UAQA: Urban Air Quality Assesment

Luca Catalano
Politecnico di Torino
Turin, Italy

Giacomo Rosso
Politecnico di Torino
Turin, Italy

Bruno Spaccavento
Politecnico di Torino
Turin, Italy

ABSTRACT

This paper extends the PRESTO architecture [13], a transformer-based model initially designed to analyze multimodal pixel time series composed from multi-source satellite imagery. The final goal is to forecast pollutant concentrations using these pixel time series which convey different types of measurements (weather, temperature, other pollutant concentrations, land characteristics, etc.). Through self-supervised learning, mimicking Masked-Language pretraining strategies, PRESTO efficiently captures spatio-temporal patterns, generating compact embedding for pixel time series. Leveraging these embedding a downstream task of pixel time series forecasting is developed. With a regression head built upon an MLP regressor, this model achieves accurate predictions. PRESTO's adaptability to multi-source and multimodal learning makes it a promising solution for efficient and versatile feature extraction on pixel time series.

ACM Reference Format:
Luca Catalano, Giacomo Rosso, and Bruno Spaccavento. 2018. UAQA: Urban Air Quality Assesment. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The utilization of machine learning alongside extensive remote sensing datasets has led to substantial societal benefits in various domains. Applications may range from monitoring progress on sustainable development goals to enhancing weather or pollutants forecasting, from bolstering disaster management capabilities to improving smart city design.

Machine learning models tailored for remote sensing applications face challenges in handling multimodal and multi-source data, particularly in scenarios where labelled datasets are scarce. The main challenges in this field are:

- **Highly multimodal and multi-source data.**
- **A highly informative temporal dimension.**
- **Missing data handling.**
- **Unavailability of labelled datasets.**

These scenarios have brought the research towards self-supervised techniques and the models that are best suited for this kind of task are considered to be transformer-like architectures[14].

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

This paper introduces a novel approach that combines a customized transformer architecture named PRESTO[13] for feature extraction and an MLP regressor for forecasting. The proposed method is designed to address the unique characteristics of remote sensing data, including temporal dynamics and information collected from diverse sensors. The main contribution may be summarized as follows:

- **PRESTO extension** to different data sources (multisource model) and to a specific finer time granularity (daily concerning the PRESTO monthly granularity).
- **Preprocess pipeline** to align both spatially and temporally different sources.
- **Generalization of PRESTO pretrain strategy** to encompass different sources and strategies.
- **Forecasting pipeline** using the PRESTO encoder and an MLP regressor.
- **Label weighting** to effectively train the model using both golden and synthetic labels.

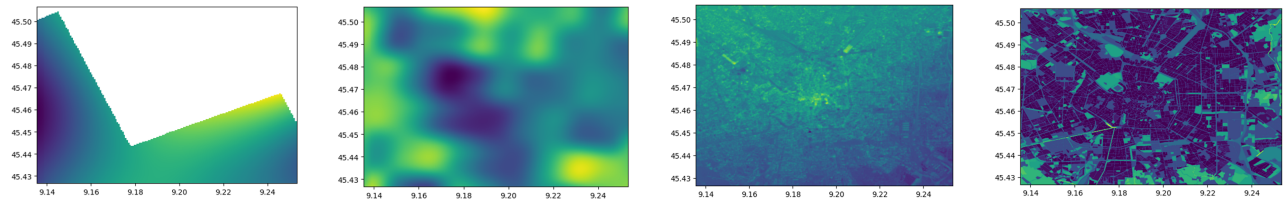
Specifically, our work focused on the Urban area of Milan and we used both different satellites as time-dynamic data sources and some time-static measurements for the pretraining of PRESTO[13]. For the downstream forecasting task, certain measurements obtained from monitoring stations in Milan served as the ground truth, while additional synthetic labels were created to augment the training dataset.

2 RELATED WORKS

One key aspect in the design of machine learning algorithms for remote sensing is the consideration of the characteristics inherent to remote sensing data. These characteristics encompass the highly multimodal nature of the data, stemming from a multitude of Earth-observing satellites equipped with diverse sensors. Additionally, the temporal dimension holds paramount importance due to the dynamic nature of Earth's landscapes, necessitating approaches that emphasize pixel time series modelling.

