POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Understanding traffic matrix estimation with eXplainable AI (XAI)

Supervisors

Prof. Guido MARCHETTO

Dr. Alessio SACCO

Candidate

Cristian ZILLI

December 2022

Abstract

Diagnostics constitute a foundation for the management, maintenance, and improvement of computer networks. To this end, traffic matrices are an effective element of diagnostics by representing directed traffic flows between pairs of networked nodes in a compact manner. In detail, a traffic matrix is a two-dimensional array where each row (column) corresponds to a node, and each cell contains the value of traffic flow between the row and column nodes, obtained by aggregating link load measurements over a sampling time interval. The collection of this data serves the purpose of enacting strategies for infrastructural enhancement and traffic engineering in a conscientious, informed way. However, such information can often be only partially available: this is the case, for example, of networks dealing with massive volumes of traffic such that telemetry operation may put heavy computational strain on the measuring devices, causing these to suffer degradation of performance for their networking functions (e.g. forwarding throughput). Simply scaling the computational resources up to satisfy the requirements' overhead is not always feasible and is an expensive solution. This is a reason for the prominent interest in the problem of traffic matrix estimation and completion, namely the problem of inference of traffic flows via statistical or Artificial Intelligence (AI)-based techniques. This work focuses mostly on the category of AI and data-driven as a regression tool to tackle the aforementioned problem. At the same time, we aim to solve another issue that stems directly from the intrinsic nature of AI: its lack of human interpretability. The main contribution of this thesis is the comparison (in terms of different error metrics) of several models for traffic matrix completion and the explanation of the decision process of the black box-like techniques via eXplainable Artificial Intelligence (XAI) methods such as saliency maps. Experimental results show that the accuracy of Machine Learning (ML)-based and statistical models highly depends on the set of network conditions, i.e., the dataset used. The complexity of traffic and the absence of clear patterns alter the ability of the model to generalize the findings among different network traces. On the other hand, the study of the model decision process via XAI demonstrates that models are majorly influenced in their inference process by the surrounding square around the missing traffic matrix cell.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Traffic flows statistics are a key product of network diagnostics, and traffic matrices are a common choice for gathering those statistics in an efficient and intuitive representation; in detail, a traffic matrix is a bi-dimensional array where columns and rows correspond to the nodes which comprise a network infrastructure, and their meeting point indicates the traffic flow between the two nodes, sampled in a time interval of choice; each node might correspond to a single or multiple aggregated network devices, with the latter choice being sensible in large scale contexts. In other words, given a matrix M, for each pair of nodes i,j, the cell M_{ij} corresponding to those indices contains the volume of the directed flow $f_{i \to j}$. These matrices find a multitude of uses in the fields of network and traffic engineering (e.g. capacity planning, congestion avoidance and route optimization).
With that said, while conceptually simple, traffic matrices are not always easy to obtain in practice: solutions for TM computation from measured traffic data do exist, and follow a variety of approaches that rise to the practical challenges (as exemplified in [1]), but they are not always applicable. This is because of different possible reasons, as stated in [2]:

- network devices may lack support for measuring protocols.
- network devices may suffer the computational overhead of frequent measuring under heavy load conditions; conversely, lowering the frequency of measurements reduces the quality of the information used to build the traffic matrices.
- upgrading the infrastructure so as to solve the aforementioned issues is expensive.
- even if we assume the network to be adequately equipped in both hardware and software resources, protocols for data collection generally rely on connectionless, unreliable transport mechanisms (e.g. SNMP on UDP) without retention,

therefore some diagnosed data may be lost during transmission through the network.

Because of these causes, the problem of TM estimation has seen a lot of interest around the years, and the reason why many kinds of techniques have been devised for both future matrix estimation and matrix completion.
Our focus in this thesis lies on Artificial Intelligence architectures employed as regression methods for TM completion, because of the remarkable results and attention they have been drawing as of late. In particular, these architectures (along with general Machine-Learning), have been following an upward trajectory for several years now. To translate words into numbers, the total amount of AI related publications produced by the efforts of Research & Development units all over the world has doubled between 2010 and 2021, as reported in [3], going from 160 thousands to 330 thousands, and interestingly enough this growth has seen an increment in steepness between 2016 and 2017 which holds steady to this day. This positive tendency is not only encountered in academic environments, but also finds correspondence in terms of yearly corporate investments fueling R&D departments, meaning that industries are more likely than ever to foresee the potential behind the adoption of AI. Consequentially, with a lot of interests being at stake, but also more importantly from an ethical standpoint, using Artificial Intelligence as black boxes in decision processes is not feasible, especially in critical contexts. Motivated by this conundrum, institutions for regulation and standardization have been working during the last few years towards the definition of governance paradigms that would allow for a more conscious usage of these tools. Hence the current direction of practical AI applications is to try and equip decision models with explanation suites, in order to enable the principles of trustworthiness[4], accountability and transparency[5].
This leads us to the ultimate goal of our work, that is, the proposition of several ad-hoc AI models for TM completion, their comparison with other solutions drawn from recent literature, and most importantly the application of eXplainable Artificial Intelligence techniques in order to understand the "reasoning" of the models.
As expected, the problem proved itself to be challenging because of the non immediately recognizable and inconsistent spatial patterns of traffic matrix data, for example when compared to other simpler bi-dimensional data use cases of AI such as RGB images; therefore, because avoiding under-fitting and over-fitting showed to be difficult for the neural networks, this inconsistency in the data translated to inconsistent results across the board in our tests. Conversely, a consistent pattern has been observed in our attempt to visualize the decision process of the networks via saliency maps: these collectively showed that, on average, the models assign a higher degree of importance towards the estimation to a subset of matrix entries, which are mostly localized in the close proximity of coordinate of the missing data to be predicted. Generally speaking, because entries

in the matrices which are close to one another might constitute flows with little to no real correlation, and vice versa distant entries in the matrices might represent flows with strong mutual influence, we observe that a more thoughtful placement strategy of the flow data might assist the networks in using the most relevant data in a more focused manner, hence improving performances; in particular, our hypothesis for future works is that it might prove beneficial to structure the data in such a way to keep distances between flows in the matrices inversely proportional to their correlation, since as we already mentioned, the models on average tend to privilege the surrounding cells of the missing value(s) when making an estimation.

This thesis is structured as follows. Chapter 2 offers an overview of related literature, exploring the spectra of both matrix estimation algorithms and XAI techniques, while also mentioning some of their applications to practical use cases. Chapter 3 provides high level information about the architecture of the three proposed models that constituted the test subjects for the XAI methods chosen for the analysis, and explains what these methods consist of at an intuitive level. Chapter 4 presents a formal definition of the problem of Traffic Matrix Completion. Chapter 5 briefly presents notions about the datasets of choice and the preprocessing schemata, lists the metrics employed for the comparison of the performance of models, and displays the results of our work, in terms of comparison of prediction performance in variable conditions, training and prediction times, and finally explanation of the rationale behind the estimations through visualization of saliency maps. Chapter 6 summarizes the objectives of our work and the steps we followed in order to meet them, while also suggesting a starting point for reflection on future related works.

Chapter 2

Related Work

In this chapter we review some of the existing literature about Traffic Matrix Completion and eXplainable Artificial Intelligence, so as to define the scope into which this thesis ought to be inserted.

2.1 Matrix Completion

Studies in field of matrix completion are abundant, and follow a variety of different approaches applied to disparate use cases. Considering the general problem of matrix completion and also focusing on the particular issue of network traffic inference, we proceed to mention some of these works and the relative proposed methodologies. One of the most explored approaches to the problem of completion is to solve the optimization problem of low rank matrix (and by extension tensor) completion; its formal definition is as follows:

$$min\, rank(E),\ s.t.\, E_{ij} = M_{ij}\, \forall (i,j) \in \Omega \tag{2.1}$$

where E is an estimated matrix, M is the matrix to complete, Ω is the set of all indices of the matrix; in other words, the objective is to find a matrix E that closely approximates the target M, while also minimizing the rank of the former; the main issue with this approach is that finding a solution to 2.1 is NP-Hard, therefore the main challenge in this case has been to reduce the complexity of the computation. Several workarounds have been devised for this purpose, mainly consisting in the substitution of the optimization criterion function (rank(.)) with surrogates that allow for faster resolution with lower complexity and comparable results. The problem statement in 2.1 thus becomes:

$$min\, f(E),\ s.t.\, E_{ij} = M_{ij}\, \forall (i,j) \in \Omega \tag{2.2}$$

where f(E) is the surrogate of choice, replacing rank(E). Common choices for the surrogate function are tensor nuclear norm and Schatten-p norm. In [6], a smooth

low rank tensor tree completion algorithm is described; this is a member of the populous family of the nuclear norm minimization approach[7] and introduces a solution to 2.1 that aims to minimize, rather than the sole nuclear norm, the sum of the latter with total variation norm [8], in order to improve the reconstruction capabilities in the cases of tensors possessing the property of smoothness (e.g. images), while also forgoing the need to estimate a bound for the tensor rank bound.
Some other variations of norm based approaches are exhibited in [9] and [10]; in the first, three alternative approximations of f(E) are implemented, namely EPT, MCP and SCAD, used in conjunction with a divide et impera approach by reducing the optimization problem into several, more manageable subproblems (approach known as Block Coordinate Descent), while in the second two ulterior alternatives are presented, labeled NS-LRTC and S-LRTC, whose purpose is to improve reconstruction accuracy close to tensor boundaries, which tends to be a weakness of smoothness-aware functions.
LMAFit[11] is an efficient solution for a wide array of generic matrix completion and estimation problems, and works by introducing a low rank matrix factorization model which aims to reduce computation time with relation to nuclear norm minimization approaches[7] by avoiding the computation of the nuclear norm entirely; although it possesses drawbacks such as reliance on the provision of an initial rank estimate, and lack of a guarantee to achieve a global solution, LMAFit is proven to be as empirically reliable as norm based approaches, while being much faster.
STTC-CP[12] employs a tensor representation for time series of traffic matrices, thereby enabling the application of the CANDECOMP/PARAFAC ([13], [14]) tensor decomposition technique; by exploiting the lower dimensional latent structures hence produced and by factoring the feature of temporal stability of the OD flows in temporally adjacent traffic matrices, this method aims to better capture the structural properties of traffic data and therefore to exhibit superior performance by comparison to 2-dimensional matrix based algorithms in presence of high loss rates.
STGM[2] uses traffic matrix data, partitioned and modeled as a set of gaussian distributions via spectral clustering (based on spatial affinity), as an adjuvant to a linear regression algorithm in order to improve the performance of the latter, especially under the conditions of heavy data loss.
NiTMC[15] proposes a way to exploit the potential relationship between the problems of traffic volume interpolation and network anomaly detection for the purpose of solving both in a more efficient manner, by making use of the low rank property and temporal characteristics of traffic matrices; this approach achieves improved estimation accuracy in the presence of complex (e.g. non simply gaussian) noise distribution patterns in the data when compared to models that ignore the issue of anomaly detection.

With [16], Roughan et al present the application of Gravity[17] and Tomogravity[18] models for traffic matrix estimation, to drive traffic engineering operations; in particular, estimated traffic matrices, used in conjunction with traffic optimization algorithms, are shown to provide solutions for congestion avoidance in some cases close to optimal.
As for AI based approaches, convolutional networks are often employed in this case because of their compatibility with multidimensional data. R-CNTME[19] is an example of a network made of convolutional, pooling and fully connected layers, built to tackle the traffic matrix completion problem under the often deal-breaking assumptions of limited, sparse and noisy training data, which showcases how, by using the global spatial correlations between OD flows extracted via convolution, it is possible to achieve superior performances with relation to architectures that only take into account correlations between a singular prediction target flow and the others.
In [20] a CCAE architecture, namely a cascaded autoencoder built with (mostly or exclusively, depending on the case) convolutional filters is described in its use case of electric load matrix data recovery; since the measured data here represents a time series, this is a case of revisiting mono-dimensional data, by exploiting its periodicity and transforming it into bi-dimensional data so that horizontally adjacent data constitute measurements in proximate time points, and vertically adjacent data instead constitute measurements at corresponding moments within successive periods; such a representation allows the CCAE to shine: because of its capability to extract the correlations between cells in multidimensional data, the network makes use of a larger amount of information when compared to traditional mathematical estimation methods, which only use spatial neighbors of the missing data point.
In [21] a ConvLSTM architecture (first introduced in [22]) for future traffic matrix prediction is presented, that is, a combination of CNN and LSTM models capable of extracting and modeling the spatio-temporal features of historical traffic data and ultimately generate estimated traffic matrices in successive time-steps; in this specific context, the objective of the architecture is to not only to produce projections of future traffic, but to do so in the case of sparsely available historical data, therefore of traffic matrices being comprised of mostly predicted, non directly measured values.

2.2 eXplainable Artificial Intelligence

With the complexity and opaqueness of AI models increasing enormously since their original conception, along came a surge of interest in the topic of AI interpretation; this interest, which saw a steady increase in the last decade[23], has been the

propelling force behind the development of several XAI solutions of assorted nature. Barredo Arrieta et al, provide in [23] a theoretical introduction to XAI and its core concepts, while also making the case for its necessity and providing a comprehensive and detailed taxonomy of recently developed solutions, for both general Machine Learning and Deep Learning.
Here we cite a number of these solutions and their application to various use-cases. A common technique for AI interpretation is the visualization of the attention levels of a model to features of a particular input through heatmaps: these attention levels can be computed in a multitude of ways, which usually involve extrapolating information about how the input is processed from the inner layers of neural networks. We report some of these techniques and relative use cases. In [24], Amarasinghe et al present a framework for explainable, Deep Neural Network based anomaly detection, whose purpose is to offer insight on the anomaly classification made by the network in terms of estimation confidence, relevance score of the input features towards the outcome of the classification, and by constructing textual, human friendly descriptions of the anomaly. The main tool used for implementing the framework is the Layer-wise Relevance Propagation technique[25], whose functioning consists in a backwards (output to input) layer to layer mapping of relevance attribution by each neuron, allowing to distribute the relevance values over the input sample features, all the while abiding by a total relevance conservation law; to explain this method in simpler terms we can consider the example of image classification: the objective is to perform a "pixel-wise decomposition" of a classification decision, that is, construct a heatmap of contribution over the pixels of the classified image, where each pixel is considered as an individual feature; each neuron of the last layer, for a given classification, yields a particular value which is function of all its connections with neurons of the previous layer and corresponds to the contribution of said neuron towards the output. By traversing the network in a backwards fashion, and iteratively computing the relevance factors for each neuron of each layer as function of neurons of successive layers, the input layer is eventually reached, and thus a relevance distribution over the pixel is obtained.
Grad-CAM[26] is an activation based technique for visualizing class activation maps of CNN-based models in a highly discriminative way; this means that, similarly to LRP, it produces heatmaps which highlight the classification-deciding features (more specifically, those that have a positive impact towards a particular output class), and does so by exploiting information extracted from the last convolutional layer of a CNN (namely, the gradient of the score of a class with respect to the feature maps of the layer); the choice of analyzing information from the last convolutional layer, rather than any other, be it convolutional or fully connected, is because the former reaches, inside these sort of architectures, the highest level of feature abstraction out of all convolutional predecessors, while also retaining the spatial information which is lost in the fully connected layers.

