2}\|\mathbf{D}\mathbf{h}_t\|_2^2, \quad (11)$$

where $\mathbf{D}$ is the dictionary matrix as in (2). Given a sensing matrix $\mathbf{A}$, the dictionary $\mathbf{D}$, and regularization parameters λ_1 and λ_2, we put together this correlation function, and the reconstruction error and sparsity constraint in (2), to obtain our full optimization problem:

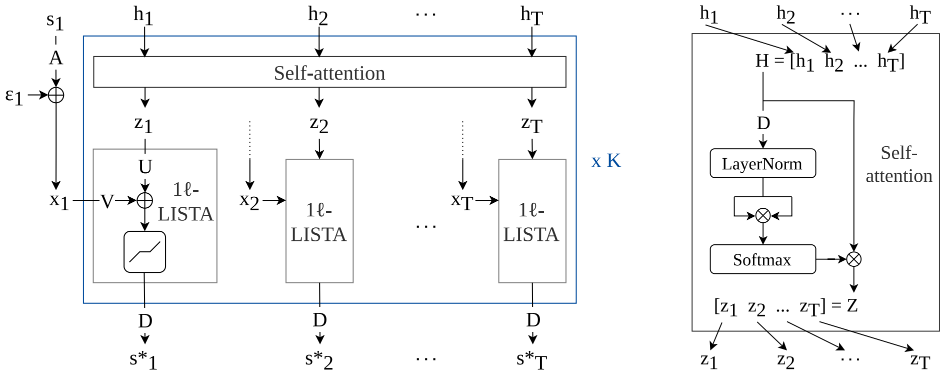
$$\min_{\mathbf{h}_1, \ldots, \mathbf{h}_T} \sum_t E(\mathbf{x}_t, \mathbf{h}_t) + \lambda_2 C_{\mathbf{D}}(\mathbf{h}_1, \ldots, \mathbf{h}_T). \quad (12)$$

We observe a similarity between (11) and the first part of (8), while the second part of (8) is replaced by our data fidelity terms and sparsity constraints in $\sum_t E(\mathbf{x}_t, \mathbf{h}_t)$ [see (2)]. More specifically, the term $\frac{1}{2}\operatorname{Tr}(\mathbf{Y}^T \mathbf{W}_B \mathbf{Y}) + \frac{1}{2}\|\mathbf{Y}\|_{\mathcal{F}}^2$ in (8) that is designed to unfold into a linear projection, is replaced by $\sum_t \|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t\|_2^2$, and the indicator function $\varphi(\mathbf{Y})$ in (8) is replaced with our sparsity constraints $\lambda_1 \sum_t \|\mathbf{h}_t\|_1$. The differences between (8) and (12) make our optimization problem tailored to the task of sparse recovery for sequential signals.

We derive an optimization algorithm for solving (12), analogous to the algorithm the authors of [33] derived for (8). We can design minimization steps for $C_{\mathbf{D}}(\mathbf{h}_1, \ldots, \mathbf{h}_T)$ and $\sum_t E(\mathbf{x}_t, \mathbf{h}_t)$ separately, and alternate between the two steps to obtain the full optimization algorithm. An elaborate description of this derivation is given in Appendix A. The update that minimizes $C_{\mathbf{D}}(\mathbf{h}_1, \ldots, \mathbf{h}_T)$ is given by

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} \operatorname{softmax}_\beta \left(\mathbf{H}^{(k)T} \mathbf{D}^T \mathbf{D} \mathbf{H}^{(k)} \right), \quad (13)$$

with $\mathbf{H}^{(k)} \in \mathbb{R}^{d \times T}$ a matrix with $\mathbf{h}_1^{(k)}, \ldots, \mathbf{h}_T^{(k)}$ as its columns, and where $\operatorname{softmax}_\beta$ applies a weighted softmax function to each of the columns in a matrix, with probabilities $\frac{\beta_t \exp(u_t)}{\sum_s \beta_s \exp(u_s)}$ and $\beta_t = \exp\left(-\frac{1}{2}\|\mathbf{D}\mathbf{h}_t\|_2^2\right)$. Note that if each $\mathbf{D}\mathbf{h}_t$ has the same



(a) The DUST network

(b) The self-attention used in DUST

Fig. 1. The proposed deep unfolding transformer (DUST) encoder for sequential sparse recovery. The inputs to the network are the measurements $\mathrm{x}_t = \mathrm{A}\mathrm{s}_t + \epsilon_t$. The sparse representations h_t are initialized with zeros and passed through K network layers. The sparse outputs of the encoder are then multiplied with the dictionary D to obtain the reconstructed signals s_t. The $\oplus$ symbol signifies elementwise addition, while $\otimes$ stands for matrix multiplication.

Algorithm 1 Attention-based sequential sparse recovery algorithm

```
Require: measurement matrix $\mathbf{A}$, dictionary $\mathbf{D}$, regularization parameters $c$, $\lambda_1$ and $\lambda_2$, measurements $\mathbf{x}_t = \mathbf{A}\mathbf{s}_t$, with $t = 1, \dots, T$
1: $\mathbf{h}_1^{(0)}, \dots, \mathbf{h}_T^{(0)} \leftarrow \mathbf{0}$
2: for $k = 1$ to $K$ do
3:     $\beta_t^{(k)} \leftarrow \exp\left(-\frac{1}{2}\|\mathbf{D}\mathbf{h}_t^{(k)}\|_2^2\right) \quad \forall t$
4:     for $t = 1$ to $T$ do
5:         $\mathbf{z}_t^{(k)} \leftarrow \lambda_2 \frac{\sum_u \beta_u^{(k)} \exp\left(\mathbf{h}_t^{(k-1)T}\mathbf{D}^T\mathbf{D}\mathbf{h}_u^{(k-1)}\right)\mathbf{h}_u^{(k-1)}}{\sum_u \beta_u^{(k)} \exp\left(\mathbf{h}_t^{(k-1)T}\mathbf{D}^T\mathbf{D}\mathbf{h}_u^{(k-1)}\right)}$
6:         $\mathbf{h}_t^{(k)} \leftarrow \phi_{\frac{\lambda_1}{c}}\left(\mathbf{z}_t + \frac{1}{c}\mathbf{D}^T\mathbf{A}^T\left(\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{z}_t^{(k)}\right)\right)$
7:     end for
8: end for
9: return $\mathbf{s}_1^\star = \mathbf{D}\mathbf{h}_1^{(K)}, \dots, \mathbf{s}_T^\star = \mathbf{D}\mathbf{h}_T^{(K)}$
```

ℓ_2 norm, the weighted softmax simplifies to an unweighted one. Since each $E(\mathbf{x}_t, \mathbf{h}_t)$ is independent from each other, the corresponding update step for this part consists of T parallel ISTA iterations (3). The full optimization algorithm then consists of alternating steps of softmax self-attention (13) and parallel ISTA operations (3) and is shown in Algorithm 1. Since initially all $\mathbf{h}_t, t = 1, \dots, T$, are identical (we initialize them to zero), the first iteration of the algorithm simplifies to an iteration of ISTA. After $K - 1$ more iterations of alternating self-attention and single-iteration ISTA steps, the sparse representations $\mathbf{h}_t^{(K)}$ are multiplied with the dictionary $\mathbf{D}$ to obtain the final reconstructed signals $\mathbf{s}_t^\star = \mathbf{D}\mathbf{h}_t^{(K)}$.

B. Proposed DUST Model

By unrolling the steps of Algorithm 1 we obtain the proposed Deep Unfolding Sparse Transformer model (DUST). The sensing matrix $\mathbf{A}$ becomes a learnable parameter of the model, which is used to obtain the compressed measurements $\mathbf{x}_t$. Then the model, shown in Fig. 1, takes as input a set of initial tokens $\mathbf{h}_t$ and the measurements $\mathbf{x}_t$, applies self-attention (line 5 in Algorithm 1) using the learned dictionary $\mathbf{D}$, followed by one layer of LISTA [20] (which we refer to as 1ℓ-LISTA). 1ℓ-LISTA reduces the calculations in line 6 of Algorithm 1 to (15), using the learnable matrices $\mathbf{U}$ and $\mathbf{V}$.

The main processing block of DUST then has the following form:

$$\mathbf{Z}^{(k+1)} = \lambda_2 \mathbf{H}^{(k)} \operatorname{softmax}_\beta\left(\mathbf{H}^{(k)T}\mathbf{D}^T\mathbf{D}\mathbf{H}^{(k)}\right), \tag{14}$$

$$\mathbf{h}_t^{(k+1)} = \phi_{\lambda_1/c}\left(\mathbf{U}\mathbf{z}_t^{(k+1)} + \mathbf{V}\mathbf{x}_t\right) \text{ for } t = 1, \dots, T. \tag{15}$$

The tokens are put through such a block of self-attention (14) and 1ℓ-LISTA (15) K times, followed by a final linear projection using the learned dictionary $\mathbf{D}$ to obtain the reconstructed signals.