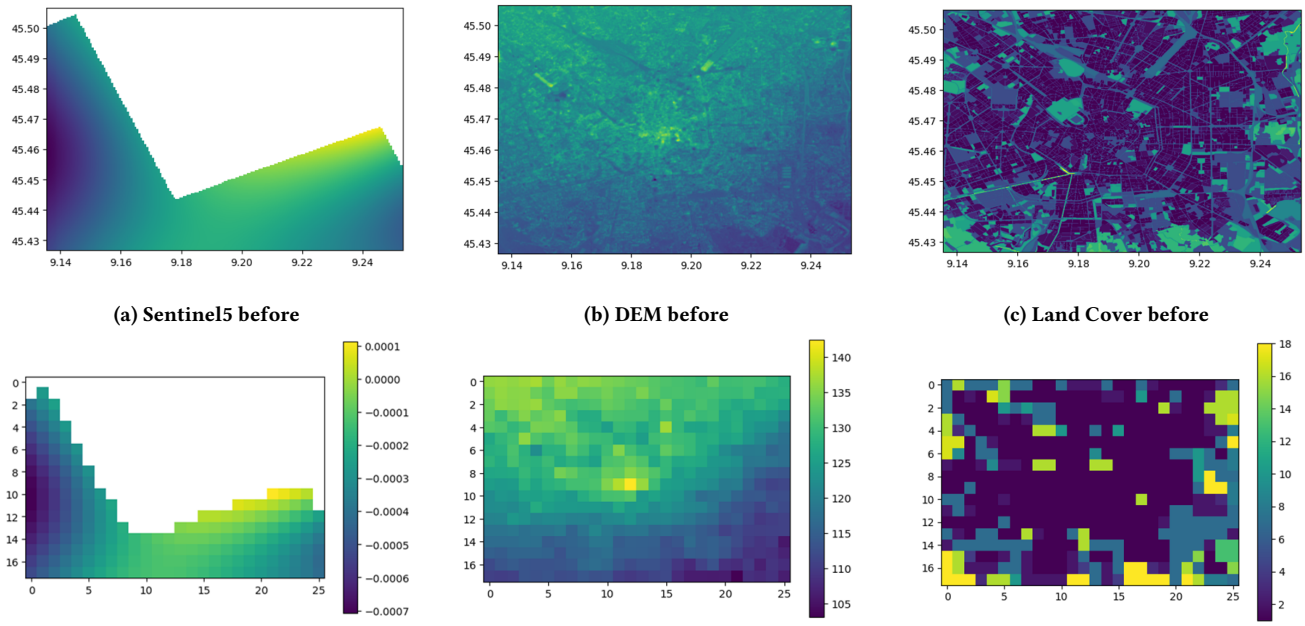
Furthermore, the unique metadata associated with remote sensing data, specifically location data and timestamps, has proven instrumental in augmenting machine learning algorithms [15]. Challenges persist in this domain, including limited availability and reliability of labelled datasets particularly in under-resourced geographies [2, 6, 8, 10], necessitating exploration into self-supervised learning strategies [1, 3, 7, 9, 11].

Prior research has predominantly treated remote sensing data to natural imagery, utilizing methodologies and architectures originally designed for ground-level photography, for example, by using a ResNet backbone [5], or by adapting masked autoencoding for image classification to satellite imagery [3, 11]. However, this



(a) Sentinel5 (b) Sentinel3 (c) DEM (d) Land Cover

Figure 1: Examples of some bands measured by data sources. Some bands have lots of nan values (1a), some sources are dynamic, with one or more measurements per day (1a, 1b), and some others are static in time, the latter measurements are valid for every considered day (1c, 1d).



(a) Sentinel5 before (b) DEM before (c) Land Cover before

(d) Sentinel5 after (e) DEM after (f) Land Cover after

Figure 2: Data sources spatially aligned: from each data source's per pixel resolution to a common shape of *(FINAL H, FINAL W)* that corresponds to a per-pixel resolution of 500m.

approach has often overlooked the holistic exploitation of all attributes inherent in remote sensing data, such as the incorporation of diverse sensors and the temporal dimension.

Notably, recent efforts have introduced models like PRESTO, a lightweight transformer-based architecture tailored to efficiently process multi-sensor pixel time series inputs, showcasing competitive performance across diverse geographies and task types [13]. Additionally, Presto demonstrates robustness in scenarios with missing input channels or temporal information, underscoring the significance of leveraging multi-sensor time series in its training process.