In [27], Chen et al make use of Grad-CAM to visualize the attention of a CNN classifier over time-frequency spectral images generated by vibration analysis of rolling elements bearing (REBs); in this context, given that the AI model is capable of detecting and classifying different faults of the REBs by analyzing said spectra, Grad-CAM generates heatmaps that represent the different level of attention over the frequency bands of the vibration, therefore highlighting the correlation (as seen by the CNN) between a particular kind of REB fault and the frequency bandwidths of the vibration it produces.
Another potential application of CAM to CNN classifiers can be seen in [28]; in this case the network is not built to deal with bi-dimensional data, but rather constitutes a network packet flow classifier, thus dealing with mono-dimensional structures; CAM provides added value to the model by visualizing the importance of each byte of a flow during classification.
Zheng et al[29] showcase the potential of visualization techniques, namely Saliency Maps and Activation Maximization[30], by explaining the inner workings of a neural network trained for job scheduling in a cloud computing environment (a concise overview of the two methods is provided in 3.2.1 and 3.2.3); by the means of these two methods, it is possible to understand how the state of the platform's resources together with the the resources' demand of a job influence the scheduling outcome (with saliency maps), and what profile of resources' demands leads he model to favor a job over the others (with Activation Maximization).
Another option for for interpretability is sensitivity analysis, whose application to neural networks was first presented in [31]; in essence, it represents a group of methods whose objective is to observe and visualize through map representations how a model responds to arbitrary perturbation of input (for example, a translation, rotation, or mirroring of a figure in an image); as such, these method share a common root with DeepLIFT and CAM variations, with the latter two being respectively gradient based and activation based, and the former being perturbation based.
A commonly encountered sub-type of sensitivity analysis is occlusion sensitivity analysis: in this case, the input is perturbed by partially removing some of its information (e.g. replacing portions of an image with non informative data), and for each output node (classes or regression output) the variation of the score function is registered; by applying occlusion over the portions of the data iteratively, it is possible to compile those score variations over an attention map. In [32], Uchiyama et al demonstrate an application of occlusion sensitivity analysis to the case of a CNN video classifier.
One more possibility for producing explanations is local approximation of the output function of the complex AI models with simpler, naturally interpretable algorithms. This solution entails, given some particular input and the relative output by the black-box network, to build surrogate models (e.g. decision trees) which exhibit congruous decision making strategies, although only for data whose features are

positioned in the surroundings of the input data; once the approximative model is generated, insight about its rationale, which is also locally representative of the opaque network, is extracted. Some examples of this approach are mentioned in the following paragraphs.

DeepSHAP[33] belongs to the family of additive feature attribution methods, approaches rooted in game theory consisting in the attribution of effect values to features of data which sum to output function of the model to explain; the framework is built by using a combination of SHAP values as feature importance metric, representing the change in expected prediction of a model when conditioned to the feature itself, and DeepLIFT[34] as a technique to approximate the hard-to-compute SHAP values, and at the same time guarantee local accuracy in approximation of the model (in other words, the approximation of the model's behaviour against a certain input).

In [35], Nascita et al detail the usage of DeepSHAP[33] as a means of explaining the proposed framework for mobile traffic classification, MIMETIC-ENHANCED; in particular, the application of DeepSHAP here consists in the attribution of importance values to input data belonging to a mobile traffic bi-flow: these values represent a confidence measure of the model's assignment of a bi-flow to a particular class.

LIME[36] is an algorithm for model-agnostic, local explanation of both classifiers and regressors; in order not to be tied to any specific architecture, the algorithm works by building linear, interpretable, surrogate models around a data sample that the non-linear model associated to a certain output, and does so by generating samples via random perturbation of the sample to be explained; these perturbated data samples are fed to the non-linear model, which associates them with an output value (e.g. a class label); by observing the variation of the outputs and its correlation with the variation of the perturbations the surrogate linear model is finally generated.

The EXPLAIN-IT framework[37] is a proposal for explanation of unsupervised (e.g clustering) learning architectures, and relies on LIME for this purpose; its system is comprised of two steps: first, the dimensionality of the data is reduced to a summarized space via clustering (or similar techinques), then these clusters are analysed in terms of the characteristics separating them from one another; this second step is achieved in two sequential substeps, namely training a black-box model model (e.g. SVM) with higher discrimination power (w.r.t. naturally interpretable white box models like decision trees) to trace boundaries around the clusters and finally explaining the discrimination criteria by applying LIME on said trained model.

Doctor XAI [38] is an ad-hoc framework conceived for the explanation of the Doctor AI Recurrent Neural Network[39], built for the prediction of a patient's clinical events, given their medical history records; the approach followed by Panigutti et

al when devising the explanation pipeline is to generate local (that is, for each given input sample and corresponding prediction) surrogate decision models which are trained by using sets of data-points (both real and synthetic medical records) similar to the explanation target as training data correlated with the corresponding Doctor AI prediction as ground truth; the outcome of this process is the creation of a model which closely mimicks the behaviour of Doctor AI, and from which interpretable decision rules can be extracted.
The contribution of this work is to mount a comparison of performance for some of the traffic matrix completion solutions (or some variation of them) we have mentioned thus far, for the use case of single coordinate inference with different degrees of noise polluting the matrix data, and to exploit Vanilla Saliency maps in order to derive visual explanations for the predictions of three proposed deep learning models, in order to understand which OD flows are the most influential towards the regression output.

Chapter 3

Background

This chapter provides a general overview of the XAI techniques this work is focused on, and the AI models they were tested upon.

3.1 Artificial Intelligence Architectures

3.1.1 Convolutional Neural Network

Convolutional Neural Networks (CNN) are a subclass of Artificial Neural Networks (ANN), widely used for problems belonging to a wide spectrum, such as Computer Vision, Natural Language Processing, Object Detection and Segmentation, Image Classification and more [40]. These architectures, in particular, are attractive when dealing with image data (and by extension matrix data), because of the reduced computational complexity, number of training parameters, and over-fitting tendencies when compared to traditional ANNs, while retaining good performance [41]. A generic representation of the model can be seen in Figure 3.1: the core of a CNN is comprised of three kinds of layers:

- Convolutional Layers, which include a set of filters (or "kernels"), multidimensional parameters that act as operands for convolution operations with the input data, which in turn produce a set of "activation maps", and represent the key component that enables the layer to distinguish features in the data.
- Pooling Layers, which apply a form of downsampling to the input data (which are generally activation maps), reducing its dimensionality.
- Fully Connected Layers, inherited from traditional ANN architectures, made of units called neurons which are linked to each neuron of adjacent layers through weighted connections, whose values are repeatedly updated during the training phase.

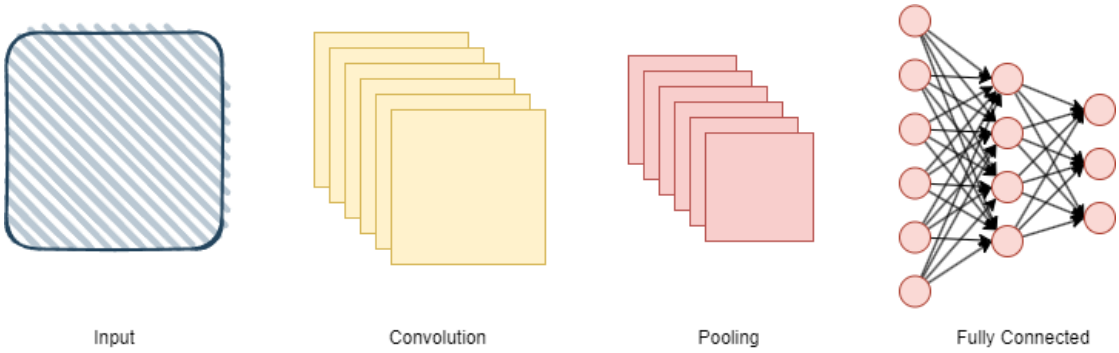


Figure 3.1: Architecture of a CNN

The first two kinds of layers are responsible for the extraction of features from the data, while the last one ultimately produces the output of the model. Since CNN is a prime and flexible alternative for working with matrix data, it is taken into consideration in this work as a regressive model for estimating missing traffic data.

3.1.2 Convolutional Autoencoder

Convolutional Autoencoders belong, as the name implies, to the family of autoencoders, with the peculiarity of employing Convolutional, Pooling, and Up-Sampling layers to build the encoder and decoder stages. Such architectures find applications in tasks like Image Denoising [42], Coloring, (De)Compression, being particularly suited for image data for reasons already stated in 3.1.1. The way they work is as follows: (see Figure 3.2)

- the encoder stage, consisting of a number of Convolutional and Pooling layers, is responsible of extracting the features from the input and compress the information into a representation known as "Latent Space".

- the decoder stage, consisting of a number of Convolutional and Up-Sampling layers, expands the data compressed inside the Latent Space up to the desired output shape.

Because of the spatial features abstraction capabilities of convolution based networks, these models are capable of generating results (e.g. images) that are devoid (to various degrees) of "spurious" portions of the input, such as visual noise, whose information may be discarded/replaced during the compression-expansion process, and for this reason, they represent a valid option as a solution to the Traffic Matrix Completion problem.

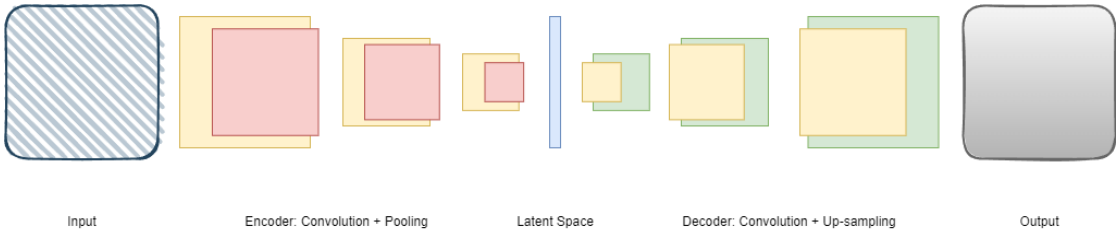


Figure 3.2: Architecture of a CAE

3.1.3 Adversarial Autoencoder

Adversarial Autoencoders (AAE) as introduced in [43], represent a form of Generative Adversarial Network (GAN), that is, a method for training generative models by pitching two neural networks against each other; the composition of a GAN is generally as follows (figure 3.3):

- A generator, which is trained to produce data samples that seemingly belong to the data space so as to fool the discriminator.
- A discriminator, which is trained to distinguish whether an input sample belongs to the data space ("true" sample) or not ("fake" sample).

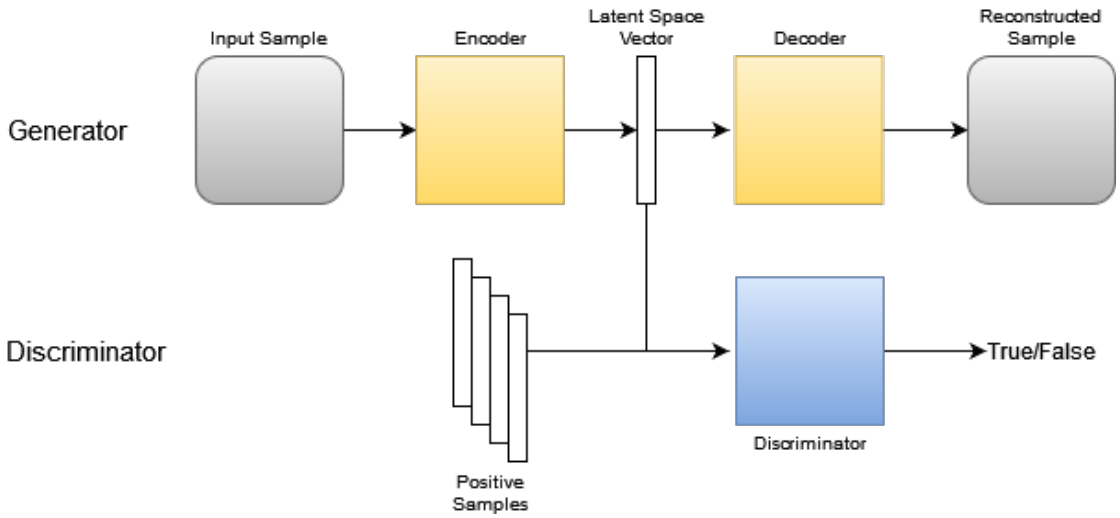


Figure 3.3: Architecture of a AAE

In the specific case of AAEs, the generative model is built as an autoencoder. The crucial aspect of these architectures is how the training procedure is led in several steps for each data batch:

- the AE is trained in a "traditional" fashion, in order to minimize the reconstruction error on the data samples.
- the discriminator is trained on a mixture of true and generated latent space vectors, so as to learn to discriminate the positives from the negatives.
- the encoder portion of the AE is trained to produce latent space vectors that the discriminator can classify as positives.

3.2 Explainable Artificial Intelligence

3.2.1 Saliency Maps

Saliency Maps were first introduced in [30], as a way to visualize convolutional classification models' spatial support for a given class in an image. The idea behind vanilla saliency is to rank the influence of single pixels of an image over the score function (of a class for classifiers, or, in the regression case, for value variation) of the output layer of a neural network. In further detail, the saliency values for each pixel are computed by differentiating the score function of choice with respect to the input image. The result is a map having the very same size as the image, where each cell constitutes the degree to which the corresponding pixel of the image is influential in defining the score value.

3.2.2 Occlusion Sensitivity Analysis

Much like saliency maps, occlusion sensitivity analysis lies in the category of post-hoc, local interpretation methods. This approach consists in perturbing parts of the input ("occluding" them) to be fed to the model, by replacing them with non-informative values, and measuring how much the prediction score corresponding to the occluded input distances itself from the original prediction relative to the unmodified matrix. This distance is interpreted as the importance score of the occluded part of the image with respect to the prediction: the higher the difference, the more influential is that portion of the input. These importance scores are ultimately compiled into an importance score matrix, of the same size as the input matrices, which is the result of the analysis. The occlusion process may be carried out on an arbitrary granularity: masks may cover portions of an image sizing from even a single pixel up to large portions of the input, and should aim to occlude significant portions of the sample (e.g. the set of pixels showing a class's peculiar feature) to achieve useful results. Although this methodology was taken into consideration, no consistent, indicative information about the models could be derived from it, therefore its results have been omitted.

