DEEPLOOP: DEEP Learning for an Optimized adaptive Optics Psf estimation

Morgan Gray, Maxime Dumont, Olivier Beltramo-Martin, Jean-Charles Lambert, Benoit Neichel, Thierry Fusco

DEEPLOOP: DEEP Learning for an Optimized adaptive Optics Psf estimation

Morgan Gray[a], Maxime Dumont[b,a,c], Olivier Beltramo-Martin[d,a], Jean-Charles Lambert[a], Benoit Neichel[a], and Thierry Fusco[b,a]

[a]Aix Marseille Univ, CNRS, CNES, LAM, Marseille, France
[b]DOTA, ONERA, Université Paris Saclay, 91123 Palaiseau, France
[c]INESCTEC, FEUP, Porto, Portugal
[d]Spaceable, 75010 Paris, France

ABSTRACT

DEEPLOOP is a Python toolbox, originally dedicated to the estimation of the parameters of an Adaptive Optics (AO) Point Spread Function (PSF), describing the atmospheric turbulence and the static modes of a telescope. This toolbox is using the Tensorflow/Keras deep learning API and a Graphical Processor Unit (GPU) computing framework. DEEPLOOP is based on a small set of Python scripts dedicated to the data loading, to the Neural Network (NN) models architectures and their compiling, to the training methods, to the learning curves display and to the performances evaluation on the test sets. This toolbox has a great flexibility: it enables to make simulations on a specific parameters grid (for searching the best hyperparameters configuration), to parallelize the calculations on several GPUs (synchronous data parallelism on the same node), and to use some specific 'on-the-fly' images loading for each batch, in order to use very few Random Access Memory (RAM).

In this paper, we will first explain the main characteristics of this toolbox. Then, the first results with data simulations on Keck II telescope will be presented.

Keywords: Deep Learning, Neural Network, DEEPLOOP, Adaptive Optics, Point Spread Function

1. INTRODUCTION

DEEPLOOP is a library of Python scripts (Python versions ≥ 3.6). Each script is a container for a set of classes and functions. During a simulation, the scripts that should be used, are :

a main script (*DLPSF_TrainNN.py*) to define all the numerous configuration parameters (NN models architectures, learning methods, hyperparameters values...) and to run a simulation on GPU(s);

a script for data loading, either to load all the images into the RAM of the GPUs' node (*DLPSF_LoadFunc.py*), or to perform an '*on-the-fly*' loading of the images of each batch (*DLPSF_LoadFuncBG.py*): this allows to use very few RAM, whatever is the size of the dataset during the training step;

a script for choosing one of the available NN models architectures (*DLPSF_NetworkModels.py*). These architectures are not fixed, and can be modified and parameterized very easily by changing the appropriate parameters values in the command line;

a script for compiling, saving and loading a NN model (*DLPSF_CompileFunc.py*) including several loss functions, optimizers, learning rate schedulers and callbacks;

a script for choosing the training method (*DLPSF_TrainFunc.py*): classical training or cross-validation or calculations on a grid of hyperparameters values for a NN optimization research;

a script to display the learning performances (*DLPSF_Results.py*), in order to study the good convergence of the NN;

a script to test the generalization performances (*DLPSF_Predict.py*). Once the NN has been trained with the verification of the several learning curves, it is necessary to evaluate the trained NN on some test sets to study its performances on new data that have never been seen before.

2. SETTING-UP A SIMULATION WITH DEEPLOOP

DEEPLOOP was originally designed to enhance from an image, the retrieval of the parameters of the PSFAO21 model [1], in comparison to PSF-fitting approaches. In this section, we will take an example from the previous issue and describe some of the main parameters you need to define when you run a simulation on a node of GPUs. As there is currently no Graphical User Interface, if you want to run a simulation, you need to use a command line in a job script. For instance, as the main script is named $DLPSF\_TrainNN.py$, in order to pass all the values of 3 chosen parameters into the job script, you will write :

$python3 \quad ./DLPSF\_TrainNN.py \quad --param1 = val1 \quad --param2 = val2 \quad --param3 = val3$

All the parameters described in this section and other very helpful information (which could not be all given in this paper) are described inside the main script.

Of course, you are not obliged to specify the values of all the parameters you can define in DEEPLOOP, as they have default values in the main script. There are 2 versions for the main script: the first version, named $DLPSF\_TrainNN.py$, is loading all the data into the RAM, before starting the training step; the second version, named $DLPSF\_TrainNNwithBG.py$, is loading into the RAM, only the data from each batch: this version is using Numpy generators or Tensorflow tensors generators (by using also the API $tensorflow.data.Dataset$) in order to use very few RAM of the GPUs' node during the simulation.