3 METHOD

3.1 Pretraining Data Sources

The time dynamic data sources used are all coming from satellite imagery in *.tiff* format. Therefore, each file contains a timestamp, one measurement for each quantity that the satellite has to measure (called *Band*) and its metadata (spatial resolution, temporal resolution, bounding box, etc.). The specific sources are:

- **Era**: several variables related to weather conditions measured daily at city level, with an 11 km resolution.
- **Sentinel3**: two daily measures of temperature with 1 km resolution.
- **Sentinel5**: several pollutants measured daily with resolution 5.5 x 3.5 km.

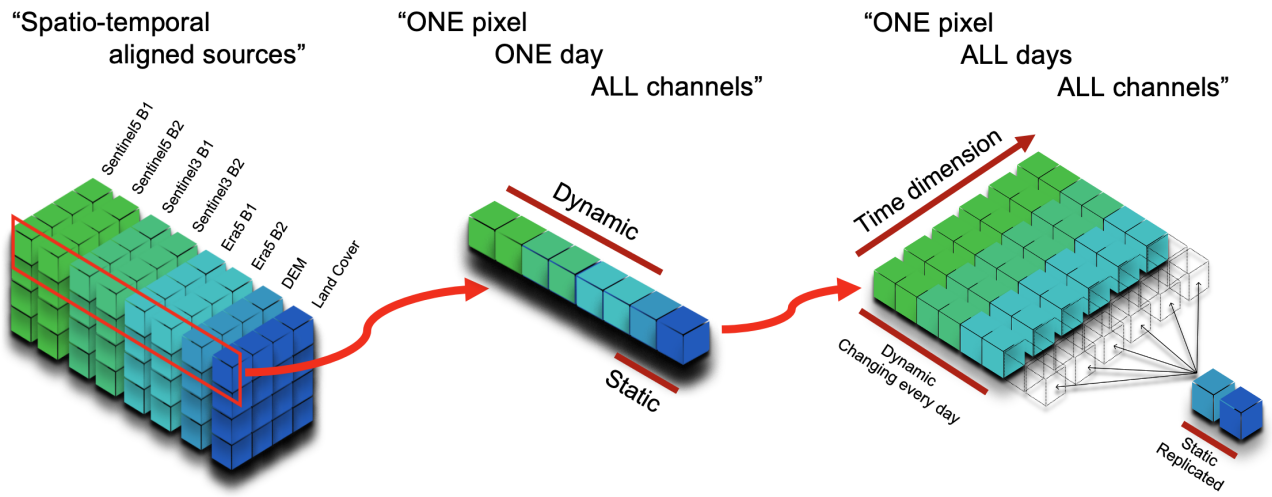


Figure 3: Pixel Time Series

The time static data sources used are also in *.tiff* format but consists of one single file that remains constant for the whole duration of the period analyzed. The specific sources are:

- **Dem**: Digital Elevation Model of the city with 10m resolution.
- **Land Cover** 10m resolution map of land use classification.

Given that the Land Cover is the only data source whose values are not to be considered real values but categorical (it is a classification/taxonomy of the geographical area where each pixel has a value that represents one specific class of the 27 available classes) we need to perform a manipulation of the data:

Initial LC matrix := [*height, width*] each value $\in \{1, ..., 27\}$

Final LC matrix := [*height, width, 27*] each value $\in \{0, 1\}$

What is happening is that we simulate one hot encoding that is performed along the third dimension. This procedure is also used because of the resizing of the images we will do. Averaging the class values would have had no meaning, instead with this one hot representation we will have for each class a value in $[0, 1]$ that represents the fraction of pixels that have been averaged that belong to that class.

3.2 Finetuning additional Data Sources

- **Golden Truth**: Ground truth punctual measurements of pollutants.
- **Synthetic Labels**: .tiff file generated starting from the punctual measurements through an *XGBoost* model.