3.2.3 Activation Maximization

The purpose of Activation Maximization, as thoroughly described in [44], is to visualize the "behaviour" of hidden layers inside of deep models. This is achieved by solving (locally, see [44]) the optimization problem of finding the input which maximizes the activation of the units of the model. In other words it is a matter of artificially building an input sample that the network will, with the utmost confidence, translate to a particular output class or value. Although very interesting insight may be deduced when applying this sort of technique to particular fields (i.e. computer vision, image classification), in our case the results were not intuitive and as such have been omitted as well.

Chapter 4

Solution

The traditional formal definition for the problem of Traffic Matrix estimation was pioneered by Y. Vardi in [45]; we will use this definition, known as the tomography model, as a reference to frame our case. The statement of the problem is as follows: given a set of directed traffic flow volumes, measured from the L links of a network with N nodes, sampled in a given time interval, we intend to compute the amount of traffic running between the $C = N(N-1)$ Origin-Destination (OD) couples of the network. We work under the assumption that the network is a strongly connected directed graph, meaning that for any pair of nodes i, j $\in N$, there exist two paths $p_{i \to j}$ and $p_{j \to i}$ that connect said nodes in both directions. At a particular sampling time t, we identify three main components in our formulation:

- X_t, a column vector sized Cx1 containing the measurements of the OD flows between each pair of nodes; essentially, this is a traffic matrix.
- Y_t, column vector sized Lx1 containing the directed flow volumes traversing each link of the network.
- the routing matrix A, sized LxC, containing information about the network routing configuration, and defined as follows:

$$\begin{cases} A_{lc} = 0 & if\ link\ l \notin p_{i \to j} \\ A_{lc} = 1 & if\ link\ l \in p_{i \to j} \end{cases}$$

  where i and j are the indices of the nodes constituting the directed pair c.

Starting from the aforementioned components, a linear relationship between the the OD traffic flows and the the volume of data going through the links of the network can be defined in these terms:

$$Y_t = AX_t \tag{4.1}$$

Our objective is finding X_t, for a given Y_t and A, hence we aim to solve the inverse of the linear problem 4.1. Unfortunately, in most real networks, the number of links L is way smaller than the number of OD pairs C, therefore:

- matrix A is not invertible.
- the inverse problem is severely underconstrained.

Several solutions have been devised in order to solve this issue (see [45], [18]), for example consisting in posing additional constraints to the equation in order to turn it into a determined system, or by using approximations models.
Our approach, while sharing the same objective of estimating the OD flows in X_t, does not involve using either the routing information contained in A or the link loads information from X_t; instead, we assume partial information about the OD flows to be available, and we seek to fill the gaps in the data by leveraging the capabilities of several algorithms to "understand" the spatial relationships which run among the flows themselves. We can recognize three pieces in the formalization of this new problem:

- matrix $\tilde{X}_t$, sized NxN, containing the partially measured information about the OD traffic flows.
- matrix X_t, sized NxN, containing the full information about the OD traffic flows.
- function $f(.)$, representing the non-linear function describing the matrix completion algorithm.

By putting all the pieces together, we obtain:

$$X_t = f(\tilde{X}_t) \tag{4.2}$$

In essence, rather than avoiding the measurement of traffic flow data directly from the network, and instead computing it by exploiting its relationships with other easy to obtain information, we accept to sample a subset of that data, and estimate the missing flow volumes through regression techniques.

Chapter 5

Results

The following chapter provides some details about the data used in our tests, the preprocessing operations and successively expands upon the used architectures in terms of training and prediction time, prediction performance and prediction visualization using the aforementioned XAI techniques.

5.1 Datasets

Three different datasets were chosen for this study:

- the Abilene (2004) dataset, featuring 48386 samples, sized 12 by 12, measured in five minutes intervals.
- the commonly used Geant (2005), sporting a total of 11460, 22 by 22 matrices representing traffic demand spanning over four months, with a granularity of fifteen minutes.
- a set of matrices derived from the network traffic traces recorded from the WIDE network and made publicly available by the MAWI group. Ten consecutive traces, spanning over two hours and thirty minutes from samplepoint-F were considered, dated 2020. Traffic matrices were generated by aggregating traffic by address prefix, with a one second granularity, and by filtering smaller flows to keep the matrices' size at a reasonable level. The result is a collection of 9010, 24 by 24 matrices.

5.2 Metrics

Two different metrics were chosen for comparison and evaluation purposes:

- Normalized Root Mean Squared Error defined as:

$$NRMSE = \frac{N}{\sum_{i=0}^{N-1} y_{t_i}} \sqrt{\frac{\sum_{i=0}^{N-1} (y_{t_i} - y_{p_i})^2}{N}} \tag{5.1}$$

- Normalized Mean Absolute Error defined as:

$$NMAE = \sum_{i=0}^{N-1} \frac{|y_{t_i} - y_{p_i}|}{|y_{t_i}|} \tag{5.2}$$

where:

- y_{t_i} and y_{p_i} represent respectively the i-th observed value and i-th corresponding predicted value.
- N represents the number of considered matrix samples.

NMAE represents an immediately intuitive representation of the magnitude of the average estimation error, NRMSE instead puts more emphasis on the occurrence of large errors, allowing to evaluate performance from a different standpoint. Normalized metrics were preferred in order to take into account differences in the normalization schemata and make the measured values inter-comparable.

5.3 Benchmark Algorithms

In order to properly evaluate the performances of the proposed AI models, tests and measurements were additionally made on a set of other techniques fit to solve the problem of matrix completion. Observations including some practical remarks about the usage of all the benchmarked models follow in the next subsections.

5.3.1 Convolutional Autoencoder

Although somewhat redundant with 5.3.3, we explored this option via tuning so as to better understand how details in the architecture influenced the end results and whether using a shallower, simpler model could bring any benefit. In details, mixed deep and convolutional autoencoders and fully-convolutional autoencoders were the object of study, and they mostly yielded similar results, with the fully-convolutional implementations pulling slightly ahead. The implementation of these architectures substantially constitute de-noisers for the input matrices, where noise is to be considered to be the (set of) missing values. These models produce whole matrices as output and require no further pre-processing of inputs (no shifting), unlike the proposed implementation of CNN, because they are intrinsically capable of

estimating more than one value at a time for a single input, also meaning that a single trained model can be used to estimate values a different positions at the same time.

5.3.2 Adversarial Autoencoder

The proposed implementation of AAE, very similar on the autoencoder side to 5.3.1, was analysed in order to determine whether adding an adversarial component to the simpler architecture could improve its performance in the case study.

5.3.3 Cascaded Convolutional Autoencoder

A (partial) implementation of the CCAE architecture proposed in [46] was tested, only missing the tail end of the model, namely the reshaping and de-normalizing layers, which are unneeded in our case. All the parameters and the rest of the overall architecture of the network were kept as described, except for the input layer of the network, which was changed to fit the case of the chosen datasets. The metrics were calculated by extracting the predicted value(s) from the output matrices and comparing them with the corresponding truth.

5.3.4 Convolutional Neural Network

Among the AI solutions presented in this section, the employed CNN implementation is the simplest, having the smallest number of trainable parameters.

5.3.5 Convolutional Long Short-Term Memory Network

ConvLSTM models are generally employed when capturing correlations of data over time is of essence. The presented case study, however, pertains the capturing of the sole spatial features of matrix data, therefore the tested implementation may be considered a degenerate version of a ConvLSTM, where only a single input time step is considered when producing an output.

5.3.6 k-Nearest Neighbours

Several values were tested for k, ranging from 5 to 20, but performance differences were marginal at best so we settled for using 5 uniformly; distances were computed as Euclidean distances. The model was fed with full, normalized matrices, reshaped into mono-dimensional arrays in order to be fed to the regressor.

5.3.7 LMAFit

The approach described in [11] was taken into consideration. Because the algorithm requires an estimation of matrix rank as input (k), for each coordinate we empirically determined the k value which minimizes NMAE, and proceeded to observe that in almost all cases such value also minimizes NRMSE. Because of this approach, all the reported metrics correspond to the model working in the best condition possible, for each coordinate.

5.3.8 Smooth Low Rank Tensor Tree Completion

Another tested solution was STTC, an algorithm based on low tensor tree rank and total variation minimization, as described in [6]. Since this approach was originally conceived as an RGB image completion technique, some additional pre-processing to the normalized traffic matrices was applied, in order to be compliant with the required input format. In particular, because traffic matrices only span over two dimensions, rendering them akin to single-channel (grayscale) images, each matrix was artificially expanded by adding two additional channels, each one an exact copy of the original matrix. Estimated values were ultimately extracted from the output structure, corresponding to the completed input tensor (therefore, at each missing values coordinate, three identical predicted values are found along the channels axis). A value of 0.0005 was used for the ρ factor.

5.3.9 Smooth Low Rank Tensor Completion

The last model we introduce in this comparison is the one presented in [10], we refer to it as LRTC. Like STTC (see 5.3.8), this is natively meant to work on images, therefore the same steps for adding two more modes to the traffic matrices were taken. Also similarly to LMAFit, an initial rank estimation is needed, so we opted for the same empirical approach of determining the best estimation for each case with respect to NMAE and NRMSE.

5.4 Preprocessing

Different data normalization schemata were used, in order to accommodate the peculiarities of each model. Table 5.1 provides an overview on the ranges the data was scaled into and the numerical values representing missing data points in the matrix.

In this regard, CCAE follows the guidelines from [46], LMAFit and STTC are tested as if working on an image completion problem (no missing data values are reported, since both algorithms rely on different masking approaches, rather than

	interval	missing data value
AAE	[1,10]	-1
CAE	[1,10]	-1
CCAE	[0.01,1]	0
CNN	[1,10]	-1
ConvLSTM	[0,1]	-1
k-NN	[1,10]	-1
LMAFit	[0,1]	/
LRTC	[0,1]	/
STTC	[0,1]	/

Table 5.1: Data normalization and masking schemata

placeholder values, to detect missing entries), ConvLSTM uses a very common normalization schema for the problem at hand, while AAE, CAE, CNN and kNN adopt a completely different scale. The reason for this choice is the distribution of values in the dataset affecting the performance of those models when using a [0,1] normalization; rescaling the matrices in the [1,10] range instead, led to better training stability for the neural networks (both training and validation loss kept a mostly monotonic trend, with less oscillation with reference to [0,1] normalization) and yielded improved performance metrics (better NMAE across the board). Table 5.2 reports the per-dataset maximum, minimum, and average NMAE values of models using [0,1] and [1,10] intervals. It is worth mentioning that the largest gaps in terms of metrics' manifest when models work with the Geant and Mawi datasets, whose data distributions are less uniform than Abilene's, and whose spectrum of values is a lot wider.

	[0,1]			[1,10]		
	Min	Max	Avg	Min	Max	Avg
Abilene	9.75%	41.29%	23.25%	3.12%	15.23%	9.58%
Geant	7.44%	103.79%	82.39%	0.19%	12.13%	5.81%
Mawi	32.01%	149.72%	88.22%	3.02%	26.84%	12.25%

Table 5.2: NMAE comparison between normalization schemes

In addition to data normalization, CNN and ConvLSTM in particular also require another step for the data to take before inference: since the models only output a single value for each input matrix, and not full matrices like the tested autoencoders, we aimed to provide the networks with different inputs for each coordinate of the same matrix to predict; this is necessary because feeding the networks with a

particular sample missing several values as-is, would always necessarily bring the models to return the same exact prediction. This issue was solved by rearranging the matrices, in order to place the inference target in the center, thus differentiating the input according to which position we are trying to fill.

5.5 Performance

The results of our benchmarks are shown in tables 5.3 to 5.14, and in figures 5.1 to 5.6. All tests were led by referring to the same trios of coordinates (i.e. the position of the value to predict inside a traffic matrix), namely (1,3), (8,9), (11,7) for Geant, (3,4), (2,8), (7,9) for Abilene and (1,8), (11,2), (19,14) for Mawi, which were chosen randomly.
Metrics in 5.3, 5.5, 5.7, 5.4, 5.6, 5.8 refer to models producing predictions when only the value of the indicated coordinate is missing.
Charts 5.1, 5.3, 5.5 and 5.2, 5.4, 5.6 represent the performance trends of the models when predicting the value of a given coordinate, for increasing percentages of missing matrix data. The values were obtained by computing NMAE and NRMSE over the predictions of the test sets, for each percentage of values in the matrix missing, and for each of the considered coordinates. The set of missing coordinates for each percentage was obtained via random sampling. The measuring process was repeated for up to 20 times for each model, and finally the displayed values were produced by computing the mean of the values over the 20 iterations and among the three coordinate of each dataset. From the same process, 90%-confidence intervals for each percentage were also derived. Because we decided to set the upper limit of NMAE and NRMSE to 1 in these charts, models that yielded poor performances in some of the tested conditions may not be represented.
The values in 5.9, 5.10 and 5.11 show the training time for each of the trainable models along with the number of epochs, and how the number of parameters of neural networks heavily impacts fitting times.
The measurements in 5.12, 5.13 and 5.14 were computed by averaging the prediction time of the models over a subset of the Geant, Abilene and Mawi datasets, respectively. Details about the relative performances of each model are listed in the following subsections.

