Deep learning-based super-resolution of climate (forecast) data

Carlos Alberto Gómez Gonzalez

PyconES 2021

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

STARS POST-DOCTORAL PROGRAMME


About me

`whoami && hostnamectl`

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Postdoctoral fellow at the Earth Sciences department of the Barcelona Supercomputing Center (BSC) on Artificial Intelligence (AI) and Machine Learning (ML) projects

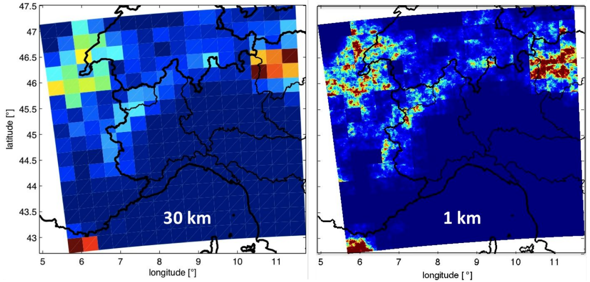
About the BSC:

- MareNostrum is the most powerful supercomputer in Spain
- BSC hosts other clusters, such as the Power-CTE, composed of 52 compute nodes each with 4 NVIDIA V100 (Volta) GPUs
- ES department hosts 100+ researchers with expertise in climate science, atmospheric composition among other topics

Cross-disciplinarity and data science

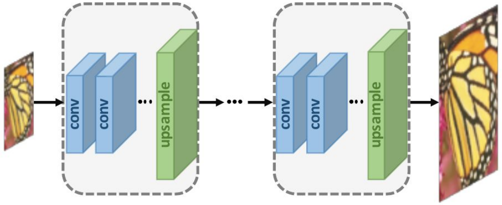
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Climate Science



- Problem definition (need)
- Background and baseline approaches
- Data sources identification
- Validation metrics

Computer vision



- (Re)framing problems
- Cutting edge DL approaches for Earth science applications

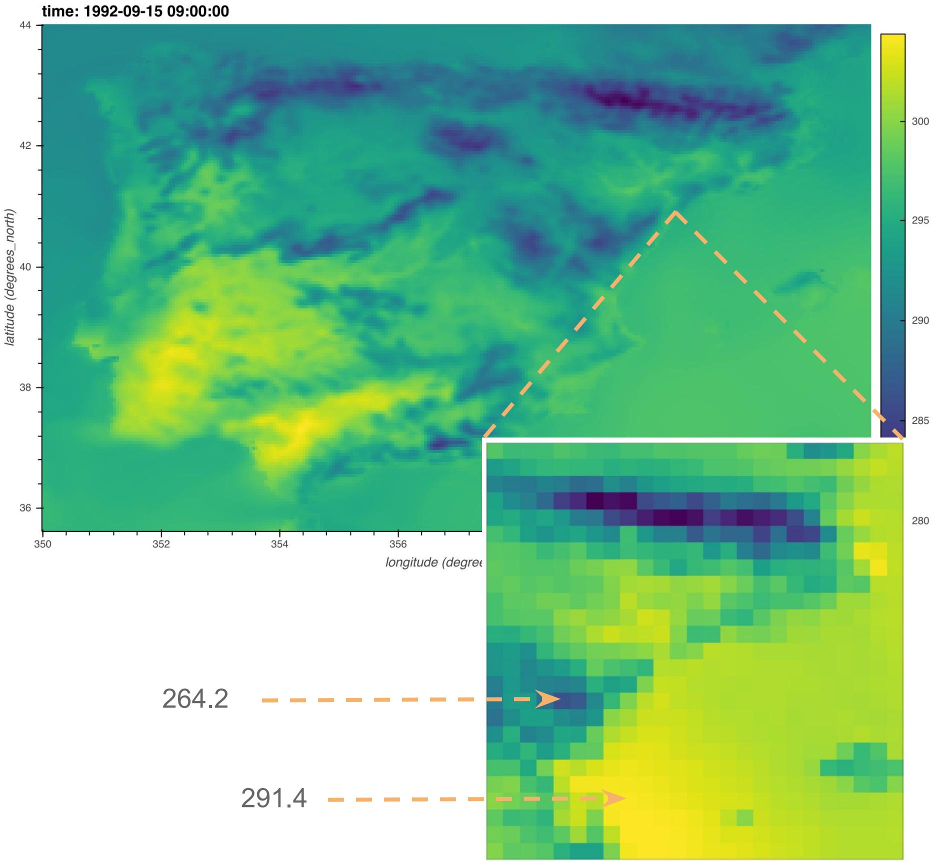
AI engineering

```python
import os
import datetime
import tensorflow as tf
from tensorflow.keras.utils import Progbar
import horovod.tensorflow as hvd
import numpy as np
from dl4ds.resnet_int import resnet_int
from dl4ds.resnet_rec import resnet_rec
from dl4ds.resnet_spc import resnet_spc
from dl4ds.discriminator import residual_discriminator
from dl4ds.dataloader import create_pair_hr_lr
from dl4ds.utils import (Timing, list_devices,
                         set_gpu_memory_growth,
                         set_visible_gpus)
```

