POLITECNICO DI TORINO

MASTER's Degree in DATA SCIENCE AND ENGINEERING

MASTER's Degree Thesis

A Transformer-based approach to air quality prediction in Milan through satellite imagery combined with meteorological and morphological data

Supervisors

Prof. Giuseppe RIZZO

Dott. Giacomo BLANCO

Candidate

Luca CATALANO

Company Supervisors

Dott. Luca BARCO

Dott. Lorenzo INNOCENTI

A.A 2023/2024

Summary

Air pollution is one of the most pressing problems of our time, causing serious issues for human health and the environment. It is a key factor in the development of respiratory ailments, ranging from mild conditions like asthma to more severe complications such as decreased lung function, cardiovascular diseases, and premature mortality. Additionally, it has serious environmental repercussions, such as ecosystem degradation, biodiversity loss, and adverse effects on plant growth and agricultural productivity.

To better understand and mitigate this problem, remote sensing technologies have been adopted to monitor large areas, facilitating comprehensive air quality assessments. For example, satellite data from the Copernicus Sentinel-5p mission provide valuable information and are useful for forecasting pollutants.

In this study, Copernicus Sentinel-3 and Sentinel-5p data combined with other meteorological and morphological data were used to predict the air quality in the city of Milan, focusing on five pollutants: ozone (O3), nitrogen dioxide (NO2), sulfur dioxide (SO2), particulate matter with a diameter of less than 10 micrometres (PM10), and less than 2.5 micrometres (PM2.5).

Deep learning models, such as CNN, RNN, and LSTM, have demonstrated considerable success in analyzing such data due to their ability to uncover complex relationships. Expanding on this success, this study explores the adaptation of TimeSformer, a transformer-based model initially designed for video classification, to work on satellite data for forecasting tasks. The primary modification is the initial encoding, which is designed to accommodate diverse sources, reflecting the heterogeneous nature of the dataset. The developed model is capable of forecasting all pollutants simultaneously, creating a forecasting mechanism that could be of significant value to policymakers, as it enables them to implement timely protection and prevention measures. It achieved a Mean Absolute Percentage Error of 30.6%, improving the prediction accuracy for three out of the five pollutants compared to a baseline model.

A mamma e papà

Table of Contents

List of Tables

List of Figures

Acronyms

EEA
European Environment Agency

WHO
World Health Organization

ESA
European Space Agency

PM10
Particulate Matter with a diameter of 10 micrometres or less

PM2.5
Particulate Matter with a diameter of 2.5 micrometres or less

O3
Ozone

NO2
Nitrogen Dioxide

SO2
Sulfur Dioxide

DEM
Digital Elevation Model

LC
Land Cover

SL
: Spatial Level

NaN
: Not a Number

MLP
: Multilayer perceptron

CNN
: Convolutional Neural Network

RNN
: Recurrent neural network

LSTM
: Long Short Term Memory

ViT
: Vision Transformer

MSE
: Mean squared error

MAE
: Mean absolute error

MAPE
: Mean absolute percentage error

Chapter 1

Introduction

The work presented in this thesis aims to study and analyze heterogeneous data sources to build a deep learning model for predicting air quality in the central area of Milan. In particular, satellite data obtained from the Copernicus missions and meteorological data have been used. The following chapter provides an overview of the pollution problem.

1.1 The problem of air quality

Pollution represents a significant global issue, as it has the potential to cause severe health problems and damage the surrounding environment. It can result in significant respiratory and cardiovascular complications for individuals and environmental degradation. Pollution can manifest in several different forms, including contamination of the air, water and soil. Air quality, which indicates the cleanliness of the air, is influenced by both natural and human activities. Key sources of air pollution include agricultural practices and natural disasters such as fires and volcanic eruptions, vehicle emissions and industrial processes. This is confirmed by a study [1] done in the city of Milan in 2020. The research reveals that since late March 2020, reductions in private transportation (-77%), heavy and light vehicles (-39%), and industrial production (-20%) resulted in a one-third decrease in NO2 levels.

1.2 Principal pollutants

This work analyzes five pollutants, which are monitored by the European Environment Agency (EEA) to assess air quality and ensure public health and environmental protection across Europe. These are Nitrogen Dioxide (NO_2), Sulfur Dioxide (SO_2), Particulate Matter of at most 10 micrometres in diameter (PM10), Fine Particulate

Matter of at most 2.5 micrometres in diameter (PM2.5), and Ozone (O_3). Each of these pollutants has unique characteristics and sources, yet together they constitute a significant problem to both human health and environmental integrity.

1. **Nitrogen Dioxide (NO_2):** It is a red-brown gas at ordinary temperature with a suffocating, irritating odour that is particularly humid and characteristic. NO_2 is denser than air, so its vapours tend to remain at ground level. It is a strong irritant of the pulmonary tract; even at moderate concentrations in the air, it causes coughing, chest pain, convulsions and circulatory failure. It can also cause irreversible damage to the lungs, which can occur many months after the attack.

2. **Sulfur Dioxide (SO_2):** SO_2 is a toxic gas, that is released naturally by volcanic activity and is produced as a by-product of copper extraction and the burning of sulfur-bearing fossil fuels. Inhaling SO_2 can cause respiratory issues, particularly in humans affected by asthma or other respiratory ills. It also contributes to the environment's formation of fine particulate matter and acid rain.

3. **Particulate Matter (PM_{10} and $PM_{2.5}$):** Particulate matter is one of the most frequent pollutants in urban areas. PM_{10} refers to particles with a diameter of 10 micrometres or less, while $PM_{2.5}$ refers to particles with a diameter of 2.5 micrometres or less. Examples of substances present in particulate matter are natural and artificial fibres, pollen, spores, carbonaceous particles, metals, silica and liquid pollutants. Particulate matter can be found both in open and closed places, but generally, its concentration is higher in closed places and urban and industrial areas. Particulate matter is dangerous to the health of humans and other living beings. The World Health Organization (WHO) has classified particulate matter as carcinogenic, i.e. capable of causing tumours or promoting their onset and propagation.

4. **Ozone (O_3):** Ground-level O3 is a byproduct of fossil fuel and coal burning. It is formed when nitrogen oxide and volatile organic compounds (VOC) released from internal combustion engines and fossil fuel power plants interact with oxygen and UV rays. As ozone pollution sources come from fossil fuel automobiles, it is commonly found in urban city environments and densely populated areas. It is also more likely to occur during sunny, high-temperature conditions as well as periods of high air pressure, where air becomes stagnant and pollutants are more concentrated over certain areas. Ozone pollution poses great harm to human health and increases the risks of respiratory illnesses such as asthma and lung cancer. There has also been some limited evidence showing that repeated, long-term exposures to O_3 can inhibit the growth of lung tissue in children and worsen cardiovascular diseases such as atherosclerosis.

Table 1.1 illustrates the classification of pollutant concentrations as defined by the European Environment Agency (EEA). It presents the concentration ranges in micrograms per cubic meter (μg/m^3) for different air quality categories, providing a clear framework for understanding the levels of pollution and their potential impacts on health and the environment.

Pollutant	**Good**	**Fair**	**Moderate**	**Poor**	**Very Poor**	**Extremely Poor**
PM2.5	0-10	10-20	20-25	25-50	50-75	75-800
PM10	0-20	20-40	40-50	50-100	100-150	150-1200
NO_2	0-40	40-90	90-120	120-230	230-340	340-1000
O_3	0-50	50-100	100-130	130-240	240-380	380-800
SO_2	0-100	100-200	200-350	350-500	500-750	750-1250

Table 1.1: Classification of pollutant concentrations according to the European Environment Agency (EEA). The table displays the concentration ranges in $\mu g/m^3$ for various air quality categories.

1.3 The importance of monitoring air quality

The awareness of the detrimental effects of pollution on the environment and human health has compelled the authorities to take a more serious look at the environment and to encourage a desire to learn more about the quality of air that surrounds us. To control atmospheric pollutant levels, agencies like the European Environment Agency (EEA) have established air quality directives that set thresholds countries must not exceed on a yearly, daily, and hourly basis [2]. Over the years, new technologies have been developed to analyze air quality. For instance, the European Commission and the European Space Agency (ESA) created the Copernicus program to utilize satellite data for air quality monitoring. This program provides extensive coverage and can effectively oversee vast regions. On the other hand, ground-based sensors, which collect non-satellite data, are restricted to particular locations and can be expensive and complicated to set up and sustain. While satellite imagery can address certain issues associated with managing local stations and covered areas, it also has some disadvantages. First of all the presence of missing values, which often occurs due to the presence of clouds that cover the satellite's view of the Earth's surface. This limitation is particularly problematic because it generates gaps in the data, making it difficult to manage in the data pre-processing phase. Secondly, the resolution of satellite data is low, avoiding the ability to make good forecasts of pollutant concentrations at the ground level.

The scope of this work is to develop an accurate and high-resolution air quality model. The significance lies in delivering more realistic air quality predictions with specific spatial and temporal resolutions, which are essential for public health, regulatory compliance, and environmental protection.

The thesis is organized in the following way:

- **Chapter 2** shows previous works in the fields of pollutant forecasting and computer vision models used for natural and satellite images.
- **Chapter 3** explains the datasets adopted and the construction of the timeseries used in the model.
- **Chapter 4** presents the two methodologies applied to solve the problem.
- **Chapter 5** reports the results obtained using the developed methodologies.
- **Chapter 6** summarizes the work done in the thesis and explores potential future works.

Chapter 2

Related Works

Air quality monitoring is a field that has been widely studied over the years. In the beginning, researchers tried to analyze the quality of the air and the environment with statistical models; then, with the invention of Convolutional Neural Networks (CNN), they moved on to Deep Learning models, which are a promising alternative. This chapter provides a comprehensive overview of the methodologies that influence current approaches in this field. The chapter is divided into 3 sections, which although seem unrelated to each other, all contributed to the realization of this research. The first one introduces various models developed for predicting air pollution. This research highlights an unexplored area, noting that existing literature lacks models that incorporate diverse collections of satellite data or those that predict multiple pollutants simultaneously. This study aims to address these gaps and advance the field. The second section, however, focuses on related works involving Vision Transformers (ViTs), examining their innovative uses and potential in computer vision activities. There is also a small introduction to the TimeSformer, which is the embryonic model of this research. The final section explores the deep learning models used in Earth observation activities in tasks other than forecasting, which show how to preprocess and use satellite data.

2.1 Approaches for air pollutant concentrations: from classical methods to deep learning techniques

A lot of studies have been devoted to the development of different models to predict air pollutant concentrations. The existing methods can be classified into three categories: statistical, machine learning and deep learning methods.

Machine learning methods comprise deterministic and statistical models. Stirnberg et al. [3] present a statistical model developed to predict hourly concentrations of PM10. Their model integrates satellite-borne aerosol optical depth (AOD) data with various meteorological and land-use parameters, recognizing the significant impact of weather conditions on air quality. This approach uses the Gradient Boosting Regressor Tree (GBRT), which combines multiple decision trees using boosting and gradient descent techniques. Four distinct seasonal models are analyzed to account for seasonal variations in air quality, utilizing a comprehensive dataset spanning seven years, where temporal features are particularly relevant.

In this field, some classical regression-based algorithms, such as Multiple Linear Regression (MLR) and Auto-Regressive Integrated Moving Average (ARIMA), including its seasonal variant SARIMA, are extremely used. Hong et al. [4] present a study conducted in Labuan, Malaysia, that compares exponential triple smoothing (ETS) and seasonal autoregressive integrated moving average (SARIMA) models to analyze air pollutants data. The results indicate that SARIMA is the optimal model for forecasting PM10, NO_2 and O_3. On the other hand, algorithms like Support Vector Regression, Decision Trees, Random Forest, and K-Nearest Neighbors offer robust alternatives: they are able to capture non-linear features but may struggle to extract complex spatio-temporal correlations [5, 6, 7]. Regarding the deep learning models, Cabaneros et al. [8] review the use of Artificial Neural Networks (ANNs) for long-term prediction of outdoor pollutants, highlighting the predominance of meteorological and source emissions predictors in identified works.

In the literature, there are also several summaries of the most commonly used methods in the field of pollutant prediction. Bai et al. [9], for example, provides a comprehensive overview of statistical prediction, numerical prediction and hybrid models, along with a comparison of their advantages and disadvantages. On the other hand, Masih et al. [10] survey machine learning methods for air pollutant concentration prediction from 2013 to 2018, focusing on the fundamentals of machine learning techniques and their role in improving predictive performance. In addition, Liao et al. [11] provide a brief review of recent attempts to utilize deep learning methods in air pollution prediction, emphasizing their potential in exploring non-linear spatio-temporal correlations across multiple scales of air pollution. Later on, Masood and Ahmad [12] present an overview of artificial intelligence-based methods commonly used for air pollution prediction between 2003 and 2021.

A multimodal machine learning model is AQNet [13], which uses a new dataset that has information about altitude, population density, environmental classification of local areas, and satellite data. The realization of this dataset comes from imagery from Sentinel2, NO_2 concentration data from Sentinel5P, and tabular data. Finally, more recent deep learning technologies, such as long-short-term Short Term Memory models (LSTMs), excel at capturing non-linear spatio-temporal correlations in timeseries, thereby improving the accuracy of pollutant concentration prediction. These methods can effectively extract spatial and temporal correlations from raw data, enabling more accurate predictions of air pollutant concentrations [14, 15].

In addition, hybrid deep learning models, like Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM), integrate CNNs and LSTMs to extract better spatio-temporal correlations [16, 17, 18, 19]. A famous model is DAQFF [20], which uses a 1D-Convolutional Neural Network (CNN) with a Bidirectional Long Short-Term Memory (Bi-LSTM) network. The 1D-CNN extracts local features and spatial correlation from the data, while the Bi-LSTM component captures spatio-temporal dependencies. This algorithm is compared with traditional models, including the aforementioned ARIMA and Support Vector Regression (SVR), and in both situations, the new model outperformed the traditional ones.

2.2 Vision Transformers

Transformers, proposed by Vaswani et al. [21], are nowadays the best-performing models in NLP tasks. They introduce the attention mechanism, which allows the model to assign different weights to the various parts of the input to focus more on the most relevant information.