5.5.1 Results - Geant Dataset

Table 5.3 and chart 5.1 display the NRMSE values computed under different conditions for each model; from these, we can identify the disparities in performance between the solutions:

	(1,3)	(8,9)	(11,7)
AAE	8.0476e-03	3.0087e-02	1.0232e-01
CAE	5.4832e-03	2.6196e-02	8.4371e-02
CCAE	1.9334e-01	3.4919e-03	7.8917e-01
CNN	1.5464e-02	8.9110e-05	5.5814e-02
ConvLSTM	9.0470e-01	5.7056	9.4017e-01
k-NN	3.0486e-04	5.1705e-08	8.4772e-02
LMAFit	9.5433e-01	3.0438e-01	9.9954e-01
LRTC	4.1810e-01	1.8572e-01	4.1864e-01
STTC	2.6795	9.3372e-01	1.7522

Table 5.3: Normalized Root Mean Squared Error values (Geant)

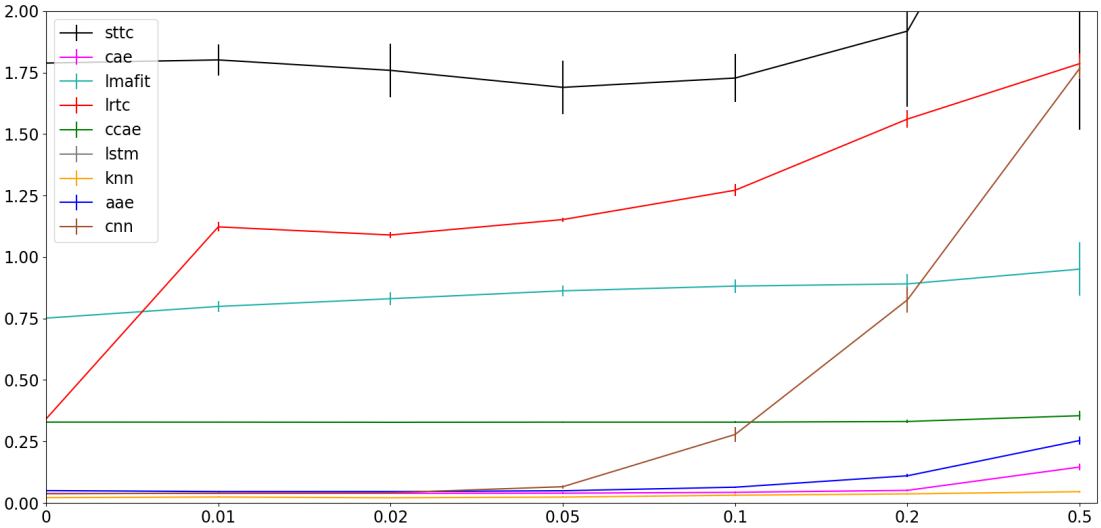


Figure 5.1: NRMSE for increasing noise ratios (Geant)

- kNN, CAE and AAE present similar NRMSE values for lower noise ratios (NR), but they diverge for higher numbers of missing values; in particular, AAE and CAE, which follow an almost identical progression up to NR = 10%, start performing worse than kNN for NR $\geq$ 20%; on the other hand, kNN has stable performance even for high NR ($\sim$ 50%)

- CNN performs remarkably well for low noise conditions ($\leq$5%), matching the AI/kNN group, but undergoes a substantial decline for NR $\geq$ 10%, becoming one of the worst overall performers, denoting lack of robustness against perturbations in the input conditions.

- CCAE, although producing on average more large (or larger in magnitude) errors than the models described so far (bar CNN with high NR), is almost unaffected by growing NR.
- ConvLSTM produces by far the worst performance out of the batch, and as such is not displayed in the y-axis range.
- LRTC, LMAFit, STTC all fall behind the AI/kNN group, to different degrees; in further detail, LRTC has the best NRMSE performance in absence of noise, but follows the sharpest fall-off when even the slightest amount of perturbation is added, LMAFit is on average the best and stablest performer of the three, and conversely STTC is the worst.

	(1,3)	(8,9)	(11,7)
AAE	0.0648	0.1020	0.1164
CAE	0.0580	0.1438	0.1098
CCAE	0.3248	0.0451	0.5159
CNN	0.0073	0.0078	0.1221
ConvLSTM	1.5227	1.7935	1.2675
k-NN	0.0075	0.0010	0.0817
LMAFit	0.8022	0.6643	0.9975
LRTC	0.6680	0.4584	1.7429
STTC	0.8140	0.9507	0.9657

Table 5.4: Normalized Mean Absolute Error values (Geant)

Chart 5.2 show values of NMAE only up to 100%, which we deemed a reasonable threshold of acceptable performance. At a glance, the progressions follow the same course seen in 5.1, with some differences:

- kNN undisputedly produces the best predictions under any condition.
- CCAE actually performs closer to AAE and CAE for NR $\geq 20\%$.
- LMAFit is the best out of the three low rank optimization based algorithms, overtaking (on average) LRTC under zero noise conditions; this is due to LRTC yielding consistently inaccurate estimations for the coordinate (11,7).

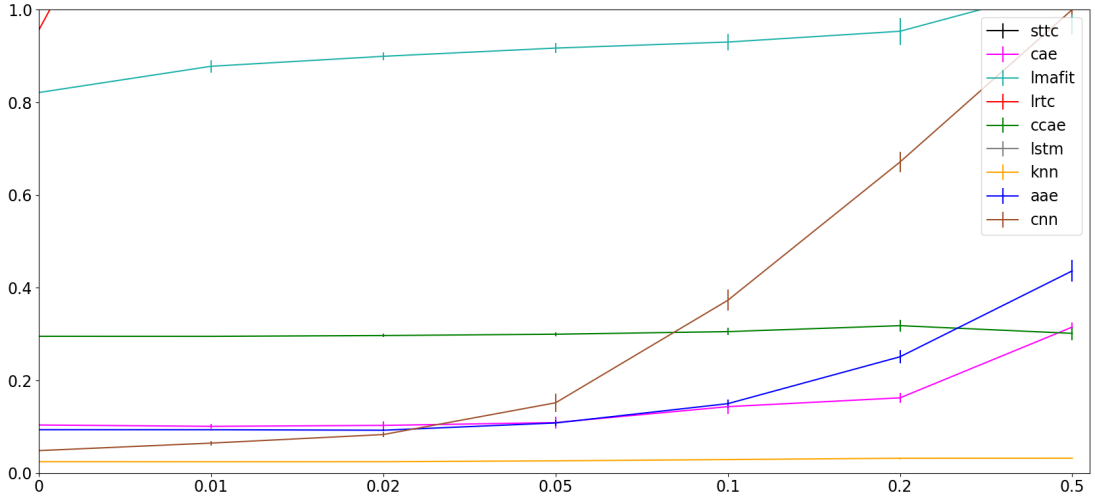


Figure 5.2: NMAE for increasing noise ratios (Geant)

5.5.2 Results - Abilene Dataset

	(3,4)	(2,8)	(7,9)
AAE	2.4687e-02	4.7033e-02	3.3800e-02
CAE	2.5791e-02	3.7324e-02	2.1194e-02
CCAE	4.4652e-02	1.5951e-01	6.7021e-02
CNN	1.6813e-02	1.2987e-02	1.0694e-02
ConvLSTM	2.0497e-01	5.5820e-01	5.3280e-01
k-NN	2.4597e-02	2.7391e-02	1.6710e-02
LMAFit	5.1509e-01	9.5634e-01	5.4320e-01
LRTC	2.2056e-01	2.0070e-01	2.2213e-01
STTC	1.7374e-01	3.2477e-01	2.5423e-01

Table 5.5: Normalized Root Mean Squared Error values (Abilene)

Table 5.5 and Chart 5.3 tell how the differences in performance among the models are diminished for the Abilene dataset, when compared to Geant and Mawi, although, some of the tendencies observed in 5.1 still persist:

- kNN, AAE and CNN are the best performers, with the latter demonstrating a higher stability as the NR grows, with relation to the Geant case.
- CCAE places only slightly behind the previous three algorithms.
- out of the models using [1,10] normalization, CAE struggles the most in the

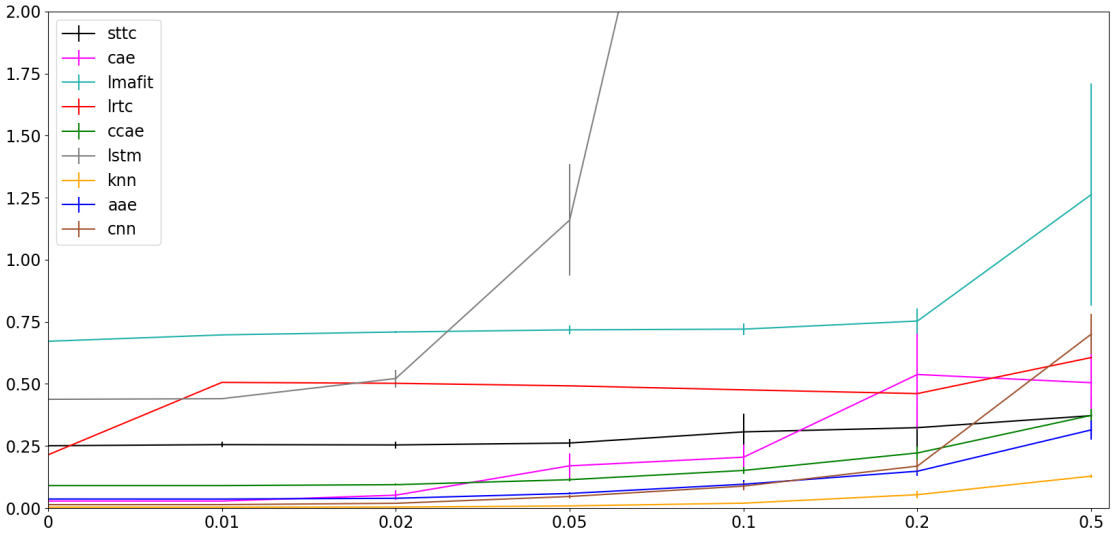


Figure 5.3: NRMSE for increasing noise ratios (Abilene)

Abilene case, having not only higher NRMSE values than the others, but also higher performance variability for differing sets of randomly perturbed coordinates of the matrices as noise is introduced.

- ConvLSTM is yet again the worst average performer, but sees a large improvement for low NR when compared to the Geant case.

- STTC appears to be more compatible with Abilene than it is with Geant and Mawi, most likely because the algorithm works closer to optimally with the "smoother" data of the dataset; the result is that the solution is generally superior to LMAFit and LRTC, and for high NR ($\geq 20\%$) is better than CAE and comparable with CCAE, AAE.

Table 5.6 and chart 5.4 corroborate the points made about the NRMSE performance of the models with Abilene, by reproducing the smaller performance gap among the algorithms with respect to the Geant case; all the trends already encountered in the NRMSE case are replicated with NMAE, with a minor discrepancy: the tendency of LMAFit is less stable than the NRMSE case, showing that while the magnitude of errors and/or the frequency of errors woth high magnitude do not increase substantially for increasing NR, as shown by the NRMSE measurements, the average error still increases noticeably, especially from the zero noise case onward, in a similar fashion to LRTC; another peculiarity of LMAFit is how the average error unintuitively seems to decrease when going from NR = 20% to 50%.

	(3,4)	(2,8)	(7,9)
AAE	0.1212	0.1551	0.1287
CAE	0.1313	0.1295	0.1126
CCAE	0.1635	0.2504	0.2025
CNN	0.0977	0.0966	0.0768
ConvLSTM	0.3417	0.6624	0.4227
k-NN	0.0591	0.0417	0.0384
LMAFit	0.6359	0.6941	0.7166
LRTC	0.4497	0.4885	0.4631
STTC	0.3289	0.5101	0.5206

Table 5.6: Normalized Mean Absolute Error values (Abilene)

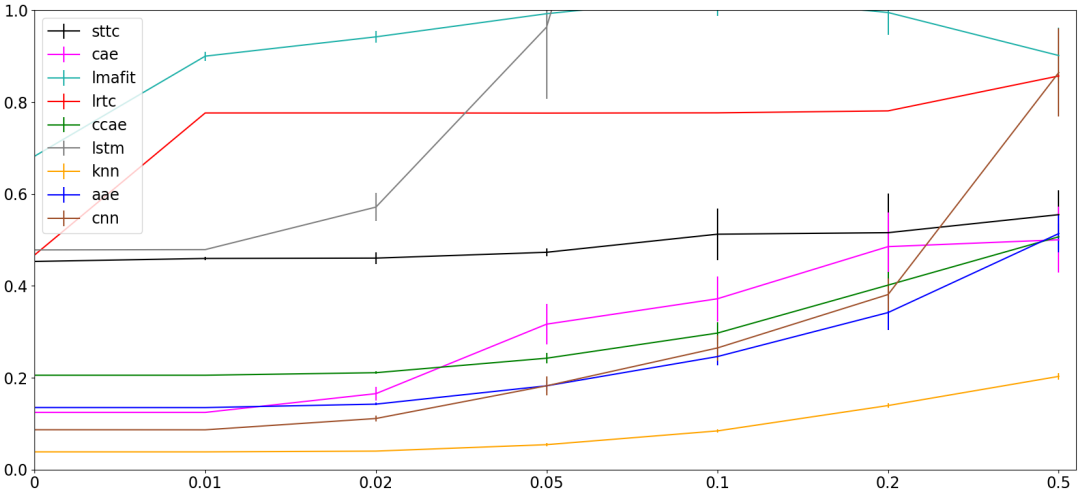


Figure 5.4: NMAE for increasing noise ratios (Abilene)

5.5.3 Results - Mawi Dataset

Table 5.7 and chart 5.5 confirm the superiority of kNN/AI models using the [1,10] normalization schema in all cases, with a glaring disparity; more specifically:

- kNN, CAE, AAE are consistent with the Geant and Abilene cases.
- CNN reports better performance with Mawi, especially for high NR, for which it does not decline as harshly with relation to the instances of the other two datasets.
- CCAE, despite its trademark noise tolerance, distances itself the most from the the other AI models in this particular case, being closer in performance to

	(1,8)	(11,2)	(19,14)
AAE	1.0915e-01	2.6857e-01	4.6037e-02
CAE	1.2137e-01	2.0914e-01	1.0975e-02
CCAE	7.8098e-01	8.4875e-01	4.1693e-01
CNN	3.6130e-02	1.2510e-01	2.3376e-03
ConvLSTM	8.7972e-01	9.2567e-01	1.7987e-01
k-NN	9.7873e-02	1.7987e-01	4.8726e-03
LMAFit	1.4659	1.0001	1.1202
LRTC	4.1322e-01	6.0398e-01	4.8981e-01
STTC	9.4162e-01	1.0069	7.9512e-01

Table 5.7: Normalized Root Mean Squared Error values (Mawi)

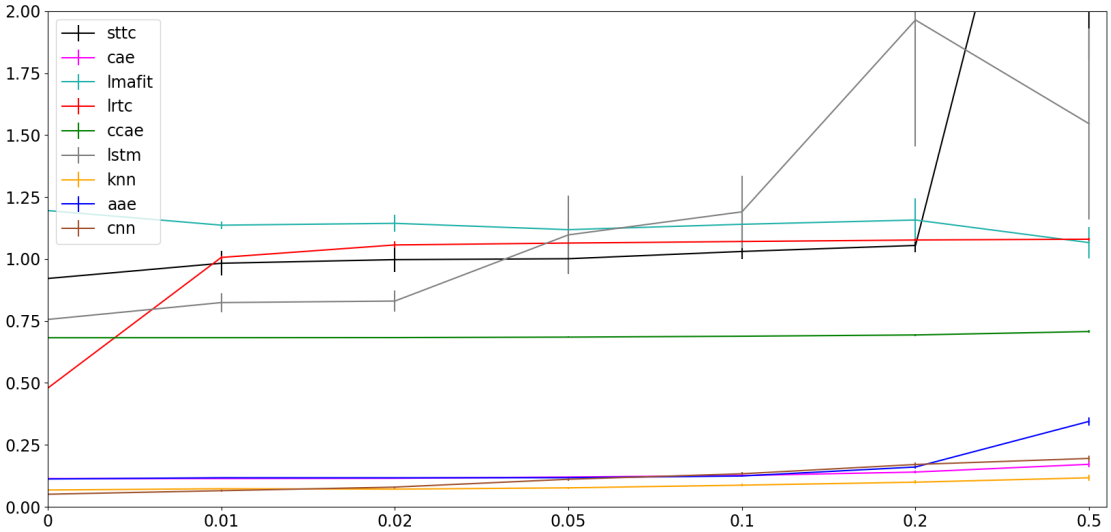


Figure 5.5: NRMSE for increasing noise ratios (Mawi)

the low rank optimization algorithms.

