CAMBRIDGE
UNIVERSITY PRESS

DL4DS—Deep learning for empirical downscaling

Carlos Alberto Gomez Gonzalez [ORCID]

Earth Sciences Department, Barcelona Supercomputing Center, Barcelona, Spain
E-mail: carlos.gomez@bsc.es

Abstract

A common task in Earth Sciences is to infer climate information at local and regional scales from global climate models. Dynamical downscaling requires running expensive numerical models at high resolution, which can be prohibitive due to long model runtimes. On the other hand, statistical downscaling techniques present an alternative approach for learning links between the large- and local-scale climate in a more efficient way. A large number of deep neural network-based approaches for statistical downscaling have been proposed in recent years, mostly based on convolutional architectures developed for computer vision and super-resolution tasks. This paper presents deep learning for empirical downscaling (DL4DS), a python library that implements a wide variety of state-of-the-art and novel algorithms for downscaling gridded Earth Science data with deep neural networks. DL4DS has been designed with the goal of providing a general framework for training convolutional neural networks with configurable architectures and learning strategies to facilitate the conduction of comparative and ablation studies in a robust way. We showcase the capabilities of DL4DS on air quality Copernicus Atmosphere Monitoring Service (CAMS) data over the western Mediterranean area. The DL4DS library can be found in this repository: https://github.com/carlos-gg/dl4ds

Impact Statement

This paper presents DL4DS the first open-source library with state-of-the-art and novel deep learning algorithms for empirical downscaling.

1. Introduction

Downscaling aims to bridge the gap between the large spatial scales represented by global climate models (GCMs) to the smaller scales required for assessing regional climate change and its impacts (Maraun and Widmann, 2017). This task can be approached via dynamical or statistical downscaling. In the former, a high-resolution regional climate model is nested into a GCM over the domain of interest (Rummukainen, 2010), while the latter aims to learn empirical links between the large- and local-scale climate that are in turn applied to low-resolution climate model output. Statistical or empirical downscaling comes with the benefit of a lower computational cost compared to its dynamical counterpart.

Machine learning (ML) can be defined as the study of computer algorithms that improve automatically by finding statistical structure (building a model) from training data for automating a given task. The field of ML started to flourish in the 1990s and has quickly become the most popular and most successful

---

subfield of artificial intelligence (AI) thanks to the availability of faster hardware and larger training datasets. Deep learning (DL; Bengio et al., 2021) is a specific subfield of ML aimed at learning representations from data putting emphasis on learning successive layers of increasingly meaningful representations (Chollet, 2021). DL has shown potential in a wide variety of problems in Earth Sciences dealing with high-dimensional and complex data and offers exciting new opportunities for expanding our knowledge about the Earth system (Huntingford et al., 2019; Reichstein et al., 2019; Dewitte et al., 2021; Irrgang et al., 2021).

The layered representations in DL are learned via models called neural networks, where an input n-dimensional tensor is received and multiplied with a weights tensor to produce an output. A bias term is usually added to this output before passing the result through an element-wise nonlinear function (also called activation). Convolutional neural networks (CNNs; LeCun et al., 1989) are a type of neural network that rely on the convolution operation, a linear transformation that takes advantage of the implicit structure of gridded data. The convolution operation uses a weight tensor that operates in a sliding window fashion on the data, meaning that only a few input grid points contribute to a given output and that the weights are reused since they are applied to multiple locations in the input. CNNs have been used almost universally for the past few years in computer vision applications, such as object detection, semantic segmentation and super-resolution. CNNs show performances on par with more recent models such as visual Transformers (Liu et al., 2022) while being more efficient in terms of memory and computations.

In spite of recent efforts in the computer science community, the field of DL still lacks solid mathematical foundations. It remains fundamentally an engineering discipline heavily reliant on experimentation, empirical findings and software developments. While scientific software tools such as Numpy (Harris et al., 2020), Xarray (Hamman and Hoyer, 2017), or Jupyter (Kluyver et al., 2016) have an essential role in modern Earth Sciences research workflows, state-of-the-art domain-specific DL-based algorithms are usually developed as proof-of-concept scripts. For DL to fulfill its potential to advance Earth Sciences, the development of AI- and DL-powered scientific software must be carried out in a collaborative and robust way following open-source and modern software development principles.

2. CNN-Based Super-Resolution for Statistical Downscaling

Statistical downscaling of gridded climate variables is a task closely related to that of super-resolution in computer vision, considering that both aim to learn a mapping between low-resolution and high-resolution grids (Wang et al., 2021). Unsurprisingly, several DL-based approaches have been proposed for statistical or empirical downscaling of climate data in recent years (Vandal et al., 2017; Höhlein et al., 2020; Leinonen et al., 2020; Liu et al., 2020; Stengel et al., 2020; Harilal et al., 2021). Most of these methods have in common the use of convolutions for the exploitation of multivariate spatial or spatiotemporal gridded data, that is 3D (height/latitude, width/longitude, channel/variable) or 4D (time, height/latitude, width/longitude, and channel/variable) tensors. Therefore, CNNs are efficient at leveraging the high-resolution signal from heterogeneous observational datasets, from now on called predictors or auxiliary variables, while downscaling low-resolution climatological fields.

In our search for efficient architectures for empirical downscaling, we have developed DL4DS, a library that draws from recent developments in the field of computer vision for tasks such as image-to-image translation and super-resolution. DL4DS is implemented in Tensorflow/Keras, a popular DL framework, and contains a collection of building blocks that abstract and modularize a few key design strategies for composing and training empirical downscaling DL models. These strategies are discussed in the following section.