3.3 Data structures

To work with pixel time series built upon this highly multimodal data two main problems arise: missing data and different resolution, both in spatial and temporal domain. This means that some satellites make multiple measurements per day (Sentinel3) whereas others make only one (Sentinel5) and the measurements differ in spatial resolution from each other. Hence, it is possible that even if the *bounding box* is the same, the pixels among the sources may represent a different portion of the geographical area. Moreover, the delicate sensors provide lots of missing values even if the daily measurement is available. The solutions to these problems that we decided to implement are:

- **Spatial alignment**: all the sources are aligned to a final shape *(FINAL H, FINAL W)* that corresponds to a resolution of 500 m per pixel. This means that when an image has a finer resolution, groups of adjacent pixels are averaged. Whereas if the initial resolution is courser the final image is padded and interpolated to reach the final resolution.
- **Temporal alignment**: all the sources are brought to the minimum daily frequency, in our case one measurement per day is considered for each source.
- **Spatial missing data**: when one of the sources is missing a measurement of one of its bands, the relative nan value is filled with the average (computed along the time dimension) of that band.
- **Temporal missing data**: when one of the sources is missing an entire day, synthetic data is generated to cover that day. The values correspond to the average along the whole time dimension of the measurements for each band.
- **Temporal static data**: when we consider static sources the same value is used for each timestamp.

Once the first preprocessing step is completed we have for each source exactly one daily measurement for each band with the same per-pixel resolution. This means that all the sources are now spatio-temporal aligned.

The second step is the generation of the pixel time series. Once the sources are aligned, we retrieve for each pixel all the available

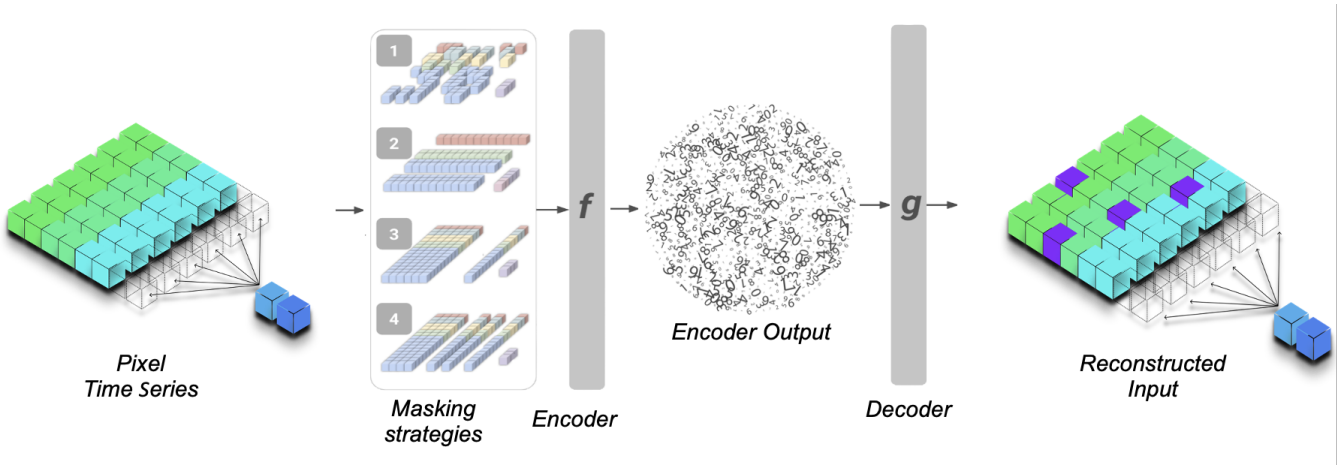


Figure 4: Pretrain model

daily measurements. At this point, a pixel time series appears as follows:

$$PixelTimeSeries := [NumDays, NumBands]$$

where *NumBands* is the concatenation of all the measured quantities by the sources. Notice that we will have several Pixel Time Series equal to the final number of pixels which is *FINAL H * FINAL W*.

We decided to generate a dataset of pixel time series. Each element of the dataset will be a segment of a pixel time series, each pixel corresponds to a specific geographic area that is characterized by its latitude and longitude. Moreover, each segment will be associated with a starting day and a duration, the generated segments may or may not overlap depending on how the rolling window over the pixel time series is parameterized. To further characterize the daily input, specifically with information about the seasonality trends and possible periodic behaviours, we decided to use information about *day of week* and *day of year*. So each element of the dataset we built is composed as follows:

- **Pixel Time Series**: this is the series of concatenated measurements of a contiguous period of length *Num Timesteps*.

$$PixelTimeSeries := [NumTimeSteps, NumBands]$$

- **Hard Mask**: this contains information of where and when the data inside the time series is not original data but it has been substituted by the band average (1 if the original data is unavailable, 0 otherwise).