The idea of Dosovitskiy et al. [22] is therefore to look for a way to use the same methodology for images. The model created is called Vision Transformer (ViT). This model involves dividing an image into small, fixed blocks called patches, which are then transformed into feature vectors. Positional information is added to these vectors and provided to a Transformer encoder, which uses the attention mechanism to grasp the semantic relationships between the different patches. The relative importance of each patch is calculated by comparing it with all the other patches. This relationship is expressed through an array of numbers between 0 and 1. These coefficients are then used to generate a weighted representation of the characteristics of the image. Previously, Cordonnier et al. [23] proposed a similar model with the only difference of using 2x2 patches, which led to a performance result only on low and medium resolution images. Several studies also try to work on global self-attention, looking for tricks to avoid long training times by comparing each patch with all the others. Parmar et al. [24] restrict self-attention to local neighbourhoods, which allows local multi-head dot-product self-attention blocks to replace convolutions entirely [25, 26, 27]. Sparse Transformers [28] and other scalable self-attention models [29, 30] also aim to make global self-attention feasible for images. The use of these ViT models for images led to their application to videos. Models like Video Vision Transformer (ViViT) [31] and TimeSformer (Time-Space Transformer) [32] set new benchmarks. ViViT extends the ViT architecture to handle video inputs by factorizing the spatio-temporal dimensions, enabling efficient processing of video frames as sequences of patches [31]. Similarly, TimeSformer decomposes the attention mechanism into spatial and temporal components [32].

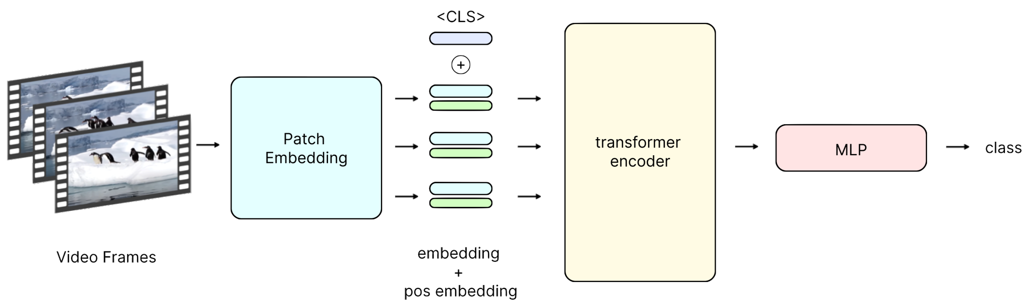

Figure 2.1: TimeSformer model architecture for the forecasting task. The model processes input video frames by first passing them through a patch embedding layer. The embedded patches are then fed into the encoder block. The final embeddings from the encoder are passed to a classification head for the forecasting output.

Figure 2.1, shows the architecture of the TimeSformer, which is the starting point for the analysis of this research. It takes as input a set of F RGB images ($X \in \mathbb{R}^{H \times W \times 3 \times F}$) of size $H \times W$ extracted from the original video with three-channel. Next, as presented in the ViT model [22], each frame is decomposed into N non-overlapping patches, each of size $P \times P$, such that the N patches cover the entire frame, i.e. $N = \frac{HW}{P^2}$. These patches are flattened into vectors $x(p,t) \in \mathbb{R}^{3P^2}$ and linearly mapped in an embedding thanks to a learnable matrix $E \in \mathbb{R}^{D \times 3P^2}$. At this point, a positional embedding $e_{\text{pos}} \in \mathbb{R}^{2 \times D}$ which represents spatio-temporal information is added to each patch (p,t). The embedding is therefore obtained from the equation

$$z(p,t) = Ex(p,t) + e_{\text{pos}}(p,t)$$

This embedding is then the input of the transformer, to which a learnable vector is added for classification tasks, called CLS token embedding. The Transformer consists of a series of encoding blocks, that calculate the multi-head self-attention. The final embedding of the clip is obtained from the final block for the classification token. Finally, a 1-hidden layer MLP is used to predict the classes of the final videos. In Figure 2.2, two types of transformer blocks are shown, which mainly differ in the way in which attention is calculated.

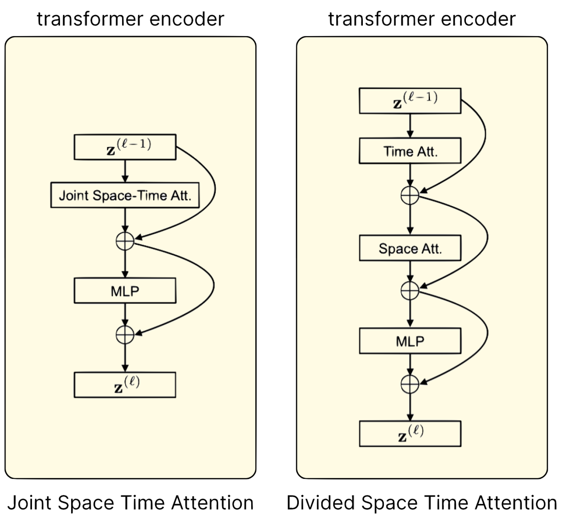

Figure 2.2: Transformer encoder block. It compares the joint space-time attention mechanisms with the divided space-time attention mechanism.

In the first case, called Joint Space-Time Attention, the output is obtained with one or more single attention heads for time and space. In this approach, the weighted sum of value vectors is calculated using the self-attention coefficients of each attention head. Next, the concatenation of these all-head vectors is projected and passed through an MLP, using the residual connections after each operation to maintain the integrity of the original information.

In the second case, the attention mechanism employed is the "Divided Space-Time Attention", which sequentially applies temporal and spatial attention. This method permits more scalable management of attention, thereby reducing the computational load associated with joint computation across spatial and temporal dimensions. In this case, the temporal attention is first computed by comparing each patch to all patches at the same spatial location in other frames. The resulting encoding is then fed back for the calculation of spatial attention, which compares each path with the other patches in the same image and is subsequently passed to the MLP.

Comparing these two types of attention mechanisms, the computational efficiency of the divided space-time attention mechanism is a significant advantage. In the joint space-time attention model, the number of per-patch comparisons is $(NF+1)$, where N is the number of patches per frame and F is the number of frames. This complexity arises because each patch attends to every other patch across all frames simultaneously. Conversely, in the divided attention model, the per-patch comparisons are significantly reduced to $(N+F+2)$. This reduction is achieved by first computing temporal attention across frames and then computing spatial attention within each frame.

2.3 Deep learning models in earth observation tasks

The use of deep learning models in satellite image analysis has gained considerable popularity in research, allowing such methods to be used for a variety of tasks, such as land cover classification, vegetation monitoring, urban development analysis, and disaster detection. This section aims to illustrate the most innovative approaches used in the field of satellite images, divided by the most popular models.

CNNs are widely used for their ability to effectively capture spatial hierarchies in data through convolutional operations. Geo-bench model [33] shows the ability of a Convolutional Neural Network (CNN) to outperform the accuracy for land cover classification and change detection tasks. For this final task, MSFCTNet [34] integrates a Convolutional Neural Network (CNN) with a transformer. This model consists of a Siamese CNN followed by a CNN-Transformer module that captures both global and local features. On the other hand, SEIFNET [35] presents a different approach. It employs a Siamese network based on ResNet18 [36] as its backbone to obtain multiscale feature maps, then it has the Spatio-Temporal Differential Enhancement Module (ST-DEM), which tries to extract contextual information, the Adaptive Content Fusion Module (ACFM) for contextual information fusion, and a refinement module (RM) to optimize boundary details. This architecture follows an encoder-decoder framework. The ST-DEM, part of the encoder, includes two branches: the Connection branch, which combines features along the channel dimension using convolution, batch normalization, and activation functions; and the Subtraction branch, which computes the absolute difference between bi-temporal feature maps. In the decoder, the ACFM refines multiscale targets through a three-step process: feature upsampling to match the resolution of low-level feature maps, weight calculation via a pooling and fusion process, and feature reintegration to combine complementary information. Finally, the RM further refines these features using the Convolutional Block Attention Module (CBAM). A different study [37], always related to CNN models-based, introduces how the techniques of resizing and normalization affect the performance on remote sensing tasks.

LSTM models, instead, are introduced to capture temporal dependencies in sequential data. Zhao et al. [38] evaluate the performance of LSTMs alongside 3D-CNNs and ViTs, demonstrating their effectiveness in temporal representation data.

After the discovery that ViT models can handle long-range dependencies and large-scale data, Yuan et al. [39] introduce a pre-trained ViT model specifically designed for Sentinel2 data and compare it against baseline models including Random Forest (RF), SITS-BERT, CNN-Transformer and Convolutional Recurrent Neural Networks (ConvRNN). In this case, the encoder takes as input timeseries of patches extracted from 3D-CNN, which are then merged with the other embeddings generated using a sin-positional encoding for the day of year information. These embeddings are then fed into the Transformer, which captures relevant spatial and temporal dependencies.

Another study made by Tarasiou et al. [40] illustrates how ViTs can be employed to process satellite image timeseries, leveraging their self-attention mechanisms to capture intricate spatio-temporal relationships. Fibaek et al. [41] provide a comprehensive evaluation of geospatial foundation models, including ViTs, reinforcing their potential in various geospatial analysis applications.

Discussing transformer-based models, one that gains particular success in the realm of satellite data is Presto [42], which is of particular interest for the work presented here. The architecture of Presto, a lightweight transformer-based model, is specifically designed for processing timeseries data from remote sensing pixels. The model employs a masked autoencoding framework consisting of an encoder and a decoder. During pre-training, part of the input data is masked, and the encoder processes the unmasked part, while the decoder attempts to reconstruct the masked portion from the encoded data (Figure 2.3). This pre-training strategy enables the model to learn robust representations from incomplete data, making it highly efficient for various downstream tasks.

Various masking strategies are employed to encourage the model to learn robust representations. These include:

- **Random Masking**: This involves randomly masking a portion of the input data.

- **Channel Group Masking**: Entire groups of channels are masked simultaneously. This type of masking simulates the absence of data from specific sensors or sources and helps the model generalize better when dealing with inputs from incomplete or faulty sources.

- **Contiguous Timestep Masking**: This strategy involves masking continuous temporal sequences rather than individual time points. In this way, the model learns to interpolate missing information over longer periods, improving its ability to handle temporal gaps in the input data.

- **Random Timestep Masking**: Similar to random masking but applied specifically to time steps, this technique randomly selects various temporal points to mask. This strategy helps the model become robust to incomplete or irregular temporal data.

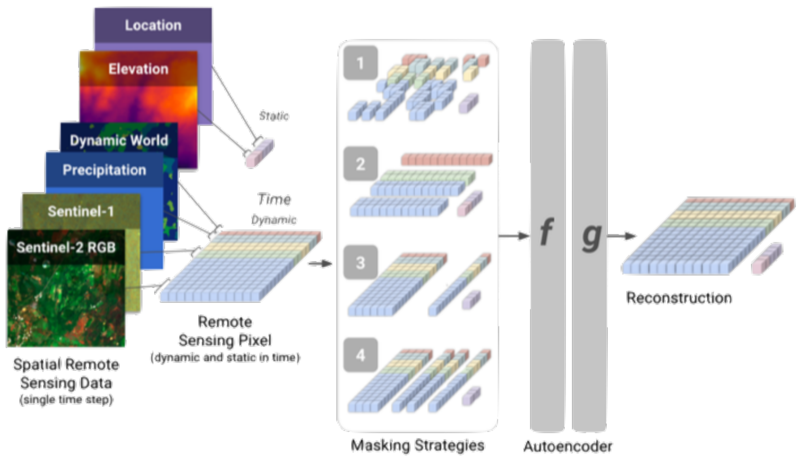

Figure 2.3: Presto transformer architecture [42]. The encoder-decoder model is trained to reconstruct the original timeseries. During fine-tuning, the decoder is discarded, and only the encoder's output is utilized.

The encoder, after pre-training, is used for fine-tuning or as a feature extractor. Presto is capable of handling both static and dynamic inputs over time, as well as metadata for each pixel timeseries.

The main components of Presto's architecture include:

Input Transformation

The input x is transformed into several tokens, each represented by an embedding e, to be subsequently processed by the transformer.

Channel Grouping and Projection

The input variables, divided into temporal segments, are grouped into channel groups C based on the type of sensor or source. These channel groups are projected into a common latent space of dimension d through separate learned linear projections h. Mathematically, this can be represented as:

$$h_{C_i} = W_{C_i} x_{C_i} + b_{C_i}$$

where x_{C_i} represents the input data for channel group C_i, and W_{C_i} and b_{C_i} are the learned weights and biases for channel group C_i.

Categorical Class Embedding

Categorical classes are embedded by indexing into an embedding matrix:

$$e_{cat} = \text{Embedding}(x_{cat})$$

where x_{cat} represents the categorical input.

Positional, Temporal, and Channel Encodings

Unlike natural images, where data and labels are standalone, remote sensing labels are intrinsically tied to a specific location and time on Earth, defined by latitude/longitude and timestamp. The architecture addresses this by adding encodings to the embeddings to convey the data point's location, timestamp, and channel group. The complete encoding is a concatenation of positional, monthly, and learned channel encodings, resulting in a dimension d_e.

- **Sinusoidal Positional Encoding**: it provides a unique representation for each position in the sequence.

$$p_s(p, 2i) = \sin\left(\frac{p}{10000^{2i/d_e}}\right)$$

$$p_s(p, 2i+1) = \cos\left(\frac{p}{10000^{2i/d_e}}\right)$$

  where p is the position index of the data point in the sequence, and i is the dimension index.

- **Monthly Encoding**: it helps the model to capture seasonal patterns by using monthly information.

$$p_m = \text{Embedding}(m)$$

  where m is the input representing the month.

- **Channel Encoding**: it distinguishes between different sensor or source types.

$$p_{ch} = W_{ch} \cdot C$$

  where W_{ch} are the learnable weights for the channel encoding, and C represents the channel group.