3. DL4DS Core Design Principles and Building Blocks

The core design principles and general architecture of DL4DS are discussed in the following subsections. An overall architecture of DL4DS is shown in Figure 1, while some of the building blocks and network architectures are shown in Figures 2 and 3.
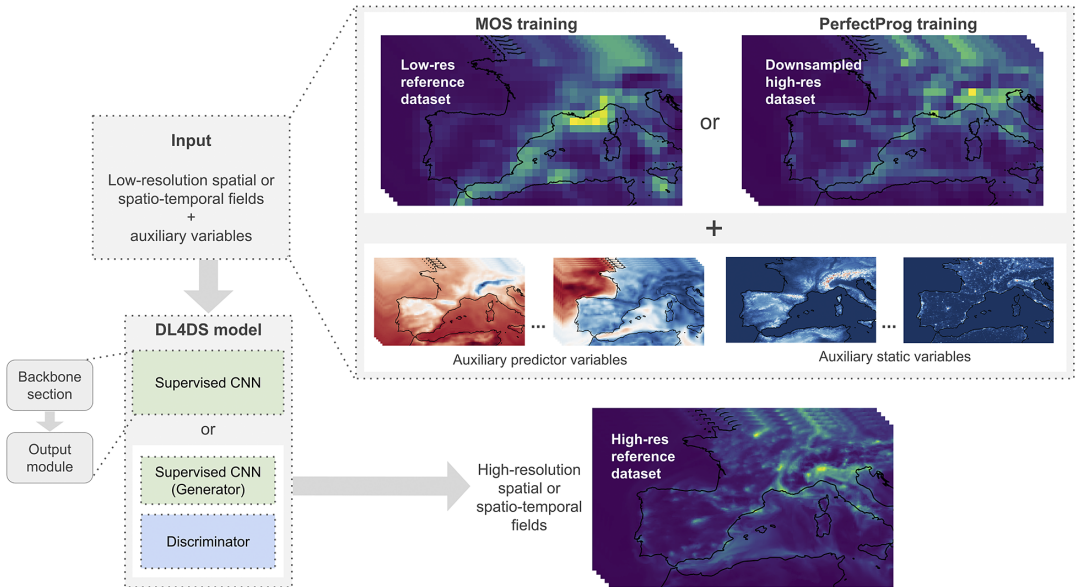
Figure 1. *General architecture of DL4DS. A low-resolution gridded dataset can be downscaled, with the help of auxiliary predictor and static variables, and a high-resolution reference dataset. The mapping between the low- and high-resolution data is learned with either a supervised or a conditional generative adversarial DL model.*
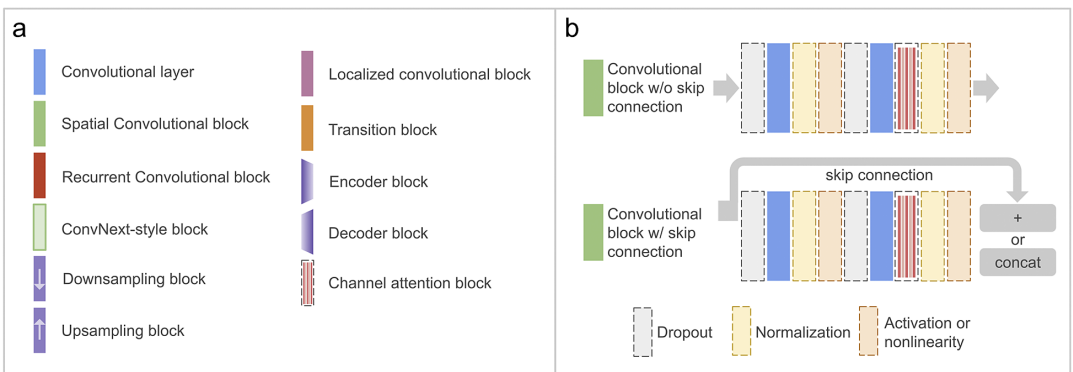

Figure 2. *Panel (a) shows the main blocks and layers implemented in DL4DS. Panel (b) shows the structure of the main spatial convolutional block, a succession of two convolutional layers with interleaved regularization operations, such as dropout or normalization. Blocks and operations shown with dashed lines are optional.*

3.1. Type of statistical downscaling

Statistical downscaling methods aim to derive empirical relationships between an observed high-resolution variable or predictand and low-resolution predictor variables (Maraun and Widmann, 2017). Two main types of statistical downscaling can be defined depending on the origin of the predictors used for training; Model output statistics (MOS), where the predictors are taken directly from GCM outputs,
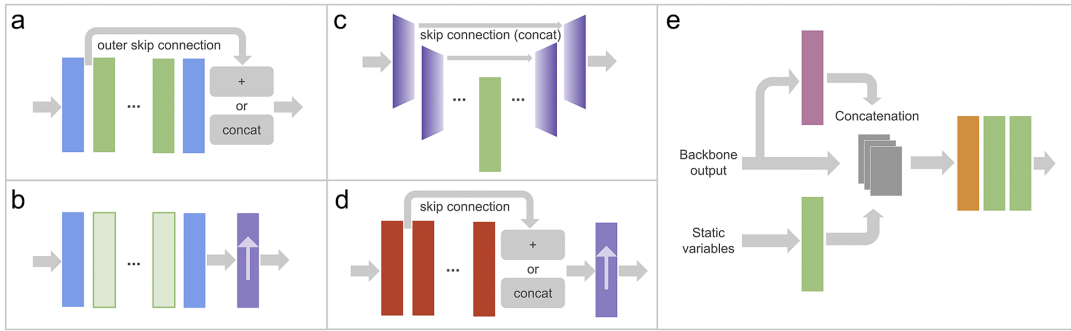
Figure 3. *DL4DS supervised DL models, as well as generators, are composed of a backbone section (examples in panels [a–d]) and an output module (panel [e]). Panel (a) shows the backbone of models for downscaling pre-upsampled spatial samples using either residual or dense blocks. Panel (b) presents the backbone of a model for downscaling spatial samples using ConvNext-like blocks and one of the post-upsampling blocks described in Section 3.4.1. Panel (c) shows the backbone of a model for downscaling pre-upsampled spatial samples using an encoder-decoder structure. Panel (d) shows the backbone of a model for downscaling spatiotemporal samples using recurrent-convolutional blocks and a post-upsampling block. These backbones are followed by the output module (see Section 3.4.2) shown in panel (e). The color legend for the blocks used here is shown in Figure 2a.*

and Perfect Prognosis (PerfectProg) methods, that relies on observational datasets for both predictand and predictors (see Figure 1). Most of the DL-based downscaling methods proposed to date, work in PerfectProg setup, where a high-resolution observational dataset is used to create paired training samples via a downsampling or coarsening operation (Vandal et al., 2017; Leinonen et al., 2020; Stengel et al., 2020). The trained model is then applied to the desired, unseen during training, low-resolution data in a domain-transfer fashion. Other approaches such as those proposed by Höhlein et al. (2020) or Harilal et al. (2021), model a cross-scale transfer function between explicit low-resolution and high-resolution datasets. DL4DS supports both training frameworks, with explicitly paired samples (in MOS fashion) or with paired samples simulated from high-resolution fields (in PerfectProg fashion).