2.1 GPU configuration

First of all, by definig the $nbGPU$ parameter, you will specify either you run the simulation on CPUs ($nbGPU = 0$), or on one GPU ($nbGPU = 1$), or on all GPUs available on the same node ($nbGPU = -1$). Therefore, if you want to use only one GPU, you have to start your command line by writing :

$python3 \quad ./DLPSF\_TrainNN.py \quad --nbGPU = 1$

In the case of using all GPUs on the same node, the code will use the *distribute.MirroredStrategy()* API from Tensorflow, which enables to distribute computations across multiple devices with *data parallelism*: a single model gets replicated on multiple GPUs (of the same node), each of them processes different batches of data and then, they merge their results at the end of each epoch.

2.2 Loading the initial dataset

Then, you will have to define the original dataset (used for the Training, Validation and Test sets), the total number of images you want to use in this dataset and the kind of normalization you apply on the images, by defining respectively the 3 parameters : $dataName$, $nbImages$ and $typNorm$. Therefore, you need to add this following expression in your command line :

$--dataName = PSFAO21\_NONOISE\_STATIC \quad --nbImages = 280000 \quad --typNorm = 1$

2.3 Training, validation and test sets

From the initial dataset, you have to specify whether you want to create, first a *test set* and then a *validation set* by defining respectively the 2 parameters : $testSet$ and $validSet$. In both cases, you need to specify the fraction of the previous dataset you want to extract, in order to create your *test set* and your *validation set*, by defining respectively the 2 parameters : $testSize$ and $validSize$. These 2 previous parameters can be a fraction (between 0 and 1), or an integer which therefore indicates the number of images in each case.

with one validation set : $--testSet = yes \quad --testSize = 0.15 \quad --validSet = 1 \quad --validSize = 40000$

If you want to make a *cross-validation* with several folds, you will have to define $validSet = 2$ and to specify the number of folds you want to create from your initial dataset by defining the value of the parameter $kFoldCV$. Therefore, you need to add this following expression in your command line :

with KFold cross-validation : $--testSet = yes \quad --testSize = 0.15 \quad --validSet = 2 \quad --kFoldCV = 6$

2.4 Neural Network model

Then, you need to specify the NN model you want to use by defining the $nnModel$ parameter and a specific architecture by defining the following parameters : $nbFiltCL$ (number of filters for each Convolution Layer (CL)), $kernelSizes$ (filters sizes for each CL), $stridesPar$ (strides values for each CL), $poolSize$ (poolsize values for each CL), $nbNeuDL$ (number of neurons for each Dense Layer (DL)) and $dropout$ (dropout rate values). Therefore, in the case of a VGGNet based architecture with 5 CL and 2 DL (without specifying the third and last DL having a number of neurons equal to the number of parameters to estimate), you need to add this following expression in your command line :

$--nnModel = VGGNet \quad --nbFiltCL = 64,128,128,256,256 \quad --kernelSizes = 3,3,3,3,3$

$--stridesPar = 1,1,1,1,1 \quad --poolSize = 2,0,2,0,2 \quad --nbNeuDL = 128,64 \quad --dropout = 0.4,0.4$

Many other classical NN models are available by changing the $nnModel$ parameter : you can also use a ResNet, Inception, Xception, Variational Autoencoder or UNet based model and specify, for some of them, some of the hyperparameters that have been previously described and that will define the specific architecture you want to use and to test.

2.5 Other hyperparameters

Once you have defined the architecture of your NN, you can specify some other classical hyperparameters in the command line, as the size of each batch on one GPU ($batchSize$), an optimizer algorithm proposed by Keras ($optimiZer$) or the maximal number of epochs for the training step ($epochTrain$). Therefore, you need to add this following expression in your command line :

$--batchSize = 32 \quad --optimiZer = Adam \quad --epochTrain = 100$

By defining the parameter $learnSche$, you can either use some *learning rate schedulers* proposed by Keras, or some other *learning rate schedulers* that have been specifically implemented in DEEPLOOP and that are defined by several mathematical power laws. In order to choose one of these specific *learning rate schedulers*, you have to specify also 3 other parameters ($learnR0$, $learnP2$ and $learnP3$) that are defining the mathematical expression of the power law you have chosen. Therefore, you need to add this following expression in your command line :

$--learnSche = 5 \quad --learnR0 = 1e-4 \quad --learnP2 = 5e-6 \quad --learnP3 = 200$