Analogous to the comparison of our optimization problem with [33] in the previous section, the softmax self-attention (14) is identical up to the scaling factor λ_2 to (9), while the linear projection followed by ReLU in (10) is replaced by a layer of LISTA (15) in DUST.

The learnable sensing matrix $\mathbf{A}$ is randomly initialized. The learned dictionary $\mathbf{D}$ is initialized with the discrete cosine transform, analogous to the deep unfolding RNNs [30], [31], [32], since natural video frames are generally sparse in the frequency domain. For other applications, you could choose another initialization appropriate to the task at hand, or use a random initialization. Given the initial values for $\mathbf{A}$ and $\mathbf{D}$, indicated as $\mathbf{A}_i$ and $\mathbf{D}_i$, we can initialize $\mathbf{U}$ and $\mathbf{V}$ in (15) with the values used in ISTA as $\mathbf{U} = \mathbf{I} - \frac{1}{c}\mathbf{D}_i^T\mathbf{A}_i^T\mathbf{A}_i\mathbf{D}_i$ and $\mathbf{V} = \frac{1}{c}\mathbf{D}_i^T\mathbf{A}_i^T$. $\mathbf{U}$ and $\mathbf{V}$ are then trained independently of $\mathbf{A}$ and $\mathbf{D}$.

The learnable parameters of DUST are the matrices $\mathbf{A}$, $\mathbf{D}$, $\mathbf{U}$ and $\mathbf{V}$, and the parameters λ_1, λ_2 and c. These parameters are tied between all blocks, similar to the deep unfolding RNNs. The model is trained by minimizing the mean squared error loss $\frac{1}{JT}\sum_{j,t}\|\mathrm{s}_{j,t} - \mathrm{s}_{j,t}^\star\|_2^2$ between the original and reconstructed time-series signals, where J is the number of training samples.

The main difference between the unfolded Transformer [33] and DUST is the ability to interpret our network as the operation of a sparse recovery algorithm, with as most prominent architectural difference the incorporation of a LISTA layer (15) in the network instead of (10). The input to the network is for our model not the compressed measurements, but the initial values of the sparse representations, while the measurements are taken in as side information through the LISTA layer. This LISTA layer, with the soft thresholding activation function, promotes the sparsity of the generated signal representations. Additionally, DUST puts these sparse representations through a final linear projection, in order to obtain the reconstructed signals.

C. Token Normalization

The weighted softmax operation in (13), derived from the energy function (11), differs from the plain softmax operator used in the vanilla Transformer. The attention weight for the contribution of token $\mathbf{h}_s$ to token $\mathbf{h}_t$ is given by:

$$\frac{\beta_s \exp\left(\mathbf{h}_t^T \mathbf{D}^T \mathbf{D} \mathbf{h}_s\right)}{\sum_i \beta_i \exp\left(\mathbf{h}_t^T \mathbf{D}^T \mathbf{D} \mathbf{h}_i\right)}, \tag{16}$$

where the weights β_i are given by $\beta_i = \exp\left(-\frac{1}{2}\|\mathbf{D}\mathbf{h}_i\|_2^2\right)$. Note that the weights cancel out when each β_i has the same value, or equivalently, when each vector $\mathbf{D}\mathbf{h}_i$ has the same norm. Therefore, we tried to normalize the vectors $\mathbf{q}_t = \mathbf{D}\mathbf{h}_t$ to zero mean and unit variance before calculating the attention map using the unweighted softmax, which turned out to significantly improve the reconstruction performance of the model.

This normalization is exactly the same as in the Layer Normalization used in vanilla Transformers, albeit without the learnable affine transform that often follows it. Differently from the vanilla Transformer, the normalization is mixed into the self-attention operation and applied on the query tokens $\mathbf{D}\mathbf{h}_t$, instead of the sparse representations $\mathbf{h}_t$ themselves in a separate LayerNorm layer before the self-attention. In either case, the normalization helps to stabilize the training of the DUST network and improves its final performance. Since the unfolded Transformer in [33] uses the same weighted softmax operation, it can also be applied there. The performance improvement for the unfolded Transformer by applying token normalization is even larger than for DUST, as is shown in Section IV-D.

D. Multi-Head Attention

So far we have derived a self-attention operation with a single attention head. In order to allow the model to attend to different pieces of information, resulting from different signal representations, we use multiple dictionary projections or attention heads which perform attention operations in parallel. With this in mind, instead of only using one temporal correlation function $C_{\mathbf{D}}$ in (12), we take the average of M functions $C_{\mathbf{D}_m}$, each with a different learnable dictionary:

$$\min_{\mathbf{h}_1,\ldots,\mathbf{h}_T} \sum_t E(\mathbf{x}_t, \mathbf{h}_t) + \frac{\lambda_2}{M} \sum_m C_{\mathbf{D}_m}(\mathbf{h}_1, \ldots, \mathbf{h}_T). \tag{17}$$

The minimization step for $\sum_t E(\mathbf{x}_t, \mathbf{h}_t)$ remains the same [see (3)], whereas the averaging of functions $C_{\mathbf{D}_m}$ translates into the averaging of self-attention operations:

$$\mathbf{h}_t^{(k+1)} = \frac{1}{M} \sum_m \mathbf{H}^{(k)} \operatorname{softmax}_\beta\left(\mathbf{H}^{(k)T} \mathbf{D}_m^T \mathbf{D}_m \mathbf{H}^{(k)}\right). \tag{18}$$

The unfolding of (17) is the same as for DUST, with the difference that we now obtain a multi-head self-attention layer $\lambda_2 \sum_m \mathbf{H}^{(k)} \operatorname{softmax}\left(\mathbf{H}^{(k)T} \mathbf{D}_m^T \mathbf{D}_m \mathbf{H}^{(k)}\right)$ instead of (14).

IV. Experimental Results

In this section, we test the performance of the basic DUST model and the multi-head version MH-DUST. They are compared with deep unfolding RNNs [30], [31], [32], as well as a generic Vision Transformer (ViT) [47] and the deep unfolding Transformer from [33]. We conduct experiments on video reconstruction from compressed measurements, as well as video denoising tasks.

A. Datasets and Preprocessing

In our experiments we used the following three real-world video datasets:

- CUHK Avenue [48] consists of 16 training videos and 21 test videos with a resolution of 640 × 360 pixels, containing people walking in front of a station entrance.
- UCSD Anomaly Detection [49] has 50 train and 48 test videos of pedestrians and the occasional vehicle. Videos from one scene have a resolution of 238 × 158 pixels, while the second scene has a 360 × 240 pixels resolution.
- ShanghaiTech Campus [50] contains 330 train and 107 test videos of resolution 856 × 480 pixels of pedestrians and an occasional bicycle or scooter. It covers 13 different scenes with a variety of view angles.

For the Avenue and ShanghaiTech dataset, we convert the videos to grayscale and downsample the frames to a vertical resolution of 160 pixels. The UCSD dataset is already in grayscale and has a lower resolution, so we do not downsample these videos. Furthermore, the training videos of each dataset are split into 80% for the training set and 20% for the validation set, while the test sets are used as is. In order to reduce the computational time needed to conduct experiments, the length of each video is limited to a maximum of 200 frames.

In order to reconstruct the videos, they are split into overlapping or non-overlapping patches of 16 × 16 pixels. These patch videos are then further split into non-overlapping clips of 20 frames each. The last processing steps differ slightly depending on the reconstruction task at hand.

1) Compressed Sensing: In this setting, each 16 × 16 patch is flattened into a 256 × 1 vector and multiplied with the learnable $m \times 256$ sensing matrix $\mathbf{A}$ of the model, without adding extra noise. Each patch clip sample then becomes a sequence of 20 vectors of size m, or equivalently a $m \times 20$ matrix.

2) Denoising: A special case of compressed sensing is when the sensing matrix in (1) is the identity. The recovery of the signals then becomes a denoising task. For this task, we simply flatten the patches and add Gaussian noise with the specified noise level. This noise level is the standard deviation of the noise that is added to pixel values in the range [0, 255]. Each patch clip input sample is then a 256×20 matrix.

3) Noisy Compressed Sensing: This scenario is a combination of the two previous operations. The flattened patches are multiplied with the sensing matrix $\mathbf{A}$ and are then corrupted by adding i.i.d. Gaussian noise with standard deviation σ.

B. Experimental Settings

All models are implemented using PyTorch [51]. Each network is trained to minimize the mean square error (MSE) between the output of the network and the original signal sequence, using the Adam optimizer. We use gradient clipping during backpropagation in order to stabilize training.

For each model tested in this work we use the same number of layers/blocks, that is, 3. Each block in a model shares the same parameters, apart from the ViT, which has separate weights in each Transformer block. The number of parameters of each network is listed in Table VIII.