3.2. Multivariate modeling and auxiliary variables

Using high-resolution static predictors fields, such as, topography, distance from the sea, and land-sea mask, as well as high- or intermediate-resolution essential climate variables, can help improve the inference capacity of empirical downscaling methods (Maraun and Widmann, 2017; Vandal et al., 2017; Höhlein et al., 2020). The most straightforward way to include these additional predictors and static variables is to merge (concatenate) them across the channel dimension with the input low-resolution fields. DL4DS can handle an arbitrary number of time-varying high- or intermediate-resolution predictors and high-resolution static variables. Additionally, static high-resolution fields are passed through a convolutional block in the output module as shown in Figure 3e in order to emphasize high-resolution topographic information.

3.3. Preprocessing and normalization

DL4DS implements a couple of preprocessing methods aimed at normalizing or standardizing multi-dimensional gridded datasets, a required step when splitting the data into train and validation/test sets used for training and testing DL models. These procedures, namely mix-max and standardization scalers, are implemented as classes (dl4ds.MinMaxScaler and dl4ds.StandardScaler) in the preprocessing module of DL4DS. These classes extend the functionality of methods from the Scikit-learn package (Pedregosa

et al., 2011) aimed at scaling tabular or 2D data. DL4DS scalers can handle 3D and 4D datasets with NaN values in either numpy.ndarray or xarray.Dataarray formats.

3.4. Architecture of the super-resolution models in DL4DS

The super-resolution networks implemented in DL4DS aim at learning a mapping from low or coarse-resolution to high-resolution grids. The super-resolution networks in DL4DS are composed of two main parts, a backbone section, and an output module, as depicted in Figure 1.

3.4.1. Backbone section

We adopt several strategies for the design of the backbone sections, as shown in Figure 3a–d. The main difference is the particular arrangement of convolutional layers. In DL4DS, the backbone type can be set with the backbone parameter of the training classes mentioned in Section 3.4.1:

- Convnet backbone—Composed of standard convolutional blocks. As shown in Figure 2b, each spatial convolutional block is composed of two convolutional layers (using 3×3 kernels) with some interleaved optional operations, such as dropout (Srivastava et al., 2014), batch/layer normalization or a channel attention block (discussed in Section 3.4.1).
- Resnet backbone—Composed of residual blocks or convolutional blocks with skip connections[1] that learn residual functions with reference to the layer inputs. Residual skip connections have been widely employed in super-resolution models (Wang et al., 2021) in the past. In DL4DS, we implemented cross-layer skip connections within each residual block and outer skip connections at the backbone section level (He et al., 2016).
- Densenet backbone—Composed of dense blocks (see Figure 2b). As in the case of the *Resnet* backbone, we implement cross-layer skip connections within each dense block and outer skip connections at the backbone section level (Huang et al., 2017).
- Unet backbone—Inspired by the structure of the U-Net DL model (Ronneberger et al., 2015), this backbone is composed of encoder and decoder blocks with skip connections, as depicted in Figure 3c. The encoder block consists of a convolutional block plus downsampling via max pooling,[2] while the decoder block consists of upsampling, concatenation with skip feature channels[3] from the encoder section, followed by a convolutional block. This backbone is only implemented for pre-upsampled spatial samples.
- Convnext backbone—Based on the recently proposed ConvNext model (Liu et al., 2022), we implemented a backbone with depth-wise convolutional layers and larger kernel sizes (7x7). To the best of our knowledge, this is the first time that a ConvNext-style block architecture has been applied to the task of statistical downscaling.

3.4.1.1. Spatial and spatiotemporal modeling.

While CNNs excel at modelling spatial or gridded data, hybrid convolutional and recurrent architectures are designed to exploit spatiotemporal sequences. Convolutional Long Short-Term Memory (ConvLSTM, Shi et al., 2015) and convolutional Gated-Recurrent-Units (Ballas et al., 2015) are examples of models that have been used for downscaling time-evolving gridded data (Leinonen et al., 2020; Harilal et al., 2021). DL4DS can model either spatial or spatiotemporal data, by either using standard convolutional blocks or recurrent convolutional blocks. The spatiotemporal network in Figure 3d contains recurrent convolutional blocks to handle 4D training

---

[1] Skip or shortcut connections are connections that feed the output of a particular layer to later non-sequential or non-adjacent layers in the network. Element-wise addition is used in residual blocks, while concatenation is used in dense blocks or in decoder blocks of a U-Net.

[2] Max pooling is a sample-based discretization process that selects the maximum element from the region of the feature map covered by the filter.

[3] A feature channel or feature map refers to the output of a convolutional layer.

samples (with time dimension). The structure of these recurrent convolutional blocks is similar to that of the main convolutional block shown in Figure 2b but using convolutional LSTM layers instead of standard convolutional ones.

3.4.1.2. Channel attention. In DL4DS, we implement a channel attention mechanism based on those of the Squeeze-and-Excitation networks (Hu et al., 2018) and the Convolutional Block Attention Module (Woo et al., 2018). This attention mechanism exploits the inter-channel relationship of features by providing a weight for each channel in order to enhance those that contribute the most to the optimization and learning process. First, it aggregates spatial information of a feature map by using average pooling.[4] The resulting vector is then passed through two 1x1 convolutional blocks and a sigmoid activation to create the channel weights (attention maps) which are multiplied element-wise to each corresponding feature map. The channel attention mechanism is integrated as an optional step in DL4DS convolutional blocks to get refined feature maps throughout the network.