2.6 With a .json parameters file

Each time you run a simulation, a file with all parameters and their corresponding values is created and saved in a .json format. If you want to run a new simulation with the same parameters and the same previous values, except for some of them, you can use the .json file that was previously created. For instance, with a file named $simParam.json$, if you want to change only the number of epochs for the training step, you have to write :

$python3 \quad ./DLPSF\_TrainNN.py \quad --epochTrain = 200 \quad --config = simParam.json$

2.7 With a gridsearch

When you want to optimize the architecture of a NN, or the values of some other hyperparameters that have been described previously, it is necessary to run many simulations. We decided *not to use* the scikit-learn's $GridSearchCV$ function, because we wanted to have more flexibility on the parameter grids, in particular to be able to take into account the *specific learning rate schedulers* (with their own parameters) that we have implemented in DEEPLOOP, in order to make possible the evolution of the learning rate values during the training step.

In this case, you must specify the $gridSearch$ parameter : $--gridSearch = yes$.

Therefore, the main script will not take into account all the hyperparameters defined as before: it will load a dictionary allowing to build a list of simulations to be performed from the different possible combinations of hyperparameters that have been defined in this dictionary.

3. FIRST SIMULATIONS RESULTS

We have used DEEPLOOP on PSFs simulations generated from the segmented and AO-assisted Keck-II telescope. The PSFs were simulated with the PSFAO21 model [1], which is parametric and includes 6 parameters to describe the atmospheric contribution, as well as 9 additional parameters to describe the mirror phase errors of the Keck-II primary mirror. Therefore, the resulting PSF is : $PSF = PSF_{atm} \bigotimes PSF_{tel}$,

PSF_{atm} was generated from the Power Spectrum Density (PSD) described in [2].

In order to reduce the dimensionality of the problem, PSF_{tel} is generated via the Keck-II pupil with only 9 modes of phase errors, that are the most energetic within the 200 on-sky observations available with the Keck-II telescope [3]. For more details on this work, see Appendix A.

The resulting PSFs have been simulated by varying these 15 parameters on appropriate ranges and they have been estimated with a statistical study that also validated the accuracy of the model to represent the on-sky PSFs [1]. We have generated more than 400000 PSFs, from known parameters that were uniformly distributed in consistent ranges, with the performance of the AO system, and with the quality of the observation site.

Subsequently, the objective was to infer, with DEEPLOOP, the parameters of the simulated PSFs produced by the PSFAO21 model [1].

For the simulations presented in this section, as the PSF model has 15 parameters and considering this fit as a regression issue, we have decided to fit it by using a simple VGGNet based architecture (similar to the one chosen in [4], but with 7 CL and 3 DL). The parameters of the simulation were as follows :

$--nbFiltCL = 64, 128, 128, 256, 256, 512, 512$ $--kernelSizes = 3, 3, 3, 3, 3, 3, 3$

$--stridesPar = 1, 1, 1, 1, 1, 1, 1$ $--poolSize = 2, 0, 2, 0, 2, 0, 4$ $--nbNeuDL = 256, 256$

$--learnSche = 5$ $--learnR0 = 2e-4$ $--learnP2 = 2e-6$ $--learnP3 = 200$ $--batchSize = 32$

This final configuration is the result of an optimization work that was performed using the *gridSearch* functionality described in 2.7, in order to search for a suitable VGGNet architecture and several optimized hyperparameters (in particular for the learning rate decay law of the scheduler).