$$HardMask := [NumTimeSteps, NumBands]$$

- **Lat Lon**: this contains the spatial information (latitude, longitude) of the pixel we are analyzing.

$$LatLon := [NumTimeSteps, 2]$$

- **Day of Year**: for each of the days contained in the pixel time series there is the corresponding day of year.

$$DayOfYear := [NumTimeSteps]$$

- **Day of week**: for each of the days contained in the pixel time series there is the corresponding day of week.

$$DayOfWeek := [NumTimeSteps]$$

3.4 Data structures for Finetuning

The task of forecasting implies that for each pixel time series, we have to provide a target value that is either a golden truth or, if not available, the synthetic labels of the following day. Namely, if we have a pixel time series that ends on *day t* (e.g. 7^{th} of January) we provide the target values of *day t+1* (e.g. 8^{th} of January). Moreover, this dataset will also provide for each element a loss factor that will be used to weigh the loss based on how much we trust the provided target value. This loss factor will be 1 if we use the *Golden Truth* or it will decrease depending on how far we are from the nearest stations that measure a *Golden Truth*. Hence, the element of the dataset will be all the previous ones with two new information:

- **Labels**: a list of target values for a specific day and a specific pixel.

$$Labels := [NumPredictions]$$

- **Loss Factors**: how confident we are of the labels, inversely proportional w.r.t. the geographical distance to the nearest golden label. This factor is in the range $[0.3, 1]$.

$$LossFactors := [NumPredictions]$$

3.5 Model

The PRESTO[13] architecture is a transformer-like model that is based on an encoder-decoder structure[14]. The peculiar behaviours are in how the initial embedding is provided to the encoder.

Each pixel time series:

$$x := [NumTimesteps, NumBands]$$

that enters in the model is transformed into many tokens (each represented by an embedding e) to be processed by the PRESTO transformer[13].

For each timestep, $0 \leq i \leq NumTimesteps$, the input *NumBands* are split into C channel groups according to their type of sensor or

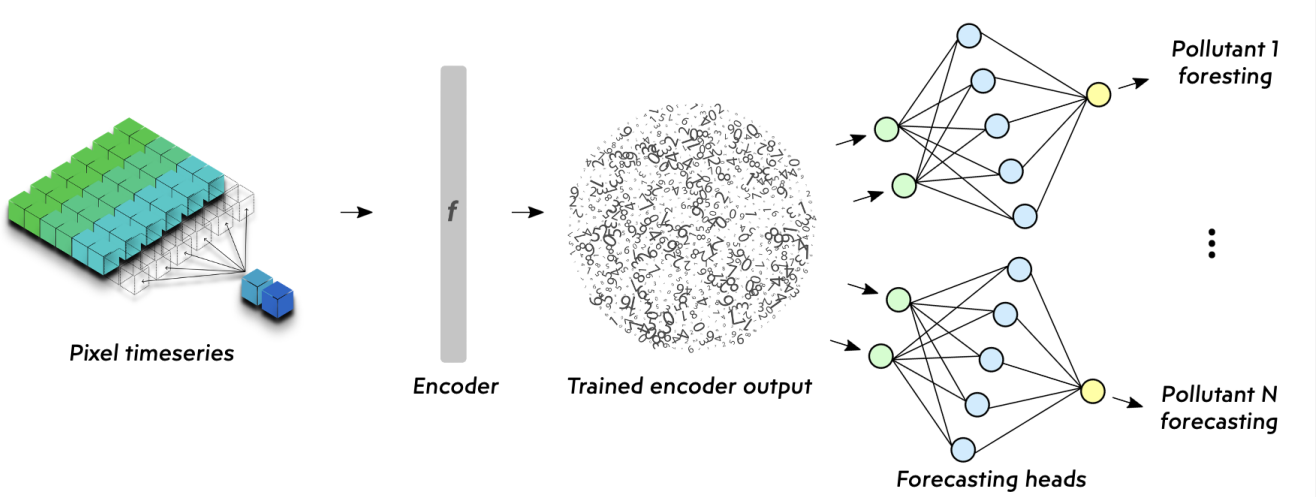


Figure 5: Forecasting model

source (each data source corresponds to a channel group). We then project each of them to a common latent space of dimension d_e by separate learned linear projections. So we end up with a sequence of tokens like:

$$x := [NumTimesteps * NumChannelGroups, d_e]$$

To the representation of each token (one per channel group) is further added a series of embedding:

- **Positional embedding**: the classical embedding used in transformers to characterize the position of an element within a sequence, obtained through the sampling of sinusoidal functions.
- **Channel Group Embedding**: an embedding to add to represent the band group.
- **Lat Lon Embedding**: an embedding to inject geospatial awareness into the model.
- **Day of Year Embedding**: an embedding to inject temporal awareness with respect to the year.
- **Day of Week Embedding**: an embedding to inject temporal awareness with respect to the week.

