CorKit: A Deep Learning Framework for Advanced LASCO Image Calibration and Restoration

Jorge Enciso[1]

[1]First Affiliation

Key Points:

- Image Reconstruction
- Coronagraph calibration
- Deep Learning

Corresponding author: Jorge Enciso, `jorged.encyso@gmail.com`

Abstract

1 Introduction

2 L-Multilayered UNet-like Partial Convolution Network

2.1 Partial Convolutions

Great efforts have been made to create models that coherently reconstructs missing data chunks of any nature. One of the best approaches is Partial Convolutional Neural Networks. They are traditional convolutional layers that incorporate a scaling factor to the mapped output from the kernel. Also they introduce mask updates for each forward pass reducing the missing area step-wise:

$$O = W^T(X \odot M)\frac{sum(1)}{sum(M)} + b \tag{1}$$

This work proposes a L multi-layered UNet-like neural network, each layer specializing on a masked piece of the ground truth image (I_{gt}) defined by the prior mask (M_{l-1}) and the updated mask or the time step (M_l).

2.2 Loss function

For the purpose of training the model, this work adheres to the loss function presented by ... changing some model-specific features for this particular architecture. Pixel-wise, perceptual, style and physics informed terms will be employed to construct an adequate constraint where the global minima is easier to find. We first define all loss functions as scalar fields from θ linear space to the real numbers. In this case, we are representing the loss scalar fields with $\mathcal{L}$ and the functionals with $\mathcal{F}$:

$$\mathcal{L} : \theta \to R \tag{2}$$

As well as the traditional lagrangian functional:

$$\mathcal{F} : W^{1,1}(R^n) \to R \tag{3}$$

For this task, we are implementing a layer wise loss function. The first term is the by **pixel loss** composed by an inner and difference terms:

$$\mathcal{L}_{pixel}(\alpha_1, \alpha_2; \theta) := \alpha_1 \mathcal{L}_{inner}(\theta) + \alpha_2 \mathcal{L}_{diff}(\theta) \tag{4}$$

The $\mathcal{L}_{inner}$ focus on the masked section of the model's output:

$$\mathcal{L}_{inner}(\theta) := \sum_{l=1}^{L} ||(1 - M_l) \odot (I^l_{out}(\theta) - I_{gt})||_1 \tag{5}$$

Finally, the $\mathcal{L}_{diff}$ represents the pixel loss for the masked section of the image derived from the XOR opperation of both the last layer's mask and the current one:

$$\mathcal{M}(l) := M_{l-1}M_l \tag{6}$$

$$\mathcal{L}_{diff}(\theta) := \sum_{l=1}^{L} ||\mathcal{M}(l) \odot (I^l_{out}(\theta) - I_{gt})||_1 \tag{7}$$

Where M_0 is the initial mask.

For the second term, high order representations of the input and ground truth images are generated from a pretrained *VGG 19* architecture, this set P of hidden representations ψ_p are compared from the feature space level. Using the layers *4*, *9* and *18*, just so we have 3 different high order representations, we define our loss function as follows:

$$\mathcal{L}_{inner}(l,p;\theta) := ||\psi_p^{I^l_{out}(\theta)\odot(1-M_l)} - \psi_p^{I_{gt}\odot(1-M_l)}||_1 \tag{8}$$

$$\mathcal{L}_{diff}(l,p;\theta) := ||\psi_p^{I^l_{out}(\theta)\odot\mathcal{M}(l)} - \psi_p^{I_{gt}\odot\mathcal{M}(l)}||_1 \tag{9}$$

$$\mathcal{L}_{perceptual}(\alpha_3;\theta) := \alpha_3 \sum_{l=1}^{L}\sum_{p\in P}\left(\frac{\mathcal{L}_{inner}(l,p;\theta)+\mathcal{L}_{diff}(l,p;\theta)}{N_{\psi_p^{I_{gt}}}}\right) \tag{10}$$

Analysing the style of an image involves the usage of a correlative matrix that explains its features (Gram matrix), this way we lastly define the style loss term as follows:

$$\mathcal{L}_{inner}(l,p;\theta) := ||(\psi_p^{I^l_{out}(\theta)\odot(1-M_l)})^T(\psi_p^{I^l_{out}(\theta)\odot(1-M_l)}) - (\psi_p^{I_{gt}\odot(1-M_l)})^T(\psi_p^{I_{gt}\odot(1-M_l)})||_1 \tag{11}$$

$$\mathcal{L}_{diff}(l,p;\theta) := ||(\psi_p^{I^l_{out}(\theta)\odot\mathcal{M}(l)})^T(\psi_p^{I^l_{out}(\theta)\odot\mathcal{M}(l)}) - (\psi_p^{I_{gt}\odot\mathcal{M}(l)})^T(\psi_p^{I_{gt}\odot\mathcal{M}(l)})||_1 \tag{12}$$

$$\mathcal{L}_{style}(\alpha_4,\alpha_5;\theta) := \sum_{l=1}^{L}\sum_{p\in P}\frac{1}{F_p}(\alpha_4\mathcal{L}_{inner}(l,p;\theta)+\alpha_5\mathcal{L}_{diff}(l,p;\theta))(l,p;\theta) \tag{13}$$

Where $F_p = C_p^3 H_p W_p$: number of channels, height and width of the feature extractor output space.