3.4.1.3. Upsampling techniques. The upsampling operation is a crucial step for DL-based empirical downscaling methods. In computer vision, upsampling refers to increasing the number of rows and columns, and therefore the number of pixels, of an image. Increasing the size of an image is also often called upscaling, which oddly enough carries the opposite meaning in the weather and climate jargon. When it comes to downscaling or super-resolving gridded data, it is of utmost importance to preserve fine-scale information, or to transfer it from the high-resolution to the low-resolution fields, while increasing the number or grid points in both horizontal and vertical directions. In DL4DS, we implement several well-established upsampling methods proposed in the super-resolution literature (Wang et al., 2021). These methods belong to two main upsampling strategies, depending on whether the upsampling happens before the network or inside of it: pre-upsampling and post-upsampling. In the former case, the models are fed with a pre-upsampled (via interpolation) low-resolution input and a reference high-resolution predictand. In the latter case, the low-resolution input is directly fed together with the high-resolution predictand. In this scenario, we learn a mapping from low-resolution to high-resolution fields in low-dimensional space with end-to-end learnable layers integrated at the end of the backbone section (see Figure 3b,d). This results in an increased number of learnable parameters, but an increased computational efficiency thanks to working on smaller grids. When working with DL4DS, the upsampling method can be set with the upsampling parameter of the training classes discussed in Section 3.5:

- PIN pre-upsampling—Models trained with interpolation-based pre-upsampled input.
- RC post-upsampling—Models including a resize convolution block (bilinear interpolation followed by a convolutional layer).
- DC post-upsampling—Models including a deconvolution (also called transposed convolution) block (Dong et al., 2016). With a transposed convolution we perform a transformation opposite to that of a normal convolution. In practice, the image is expanded by inserting zeros and performing a convolution. This upsampling method can sometimes produce checkerboard-like patterns that hurt the quality of the downscaled or super-resolved products.
- SPC post-upsampling—Models including a subpixel convolution block (Lim et al., 2017). A subpixel convolution consists of a regular convolution with s^2 filters (where s is the scaling factor) followed by an image-reshaping operation called a phase shift. Instead of putting zeros in between grid points as it is done in a deconvolution layer, we calculate more convolutional filters in lower resolution and reshape them into an upsampled grid.

---

[4] Average pooling is a sample-based discretization process that computes the average value from the region of the feature map covered by the filter.

In the context of empirical downscaling, pre-upsampling has been used by Vandal et al. (2017), Harilal et al. (2021), resize convolution by Leinonen et al. (2020), transposed convolution by Höhlein et al. (2020) and sub-pixel convolution by Liu et al. (2020). Choosing an upsampling method depends on the problem at hand (MOS or PerfectProg training) and the computational resources available.

3.4.2. Output module

The second part of every DL4DS network is the output module, as shown in Figure 3e. Here we concatenate the intermediate outputs of three branches: the backbone section feature channels, the backbone section output passed through a localized convolutional block, and a separate convolutional block over the input high-resolution static variables. These concatenated feature channels are in turn passed through a transition block, a 1x1 convolution used to control the number of channels, and two final convolutional blocks. The first of these two convolutional blocks in the output module applies a channel attention block by default.

3.4.2.1. Localized convolutional block
CNNs bring many positive qualities for the exploitation of spatial or gridded data. However, the property of translation invariance of CNNs is usually sub-optimal in geospatial applications where location-specific information and dynamics need to be taken into account. This has been addressed by Uselis et al. (2020) for the task of weather forecasting. Within DL4DS, we implement a Localized Convolutional Block (LCB) located in the output module of our networks. This LCB consists of a bottleneck transition layer, via 1x1 convolutions to compress the number of incoming feature channels, followed by a locally connected layer with biases (similar to a convolutional one, except that the weights here are not shared). To the best of our knowledge, this is the first time the idea of LCBs has been applied to the task of statistical downscaling.

3.5. Loss functions
Loss functions are used to measure reconstruction error while guiding the optimization of neural networks during the training phase.

3.5.1. Supervised losses
DL4DS implements pixel-wise losses, such as the Mean Squared Error (MSE or L2 loss) or the Mean Absolute Error (MAE or L1 loss), as well as structural dissimilarity (DSSIM) and multi-scale structural dissimilarity losses (MS-DSSIM), which are derived from the Structural Similarity Index Measure (SSIM, Wang et al., 2004). The SSIM is a computer vision metric used for measuring the similarity between two images, based on three comparison measurements: luminance, contrast, and structure. The MS-DSSIM loss has been used by Chaudhuri and Robertson (2020) for the task of statistical downscaling.

3.5.2. Adversarial losses
Generative adversarial networks (GANs; Goodfellow et al., 2014) are a class of ML models in which two neural networks contest with each other in a zero-sum game, where one agent's gain is another agent's loss. This adversarial training was extended for its use with images for image-to-image translation problems by Isola et al. (2017). Conditional GANs map input to output images while learning a loss function to train this mapping, usually allowing a better reconstruction of high-frequency fine details. DL4DS implements a conditional adversarial loss (Isola et al., 2017) by training the two networks depicted in Figure 4. The role of the generator is to learn to generate high-resolution fields from their low-resolution counterparts while the discriminator learns to distinguish synthetic high-resolution fields from reference high-resolution ones. Through iterative adversarial training, the resulting generator can produce outputs consistent with the distribution of real data, while the discriminator cannot distinguish between the generated high-resolution data and the ground truth. Adversarial training has been used in DL-based downscaling approaches proposed by Chaudhuri and Robertson (2020), Leinonen et al. (2020) and
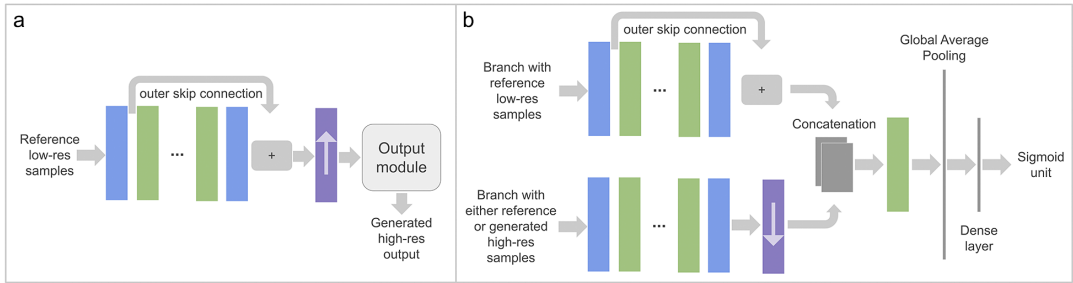
Figure 4. *Example of a conditional generative adversarial model for spatiotemporal samples in post-upsampling mode (see Section 3.4.1). Two networks, the generator shown in panel (a), and discriminator shown in panel (b), are trained together optimizing an adversarial loss (see Section 3.5). The color legend for the blocks used here is shown in Figure 2a.*

Stengel et al. (2020). In out tests, the generated high-resolution fields produced by the CGAN generators exhibit moderate variance, even when using the Monte Carlo dropout technique (Gal and Ghahramani, 2016) which amounts to applying dropout at inference time.