- ConvLSTM displays performance on average close to the LMAFit/STTC/LRTC group, but lacks the noise tolerance of the latter.
- STTC, LMAFit and LRTC produce very close results to one another, particularly for $1\% \leq \text{NR} \leq 20\%$; instead, in absence of noise, LRTC once again outperforms the other two, and for $\text{NR} > 20\%$ the estimations of STTC worsen outstandingly more.

Table 5.8 and chart 5.6 validate the same observations made for NRMSE; the sole dissimilarity, although not too prominent, is found with the low rank

	(1,8)	(11,2)	(19,14)
AAE	0.1358	0.2506	0.0622
CAE	0.1146	0.2622	0.0418
CCAE	0.4883	0.6581	0.4958
CNN	0.0909	0.2002	0.0329
ConvLSTM	1.1882	1.3908	0.3878
k-NN	0.0702	0.1931	0.0428
LMAFit	1.0325	1.000	1.1243
LRTC	0.6259	0.6340	0.5993
STTC	1.0402	1.3833	0.8584

Table 5.8: Normalized Mean Absolute Error values (Mawi)

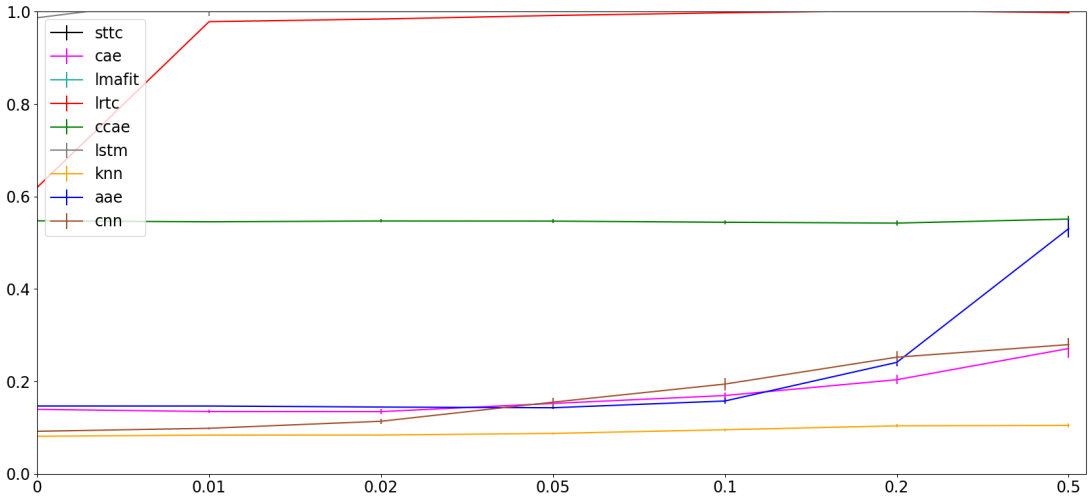


Figure 5.6: NMAE for increasing noise ratios (Mawi)

optimization algorithms relative performance: according to our measurements, LRTC universally yields better NMAE than STTC and LMAFit, meaning the former is more pronouncedly prone to predict with large errors than the latter two, despite the average error being lower.

5.5.4 Training and Prediction Times

	time (seconds)	epochs
AAE	4857.32	50
CAE	1086.93	100
CCAE	9513.83	50
CNN	336.75	50
ConvLSTM	1522.94	50
k-NN	0.0030434132	/

Table 5.9: Training Times (Geant)

	time (seconds)	epochs
AAE	6669.86	50
CAE	1777.52	100
CCAE	8545.44	50
CNN	678,43	50
ConvLSTM	2259.75	50
k-NN	0.0034039021	/

Table 5.10: Training Times (Abilene)

	time (seconds)	epochs
AAE	1673.19	50
CAE	864.77	100
CCAE	12150.23	50
CNN	297.80	50
ConvLSTM	2272.37	50
k-NN	0.0045638084	/

Table 5.11: Training Times (Mawi)

The values in 5.9, 5.10 and 5.11 represent total training times and epochs for the model, and come as no suprise:

- AAE and CAE, despite the comparable performances, have way different training times, with the first needing times-per-epoch from $\sim$3.9 to $\sim$9 higher than the second, given the more complex learning process.

- CCAE, possessing the highest count of trainable parameters, is the slowest model to train.
- vice-versa, CNN is the fastest.
- kNN, while included, does not undergo training in the same sense as neural networks, but rather the times reported refer to the time taken to store the training data in memory; hence, the way lower values.
- ConvLSTM places itself in the middle of the pack.

	time (seconds)
AAE	1.3671e-04
CAE	2.7903e-04
CCAE	2.8152e-04
CNN	1.8614e-04
ConvLSTM	8.9529e-04
k-NN	1.2067e-04
LMAFit	3.3232e-04
LRTC	8.8121e-03
STTC	2.4552e-01

Table 5.12: Average Prediction Times (Geant)

	time (seconds)
AAE	7.5961e-05
CAE	1.1945e-04
CCAE	1.1139e-03
CNN	1.0714e-04
ConvLSTM	2.5503e-04
k-NN	2.7166e-04
LMAFit	4.3060e-04
LRTC	3.4409e-03
STTC	3.1745e-01

Table 5.13: Average Prediction Times (Abilene)

The statistics in tables 5.12, 5.13 and 5.14 can be summarized as follows:

- AAE, CAE, CNN, kNN can be placed in the same bracket of average prediction times.

	time (seconds)
AAE	4.4790e-04
CAE	4,5784e-04
CCAE	7.4157e-03
CNN	1.5564e-04
ConvLSTM	1.8944e-03
k-NN	2.7014e-04
LMAFit	6.2203e-03
LRTC	8.0102e-03
STTC	2.0548e-01

Table 5.14: Average Prediction Times (Mawi)

- ConvLSTM and CCAE are the slowest AI solutions.
- LMAFit is, on average, considerably faster than the other low rank optimization based algorithms.
- LRTC is slower than LMAFit, but much quicker than STTC, which is the slowest overall by circa two orders of magnitude at best.

5.6 XAI Results

Following the study of performance, the focus now shifts to the behavioural interpretation of the neural networks used in this scope. The analysis revolves around three of the models: CNN, CAE and AAE.

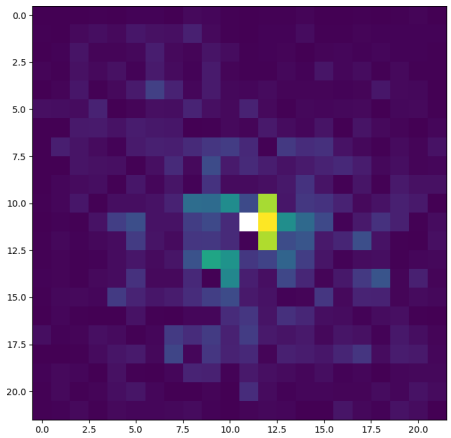
5.6.1 Saliency Maps

Saliency maps were the primary tool for explanation in this work, being an intuitive and efficient method for visualizing the rationale behind the estimations of the convolutional models. All the presented maps refer to the same color palette, with brighter (e.g. yellow) colors indicating higher saliency values, and darker colors(e.g. dark blue/green) conversely indicate lower saliency. For the sake of a more immediate visual comparability between different models, the maps generated by AAE and CAE were shuffled and centered around the target coordinate, in order to conform to the CNN configuration. Two types of maps are presented:

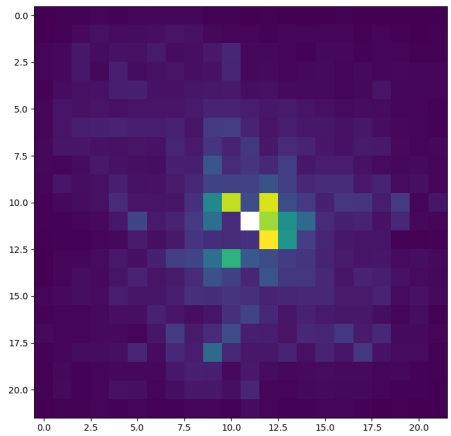
- single prediction saliency maps, computed for single, randomly sampled matrices.

- average prediction saliency maps, computed by averaging the saliency values over the test sets.

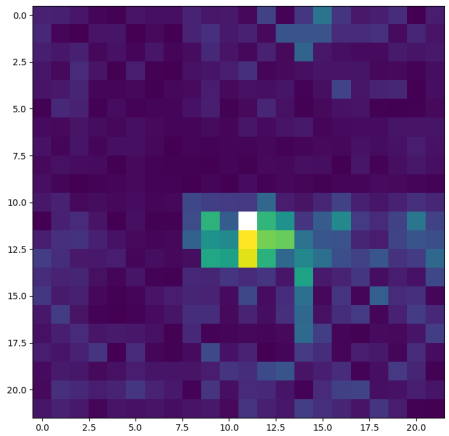
The choice of computing an average saliency over a large amount of data is due to the intent of providing a global understanding of how the model interacts with the dataset, and not only explain the reasons for a single prediction. Starting from the single prediction saliency maps, it appears sometimes possible to find a common pattern among the three models, in which the most influential data flows for the prediction fall within the same region of the matrix (Figures 5.14 **(a)**, **(c)**, **(e)**) or even coincide (Figures 5.10 **(a)**, **(c)**, **(e)**), while there are cases in which the attention of the three models focuses on completely different flows (Figures 5.15 **(a)**, **(c)**, **(e)**). This result provides some interesting insight, meaning that despite their comparable performances, the three models do not always weigh the information they are fed with in the same manner, and possess specific nuances in their operation that depend not only on the patterns of the data they were trained with, but also on the structural details of the architectures (e.g the size and number of convolutional filters). On the other hand, average saliency maps display a more consistent pattern: despite the models focusing on variably large sections of the matrices, all three of them are, on average, more heavily influenced by OD flows in close proximity to the inference target, and gradually exhibit less attention as the distance from the position of said target grows. This behaviour is symptomatic of a potential issue with these architectures, when dealing with this type of data: flows placed in neighboring positions inside a traffic matrix might in reality be weakly related to one another, since the reason they occupy adjacent cells might simply be due to how the algorithms of collection and compilation into the bi-dimensional structure process the data. For these reasons, it might be worth exploring whether finding a criterion to distribute traffic flows inside the matrix in a such a way that their positional distance reflects their actual mutual correlation can be beneficial to the performances of these models.



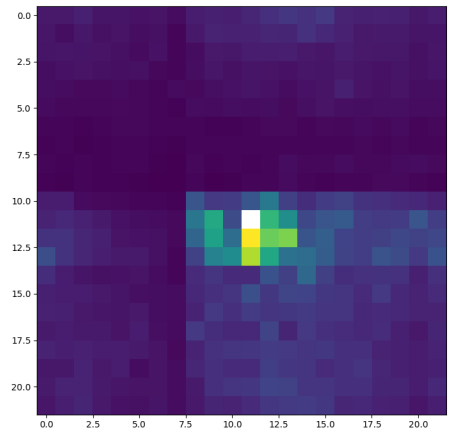
(a) CNN (sample)



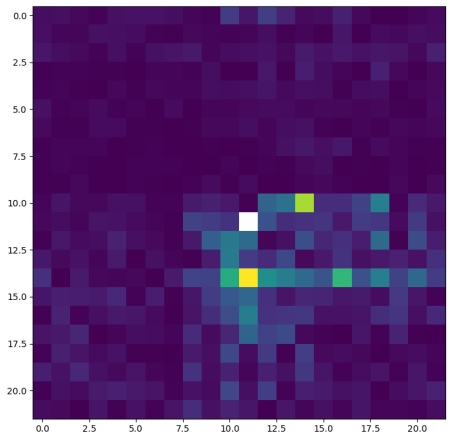
(b) CNN (average)



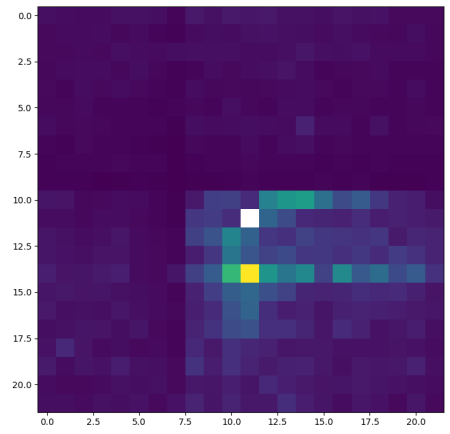
(c) CAE (sample)



(d) CAE (average)

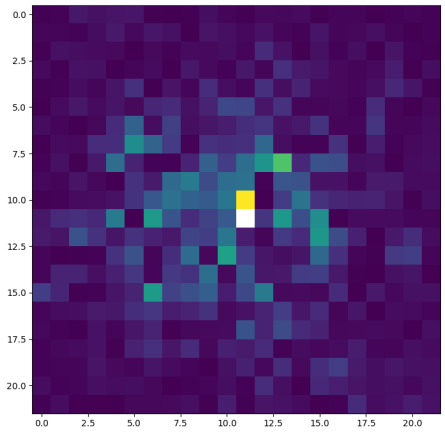


(e) AAE (sample)

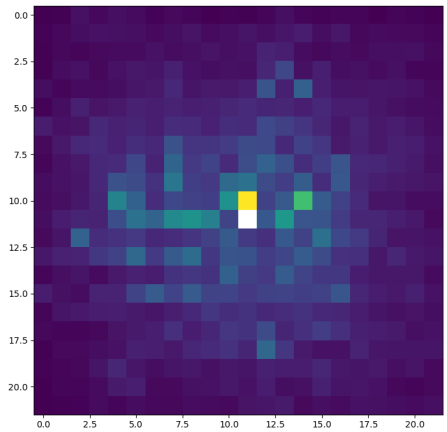


(f) AAE (average)

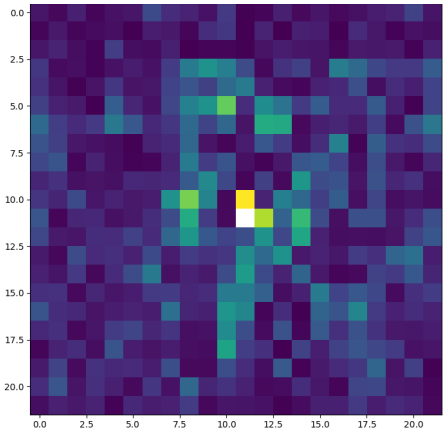
Figure 5.7: Saliency Maps (Geant, coordinate 1,3)