In each model, the $m \times 256$ sensing matrix $\mathbf{A}$ is randomly initialized using Glorot initialization, where the value $m < 256$ depends on the chosen compressed sensing rate $\frac{m}{256}$ (CS rate). For the video denoising task, however, the sensing matrix is fixed to the square identity matrix. The overcomplete 256×1024 dictionary $\mathbf{D}$ used in the deep unfolding models is initialized with the DCT transform. For the multi-head version of DUST, each dictionary $\mathbf{D}_1 \ldots, \mathbf{D}_M$ is initialized with the DCT, but we make each one of them slightly different by Gaussian noise with a standard deviation of 3×10^{-4}, to encourage the model to learn different dictionaries. Furthermore, for DUST and MH-DUST the starting values of c, λ_1, and λ_2 are 1, 0.1, and 0.4 respectively, while we use the values 1, 0.3, and 0.02 for SISTA-RNN, ℓ_1-ℓ_1-RNN, and reweighted-RNN. Other parameters are initialized according to their corresponding papers.

All models are trained using a batch size of 64 patch clips. The deep unfolding RNNs use a learning rate of 10^{-4}, while ViT, the unfolded Transformer, DUST and MH-DUST use a learning rate of 3×10^{-4} in most cases. For the compressed sensing task, DUST and MH-DUST use a learning rate of 10^{-3} instead. This learning rate is reduced with a factor 0.3 each times the validation loss does not decrease for 5 consecutive epochs. On the Avenue and UCSD dataset the models are trained for 100 epochs, while on ShanghaiTech we train for 40 epochs due to the larger size of the dataset.

The reconstruction performance is measured using the peak signal-to-noise ratio (PSNR) between the original and reconstructed frames, after stitching the patches back together. The PSNR is then averaged over all frames in the test set. We use the same operation to calculate the structural similarity index measure (SSIM).

C. Sensitivity to Initial Parameter Values

The initial values for c, λ_1, and λ_2 were determined empirically. In fact, the exact starting values for these parameters do not have a significant impact on the final performance of the model, since the optimal values are also learned during training. In Table I, we provide the average PSNR values for compressed sensing on the Avenue dataset when varying each of these parameters. As can be seen in the table, varying the starting value for each of these parameters has little effect on the reconstruction quality after training.

TABLE I
EFFECT OF INITIAL PARAMETER VALUES ON FINAL MODEL PERFORMANCE

λ_1	0.01	0.03	0.1	0.3	1	3	10
PSNR	37.54	37.61	37.61	37.67	37.65	37.66	37.59
λ_2	0.05	0.1	0.2	0.4	0.6	0.8	1
PSNR	37.64	37.56	37.55	37.61	37.60	37.50	37.35
c	0.1	0.5	1	1.5	2	3	4
PSNR	37.45	37.52	37.61	37.61	37.58	37.56	37.48

D. Token Normalization

Prior to calculating the attention map in DUST, we normalize the tokens to get rid of the reweighing in the softmax function, as described in Section III-C. Here, we show that this normalization has a significant positive effect on the performance of the model. The tests are performed for the compressed sensing task on the Avenue dataset, with CS rate 0.2 and non-overlapping patches.

We observed that the normalization helps to stabilize the training process, especially in the beginning, which helps the model reach a better final performance. In Table II(a) that contains the results for DUST, in the row *w/o norm.* the attention tokens were not normalized and the weighted softmax function is used as in (13), while in the row *w/ norm.*, we did apply the normalization. On the Avenue and UCSD dataset, we gain over 1 dB in PSNR and more than 1.2% in SSIM by using the token normalization over the weighed softmax. On the ShanghaiTech dataset, the model is not even able to converge without normalization.

TABLE II
THE EFFECT OF TOKEN NORMALIZATION ON COMPRESSED SENSING PERFORMANCE FOR THE AVENUE DATASET

(a) Proposed DUST

	Avenue		UCSD		ShanghaiTech	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
w/o norm.	35.24	0.9526	33.51	.9565	*	*
w/ norm.	**36.30**	**0.9653**	**34.53**	**0.9688**	**35.53**	**0.9407**

(b) Unfolded Transformer [33]

	Avenue		UCSD		ShanghaiTech	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
w/o norm.	10.59	0.0624	8.86	0.0116	*	*
w/ norm.	**33.64**	**0.9289**	**32.54**	**0.941**	**33.31**	**0.9145**

* Training diverged

In the same way, we make the comparison for the unfolded Transformer [33] in Table II(b). We test the difference between

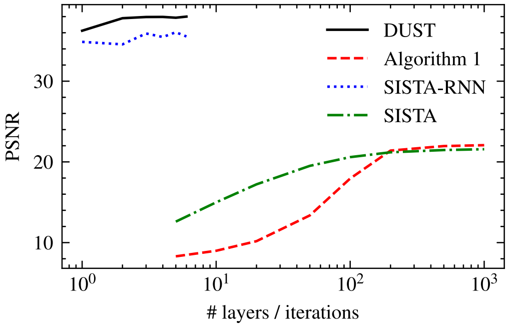


Fig. 2. Compressed sensing performance on the avenue dataset for SISTA, Algorithm 1, and the corresponding deep unfolding networks.

using the weighed softmax, and normalizing the attention tokens before computing the attention matrix and applying plain softmax. Without normalization, this architecture performs very poorly on the Avenue and UCSD dataset with 11.59 and 8.86 dB PSNR, respectively. Like DUST, the architecture also diverges on ShanghaiTech. The token normalization allows the unfolded Transformer to perform decently on all three datasets.

Given these results, we use token normalization in both DUST and the unfolded Transformer in all further experiments.

E. Optimization Algorithms vs Deep Unfolding

The relation between Algorithm 1 and the unfolding into DUST is illustrated in Fig. 2. We show the reconstruction performance for compressed sensing at a rate 0.2 on the Avenue test set, for different numbers of model layers or algorithm iterations. For DUST, we trained networks from 1 to 6 layers, while Algorithm 1 was tested for 5, 10, 20, 50, 100, 200, and 500 iterations. The algorithm needs at least an order of magnitude more iterations to converge compared to DUST, while DUST still achieves PSNR that is at least 16 dB higher. For comparison, we also include results for SISTA and SISTA-RNN in the figure, with the same numbers of layers and iterations. While Algorithm 1 only gains 0.5 dB over SISTA, this difference is amplified when training the corresponding deep unfolding networks, where DUST achieves an increase of more than 2 dB in PSNR over SISTA-RNN.

F. Multi-Head Attention

Next, we want to find the optimal number of attention heads for the multi-head DUST model MH-DUST. We compare the base DUST architecture and MH-DUST with different numbers of heads on Avenue for compressed sensing with CS rate 0.2, as well as denoising with noise level 20. The results on both tasks are shown in Table III. The single-head case corresponds with the basic DUST model. For the compressed sensing task, multi-head attention brings a slight advantage of up to 0.1 dB in PSNR on the Avenue and ShanghaiTech dataset, while on UCSD we see no apparent benefit. On the denoising task, we see improvements of 0.1, 0.2, and 0.3 dB in PSNR using multi-head attention on respectively the UCSD, ShanghaiTech, and Avenue datasets.

TABLE III
RECONSTRUCTION PERFORMANCE FOR DIFFERENT NUMBERS OF ATTENTION HEADS ON THE AVENUE DATASET

(a) Compressed Sensing

	Avenue		UCSD		ShanghaiTech	
# Heads	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
1	36.44	0.9665	**34.56**	**0.9690**	35.51	0.9404
2	36.46	0.9666	**34.56**	0.9689	35.43	0.9396
4	36.48	0.9670	34.51	0.9686	**35.61**	**0.9415**
6	**36.52**	**0.9672**	34.53	0.9688	35.47	0.9402
8	36.48	0.9669	34.52	0.9687	35.44	0.9391

(b) Denoising

	Avenue		UCSD		ShanghaiTech	
# Heads	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
1	34.90	0.9478	34.54	0.9569	35.71	0.9363
2	35.19	**0.9507**	34.64	0.9585	**35.89**	**0.9369**
4	35.02	0.9489	**34.66**	**0.9590**	35.39	0.9295
6	**35.21**	0.9506	**34.66**	0.9589	35.44	0.9306
8	35.03	0.9493	**34.66**	0.9588	35.46	0.9307

Since the multi-head attention is not clearly outperforming the single-head model, in further experiments we evaluate both single-head and multi-head DUST with 4 attention heads. We can conclude that multi-head attention mostly shines when the problems become harder, for instance, denoising at higher noise levels or noisy compressed sensing, where the added modeling complexity does improve the reconstruction significantly. The choice between DUST and MH-DUST thus depends on the specifics of the reconstruction problem.

G. Performance Comparison

In this section, we compare our DUST and MH-DUST models with the deep unfolding RNNs SISTA-RNN, ℓ_1-ℓ_1-RNN and reweighted-RNN, as well as the unfolded Transformer [33] and a generic ViT model on the three real-world video dataset CUHK Avenue, UCSD Anomaly Detection and ShanghaiTech campus. We perform experiments for compressed sensing at different CS rates, video denoising for different noise levels, and CS with additive Gaussian noise.