3.1 Without noise

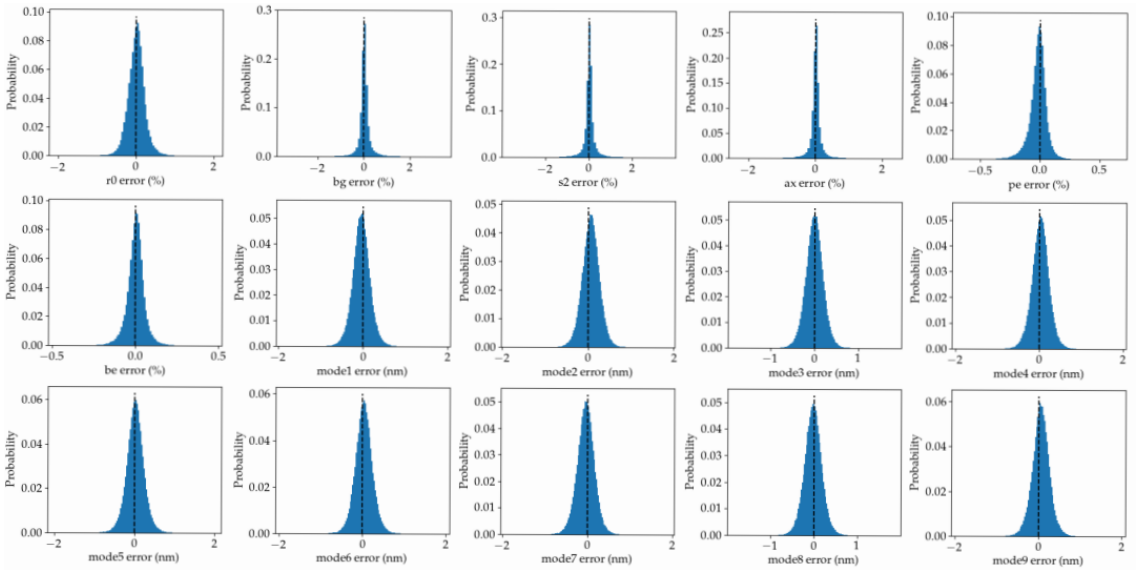

Figure 1. Histograms of the estimation errors (with DEEPLOOP) for the 15 model parameters. The PSFs were generated without noise but contained the 9 modes representing the cophasing errors of the primary mirror of the Keck II telescope.

First, we have simulated PSFs in H-band (1.65µm), without noise. Then, with DEEPLOOP, we have trained a VGGNet based NN in order to estimate the 15 parameters characterizing the PSFs according to the morphology of the 150x150 pixels size images. Note that the field of view of the PSF is chosen to reach 2 times the cutoff frequency of the Deformable Mirror (DM): a large portion of the image is thus strongly constrained by r0 only.

Fig. 1 shows that our NN is able to estimate the 6 key parameters that characterize the atmospheric turbulence residual (atmospheric parameters) to the *nearest percent*, as well as to infer segment alignment errors (telescope parameters) of the *order of nm*.

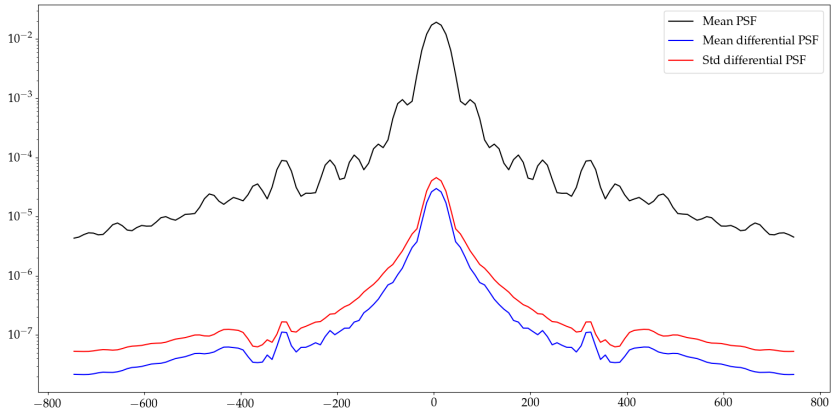

Figure 2. Comparison of the average profile of the simulated PSFs (x-axis in mas) with the average residual profile and the StD, after the subtraction of the modeled PSF and by including atmospheric and telescope parameters.

Fig. 2 compares the average profile of the simulated PSFs with the average residual profile (bias) and the quadratic residual profile or the Standard Deviation (StD). These three profiles, obtained on a test set of 40 000 PSFs, show that the PSFs reconstruction is *better than 0.1% in bias and StD, on all spatial frequencies*.