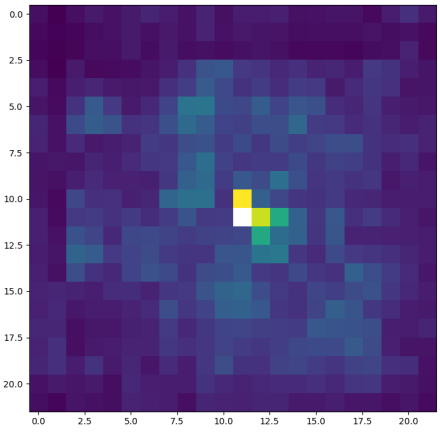
(a) CNN (sample)



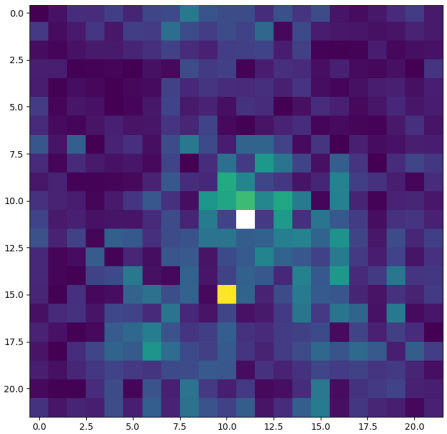
(b) CNN (average)



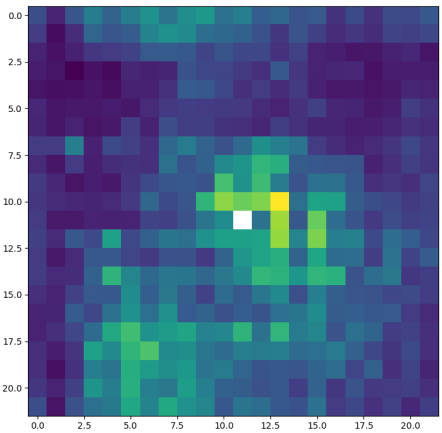
(c) CAE (sample)



(d) CAE (average)

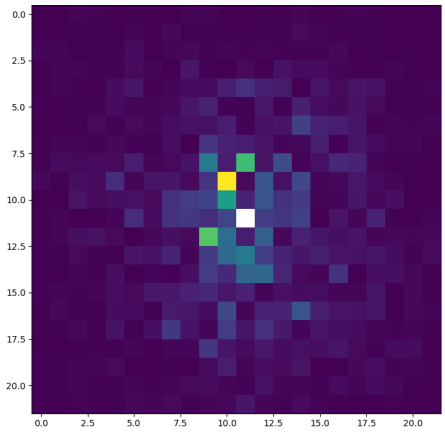


(e) AAE (sample)

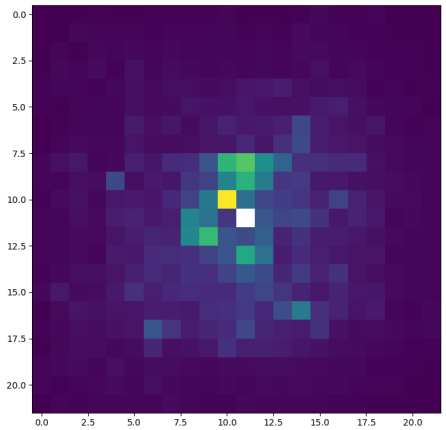


(f) AAE (average)

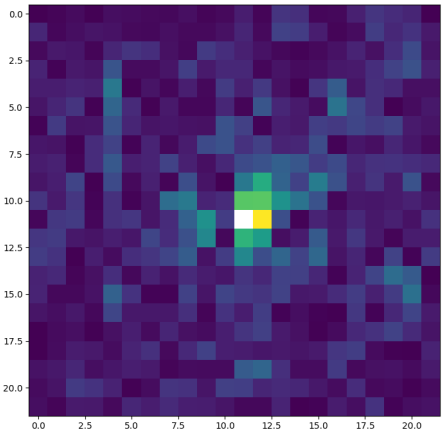
Figure 5.8: Saliency Maps (Geant, coordinate 8,9)



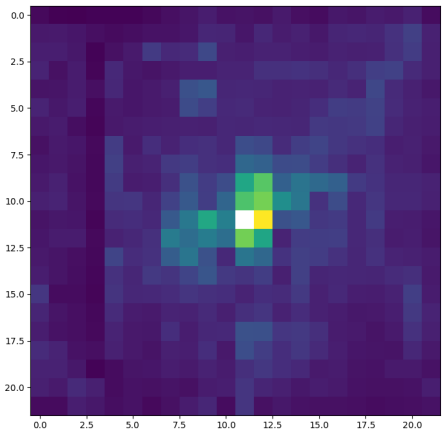
(a) CNN (sample)



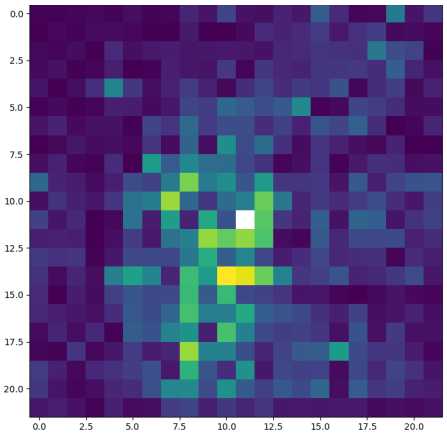
(b) CNN (average)



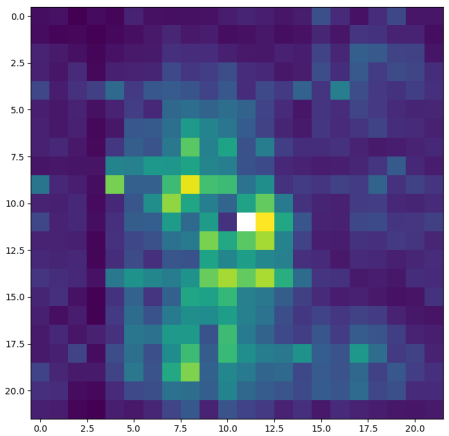
(c) CAE (sample)



(d) CAE (average)

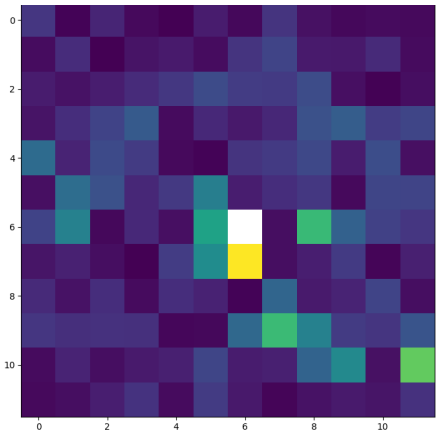


(e) AAE (sample)

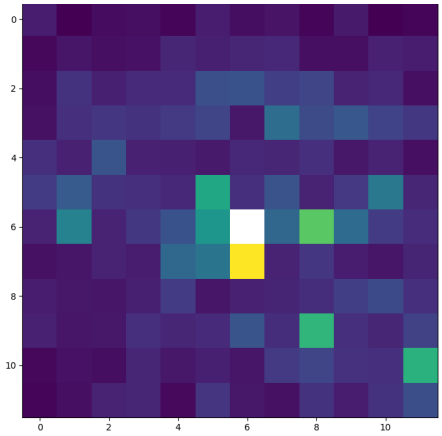


(f) AAE (average)

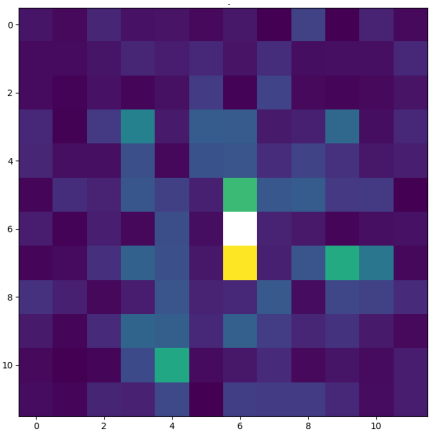
Figure 5.9: Saliency Maps (Geant, coordinate 11,7)



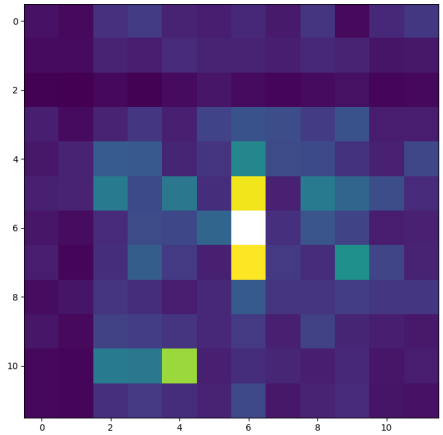
(a) CNN (sample)



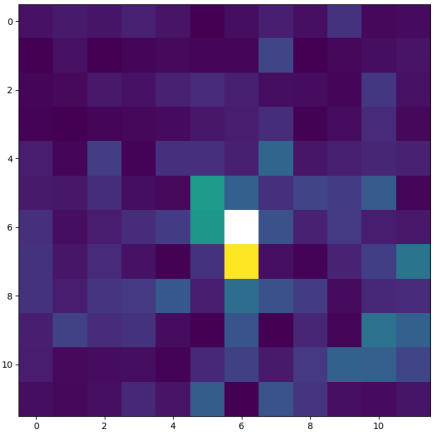
(b) CNN (average)



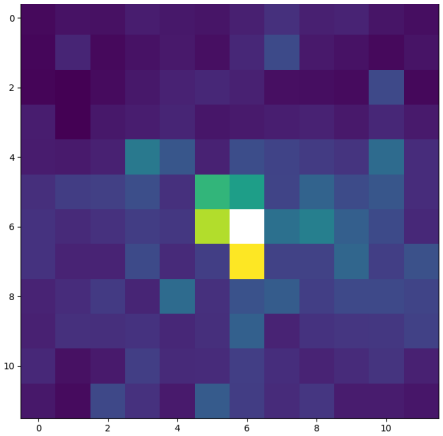
(c) CAE (sample)



(d) CAE (average)

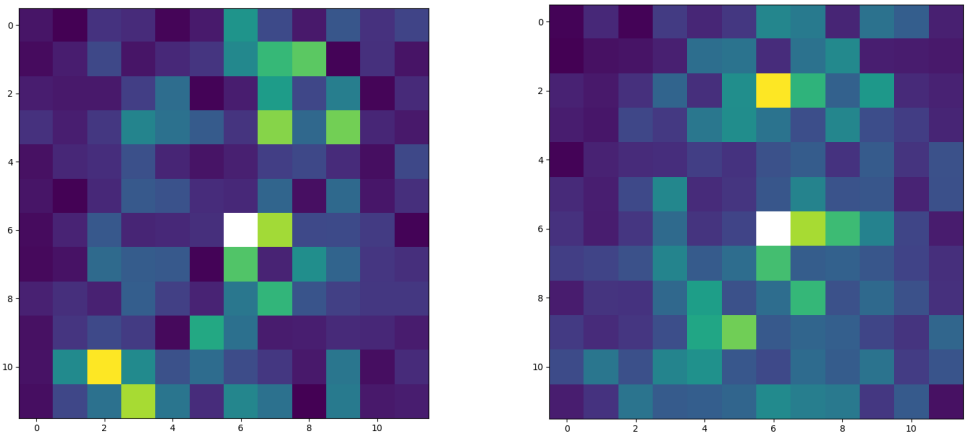


(e) AAE (sample)



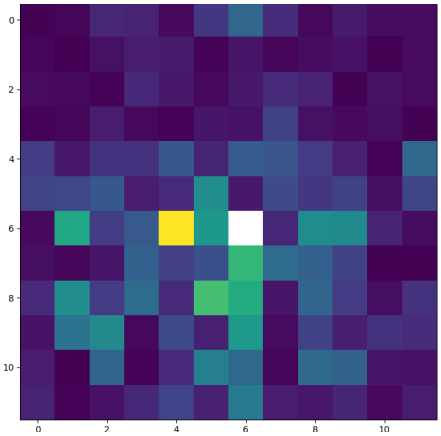
(f) AAE (average)

Figure 5.10: Saliency Maps (Abilene, coordinate 3,4)

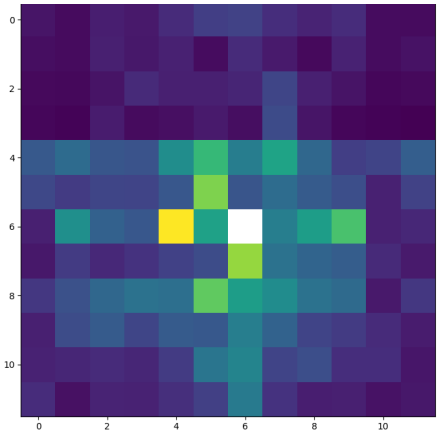


(a) CNN (sample)

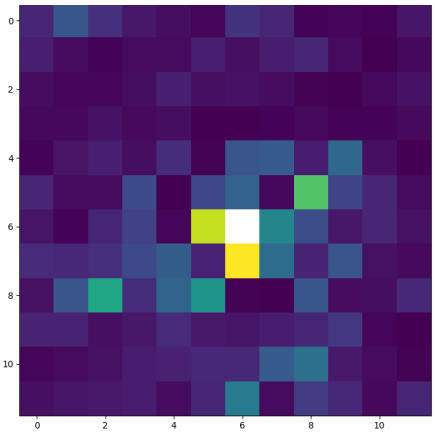
(b) CNN (average)



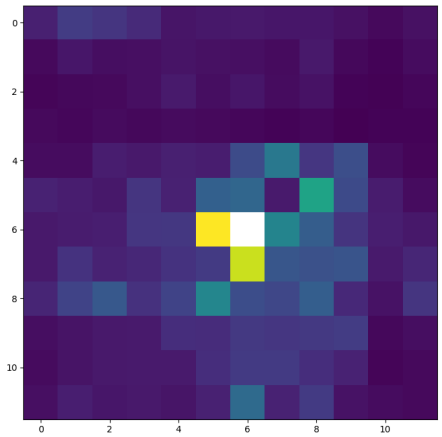
(c) CAE (sample)



(d) CAE (average)