1) Compressed Sensing: For the compressed sensing experiments, we extract overlapping patches from the videos into clips of 20 consecutive patches. The compressed sensing rate $\frac{m}{256}$ ranges from 0.1 to 0.5, and the average PSNR and SSIM values for each model in each setting are reported in Table IV. On all three datasets, reweighted-RNN had the tendency to suddenly diverge during training. Even after trying out different values for the learning rate and other hyperparameters, we could not get reweighted-RNN to train properly in some settings, indicated in the table.

In general, the main contenders in these experiments are the reweighted-RNN and the single- and multi-head DUST models. SISTA-RNN and ℓ_1-ℓ_1-RNN perform consistently worse than reweighted-RNN, and ViT and the unfolded Transformer are in almost all cases several dB in PSNR and several percent

TABLE IV
COMPRESSED SENSING

(a) Avenue test results

	SISTA-RNN		ℓ_1-ℓ_1-RNN		Reweighted-RNN		ViT		Unfolded Transf.		DUST		MH-DUST	
CS Rate	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.1	32.43	0.9104	33.65	0.9409	*	*	34.30	0.9530	32.25	0.9054	**35.00**	**0.9550**	34.97	0.9548
0.2	35.91	0.9591	36.82	0.9697	37.28	0.9762	36.71	0.9732	34.35	0.9405	37.95	0.9766	**38.02**	**0.9770**
0.3	38.08	0.9754	39.03	0.9817	39.53	0.9845	36.41	0.9727	36.09	0.9605	**40.08**	**0.9854**	39.84	0.9845
0.4	40.18	0.9849	40.96	0.9882	**41.87**	**0.9906**	39.20	0.9854	38.07	0.9750	41.81	0.9900	41.81	0.9899
0.5	42.02	0.9901	42.93	0.9924	**43.88**	**0.9939**	38.05	0.9875	39.76	0.9828	43.57	0.9931	43.61	0.9932

(b) UCSD test results

	SISTA-RNN		ℓ_1-ℓ_1-RNN		Reweighted-RNN		ViT		Unfolded Transf.		DUST		MH-DUST	
CS Rate	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.1	30.84	0.9139	31.31	0.9363	*	*	**33.29**	**0.9595**	30.99	0.9153	32.71	0.9537	32.83	0.9551
0.2	34.20	0.9589	34.40	0.9672	*	*	24.00	0.6377	33.10	0.9469	35.97	0.9764	**36.05**	**0.9769**
0.3	36.74	0.9757	37.09	0.9809	37.78	0.9847	26.03	0.7435	34.55	0.9544	38.52	0.9856	**38.53**	**0.9858**
0.4	39.17	0.9855	39.59	0.9880	40.15	0.9902	25.79	0.7309	36.55	0.9702	40.82	0.9908	**40.89**	**0.9909**
0.5	41.47	0.9913	42.10	0.9927	42.32	0.9934	28.14	0.8586	38.64	0.9808	**42.99**	**0.9940**	42.98	**0.9940**

(c) ShanghaiTech test results

	SISTA-RNN		ℓ_1-ℓ_1-RNN		Reweighted-RNN		ViT		Unfolded Transf.		DUST		MH-DUST	
CS rate	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.1	31.48	0.8844	32.36	0.9005	*	*	**33.18**	**0.9076**	31.66	0.8865	33.08	0.9060	33.07	0.9058
0.2	34.82	0.9397	35.44	0.9473	*	*	35.85	0.9486	34.08	0.9344	**35.94**	**0.9497**	35.86	0.9486
0.3	37.06	0.9631	37.75	0.9675	**37.97***	**0.9699***	37.75	0.9673	36.03	0.9586	37.90	0.9677	37.91	0.9675
0.4	39.16	0.9770	39.75	0.9797	**40.30***	**0.9812***	38.70	0.9769	37.63	0.9721	40.04	0.9801	40.01	0.9801
0.5	41.22	0.9858	41.71	**0.9871**	*	*	41.18	0.9862	39.56	0.9818	41.78	0.9870	**41.80**	0.9870

* Training diverged; if a result is given, this was obtained after considerable hyperparameter tuning.

in SSIM below reweighted-RNN and the DUST models. The reason ViT and the unfolded Transformer do not perform well in these experiments is because they do not incorporate the priors specific to this recovery problem that are embedded into the architecture of the deep unfolding RNNs and DUST. Overall, our DUST models consistenly improve over reweighted-RNN by 0.6-0.7 dB in PSNR as well as in SSIM. The comparison between basic DUST and the multi-head version MH-DUST is rather inconclusive, with differences mostly below plus or minus 0.1 dB in PSNR.

2) Video Denoising: In this setting, we fix the sensing matrix to the identity in all models and add Gaussian noise to the frame patches. We again test all models on the three datasets Avenue, UCSD, and ShanghaiTech, using noise with a standard deviation from 10 to 100. The results are shown in Table V. Similar to the compressed sensing experiments, SISTA-RNN, ℓ_1-ℓ_1-RNN, ViT, and the unfolded Transformer perform well below reweighted-RNN and DUST. Again, our DUST models improve over reweighted-RNN with a significant margin. In this case, however, our multi-head attention is able to consistently be better than single-head DUST.

3) Noisy Compressed Sensing: For completeness, we also consider the combination of the previous two scenarios, adding Gaussian noise to compressed measurements. We generate measurements from non-overlapping patches at a compression ratio of 0.2 and add noise with a standard deviation ranging from 10 to 100. The average PSNR and SSIM are reported in Table VI for the Avenue dataset. At low noise levels, MH-DUST and DUST are on par with reweighted-RNN, while at higher noise levels, DUST improves over reweighted-RNN with 0.8-2 dB. MH-DUST adds another 0.5 to 0.8 dB on top of this. Like before, SISTA-RNN, ℓ_1-ℓ_1-RNN, ViT and the unfolded Transformer cannot match reweighted-RNN and (MH-)DUST.

In conclusion, the Vision Transformer and unfolded Transformer cannot compete with the problem specific architectures, as they lack the specific priors related to these recovery problems. Our DUST method consistently improves over reweighted-RNN and the other RNNs, while its multi-head variant has an edge in more difficult settings like noisy compressed sensing.

A visual example of the performance of each model is given in Fig. 3. It shows the difference in absolute value between the reference frame and the reconstruction from a noisy video with noise level 20 of each network, where yellow indicates the highest errors and purple shows the lowest errors. For the RNNs, reweighted-RNN is visually the best model, which corroborates the global PSNR and SSIM results. ViT suffers from severe errors in some patches, which may indicate that ViT mostly memorizes the training data and is unable to generalize to unseen data, like the person walking in the foreground or

TABLE V
VIDEO DENOISING

(a) Avenue test results

	SISTA-RNN		ℓ_1-ℓ_1-RNN		Reweighted-RNN		ViT		Unfolded Transf.		DUST		MH-DUST	
σ	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
10	34.83	0.9385	36.95	0.9641	37.66	0.9695	27.72	0.9186	35.19	0.9446	38.02	0.9698	**38.17**	**0.9709**
20	31.14	0.8739	33.83	0.9351	34.61	0.9457	27.22	0.8912	33.06	0.9122	34.90	0.9478	**35.02**	**0.9489**
50	26.68	0.7345	29.86	0.8660	30.71	0.8856	23.80	0.7413	30.00	0.8442	31.44	0.8977	**31.62**	**0.9024**
100	23.96	0.6223	26.66	0.7640	27.49	0.7910	20.40	0.6574	28.16	0.7992	28.28	0.8065	**28.46**	**0.8116**

(b) UCSD test results

	SISTA-RNN		ℓ_1-ℓ_1-RNN		Reweighted-RNN		ViT		Unfolded Transf.		DUST		MH-DUST	
σ	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
10	35.03	0.9452	36.87	0.9668	*	*	30.48	0.9380	35.27	0.9474	37.63	0.9722	**37.82**	**0.9733**
20	31.53	0.8955	33.28	0.9430	34.26	0.9569	28.28	0.9075	32.29	0.9091	34.54	0.9569	**34.66**	**0.9590**
50	26.85	0.7397	29.25	0.8786	30.26	0.9086	24.63	0.8328	29.03	0.8562	30.47	0.9130	**30.53**	**0.9150**
100	24.14	0.6241	26.40	0.7853	27.32	0.8291	20.72	0.7312	27.21	0.8315	27.39	0.8326	**27.42**	**0.8375**

(c) ShanghaiTech test results

	SISTA-RNN		ℓ_1-ℓ_1-RNN		Reweighted-RNN		ViT		Unfolded Transf.		DUST		MH-DUST	
σ	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
10	35.26	0.9186	37.12	0.9466	37.44	0.9502	22.67	0.8028	36.54	0.9430	38.24	0.9581	**39.34**	**0.9667**
20	32.12	0.8637	33.58	0.9001	34.10*	0.9086*	21.38	0.7444	34.47	0.9053	**35.71**	**0.9363**	35.39	0.9295
50	28.18	0.7533	30.04	0.8241	*	*	19.84	0.7008	30.68	0.8311	31.53	0.8650	**31.94**	**0.8743**
100	25.53	0.6552	27.58	0.7487	27.92	0.7646	18.39	0.6470	28.44	0.7799	28.89	0.7998	**29.16**	**0.8067**

* Training diverged; if a result is given, this was obtained after considerable hyperparameter tuning.