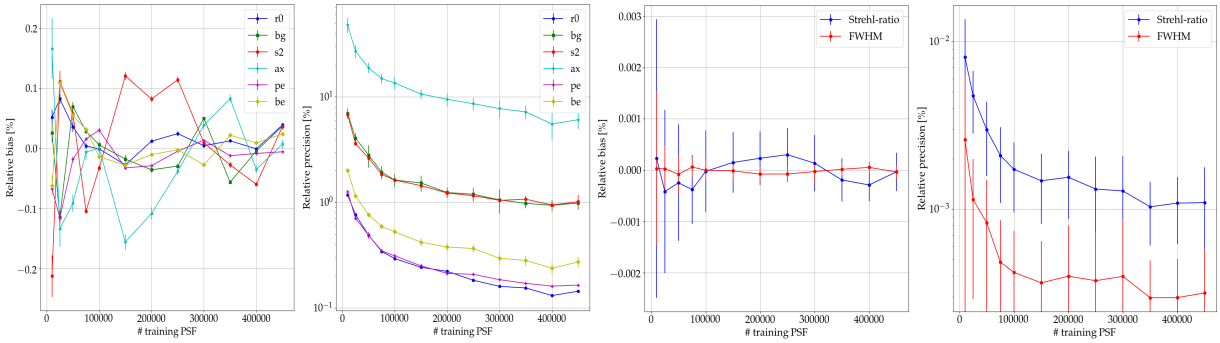

Figure 3. Left : Bias and StD for the 6 parameter estimates; Right : PSF-based metrics (Strehl ratio and FWHM). In both cases, the results are given as a function of the number of PSFs used during the training step.

Fig. 3 shows the sensibility of our NN performance with respect to the number of PSFs (used during the training step). To obtain this assessment, we focused on the errors (bias and StD) over the 6 atmospheric parameters (no telescope aberrations) as well as the Strehl ratio and the Full Width Half Maximum (FWHM) errors. Results are rather intuitive: the more PSFs we use to train the network, the better the precision becomes. Note that the bias remains very good all over the range of training PSFs we considered for this study. Basically, by multiplying this number of training PSFs ten times, we can reduce the error in StD by a factor up to 10, although the gain is concentrated in the first stage of the range of training PSFs below 100k. In such a problem where we try to estimate a few parameters from an image with a 150×150 pixels size, this requires to simulate 10k up to 100k images. This order of magnitude is of course problem-dependent but seems to be relevant for

the extraction of parameters from a PSF. Moreover, depending on the needs of sensitivity on specific metrics, it may be necessary to explore a larger range of training PSFs (1% of precision on s2 for 300k PSFs). A similar analysis will be pursued in the presence of noise.

This study allows us to confirm the very good convergence of our NN on PSFs of realistic shape. Traditional methods of PSF model fitting [1] encounter problems of coupling between atmospheric and telescope parameters when these parameters are estimated jointly, so that reaching such a level of accuracy with these traditional approaches can be compromised without strong a priori, even at very high Signal to Noise Ratio (SNR) [3]. It is then a sensitivity problem : there is a set of (potentially very) different parameters that lead to very similar PSFs; the model fit can then converge to a local minimum depending on the starting point and minimize the atmospheric contribution.

With DEEPLOOP, the VVGNet based model is trained on a set of very different configurations and is thus able to better separate these situations.

3.2 With noise

We also used DEEPLOOP and our NN on noisy data by adding Gaussian noise (readout noise, detector noise) and Poisson noise (star flux, sky background) with the specifications of the NIRC2 detector and the Mauna Kea site to obtain a K-band magnitude 14 (magnitude achievable during observations of the Galactic center). In this case of noisy data, the simulations included only the 6 atmospheric parameters. Moreover, we have also fitted the PSFAO21 model [1] parameters (no model error) with the optimizer (gradient descent) of the P3 library and by setting the initial conditions to the true simulation parameters for each PSF of the test set.

Thus, to each of these PSFs is associated a first vector of 6 parameters *estimated by the NN* with DEEPLOOP, and another vector of 6 parameters *estimated via the model fitting method*: this allowed us to compare DEEPLOOP performances with a state-of-the-art approach.

The choice of fixing the initial conditions of the model fitting to the ground truth allows to avoid the convergence problems towards local minima very far from the global minimum and to identify the robustness of the two methods to the measurement noise. Note that the criterion to be minimized *in the case of the model adjustment* is weighted by the variance of the noise per pixel (which is fixed in simulation and therefore known).

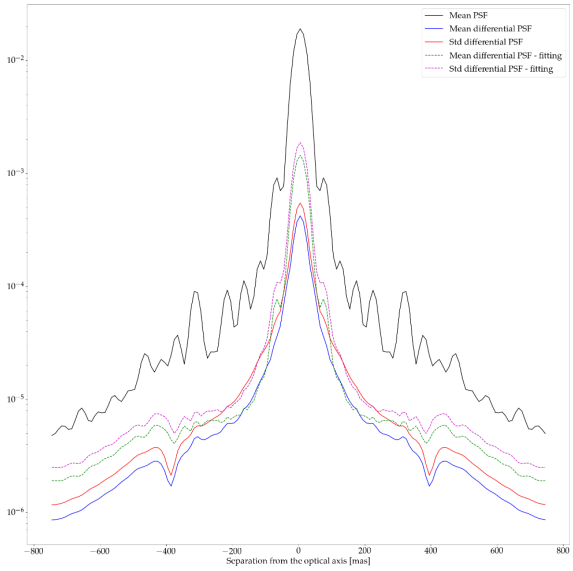

Figure 4. Comparison of the average profile of simulated PSFs on a test sample of 40,000 PSFs with the average residual profile and the StD after PSF subtraction either with DEEPLOOP or with a psf-fitting approach. With DEEPLOOP, the reconstruction is better and more precisely with a gain of a factor 2 to 5.