When working with DL4DS, the training strategy is controlled by choosing one of the training classes: dl4ds.SupervisedTrainer or dl4ds.CGANTrainer, for using supervised or adversarial losses accordingly.

4. Experimental Results

Downscaling applications are highly dependent on the target variable/dataset at hand, and therefore the optimal model architecture should be found in a case-by-case basis. In this section, we give a taste of the capabilities of DL4DS without conducting a rigorous comparison of model architectures and learning configurations. For our tests, we use data from the Copernicus Atmosphere Monitoring Service (CAMS) reanalysis (Inness et al., 2019), the latest global reanalysis dataset of atmospheric composition produced by the European Centre for Medium-Range Weather Forecasts (ECMWF), which provides estimates of Nitrogen dioxide (NO2) surface concentration. We select NO2 data from the period between 2014 and 2018 at a 3-hourly temporal resolution which results in $\sim$14.6 k temporal samples. For our low-resolution dataset, we use the CAMS global reanalysis (CAMSRA) with a horizontal resolution of $\sim$80 km. Our high-resolution target is the CAMS regional reanalysis produced for the European domain with a spatial resolution of $\sim$10 km (0.1°). We also include predictor atmospheric variables from the ECMWF ERA5 reanalysis (Hersbach et al., 2020), namely 2 m temperature and 10 m wind speed, with an intermediate resolution of $\sim$25 km (0.25°). Finally, we include as static variables: topography from the Global Land One-km Base Elevation Project and a land-ocean mask derived from it, and a layer with the urban area fraction derived from the land cover dataset produced by the European Space Agency. In order to make our test less heavy on memory requirements, we spatially subset the data and focus on the western Mediterranean region, as shown in Figure 5.

We showcase eight models trained with DL4DS, without the intention of a full exploration of possible architectures and learning strategies. Different loss functions, backbones, learning strategies, and other parameters are combined in the model architectures detailed in Table 1. The training is carried out using a single cluster node with four NVIDIA V100 GPUs. The data is split into train and test sets by reserving the last year of data for test and validation. We fit the dl4ds.StandardScaler on the training set (getting the global mean and standard deviation) and applied it to both training and test sets (subtracting the global mean and dividing by the global standard deviation). All models are trained with the Adam optimizer (Kingma and Ba, 2015), for 100 epochs in the case of supervised CNNs and 18 epochs in the case of the conditional adversarial models.
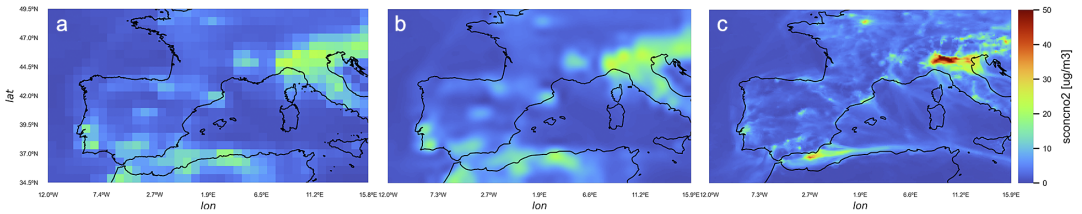
Figure 5. *A reference NO2 surface concentration field from the low-resolution CAMS global reanalysis is shown in panel (a). In panel (b), we present a resampled version, via bicubic interpolation, of the low-resolution reference field. This interpolated field looks overly smoothed and showcases the inefficiency of simple resampling methods at restoring fine-scale information. Panel (c): the corresponding high-resolution field from the CAMS regional reanalysis. Both low- and high-resolution grids were taken from the holdout set for the same time step. The maximum value shown corresponds to the maximum value in the high-resolution grid.*

Table 1. DL4DS models showcased in Section 4.

Panel	Downscaling	Learning	Sample type	Backbone	Upsampling	LCB	Loss
a	PerfectProg	Supervised	Spatial	Unet	PIN	No	Mae
b	MOS	Supervised	Spatial	Unet	PIN	No	Mae
c	PerfectProg	Supervised	Spatial	Resnet	SPC	No	Dssim+mae
d	MOS	Supervised	Spatial	Resnet	SPC	Yes	Dssim+mae
e	MOS	Supervised	Spatial	Resnet	SPC	Yes	Mae
f	MOS	Adversarial	Spatial	Resnet	SPC	Yes	Mae
g	MOS	Supervised	Spatiotemp	Resnet	SPC	Yes	Dssim+mae
h	MOS	Supervised	Spatial	Convnext	SPC	Yes	Mae

Note. The column named "panel" points to the corresponding plot in Figures 6–8.
Abbreviations: LCB, localized convolutional block; MOS, model output statistics; SPC, subpixel convolution; PIN, Pre-upsampling via INterpolation.

Figure 6 shows examples of predicted high-resolution fields produced by the models detailed in Table 1 and corresponding to the reference fields of Figure 5. These give a visual impression of the reconstruction ability of each model. Figures 7 and 8 show the Pearson correlation and Root Mean Square Error (RMSE) for each one of the aforementioned models. The metrics in these maps are computed for each grid point independently and for the whole year of 2018 (containing 2,920 holdout grids). Table 2 gives a more complete list of metrics (including the MAE, SSIM, and peak signal-to-noise ratio [PSNR]) computed for each grid pair separately (time step) and then summarized by taking the mean and standard deviation over the samples of the holdout year.