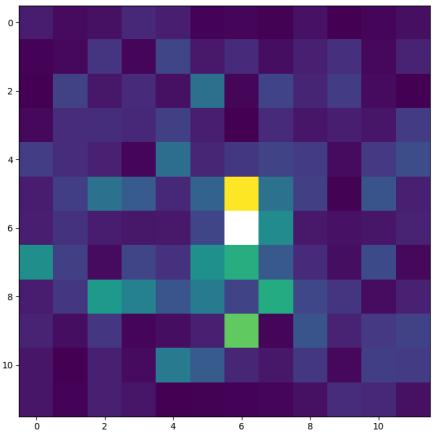


(e) AAE (sample)

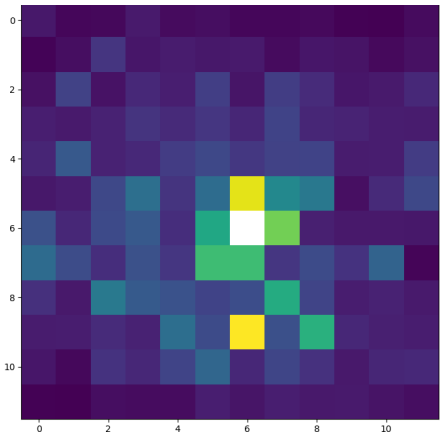


(f) AAE (average)

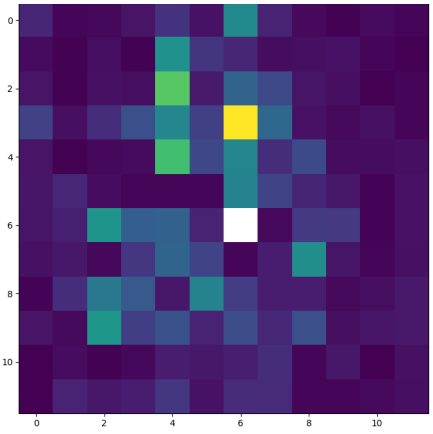
Figure 5.11: Saliency Maps (Abilene, coordinate 2,8)



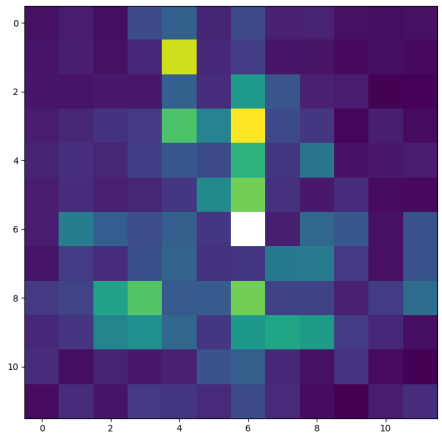
(a) CNN (sample)



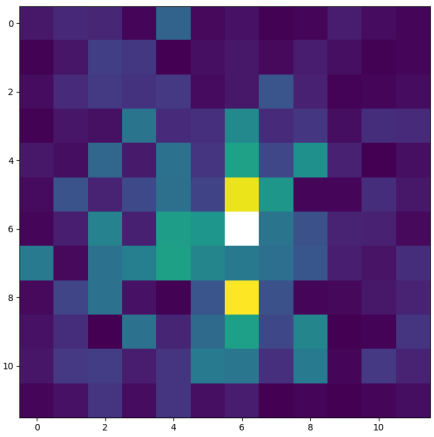
(b) CNN (average)



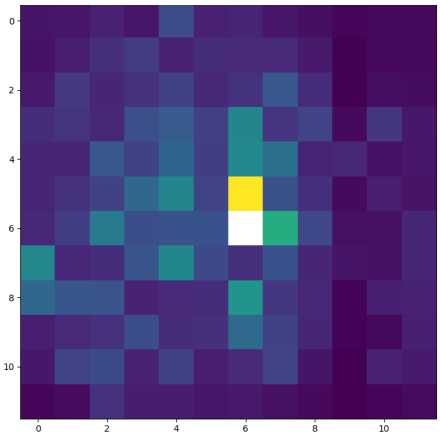
(c) CAE (sample)



(d) CAE (average)

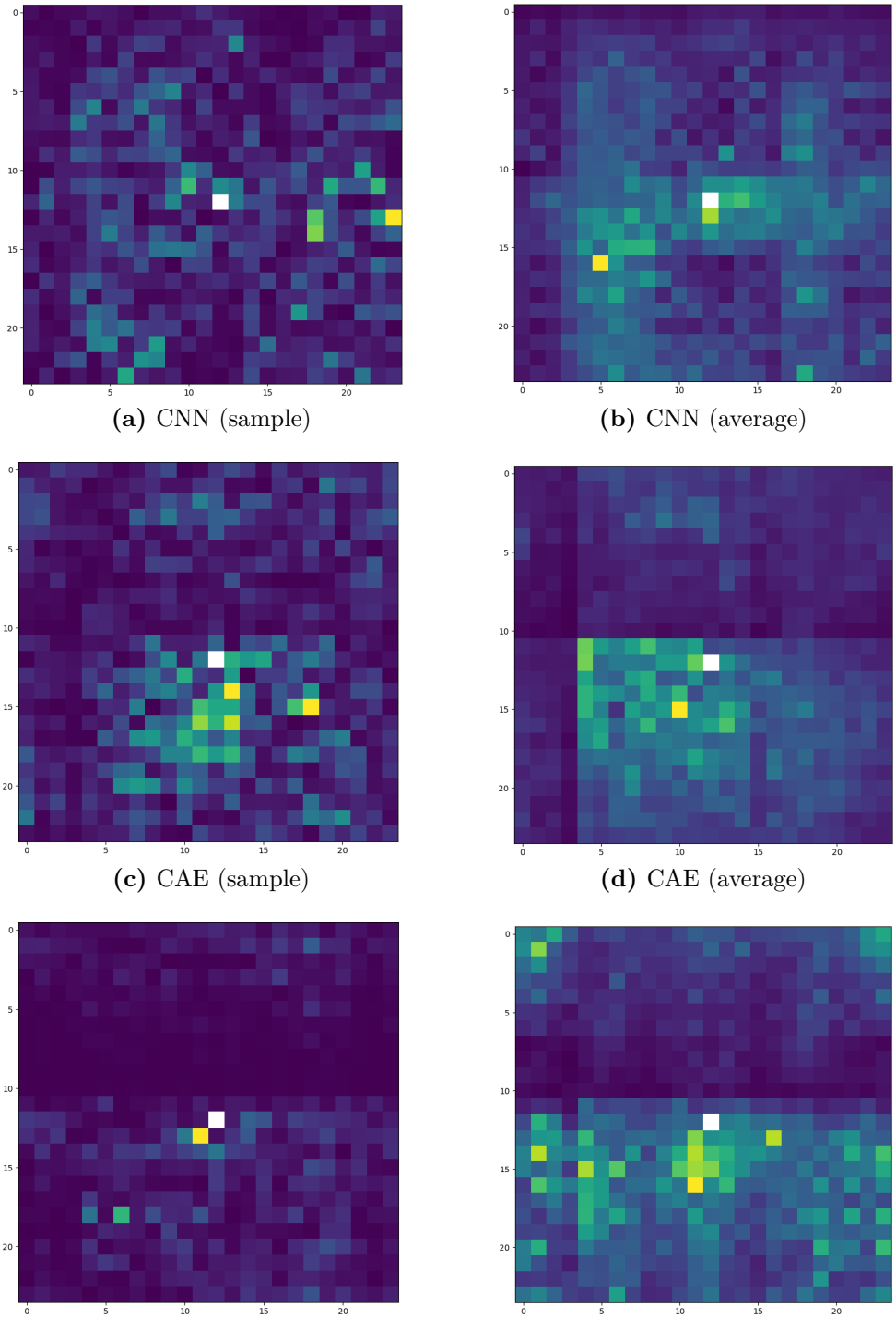


(e) AAE (sample)



(f) AAE (average)

Figure 5.12: Saliency Maps (Abilene, coordinate 7,9)



(a) CNN (sample)

(b) CNN (average)

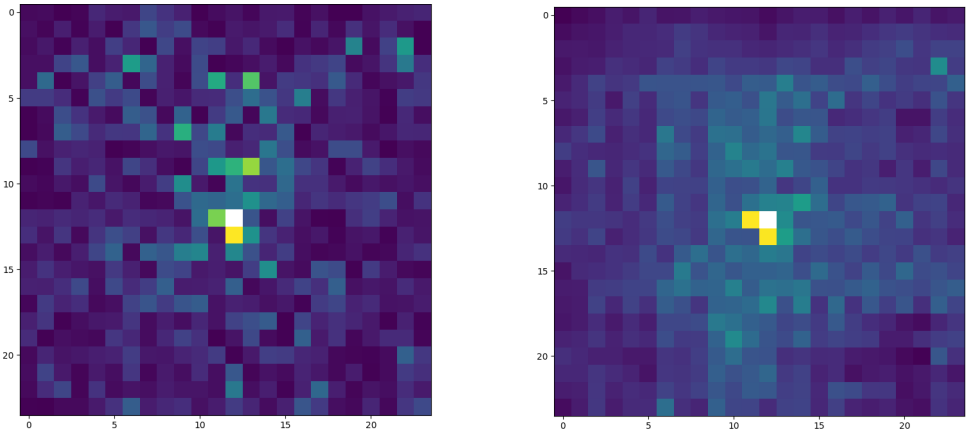
(c) CAE (sample)

(d) CAE (average)

(e) AAE (sample)

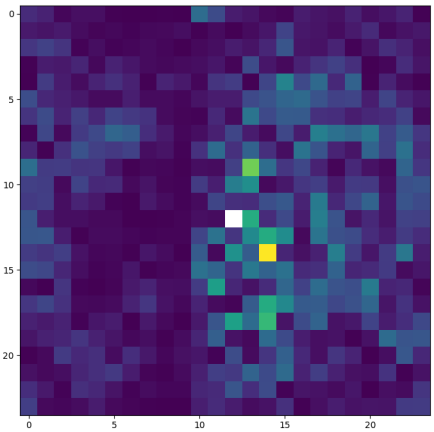
(f) AAE (average)

Figure 5.13: Saliency Maps (Mawi, coordinate 1,8)

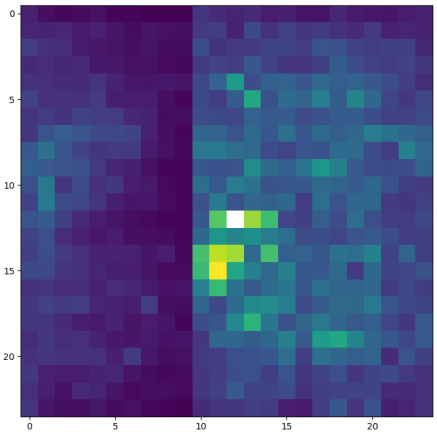


(a) CNN (sample)

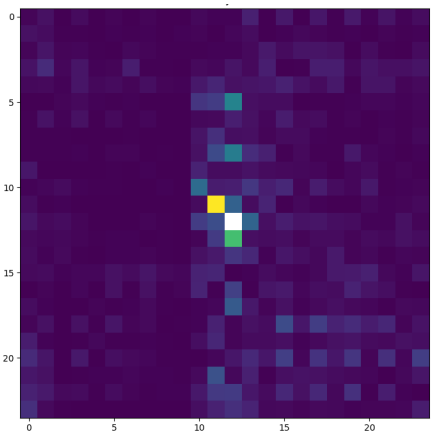
(b) CNN (average)



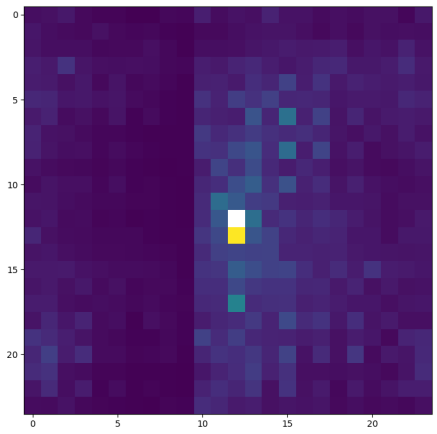
(c) CAE (sample)



(d) CAE (average)

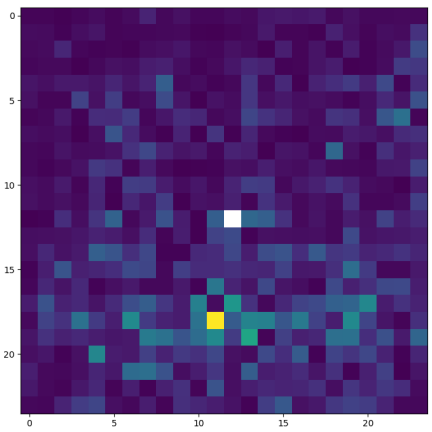


(e) AAE (sample)

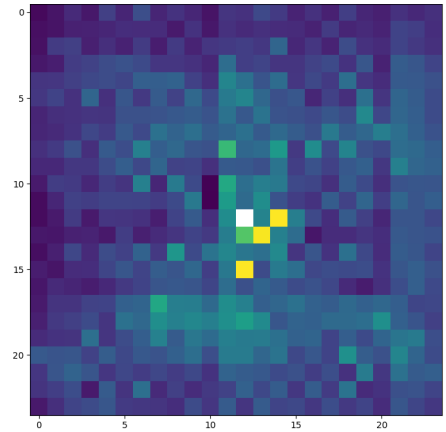


(f) AAE (average)

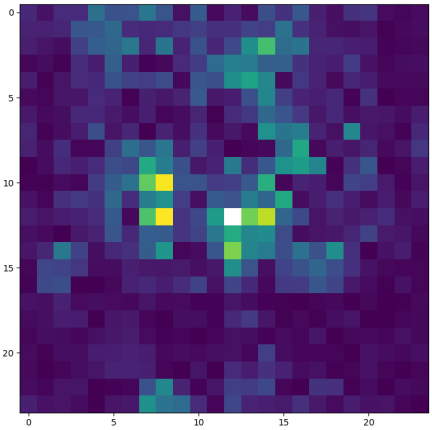
Figure 5.14: Saliency Maps (Mawi, coordinate 11,2)



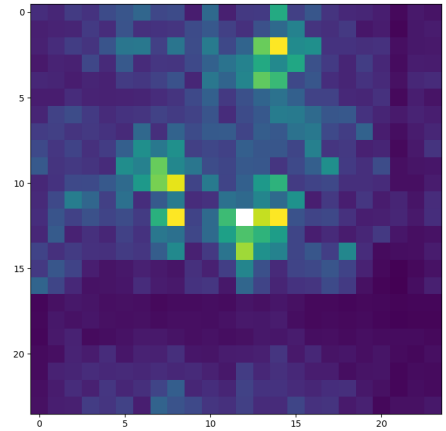
(a) CNN (sample)