This representation is then fed into the well-known encoder-decoder structure.

3.6 Finetuning Model

The model used for the forecasting task is built upon a pretrained PRESTO architecture. Once Presto has been pretrained on the dataset its encoder is used as a feature extractor for the pixel time series and the so-generated embedding becomes the input to multiple forecasting heads, one for each measure we want to predict. The final model can be depicted as follows:

- **PRESTO Encoder**: the encoder of presto takes a pixel time series and returns a vectorial representation of it.
- **Regression Heads**: for each one of the measures we want to predict an MLP regressor is built. This regressor takes as input the vectorial representation provided by PRESTO and outputs a value that is the prediction.

3.7 Train & Evaluation

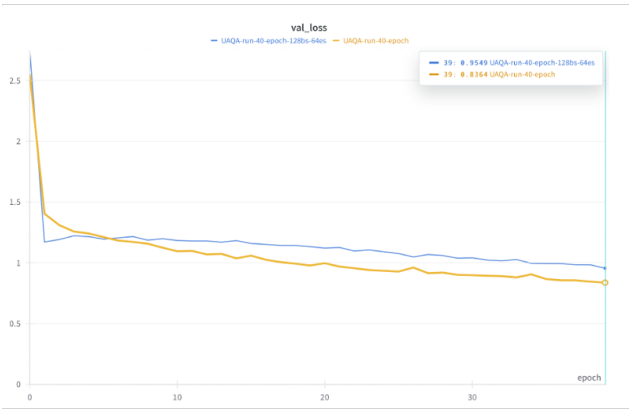
The initial architecture of PRESTO was specifically designed with a different time granularity (month instead of days) than what we needed. Therefore every pretrained checkpoint available was unfortunately not suitable for our task. Hence, we have to retrain from scratch our version of the model on the pixel time series aforementioned.

The training strategy is very similar to the one of Large Language Models like BERT[4] known as Masked Language. The strategy consists of masking some of the values of the input and letting the transformer reconstruct it. At each batch, one of the following masking strategies is selected at random.

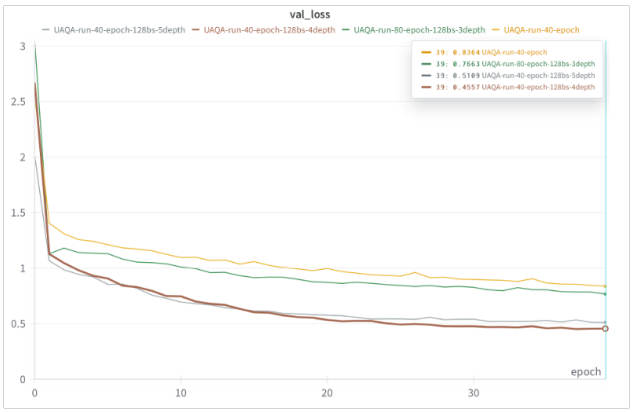
- **Group Bands**: select at random some band groups and mask them in all the pixel time series in the batch.
- **Random Timesteps**: select at random and mask some timestamps of all the pixel time series in the batch.
- **Chunk Timesteps**: select at random some contiguous number of timestamps of all the pixel time series in the batch.
- **Random Combinations**: select at random some values and mask them.

The training is done by minimizing the reconstruction error. This training strategy aims to let the encoder extract a representation that is informative enough to be able to reconstruct the original input with the decoder.