For this particular task and datasets, and according to the metrics of Table 2, we find that models trained in MOS fashion perform better than those in PerfectProg and that a supervised model with residual blocks and subpixel convolution upsampling provides the best results. Also, we note that models trained with an LCB perform better than those without it, thanks to the fact that the LCB learns grid point- or location-specific weights. As expected, when the same network is optimized with respect to a DSSIM+MAE loss, it reaches higher scores in the SSIM and PSNR metrics. Training the CGAN model for more epochs could improve the results but would require longer runtimes and computational resources.
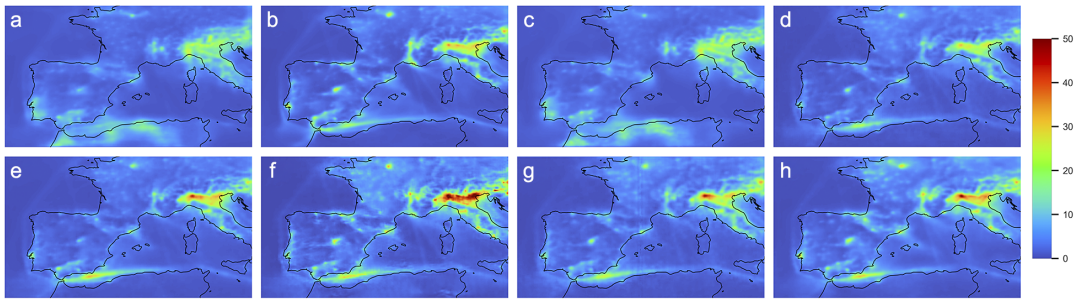
Figure 6. *Examples of downscaled products obtained with DL4DS, corresponding to the reference grid shown in Figure 5a. The models corresponding to each panel are detailed in Table 1.*
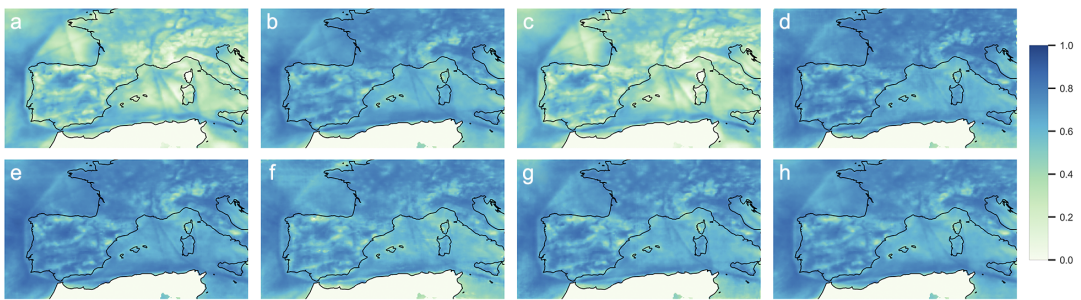

Figure 7. *Pixel-wise Pearson correlation for each model, computed for the whole year of 2018.*
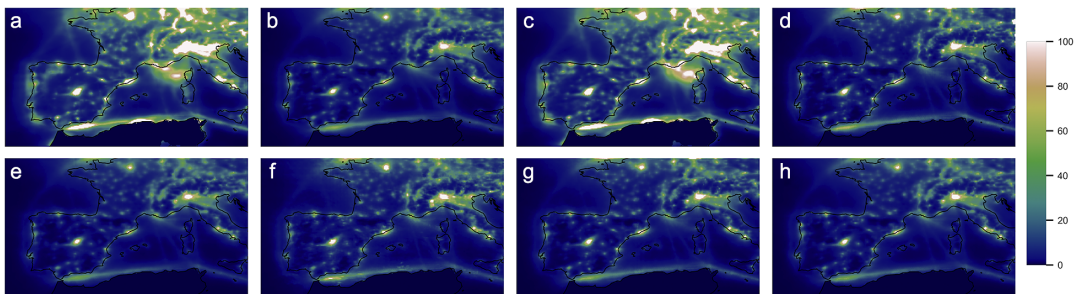

Figure 8. *Pixel-wise RMSE for each model, computed for the whole year of 2018. The dynamic range is shared for all the panels, with a fixed maximum value to facilitate the visual comparison.*

5. Summary and Future Directions

In this paper, we have presented DL4DS, a Python library implementing state-of-the-art and novel DL algorithms for empirical downscaling of gridded data. It is built on top of Tensorflow/Keras (Chollet et al., 2015), a modern and versatile DL framework. Among the key strengths of DL4DS are its simple API consisting of a few user-facing classes controlling the training procedure, its support for distributed GPU training (via data parallelism) powered by Horovod (Sergeev and Balso, 2018), the generous number of customizable DL-based architectures included, the possibility of composing new architectures based on the general purpose building blocks offered, the option of saving and retraining models, and its support of different downscaling frameworks that learn either with explicit low-resolution and high-resolution data

Table 2. Metrics computed for each time step, downscaled product with respect to the reference grid, of the holdout year for the models showcased in this section.

Panel	MAE	RMSE	PearCorr	SSIM	PSNR
a	2.58 ± 0.92	4.37 ± 1.50	0.64 ± 0.11	0.81 ± 0.05	32.95 ± 2.97
b	1.56 ± 0.54	2.70 ± 0.87	0.85 ± 0.04	0.89 ± 0.03	34.61 ± 2.78
c	2.58 ± 0.93	4.40 ± 1.47	0.64 ± 0.11	0.88 ± 0.04	38.85 ± 3.00
d	1.60 ± 0.60	4.75 ± 5.03	0.76 ± 0.20	**0.99 ± 0.01**	**64.87 ± 6.28**
e	**1.49 ± 0.51**	**2.56 ± 0.84**	**0.88 ± 0.03**	0.90 ± 0.03	35.23 ± 2.83
f	1.70 ± 0.58	2.87 ± 0.90	0.84 ± 0.04	0.87 ± 0.03	34.60 ± 2.81
g	1.53 ± 0.56	2.68 ± 0.96	0.86 ± 0.04	0.89 ± 0.03	34.97 ± 3.03
h	1.51 ± 0.55	2.64 ± 0.90	0.87 ± 0.03	0.90 ± 0.03	35.03 ± 2.97

Note. Best models depending on the loss/metric used. Model d is the best for SSIM and PSNR (related) metrics. Model e for MAE-RMSE-PearCor.
Abbreviations: MAE, mean absolute error; PSNR, peak signal-to-noise ratio; RMSE, root-mean-square error; SSIM, structural similarity index measure.