TABLE VI
NOISY COMPRESSED SENSING ON AVENUE

	SISTA-RNN		ℓ_1-ℓ_1-RNN		Reweighted-RNN		ViT		Unfolded Transf.		DUST		MH-DUST	
σ	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
10	33.34	0.9222	35.02	0.9518	**35.82**	0.9607	29.93	0.9035	33.08	0.9208	35.74	**0.9607**	35.63	0.9601
20	32.32	0.9015	34.02	0.9370	35.02	0.9520	30.05	0.8930	32.92	0.9165	35.04	0.9543	**35.06**	**0.9546**
50	29.16	0.8159	30.80	0.8771	32.31	0.9148	29.91	0.8789	32.20	0.9054	33.16	0.9322	**33.93**	**0.9437**
100	26.58	0.7218	28.29	0.8058	30.10	0.8675	29.05	0.8574	31.51	0.8938	32.11	0.9135	**32.60**	**0.9256**

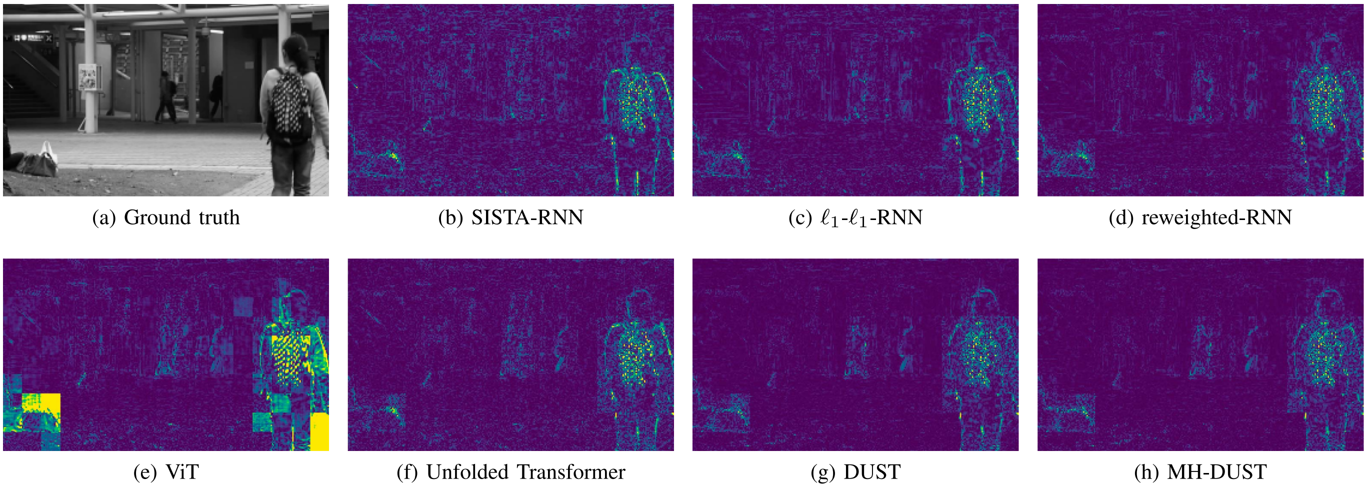
the bags in the bottom left corner. DUST and MH-DUST yield the best results visually, which supports the quantitative results in Table V.

H. Longer Sequences

In this section, we evaluate the models on longer sequences, since we expect the Transformer based models to be better able to handle these compared to the RNNs. The first experiment uses the models trained for compressed sensing on Avenue in Section IV-G, using a CS rate of 0.2 and clips of 20 frames long, and evaluates them on longer video sequences; namely, patch clips of 25, 50, and 100 frames. The average PSNR and SSIM for each model and clip length is shown in Table VII(a). The unfolded Transformer, DUST and MH-DUST achieve almost exactly the same performance, irrespective of the length of the video clips. Among the Transformer based architectures, only ViT sees a drop in performance of 0.8 dB and 2.1 dB in PSNR for sequences of length 50 and 100. For the deep unfolding RNNs however, at a clip length of 100 frames, SISTA-RNN sees a drop of 1.2 dB in PSNR, ℓ_1-ℓ_1-RNN loses more than 10 dB, and the output of reweighted-RNN even diverges, which results in numerical overflows and errors.

The reason for this difference between the RNNs and Transformers is the way they process a signal sequence. Since the RNNs reconstruct the sequence frame by frame, using the previous reconstruction as a guide, the errors accumulate over time. On the other hand, the self-attention in the Transformers just processes more tokens at a time and no error accumulation occurs. The other parts in the Transformer architecture even operate on each token independently, so the only part where the sequence length makes any difference is in the self-attention layer.

This error accumulation is illustrated in Fig. 4 for the compressed sensing task, which shows the PSNR for each frame, averaged over all video clips extracted from the Avenue test set. When evaluated on both clip lengths of 20 and 50 frames, the average frame PSNR remains fairly constant over the whole


(a) Ground truth (b) SISTA-RNN (c) ℓ_1-ℓ_1-RNN (d) reweighted-RNN

(e) ViT (f) Unfolded Transformer (g) DUST (h) MH-DUST

Fig. 3. Visualization of the absolute error on an example frame for video denoising on the avenue dataset.

TABLE VII
Compressed Sensing for Longer Video Clips on the Avenue Dataset

(a) Evaluation of original models on longer clips

	SISTA-RNN		ℓ_1-ℓ_1-RNN		Reweighted-RNN		ViT		Unfolded Transf.		DUST		MH-DUST	
Clip Length	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
20	35.91	0.9591	36.82	0.9697	37.28	0.9762	36.71	0.9732	34.35	0.9405	37.95	0.9766	**38.02**	**0.9770**
25	35.95	0.9593	36.83	0.9702	36.86	0.9758	36.69	0.9732	34.33	0.9404	37.94	0.9765	**38.02**	**0.9770**
50	35.62	0.9596	34.06	0.9676	36.21	0.9745	35.91	0.9731	34.34	0.9404	37.96	0.9766	**38.06**	**0.9772**
100	34.70	0.9591	26.17	0.9536	†	†	34.61	0.9725	34.33	0.9403	37.99	0.9768	**38.09**	**0.9774**

(b) Model performance after finetuning on the corresponding clip lengths

	SISTA-RNN		ℓ_1-ℓ_1-RNN		Reweighted-RNN		ViT		Unfolded Transf.		DUST		MH-DUST	
Clip Length	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
25	35.98	0.9598	36.89	0.9704	36.89	0.9759	36.90	0.9737	34.38	0.9410	37.96	0.9767	**38.04**	**0.9772**
50	35.60	0.9600	35.59	0.9701	36.29	0.9746	36.80	0.9736	34.36	0.9408	37.98	0.9768	**38.06**	**0.9772**
100	34.76*	0.9595*	*	*	*	*	36.78	0.9736	34.37	0.9408	37.99	0.9768	**38.08**	**0.9774**

† Model output diverged * Training diverged

clip for all Transformer based models, showing that the architecture is robust to changes in input length. On the other hand, the RNNs process the video clips sequentially, which means that they cannot use the information in the subsequent frames and forget information over time. For the first few frames, the information the RNNs can use is limited, which results in low PSNR values for these frames. In later frames, the reconstruction errors accumulate and the PSNR drops over time. This dropping reconstruction quality is especially prominent when evaluating on clips of length 50, where all RNNs show significant drops in PSNR towards the end of the clip, which explains their poor performance on long videos. Specifically for reweighted-RNN, we also observed throughout the experiments that during the reconstruction of a patch clip, sometimes the norm of the vector representation explodes. This divergence also caused several failed training runs as reported in previous sections. The presence of these failed reconstructions might also explain the steeper decline in PSNR in Fig. 4 for reweighted-RNN at clip length 20.

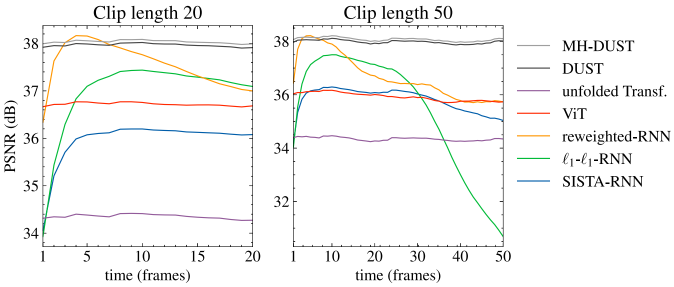


Fig. 4. Compressed sensing reconstruction quality per frame for each network, averaged over each video clip in the avenue test set.