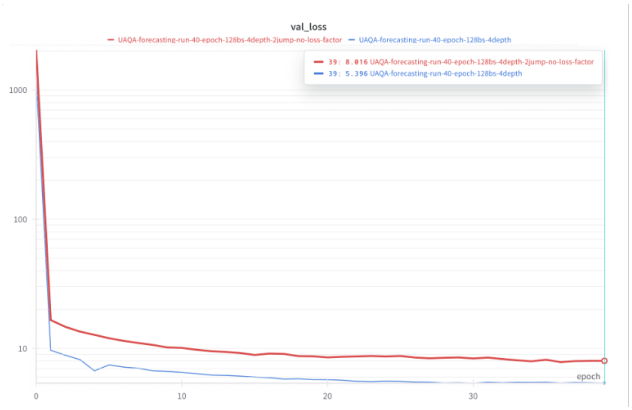
In our setting, the values of the input are not masked with a special token as done in Natural Language Processing, because we have no counterpart of [MASK] token available for real numbers (all numbers bring an intrinsic meaning). What happens is that the selected values to be masked are replaced with a value 0 before the encoding and the optimization strategy is the minimization of the Mean Square Error between the original value and the output of



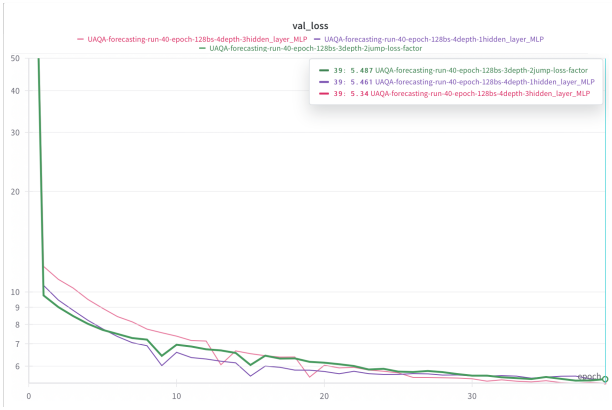
(a) Embedding Size comparison



(b) Encoder/Decoder depth comparison



(a) With and without loss factor comparison



(b) number of MLP hidden layers comparison

the model of that value, which corresponds to the reconstructed value by the model.

Moreover, to avoid the possibility that the encoder-decoder concentrates too much on the static data (which are equal for each element of the dataset and this could lead to learning an identity transformation), we decided to apply a training strategy that masks only the dynamic data and therefore train the model using a reconstruction loss (*MSE loss*[12]) that considers only the reconstruction of the dynamic data. The evaluation error is instead computed on all the data because at inference time we are also interested in understanding whether the overall reconstruction is reliable.

$$\textbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

The training of the forecasting model is done by minimizing the Mean Square Error weighted by the loss factors of each batch element for all the target values.

$$\textbf{MSE Weighted} = \frac{1}{n}\sum_{i=1}^{n} loss\ factor_i * (y_i - \hat{y}_i)^2 \tag{2}$$

4 EXPERIMENTS

For all the experiments we used the measurements available –from *2018-04-30* to *2022-12-31* for the train and validation split, whereas for the test we used data never seen by the model, specifically from *2023-01-01* to *2023-11-22*. All the visualized losses will be on the validation part of the dataset.

All the experiments are optimized with Adam on the MSEloss with default parameters.

The specifics of the models used forced us to first pretrain PRESTO and find an architecture that best suited the Masked Language Strategy, and only then train the forecasting model with the pretrained PRESTO encoder. During the fine-tuning, the encoder of PRESTO is not frozen and therefore updated through backpropagation.

The first group of experiments was designed to find the best configuration possible for the encoder-decoder structure. We worked with different possible values of *embedding size* and *encoder/decoder depth* (number of attention blocks in a cascade structure) in table 1.

We have found that the best performing architecture on the pretraining task is a PRESTO with the following :

- **Embedding Size** = 128
- **Encoder/Decoder Depth** = 4

Hyperparameter	Tested values
embedding size	{64, **128**}
encoder/decoder depth	{2, 3, **4**, 5}

Table 1: Table of pretraining hyperparameters.

The second group of experiments was designed to find the best configuration possible for the MLP architecture structure. We first checked if the loss factor was useful or not as a strategy. Moreover, we worked with different possible values of *MLP hidden layer number* and used the PReLU as activation function. The table 2 shows the tested values.

Hyperparameter	Tested values
with loss factor	{**True**, False}
MLP hidden layers	{1, 2, **3**}

Table 2: Table of finetuning hyperparameters.