Fig. 4 shows the comparison of the two methods on the mean and on the quadratic residual profiles. The NN used with DEEPLOOP, allows to *decrease the residual error on average by a factor of 2 to 5 on all spatial frequencies*.

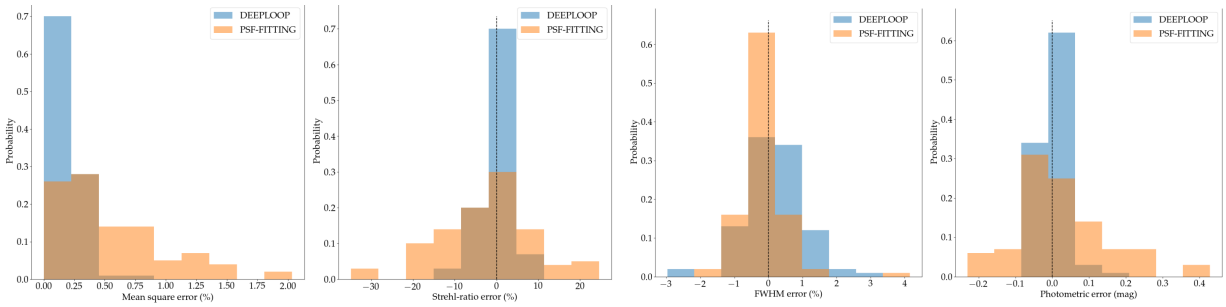

Figure 5. Comparison of the error obtained on 4 key metrics that characterize the PSF: the pixel-to-pixel Root Mean Square Error (RMSE) between the estimated PSF and the true PSF, the error on the Strehl Ratio, the FWHM and the Photometric Error. These 4 metrics are obtained by estimating the PSF via DEEPLOOP or via a classical criterion minimization approach weighted by the noise statistics.

Fig. 5 shows histograms of 4 parameters obtained by comparing the modeled PSF with the ground truth. DEEPLOOP improves overall the characterization of the PSF by a factor of 2, compared to the classical model fitting approach which is however completely optimized.

4. CONCLUSION

The Deep Learning and the data driven approach allow us to improve the characterization of the PSF compared to a model fitting approach, which offers applications on the diagnosis of the AO system via the observed PSF, but also the characterization of atmospheric and static aberrations. Thus, our long-term ambition is to use DEEPLOOP in AO PSF reconstruction, in order to **merge telemetry and image data** (from low order Wavefront Sensor for example).

DEEPLOOP, having a modular and very flexible aspect, is currently answering a regression problem for PSFs estimation, but it can also be used for problems such as **Wavefront analysis** or **AO control laws**. Thus, new NN models architectures are currently being implemented in order to use also DEEPLOOP and its numerous functionalities for these 2 previous issues.

The source code can be retrieved from Gitlab ($https://gitlab.lam.fr/deeploop$).

APPENDIX A. PARAMETERS' IDENTIFICATION ON KECK-II DATA

The PSF fitting is done using the P3 library code ($https://github.com/oliviermartin-lam/P3$), in order to observe the range of evolution of the static aberration parameters. From manually imposed initial conditions on the parameters, the fitting based approach identifies the parameters of the on-sky PSFs. During this step, a coupling between the atmospheric parameters and the telescope parameters appears, and degrades the accuracy of the parameters fitting: the minimization will overestimate the contribution of the static aberrations; conversely, it is very difficult to represent the static aberrations only with the atmospheric contribution.

This could be observed through aberrant results on some parameters, in particular β, the asymptotic slopes of the PSD at large spatial frequencies within the AO correction radius [1]. In the situation of an AO system correcting all atmospheric aberrations with the same relative level, we would obtain $\beta = 11/6$. In AO, we must follow this trend, or, at least obtain a lower value of β because the PSD pattern introduced by the wavefront measurement noise follows a k^{-2} power law. Therefore, β should usually range between 1 and 1.83 for nominal AO correction but can reach higher values in the presence of non-atmospheric aberrations. Nevertheless, there is no physical reason to have $\beta \geq 2$, and the adjustment, with static aberrations, getting values of $\beta \geq 2$ is not normal. It is therefore difficult to adjust both PSD parameters and static aberrations at the same time.