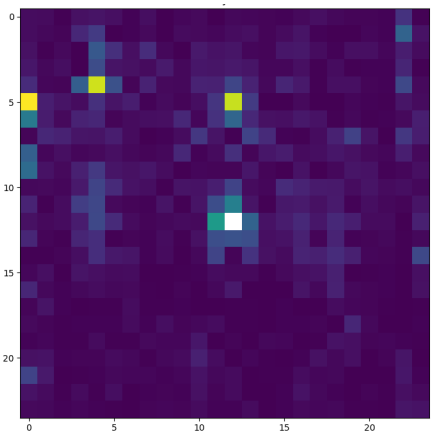
(b) CNN (average)



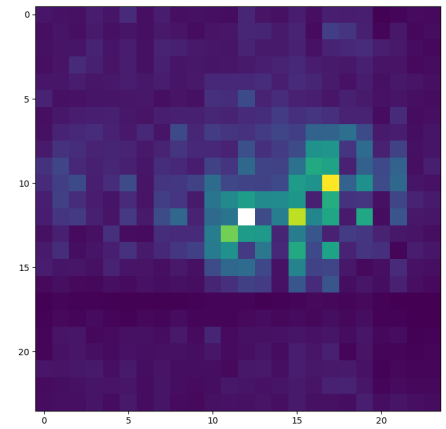
(c) CAE (sample)



(d) CAE (average)



(e) AAE (sample)



(f) AAE (average)

Figure 5.15: Saliency Maps (Mawi, coordinate 19,14)

Chapter 6

Conclusion

The work reported in this thesis was led with the overarching goal of verifying the applicability of the techniques for AI explanation found in literature, to the cases of neural networks trained to solve the problem of Traffic Matrix Completion. The study was carried in three steps:

- first, we focused on the problem of completion itself, aiming to grasp its nature and the technical challenges it derives from, surveying the landscape of existing algorithms and finally mounting a comparison among a selection of these algorithms by a number of different metrics, observing how in our case the AI based approaches generally outperform their competitors.
- secondly, after scrutinizing a set of XAI methods and the intuitive value of the information they provided about the algorithms we tested, we chose to present the application of saliency computation to three different AI architectures under different conditions.
- Finally, we discussed the patterns observed in the behaviour of the networks through the saliency maps, the extent of the insight they provided about both single predictions and on average, and, based on this information, speculated about how mindfully arranging the information inside of traffic matrices, given prerequisite knowledge about the network flows and their correlations, could lead to an improvement of performances.

Bibliography

[1] Matthias Grossglauser and Jennifer Rexford. «Passive Traffic Measurement for IP Operations». In: (Apr. 2002) (cit. on p. 1).

[2] Huibin Zhou, Dafang Zhang, and Kun Xie. «Accurate traffic matrix completion based on multi-Gaussian models». In: *Computer Communications* 102 (2017), pp. 165–176. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2016.11.011. URL: https://www.sciencedirect.com/science/article/pii/S0140366416306223 (cit. on pp. 1, 5).

[3] Daniel Zhang et al. *The AI Index 2022 Annual Report*. Mar. 2022 (cit. on p. 2).

[4] AI HLEG. *Ethics Guidelines for Trustworthy AI*. 2019 (cit. on p. 2).

[5] Helena Webb. «A governance framework for algorithmic accountability and transparency». In: 2019 (cit. on p. 2).

[6] Yipeng Liu, Zhen Long, and Ce Zhu. «Image Completion Using Low Tensor Tree Rank and Total Variation Minimization». In: *IEEE Transactions on Multimedia* 21.2 (2019), pp. 338–350. DOI: 10.1109/TMM.2018.2859026 (cit. on pp. 4, 21).

[7] Emmanuel J. Candes and Benjamin Recht. *Exact Matrix Completion via Convex Optimization*. 2008. DOI: 10.48550/ARXIV.0805.4471. URL: https://arxiv.org/abs/0805.4471 (cit. on p. 5).

[8] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. «Nonlinear total variation based noise removal algorithms». In: *Physica D: Nonlinear Phenomena* 60.1 (1992), pp. 259–268. ISSN: 0167-2789. DOI: https://doi.org/10.1016/0167-2789(92)90242-F. URL: https://www.sciencedirect.com/science/article/pii/016727899290242F (cit. on p. 5).

[9] Shangqi Gao and Xiahai Zhuang. «Robust approximations of low-rank minimization for tensor completion». In: *Neurocomputing* 379 (2020), pp. 319–333. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.10.086. URL: https://www.sciencedirect.com/science/article/pii/S0925231219315280 (cit. on p. 5).

[10] Shangqi Gao and Qibin Fan. «A Mixture of Nuclear Norm and Matrix Factorization for Tensor Completion». In: *Journal of Scientific Computing* 75.1 (Apr. 2018), pp. 43–64. ISSN: 1573-7691. DOI: 10.1007/s10915-017-0521-9. URL: https://doi.org/10.1007/s10915-017-0521-9 (cit. on pp. 5, 21).

[11] Zaiwen Wen, Wotao Yin, and Yin Zhang. «Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm». In: *Mathematical Programming Computation* 4 (Dec. 2012). DOI: 10.1007/s12532-012-0044-1 (cit. on pp. 5, 21).

[12] Huibin Zhou, Dafang Zhang, Kun Xie, and Yuxiang Chen. «Spatio-temporal tensor completion for imputing missing internet traffic data». In: *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. 2015, pp. 1–7. DOI: 10.1109/PCCC.2015.7410315 (cit. on p. 5).

[13] J. Douglas Carroll and Jih-Jie Chang. «Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition». In: *Psychometrika* 35.3 (Sept. 1970), pp. 283–319. ISSN: 1860-0980. DOI: 10.1007/BF02310791. URL: https://doi.org/10.1007/BF02310791 (cit. on p. 5).

[14] Richard A. Harshman. «Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-model factor analysis». In: 1970 (cit. on p. 5).

[15] Fu Xiao, Lei Chen, Hai Zhu, Richang Hong, and Ruchuan Wang. «Anomaly-Tolerant Network Traffic Estimation via Noise-Immune Temporal Matrix Completion Model». In: *IEEE Journal on Selected Areas in Communications* 37.6 (2019), pp. 1192–1204. DOI: 10.1109/JSAC.2019.2904347 (cit. on p. 5).

[16] Matthew Roughan, Mikkel Thorup, and Yin Zhang. «Traffic Engineering with Estimated Traffic Matrices». In: *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*. IMC '03. Miami Beach, FL, USA: Association for Computing Machinery, 2003, pp. 248–258. ISBN: 1581137737. DOI: 10.1145/948205.948237. URL: https://doi.org/10.1145/948205.948237 (cit. on p. 6).

[17] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. «Experience in Measuring Backbone Traffic Variability: Models, Metrics, Measurements and Meaning». In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*. IMW '02. Marseille, France: Association for Computing Machinery, 2002, pp. 91–92. ISBN: 158113603X. DOI: 10.1145/637201.637213. URL: https://doi.org/10.1145/637201.637213 (cit. on p. 6).

[18] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greenberg. «Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads». In: *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '03. San Diego, CA, USA: Association for Computing Machinery, 2003, pp. 206–217. ISBN: 1581136641. DOI: 10.1145/781027.781053. URL: https://doi.org/10.1145/781027.781053 (cit. on pp. 6, 17).

[19] Rashida Ali Memon, Sameer Qazi, and Bilal Muhammad Khan. «Design and Implementation of a Robust Convolutional Neural Network-Based Traffic Matrix Estimator for Cloud Networks». In: *Wireless Communications and Mobile Computing* 2021 (June 2021), p. 1039613. ISSN: 1530-8669. DOI: 10.1155/2021/1039613. URL: https://doi.org/10.1155/2021/1039613 (cit. on p. 6).

[20] Xin Wang, Yuanyi Chen, Wei Ruan, Qiang Gao, Guode Ying, and Li Dong. «Intelligent Detection and Recovery of Missing Electric Load Data Based on Cascaded Convolutional Autoencoders». In: *Scientific Programming* 2020 (Dec. 2020), p. 8828745. ISSN: 1058-9244. DOI: 10.1155/2020/8828745. URL: https://doi.org/10.1155/2020/8828745 (cit. on p. 6).

[21] Van An Le, Phi Le Nguyen, and Yusheng Ji. «Deep Convolutional LSTM Network-based Traffic Matrix Prediction with Partial Information». In: *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2019, pp. 261–269 (cit. on p. 6).

[22] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting*. 2015. DOI: 10.48550/ARXIV.1506.04214. URL: https://arxiv.org/abs/1506.04214 (cit. on p. 6).

[23] Alejandro Barredo Arrieta et al. «Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI». In: *Information Fusion* 58 (2020), pp. 82–115. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2019.12.012. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103 (cit. on pp. 6, 7).

[24] Kasun Amarasinghe, Kevin Kenney, and Milos Manic. «Toward Explainable Deep Neural Network Based Anomaly Detection». In: *2018 11th International Conference on Human System Interaction (HSI)*. 2018, pp. 311–317. DOI: 10.1109/HSI.2018.8430788 (cit. on p. 7).

[25] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. «On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation». In: *PLOS ONE* 10.7 (July 2015), pp. 1–46. DOI: 10.1371/journal.pone.

0130140. URL: https://doi.org/10.1371/journal.pone.0130140 (cit. on p. 7).

[26] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. «Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization». In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74 (cit. on p. 7).

[27] Han-Yun Chen and Ching-Hung Lee. «Vibration Signals Analysis by Explainable Artificial Intelligence (XAI) Approach: Application on Bearing Faults Diagnosis». In: *IEEE Access* 8 (2020), pp. 134246–134256. DOI: 10.1109/ACCESS.2020.3006491 (cit. on p. 8).

[28] Igor Cherepanov, Alex Ulmer, Jonathan Geraldi Joewono, and Jörn Kohlhammer. *Visualization Of Class Activation Maps To Explain AI Classification Of Network Packet Captures*. 2022. DOI: 10.48550/ARXIV.2209.02045. URL: https://arxiv.org/abs/2209.02045 (cit. on p. 8).

[29] Ying Zheng, Ziyu Liu, Xinyu You, Yuedong Xu, and Junchen Jiang. «Demystifying Deep Learning in Networking». In: *Proceedings of the 2nd Asia-Pacific Workshop on Networking*. APNet '18. Beijing, China: Association for Computing Machinery, 2018, pp. 1–7. ISBN: 9781450363952. DOI: 10.1145/3232565.3232569. URL: https://doi.org/10.1145/3232565.3232569 (cit. on p. 8).

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2013. DOI: 10.48550/ARXIV.1312.6034. URL: https://arxiv.org/abs/1312.6034 (cit. on pp. 8, 14).

[31] Matthew D Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013. DOI: 10.48550/ARXIV.1311.2901. URL: https://arxiv.org/abs/1311.2901 (cit. on p. 8).

[32] Tomoki Uchiyama, Naoya Sogi, Koichiro Niinuma, and Kazuhiro Fukui. *Visually explaining 3D-CNN predictions for video classification with an adaptive occlusion sensitivity analysis*. 2022. DOI: 10.48550/ARXIV.2207.12859. URL: https://arxiv.org/abs/2207.12859 (cit. on p. 8).

[33] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions*. 2017. DOI: 10.48550/ARXIV.1705.07874. URL: https://arxiv.org/abs/1705.07874 (cit. on p. 9).

[34] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. «Learning Important Features Through Propagating Activation Differences». In: (2017). DOI: 10.48550/ARXIV.1704.02685. URL: https://arxiv.org/abs/1704.02685 (cit. on p. 9).

[35] Alfredo Nascita, Antonio Montieri, Giuseppe Aceto, Domenico Ciuonzo, Valerio Persico, and Antonio Pescapé. «XAI Meets Mobile Traffic Classification: Understanding and Improving Multimodal Deep Learning Architectures». In: *IEEE Transactions on Network and Service Management* 18.4 (2021), pp. 4225–4246. DOI: 10.1109/TNSM.2021.3098157 (cit. on p. 9).

[36] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. DOI: 10.48550/ARXIV.1602.04938. URL: https://arxiv.org/abs/1602.04938 (cit. on p. 9).

[37] Andrea Morichetta, Pedro Casas, and Marco Mellia. «EXPLAIN-IT». In: *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*. ACM, Dec. 2019. DOI: 10.1145/3359992.3366639. URL: https://doi.org/10.11452F3359992.3366639 (cit. on p. 9).

[38] Cecilia Panigutti, Alan Perotti, and Dino Pedreschi. «Doctor XAI: An Ontology-Based Approach to Black-Box Sequential Data Classification Explanations». In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. FAT* '20. Barcelona, Spain: Association for Computing Machinery, 2020, pp. 629–639. ISBN: 9781450369367. DOI: 10.1145/3351095.3372855. URL: https://doi.org/10.1145/3351095.3372855 (cit. on p. 9).

[39] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. *Doctor AI: Predicting Clinical Events via Recurrent Neural Networks*. 2015. DOI: 10.48550/ARXIV.1511.05942. URL: https://arxiv.org/abs/1511.05942 (cit. on p. 9).

[40] Dulari Bhatt, Chirag Patel, Hardik Talsania, Jigar Patel, Rasmika Vaghela, Sharnil Pandya, Kirit Modi, and Hemant Ghayvat. «CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope». In: *Electronics* 10.20 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10202470. URL: https://www.mdpi.com/2079-9292/10/20/2470 (cit. on p. 11).

[41] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE] (cit. on p. 11).

[42] Ademola E. Ilesanmi and Taiwo O. Ilesanmi. «Methods for image denoising using convolutional neural network: a review». In: *Complex & Intelligent Systems* 7.5 (Sept. 2021), pp. 2179–2198. ISSN: 2198-6053. DOI: 10.1007/s40747-021-00428-4. URL: https://doi.org/10.1007/s40747-021-00428-4 (cit. on p. 12).

[43] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. *Adversarial Autoencoders*. 2016. arXiv: 1511.05644 [cs.LG] (cit. on p. 13).

[44] Dumitru Erhan, Y. Bengio, Aaron Courville, and Pascal Vincent. «Visualizing Higher-Layer Features of a Deep Network». In: *Technical Report, Univeristé de Montréal* (Jan. 2009) (cit. on p. 15).

[45] Y. Vardi. «Network Tomography: Estimating Source-Destination Traffic Intensities from Link Data». In: *Journal of the American Statistical Association* 91.433 (1996), pp. 365–377. ISSN: 01621459. URL: http://www.jstor.org/stable/2291416 (visited on 11/22/2022) (cit. on pp. 16, 17).

[46] Xin Wang, Yuanyi Chen, Wei Ruan, Qiang Gao, Guode Ying, and Li Dong. «Intelligent Detection and Recovery of Missing Electric Load Data Based on Cascaded Convolutional Autoencoders». In: *Scientific Programming* 2020 (Dec. 2020), pp. 1–20. DOI: 10.1155/2020/8828745 (cit. on pp. 20, 21).