- Smart testing and model design/tuning
- Development of robust and efficient code
- Reusability/reproducibility

Introduction

Computer Vision for Earth Science

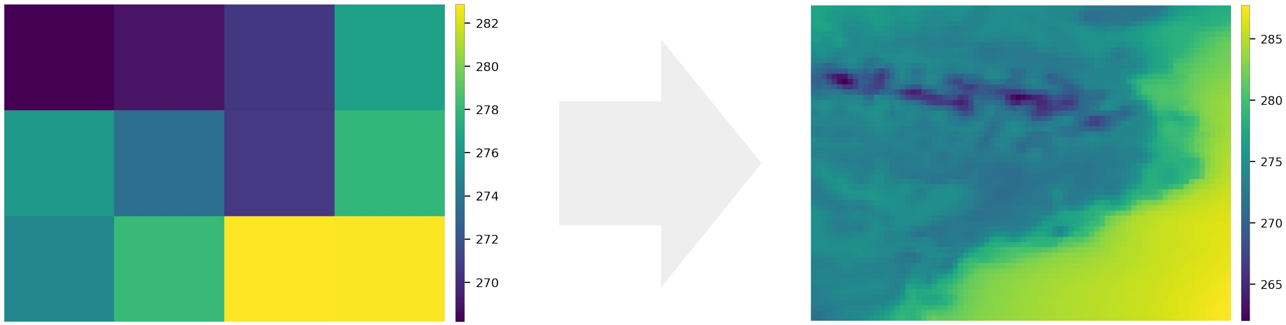


- Spatio-temporal processes
- Common structural prior. Gridded data can be treated as arrays of pixels (images or videos)
- Tasks in CV that relate to a problem in ES:
  - Next frame video prediction → regression and forecasting
  - Super-resolution → downscaling
  - Object recognition → pattern finding
  - Inpainting → missing data filling
  - Image to image translation → transfer functions, regression

Computer Vision for Earth Science (reality)

Why super-resolution or downscaling

- Super-resolution aims to convert a given low-resolution image with coarse details to a corresponding high-resolution image with better visual quality and refined details
- Similar goal to that of Statistical Downscaling (SD) techniques in Climate science. SD is much cheaper that running the physical model at higher resolution
- Having more resolution (giving local insights) is important for many applications
- Careful with the terminology:
  - downscaling in climate science == upscaling in CV (lower → higher resolution)