(MOS-like) or only with a high-resolution dataset (PerfectProg-like). DL4DS is a powerful tool for reproducible empirical downscaling experiments and for network architecture optimization. A thorough ablation study varying the depth and width of each backbone, in combination with other design choices, such as the upsampling method, and tested over several benchmark datasets, shall allow a complete comparison of CNN-based DL models for empirical downscaling.

DL has shown promise in post-processing and bias correction tasks (Rasp and Lerch, 2018; Grönquist et al., 2020), so a natural research direction is to explore the application of DL4DS to related post-processing problems. Other interesting research directions are the implementation of uncertainty estimation techniques besides Monte Carlo dropout, the extension of DL4DS models to the case of non-gridded reference data (observational station data), the implementation of alternative adversarial losses such as the Earth mover's distance (Wasserstein GAN), and the inclusion of alternative DL algorithms such as normalizing flows (Groenke et al., 2020).

References

Ballas N, **Yao L**, **Pal C and Courville A** (2015) Delving deeper into convolutional networks for learning video representations. https://dblp.org/rec/journals/corr/BallasYPC15.html.

Bengio Y, **Lecun Y and Hinton G** (2021) Deep learning for ai. *Communications of the ACM 64*(7), 58–65.

Chaudhuri C and Robertson C (2020) Cligan: A structurally sensitive convolutional neural network model for statistical downscaling of precipitation from multi-model ensembles. *Water 12*(12), 3353.

Chollet F (2021) *Deep Learning with Python*. New York: Manning.

Chollet F et al. (2015) Keras. Available at https://keras.io/getting_started/faq/#how-should-i-cite-keras.

Dewitte S, **Cornelis JP**, **Müller R and Munteanu A** (2021) Artificial intelligence revolutionises weather forecast, climate monitoring and decadal prediction. *Remote Sensing 13*(16), 3209.

Dong C, **Loy CC and Tang X** (2016) Accelerating the super-resolution convolutional neural network. In Leibe B, Matas J, Sebe N and Welling M (eds), *Computer Vision – ECCV 2016*. Cham: Springer International Publishing, pp. 391–407.

Gal Y and Ghahramani Z (2016) Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML'16)*, vol *48*, pp. 1050–1059. Available at https://dl.acm.org/doi/10.5555/3045390.3045502.

Goodfellow I, **Pouget-Abadie J**, **Mirza M**, **Xu B**, **Warde-Farley D**, **Ozair S**, **Courville A and Bengio Y** (2014) Generative adversarial nets. In Ghahramani Z, Welling M, Cortes C, Lawrence N and Weinberger KQ (eds), *Advances in Neural Information Processing Systems*, vol. *27*. Curran Associates, Inc. Available at https://papers.nips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

Groenke B, **Madaus L and Monteleoni C** (2020) Climalign: Unsupervised statistical downscaling of climate variables via normalizing flows. In *Proceedings of the 10th International Conference on Climate Informatics*. New York, NY: Association for Computing Machinery, pp. 60–66.

Grönquist P, **Yao C**, **Ben-Nun T**, **Dryden N**, **Dueben P**, **Li S and Hoefler T** (2020) Deep learning for post-processing ensemble weather forecasts. *Philosophical Transactions of the Royal Society A*, published online 15 February 2021. Available at https://doi.org/10.1098/rsta.2020.0092

Hamman J and Hoyer S (2017) Xarray: N-d labeled arrays and datasets in python. *Journal of Open Research Software 5*, 10.

Harilal N, **Singh M and Bhatia U** (2021) Augmented convolutional lstms for generation of high-resolution climate change projections. *IEEE Access 9*, 25208–25218.

Harris CR, **Millman KJ**, **van der Walt SJ**, **Gommers R**, **Virtanen P**, **Cournapeau D**, **Wieser E**, **Taylor J**, **Berg S**, **Smith NJ**, **Kern R**, **Picus M**, **Hoyer S**, **van Kerkwijk MH**, **Brett M**, **Haldane A**, **del Ro JF**, **Wiebe M**, **Peterson P**, **Gérard-Marchant P**, **Sheppard K**, **Reddy T**, **Weckesser W**, **Abbasi H**, **Gohlke C and Oliphant TE** (2020) Array programming with NumPy. *Nature 585*(7825), 357–362.

He K, **Zhang X**, **Ren S and Sun J** (2016) Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. Available at https://ieeexplore.ieee.org/document/7780459.

Hersbach H, **Bell B**, **Berrisford P**, **Hirahara S**, **Horányi A**, **Muñoz-Sabater J**, **Nicolas J**, **Peubey C**, **Radu R**, **Schepers D**, **Simmons A**, **Soci C**, **Abdalla S**, **Abellan X**, **Balsamo G**, **Bechtold P**, **Biavati G**, **Bidlot J**, **Bonavita M**, **De Chiara G**, **Dahlgren P**, **Dee D**, **Diamantakis M**, **Dragani R**, **Flemming J**, **Forbes R**, **Fuentes M**, **Geer A**, **Haimberger L**, **Healy S**, **Hogan RJ**, **Hólm E**, **Janisková M**, **Keeley S**, **Laloyaux P**, **Lopez P**, **Lupu C**, **Radnoti G**, **de Rosnay P**, **Rozum I**, **Vamborg F**, **Villaume S and Thépaut J-N** (2020) The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society 146*(730), 1999–2049.

Höhlein K, **Kern M**, **Hewson T and Westermann R** (2020) A comparative study of convolutional neural network models for wind field downscaling. *Meteorological Applications 27*(6), e1961.

Hu J, **Shen L and Sun G** (2018) Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141. Available at https://ieeexplore.ieee.org/document/8578843.

Huang G, **Liu Z**, **Maaten LVD and Weinberger KQ** (2017) Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA: IEEE Computer Society, pp. 2261–2269. Available at https://www.researchgate.net/publication/320966887_Image-to-Image_Translation_with_Conditional_Adversarial_Networks.

Huntingford C, **Jeffers ES**, **Bonsall MB**, **Christensen HM**, **Lees T and Yang H** (2019) Machine learning and artificial intelligence to aid climate change research and preparedness. *Environmental Research Letters 14*(12), 124007.