As seen in the loss terms, we are comparing each layer's output with the ground truth, inducing a teacher forcing like training process that removes any dependency from prior layers.

3 Data

Interval times from the historical CME records has been used, this augments the amount of scenarios where information can be lost. The models were trained by coronagraph product: LASCO C3. These images were downloaded and further processed with a python open-source calibration library named CorKit, which imitates SolarSoft functionalities.

Each data sample was normalized using histogram equalization mappings, and finally resized into a resolution of 1024x1024. The masks that imitate usual missing blocks are randomly generated for each ground truth image, they are identity mappings with a 32nx32n sized chunk dropped.

4 Training

This model, as seen in Table 1, was pre-trained using Adam optimizer, with a learning rate of 0.0002 (lr) and gradient clip at 0.005 (gc) for 20 hours with a Nvidia RTX 4070 dropping the physical constraints from the loss function. Then fine-tuned with a learning rate of 0.00005 and gradient clip of 0.0005 including the physical constraint terms.

Table 1. Training hyperparameters.

	bs	lr	gc	L	α_1	α_2	α_3	α_4	α_5	λ_1	λ_2
Pretraining	4	0.0002	0.005	2	6	6	0.05	120	120	0	0
Fine-tuning	4	0.00005	0.0005	-	0.2	0.8	0.2	1.2	1.8	0	0

5 Results and Discussion

6 Conclusions and future work

Our work proposes a novel framework for image calibration and image restoration, effectively improving fuzzy logic mechanism for missing blocks inpainting. The architecture used in this work could have been more robust adding more layers, but the computational resources available restraint this possibility, that's why it's recommended to try this architectures with more layers. Also, including local residual connections for the encoder and decoder separately could enhance information flow and furthermore the performance in general. Ultimately, another suitable approach could be a multi-modal network that analyses the position, and time between different images, generating a joint calibration routine that would improve CME dynamics capture. The GitHub repository where the source code is allocated is accessible to the reader in the following link: `https://github.com/Jorgedavyd/DL-based-Coronagraph-Inpainting`

Acronyms

LR Learning rate
GC Graddient clip
CME Coronal Mass Ejection
UNet U-shaped neural network
LASCO Large Angle and Spectrometric Coronagraph
VGG Visual Geometry Group
XOR Exclusive or

Notation

$\odot$ Hadamard product
$\boldsymbol{W^{p,k}(X)}$ Sóvolev space with functions' domain begin the set X.
$\boldsymbol{\mathcal{L}}$ General scalar field.
$\boldsymbol{\mathcal{F}}$ General functional field.
$\boldsymbol{W}$ Convolutional weights matrix $\in \mathcal{M}_n^m(R)$.
$\mathbf{1}$ Unit vector $\in \mathcal{R}^{c\times h\times w}$.
$\boldsymbol{A^T}$ Given the matrix A, its tranposed.
$\boldsymbol{a, \alpha}$ General vector $\in R^n$, bold means $n \geq 2$.
$\boldsymbol{\mathcal{M}_n^m(X)}$ Space of matrices $m \times n$ with elements from the set X.
XOR operator.
$\boldsymbol{\delta}$ Functional derivative.
$\boldsymbol{sum()}$ Operator that sums all elements from input matrix.
$\boldsymbol{||x||_1}$ Manhattan distance, L1 norm.

7 Open Research

The LASCO C3 data used for training purposes in the study are available at Naval Research webpage via `https://lasco-www.nrl.navy.mil/lz/level_05` with free access.

1.0.15 of CorKit used for level 1 image calibration is preserved at `https://github.com`, available via MIT License and developed openly at `https://github.com/Jorgedavyd/corkit`. 2.1.1 of PyTorch used to create the architecture and training the model with automatic differentiation is preserved at `https://github.com`, available via BSD 3-Clause License and developed openly at `https://github.com/pytorch/pytorch/`. 6.0.0 of Astropy used for image visualization and fits files management is preserved at `https://github.com`, available via BSD 3-Clause License and developed openly at `https://github.com/astropy/astropy/`. 3.8.2 of Matplotlib used for image visualization is preserved at `https://github.com`, available via BSD 3-Clause License and developed openly at `https://github.com/astropy/astropy/`.

Acknowledgments

The SOHO/LASCO data used here are produced by a consortium of the Naval Research Laboratory (USA), Max-Planck-Institut fuer Aeronomie (Germany), Laboratoire d'Astronomie (France), and the University of Birmingham (UK). SOHO is a project of international cooperation between ESA and NASA.