The transformer input $E \in \mathbb{R}^{(T\cdot|C_d|+|C_s|)\times d_e}$, where T represents the number of temporal segments, $|C_d|$ is the number of dynamic channels, $|C_s|$ is the number of static channels, d_e is the dimension of the embeddings, is thus a concatenation of dynamic variables, topological data, and coordinates:

$$E = [p_s, p_m, p_{ch}, x]$$

The transformer processes this through various layers of encoding and decoding, learning its latent representations. Each transformer layer performs self-attention and feed-forward operations, enabling the model to capture long-term relationships between temporal and spatial tokens. The transformer's output is an encoded representation of the input that can be used for various tasks.

Chapter 3

Datasets analysis

This chapter provides a comprehensive overview of the datasets used in this study, highlighting their characteristics, sources and relevance. It is structured into different sections to facilitate detailed exploration of the datasets. Each block will then contain insights into the sources, formats and attributes used in the model for each data source. The problems of each dataset will also be highlighted, if any. Finally, a final section will be dedicated to how it is decided to structure the data preprocessing and therefore resolve any problems mentioned above.

3.1 Datasets description

The sources of the datasets can be divided into two groups: dynamic sources, which include Sentinel3, Sentinel5P and Era5 and static sources, which include DEM and Land Cover. Dynamic data change daily, instead, static data are always the same. The dataset covers the period from April 30th, 2018, corresponding to the deployment of Sentinel5P, to the end of 2023. For this research, the study area selected is the central part of Milan, as depicted in Figure 3.1. This area was chosen because it encompasses all the relevant land sensors, that collect pollutant concentrations.

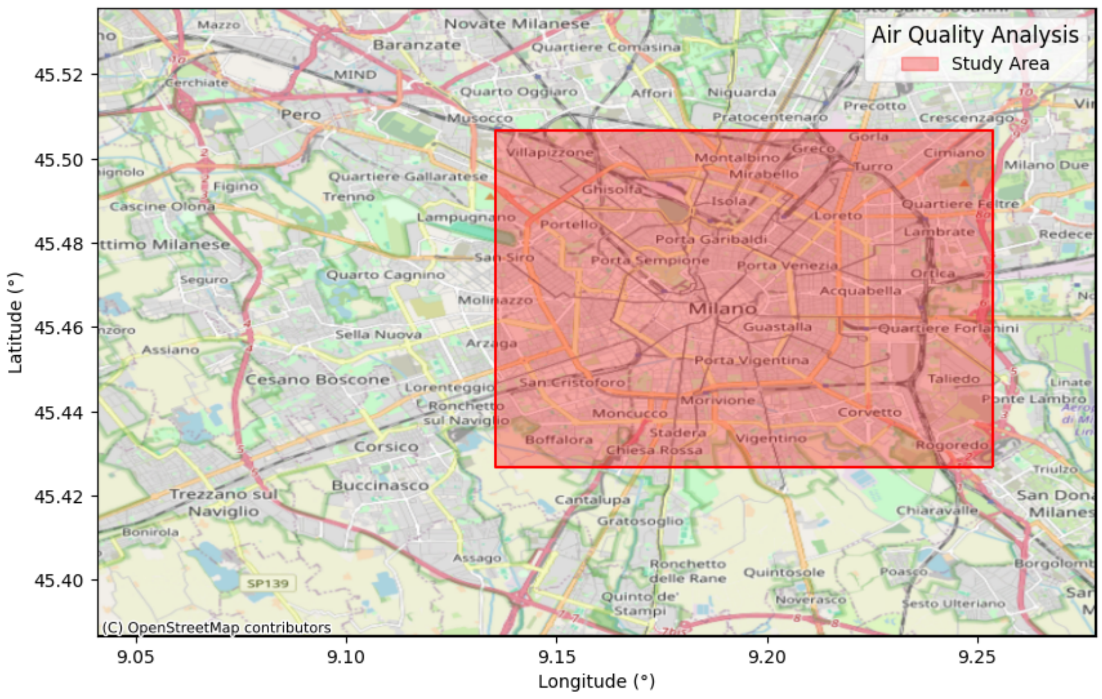

Figure 3.1: The area analyzed in Milan is the central part of the city, chosen because it includes all the pertinent land sensors that gather pollutant concentrations.

3.1.1 Static Data

Land Cover

The Urban Atlas 2018 Land Use/Land Cover (LU/LC) dataset [43], provided by the Copernicus program, serves as a static resource for land monitoring. This dataset offers a consistent snapshot of land use and coverage at a 10-meter resolution specifically for the year 2018. It consists of a wide range of categories, such as residential areas, commercial areas, green spaces, transport infrastructure, and water bodies, mapped with a minimum mapping unit of 0.25 hectares. Table 3.1 shows the full list of labels.

The LU/LC of Urban Atlas 2018 is used in GeoTIFF format. To better clarify the type of data available, Figure 3.2 provides the land cover of the city centre of Milan originally provided by the dataset.

Band
Continuous Urban Fabric (S.L. > 80%)
Discontinuous Dense Urban Fabric (S.L.: 50% - 80%)
Discontinuous Medium Density Urban Fabric (S.L.: 30% - 50%)
Discontinuous Low Density Urban Fabric (S.L.: 10% - 30%)
Discontinuous Very Low Density Urban Fabric (S.L. < 10%)
Isolated Structures
Industrial, commercial, public, military and private units
Fast transit roads and associated land
Other roads and associated land
Railways and associated land
Port areas
Airports
Mineral extraction and dump sites
Land without current use
Green urban areas
Sports and leisure facilities
Open spaces with little or no vegetation (beaches, dunes, bare rocks, glaciers)
Arable land (annual crops)
Permanent crops (vineyards, fruit trees, olive groves)
Pastures
Complex and mixed cultivation patterns
Orchards at the fringe of urban classes
Forests
Herbaceous vegetation associations (natural grassland, moors...)
Wetlands
Water bodies

Table 3.1: Urban Atlas Land Cover 2018 bands

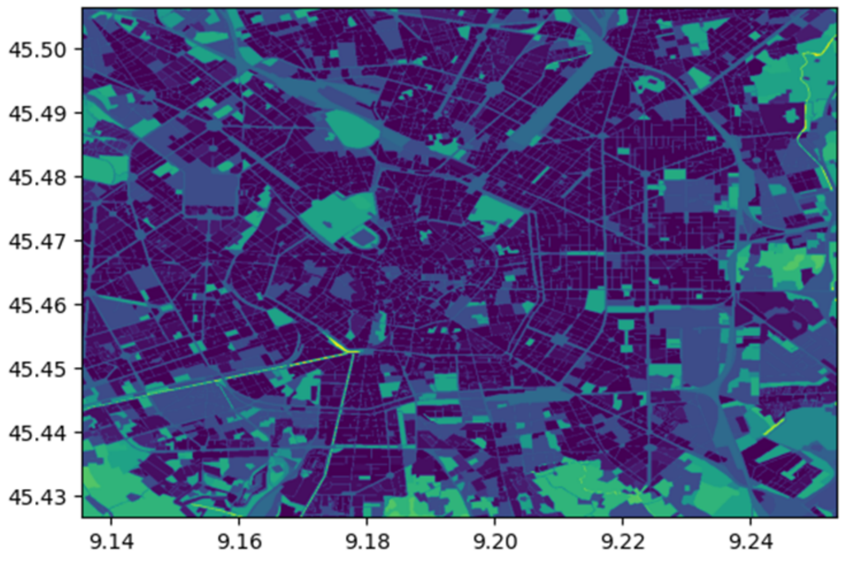

Figure 3.2: Land Cover of the city centre of Milan (original taxonomy).

DEM

The Digital Elevation Model (DEM) [44] provided by the Copernicus program, in collaboration with the German Aerospace Center (DLR) and Airbus Defense and Space, collects characteristics inherent to the earth's surface, including the type of terrain present in a certain area, man-made structures such as buildings and infrastructure, and natural elements such as vegetation. The dataset is obtained from a rework of the dataset known as WorldDEMTM, which includes the flattening of water bodies and the constant flow of rivers. Additionally, special modifications have been applied to coastlines, airports, and terrain structure corrections. Underpinning the collected measurements is interferometric synthetic aperture radar (InSAR) to generate high-precision elevation data. This dataset is provided in several formats, including DGED (Defense Gridded Elevation Data), DTED (Digital Terrain Elevation Data), and INSPIRE (Infrastructure for Spatial Information in Europe). Each format has specific grid spacing, file formats (GeoTIFF for DGED and DTED), and coordinate reference systems. Of interest to this research is the DGED format, which uses a 32-bit floating-point data type and comes in GeoTIFF format.

The data resolution for the city of Milan is 10 m per pixel; in some cases, it is also obtained or reprocessed using LiDAR instrumentation. To better clarify the type of data available, Figure 3.3 shows an example of GeoTIFF.

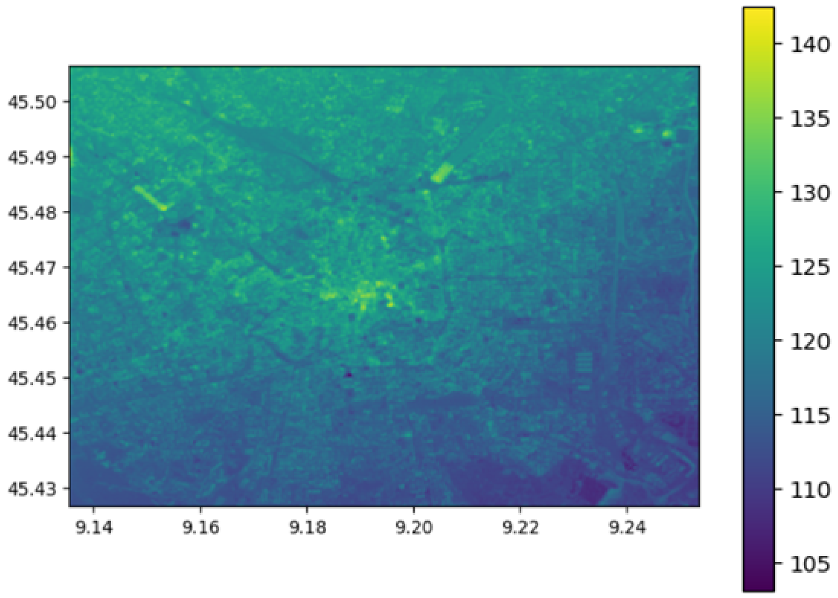

Figure 3.3: DEM for the city centre of Milan.

3.1.2 Dynamic Data

Era5

ERA5 [45] is the latest of five European Center for Medium-Range Weather Forecasts (ECMWF) reanalyses of global climate and weather. The data collected includes a time window ranging from 1940 to the present day. These data concern climate parameters, which have been added and improved over the years regarding instrumentation precision and spatial and temporal resolution. Although data is collected hourly, preliminary ERA5 data is available within five days (ERA5-T), with subsequent validation within two months. The dataset is created by combining model data with global observations. This process is similar to that used by numerical weather prediction centres and allows the production of a globally complete and coherent data set. Among the biggest improvements, undoubtedly, are updates to cloud and precipitation patterns, improved parameterizations for convection, turbulent mixing, and surface interaction, as well as better representations of evaporation in bare soils and conditions of the snow cover. The oceanic component has also been enhanced with a new wave adduction scheme. The data resolution is on a regular grid of 0.25 degrees for reanalysis and 0.5 degrees for uncertainty estimation, while for ocean waves it is 0.5 and 1 degree, respectively. The data used are collected in a CSV file and Table 3.2 contains the subset of bands used in the context of this work. To better clarify the type of data available, Figure 3.4

shows the band named *d2m* of the dataset for the city of Milan.

Name	Meaning	Unit of measure
u10	10m U-component of wind	meters per second (m/s)
v10	10m V-component of wind	meters per second (m/s)
d2m	2m dew point temperature	Kelvin (K)
msl	Mean sea level pressure	Pascal (Pa)
mcc	Medium cloud cover	fraction (0 to 1)
skt	Skin temperature	Kelvin (K)
ssr	Surface solar radiation downwards	Watts per square meter (W/m^2)
tp	Total precipitation	meters (m)

Table 3.2: ERA5 bands

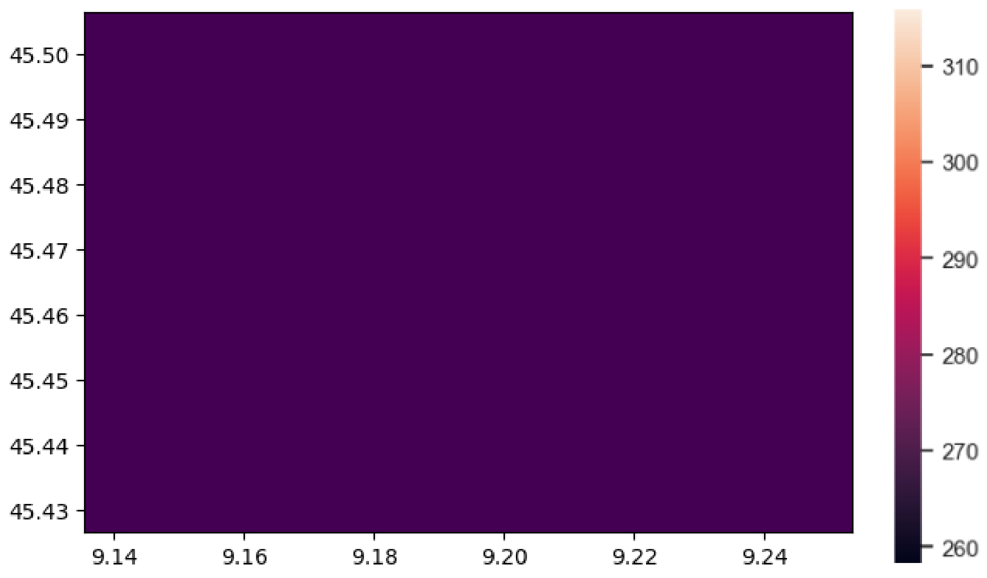

Figure 3.4: ERA5 band (*d2m*) for the city centre of Milan.