To overcome this coupling, we propose to adjust separately the atmospheric parameters and the telescope. Therefore, we propose to make a *split fitting*:

First of all, the atmospheric parameters (from the PSD) will be adjusted while fixing those of the telescope. The goal is to have a first idea of the range in which the atmospheric parameters evolve. We also obtain a confidence interval on which we can base ourselves for the continuation.

Then, it is the turn of the telescope parameters (static aberrations) to be adjusted. The atmospheric parameters (turbulence) will also be adjusted within the confidence intervals identified in the first step.

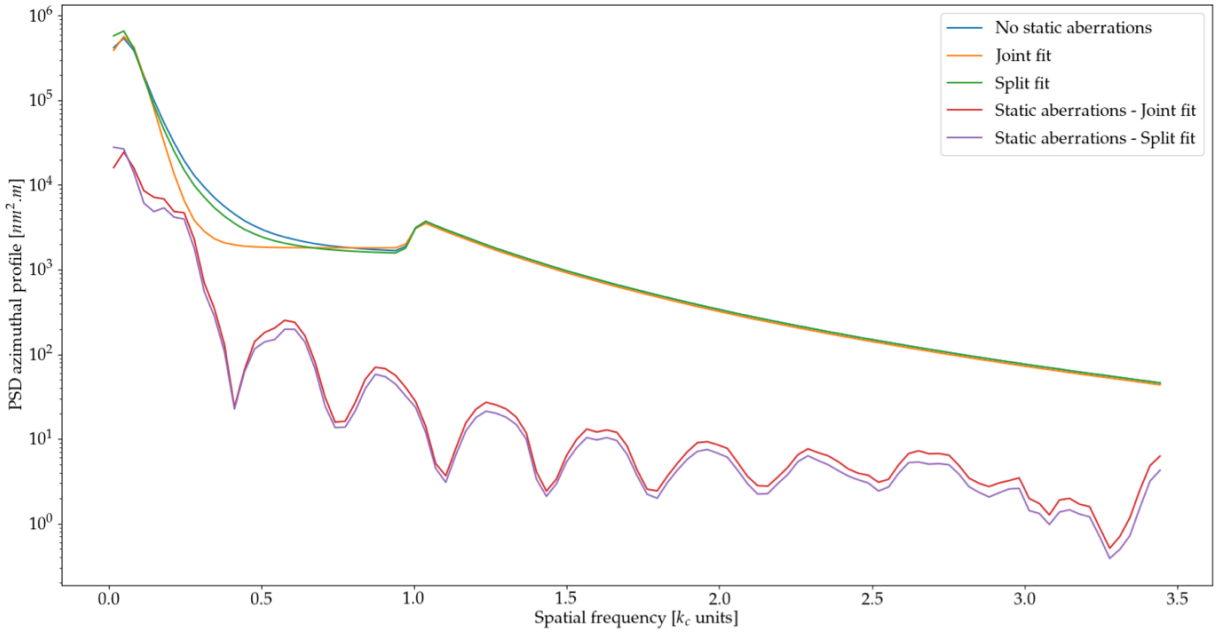

Figure 6. Comparison of PSDs between *joint fitting* and *split fitting*.

Fig. 6 shows the comparisons between the PSDs obtained between *joint fitting* (adjustment of parameters at the same time) and *split fitting* (adjustment in 2 steps). The idea of this new fitting is to decouple the β parameter, which has an outlier value during the joint fitting (orange curve). The ground truth of the dynamic parameters fitting is the blue curve, which corresponds to a fitting without static aberration (the true values of the atmospheric contribution).

As β represents the slope of the PSD curve, it is overestimated in the case of *joint fitting* (orange curve). During the *split fitting*, without static aberration (green curve), we get closer to the real β, which is the objective. We then have to check that during the second step (adjustment on the static parameters), the *split fitting* returns a less energetic static PSD than for the *joint fitting* : it is indeed the case as the red curve is above the blue curve. We have therefore minimized the coupling of β with the static parameters before the cut-off frequency. Therefore, the *split fitting* is more precise and does not involve couplings between parameters.

In order to recover the most energetic modes, we fitted the PSFAO21 model over the 200 NIRC2 PSFs acquired at Keck II during engineering time. The model included the 6 atmospheric parameters as well as the 35 piston values in order to take into account the differential pistons across the segmented pupil.