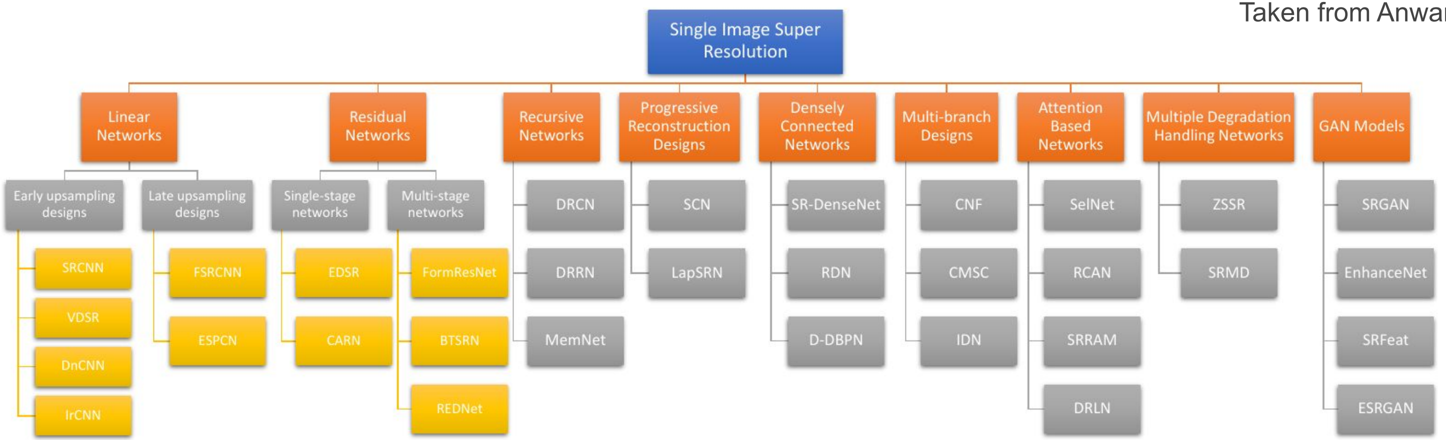


DL-based super-resolution

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

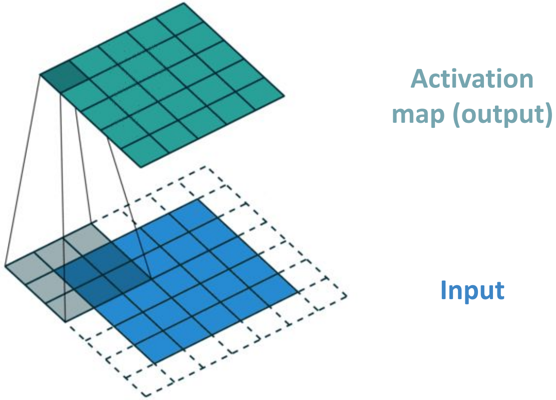
- A large number of super-resolution models have been proposed in the field of computer vision

Taken from Anwar et al., 2020

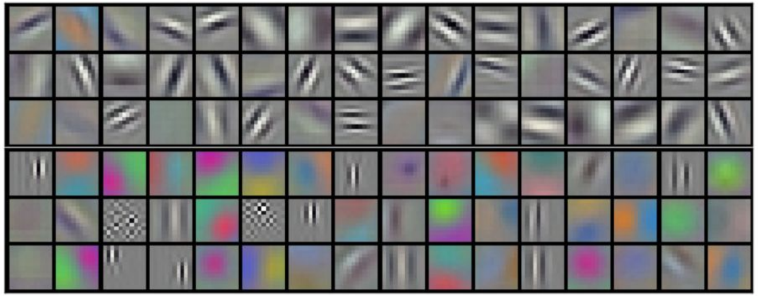


- These ideas have inspired DL-based downscaling methods in climate science, e.g., Vandal et al. 2017, Leinonen et al. 2020, Stengel et al. 2020, Wang et al. 2021, etc

CNNs in a nutshell



2D convolution using a kernel size of 3 (sliding shadow) with stride of 1 and padding



96 convolutional kernels of size 11×11×3 learned by the first convolutional layer of an image classification CNN. From Krizhevsky et al. 2012

- The convolutional layer is the core building block of a CNN and does most of the computational heavy lifting
- Its parameters consist of a set of learnable filters (see image on the top right)
- During the forward pass, we slide each filter across the width and height of the input and compute dot products between the entries of the filter and the input at any position

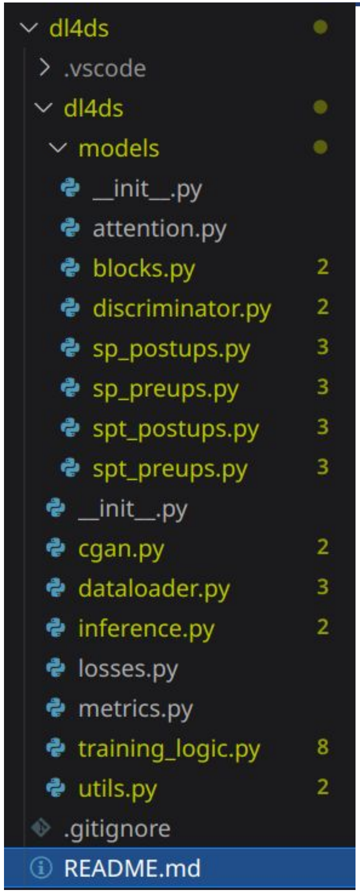
Python and scientific software: DL4DS

Scientific Software Development matters

- Ugly and tedious job (the most common approach is to publish proofs of concepts)
- (We) scientists suck at programming !!!
  - DRY (Don't Repeat Yourself) **
  - Modularization, abstraction, testing is beneficial in scientific sw development
- Reproducibility crisis is even more relevant in ML/AI
- Python is a vital tool in modern science, data science and ML/AI
- Examples of scientific packages: numpy, scikit-learn, xarray, many other niche packages

** https://www.earthdatascience.org/courses/earth-analytics/automate-science-workflows/write-efficient-code-for-science-r/

DL4DS package



DL4DS (Deep Learning for DownScaling)

The different design choices can be combined into 48 different network architectures:

Training	Sample type	Backbone block	Upsampling strategy
Supervised (non-adversarial)	Spatial	Plain Conv	Pre-upsampling: interpolation
Adversarial (conditional)	Spatio-temporal	Residual	Post-upsampling: sub-pixel convolution (SPC)
		Dense	Post-upsampling: resize convolution (RC)
			Post-upsampling: deconvolution (DC)

DL4DS package

dl4ds > dl4ds > models > blocks.py > ...