Sentinel3

Sentinel3 [46] dataset collects data from a satellite launched in 2016 and its suite of advanced instruments, among which are two large optical instruments: the Ocean and Land Color Instrument (OLCI) with 21 spectral channels with a wavelength ranging from 0.4 to 1.0 μm, and the Sea and Land Surface Temperature Radiometer (SLSTR) with 9 spectral channels with a wavelength ranging from 0.5 μm to 13 μm, acquiring data in both the nadir and oblique directions [47]. Additionally, there is a SAR Radar Altimeter (SRAL) and a Microwave Radiometer (MWR), complemented by a suite of Precise Orbit Determination (POD) instruments. Sentinel3 provides two daily measurements with a resolution of 300 meters for visual images and 1 kilometre for thermal measurements. Table 3.3 contains the bands and their resolutions, reporting only those used in this work. To better clarify the type of data available, Figure 3.5 provides an example of $F1$ band in the centre of the city of Milan in April 2018.

Band	**Meaning of Measure**	**Resolution Band per pixel**
F1	Visible and near-infrared	1000 m
F2	Red edge	1000 m
S7	Thermal infrared (nadir)	1000 m
S8	Thermal infrared (oblique)	1000 m
S9	Thermal infrared	1000 m

Table 3.3: Sentinel3 spectral bands

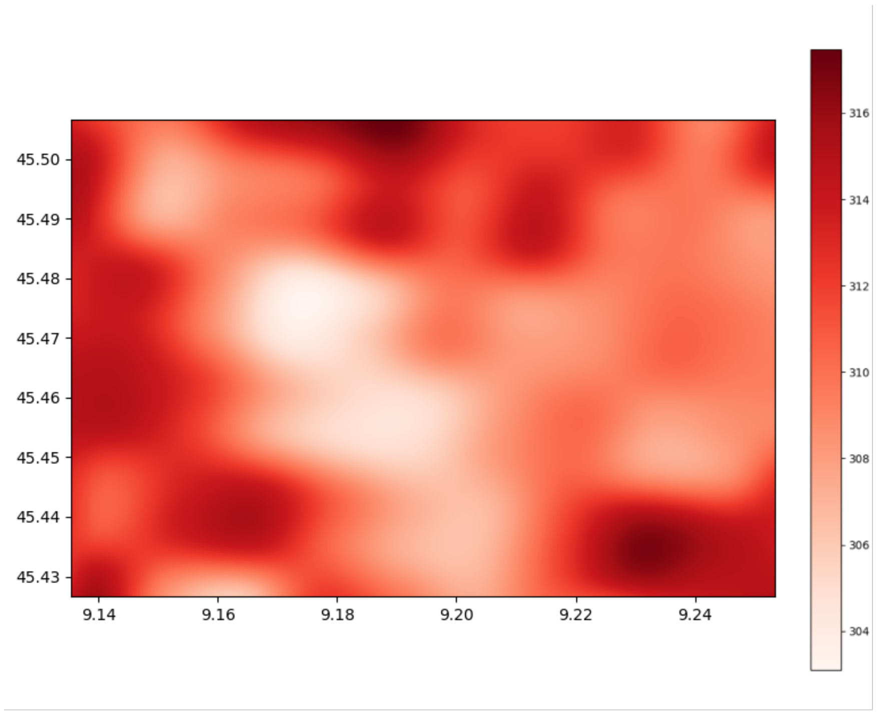

Figure 3.5: Example of Sentinel3 band F1 collected in the city centre of Milan in April 2018.

Sentinel5P

Sentinel5P [48] is a satellite, which carries the TROPOspheric Monitoring Instrument (TROPOMI), which is a nadir imaging spectrometer. The sensor measures electromagnetic spectral wavelengths between UV (270-320 nm) and SWIR (2305-2385 nm) to monitor air pollution. The TROPOMI sensor has a neighbour pixel size of 7×3.5 km^2 for most spectral bands, except for the UV-1 and SWIR bands, which have pixel sizes of 7×28 km^2 and 7×7 km^2, respectively. Since its launch, the TROPOMI sensor has been used worldwide in numerous research projects related to mapping, monitoring, and modelling air pollution. TROPOMI provides accurate daily measurements of various atmospheric constituents. Its main challenges include relatively low resolution, limiting the ability to collect emissions in detail. Additionally, observations may be affected by cloud cover and variable atmospheric conditions. Not all the data collected by Sentinel5P for a single day in the city of Milan are considered input to the model, and therefore in Table 3.4 there is a complete list and a single explanation of those of greatest interest. To better clarify the type of data, Figure 3.6 provides an example of a band extracted from the GeoTIFF in the city of Milan in April 2018.

Measured Bands	**Description**
CO	Carbon Monoxide
HCHO	Formaldehyde
NO_2	Nitrogen Dioxide
O_3	Ozone
SO_2	Sulfur Dioxide
CLOUD_TOP_PRESSURE	Cloud Top Pressure
CLOUD_BASE_PRESSURE	Cloud Base Pressure
AER_AI_340_380	Aerosol Index (340-380 nm)
AER_AI_354_388	Aerosol Index (354-388 nm)

Table 3.4: Sentinel5P spectral bands measured by TROPOMI sensor.

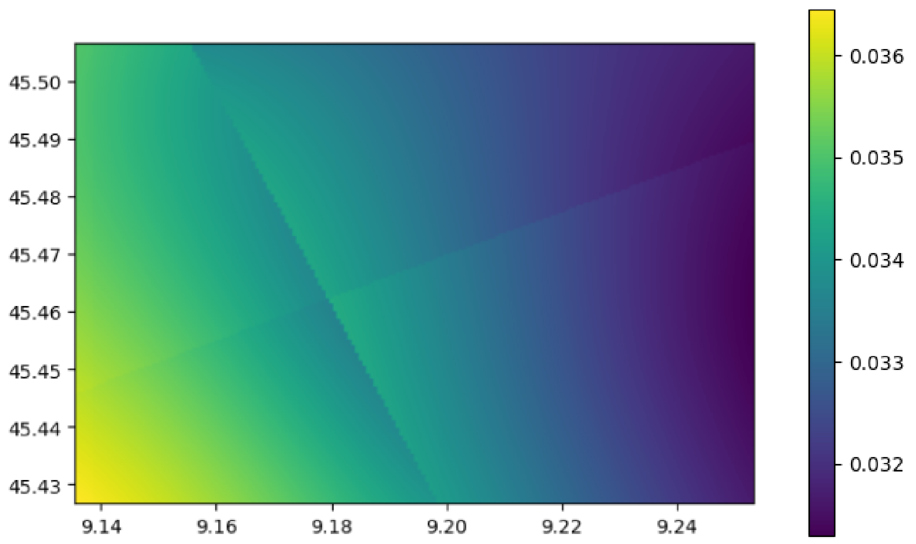

Figure 3.6: Example of Sentinel5P band (CO) collected in the city centre of Milan in April 2018.

3.1.3 Stations data

Stations data are used as labels to train and test the model. This dataset is manually created and contains punctual data from the city of Milan. It is formatted as a daily GeoTIFF file, where each band contains data on collected pollutants (i.e. PM10, PM2.5, O_3, NO_2, and SO_2). Each pixel in a band, instead, represents an area of the city and holds a numerical value indicating the concentration of a pollutant.

These values are of two types:

- **Hard Labels**: values collected by local land stations and downloaded from the open portal of Milan municipality [49].

- **Soft Labels**: data generated by five distinct XGBoost-based models, one for each pollutant. These models estimate values for areas without land sensors.

There are five terrestrial sensors and all of them are located in the centre of Milan. Figure 3.7 illustrates the positions of land stations within Milan, along with the corresponding pollutants monitored by each station.

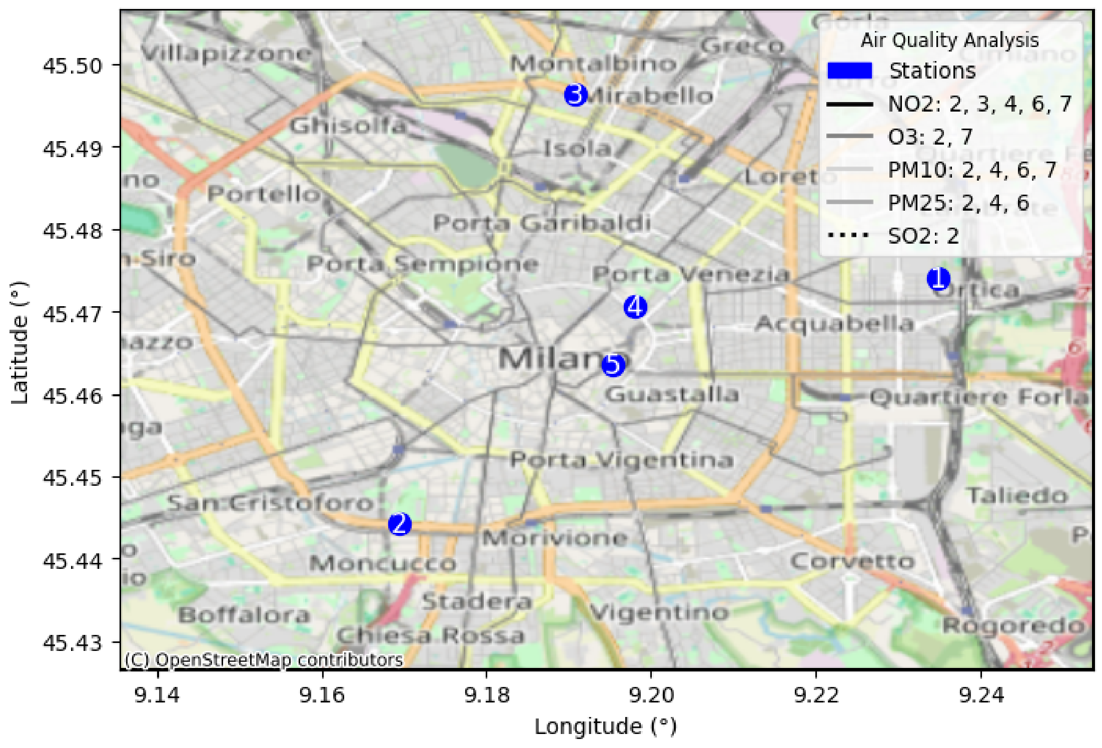

Figure 3.7: Stations in the city of Milan and pollutants collected.

The difference between hard and soft labels is particularly useful in the model training phase, which will be further discussed in the next chapter (Chap. 4).

3.2 Data preprocessing

The different steps necessary to prepare the data for analysis are presented. These are: data cleaning, management of missing values, temporal and spatial alignment of the data, creation of timeseries, and generation of labels for forecasting.

3.2.1 Data cleaning and space-time alignment

As fully described above in the chapter, data are divided into static and dynamic. The dynamic data sources (Sentinel5P, Sentinel3 and Era5), i.e., those that vary over time (in this case, with a daily frequency), all come from satellite images in GeoTIFF format. Each file therefore contains a timestamp, a measurement for each quantity that the satellite must measure, and the related metadata (spatial resolution, temporal resolution, bounding box). Static data sources (DEM and Land Cover), on the other hand, are also in GeoTIFF format or traced back to this format, but consist of a single file that remains constant for the entire duration of the analyzed period.

For the Land Cover dataset, the categories have been grouped into 10 macro-categories, obtaining the taxonomy visible in Table 3.5. Figure 3.8 shows the output of a land cover band after the categories are grouped. After grouping the labels of the land cover into 10 macro-categories, a raster data set is created where each channel represents one of these 10 labels, and each pixel within each channel contains the percentage of that label in the area represented by the pixel.

Category	**Contains**
Urban	Continuous Urban Fabric (S.L. > 80%), Discontinuous Dense Urban Fabric (S.L.: 50% - 80%), Discontinuous Medium Density Urban Fabric (S.L.: 30% - 50%), Discontinuous Low Density Urban Fabric (S.L.: 10% - 30%), Discontinuous Very Low Density Urban Fabric (S.L. < 10%), Isolated Structures, Industrial, commercial, public, military and private units, Construction sites
Road	Fast transit roads and associated land, Other roads and associated land
Railways	Railways and associated land
Port	Port areas
Airports	Airports
Extraction	Mineral extraction and dump sites
NoUse	Land without current use
Green	Green urban areas, Arable land (annual crops), Permanent crops (vineyards, fruit trees, olive groves), Pastures, Complex and mixed cultivation patterns, Orchards at the fringe of urban classes, Forests, Herbaceous vegetation associations (natural grassland, moors...)
OpenSpaces	Sports and leisure facilities, Open spaces with little or no vegetation (beaches, dunes, bare rocks, glaciers)
Water	Wetland, Water bodies

Table 3.5: Land Cover Category Mapping

To work with timeseries built on these data, two fundamental problems arise: missing data (both of single measurements within the day and of entire days) and different resolutions, both in the spatial and temporal domains. This means that some satellites make multiple measurements per day (i.e. Sentinel3) while others only make one, and the measurements differ from each other in their spatial resolution. Therefore, it is possible that even if the bounding box is the same, the

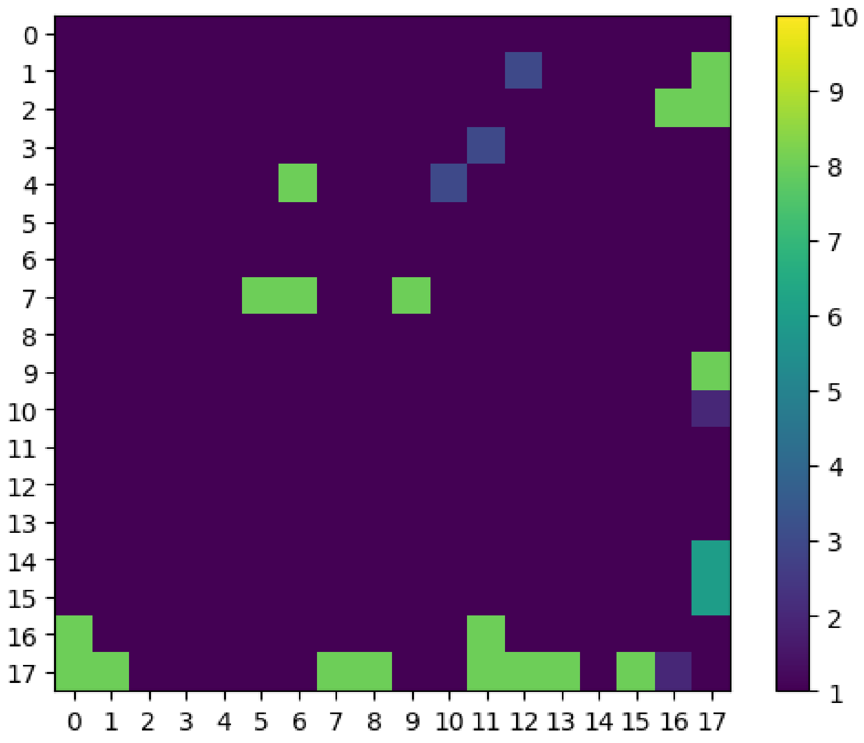

Figure 3.8: Land Cover of the city centre of Milan (resized taxonomy). The categories of the LU/LC Urban Atals 2018 dataset have been grouped into 10 macro-categories.

pixels between the sources may represent a different portion of the geographic area due to the reprojection algorithm. The implemented solutions are the following:

- **Spatial alignment**: all sources are aligned to a final resolution of 500 m per pixel, which corresponds to an image of shape 17 x 18. This means that when an image has a finer resolution, groups of adjacent pixels are averaged. If the initial resolution is smoother, the final image is filled and interpolated to reach the final resolution.