Apart from evaluating the models on longer clips, we also finetune the models for these settings. We divide the initial learning rate by 10 compared to training from scratch, and finetune each model for 20 epochs on the Avenue dataset, with CS rate 0.2 and the specified clip length. The results are listed

TABLE VIII
SIZE AND COMPUTATIONAL COSTS OF EACH MODEL FOR COMPRESSED SENSING ON THE AVENUE DATASET

	# Parameters (Millions)	Training Time (s/epoch)	Inference Time (s)
SISTA-RNN	0.34	30	42
ℓ_1-ℓ_1-RNN	1.32	55	64
reweighted-RNN	2.37	65	72
ViT	2.46	31	21
unfolded Transf.	0.49	10	20
DUST	1.38	6	20
MH-DUST	2.42	7	21

in Table VII(b). For most models, the effects of finetuning are rather limited, with differences only in the last digit for the PSNR and SSIM values. Only for ViT and ℓ_1-ℓ_1-RNN, we see some significant improvements. With finetuning, ViT now has about the same performance for each clip length, while ℓ_1-ℓ_1-RNN gains 1.5 dB in PSNR for 50 frame clips, which is still 1.2 dB below the base performance on the clips of length 20.

Overall, finetuning does not make a difference for the deep unfolding Transformers, since they already perform equally well on a wide range of sequence lengths. The RNN models on the other hand, see large drops in reconstruction quality when the input sequences become longer, even when finetuned on these specific settings. We would like to mention that while finetuning the RNN can help for that specific clip length, but as a result loses performance on the original 20 frame clips, which does not happen for the Transformer models.

I. Model Complexity

Furthermore, we compare the number of parameters, training time, and inference speed for each of the models in Table VIII. We train each model for 100 epochs for compressed sensing on the Avenue dataset with a CS rate of 0.2 on an NVIDIA GeForce RTX 3090 GPU. We list the average training time in seconds per epoch, as well as the total time to run the model on the test set, to showcase the inference speed.

DUST has significantly less parameters than reweighted-RNN, showing that the improvements in performance do not come from increased model complexity, but rather an architectural advantage. The Transformer models are up to 10 times faster to train than the RNNs, and their processing speed is 2-3 times higher at inference. This is due to the more parallel nature of the computations in Transformers, as well as more simple activation functions, e.g., soft thresholding or ReLU compared to the double plateau thresholding function in ℓ_1-ℓ_1-RNN and reweighted-RNN. Additionally, DUST has nearly half the number of parameters compared to ViT and is 5 times faster to train, due to its single linear layer compared to the two-layer feedforward block in ViT. The unfolded Transformer has less parameters than DUST, but is somewhat slower to train. Nevertheless, both ViT and the unfolded Transformer perform significantly worse than DUST as shown in previous sections, due to the priors that are embedded into the DUST architecture.

TABLE IX
VIDEO DENOISING PERFORMANCE (PSNR) ON THE AVENUE DATASET

	$\sigma = 10$	$\sigma = 20$	$\sigma = 50$
pretrained FastDVDnet	32.39	25.72	18.29
FastDVDnet	**38.52**	**35.37**	**31.71**
pretrained PaCNet	36.43	30.20	23.74
pretrained RVRT	31.69	25.65	18.08
RVRT	28.15	22.26	15.01
DUST	38.02	34.90	31.44
MH-DUST	38.17	35.02	31.62

J. Comparison With SOTA Video Denoising Models

Additionally, we compare the video denoising performance of our approach with other video denoising methods, namely, FastDVDnet [52], PaCNet [46], and RVRT [53]. Since these models were implemented for RGB videos instead of the grayscale videos as we evaluate in this paper, we adapted their code to use the same noise values for each color channel and then fed them the same grayscale videos as for DUST for a fair comparison. Each of these works provide weights pretrained on the DAVIS dataset [54]. We evaluate these models on the Avenue test set. Furthermore, we train FastDVDnet and RVRT from scratch on Avenue. For PaCNet this was not possible as they did not provide training details or training code.

The denoising performance in PSNR for each model is shown in Table IX. Only FastDVDnet trained on Avenue is able to best our MH-DUST network by 0.1-0.4 dB. Depending on the noise level, the pretrained models all lose at least 2 and up to 13 dB compared to DUST. RVRT, in particular, was not able to train at all on the Avenue dataset and performs even worse than pretrained RVRT. On top of that, on our NVIDIA GeForce RTX 3090 GPU, DUST can be trained in under 12 minutes and performs inference at 730 frames/s. On the other hand, FastDVDnet requires 32 hours of training on the same computer, and can run inference at 69 frames/s. RVRT requires more than 8 days of training and runs at 23 frames/s. Finally, PaCNet is only able to achieve an inference speed of 0.09 frames/s, more than 8000 times slower than DUST. Furthermore, DUST and MH-DUST have respectively 1.38 and 2.42 million parameters. FastDVDnet and PaCNet have 2.48 and 2.87 million parameters, on par with MH-DUST and double the size of single-head DUST, while RVRT contains 12.8 million parameters, an order of magnitude more than DUST. In conclusion, our methods yield competitive denoising performance at a fraction of the computational cost compared to state-of-the-art video restoration methods.

V. CONCLUSION

In this work, we design an optimization algorithm for the recovery of sequential signals, taking into account the sparsity of the signals in a dictionary and their correlation over time. The derived optimization problem is unfolded into a Transformer based neural network architecture. We experiment with different modifications to the base model, such as the normalization of attention tokens and the use of multi-head attention. Extensive experiments on video reconstruction from low-dimensional and/or noisy measurements show that our deep unfolding Transformer networks outperform state-of-the-art deep unfolding RNNs, generic Transformer networks, and state-of-the-art video restoration methods in reconstruction performance, while having less model parameters. In addition, our proposed networks require significantly less computational resources to train and perform inference than other approaches.

Appendix A
Derivation of the DUST Optimization Algorithm

We elaborate on the derivation of our optimization algorithm from (12). Starting with the temporal correlation function (11), it was shown in Theorem 3.1 from [33] that the energy function

$$\sum_t \sum_\tau -\exp\left(\frac{1}{2}\|\mathbf{h}_t - \mathbf{h}_\tau\|_2^2\right) + \sum_t \frac{1}{2}\|\mathbf{h}_t\|_2^2 \tag{19}$$

can be minimized with the update rule

$$\begin{aligned}\mathbf{h}_t^{(k+1)} &= \frac{\sum_\tau \beta_\tau^{(k)} \exp\left(\mathbf{h}_t^{(k)T}\mathbf{h}_\tau^{(k)}\right)\mathbf{h}_t^{(k)}}{\sum_\tau \beta_\tau^{(k)} \exp\left(\mathbf{h}_t^{(k)T}\mathbf{h}_\tau^{(k)}\right)}\\ &= \mathbf{H}^{(k+1)}\,\mathrm{softmax}_\beta\left(\mathbf{H}^{(k)T}\mathbf{H}^{(k)}\right),\end{aligned} \tag{20}$$

with $\beta_\tau^{(k)} = \exp\left(-\frac{1}{2}\|\mathbf{h}_t^{(k)}\|_2^2\right)$ and $\mathbf{H}^{(k)}$ a concatenation of the vectors $\mathbf{h}_1^{(k)}, \ldots, \mathbf{h}_T^{(k)}$. Furthermore, they introduced learnable parameters into this update step using the reparameterization $\mathbf{y}_t = \mathbf{D}\mathbf{h}_t$, such that we obtain the update step (13) for (11).

In order to derive an update rule for

$$\min_{\mathbf{h}_1,\ldots,\mathbf{h}_T} \sum_t \frac{1}{2}\|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t\|_2^2 + \lambda_1\|\mathbf{h}_t\|_1, \tag{21}$$

we observe that the terms are independent for each t. Therefore, each of these terms can be minimized independently with iterations of ISTA (3) as in Section II-A.

In order to minimize the sum of (11) and (21), we can use the proximal alternating inexact minimization algorithm (P-AIM) from [33]. They proved that an energy function that is a sum of multiple terms, can be minimized by applying (proximal) gradient descent steps for each part separately in succession. The resulting algorithm, where one iteration consists of executing the (proximal) gradient descent update corresponding to each part consecutively, is shown to converge to a small region around the optimal solution. Since our optimization problem (12) fits within the scope of P-AIM, we can apply this method to derive our Algorithm 1, where one iteration consist of the descent step for (11), a gradient descent on $\sum_t \frac{1}{2}\|\mathbf{x}_t - \mathbf{A}\mathbf{D}\mathbf{h}_t\|_2^2$, and the soft thresholding proximal operator corresponding to $\sum_t \|\mathbf{h}_t\|_1$.