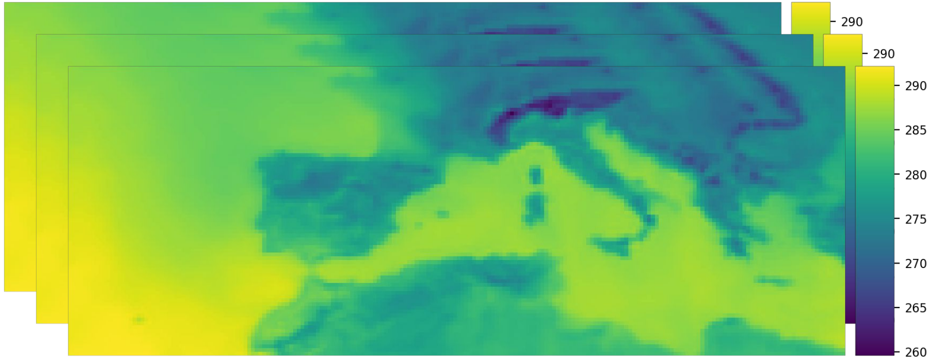
```python
import tensorflow as tf
from tensorflow.keras.layers import (Add, Conv2D, ConvLSTM2D,
                                     SeparableConv2D, BatchNormalization,
                                     LayerNormalization, Activation,
                                     SpatialDropout2D, Conv2DTranspose,
                                     SpatialDropout3D, Concatenate)

from .attention import ChannelAttention2D


class ConvBlock(tf.keras.layers.Layer):
    """Convolutional block.

    References
    ----------
    [1] Effective and Efficient Dropout for Deep Convolutional Neural Networks:
    https://arxiv.org/abs/1904.03392
    [2] Rethinking the Usage of Batch Normalization and Dropout:
    https://arxiv.org/abs/1905.05928
    """
    def __init__(self, filters, strides=1, ks_cl1=(3,3), ks_cl2=(3,3),
                 activation='relu', normalization=None, attention=False,
                 dropout_rate=0, dropout_variant=None, depthwise_separable=False,
                 **conv_kwargs):
        super().__init__()
        self.normalization = normalization
        self.attention = attention
        self.dropout_variant = dropout_variant
        self.dropout_rate = dropout_rate
        self.depthwise_separable = depthwise_separable
```

- Written on top of Tensorflow and Keras
- Series of custom layers and model architectures
- Allows easier model design and hyper-parameter optimization
- Distributed training (Horovod)

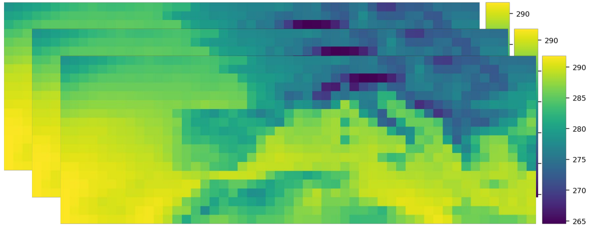
Climate gridded data

Daily surface air temperature grids from 1981 to 2020 (~15k time samples) for the Mediterranean region

Observational reference (ERA5 tas 0.25º)

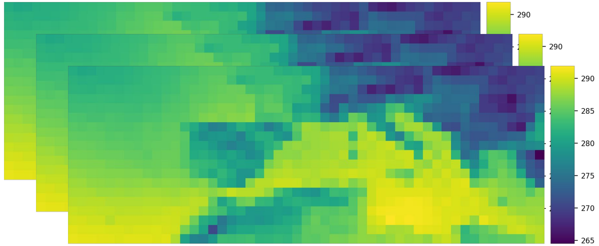


Seasonal forecast (SEAS5 tas 1º, 4x coarser)



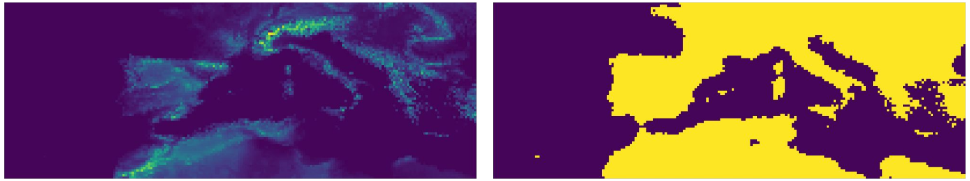
Ensemble member 1

...



Ensemble member n

Static fields (elevation and land-ocean mask)



DataGenerator for training samples

```python
class DataGenerator(tf.keras.utils.Sequence):
    """
    A sequence structure guarantees that the network will only train once on
    each sample per epoch which is not the case with generators.
    Every Sequence must implement the __getitem__ and the __len__ methods. If
    you want to modify your dataset between epochs you may implement
    on_epoch_end. The method __getitem__ should return a complete batch.

    """
    def __init__(self,
        array,
        scale=4,
        batch_size=32,
        patch_size=None,
        time_window=None,
        topography=None,
        landocean=None,
        predictors=None,
        model='resnet_spc',
        interpolation='bicubic',
        repeat=None,
        ):
        """
        Parameters
        ----------
```

```python
    def __getitem__(self, index):
        """
        Generate one batch of data as (X, y) value pairs where X represents the
        input and y represents the output.
        """
```