Therefore, we obtained a dataset of 35 times 200 coefficients from which we aimed at selecting the most energetic modes. A first approach is to simply get the median value of the estimated coefficients over the 200 PSFs. However the uncertainty on the estimates may be rather different from a PSF to another one and we have implemented a weighted average to diminish the impact of the most uncertain estimations. The absolute weighted median value of coefficients as a function of the modes is represented in Fig. 7 and shows that 9 modes

(the most energetic and representative of the PSFs) mostly explain the total wavefront error due to the cophasing errors (we excluded tip and tilt).

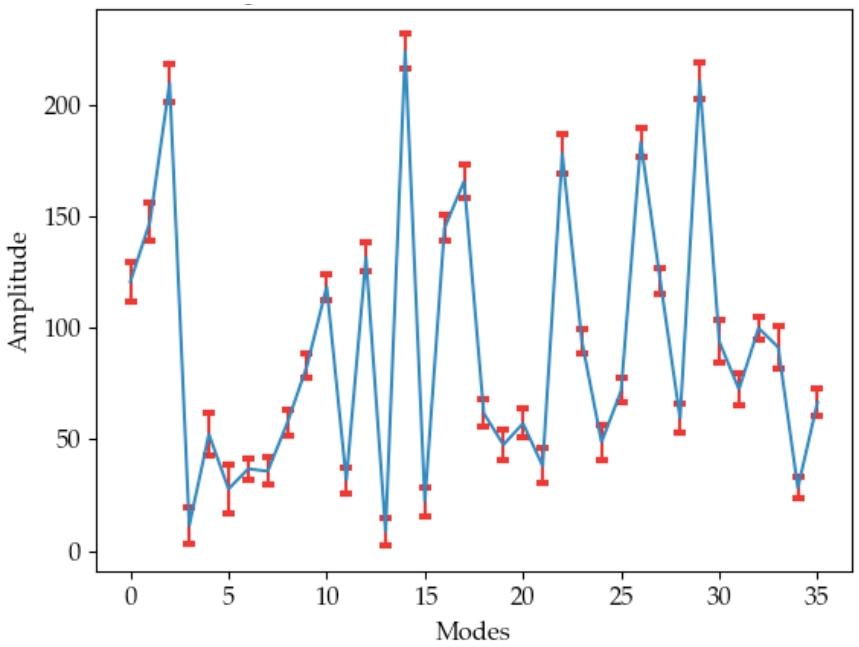

Figure 7. Amplitude in nm of the static aberrations as a function of the modes.

ACKNOWLEDGMENTS

This research has made use of computing facilities operated by the CeSAM data center at LAM (Marseille, France). The Centre de Calcul Intensif d'Aix-Marseille (Marseille, France) is also acknowledged for granting access to its high performance computing resources. This work benefited from the support of the the French National Research Agency (ANR) with WOLF (ANR-18-CE31-0018), APPLY (ANR-19-CE31-0011) and LabEx FOCUS (ANR-11-LABX-0013), the Programme Investissement Avenir F-CELT (ANR-21-ESRE-0008), the Action Spécifique Haute Résolution Angulaire (ASHRA) of CNRS/INSU co-funded by CNES, the ECOS-CONYCIT France-Chile cooperation (C20E02), the ORP H2020 Framework Programme of the European Commission's (Grant number 101004719) and STIC AmSud (21-STIC-09).

REFERENCES

[1] Beltramo-Martin, O., Fétick, R., Neichel, B., and Fusco, T., "Joint estimation of atmospheric and instrumental defects using a parsimonious point spread function model. on-sky validation using state of the art worldwide adaptive-optics assisted instruments," *A&A* **643**, A58 (2020).

[2] Fétick, R., Fusco, T., Neichel, B., Mugnier, L., Beltramo-Martin, O., Bonnefois, A., Petit, C., Milli, J., Vernet, J., Oberti, S., and Bacon, R., "Physics-based model of the adaptive-optics-corrected point spread function. applications to the sphere/zimpol and muse instruments," *A&A* **628**, A99 (2019).

[3] Dumont, M., Beltramo-Martin, O., Gray, M., and Neichel, B., "Deep learning for adaptive optics psf prediction." https://ao4astro2.sciencesconf.org/resource/page/id/5 (2021).

[4] Herbel, J., Kacprzak, T., Amara, A., Refregier, A., and Lucchi, A., "Fast point spread function modeling with deep learning," *JCAP* **2018**, 054 (2018).