Appendix B
Effect of the F Matrix

The authors of [30] assume each signal in a sequence is linearly predictable through a matrix $\mathbf{F}$: $\mathbf{s}_t = \mathbf{F}\mathbf{s}_{t-1} + \mathbf{n}_{t-1}$, where $\mathbf{n}$ is zero mean Gaussian noise. This results in the correlation term $C(\mathbf{h}_t, \mathbf{h}_{t-1}) = \frac{1}{2}\|\mathbf{D}\mathbf{h}_t - \mathbf{F}\mathbf{D}\mathbf{h}_{t-1}\|_2^2$ in (5). When you extend this approach to multiple time steps, you obtain higher powers of $\mathbf{F}$. To facilitate the self-attention operation, we have to opt for the symmetric constraint $\frac{1}{2}\|\mathbf{F}\mathbf{D}\mathbf{h}_t - \mathbf{F}\mathbf{D}\mathbf{h}_{t-1}\|_2^2$. However, this is equivalent to projection using a different dictionary $\mathbf{D}' = \mathbf{F}\mathbf{D}$. Since the dictionary is learned anyway, introducing an extra matrix $\mathbf{F}$ should not make any difference. In Table X, we compare the reconstruction performance of single-head and multi-head DUST on the Avenue dataset for the different reconstruction tasks, with and without the $\mathbf{F}$ matrix. The results mostly differ in the least significant digit and lie within the natural variability between training runs.

TABLE X
Effect of the F Matrix on Reconstruction Performance

(a) DUST

	Avenue		UCSD		Shanghaitech	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
CS						
w/o F	37.97	0.9766	36.04	0.9767	35.86	0.9486
w/ F	37.91	0.9762	36.05	0.9768	35.84	0.9489
Denoising						
w/o F	34.90	0.9478	35.54	0.9569	35.71	0.9363
w/ F	35.14	0.9505	34.58	0.9575	35.59	0.934
Noisy CS						
w/o F	35.09	0.9547	33.92	0.963	34.86	0.9332
w/ F	35.15	0.9554	33.96	0.9634	34.90	0.9334

(b) MH-DUST

	Avenue		UCSD		Shanghaitech	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
CS						
w/o F	37.94	0.9765	36.02	0.9769	35.88	0.9489
w/ F	37.89	0.9763	35.99	0.9767	35.84	0.9483
Denoising						
w/o F	35.02	0.9489	34.66	0.959	35.39	0.9295
w/ F	35.20	0.9508	34.66	0.9591	35.54	0.9318
Noisy CS						
w/o F	35.12	0.9553	33.95	0.9634	34.89	0.9337
w/ F	35.13	0.9556	33.98	0.9637	34.84	0.9329

Appendix C
Visualization of the Learned Matrices

In Fig. 5, we show the initial values of the sensing matrix $\mathbf{A}$ and the dictionary $\mathbf{D}$, as well as their values after training DUST for compressed sensing on the Avenue dataset. In Fig. 6, we do the same for the matrix $\mathbf{U}$, which is initialized using the formula: $\mathbf{U} = \mathbf{I} - \frac{1}{c}\mathbf{D}^T\mathbf{A}^T\mathbf{A}\mathbf{D}$. Additionally, we recalculate $\mathbf{U}$ using the learned values of $\mathbf{A}$, $\mathbf{D}$, and c, which is very different from the plain learned $\mathbf{U}$. The same conclusion can be reached for the matrix $\mathbf{V}$. As in LISTA, the independent $\mathbf{U}$ and $\mathbf{V}$ matrices add extra flexibility to the model and help to increase the reconstruction performance.

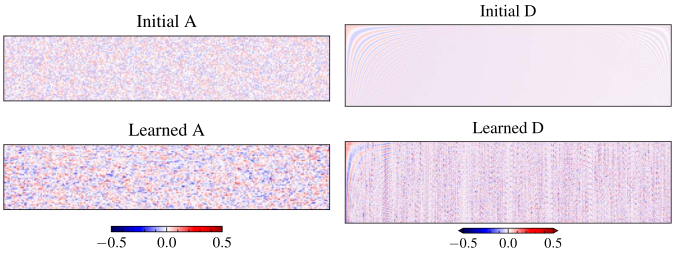


Fig. 5. Visualization of the initial and learned sensing matrix $\mathbf{A}$ and dictionary $\mathbf{D}$.

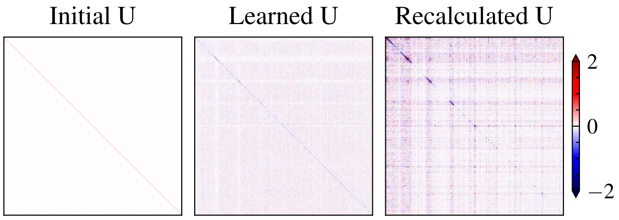


Fig. 6. The difference between the initialization of the matrix $\mathbf{U}$, the learned values, and the recalculation from the learned parameters $\mathbf{A}$, $\mathbf{D}$, and c.

References

[1] L. Weizman, Y. C. Eldar, and D. Ben Bashat, “Compressed sensing for longitudinal MRI: An adaptive-weighted approach,” *Med. Phys.*, vol. 42, no. 9, pp. 5195–5208, 2015.

[2] X. Lu, B. Xu, T. S. Yeo, W. Su, and H. Gu, “Co-located MIMO radar target detection in cluttered and noisy environment based on 2D block sparse recovery,” *IEEE Trans. Signal Process.*, vol. 69, pp. 3431–3445, 2021.

[3] R. G. Baraniuk, T. Goldstein, A. C. Sankaranarayanan, C. Studer, A. Veeraraghavan, and M. B. Wakin, “Compressive video sensing: Algorithms, architectures, and applications,” *IEEE Signal Process. Mag.*, vol. 34, no. 1, pp. 52–66, Jan. 2017.

[4] L. Wang, Z. Xiong, H. Huang, G. Shi, F. Wu, and W. Zeng, “High-speed hyperspectral video acquisition by combining Nyquist and compressive sampling,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 857–870, Apr. 2019.

[5] D. M. Malioutov, S. R. Sanghavi, and A. S. Willsky, “Sequential compressed Sensing,” *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 2, pp. 435–444, Apr. 2010.

[6] N. Vaswani and W. Lu, “Modified-CS: Modifying compressive sensing for problems with partially known support,” *IEEE Trans. Signal Process.*, vol. 58, no. 9, pp. 4595–4607, Sep. 2010.

[7] A. Charles, M. S. Asif, J. Romberg, and C. Rozell, “Sparsity penalties in dynamical system estimation,” in *Proc. 45th Annu. Conf. Inf. Sci. Syst.*, 2011, pp. 1–6.

[8] J. F. C. Mota, N. Deligiannis, A. C. Sankaranarayanan, V. Cevher, and M. R. D. Rodrigues, “Adaptive-rate reconstruction of time-varying signals with application in compressive foreground extraction,” *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3651–3666, Jul. 2016.

[9] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, “Recon-Net: Non-iterative reconstruction of images from compressively sensed measurements,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 449–458.

[10] W. Shi, F. Jiang, S. Liu, and D. Zhao, “Image compressed sensing using convolutional neural network,” *IEEE Trans. Image Process.*, vol. 29, pp. 375–388, 2020.

[11] Y. Sun, J. Chen, Q. Liu, B. Liu, and G. Guo, “Dual-path attention network for compressed sensing image reconstruction,” *IEEE Trans. Image Process.*, vol. 29, pp. 9482–9495, 2020.

[12] D. Ye, Z. Ni, H. Wang, J. Zhang, S. Wang, and S. Kwong, “CSformer: Bridging convolution and transformer for compressive sensing,” *IEEE Trans. Image Process.*, vol. 32, pp. 2827–2842, 2023.

[13] K. Xu and F. Ren, “CSVideoNet: A real-time end-to-end learning framework for high-frame-rate video compressive sensing,” in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, Mar. 2018, pp. 1680–1688.

[14] A. L. Mur, B. Montcel, F. Peyrin, and N. Ducros, “Deep neural networks for single-pixel compressive video reconstruction,” in *Proc. SPIE Unconventional Opt. Imag. II*, vol. 11351, Apr. 2020, pp. 71–80.

[15] L. Wang, M. Cao, Y. Zhong, and X. Yuan, “Spatial-temporal transformer for video snapshot compressive imaging,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 7, pp. 9072–9089, Jul. 2023.

[16] A. Vaswani et al., “Attention is all you need,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, p. 6000–6010.

[17] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos, “Using deep neural networks for inverse problems in imaging: beyond analytical methods,” *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 20–36, Jan. 2018.

[18] N. Shlezinger, Y. C. Eldar, and S. P. Boyd, “Model-based deep learning: On the intersection of deep learning and optimization,” *IEEE Access*, vol. 10, pp. 115384–115398, 2022.

[19] V. Monga, Y. Li, and Y. C. Eldar, “Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing,” *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021.

[20] K. Gregor and Y. LeCun, “Learning fast approximations of sparse coding,” in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 399–406.