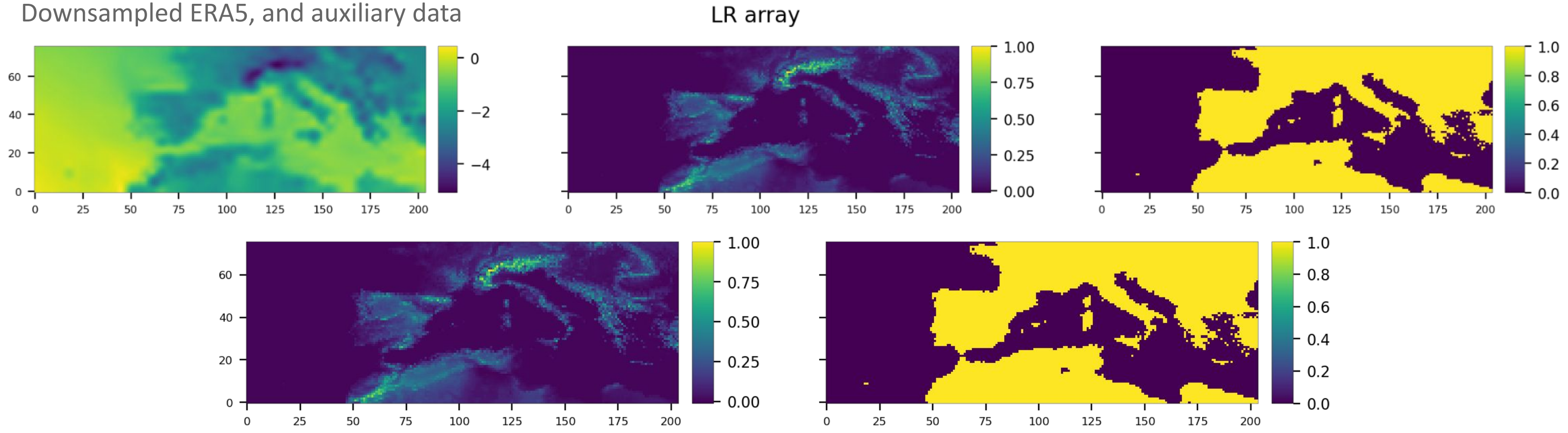
- We subclass keras Sequence
- __getitem__ returns a batch of samples (X, y) for supervised training
- All the preprocessing is done here (cropping, resizing, slicing, etc)

Training samples (pre-upsampling)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Inputs (x)

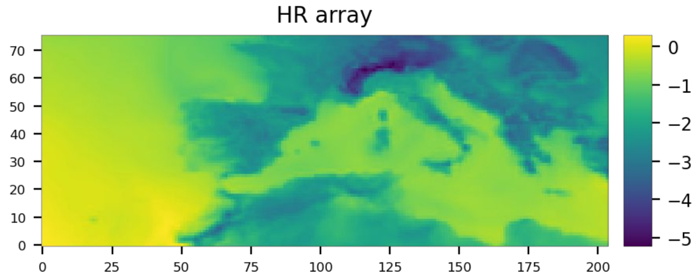
Downsampled ERA5, and auxiliary data



Elevation

Ocean / land

Output (y)



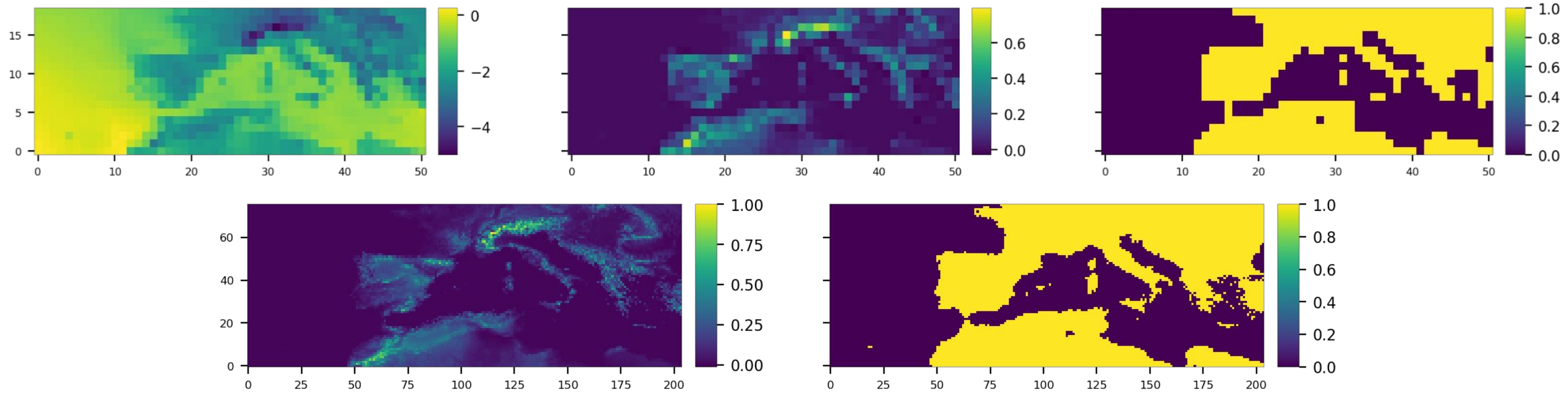
ERA5 temperature observational reference