- **Temporal alignment**: all sources are brought to the minimum daily frequency, in this case, one measurement per day is therefore considered for each source. In the case of multiple measurements, the choice is to take the one carried out in the time slot most similar to that of the collection of the other measurements.

- **Missing spatial data**: when one of the sources is missing a measurement of one of its bands, this value is filled with the average (calculated along the temporal dimension) of that band.

- **Missing temporal data**: When one of the sources is missing for an entire day, synthetic data is generated to cover that day. The values correspond to the average along the entire temporal dimension of the measurements for each band.

- **Static time data**: considering static sources, the same value is used for each timestamp.

Thus, for each source, exactly one daily measurement for each band with the same resolution per pixel is obtained. This means that all sources are now aligned spatio-temporally.

3.2.2 Creation of timeseries

After preprocessing the data, the timeseries are generated by merging all bands from the different data sources for the same day. This process allows the creation of a temporal sequence that integrates the various measurements collected over April 2018 and 2023. Each element of the dataset will therefore be a timeseries of fixed length N. During the experimental phase, it is decided to create the timeseries with a stride equal to 1, and therefore each timeseries overlapped with the next (i.e. if the first timeseries is $[t, t+N]$ the next one will be $[t+1, t+N+1]$).

Figure 3.9 illustrates a sample representation of the input data to the model: dynamic bands vary between the two days, while static bands remain consistent and are repeated for each day.

To further characterize the input with information on seasonality trends and possible periodic behaviours, temporal information, expressed as day of the week and day of the year, is used. In addition, spatial information, such as latitude and longitude, is added.

Finally, the final format of the dataset is composed by:

- **Timeseries**: it is the series of concatenated measurements of a contiguous period.

- **Latitude and Longitude**: contains the spatial information (latitude, longitude) of the analyzed area.

- **Day of the year**: for each of the days contained in the timeseries, the corresponding day of the year.

- **Day of the week**: for each of the days contained in the timeseries, the corresponding day of the week.

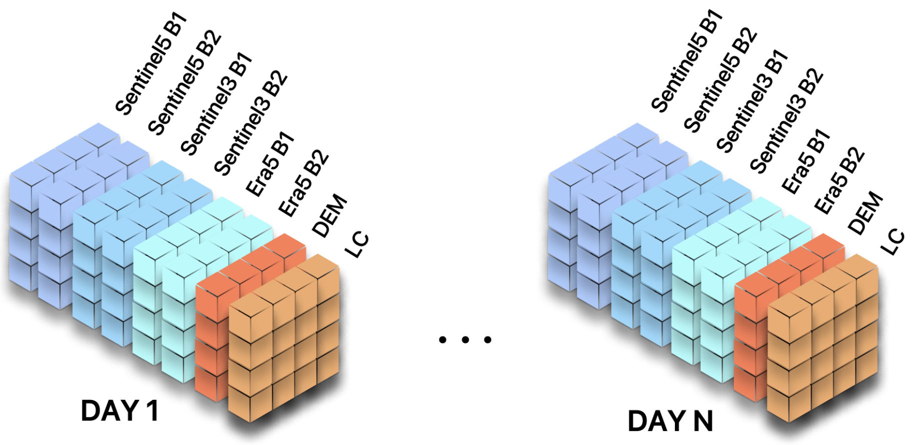

Figure 3.9: Example of timeseries, the input of the model. All bands for each day are grouped together, with dynamic bands varying between days while static bands remain consistent and are repeated daily.

3.2.3 Generation of ground truth labels

The task of pollutant prediction involves providing a target value for each timeseries. This target value can be either the next day's hard labels or, if unavailable, the next day's soft labels. Given that the timeseries of pixels concludes on day t, the input to the model incorporates the target values for day $t + 1$. Furthermore, each item in the dataset includes a loss factor used to weigh the loss according to the trust level of the provided target value. This loss factor is 1 when using the hard label and decreases based on the distance from the nearest stations that measure the hard label.

Chapter 4

Methodology

This chapter provides a brief description of the model used and the related adaptations made to improve performance. Furthermore, it shows a brief introduction of the loss function and the metrics used to train and evaluate the models.

4.1 Model

For the analysis and prediction of pollutants, the TimeSformer model presented in Chapter 2 is used. This model takes as input a series of discontinuous video frames, which in this case are the timeseries extensively described in Chapter 3. This input then passes through a Convolution Layer to reduce the individual rasters into patches and generate embeddings, which are then passed to the transformer encoder, immediately after adding the positional embedding. The difference from the original model lies simply in the elimination of the CLS token and the classification head (MLP), due to the different task: regression rather than classification. The classification head is replaced by five regression heads, which consist of MLPs with 3 hidden layers, one for each pollutant to be predicted. Figure 4.1 shows the architecture of the TimeSformer model adapted to the specific task.

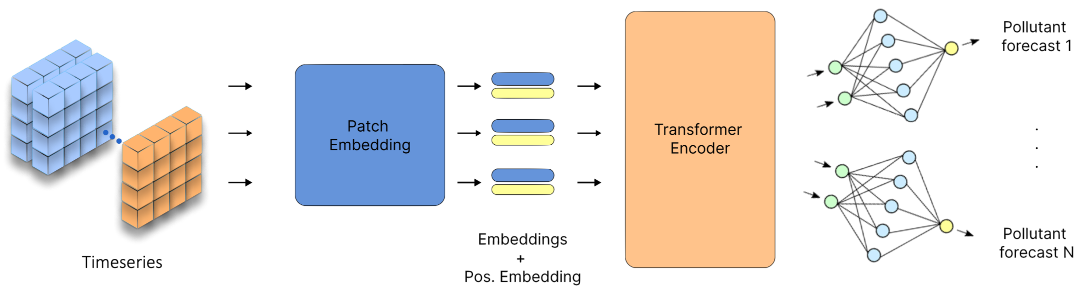

Figure 4.1: Architecture of the TimeSformer model adapted for the task of pollutant prediction

The architecture of the model consists of the following components:

Input and Patch embedding

The input of the model $X \in \mathbb{R}^{B \times T \times C \times H \times W}$ is a timeseries with dimensions $[B, T, C, H, W]$, where B is the batch size, T is the number of timesteps, C is the number of bands, H is the height, and W is the width.

Each frame is divided into patches of size $P \times P$ by the convolution layer, which focuses on the local regions of the image. Convolutional layers are designed to detect and capture spatial hierarchies of patterns within images by systematically applying filters across the input data. Each filter, or kernel, slides over the image, computing dot products with local patches of the input at every position.

It reduces each raster into patches and generates the embedding:

$$x(p,t) = \mathrm{Conv}(X) \in \mathbb{R}^{B \times T \times N \times D}$$

where $N = \frac{H \times W}{P^2}$ is the number of patches per frame and D is the dimensionality of the embeddings.

Positional embedding

Each latent vector $x(p,t) \in \mathbb{R}^{B \times T \times N \times D}$ is summed with the positional embedding $e_{\text{pos}}(p,t) \in \mathbb{R}^{N \times T \times D}$ to include spatio-temporal information, obtaining the final embedding $z(p,t) \in \mathbb{R}^{B \times T \times N \times D}$. Therefore for each patch (p,t):

$$z(p,t) = Ex(p,t) + e_{\text{pos}}(p,t)$$

where $E \in \mathbb{R}^{D\times 3P^2}$ is the embedding matrix and $e_{\text{pos}}(p,t) \in \mathbb{R}^D$ is the positional embedding.

Transformer Encoder

The embeddings $Z \in \mathbb{R}^{B\times T\times N\times D}$ are passed through the transformer encoder, which uses a combination of temporal and spatial attention to capture dependencies in the timeseries data. The transformer encoder uses the Divided Space Time Attention mechanism, where temporal attention and spatial attention are applied separately, one after the other. In particular,

- **Temporal Attention**:

  The output of the temporal attention is:

  $$\mathbf{Z}^t = \text{Attention}(\mathbf{Q}^t, \mathbf{K}^t, \mathbf{V}^t) \in \mathbb{R}^{B\times T\times N\times D}$$

  where,

  $$\text{Attention}(\mathbf{Q}^t, \mathbf{K}^t, \mathbf{V}^t) = \text{softmax}\left(\frac{\mathbf{Q}^t(\mathbf{K}^t)^\top}{\sqrt{d_{k,t}}}\right)\mathbf{V}^t$$

  $$\mathbf{Q}^t, \mathbf{K}^t, \mathbf{V}^t = \mathbf{ZW}_t^Q, \mathbf{ZW}_t^K, \mathbf{ZW}_t^V,\ \mathbf{Q}^t, \mathbf{K}^t, \mathbf{V}^t \in \mathbb{R}^{B\times T\times N\times D}$$

  $d_{k,t}$ is the dimension of the keys.

- **Spatial Attention**:

  The output of the spatial attention is:

  $$\mathbf{Z}^s = \text{Attention}(\mathbf{Q}^s, \mathbf{K}^s, \mathbf{V}^s) \in \mathbb{R}^{B\times T\times N\times D}$$

  where,

  $$\text{Attention}(\mathbf{Q}^s, \mathbf{K}^s, \mathbf{V}^s) = \text{softmax}\left(\frac{\mathbf{Q}^s(\mathbf{K}^s)^\top}{\sqrt{d_{k,s}}}\right)\mathbf{V}^s$$

  $$\mathbf{Q}^s, \mathbf{K}^s, \mathbf{V}^s = \mathbf{ZW}_s^Q, \mathbf{ZW}_s^K, \mathbf{ZW}_s^V,\ \mathbf{Q}^s, \mathbf{K}^s, \mathbf{V}^s \in \mathbb{R}^{B\times T\times N\times D}$$

  $d_{k,s}$ is the dimension of the keys

- **Combination of Attention Mechanisms and Linear Layer**:
  The embeddings generated from the spatial and temporal attention heads

  $$\mathbf{Z}^t, \mathbf{Z}^s \in \mathbb{R}^{B \times T \times N \times D}$$

  are then summed and normalized. The final embedding $\mathbf{Y}$ is passed through a linear layer.

  $$\mathbf{Y} = \text{Linear Layer}(\text{LayerNorm}(\mathbf{Z}^s + \mathbf{Z}^t)) \in \mathbb{R}^{B \times T \times N \times D}$$

Upsampling

The upsampling module is employed to scale up embeddings originally sized for n patches to the dimensions $H \times W$. This adjustment is necessary to enable predictions for each pixel within the bounding box, specifically regarding pollutant values. Formally, the output of the transformer encoder is reshaped $\mathbf{Y} \in \mathbb{R}^{B \times T \times H' \times W' \times D}$

Next, the reshaped embedding $\mathbf{Y}$ is upsampled to the original image resolution (H, W) using an upsampling method, so $\mathbf{Y} \in \mathbb{R}^{B \times T \times H \times W \times D}$

Finally, the time and spatial dimensions are flattened to prepare the embedding for the regression heads, $\mathbf{Y} \in \mathbb{R}^{B \times (T \cdot H \cdot W) \times D}$

Regression Heads

The embedding $\mathbf{Y}$ is then passed to several regression heads corresponding to the pollutants to be predicted. Each head is an MLP with 3 hidden layers:

$$\hat{y}_i = \text{MLP}_i(\mathbf{Y}) \in \mathbb{R}^{B \times T}$$

where $\hat{y}_i$ is the prediction for pollutant i and MLP_i represents the regression head for pollutant i.

The MLP is a feed-forward neural network composed of multiple fully connected layers. Each layer is followed by an activation function (in this study ReLU is used). The general formulation of an MLP with L layers is:

$$\hat{y} = \mathbf{W}_L \mathbf{h}_{L-1} + \mathbf{b}_L \in \mathbb{R}^{B \times 1}$$

where $\mathbf{W}_i$ and $\mathbf{b}_i$ are the weights and biases of the i-th layer, and

$$\mathbf{h}_l = \text{ReLU}(\mathbf{W}_l \mathbf{Y} + \mathbf{b}_l) \in \mathbb{R}^{B \times D}$$

Numerically, the input size consists of a height of 17 pixels and a width of 18 pixels. Each image is divided into patches of size 2x2. The model processes 7 frames and employs a divided space-time attention mechanism. The architecture includes 4 layers, each with an equal number of attention heads and an MLP ratio of 4.0. The final embedding obtained for each pixel is of shape 512, which serves as the input to the five MLP regressors, one for each pollutant, to generate the predicted value of the pollutant.

4.2 Model adaptations

The principal alterations implemented in the TimeSformer model are the incorporation of a patch embedding module and the introduction of novel positional embeddings, which serve to augment the data with spatial and temporal information intrinsic to the sequence. Ultimately, a decoder is employed to include the weather forecast for the day being predicted alongside other inputs.