Inness A, **Ades M**, **Agust-Panareda A**, **Barré J**, **Benedictow A**, **Blechschmidt A-M**, **Dominguez JJ**, **Engelen R**, **Eskes H**, **Flemming J**, **Huijnen V**, **Jones L**, **Kipling Z**, **Massart S**, **Parrington M**, **Peuch V-H**, **Razinger M**, **Remy S**, **Schulz M and Suttie M** (2019) The cams reanalysis of atmospheric composition. *Atmospheric Chemistry and Physics 19*(6), 3515–3556.

Irrgang C, **Boers N**, **Sonnewald M**, **Barnes EA**, **Kadow C**, **Staneva J and Saynisch-Wagner J** (2021) Towards neural Earth system modelling by integrating artificial intelligence in Earth system science. *Nature Machine Intelligence 3*(8), 667–674.

Isola P, **Zhu J-Y**, **Zhou T and Efros AA** (2017) Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Available at https://www.researchgate.net/publication/320966887_Image-to-Image_Translation_with_Conditional_Adversarial_Networks.

Kingma DP and Ba J (2015) Adam: A method for stochastic optimization. In Bengio Y and LeCun Y (eds), *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings. Available at https://dblp.org/rec/journals/corr/KingmaB14.html.

Kluyver T, **Ragan-Kelley B**, **Pérez F**, **Granger B**, **Bussonnier M**, **Frederic J**, **Kelley K**, **Hamrick J**, **Grout J**, **Corlay S**, **Ivanov P**, **Avila D**, **Abdalla S and Willing C** (2016) Jupyter notebooks – A publishing format for reproducible computational workflows. In Loizides F and Schmidt B (eds), *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, pp. 87–90. Available at https://eprints.soton.ac.uk/403913.

LeCun Y, **Boser B**, **Denker JS**, **Henderson D**, **Howard RE**, **Hubbard W and Jackel LD** (1989) Backpropagation applied to handwritten zip code recognition. *Neural Computation 1*(4), 541–551.

Leinonen J, **Nerini D and Berne A** (2020) Stochastic super-resolution for downscaling time-evolving atmospheric fields with a generative adversarial network. *IEEE Transactions on Geoscience and Remote Sensing*, *59* 1–13. Available at https://ieeexplore.ieee.org/document/9246532.

Lim B, **Son S**, **Kim H**, **Nah S and Lee KM** (2017) Enhanced deep residual networks for single image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Available at https://www.researchgate.net/publication/319277381_Enhanced_Deep_Residual_Networks_for_Single_Image_Super-Resolution.

Liu Y, **Ganguly AR and Dy J** (2020) *Climate Downscaling Using YNet: A Deep Convolutional Network with Skip Connections and Fusion*. New York, NY: Association for Computing Machinery, pp. 3145–3153.

Liu Z, **Mao H**, **Wu C-Y**, **Feichtenhofer C**, **Darrell T and Xie S** (2022) A convnet for the 2020s. Available at https://www.kdd.org/kdd2020/accepted-papers/view/climate-downscaling-using-ynet-a-deep-convolutional-network-with-skip-conne.

Maraun D and Widmann M (2017) *Statistical Downscaling and Bias Correction for Climate Research*. Cambridge: Cambridge University Press.

Pedregosa F, **Varoquaux G**, **Gramfort A**, **Michel V**, **Thirion B**, **Grisel O**, **Blondel M**, **Prettenhofer P**, **Weiss R**, **Dubourg V**, **Vanderplas J**, **Passos A**, **Cournapeau D**, **Brucher M**, **Perrot M and Duchesnay E** (2011) Scikit-learn: Machine learning in python. *Journal of Machine Learning Research 12*, 2825–2830.

Rasp S and Lerch S (2018) Neural networks for postprocessing ensemble weather forecasts. *Monthly Weather Review 146*(11), 3885–3900.

Reichstein M, **Camps-Valls G**, **Stevens B**, **Jung M**, **Denzler J**, **Carvalhais N and Prabhat** (2019) Deep learning and process understanding for data-driven Earth system science. *Nature 566*(7743), 195–204.

Ronneberger O, **Fischer P and Brox T** (2015) U-net: Convolutional networks for biomedical image segmentation. In Navab N, Hornegger J, Wells WM and Frangi AF (eds), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, pp. 234–241.

Rummukainen M (2010) State-of-the-art with regional climate models. *WIREs Climate Change 1*(1), 82–96.

Sergeev A and Balso MD (2018) Horovod: fast and easy distributed deep learning in tensorflow. Available at https://www.semanticscholar.org/paper/Horovod:-fast-and-easy-distributed-deep-learning-in-Sergeev-Balso/e2c8726d092aea573e69f5b0a2654225883cfacf.

Shi X, **Chen Z**, **Wang H**, **Yeung D-Y**, **Wong W-k and Woo W-c** (2015) Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*, vol. *1*. Cambridge, MA: MIT Press, pp. 802–810.

Srivastava N, **Hinton G**, **Krizhevsky A**, **Sutskever I and Salakhutdinov R** (2014) Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*(1), 1929–1958.

Stengel K, **Glaws A**, **Hettinger D and King RN** (2020) Adversarial super-resolution of climatological wind and solar data. *Proceedings of the National Academy of Sciences 117*(29), 16805–16815.

Uselis A, **Lukoševičius M and Stasytis L** (2020) Localized convolutional neural networks for geospatial wind forecasting. *Energies 13*(13), 3440.

Vandal T, **Kodra E**, **Ganguly S**, **Michaelis A**, **Nemani R and Ganguly AR** (2017) Deepsd: Generating high resolution climate change projections through single image super-resolution. In *Proceedings of the 23rd ACM SIGKDD International Conference*. New York, NY: Association for Computing Machinery, pp. 1663–1672.

Wang Z, **Bovik A**, **Sheikh H and Simoncelli E** (2004) Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*(4), 600–612.

Wang Z, **Chen J and Hoi SH** (2021) Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 43*(10), 3365–3387.

Woo S, **Park J**, **Lee J-Y and Kweon IS** (2018) Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)* Available at https://www.semanticscholar.org/paper/de95601d9e3b20ec51aa33e1f27b1880d2c44ef2.