Training samples (post-upsampling)

Inputs (x)

Downsampled ERA5, and auxiliary data

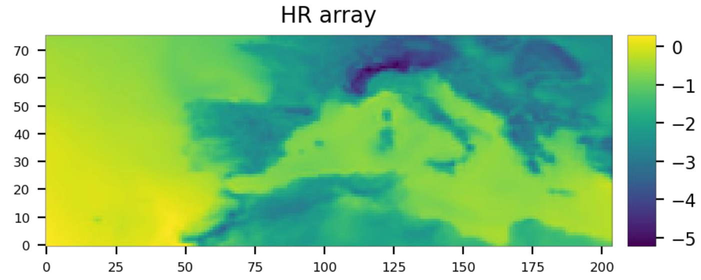
LR array



Elevation

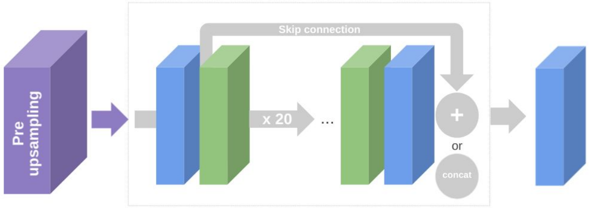
Ocean / land

Output (y)

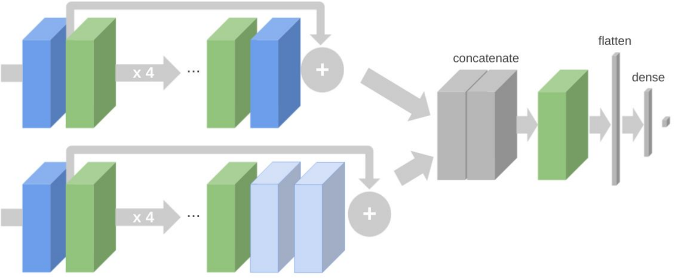


ERA5 temperature observational reference

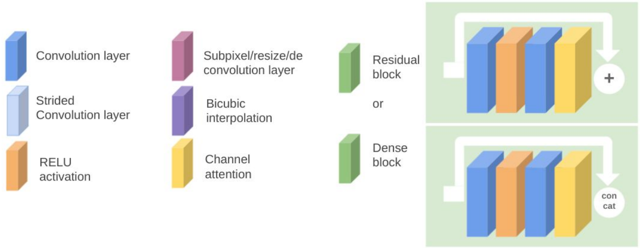
Network architectures (pre-upsampling)



Supervised network or Generator



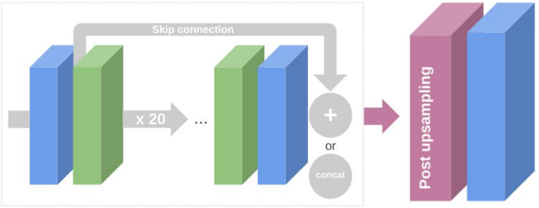
Residual Discriminator



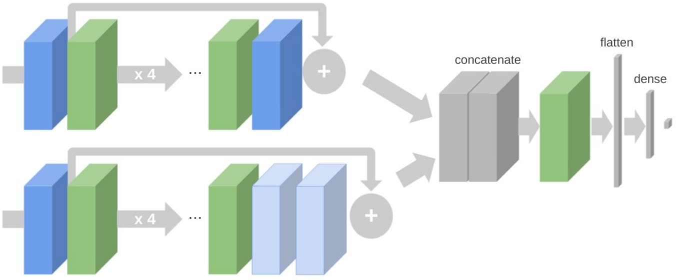
- The network can be trained alone in a fully **supervised** manner
- Both networks, generator and discriminator can be trained in a **conditional adversarial** fashion
- For a pre-upsampling architecture the images are upsampled/resized via bicubic interpolation before entering the network
- The network feature learning is done in HR and is more computationally expensive

Network architectures (post-upsampling)