4.2.1 Patch embedding per single source

The patch embedding in the original model consists of a single convolutional layer. However, since the input to the model is constructed from data collected from various sources, this work introduces the generation of a patch embedding for each source dataset and then the union of all latent vectors corresponding to the same patch into a single embedding.

To achieve this, the patch embedding module is modified by adding several convolutional layers equivalent to the number of source datasets. Finally, several strategies are implemented to generate the final latent vector: MLP, concatenation, average pooling, and attention mechanism. Formally, let $X^{(s)}$ denote the input tensor from the source dataset s. For each source dataset s, the module applies a convolutional layer to obtain patch embeddings:

$$E^{(s)} = \text{Conv}(X^{(s)})$$

where $E^{(s)}$ is the patch embedding for source s, and Conv represents the convolution operation.

Next, to combine the patch embeddings from all source datasets into a unified embedding E, different aggregation methods are explored:

- **MLP (Multi-Layer Perceptron)**: it combines embeddings, learning and synthesizing the most significant features of the input data.

$$E = \text{MLP}([E^{(1)}, E^{(2)}, \ldots, E^{(S)}])$$

- **Concatenation**: the embeddings are generated within a smaller latent space, and these vectors are then merged together to obtain the final vector of the desired shape. This method is straightforward and faster in execution.

$$E = \text{Concatenate}([E^{(1)}, E^{(2)}, \ldots, E^{(S)}])$$

- **Average Pooling**: it computes the arithmetic mean of the embeddings, creating a representative final embedding. It is computationally efficient.

$$E = \frac{1}{S} \sum_{s=1}^{S} E^{(s)}$$

- **Attention Mechanism**: it assigns weights to each embedding based on their relative importance and combines them into a weighted sum. This approach allows for giving more relevance to the most informative embeddings, producing dynamic and contextual representations.

$$E = \sum_{s=1}^{S} \alpha^{(s)} E^{(s)}$$

  where $\alpha^{(s)}$ are attention weights calculated based on the embeddings $E^{(s)}$.

4.2.2 Geo-spatial and time information

Tseng et al. [42] present a pre-trained model on satellite data, named PRESTO, discussed in Chapter 2, which consists of a transformer and uses geospatial information, latitude and longitude, and a temporal datum, the month of the year, as positional embeddings.

Similarly to their work, the two positional embeddings already present in the TimeSformer with the use of the divided space-time attention mechanism for the transformer encoder, $e_{\text{pos}} \in \mathbb{R}^{B \times N \times T \times D}$ and $e_{\text{time}} \in \mathbb{R}^{B \times N \times T \times D}$ are modified, with information regarding latitude, longitude, the day of the week, and the day of the year. This choice allows the model to learn relationships based on geographical area and seasonality.

The latitude and longitude values are passed as input to the model, $X_{latlon} \in \mathbb{R}^{B \times T \times 3 \times H \times W}$, where B is the batch size, T is the number of days that composed the timeseries in input, H is the height of the image grid, W is the width of the image grid and 3 corresponds to the number of encodings calculated for each pixel, allowing the representation of the geographic position in a three-dimensional space.

The formulas for these encodings, respectively X, Y, Z in a 3D space, are calculated as follows:

$$X = \cos(\mathrm{lat}) \times \cos(\mathrm{lon})$$

$$Y = \cos(\mathrm{lat}) \times \sin(\mathrm{lon})$$

$$Z = \sin(\mathrm{lat})$$

This matrix is then passed through a LatLon Embedding Module to generate embeddings for each patch produced by the model. The output of this module is $e_{\mathrm{latlon}} \in \mathbb{R}^{B\times N\times T\times D}$, where N is the number of patches, T is the number of timesteps and D is the length of the embedding.

For the spatial positional embedding, both e_{pos} and e_{latlon} are added to the output of the Patch Embedding Module:

$$e_{\mathrm{spatial}} = e_{\mathrm{pos}} + e_{\mathrm{latlon}}$$

The values representing the day of the year (doy) and the day of the week (dow) are calculated using sine and cosine functions as follows:

$$p_{\mathrm{dow}_{2i}} = \sin\left(\frac{2\pi \times \mathrm{dow}_i}{7}\right), \quad p_{\mathrm{dow}_{2i+1}} = \cos\left(\frac{2\pi \times \mathrm{dow}_i}{7}\right)$$

$$p_{\mathrm{doy}_{2i}} = \sin\left(\frac{2\pi \times \mathrm{doy}_i}{366}\right), \quad p_{\mathrm{doy}_{2i+1}} = \cos\left(\frac{2\pi \times \mathrm{doy}_i}{366}\right)$$

The embeddings p_{dow} and p_{doy} are both of shape $\mathbb{R}^{B\times D/4}$, where B is the batch size and D is the original embedding dimension. These embeddings are obtained by repeating the sine and cosine values and concatenating them together to form a tensor of shape $\mathbb{R}^{B\times D/2}$. This concatenated tensor is then merged with the original e_{time}, which now has a shape of $\mathbb{R}^{B\times D/2}$ (instead of $\mathbb{R}^{B\times D}$).

The final temporal embedding is computed as:

$$e'_{\mathrm{time}} = \left[\mathrm{p}_{\mathrm{dow}}, \mathrm{p}_{\mathrm{doy}}\right] \oplus e_{\mathrm{time}} \in \mathbb{R}^{B\times D},$$

where $\oplus$ denotes concatenation. This approach integrates temporal features represented by day of the week and day of the year into the overall temporal representation used in the model.

4.2.3 Decoder for weather forecast

To improve the TimeSformer model's ability to predict pollutants, a decoder is added. It takes as input both the embeddings generated by the transformer encoder and weather forecasts for the days for which predictions are desired. This choice is

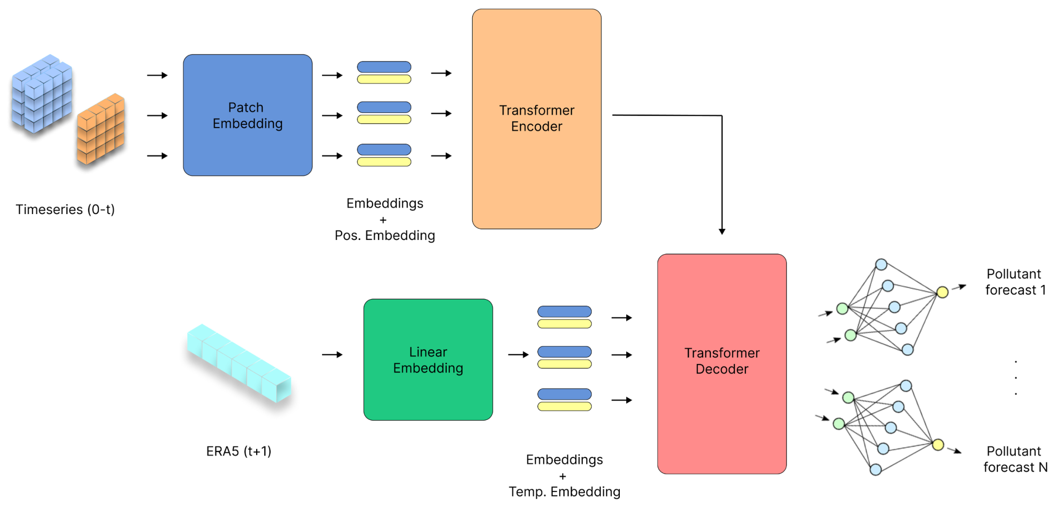

Figure 4.2: Architecture of the TimeSformer model with decoder adapted for the task of pollutant prediction

motivated by recent studies that indicate weather conditions significantly impact pollutant levels [9].

Figure 4.2 shows the architecture with the added decoder. The encoder part remains identical, while the decoder takes as input the embeddings in output from the encoder, $Z_{enc} \in \mathbb{R}^{B \times T \times HW \times D}$ and the latent vectors of the weather forecasting. This data comes from Era5, denoted as $X_{\text{era}} \in \mathbb{R}^{B \times T_f \times N}$, where T_f is the number of future days for which pollutant predictions are desired, and N is the number of bands of the dataset. The weather data is passed through a linear layer, generating $Z_{\text{era}} \in \mathbb{R}^{B \times T_f \times D}$. These embeddings are then summed with the temporal positional embeddings, consisting of day of year, $\text{doy} \in \mathbb{R}^{B \times T_f \times D/4}$, day of week, $\text{dow} \in \mathbb{R}^{B \times T_f \times D/4}$ and e_{time} learnable as in the classic TimeSformer, $e_{\text{time}} \in \mathbb{R}^{B \times T_f \times D/2}$.

Formally,

$$e'_{\text{time}} = [\text{dow}, \text{doy}] \oplus e_{\text{time}} \in \mathbb{R}^{E},$$

$$Z'_{\text{era}} = Z_{\text{era}} + e'_{\text{time}}$$

So, the decoder takes as input the embeddings Z'_{era} and the outputs from the encoder $Z_{enc} \in \mathbb{R}^{B \times T \times HW \times D}$.

This module, *Dec*, applies multi-head attention followed by a linear layer, which can be expressed as:

$$Z_{\text{decoded}} = Dec(Z'_{era}, Z_{enc})$$

At the output of the decoder, the embeddings pass through the forecasting heads, which consist of 3-hidden layer MLPs. The final predictions $\hat{Y}$ are generated as:

$$\hat{Y} = \text{MLP}(Z_{\text{decoded}})$$

In this encoder-decoder architecture, various configurations for the cross attention mask are proposed to enhance the model's performance:

1. **Target pixel's embeddings**: concentrate the model's attention precisely on the embeddings of the target pixel to utilize the most relevant information for prediction.
2. **Surrounding pixels' embeddings**: expand the attention range to include embeddings of the surrounding pixels within a 3x3 square grid, allowing the model to consider spatial context crucial for predicting pollutant levels influenced by nearby areas.
3. **Target pixel's temporal embeddings**: ensure the preservation of comprehensive temporal information for each pixel by returning every frame embedding of each pixel to the encoder.
4. **Target and surrounding pixels' temporal embeddings**: modify the cross-attention mask to focus on both the embeddings of the target pixel and those of the surrounding pixels, while preserving the temporal information for each pixel by returning every frame embedding of each pixel to the encoder. This adaptation allows the model to effectively exploit both temporal and spatial context.

4.3 Loss functions and metrics

To train the model, it is decided to predict every single pollutant for every pixel of the bounding box. As anticipated in Chapter 3, the labels are divided into hard labels, meaning data collected by territorial sensors, and soft labels, synthetic data generated by an algorithm based on XGBoost.

To integrate this information into the model, a weighted loss is adopted to allow the model to better adhere to the more trusted hard labels rather than synthetic ones. The weight in the loss function is inversely proportional to the distance to the closest sensor and evaluated per individual pixel, resulting in the same for each pollutant in the given pixel.

Different loss functions are experimented with, the Mean Squared Error (MSE), the Mean Absolute Percentage Error (MAPE), and the Mean Absolute Error (MAE).

Formally,

- **Mean Square Error (MSE)**: it is computed as the mean of the squared differences between predicted values $\hat{y}_i$ and actual values y_i, adjusted by loss factor$_i$, across n data points.

  $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} \text{loss factor}_i \cdot (y_i - \hat{y}_i)^2$$

  where loss factor$_i$ represents the confidence coefficient associated with the pixel i

- **Mean Absolute Error (MAE)**: It measures the average absolute difference between predicted values $\hat{y}_i$ and actual values y_i adjusted by loss factor$_i$, across n data points.

  $$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} \text{loss factor}_i \cdot |\hat{y}_i - y_i|$$

  where loss factor$_i$ represents the confidence coefficient associated with the pixel i

- **Mean Absolute Percentage Error (MAPE)**: It calculates the average absolute percentage difference between predicted values $\hat{y}_i$ and actual values y_i, adjusted by loss factor$_i$, across n data points. It is particularly useful when the scale of the data varies widely.

  $$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \text{loss factor}_i \cdot \left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

  where loss factor$_i$ represents the confidence coefficient associated with the pixel i

Using Mean Absolute Error (MAE) and Mean Squared Error (MSE) as loss functions, normalization of both labels and predicted outputs is crucial. It helps prevent the model from disproportionately focusing on pollutants with larger numerical values, thereby balancing its attention across all pollutants equally. This approach is particularly beneficial when aiming to predict all five pollutants simultaneously using a single model, where each pollutant may have significantly different measurement scales. The use of MAPE as a training loss, as studied by De Myttenaere et al. [50], diverges from this normalization approach. In the case of MAPE, normalization of labels is avoided as suggested by the authors, to mitigate issues associated with values close to zero, ensuring robust training dynamics across the dataset.

To further reduce the impact of soft labels, a different strategy is finally tested. In particular, the loss weight, except for values equal to 1, which correspond to hard labels, is reduced every N epochs. Formally, let $L_{i,j}$ be the loss associated with pixel (i, j) and $w_{i,j}$ the loss weight for pixel (i, j).

- For pixels with hard labels (weight value equal to 1), the weight remains unchanged.
- For pixels with soft labels, i.e. with a weight value different from 1, the weight is reduced every N epoch. For the final epochs, when $e \geq E_{\text{final}}$, the weight of the soft labels is reduced to 0. Let e be the number of epochs elapsed and k the weight reduction factor:

$$w_{i,j}(e) = \begin{cases} 0 & \text{if} \quad e \geq E_{\text{final}} \\ w_{i,j}(e-1) \cdot k & \text{if} \quad e \equiv 0 \pmod{N} \\ w_{i,j}(e-1) & \text{otherwise} \end{cases}$$

The total loss L of the model is given by the weighted sum of the individual pixel losses:

$$L = \sum_{i,j} w_{i,j}(e) \cdot L_{i,j}$$

This approach initially provides the model with a comprehensive overview of all labels, albeit with different weights on the loss, and subsequently allows the model to focus only on the most certain data.