We have found that the best performing architecture on the forecasting task is a model with:

- **Loss factor** = True
- **MLP hidden layer number** = 3

As a final comparison, we have used a plain transformer to predict the same set of pollutant concentrations. This model was not trained or built by us but had the useful function of a benchmark to compare our results with. This model was the previous and best performing approach used to solve the same task by our colleagues and mentors at LINKS Foundation. In Table 3 all the pollutant predictions on the test dataset yield indeed a lower MAE than the predictions of the old model.

POLLUTANT	OUR PRESTO	BASELINE
PM10	**4.87**	7.99
PM25	**3.95**	5.11
O3	**14.86**	15.35
NO2	**10.6**	13.63
CO	**0.12**	4.67
SO2	**0.34**	1.35

Table 3: Final comparison between pollutant forecasting error between our model and a baseline provided by LINKS Foundation

5 CONCLUSION

As introduced at the beginning the great improvement brought by the PRESTO architecture seems to be the ability to use domain knowledge to enrich the initial embedding of the pixel time series fed to the architecture. The further development of a finer time granularity and the extensions to different multimodal sources has been successful and the results are promising. This kind of model seems to be one of the most flexible and adaptive architectures in the landscapes of pixel time series analysis of satellite imagery. Further improvement would require in our opinion the extension to an attention mechanism that is not only computed in the time dimension (as per the actual transformer) but also in the geographical one. The positional embedding of the *lat lon* may be insufficient to fully exploit the geographical information. Finally, our work is open-source and available at the following link: https://github.com/LucaCatalano13/UAQA

REFERENCES

[1] Kumar Ayush, Burak Uzkent, Chenlin Meng, Kumar Tanmay, Marshall Burke, David Lobell, and Stefano Ermon. Geography-aware self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10181–10190, 2021.

[2] Patrik Olã Bressan, José Marcato Junior, José Augusto Correa Martins, Diogo Nunes Gonçalves, Daniel Matte Freitas, Lucas Prado Osco, Jonathan de Andrade Silva, Zhipeng Luo, Jonathan Li, Raymundo Cordero Garcia, et al. Semantic segmentation with labeling uncertainty and class imbalance. 2021.

[3] Yezhen Cong, Samar Khanna, Chenlin Meng, Patrick Liu, Erik Rozi, Yutong He, Marshall Burke, David Lobell, and Stefano Ermon. Satmae: Pre-training transformers for temporal and multi-spectral satellite imagery. *Advances in Neural Information Processing Systems*, 35:197–211, 2022.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[7] Neal Jean, Sherrie Wang, Anshul Samar, George Azzari, David Lobell, and Stefano Ermon. Tile2vec: Unsupervised representation learning for spatially distributed data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3967–3974, 2019.

[8] Hannah Kerner, Gabriel Tseng, Inbal Becker-Reshef, Catherine Nakalembe, Brian Barker, Blake Munshell, Madhava Paliyam, and Mehdi Hosseini. Rapid response crop maps in data sparse regions. *arXiv preprint arXiv:2006.16866*, 2020.

[9] Oscar Manas, Alexandre Lacoste, Xavier Giró-i Nieto, David Vazquez, and Pau Rodriguez. Seasonal contrast: Unsupervised pre-training from uncurated remote sensing data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9414–9423, 2021.

[10] Catherine Nakalembe, Christina Justice, Hannah Kerner, Christopher Justice, and Inbal Becker-Reshef. Sowing seeds of food security in africa. *Eos (Washington. DC)*, 102, 2021.

[11] Colorado J Reed, Ritwik Gupta, Shufan Li, Sarah Brockman, Christopher Funk, Brian Clipp, Kurt Keutzer, Salvatore Candido, Matt Uyttendaele, and Trevor Darrell. Scale-mae: A scale-aware masked autoencoder for multiscale geospatial representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4088–4099, 2023.

[12] Jiawei Ren, Mingyuan Zhang, Cunjun Yu, and Ziwei Liu. Balanced mse for imbalanced visual regression, 2022.

[13] Gabriel Tseng, Ruben Cartuyvels, Ivan Zvonkov, Mirali Purohit, David Rolnick, and Hannah Kerner. Lightweight, pre-trained transformers for remote sensing timeseries.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] Jiaxuan You, Xiaocheng Li, Melvin Low, David Lobell, and Stefano Ermon. Deep gaussian process for crop yield prediction based on remote sensing data. 31(1), 2017.