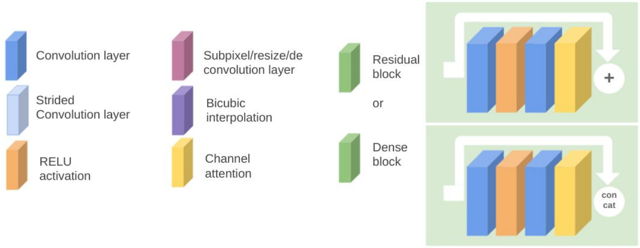
Barcelona Supercomputing Center
Centro Nacional de Supercomputación



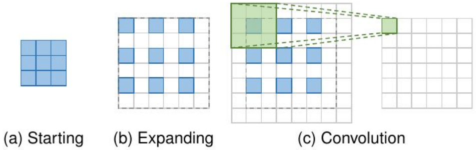
Supervised network or Generator



Residual Discriminator

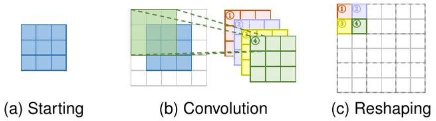


- For a post-upsampling architecture the LR images are not resized before the network
- The upsampling is performed as a layer of the network itself
- The network feature learning is done in LR and is more computationally efficient
- Deconvolution:



Taken from Wang et al., 2020

- Subpixel convolution:



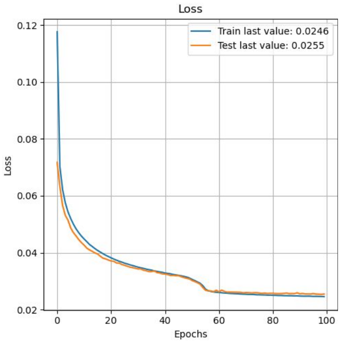
Taken from Wang et al., 2020

DL4DS training API

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

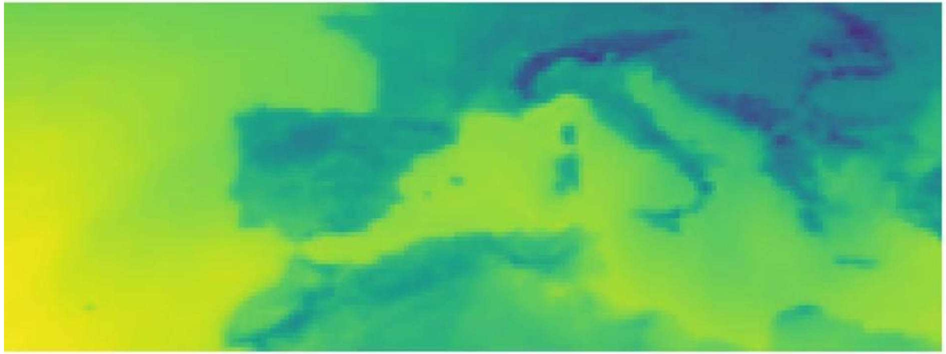
```
GEN_PARAMS = dict(n_filters=32, n_blocks=4, dropout_rate=0.2, dropout_variant='spatial')
DISC_PARAMS = dict(n_filters=32, n_res_blocks=4)

model = dds.CGANTrainer(
    'resnet_spc',
    data_train,
    data_test,
    scale=SCALE,
    topography=TOPO,
    landocean=LAOC,
    predictors_train=None,
    predictors_test=None,
    interpolation='bicubic',
    patch_size=None,
    batch_size=16,
    epochs=100,
    loss='mae',
    learning_rates=(2e-4, 2e-4),
    save=True,
    savecheckpoint_path=None,
    verbose=1,
    device='GPU',
    save_loss_history=False,
    generator_params=GEN_PARAMS,
    discriminator_params=DISC_PARAMS)

model.run()
```

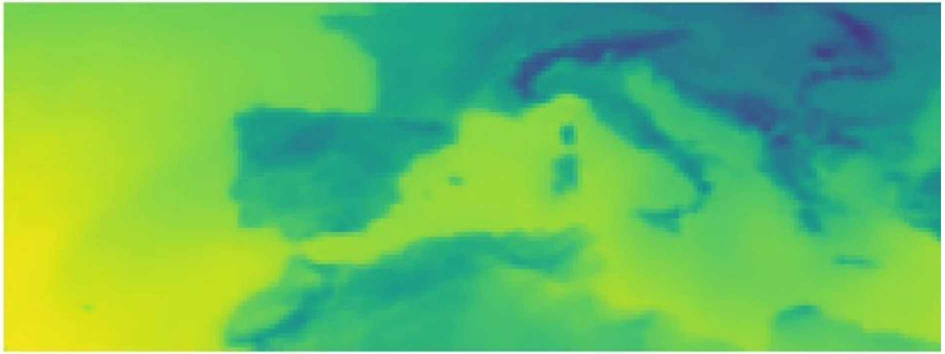
- Activations can be chosen (RELU by default)
- Adam optimizer
- Spatial dropout (0.2)
- Losses can be chosen
- Example of a learning curve for a supervised CNN with a MAE loss:



Super resolved temperature

- Examples from the ERA5 holdout dataset (samples not used during training)
- The animation shows the LR (resized) groundtruth vs the CNN reconstructed HR images

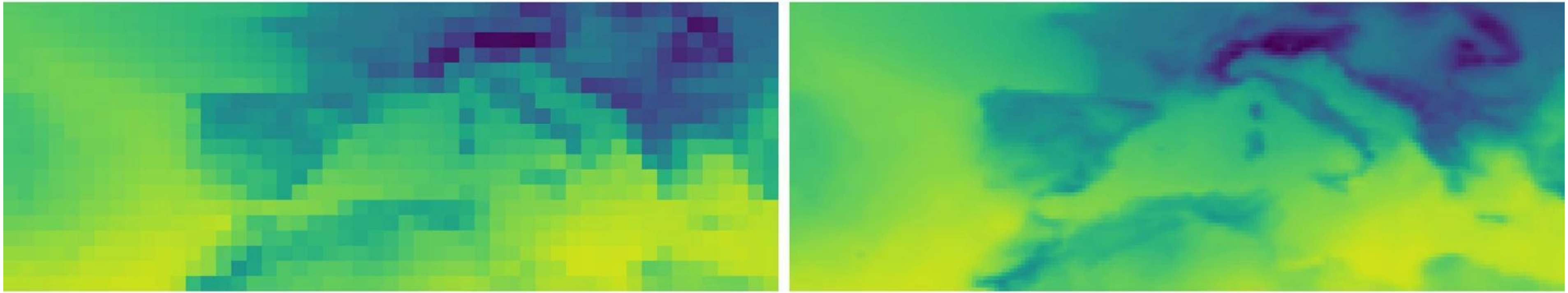




Application to seasonal forecasts

Inference on seasonal forecast data

- Examples from the SEAS5 seasonal forecast (not using during training)
- The animation shows the LR (original) SEAS5 vs the CNN predicted HR version

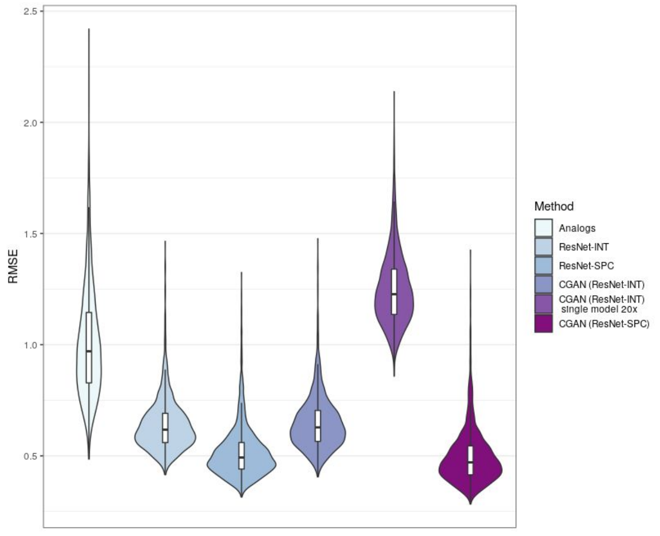


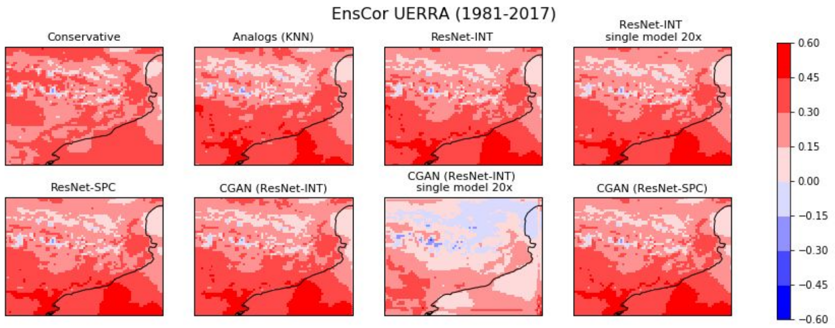
Inference on seasonal forecast data

Of course, visual inspection is not enough

Climate science metrics for forecast verification are important for evaluating if the super-resolved seasonal forecast has degraded/improved skill

This is outside of the scope of this talk





Takeaway messages

- AI/ML has great potential in Earth sciences
- Python rocks!
- Scientific software is key in modern research
- Super-resolution techniques might help improve low resolution seasonal climate forecasts
- DL4DS is part of a publication in preparation and has not yet been open sourced

Thanks!

Acknowledgements:

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement H2020-MSCA-COFUND-2016-754433 and the EU H2020 Framework Programme under grant agreements n° GA 823988 (ESiWACE-2), GA 869575 (FOCUS-Africa) and GA 869565 (VitiGEOSS).

carlos.gomez@bsc.es