During the model evaluation phase, the Mean Absolute Percentage Error (MAPE) is used as the primary metric. The model predicts all pollutants simultaneously using shared final embeddings before the forecasting heads. This unified approach allows for the calculation of an overall MAPE, which provides a comprehensive assessment of the model's performance across all pollutants. Technically, the overall MAPE is computed by averaging the MAPE values for each pollutant:

$$\text{Overall MAPE} = \frac{1}{m} \sum_{j=1}^{m} \text{MAPE}_j$$

where, m represents the number of pollutants, and MAPE_j denotes the MAPE calculated for the j-th pollutant.

Chapter 5

Results

In this chapter, the experimental setup and the results obtained using the previously described methodologies are presented.

The data used for the experiments are composed of timeseries of raster data in the city centre of Milan from April 2018 till December 2023. The training set includes data from April 30, 2018, to December 31, 2021, the validation set covers the entire year of 2022, and the test set spans the entire year of 2023.

The experiments are conducted on a workstation equipped with a single GPU, ensuring efficient computational performance. Data analysis and processing are handled using Python, which provides robust libraries for managing datasets and developing models. Neural network models are both developed and trained utilizing the PyTorch Lightning framework, renowned for its flexibility and dynamic computation capabilities. All Python packages and versions are summarized in Table 5.1.

Library	**Version**
`python`	3.10.12
`pytorch-lightning`	1.9.4
`torchmetrics`	1.3.0
`torchvision`	0.1.6
`numpy`	1.24.1
`rasterio`	1.3.4
`einops`	0.7.0
`scipy`	1.10.1
`xarray`	2023.1.0
`torch`	2.0.0
`pandas`	2.2.1

Table 5.1: Libraries used and relative versions

For training, the model uses a batch size of 16 over 150 epochs, with a learning rate of 1×10^{-3}. The optimizer selected for this task is Adam.

The metric evaluated on the test set is the Mean Absolute Percentage Error for overall performance as well as for specific pollutants (PM10, PM2.5, O_3, NO_2, SO_2).

5.1 Baseline

Table 5.2 presents the MAPE values used as baseline metrics for assessing model performance. These metrics are derived from five distinct models utilizing XGBoost. In contrast, our model predicts all five pollutants simultaneously as a single integrated model. The objective is to achieve superior performance compared to this baseline through the implementation of the model presented in Chap. 4.

PM10	**PM2.5**	**O_3**	**NO_2**	**SO_2**
0.3154	0.4604	0.2902	0.2703	0.3424

Table 5.2: MAPE baseline results derived from five distinct models, each based on XGBoost, to predict each pollutant.

5.2 TimeSformer model

Table 5.3 compares the performance of the TimeSformer architecture, used as an encoder to embed the input data and forecast air pollutants, with the baseline results.

The baseline model outperforms the TimeSformer model across all pollutants, achieving lower MAPE errors, indicating greater accuracy in predicting pollutant concentrations.

Model	**Overall**	**PM10**	**PM25**	**O_3**	**NO_2**	**SO_2**
Baseline	-	0.3154	0.4604	0.2902	0.2703	0.3424
TimeSformer	0.4856	0.4139	0.5789	0.4600	0.3506	0.6247

Table 5.3: MAPE comparison between the raw TimeSformer model and the baseline XGBoost models for forecasting air pollutants. The overall MAPE is missing for the baseline because it is evaluated using five distinct models, one for each pollutant, whereas the TimeSformer is a single model predicting all pollutants simultaneously.

After analysing the performance of the raw model, the use of a patch embedder consisting of different convolutional layers, one for each source of the dataset is implemented. Each pixel embedding generated by a different layer is combined using two custom methodologies: a linear layer or concatenation of the latent vectors. In the last method, the latent vectors are combined to form a single embedding vector, resulting in a smaller latent space for the individual vectors. This model, named EoTimeSformer, is evaluated in Table 5.4.

Model	**Combination**	**Overall**	**PM10**	**PM2.5**	**O_3**	**NO_2**	**SO_2**
TimeSformer	-	0.4856	0.4139	0.5789	0.4600	0.3506	0.6247
EoTimeSformer	MLP	**0.4343**	0.4200	0.5757	**0.3620**	**0.2826**	**0.5313**
	Concatenation	0.4465	**0.3563**	**0.4937**	0.4748	0.3262	0.5816

Table 5.4: Comparison of MAPE values for different model architectures and combination methods. The table compares the baseline TimeSformer with the enhanced EoTimeSformer using both MLP and concatenation methods for embedding combination.

Comparing the results, EoTimeSformer outperforms the raw model. Notably, the linear layer used as the combination method surpasses the concatenation method in most pollutant categories, showing significant improvements in predicting O_3, NO_2, and SO_2. However, the concatenation method demonstrates a slight advantage in predicting PM2.5 and PM10. This suggests that the learnable nature of the linear layer contributes to its superior performance in reducing overall error.

The second improvement consists of transforming the latitude, longitude, day of the year, and day of the week information into positional embeddings. Initially, the results reported in Table 5.5 are obtained only with the temporal information as positional embeddings, while keeping the geospatial information as sources in the input model. Subsequently, the latitude and longitude are also transformed into positional embeddings. Each experiment is repeated for both strategies described above to combine the latent vectors output by the patch embedding module. The attention mechanism as a strategy for combining embeddings is also tested, though this did not improve the results. Nonetheless, the use of different positional embeddings outperforms previous results.

Combination	**LAT-LON**	**Overall**	**PM10**	**PM25**	**O_3**	**NO_2**	**SO_2**
MLP	✗	0.4146	**0.3741**	**0.4741**	0.4484	0.2942	0.4822
Concatenation	✗	0.4293	0.3800	0.5098	0.3950	0.3047	0.5571
MLP	✓	**0.4131**	0.4052	0.5502	**0.3441**	0.3035	**0.4628**
Concatenation	✓	0.4284	0.3800	0.5176	0.4506	0.2878	0.5062
Attention	✓	0.5040	0.4602	0.6355	0.5062	0.3043	0.6138

Table 5.5: MAPE obtained from training EoTimeSformer using different combination methods with the inclusion or exclusion of latitude and longitude information. All experiments include day of year and day of week information.

The best configuration is obtained using all the spatial and temporal information as positional embedding and the MLP strategy at the end of the patch module. This approach allows the model to effectively leverage the positional context of the data, enhancing its ability to learn and predict the complex spatio-temporal relationships inherent in the dataset. As a result, this configuration outperforms others, providing the lowest overall MAPE and demonstrating superior performance across most pollutant categories.

Using the same configuration, hyperparameter tuning is performed, changing the embedding size and the number of frames, which make up the timeseries. Table 5.6 shows the parameters used and the results obtained. The best configuration in this optimization process is obtained with an embedding size of 768 and 7 time steps, resulting in the lowest overall MAPE so far.

Embed dim	**Timesteps**	**OVERALL MAPE**	**PM10 MAPE**	**PM25 MAPE**	**O_3 MAPE**	**NO_2 MAPE**	**SO_2 MAPE**
256	7	0.4107	0.3733	0.5130	0.4040	0.2876	0.4758
512	7	0.4172	0.3991	0.5495	0.3556	0.2849	0.4968
768	**7**	**0.3699**	0.3558	**0.4625**	**0.3236**	0.2880	0.4193
768	1	0.4415	0.3894	0.4931	0.4498	0.3007	0.5743
768	2	0.4002	0.3788	0.5141	0.3484	0.2901	0.4695
768	3	0.3976	**0.3403**	0.4791	0.3873	0.3019	0.4794
768	4	0.3987	0.3941	0.5305	0.3607	0.3000	**0.4083**
768	5	0.3925	0.3411	0.4759	0.3686	0.2831	0.4936
768	8	0.4346	0.3885	0.5694	0.3854	0.3138	0.5161
768	9	0.4072	0.3726	0.5071	0.3638	**0.2774**	0.5152
768	10	0.4310	0.4035	0.5490	0.3809	0.3070	0.5145

Table 5.6: MAPE comparison using different hyperparameters of the optimal EoTimeSformer model.

The larger embedding size of 768 likely provides the model with a richer representation of the data, allowing it to capture more complex spatio-temporal patterns. Furthermore, using 7 time steps, corresponding to one week, appears to be an optimal window for model processing. This time frame likely provides a balanced amount of temporal information, allowing the model to learn effectively without being overwhelmed by too much data or losing important temporal dependencies.

5.2.1 TimeSformer model with decoder

This section shows results regarding the architecture of the TimeSformer used as an encoder to embed the input data and the decoder, which takes as input these embeddings and the weather forecasts for the days for which pollutant levels are being predicted.

For these experiments, except all others, the model employs a batch size of 16 and runs for 100 epochs, with a learning rate of 1×10^{-4}. The optimizer selected for this task is Adam. Moreover, the best configuration of the TimeSformer encoder obtained in Table 5.6 is used.

Table 5.7 presents the initial results obtained with the new architecture, exploring various configurations for the cross-attention mask as detailed in Section 4.2.3.

Strategy	**Temp. embed.**	**Overall**	**PM10**	**PM25**	**O_3**	**NO_2**	**SO_2**
Target pixel	✗	**0.4152**	0.4363	0.5695	**0.3546**	0.3548	**0.3606**
Surrounding pixels	✗	0.4259	0.4485	0.5846	0.3754	0.3590	0.3620
Target pixel	✓	0.4220	**0.4266**	**0.5604**	0.4013	0.3584	0.3630
Surrounding pixels	✓	0.4223	0.4449	0.5815	0.3669	**0.3513**	0.3669

Table 5.7: MAPE obtained by the best EoTimeSformer model with the inclusion of the decoder, evaluating different mask strategies.

The best performance is achieved using the target pixel strategy without temporal embeddings, resulting in the lowest overall MAPE. This approach outperforms the others, particularly in O_3 and SO_2 prediction. The surrounding pixels strategy, whether frames are joined or not, consistently performs worse than the target pixel one. These findings suggest that processing individual pixels independently, rather than aggregating them into squares, allows the model to better capture the fine-grained spatio-temporal patterns necessary for accurate predictions.

To downplay the significance of soft labels, which are notably more abundant than hard labels, an approach has been to adjust the loss range not within the standard [0, 1] range, but rather within [0, 0.1] and [0, 0.3]. This adjustment aims to counteract significant overfitting observed when incorporating soft labels into the decoder model. The same configurations as previously tested are employed, and the results are presented in Table 5.8.

Strategy	**Temp. embed.**	**Weight range**	**Overall**	**PM10**	**PM25**	**O_3**	**NO_2**	**SO_2**
Target pixel	✗	[0, 0.1]	**0.4059**	0.4258	0.5407	0.3677	0.3470	**0.3482**
Target pixel	✗	[0, 0.3]	0.4308	0.4290	0.5623	0.4224	0.3624	0.3780
Surrounding pixels	✗	[0, 0.1]	0.42098	0.4349	0.5699	**0.3459**	0.3634	0.3907
Surrounding pixels	✗	[0, 0.3]	0.4196	**0.4113**	0.5410	0.4257	0.3431	0.3772
Target pixel	✓	[0, 0.1]	0.4131	0.4272	**0.5173**	0.4053	**0.3344**	0.3814
Target pixel	✓	[0, 0.3]	0.4160	0.4371	0.5371	0.4096	0.3427	0.3537
Surrounding pixels	✓	[0, 0.1]	0.4217	0.4379	0.5679	0.3541	0.3564	0.3922
Surrounding pixels	✓	[0, 0.3]	0.4196	0.4342	0.5570	0.3765	0.3558	0.3743

Table 5.8: MAPE comparison with varied loss weight ranges and mask strategies in the optimal EoTimeSformer model with the inclusion of the decoder.

The best overall performance is achieved with the target pixel strategy and a weight range of [0, 0.1], which results in the lowest MAPE. This configuration excels particularly in SO_2 prediction. Conversely, the target pixel strategy with a weight range of [0, 0.3] shows the highest overall MAPE, indicating that a broader weight range may introduce instability. Among the surrounding pixels strategies, the best results are obtained without joining temporal embeddings and using a weight range of [0, 0.3], suggesting some benefit from broader weight ranges in this setup.

5.2.2 Strategies with different training loss and methodologies

Various experiments are conducted to evaluate the impact of changing the training loss function and adopting different strategies. The results presented in Table 5.9 utilize MAE and MAPE as the loss functions, consistent with previous methodologies. Therefore, detailed experiments are provided for both encoder-only and encoder-decoder architectures using the best-performing configurations for each scenario.

The encoder-only strategy trained with the MAPE loss function achieves the best overall performance, resulting in the lowest overall MAPE of 0.3588. This configuration also shows superior performance in PM25 and SO_2 prediction. The encoder-decoder strategy with MAPE loss performs worse overall compared to the encoder-only approaches, though it excels in NO_2 prediction. Training with MAE loss generally results in higher errors, suggesting that the MAPE loss function is more effective for this model.

Decoder	**Training loss**	**Overall**	**PM10**	**PM25**	**O_3**	**NO_2**	**SO_2**
✗	MAPE	**0.3588**	0.3731	**0.4443**	0.3844	0.2957	**0.2965**
✗	MAE	0.3599	**0.3592**	0.4466	**0.3569**	0.2915	0.3453
✓	MAPE	0.3864	0.3811	0.5259	0.3655	**0.2775**	0.3822
✓	MAE	0.4066	0.4133	0.5026	0.4993	0.2947	0.3229

Table 5.9: MAPE obtained with the best architecture encoder-only and encoder-decoder with different training loss functions

To further reduce the impact of soft labels, a different strategy is tested. This approach involves gradually decreasing the weight of the soft labels as the number of epochs increases during training. This approach enables the model to focus more on the hard labels while continuing to predict pollutant levels across the entire area of Milan. The results of this strategy, using MSE, MAE and MAPE as training loss functions for both encoder and encoder-decoder architectures, are presented in Table 5.10.

Model	**Decoder**	**Train loss**	**Overall**	**PM10**	**PM25**	**O_3**	**NO_2**	**SO_2**
Baseline	-	-	-	0.3154	0.4604	0.2902	0.2703	0.3424
EoTimeSformer	✗	MSE	0.3367	0.3401	0.4741	0.3184	0.2721	**0.2789**
	✗	MAPE	**0.3066**	**0.2875**	**0.3618**	0.3087	0.2747	0.3003
	✗	MAE	0.3127	0.3253	0.3760	**0.3042**	**0.2653**	0.2926
	✓	MSE	0.4584	0.4037	0.5933	0.3951	0.3124	0.5873
	✓	MAPE	0.3500	0.3476	0.4370	0.3285	0.2956	0.3412
	✓	MAE	0.3659	0.3789	0.5119	0.3689	0.2748	0.2949

Table 5.10: MAPE comparison obtained by EoTimeSformer models trained with different loss functions and loss decay strategies for soft labels. Baseline values are reported to facilitate the comparison.

The encoder-only strategy trained with the MAPE loss function achieves the best overall performance, with an overall MAPE of 0.3066. These results confirm the findings observed in Table 5.9, demonstrating that MAPE loss training consistently outperforms other loss functions for this task.

By adopting this methodology, the best performance is achieved, surpassing the previous baseline, mentioned in Sec. 5.1, for PM10, PM25, and NO_2.

Chapter 6

Conclusions

This study presented an analysis of air quality forecasting using advanced deep learning techniques, with a particular focus on the TimeSformer model and its adaptation. The primary goal was to leverage the strengths of this model to capture intricate spatiotemporal relationships in satellite image data combined with meteorological and morphological data for accurate pollutant concentration prediction. The research was driven by the need for more precise air quality models capable of addressing the complexities of urban pollution, especially in densely populated and industrially active regions.

Key contributions of this thesis include:

1. **Model Development and Optimization**: the study successfully implemented and optimized several variants of the TimeSformer model, leading to an improved version known as EoTimeSformer. This model was designed to handle multi-source data inputs, integrating various convolutional layers. Moreover, incorporating latitude, longitude, and temporal data as positional embeddings, the models improved pollutant predictions based on geographical and seasonal variations.

2. **Performance Metrics and Comparative Analysis**: extensive experimentation revealed that the encoder-only architecture with MAPE as training loss function provided the best results. The research also included a thorough comparison of various training strategies, not only varying the loss function but also the label weighting methodology, concluding that the gradual reduction of soft label weights significantly enhanced the model's focus and predictive capabilities.

This work makes a significant contribution to the development of more effective air quality forecasting models, suggesting future directions for further improvements and applications in the field of environmental monitoring.

These include:

1. **Enhanced Data Integration**: future work could explore integrating additional data sources, such as traffic and vegetation information, to further improve model accuracy. Including such dynamic data could enhance the model's responsiveness to sudden changes in air quality.
2. **Multi-City Training**: this study uses only data collected in Milan from a limited number of land stations providing precise data. Extending the model training to multiple cities and regions simultaneously could facilitate the training process.

These future research directions hold great promise for advancing the field of pollutant forecasting. By tackling current limitations, utilizing larger and higher-quality datasets, and exploring innovative models and untested architectures, overall performance can be enhanced. These initiatives will lead to the creation of more robust, accurate, and semantically aware systems for predicting pollutants.

Bibliography

[1] Andrea Piccoli et al. «Modeling the effect of COVID-19 lockdown on mobility and NO2 concentration in the Lombardy region». In: *Atmosphere* 11.12 (2020), p. 1319 (cit. on p. 1).

[2] Zorana Jovanovic Andersen et al. *Clean air for healthy lungs–an urgent call to action: European Respiratory Society position on the launch of the WHO 2021 Air Quality Guidelines*. 2021 (cit. on p. 3).

[3] Roland Stirnberg, Jan Cermak, Julia Fuchs, and Hendrik Andersen. «Mapping and understanding patterns of air quality using satellite data and machine learning». In: *Journal of Geophysical Research: Atmospheres* 125.4 (2020), e2019JD031380 (cit. on p. 6).

[4] Wan Yun Hong, David Koh, Anis Asma Ahmad Mohtar, and Mohd Talib Latif. «Statistical analysis and predictive modelling of air pollutants using advanced machine learning approaches». In: *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. IEEE. 2020, pp. 1–6 (cit. on p. 6).

[5] Lanyi Zhang, Jane Lin, Rongzu Qiu, Xisheng Hu, Huihui Zhang, Qingyao Chen, Huamei Tan, Danting Lin, and Jiankai Wang. «Trend analysis and forecast of PM2. 5 in Fuzhou, China using the ARIMA model». In: *Ecological indicators* 95 (2018), pp. 702–710 (cit. on p. 6).

[6] Sahar Masmoudi, Haytham Elghazel, Dalila Taieb, Orhan Yazar, and Amjad Kallel. «A machine-learning framework for predicting multiple air pollutants' concentrations via multi-target regression and feature selection». In: *Science of the Total Environment* 715 (2020), p. 136991 (cit. on p. 6).

[7] WC Leong, RO Kelani, and Z Ahmad. «Prediction of air pollution index (API) using support vector machine (SVM)». In: *Journal of Environmental Chemical Engineering* 8.3 (2020), p. 103208 (cit. on p. 6).

[8] Sheen Mclean Cabaneros, John Kaiser Calautit, and Ben Richard Hughes. «A review of artificial neural network models for ambient air pollution prediction». In: *Environmental Modelling & Software* 119 (2019), pp. 285–304 (cit. on p. 6).

[9] Lu Bai, Jianzhou Wang, Xuejiao Ma, and Haiyan Lu. «Air pollution forecasts: An overview». In: *International journal of environmental research and public health* 15.4 (2018), p. 780 (cit. on pp. 6, 36).

[10] A Masih. «Machine learning algorithms in air quality modeling». In: *Global Journal of Environmental Science and Management* 5.4 (2019), pp. 515–534 (cit. on p. 6).

[11] Qi Liao, Mingming Zhu, Lin Wu, Xiaole Pan, Xiao Tang, and Zifa Wang. «Deep learning for air quality forecasts: a review». In: *Current Pollution Reports* 6 (2020), pp. 399–409 (cit. on p. 6).

[12] Adil Masood and Kafeel Ahmad. «A review on emerging artificial intelligence (AI) techniques for air pollution forecasting: Fundamentals, application and performance». In: *Journal of Cleaner Production* 322 (2021), p. 129072 (cit. on p. 6).

[13] Andrew Rowley and Oktay Karakuş. «Predicting air quality via multimodal AI and satellite imagery». In: *Remote Sensing of Environment* 293 (2023), p. 113609 (cit. on p. 6).

[14] Nabin Rijal, Ravi Teja Gutta, Tingting Cao, Jerry Lin, Qirong Bo, and Jing Zhang. «Ensemble of deep neural networks for estimating particulate matter from images». In: *2018 IEEE 3rd international conference on image, Vision and Computing (ICIVC)*. IEEE. 2018, pp. 733–738 (cit. on p. 7).

[15] Chao Zhang, Junchi Yan, Changsheng Li, Xiaoguang Rui, Liang Liu, and Rongfang Bie. «On estimating air pollution from photos using convolutional neural network». In: *Proceedings of the 24th ACM international conference on Multimedia*. 2016, pp. 297–301 (cit. on p. 7).

[16] Unjin Pak, Chungsong Kim, Unsok Ryu, Kyongjin Sok, and Sungnam Pak. «A hybrid model based on convolutional neural networks and long short-term memory for ozone concentration prediction». In: *Air Quality, Atmosphere & Health* 11 (2018), pp. 883–895 (cit. on p. 7).

[17] Unjin Pak, Jun Ma, Unsok Ryu, Kwangchol Ryom, U Juhyok, Kyongsok Pak, and Chanil Pak. «Deep learning-based PM2. 5 prediction considering the spatiotemporal correlations: A case study of Beijing, China». In: *Science of the Total Environment* 699 (2020), p. 133561 (cit. on p. 7).

[18] Jiaqi Zhu, Fang Deng, Jiachen Zhao, and Hao Zheng. «Attention-based parallel networks (APNet) for PM2. 5 spatiotemporal prediction». In: *Science of The Total Environment* 769 (2021), p. 145082 (cit. on p. 7).

[19] Rui Yan, Jiaqiang Liao, Jie Yang, Wei Sun, Mingyue Nong, and Feipeng Li. «Multi-hour and multi-site air quality index forecasting in Beijing using CNN, LSTM, CNN-LSTM, and spatiotemporal clustering». In: *Expert Systems with Applications* 169 (2021), p. 114513 (cit. on p. 7).

[20] Shengdong Du, Tianrui Li, Yan Yang, and Shi-Jinn Horng. «Deep air quality forecasting using hybrid deep learning framework». In: *IEEE Transactions on Knowledge and Data Engineering* 33.6 (2019), pp. 2412–2424 (cit. on p. 7).

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is all you need». In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 7).

[22] Alexey Dosovitskiy et al. «An image is worth 16x16 words: Transformers for image recognition at scale». In: *arXiv preprint arXiv:2010.11929* (2020) (cit. on pp. 7, 8).

[23] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. «On the relationship between self-attention and convolutional layers». In: *arXiv preprint arXiv:1911.03584* (2019) (cit. on p. 7).

[24] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. «Image transformer». In: *International conference on machine learning*. PMLR. 2018, pp. 4055–4064 (cit. on p. 7).

[25] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. «Local relation networks for image recognition». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 3464–3473 (cit. on p. 7).

[26] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. «Stand-alone self-attention in vision models». In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 7).

[27] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. «Exploring self-attention for image recognition». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10076–10085 (cit. on p. 7).

[28] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. «Generating long sequences with sparse transformers». In: *arXiv preprint arXiv:1904.10509* (2019) (cit. on p. 7).

[29] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. «Scaling autoregressive video models». In: *arXiv preprint arXiv:1906.02634* (2019) (cit. on p. 7).

[30] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. «Axial-deeplab: Stand-alone axial-attention for panoptic segmentation». In: *European conference on computer vision*. Springer. 2020, pp. 108–126 (cit. on p. 7).

[31] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. «Vivit: A video vision transformer». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 6836–6846 (cit. on pp. 7, 8).

[32] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. «Is space-time attention all you need for video understanding?» In: *ICML*. Vol. 2. 3. 2021, p. 4 (cit. on pp. 7, 8).

[33] Alexandre Lacoste et al. «Geo-bench: Toward foundation models for earth monitoring». In: *Advances in Neural Information Processing Systems* 36 (2024) (cit. on p. 10).

[34] Ming Jiang, Yimin Chen, Zhe Dong, Xiaoping Liu, Xinchang Zhang, and Honghui Zhang. «Multi-Scale Fusion CNN-Transformer Network for High-Resolution Remote Sensing Image Change Detection». In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2024) (cit. on p. 10).

[35] Yanyuan Huang, Xinghua Li, Zhengshun Du, and Huanfeng Shen. «Spatiotemporal Enhancement and Interlevel Fusion Network for Remote Sensing Images Change Detection». In: *IEEE Transactions on Geoscience and Remote Sensing* (2024) (cit. on p. 10).

[36] 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*. 2016, pp. 770–778 (cit. on p. 10).

[37] Isaac Corley, Caleb Robinson, Rahul Dodhia, Juan M. Lavista Ferres, and Peyman Najafirad. *Revisiting pre-trained remote sensing model benchmarks: resizing and normalization matters*. 2023. arXiv: `2305.13456 [cs.CV]` (cit. on p. 10).

[38] Linying Zhao and Shunping Ji. «CNN, RNN, or ViT? An Evaluation of Different Deep Learning Architectures for Spatio-Temporal Representation of Sentinel Time Series». In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 16 (2022), pp. 44–56 (cit. on p. 10).

[39] Yuan Yuan, Lei Lin, Qingshan Liu, Renlong Hang, and Zeng-Guang Zhou. «SITS-Former: A pre-trained spatio-spectral-temporal representation model for Sentinel-2 time series classification». In: *International Journal of Applied Earth Observation and Geoinformation* 106 (2022), p. 102651 (cit. on p. 10).

[40] Michail Tarasiou, Erik Chavez, and Stefanos Zafeiriou. «ViTs for SITS: Vision Transformers for Satellite Image Time Series». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 10418–10428 (cit. on p. 11).

[41] Casper Fibaek, Luke Camilleri, Andreas Luyts, Nikolaos Dionelis, and Bertrand Le Saux. «PhilEO Bench: Evaluating Geo-Spatial Foundation Models». In: *arXiv preprint arXiv:2401.04464* (2024) (cit. on p. 11).

[42] Gabriel Tseng, Ruben Cartuyvels, Ivan Zvonkov, Mirali Purohit, David Rolnick, and Hannah Kerner. «Lightweight, pre-trained transformers for remote sensing timeseries». In: *arXiv preprint arXiv:2304.14065* (2023) (cit. on pp. 11, 12, 34).

[43] *Urban Atlas LC/LU*. URL: `https://sdi.eea.europa.eu/catalogue/copernicus/api/records/fb4dffa1-6ceb-4cc0-8372-1ed354c285e6?language=all` (cit. on p. 16).

[44] *Digital Elevation Models*. URL: `https://spacedata.copernicus.eu/collections/copernicus-digital-elevation-model` (cit. on p. 18).

[45] *ERA5 Copernicus*. URL: `https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-single-levels?tab=overview` (cit. on p. 19).

[46] Craig Donlon et al. «The sentinel-3 mission: Overview and status». In: *2012 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2012, pp. 1711–1714 (cit. on p. 20).

[47] J Nieke, U Klein, F Borde, B Berruti, J Frerick, J Stroede, and C Mavrocordatos. «Sentinel-3 payload overview». In: *Proc. SPIE Europe* (2009) (cit. on p. 20).

[48] Arjuman R Reshi, Subbarao Pichuka, and Akshar Tripathi. «Applications of Sentinel-5P TROPOMI Satellite Sensor: A Review». In: *IEEE Sensors Journal* (2024) (cit. on p. 22).

[49] *Rivelazione qualità dell'aria comune di Milano*. URL: `https://dati.comune.milano.it/dataset/ds409-rilevazione-qualita-aria-2023` (cit. on p. 23).

[50] Arnaud De Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. «Mean absolute percentage error for regression models». In: *Neurocomputing* 192 (2016), pp. 38–48 (cit. on p. 38).