[21] I. Daubechies, M. Defrise, and C. De Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.

[22] Y. Yang, J. Sun, H. Li, and Z. Xu, “Deep ADMM-Net for compressive sensing MRI,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 10–18.

[23] M. Borgerding, P. Schniter, and S. Rangan, “AMP-inspired deep networks for sparse linear inverse problems,” *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4293–4308, Aug. 2017.

[24] J. Zhang and B. Ghanem, “ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1828–1837.

[25] D. You, J. Xie, and J. Zhang, “ISTA-NET++: Flexible deep unfolding network for compressive sensing,” in *Proc. IEEE Int. Conf. Multimedia Expo*, 2021, pp. 1–6.

[26] Z. Zhang, Y. Liu, J. Liu, F. Wen, and C. Zhu, “AMP-Net: Denoising-based deep unfolding for compressive image sensing,” *IEEE Trans. Image Process.*, vol. 30, pp. 1487–1500, 2021.

[27] V. Kouni, G. Paraskevopoulos, H. Rauhut, and G. C. Alexandropoulos, “ADMM-DAD Net: A deep unfolding network for analysis compressed sensing,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 1506–1510.

[28] D. Gilton, G. Ongie, and R. Willett, “Neumann networks for linear inverse problems in imaging,” *IEEE Trans. Comput. Imag.*, vol. 6, pp. 328–343, 2020.

[29] M. Shen, H. Gan, C. Ning, Y. Hua, and T. Zhang, “TransCS: A transformer-based hybrid architecture for image compressed sensing,” *IEEE Trans. Image Process.*, vol. 31, pp. 6991–7005, 2022.

[30] S. Wisdom, T. Powers, J. Pitton, and L. Atlas, “Building recurrent networks by unfolding iterative thresholding for sequential sparse recovery,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 4346–4350.

[31] H. D. Le, H. Van Luong, and N. Deligiannis, “Designing recurrent neural networks by unfolding an l1-l1 minimization algorithm,” in *Proc. IEEE Int. Conf. Image Process.*, 2019, pp. 2329–2333.

[32] H. V. Luong, B. Joukovsky, and N. Deligiannis, “Designing interpretable recurrent neural networks for video reconstruction via deep unfolding,” *IEEE Trans. Image Process.*, vol. 30, pp. 4099–4113, 2021.

[33] Y. Yang, Z. Huang, and D. P. Wipf, “Transformers from an optimization perspective,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 36958–36971.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” 2019, *arXiv:1810.04805*.

[35] T. Brown et al., “Language models are few-shot learners,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.

[36] H. Fan et al., “Multiscale vision transformers,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 6824–6835.

[37] Z. Liu et al., “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 10012–10022.

[38] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis.*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., New York, NY, USA: Springer-Verlag, 2020, pp. 213–229.

[39] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Comput. Surv.*, vol. 54, no. 10s, 2022.

[40] F. Yang, H. Yang, J. Fu, H. Lu, and B. Guo, "Learning texture transformer network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 5790–5799.

[41] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "SwinIR: Image restoration using Swin transformer," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 1833–1844.

[42] B. De Weerdt, Y. C. Eldar, and N. Deligiannis, "Designing transformer networks for sparse recovery of sequential data using deep unfolding," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2023, pp. 1–5.

[43] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[44] J. Zhang, D. Zhao, and W. Gao, "Group-based sparse representation for image restoration," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3336–3351, Aug. 2014.

[45] J. Xu, Y. Qiao, Z. Fu, and Q. Wen, "Image block compressive sensing reconstruction via group-based sparse representation and nonlocal total variation," *Circuits, Syst., Signal Process.*, vol. 38, no. 1, pp. 304–328, 2019.

[46] G. Vaksman, M. Elad, and P. Milanfar, "Patch craft: Video denoising by deep modeling and patch matching," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 2157–2166.

[47] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2022.

[48] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 FPS in MATLAB," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2720–2727.

[49] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1975–1981.

[50] W. Luo, W. Liu, and S. Gao, "A revisit of sparse coding based anomaly detection in stacked RNN framework," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 341–349.

[51] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[52] M. Tassano, J. Delon, and T. Veit, "FastDVDnet: Towards real-time deep video denoising without flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1354–1363.

[53] J. Liang et al., "Recurrent video restoration transformer with guided deformable attention," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022.

[54] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 DAVIS challenge on video object segmentation," 2018, *arXiv:1704.00675*.

Brent De Weerdt (Student Member, IEEE) received the master's degree in electrical engineering from the Vrije Universiteit Brussel and the Université Libre de Bruxelles, Belgium, in 2021. He is currently working toward the Ph.D. degree with the Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel. His research interests include image and video processing, interpretable deep learning, and deep unfolding. In 2022, he received the Ph.D. Fellowship Strategic Basic Research from the Flanders Research Foundation (FWO), Belgium.

Yonina C. Eldar (Fellow, IEEE) received the B.Sc. degree in physics, in 1995, and the B.Sc. degree in electrical engineering, in 1996, both from Tel-Aviv University (TAU), Tel-Aviv, Israel, and the Ph.D. degree in electrical engineering and computer science, in 2002, from Massachusetts Institute of Technology (MIT), Cambridge. She is currently a Professor with the Department of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel, where she holds the Dorothy and Patrick Gorman Professorial Chair and heads the Center for Biomedical Engineering. She was previously a Professor with the Department of Electrical Engineering, Technion, where she held the Edwards Chair in Engineering. She is also a Visiting Professor with MIT, a Visiting Scientist with the Broad Institute, a Visiting Research Collaborator with Princeton, an Adjunct Professor with Duke University, an Advisory Professor with Fudan University, a Distinguished Visiting Professor with Tsinghua University, and was a Visiting Professor with Stanford. Her research interests include statistical signal processing, sampling theory and compressed sensing, learning and optimization methods, and their applications to biology, medical imaging and optics. She has received many awards for excellence in research and teaching, including the IEEE Signal Processing Society Technical Achievement Award (2013), the IEEE/AESS Fred Nathanson Memorial Radar Award (2014), and the IEEE Kiyo Tomiyasu Award (2016). She was a Horev Fellow of the Leaders in Science and Technology program at the Technion and an Alon Fellow. She received the Michael Bruno Memorial Award from the Rothschild Foundation, the Weizmann Prize for Exact Sciences, the Wolf Foundation Krill Prize for Excellence in Scientific Research, the Henry Taub Prize for Excellence in Research (twice), the Hershel Rich Innovation Award (three times), the Award for Women with Distinguished Contributions, the Andre and Bella Meyer Lectureship, the Career Development Chair at the Technion, the Muriel & David Jacknow Award for Excellence in Teaching, and the Technion's Award for Excellence in Teaching (two times). She received several best paper awards and best demo awards together with her research students and colleagues including the SIAM outstanding Paper Prize, the UFFC Outstanding Paper Award, the Signal Processing Society Best Paper Award and the IET Circuits, Devices and Systems Premium Award, was selected as one of the 50 most influential women in Israel and in Asia, and is a highly cited researcher. She was a member of the Young Israel Academy of Science and Humanities and the Israel Committee for Higher Education. She is the Editor in Chief of Foundations and Trends in Signal Processing, a member of the IEEE Sensor Array and Multichannel Technical Committee and serves on several other IEEE committees. In the past, she was a Signal Processing Society Distinguished Lecturer, member of the IEEE Signal Processing Theory and Methods and Bio Imaging Signal Processing technical committees, and served as an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING, *EURASIP Journal of Signal Processing*, *SIAM Journal on Matrix Analysis and Applications*, and *SIAM Journal on Imaging Sciences*. She was Co-Chair and Technical Co-Chair of several international conferences and workshops. She is an Author of the book titled "*Sampling Theory: Beyond Bandlimited Systems*" and a Co-Author of seven other books. She is a member of the Israel Academy of Sciences and Humanities (elected 2017) and of the Academia Europaea (elected 2023), a EURASIP Fellow, a Fellow of the Asia-Pacific Artificial Intelligence Association, and a Fellow of the 8400 Health Network.

Nikos Deligiannis (Member, IEEE) received the Diploma degree in electrical and computer engineering from the University of Patras, Patras, Greece, in 2006, and the Ph.D. degree (Hons.) in engineering sciences from the Vrije Universiteit Brussel (VUB), Brussels, Belgium, in 2012. From 2013 to 2015, he was a Senior Researcher with the Department of Electronic and Electrical Engineering, University College London, London, U.K. He is currently an Associate Professor with the Department of Electronics and Informatics (ETRO), VUB, and a Principal Investigator with the imec, Leuven, Belgium. His research interests include interpretable and explainable machine learning, image and video processing, computer vision, and distributed deep learning. He is a member of the EURASIP and served as the Chair of the EURASIP Technical Area Committee on Signal and Data Analytics for Machine Learning, from 2021 to 2023. He serves as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING.