DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

IM-SAFE.EU

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Document Information

Name	Remark
Document Name	Data analytics and Artificial Intelligence
Version No.	V1.0
Due date Annex I	15/07/2020
Report date	30/04/2022 (last updated in 29/04/2022)
Number of pages	71
Lead Author	Ioana Giurgiu (IBM)
Other Authors	Mattia Rigotti (IBM), Patrycja Sanecka (Mostostal)
Dissemination level	Public

Document history

Ver.	Date	Remark	Author	Checked by
Rev. 0.1	07/12/2021	Creation of the document	I. Giurgiu	M. Rigotti
Rev. 0.2	15/01/2022	First draft for revision	All	I. Giurgiu
Rev. 0.3	15/04/2022	Document for reviewers	All	I. Giurgiu
Rev. 0.4	24/04/2022	Amendments from reviewers	All	I. Giurgiu
Rev. 0.5	29/04/2022	Document for approval	All	F. Cinquini and A. Bigaj-van Vliet
Rev. 1.0	30/04/2022	Document Uploaded	All	A. Bigaj-van Vliet

Document approval

Ver.	Date	Approval	Project's Role	Beneficiary
Rev. 1.0	30/04/2022	A. Bigaj-van Vliet	Coordinator	TNO

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Preface

This deliverable is part of the H2020 CSA IM-SAFE project and is the outcome of the fourth task of work package 4 (Digitalisation as enabling technology, task 4.4: Data analytics and artificial intelligence). It will help to set the basis of the proposal for the mandate to the European Committee for Standardization (CEN).

Task leader: IBM Research GmbH (IBM)
Contributors: MOSTOSTAL WARSZAWA S A (MOSTOSTAL)

In WP4, the relation with future standards in monitoring with the open IT standards is given, together with the data platforms available to manage different monitoring information and the data analytic technologies needed for processing those data.
This document provides a review of particular data analytics and artificial intelligence technologies, including typical workflows for data preparation and model building and deployment, as well as model adoption in the wild. Then an insight on how these workflows have been used in three insightful real scenarios is given.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Table of Content

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Table of Figures

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 958171.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

1 Problem statement

1.1 Introduction

While artificial intelligence is an incredibly powerful tool, in some industries it is still in a developmental stage. There is still exploration to be done regarding what applications are best suited for its use and refining the involved processes and workflows. However, it is already clear that its impact will be significant, in the way planning and maintenance will be done in the future. Every form of artificial intelligence, including machine learning, needs data to ingest. Today's machine learning tools leverage algorithms to parse and understand ingested data, allowing them to find trends, patterns, and other insights of value. These algorithms act as guidelines, essentially telling the system where to go, what to look for, and what to do with that information. This is relevant because those algorithms, or guidelines, need to be established and optimized to empower artificial intelligence and machine learning technologies. In civil engineering, they will be applied in various ways to complete planning, construction, and similar tasks.

Civil engineering projects call for a huge selection of tools, heavy machines, and expensive materials. Managing these assets is a challenging task, both during working hours and outside of them, as well. Smart asset tracking can help organize projects by allowing workers to see what is currently in use, and what is available. It can facilitate proper project planning and management strategies. It can also be used to discern performance, progress, and other factors, remotely by team leaders.

Machine learning technologies are also able to process massive swarms of data in a relatively short period. It also allows them to discover hidden trends, patterns, and insights, as well as make accurate predictions through the analysis of historic and current information. In civil engineering, this could potentially help discover and create new solutions to complex problems.

Risk mitigation involves analyzing the site of construction project for examples, the surrounding environment, external conditions like weather, hazards, worker safety, and much more. But because there is so much to consider, measure, and plan for, mistakes can be made and details can be missed rather easily. A machine learning could use smart technologies and contextual data to help with planning, risk mitigation and safety, by prioritizing immediate actions and assisting experts in assessing risks.

It is clear that civil engineering and construction are changing, thanks to generative and data-oriented design techniques, real-time asset management, and smarter risk mitigation and job site safety. New discoveries and avenues are explored every day, which shows the incredible promise of machine learning for engineers and development applications.

1.2 Objectives of the deliverable

This deliverable addresses several use case scenarios and how artificial intelligence, and machine learning technologies is used to provide relevant value and capabilities in the civil engineering and construction domains.

Drawing from these use case scenarios, the deliverable describes the typical machine learning flows and stages, and relevant techniques to be used at every stage. The specific aspects addressed are as follows:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

- Describing the necessary steps for data preparation, especially in assisting the model developer in selecting suitable data subsets through selection and manipulation either visually or using program code of the dataset, assisting with labeling efforts, as well as exploration and validation of the selected subsets;
- Describing the model development cycle, from understanding use case requirements to selecting appropriate machine learning models, training, evaluating, deploying and monitoring their performance in production;
- Defining the dimensions of model adoption in real world scenarios, from combining their findings and predictions with models of physical assets, to aspects revolving around explainability, fairness and robustness.

1.3 Content

The present report is divided in seven different chapters. In this section, a short description of each of them is provided.
Chapter 2 defines data analytics and artificial intelligence general workflows and sets the stage for the following chapters. Chapter 3 provides relevant guidelines for data preparation, drawn from the use case scenarios included in this deliverable. More specifically, it describes the purpose of data exploration, processing, version control and quality checks, as well as assisted labeling, feature extraction and selection, provides recommendations on how to conduct these stages and what relevant tools and techniques can be used. Chapter 4 discusses the steps of model development, including how to select appropriate models for a use case's requirements, how to train, evaluate, deploy and monitor such models successfully. Chapter 5 defines what model adoption in the real world means around several dimensions.
Chapter 6 describes a real world use case for automated visual inspection of civil infrastructures, in particular bridges. This use case revolves around the need to detect fine-grained defects in such structures from high-resolution images acquired with drone technology and processed with support from advanced artificial intelligence models.
Then, Chapter 7 describes another relevant use case for the continuous monitoring of event streams and sensor metric for the diagnosis and management of wind turbines, through anomaly detection and forecasting machine learning models.
Chapter 8 covers a third use case, for applying automatic penalties on overloaded vehicles by using intelligent neural weights in a motion system with high accuracy, and is followed by the list of references (Chapter 9).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

2 Artificial intelligence and machine learning workflows

Artificial intelligence and machine learning workflows are sequences of tasks that run subsequently one after another in any machine learning process, independent of data type, learning task or use case. A typical workflow (Figure 2.1) consists of the following phases:

- Data collection
- Data preparation
- Model building
- Model deployment

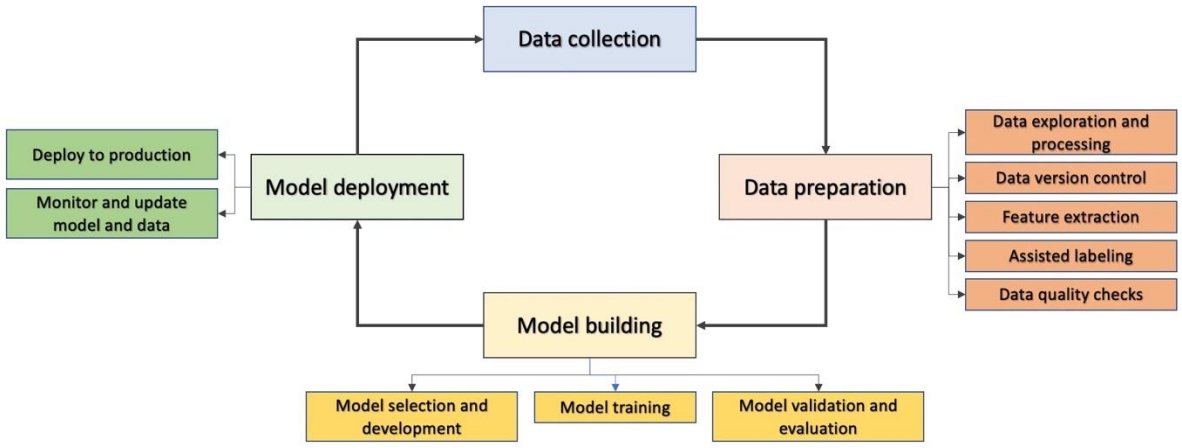

Figure 2.1 - AI / ML workflows

The first step in creating successful machine learning models and workflows is to understand the problem to be solved, characterize it and elicitate all the required knowledge from domain experts to help in collecting the relevant data and understanding the requirements of the use case. Different relevant independent variables and dependent variables need to be clearly identified by the domain expert. Independent variables include signals, control factors and noise factors while dependent variables represent the model response. Signals are stimuli required for fulfilling the model functionality. For example, in an automated visual inspection detection and segmentation model for bridges, signal is mainly the bridge picture taken by a pre-calibrated camera mounted on a drone during an inspection flight. Control factors are design parameters that can be controlled during data collection process and after deploying the model. Controlled factors may include camera resolution, pan, zoom, focus, sampling rate, color mode, distance to bridge, and so on. Noise factors influence the design but are controllable only during data collection process and are not controllable after deploying the model. The noise factors may include, but not limited to, scale changes, lightning conditions (illumination, shadows and reflectance), weather conditions, etc. Response is the primary intended functional output of the model. In this example, the output is the detection of defects in a bridge structure and the likelihood they present a level of risk that requires immediate repair and maintenance action. Finally, error states represent failure modes or effect of failure as defined by the end-user when using the predictive model. In this case, error states can be falsely detected defects (false positives) or missed defects (false negatives).

The understanding of each of these components (signal, control factors, noise factors, model response and error states) provides relevant guidance for every phase of the machine learning workflow:

- Under which conditions and when to collect data;
- What equipment to use for data collection;

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

- How much data to collect;
- How to clean and process the data;
- How to extract or select features from the data;
- Are labels needed for the learning task? If yes, how and how much data should be labeled;
- How to ensure data quality prior to model training;
- What learning model is most appropriate for the use case;
- What are the model alternatives and which to select;
- How to train, validate and evaluate the selected model;
- Where and how to deploy the model;
- Which techniques are most appropriate for monitoring and updating model and data;
- When to trigger updating and re-training of the data and model.

In rest of this document an overview of data preparation, model development and adoption in the real-world will be provided, detailing relevant stages and techniques.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

3 Data preparation

Since AI models are built and learned from data, the first step in building a model is data preparation — the process of extracting inputs to the model from data. There are a number of tools to help data scientists source data, transform data, add labels to datasets and apply quality checks. The data preparation phase is used to turn raw data into model input features used to train the model. Features are transformations on the cleaned data that provide the actual model inputs.

In the early stages of the pipeline, raw data is sourced across different data stores and lakes in an organization. The next stage involves data processing to clean, transform and extract features to generate consistent inputs in the feature selection stage. The data preparation stage involves a number of steps: sourcing data, ensuring completeness, adding labels, and data transformations to generate features, as well as performing data quality checks.

3.1 Data exploration and processing

Data exploration and processing is key to any machine learning process. Typically, the first phase is to assess quality of the data and develop a deep understanding of it. For instance, in use cases around automated visual inspection, which involve analyzing high-resolution images, a few common data problems to investigate during the data exploration phase and prior to model training include:

- Image dimensions and aspect ratios, in particular when dealing with extreme values;
- Labels composition, such as imbalance, bounding box sizes, aspect ratios for small objects;
- Data preparation and modeling approaches that are not specifically suitable for custom datasets, which tend to be significantly different than typical benchmark datasets such as COCO.

With respect to image dimensions and aspect ratios, most datasets would typically fall under the following categories:

- Uniformly distributed where most samples have the same dimensions – it is important to decide whether there is a need for resizing the samples and by how much, which depends on object area, size and aspect ratios. Resizing is particularly important for large images (over 4K+) which impose the problem of fitting them on GPUs due to memory constraints. An additional option to resizing is to train deep learning models in image patches rather than on the entire images themselves.
- Bimodal distribution where most images are in the aspect ratio range (0.7 – 1.5), similar to the COCO [19] dataset – A resizing operation through padding is generally recommended in this case.
- Extremely skewed datasets (wide images mixed with narrow ones) – Padding is typically excessive as it will generate a much too large dataset. However, sampling batches based on the aspect ratio is recommended. The mm detection framework, which is very popular amongst image processing frameworks, provides this functionality out of the box based on aspect ratios.

With respect to labels composition, it is important to know how their sizes and aspect ratios are distributed, especially since most models work well on benchmark datasets, but may not perform well on real-world data (which follows different distributions). For instance, if the size of the object to detect is very small, but enlarging the images would increase memory footprint and slow down the model training, an option is to apply a crop-size approach. On the other hand, if objects to be detected are very large, this is not usually problematic from a modeling perspective, but it is likely that classes with larger objects are underrepresented in the dataset.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Most of the time, the average area of the objects in a given class will be inversely proportional to their count.

Class imbalances typically pose a problem for object detection and segmentation tasks. When considering the problem of image classification, it is easy to oversample or downsample the dataset and control each class distribution to the model's loss. However, when datasets contain co-occurring classes, this becomes more challenging, since once resampling at the whole image level is started, multiple classes will be upsampled at the same time (not just one class).

Once data exploration is done, an integral part of any machine learning-based pipeline is to preprocess and augment data, as well as to select representative subsets of data in order to boost the model's training performance and in case large volumes of data are available.

Data augmentation is an essential and widely used regularization technique for a variety of tasks, including detection, segmentation and classification. Its purpose is to increase the size of the dataset by applying synthetic transformations to the existing samples. While for simple image classification, these transformations are applied only to the source images, in the case of more complex tasks, like detection and segmentation, the same set of transformations needs to be applied to the target as well (bounding boxes for detection and masks for segmentation). Traditional transformations that require both a source and target transform include:

- Affine transformations
- Cropping
- Rotation
- Distortions
- Scaling
- Changes to contrast or brightness
- Color jittering
- Etc.

All of the above transformations are supported in packages like albumentations, or in frameworks like OpenCV [20], PyTorch [21] or Tensorflow [22], to name a few. Full-fledged systems that provide already implemented pipelines for exploration, preprocessing and augmentation of data include Superb AI [23], databricks [24], Neptune AI [25] and others.

More advanced techniques for data augmentation can also be used, such as populating rain effect, sun flare or even adversarial noise. As a result of appropriate data augmentation, the performance of a deep learning model is expected to become more robust and accurate. In contrast to augmentations that encode invariances to data transformations, there also exists a class of augmentations that mix the information contains in different samples with appropriate changes to ground truth labels. A classic example is the mixup data augmentation [26] method which creates new data points for free from convex combinations of the input pixels and the output labels. There have been adaptations of mixup such as CutMix [27] that pastes rectangular crops of an image instead of mixing all pixels. The Mosaic data augmentation method is related to CutMix in the sense that one creates a new compound image that is a rectangular grid of multiple individual images along with their ground truths. Copy-Paste [28] combines information from multiple images in an object-aware manner by copying instances of objects from one image and paste them onto another image. Copy-Paste is akin to mixup and CutMix but only copying the exact pixels corresponding to an object as opposed to all pixels in the object's bounding box.

The selection of a subset of most representative samples from the initial dataset is necessary when large volumes of data are available or when class imbalance is present. The objective is to subsequently train a deep learning model with less, but more information-carrying data,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

in order to boost its performance. First, it is essential to understand the data at hand from the perspective of its underlying distribution, where there is redundant or biased data, which are the minority and majority classes, and whether edge cases are sufficiently represented. Such questions refer to common data challenges including the long-tail distribution problem (where particular categories or classes are less represented than others) and the existence of false positives and false negatives (where model predictions or labels falsely relate to or fail to relate a sample to a certain object or class).

An initial approach to sampling raw data is using unsupervised learning. It identifies clusters of similar samples by calculating the distance between them and allows for a simpler visualization and understanding of the dataset by reducing the dimensionality into 2D scatter plots. Several well-known can be used, such as UMAP [29], t-SNE [30] or PCA [31].

More sophisticated approaches include diversity-based and uncertainty-based sampling. Both types of methods are based on the metadata of the dataset, which should capture the requirements of what is a representative subset. These requirements are subject to a variety of parameters are that are specific to the task, industry, models used and resources. Diversity-based sampling chooses, based on distance between the samples in an embedding space, a diverse dataset that should cover all possible scenarios, to ensure that the model is even trained on edge and corner cases. Samples from different clusters are chosen using a diversifying algorithm, thereby ensuring the representation of different types of samples.

Uncertainty-based sampling is useful when a model shows good performance on some samples, but struggles on others. For instance, false positives and false negatives refer to situations where the model falsely assumes that a sample contains an object and when the model incorrectly assumes that a sample does not contain an object when in reality it does. Therefore, the model should be trained with challenging learning data, namely the data it struggles with from which it can subsequently improve. Through uncertainty-based sampling, the data where the model is uncertain is added to the training set. This creates a type of feedback loop, called active learning, where the output of the model (its performance) gives recommendations regarding its input (its training set).

In combination, metadata-based, diversity-based, and uncertainty-based sampling allow for an optimal selection of data. It enables uncertain data to be included within the training set while ensuring that similar samples are left out and metadata targets are met. In consequence, model performance is considerably improved.

3.2 Data version control

Keeping track of all the data used for models and experiments is essential and involved more than just managing and tracking files. The goal of a data version control system is to allow users to have unified data sets with a robust repository behind for all experiments. Additionally, it aims to systematize data versions, improve workflows and minimize the risk of occurring errors. Therefore, they ensure reproducibility, traceability and lineage of data, model and code.

Data versioning can follow various strategies, but at its core, it involves tracking the origins and changes over the history of data. These changes come in processing steps in a machine learning pipeline, such as data acquisition, merging, cleaning, transformation, learning and deployment. Some of the most used strategies employed are as follows:

- Caching copies of datasets – Every change to a dataset yields storing the new copy of the entire dataset. This strategy provides easy access to individual versions of data, but the memory constraint of small changes to big data require additional storage. With a name convention for files in given directory, this strategy is readily implementable. One tool that employs this strategy is Git [32].

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

- Storing the difference between datasets – It only stores changes between versions due to the memory constrain previously discussed. These differences are stored in patches, therefore restoring to a specific version rewuires adding all the patches to the dataset. Mercurial [33] uses such a strategy.

- Storing the changes relative to individual records – This strategy is implemented bu having individual history per record documents that track changes on each record. The implementation is by creating a separate file for each record, which combines to form an inidividual history per record directory that tracks all changes on individual parts. Amazon S3 [34] buckets use this particular strategy.

- Offsets in append-only datasets – Existing data is immutable and changes with new data are required to be appended to existing data. The file size and its offset asre used to track changes. This strategy, specifically used in Apache Kafka [35], is mainly addressed for append-only data such as event streams or log files.

The choice of strategy depends on the size, structure and frequency of updating the data.

In AI-powered projects which rapidly evolve through frequent experimentations and iteration, tracking model history is essential to understand and improve the performance of such developed models. Model versioning involves tracking of model inputs, hyperparameters, algorithm choice and architecture. These models are typically large binary data files. Finding the best model is an iterative process and in most cases, small changes to hyperparameters or data lead to different models. Tracking the version of such models can be done wth copy strategies supported in Git or DVC [36].

Further, tracking different stages of a pipeline, such as data preprocessinf, feature extraction and engineering, model training and hyperparameter optimization is essential as well. These can be captured as script and configuration files which can be versioned in the same manner as traditional source code. Tracking frameworks and library versions is also important in order to ensure reproducibility and avoid floating versions for dependencies. Package managers like pip support such versioning of dependencies. In addition, code and dependencies can also be packaged in virtual execution environments to ensure that environmental changes are tracked. Docker [37] is such an example.

Finally, experiments are various runs under different conditions, such as systematically varying variables and parameters. Additional focus here is on comparing results and visualizing experiment results. Some experiments are suitable withing a notebook (such as Jupyter notebooks), while other times more precise tracking is helpful, notably when exploring model options and their interactions. In the latter case, frameworks like TensorBoard [38] or Neptune AI provide an excellent set of capabilities that allow users to track their experiments in an interactive manner.

3.3 Feature selection and extraction

In many use cases, the data collected and prepared for a machine learning task is highly dimensional, namely it contains a large volume of variables that represent it. Performing feature selection and extraction in the context of preparing the data to use as input into a model has the objective of reducing the number of input variables and has a number of advantages:

- Improved model performance
- Reducing risk of model overfitting
- Faster training
- Improved data visualization
- Increased model explainability

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

3.3.1 Feature selection

Feature selection aims instead to rank the importance of the existing features in the dataset and discard less important ones. There are numerous techniques that can be employed for feature selection depending on the type of learning task and model.

Feature importance. Tree type of models, such as decision trees or random forest, can be used to rank the importance of the features used as input to a model. By providing this type or feature ranking, a user can understand how a model makes its predictions and which features have contributed to those predictions. Not only does it allow to remove features which are not bringing any benefit to the model, but it increases the model's explainability relative to its decision-making process.

Recursive feature elimination. This technique takes as input the instance of a machine learning model and the final desired number of features to use. It then recursively reduces the number of features to use by ranking them using the model's accuracy as a metric. Iterating through the number of input features as an input variable, it is possible to find out the optimal number of features a model needs by keeping track of its accuracy in every iteration.

Correlation matrix analysis. A traditional approach is to inspect the correlation of all input features to the output labels. By using Pearson correlation [39], for instance, the returned coefficient values will provide an indication of no correlation, positive or negative correlation:

- If the correlation between two features is 0, this means that changing one of these two features will not affect the other, therefore there is no correlation detected;
- If the correlation is greater than 0 (max is 1), then the features are positively correlated and therefore the increase in values of one feature will determine an increase in values of the other feature;
- If the correlation is lower than 0 (min is -1), then the features are negatively correlated and therefore an increase in values of one feature will determine a decrease in values of the other feature.

Univariate selection. This technique is a statistical method used to select the features which have the strongest relationship to the output labels. A user can decide which metrics to used to evaluate the features and the number of K best features to be retained. If the task at hand is classification, then common scoring functions are chi2, f_classif, mutual_info_classif, whereas for regression the typical functions are f_regression and mutual_info_regression.

Lasso regression. Regularization techniques are commonly used to penalize a model's parameters in an attempt to avoid that a models tried to resemble too closely the input data. In this way, the model can be simplified and is typically used when faced with overfitting problems. One type of regularization is Lasso (L1) regression [40]. When using Lasso regression, the coefficients of the input features are diminished in value if they are not positively contributing to the model training. Therefore, it is possible that some features are automatically discarded by assigning them coefficients equal to zero.

3.3.2 Feature extraction

Feature extraction aims to reduce the number of features in a dataset by creating new features from the existing ones and then discarding the original features. These new reduced set of features should then be able to summarize most of the information contained in the original set of features. In this way, a summarized version of the original features can be created from a combination of the original set.

Principal component analysis (PCA). PCA is a linear dimensionality reduction technique which takes as input the original data and tries to find a combination of the input features that bets

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

summarized the original data distribution. The objective is to reduce its original dimensionality. PCA achieves this by maximizing variances and minimizing the reconstruction error by inspecting pairwise distances. The data is projected into a set of orthogonal axes and each of the axes is ranked in order of importance. While popular, it does not consider data labels (being unsupervised) and only looks at variation, therefore there is a risk that it can lead to data misclassification. Additionally, trying to map the generated principal component features to the original ones is not necessarily trivial.

Linear discriminant analysis (LDA). Unlike PCA, LDA [41] is a supervised dimensionality reduction technique. It aims to maximize the distance between the mean of each class and minimize the heterogeneity within each class, which makes it better for classification tasks than PCA. However, due to its assumption that the input data follows a Gaussian distribution, poor classification results can be achieved if this assumption is broken.

Locally linear embedding (LLE). While PCA and LDA perform well in case of linear relationships between different features, LLE [42] is based on manifold learning and well suited for non-linear cases. A Manifold is an object of D dimensions which is embedded in an higher-dimensional space. Manifold Learning aims then to make this object representable in its original D dimensions instead of being represented in an unnecessary greater space.

t-distributed Stochastic Neighbor Embedding (t-SNE). t-SNE is a non-linear dimensionality reduction method typically used to visualize high dimensional datasets. It works by minimizing the divergence between a distribution constituted by the pairwise probability similarities of the input features in the original high dimensional space and its equivalent in the reduce low dimensional space. t-SNE makes then use of the Kullback-Leiber (KL) divergence [43] in order to measure the dissimilarity of the two different distributions. The KL divergence is then minimized using gradient descent. When using t-SNE, the higher dimensional space is modelled using a Gaussian Distribution, while the lower-dimensional space is modelled using a Student's t-distribution. This is done, in order to avoid an imbalance in the neighbouring points distance distribution caused by the translation into a lower-dimensional space.

Autoencoders. They are machine learning algorithms that use non-linear transformations to project data from a high dimension to a lower one, unlike all other dimensionality reduction techniques. There exist a variety of autoencoders, such as variational, denoising, convolutional, sparse, and so on. However, irrespective of the type of autoencoder, its architecture can be broken down into 2 main components:

1. Encoder – takes the input data and compresses it, to remove as much as possible noise and redundant information. The output of the encoder is an embedding of the input data into a lower dimensional latent space;
2. Decoder – takes as input the encoded latent space and attempts to reproduce the original inputs.

To determine how well an autoencoder is able to reconstruct input data, the most straightforward approach is to compute the mean reconstruction error between the reconstructed samples and original ones.

3.4 Assisted labeling

Since many machine and deep learning tasks require annotated data to be trained on, there is an increasing need for tools that can provide assistance with annotation and labeling. In particular in the case of civil engineering, where data primarily consists of high-resolution images, the objective is to annotate either by bounding boxes or fine-grained masks various objects or defects of interest. As such, tools like CVAT [44] and many others provide labeling capabilities as platforms that are readily available to ease the burden on data annotators and data scientists alike.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Depending on the use case, every user can have different requirements from their annotation platforms. However, there are some common aspects that are generally desirable across all platforms:

1. Automation – Annotation is a repetitive task when done entirely manually. Therefore, automating parts of the process will not only reduce average annotation time, and as a result the budget spent on it, but also keep annotators more engaged, as they can look at more new data in a similar timespan.
2. Collaboration and quality control – For a lot of AI related tasks, datasets can be quite large. It may not be feasible or desirable for a single person to annotate everything, especially since quality and collaboration often go hand-in-hand. Peer review is a common process to detect issues and improve annotation quality. Additionally, a dataset may be used to test multiple types of AI models. For instance, one data scientist may need semantic or instance annotations for segmentation tasks, while another may need bounding boxes for object detection. To overcome such challenges in the most effective way, generally whole teams develop an annotation strategy together, including a review process to ensure that every team member adheres to it. Therefore, it is all the more important for annotation tools to provide collaboration and quality control capabilities.
3. Facilitating understanding of the dataset – In order to develop high quality models, data scientists need to acquire a good and deep understanding of the dataset. Statistical information as well as filtering tools are valuable aids in this process.
4. Versatility – Although annotation tools are providing more and more capabilities, they are most often part of a larger toolchain that is used within a project. Being able to cope with changes in that toolchain is an important advantage. For instance, being able to answer questions such as “Can we adapt annotations to a specific format required by the AI model?” is relevant. Another important factor is how much control is available for deploying the tool. Can the data and services be self-hosted, instead of purely web-based?
5. Annotation capabilities – Annotation tools always imply a learning curve, especially for first time annotators. If an annotation tool supports a rich ser of features, it becomes more likely that a user will continue using it for different use cases that require different types of annotations.

Computer Vision Annotation Tool (CVAT) is an open-source tool for annotating digital images and videos. The main function of the application is to provide users with convenient annotation instruments. It is deployed via Docker and accessed via a browser-based interface and features a task-based management system optimized for collaborative work, such that users can create public tasks to split up tasks between themselves. It supports supervised machine learning tasks pertaining to object detection, image classification and image segmentation, while annotations can be done by using one of four shape types, namely boxes, polygons, polylines and points.

CVAT allows one to annotate data for each of these cases. There are some advantages and disadvantages of the tool. Among its advantages, the tool is web-based, which implies that users do not need to install the application, although the option to have it installed on-premised is also available. It is also collaborative, easy to deploy and provides support for automatic annotation. Within CVAT (Figure 3.1), annotators have a wealth of tools at their disposal for copying and propagating objects, applying filters, adjusting visual settings, performing automatic annotation via the object detection API in Google’s TensorFlow framework, and more.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 958171.

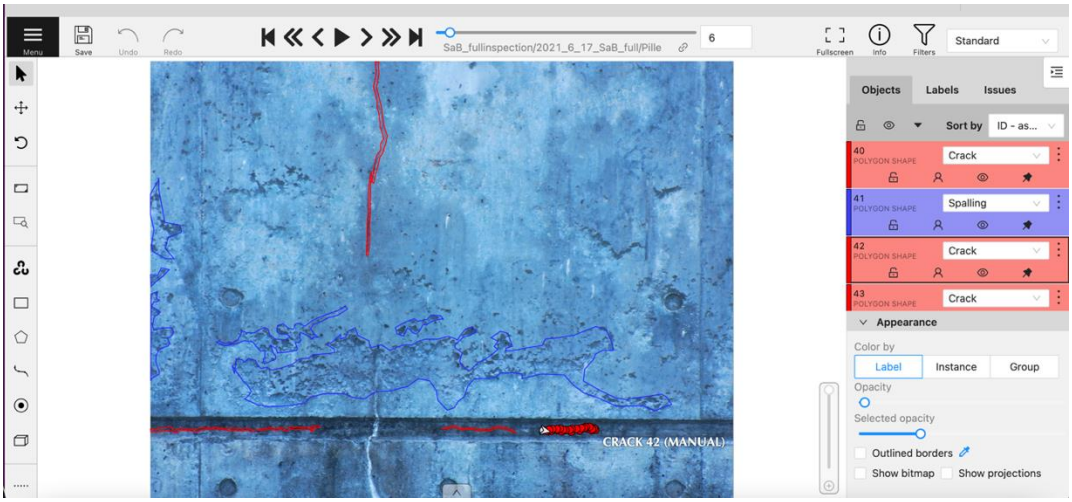

Figure 3.1 - CVAT used to label defects in a bridge structure for the automated visual inspection scenario

1. Automation – For semi-automatic annotation, a method called DEXTR [45] is provided, which works by selecting a few extreme points of an object, after which the complete segmentation is derived. CVAT is also integrated with OpenCV, an open source project around computer vision, allowing users to trace an object rapidly with visual feedback. Fully automatic annotatation capabilities annotate all objects in an image, but it is likely a user will need to perform corrections to arrive at the desired output.
2. Collaboration and quality control – Division of work and reviews are supported. Discussions among various annotations are supported as well in the form of adding comments directly on the images.
3. Facilitating understating of the dataset – CVAT includes powerful options to set and combine filters, but visually it is less intuitive compared to other platforms. This is mostly due to the lack of a thumbnail overview that matches the filter settings. Statistics are minimal and only available for jobs assigned to annotators, rather than project wide.
4. Versatility – CVAT offers Docker images to easily deploy the server and client sides. The user has full control over the entire platform and it is such free to decide how and where the service and store the data. There is no plugin system to extend functionaility, but it does offer a REST API to interact with other platforms and tools.
5. Annotation capabilities – CVAT supports polygon annotations but notably lacks a brush tool. Especially with a mouse, a polygon tools works quite well, but a brush can be useful to make corrections and many users are familiar with such tools. Besides polygons, cuboids, polylines and points are also supported. Tags are supported for both annotations and entire images, and the following values are provided as options: Boolean, one of several values, text or numbers.

Supervisely [46] is a powerful platform for computer vision development, where data scientists can not only annotate datasets, but also experiment with various deep learning models. the platform excels in its modularity. Its ecosystem allows users to tailor the platform to their needs. Custom visualization can be added directly to the data itself using IPython notebooks. Furthermore, custom data transformation steps can be created using the built-in DTL (Data Transformation Language) or python scripts.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

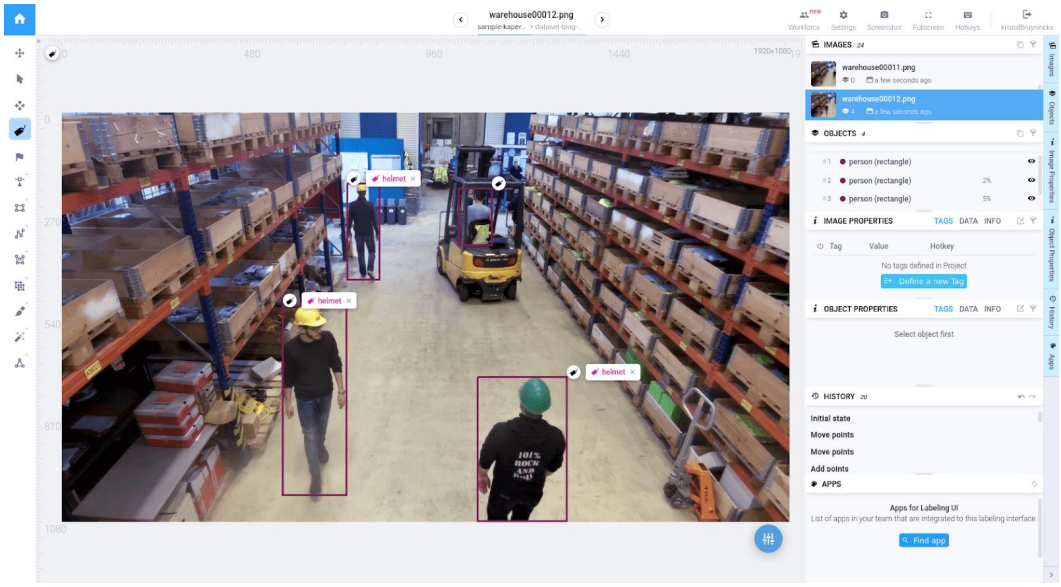

Figure 3.2 - Supervisely used for annotation

1. Automation – A semi-automatic annotation tool is provided that works out of the box with a generic model and is extremely intuitive to use. Rather than having to trace a complex object with the polygon or brush tools, a user can just indicate a bounding box. The tool will then find the dominant object within that box. If there are corrections to be made, it is easy to indicate points that should be annotated and points that should not, while the annotation is updated automatically. Fully automatic annotation can also be used, by using readily available pre-trained deep learning models or training models imported by users on their own data.
2. Collaboration and quality control – Both features are supported, with one feature worth mentioning, which is the issue tracker.
3. Facilitating understanding the dataset – Each project offers detailed tabular statistics on annotations, image tags, object tags and so on. In addition, applications can be used to generate more visual reports about relevant distributions in a dataset. While going through data, it is easy to use filters and zoom in on specific classes, tagged objects, images containing issues, making it more intuitive to use than other tools, such as CVAT.
4. Versatility – Supervisely offers an SDK (Software Development Kit) to develop own applications within its ecosystem, when the features desired are not readily available. A REST API is also supported. There is an additional option to host agents for some tasks, but a fully self-hosted solution requires an enterprise subscription. Data can be stored on major cloud providers, such as Google Cloud, Amazon Web Services S3, and others.
5. Annotation capabilities – The platform supports polygon, brush, polylines, cuboid and point annotations. Notably, holes can be created in a polygon annotation by subtracting other polygons, a feature missing in CVAT. Tags of the following types can be created: boolean, one of several values and numbers. These can be applied to both images and individual annotations.

Similar to Supervisely, **Hasty** [47] also provides a full-rounded platform that not only supports data annotation, but also training and evaluation of deep learning models for various computer vision tasks. However, while Supervisely is a purely subscription-based service, Hasty employs a different business model. It is a mix of subscription and pay-for-what-you-use.

1. Automation – For semi-automatic annotation, the same DEXTR tool used is CVAT is supported. Additionally, a notable feature is the automatic conversion of bounding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

boxes to segmentations. There are additional non-AI tools supported, such as color based magic wand and a contrast contour selection tool. For fully automated applications, an approach to maximize user convenience is taken, as no setup is required at all to use AI assistant features. Once enough samples are manually labeled, these assistants become available automatically, trained on previous annotations. The tool even includes AI assistants to automate object and image tagging.

2. Collaboration and quality control – Images can be marked as requiring a review, but issues or comments cannot be added to specific areas of an image, which is a relevant downside of the tool. Work can also not be divided within the tool, for example splitting a large set into several jobs and assigning different users and reviewers.
3. Facilitating understanding of the dataset – Basic statistics are provided in a tabular form. To filter, a separate review module is provided. The tiled view gives a nice overview of all annotations that match the filter settings. It is possible to filter on object tags, but multiple tags cannot be combined.
4. Versatility – Hasty offers third-party clouf storage for Azure blob storage, Google Cloud and Amazon S3. A fully self-hosted solution can be requested as well. An API is offered for integration into an existing pipeline of new features and applications. Export format options are rather limited though: a custom Hasty format, PNGs supplemented with JSON and COCO.
5. Annotartion capabilities – Support for polygon and brush annotations. Polygon tools cannot be used to create annotations with holes, in a similar manner as Supervisely. Annotation tags are types in a variety of types: boolean, text, number, single choice, multiple choice, however they can only be applied to individual annotations. Image annotations are available only in a Boolean form.

Of course, many other tools supporting or dedicated for annotations exist, apart from CVAT, Hasty and Supervisely. Notable to mention are LabelBox [48], SuperAnnotate [5], v7 Darwin [49], Diffgram [50] and Datatorch [51] due to their advanced features and popularity among business customers, as well as LabelMe [52] and VoTT [53], which like CVAT, are free and open source.

3.5 Data quality checks

Poor data quality leads to technical and architectural bottlenecks, which means that performing quality checks is essential within an entire machine learning pipeline from the data preparation stage to the model development stage and after model deployment.

While machine learning models rely on high-quality data, maintaining data quality can be challenging. The upstream data changes and the increasing pace of data proliferation can drastically change a model's overall performance. In case of structured data, such quality checks can verify that feature data is not missing, catch when data deviates from a specified range or exceeds a set threshold and detect extreme model inputs and outputs.

In contrast, strong data quality assessment methods for unstructured data (such as images, videos, speech, text) are somewhat lacking. Current solutions have a broad set of capabilities that can be used by data scientist to assess their data's quality at all stages of a pipeline:

- Statistical analysis – Tools enable summary statistics of key metadata of a dataset in one compact view (e.g., size, count, distribution, source, etc).
- Cleaning – Users can check for duplicates, remove outlies, fix structural errors and check for missing values with support from automatic frameworks.
- Labeling editing – Distinguish between easy and hard labels, ensure label consistency and audit the correctness of labels.
- Intuitive UI – Access to built-in functions to streamline end-to-end data quality tests and monitoring, such as MLFlow [54]. When data visualization is the main objective,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

frameworks like Tableau [55] or Microsoft Power BI [56] are well suited for both structured and unstructured data.

3.6 Dealing with Big Data

When dealing with massive amounts of data, potentially in the range of TB, the biggest challenge is managing such volume of data, from the point of view of storage, processing and analysis. Big data analytics follows the same steps regarding data preparation as before, therefore the logical steps are the same: collection, exploration and processing, version control, cleaning and feature engineering, labeling if needed and ensuring data quality. With the explosion of data, early innovation projects like Hadoop [57], Spark [58], and NoSQL [59] databases were created for the storage and processing of big data. This field continues to evolve as data engineers look for ways to integrate the vast amounts of complex information created by sensors, networks, transactions, smart devices, web usage, and more.

Object and NoSQL databases are the best solution for storing big data, since such data is collected from a variety of sources – from cloud storage to mobile applications to IoT sensors and beyond – and is both structured and unstructured. Data warehouses and data lakes are typically the go-to technologies to collect such a variety of data, maintain its metadata and statistics on its provenance and updates.

Once data is collected, processing it is challenging simply because of its sheer volume. The two most relevant types of processing that can be used in this case are batch processing and stream processing. Batch processing inspects and processes large data blocks at a time and is useful when there is a longer turnaround between collecting and analyzing data. Stream processing looks at small batches of data at once, shortening the delay time between collection and analysis for quicker decision-making. However, it is more expensive and more complex to manage.

Several types of tools work together to help you collect, process, cleanse, and analyze big data. Some of the major players in big data ecosystems are:

- **Hadoop** is an open-source framework that efficiently stores and processes big datasets on clusters of commodity hardware. This framework is free and can handle large amounts of structured and unstructured data, making it a valuable mainstay for any big data operation.
- **NoSQL databases** are non-relational data management systems that do not require a fixed scheme, making them a great option for big, raw, unstructured data. NoSQL stands for “not only SQL,” and these databases can handle a variety of data models.
- **MapReduce** [60] is an essential component to the Hadoop framework serving two functions. The first is mapping, which filters data to various nodes within the cluster. The second is reducing, which organizes and reduces the results from each node to answer a query.
- **YARN** [61] stands for “Yet Another Resource Negotiator.” It is another component of second-generation Hadoop. The cluster management technology helps with job scheduling and resource management in the cluster.
- **Spark** is an open source cluster computing framework that uses implicit data parallelism and fault tolerance to provide an interface for programming entire clusters. Spark can handle both batch and stream processing for fast computation.
- **Tableau** [55] is an end-to-end data analytics platform that allows you to prep, analyze, collaborate, and share your big data insights. Tableau excels in self-service visual

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 958171.

analysis, allowing people to ask new questions of governed big data and easily share those insights across the organization.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

4 Model building and deployment

During the modeling step, one or multiple machine learning models are selected, trained, validated and tested. The common approach to build good model is try to different algorithms and compare their performance. In the ideal scenario, the model to be selected for deployment is fitting well the historical data (low bias) and at the same time generalizes well to unseen data (low variance). Keeping in mind the bias-variance trade-off helps in building successful machine learning models.

4.1 Model development

Model creation is typically the longest stage of the development process. The goal in this step is to achieve a high degree of model accuracy, as much as possible. Three primary components determine the accuracy of machine learning models: the fit of the algorithm and the completeness of the feature set. The modeling process continues by iteratively making improvements to these factors until the required accuracy is achieved, or a progress plateau has been observed.

Determining the completeness of the feature set is the primary objective of the data preparation stage (Chapter 6) and it is completed prior to model development.

Most importantly to model creation is to select an appropriate algorithm. The algorithm is the procedure that is executed on the training data to create – or train – the model. There are literally hundreds of machine learning algorithms available to data scientists. The correct algorithm for a given machine learning problem is the prerequisite for a good model that can then become a good tool for real-world use cases. Apart from choosing an appropriate class of algorithms for a machine learning problem, there exist several other important considerations when deciding for a particular model:

- Performance – the quality of a model's results is fundamental to be taken into account when choosing a model and depending on the class of model, different metrics can be used to assess performance: accuracy, precision, recall, F1-score, mean squared error, mean average precision, intersection over union, etc.
- Explainability – explaining a model's results is paramount to its deployment and adoption in the wild. Understanding how easy it is to interpret the model outputs is important before choosing a good model candidate, although depending on the use case the best performing models might be the ones with a higher degree of complexity.
- Complexity – a complex model is able to learn more interesting patterns from the data, but at the same time it is harder to explain and maintain. Typically, complexity and explainability are inversely proportional.
- Dataset size – the amount of training data available dictates whether it is possible to consider models like neural networks which are excellent at processing and synthesizing large volumes of data or simper models should be chosen. Going beyond the amount of available data, another important consideration is how much data is indeed needed to achieve high performance. It may not always be the case that large volumes of samples are paramount to a use case.
- Dimensionality of data – this can be regarded in two different ways: the vertical size of a dataset represents the number of samples present in a dataset; the horizontal size represents the number of features. More features will often lead to better model outputs, but they will also increase the complexity of the model.
- Training time and cost – how long does it take and how much does it cost to train a model? What computational resources does it need? For instance, if the requirements and data of a use case are subject to frequent changes, models with long training

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

cycles are not a feasible choice, since the model will need to be re-trained on a more frequent basis.
- Inference time – how long does it take to run a model and make a prediction? Long runs cannot be considered for use cases that are centered around real-time or near-real-time decisions.

4.1.1 Supervised learning

Supervised learning algorithms make predictions based on a set of examples. Each input variable consists of labeled training data and a desired output variable. The model analyzes the training data to learn the function that maps the input to the output. The inferred function maps new, unknown example by generalizing from the training data to anticipate results in unseen situations.

- Classification – when the label to predict is categorical. Examples: naïve Bayes [62], SVM [63], decision trees [64], logistic regression, random forest [65], gradient boosting trees [66], kernel SVM [67], neural networks;
- Regression – when predicting continuous values. Examples: decision trees, linear regression, neural networks, random forest, gradient boosting trees;
- Forecasting – when predictions about the future are made based on past and present data and is most commonly used to analyze trends. Examples: neural networks, CART regression trees [68], Gaussian processes [69].

4.1.2 Semi-supervised learning

The biggest challenge with supervised models is that labeling data can be extremely expensive and time consuming. If labels are limited, unlabeled examples can be used to enhance supervised learning, therefore applying semi-supervised methods. Ideally, the vast majority of data used to learn from are unlabeled and only a small fraction of labeled data is used.

4.1.3 Unsupervised learning

When data available is completely unlabeled, the only option is to use unsupervised learning models. The objective of such models is to discover intrinsic patterns that underline the data, such as a clustering structure, a low-dimensional manifold, a graph or a sparse tree.

- Clustering – grouping samples such that similar examples are clustered together according to a set of criteria and distance metrics. This is used to segment the entire dataset into multiple groups and further analysis can be performed in each group to find intrinsic patterns. Examples: dbscan [70], k-means [71], hierarchical clustering [72], Gaussian mixture models, [73];
- Dimensionality reduction – due to the high dimensionality of input data, some features may be redundant or irrelevant for the task at hand. Reducing the number of variables under consideration helps to find true, latent relationships and is commonly used in the data preparation stage. Examples: PCA, t-SNE, UMAP, LDA, etc.
- Anomaly detection – detecting abnormal pattern of behavior in the data relative to an established baseline of normal behavior. Examples: one-class SVM, isolation forest, autoencoders, neural networks.

4.1.4 Reinforcement learning

Reinforcement learning is used mainly for sequential decision-making problems. Unlike for any previous type of learning, there is no need to have data in advance. Instead, the learning agent interacts with the environment and learns the optimal policy on the fly based on the feedback it received from that environment. The feedback from an agent's action has many important components. One component is the resulting state of the environment after the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

agent has acted on it. Another component is the reward (or punishment) that the agent receives from performing that particular action in that particular state. The reward is carefully chosen to align with the objective for which we are training the agent. Using the state and reward, the agent updates its decision-making policy to optimize its long-term reward.

4.2 Model training, validation and evaluation

Once a model has been developed for a particular use case and learning task, training is a process that tries to fit the best weights and biases to the model in order to minimize its loss function. Loss functions define how to optimize machine learning algorithms. Depending on the task, objectives, type of data and model, various loss functions can be used. In supervised learning, model training creates a mathematical representation of the relationship between the data features and the target labels. In unsupervised learning, it creates a mathematical representation among the data features themselves.

The objective of model validation and evaluation is to understand how the model may perform on new data. During the previous phase, training, a model learns to predict accurately by understanding the data at hand. However, instead of understanding the underlying structure in the data, a model may actually memorize the data, leading to overfitting. When overfitting goes undetected during training, it will resurface when put in production. Validation and testing techniques are meant to mitigate such problems as early as possible.

Fundamentally, model evaluation is typically based on the three-way holdout method, which consists of three stages each with a corresponding dataset:

- Training set – used for deriving relationships in the training data;
- Validation set – used for an unbiased evaluation of the model fit during hyperparameter tuning, model selection and error analysis;
- Test set – user for the final, independent evaluation with data the model has not seen during training or validation.

The steps of the holdout method are as follows (Figure 4.1):

1. Split data into training, validation and test sets;
2. Train the model on the training set with different hyperparameter settings;
3. Evaluate the model performance on the validation set and select the hyperparameters with the best performance on this set;
4. Optionally, train a new model on the combined training and validation set, using the best hyperparameter values from the previous step;
5. Test the model in the independent hold-out set;
6. Retrain the model on all the data and decide whether to deploy model in production.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

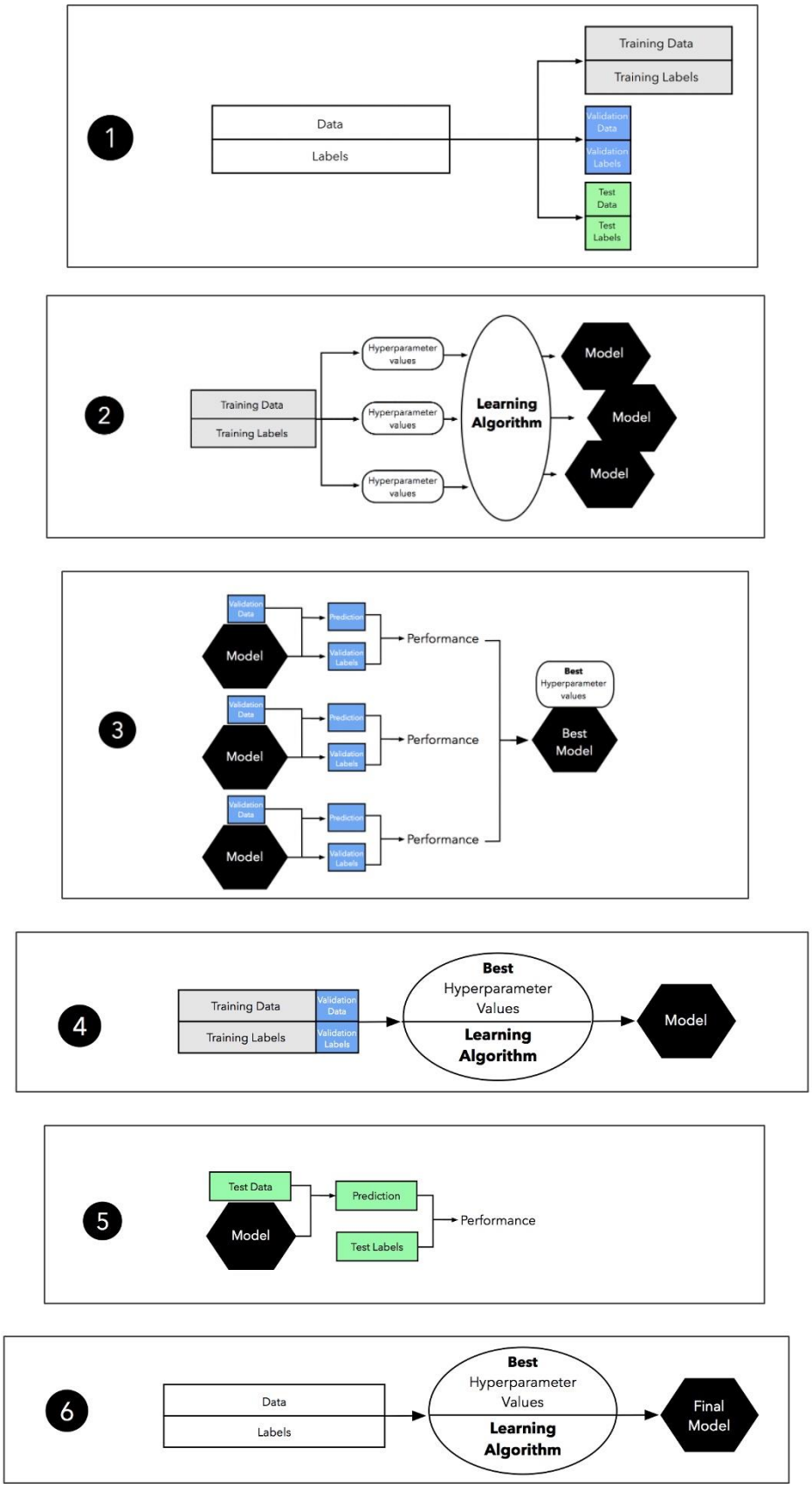

Figure 4.1 - Steps of the holdout method

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

4.2.1 Model fitting

When fitting a machine learning model on training, validation and test sets, the typical problems encountered are overfitting or underfitting.

Overfitting occurs when a model fits exactly against its training data. When this happens, the algorithm unfortunately cannot perform accurately against unseen data, defeating its purpose. If the model trains for too long on the training set or when the model is too complex, it will learn "noise" or "memorize" the data. Low error rates and a high variance are good indicators of overfitting. The validation and testing sets put aside are used to check for overfitting.

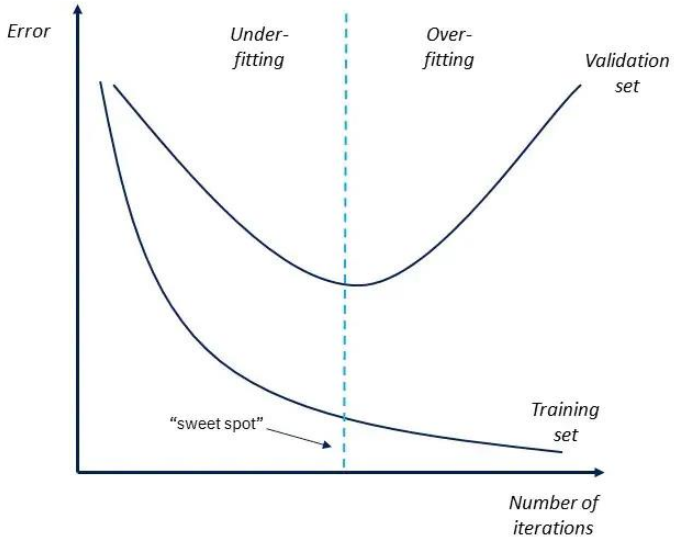

Figure 4.2 - Underfitting vs. overfitting

If overtraining or model complexity results in overfitting, then a logical prevention response would be either to pause training process earlier, also known as, "early stopping" or to reduce complexity in the model by eliminating less relevant inputs. However, if one pauses too early or excludes too many important features, then one may encounter the opposite problem, and instead, may underfit the model. Underfitting occurs when the model has not trained for enough time or the input variables are not significant enough to determine a meaningful relationship between the input and output variables.

In both scenarios, shown in Figure 4.2, the model cannot establish the dominant trend within the training dataset. As a result, underfitting also generalizes poorly to unseen data. However, unlike overfitting, underfitted models experience high bias and less variance within their predictions. This illustrates the bias-variance tradeoff, which occurs when as an underfitted model shifted to an overfitted state. As the model learns, its bias reduces, but it can increase in variance as becomes overfitted. When fitting a model, the goal is to find the "sweet spot" in between underfitting and overfitting, so that it can establish a dominant trend and apply it broadly to new datasets.

In particular, in domains such as automated visual inspection and civil engineering, where the dimensionality of the data is large and complex deep learning models are the standard choice, overfitting is the more common encountered problem. Irrespective of the learning task, be it object detection, instance segmentation, classification or others, there exist a few mitigation techniques to reduce overfitting:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

- Early stopping – pause training before the model starts learning the noise within the data;
- Train with more data – expanding the training set to include more data can increase model accuracy by providing more opportunities to learn the dominant relationships between input and output;
- Data augmentation – noisy data injected into a model might increase its robustness;
- Feature selection – identify the more relevant features by eliminating the irrelevant and redundant ones;
- Regularization – apply a penalty to the input features with larger weights to limit the amount of variance in the model. L1 and L2 regularization, as well as dropout are the most common regularization techniques used;
- Ensemble methods – using bagging and boosting techniques out of simpler models whose outputs are aggregated into a single prediction output.

4.2.2 Model error analysis

Depending on the type of learning task at hand, there exist a variety of performance metrics used to evaluate how accurate a model is. In classification, precision, recall, F-scores and confusion matrices are commonly used. In regression, mean squared error, mean absolute error and root mean squared error are the norm. In the case of detection and segmentation tasks, models report mean average precision and intersection over union. These metrics have been detailed in D2.1.

Irrespective of the task at hand, a recurring problem with model assessment practices is relying on such aggregate metrics to score models on entire benchmarks or datasets. In practice, it is difficult to understand a model's behavior from one single number. At the same time, for data scientists and business practitioners alike it may be interesting to take a deep dive across several dimensions of the input feature space and ask questions such as "*What happens to the accuracy of the segmentation model when the defect is small and its aspect ratio is extreme?*" or "*Which model achieves better localization of defects?*". Navigating the terrain of failures along multiple potential dimensions related to the task at hand, the model and the data is challenging.

As a result, model error analysis is an invaluable approach to shedding light unto how models fail, how failures are distributed and given the choice between multiple models, which is the best option relative to use case objectives.

Notably, the Error Analysis toolkit [74] developed at Microsoft Research provides model debugging error capabilities, with active data exploration and interpretability techniques. It allows users to identify and diagnose error patterns across data slices with the condition that the input data is structured.

The case for model error analysis for unstructured data is more complex, and in particular for detection and segmentation tasks that are most common in automated visual inspection scenarios for civil engineering. Typically, mean Average Precision (mAP) is the go-to metric for such tasks. However, mAP suffers from complexity issues. It is defined as the area under the precision-recall curve for detections at a specific intersection-over-union (IoU) threshold with a correctly classified ground truth (GT), averaged over all classes. The standard has become to compute mAP over 10 IoU thresholds (interval of 0.05) to get a final $mAP^{0.5:0.95}$. The complexity of this metric poses a particular challenge as error types become intertwined, making it difficult to gauge how much each error type affects mAP. Moreover, by optimizing for mAP alone, it may be inadvertently leaving out the relative importance of error types that can vary between applications and use cases.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Frameworks like HOIEM [75], TIDE [76], UAP [77] or COCO Eval [78] provide advanced capabilities to understand model errors beyond mAP, by categorizing and summarizing errors for detection and segmentation tasks into fine-grained categories based on binning false positives and false negatives. For instance, TIDE defines 6 types of relevant error types: classification, localization, classification + localization, duplicate, background and missed ground truth, as shown in Figure 4.3.

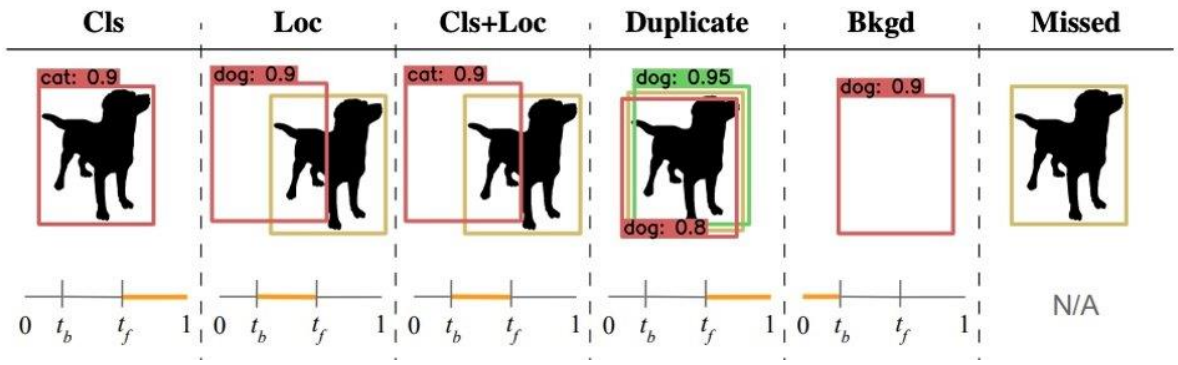

Figure 4.3 - 6 types of model errors outputted by TIDE (from [77])

Weighting the errors depending on the use case objectives (e.g., localization errors are more important than localization errors) and computing their influence on the overall mAP, it is possible to understand exactly where models fail and by how much from the ideal performance. Moreover, such a deep analysis can be done for multiple models or model versions (i.e., an alternative to model readiness techniques in Chapter 7.5.3), thus allowing an informative comparison which is critical in choosing the model that best suits a particular use case. Finally, fine-grained capabilities, such as analyzing model errors relative to specific attributes (e.g., size of a defect, aspect ratio of a defect) offer an even deeper insight into how a model will be expected to perform once deployed.

4.3 Model deployment

After a model has been validated and tested, it is deployed into a live environment to be used in various use cases for its intended purpose. The environment provides the model with the necessary hardware resources for running as well as access to the data sources that it draws data from. Then, it is integrated into a process or software, such that it is made accessible to users from their respective endpoints.

Typical options for model deployment include on-demand prediction or batch prediction modes. In some more recent cases though, where new data is constantly being acquired, models can be deployed in an embedded mode on edge and mobile devices. An illustrative example is the scenario of on-the-fly detecting defects on a car manufacturer's production line which is done by taking pictures of car components and running the defect detection model on the same mobile device. The decision to stop the production line is based exclusively on the model's output and is done in a matter of seconds. Had such a model run in batch mode in a cluster, the decision to stop the production line would be significantly delayed.

Each type of model deployment serves different purposes.

4.3.1 Batch prediction mode

In the batch scenario, models are run offline. Their compute cost is minimized, since there are fewer dependencies on external data sources and remote services. The local processing power is most times sufficient for computing algorithmically complex models and allows to debug them when failures occur or to tune hyperparameters. Batch mode implies a partitioning

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

of the data into segments that are processed sequentially. This can be achieved by using sampling schemes, like balanced or stratified sampling, or via online algorithms, such as Map Reduce.

Offline models can be optimized to handle a high volume of job instances, through scheduling with frameworks like Airflow [79] or Prefect [80].

4.3.2 On-demand prediction mode

In online mode, machine learning models are deployed via web services, by using frameworks like Flask. Once a model has been validated and tested, in on-demand mode, it first needs to be persisted. Libraries like scikit-learn offer off-the-shelf persistence and restoration functionalities. Then, the persisted model is served to a web framework which will make it available to users through a REST API. The benefit of online deployment mode is that it is typically cheaper and can provide near real-time predictions when compared to batch mode. Availability of CPU or GPU power is less of an issue if the model runs in a cluster or cloud service, since it is easily made available through API calls.

4.3.3 Embedded/Edge prediction mode

To reduce latency and data bandwidth consumption, models can be run closer to the user, on mobile or IoT devices. However, such devices have limited computation power and storage capacity and therefore complex and large models cannot be directly deployed. Instead, using quantization or aggregation techniques, edge-deployed models can be simplified while maintaining acceptable accuracy. Platforms like TensorFlow Lite [81] can be used to simplify TensorFlow models.

4.4 Monitoring of model and data in production

Once a model is deployed in production, in order to eventually be adopted, its performance needs to be constantly evaluated on real-world data. In fact, models tend to become stale over time and proper monitoring of both model and data will provide invaluable information when performance decreases and the model needs retraining or the data needs to be updated (shown in Figure 4.4).

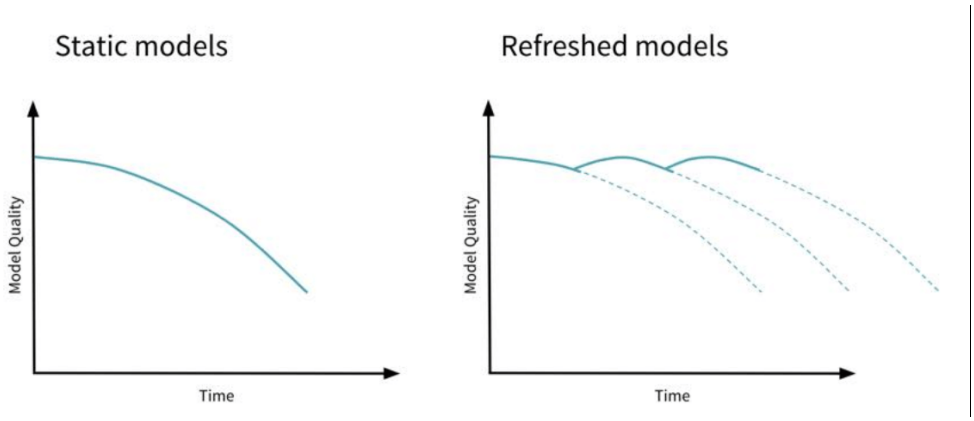

Figure 4.4 - Model staleness and performance degradation in time

Some of the challenges a model will encounter in production are:

- Data and feature drifts, which can lead to training vs. evaluation skew, where a model achieves poor performance in production despite rigorous testing and validation during development phase;
- Model drift, where a high performing model in production suddenly experiences a performance dip over time;

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

- Black-box models, where a model's predictions are difficult to interpret with respect to business objectives (Chapter 8);
- Model readiness, where newer versions of models need to be compared against older in-productions versions;
- Performance against extreme cases, where a model's performance for corner cases has to be assessed;
- Data quality issues, where there is a need to ensure that production data is processed in the same way as training data was (Chapter 6.5).

4.4.1 Data and feature drifts

Unless constantly retrained and updated, models cannot automatically adjust to changing input data, therefore detecting data and feature drifts is vital. Such drifts refer to a meaningful change in distribution between the training data and production data, or their associated features. Oftentimes, the changes that degrade a model's performance the most are changes made to the most informative features that the model uses to make predictions.

Detecting such changes can be done in the simplest way by monitoring their statistical feature values over time, for instance by investigating standard deviation, average, frequency and so on. Where continuous features are involved, divergence and distance tests such as Kullback-Leibler divergence or Kolmogorov-Smirnov [82] statistics are widely used. For categorical features, chi-squared tests, entropy or the cardinality or frequency can provide good indications of drift. In case the number of features is very large, it is preferable to first use dimensionality reduction techniques like PCA, LDA, t-SNE and others to prune them and then perform the necessary statistical tests.

Following a data drift detection, an alert can be triggered and based on how large the distribution change is, it is either possible to trigger a retraining job or build another model entirely with the new data. In the latter case, it is common that the new data will not be large enough to warrant a remodeling. It is preferable then to prepare the new data with historical (training) data and during retraining, assign higher weights to the features that drifted significantly.

4.4.2 Model drifts

Model drift occurs when the relationship between features and labels no longer holds because the learned relationships have changed over time. As a result, a model consistently returns unreliable and less accurate results over time compared to benchmarks or business metrics. It can happen in different ways:

- Instantaneous model drift – when there is a sudden drop in model performance over time and it is generally caused by data quality issues or the model being deployed in an entirely new domain for which it was not previously trained;
- Gradual model drift – occurs as a result of the natural consequences of a dynamic, changing and evolving use case, such as newly introduced features that skew the underlying pattern in the data;
- Recurring model drift – occurs as a result of seasonal events that are periodic and recurring over a fixed period a time;
- Temporary model drift – difficult to detect with rule-based methods but detectable with unsupervised methods, this type of drift can occur due to one-off events, such as adversarial attacks.

Detecting model drift can be done by using similar statistical tests as in the case of data drifts. By setting a predictive metrics threshold, a user can analyze whether a model consistently underperforms. In addition, monitoring data shift is a good indication whether it is necessary to analyze a model for degradations or drifts.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Following a model drift detection, a machine learning system can be scheduled to execute retraining od the model at predefined intervals. This is especially useful when the changes in a use case are frequent. Another option is to use online learning algorithms to improve the model as new data becomes gradually available.

4.4.3 Model readiness

When multiple versions of a model or multiple models tackling the same problem exist, it is possible for a new version or model to significantly improve average performance and yet introduce errors that the old model did not make. Those errors can be rare yet so detrimental as to nullify the benefit of the improved model. In some cases, post-processing pipelines built on top of a model can break. In other cases, users are so accustomed to the behavior of the old system that any introduced error contributes to a perceived “regression” in performance.

Requirements for in-the-wild model readiness go beyond accuracy, and include explainability, fairness (Chapter 8), as well as compatibility and regression minimization. One approach recently proposed by Amazon is to force the data representations computed by the new model to exist in the same space as the representations of the old model [83]. Another method is to perform positive-congruent training (PC training), which aims to train a new classifier without introducing harmful errors relative to the old model [84]. PC training is not just forcing the new model to mimic the old model, a process known as model distillation [85]. Model distillation mimics the old model, including its errors. PC training aims to mimic the old model only in the case of right predictions.

Another version of the incompatibility problem arises when a model is to be deployed on different devices with different resource constraints. A typical example is to have a large and powerful model running on a GPU cluster and smaller versions of it running on edge devices. To ensure compatibility, it is not enough for the smaller models to approximate the accuracy of the large model. In fact, they also need to approximate its architecture. A recent breakthrough has shown how to enforce this type of compatibility across platforms [86].

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 958171.

5 Trustworthy adoption of AI / ML models

5.1 Introduction

Advancements in the fields of AI and machine learning over the past few years have been nothing short of amazing, to the point that machine learning models have been gradually integrated in processing, analysis and decision making systems to assist or augment human abilities or even act fully autonomously, sometimes with far reaching and unanticipated consequences.

As these technologies become more pervasive in our lives, journalists, activists, and academics are starting to uncover problematic aspects stemming from the use of certain AI methods. Issues for instance related to racial and societal bias have been highlighted in relation to the use of data-driven and algorithmic decision making in the case of pretrial detention decisions [87] and predictive policing [88]. Imbalance in medical imaging datasets for computer-aided diagnosis was shown to produce biased diagnosis across genders [89]. And these are only few of the fairness, bias, accountability and interpretability issues tied to the unexamined deployment of AI systems that have been documented in the literature (see e.g., [90]).

It is important to point out that while these issues are critically important because of the broad ethical and societal consequences that they entail, in the context of this document they are also crucial because they imply, and often are originated by, technical and engineering factors that have the potential of also affecting strictly technical applications of AI and data-driven decision making. The perspective of this chapter will therefore be rather technical and in the extend that it will examine the ethical and socio-technical questions related to the deployment of machine learning and AI systems, it will do so both with the goal of cautioning practitioners and technical users on the broad social implications, as well as in the perspective of pointing out mechanisms that might represent risks originating from the use of these technologies even in a strictly technical domain.

5.2 Guidelines for trustworthy AI

The challenges inherent to the socio-technical and societal implications of AI adoption examined and reported by academics, journalists and activists alluded to in the previous section have been increasingly brought under the scrutiny of legislators. On April 8, 2019, following more than 500 public consultations [91] the European Commission published a report on Ethics guidelines for trustworthy AI, whose intended scope is to put forth key requirements to establish trust in human-centric Artificial Intelligence [92]. The report was compiled by an independent High-Level Expert Group composed of 52 experts with relevant expertise from academia, civil society and industry.
The guidelines indicate that a trustworthy AI system should be:

1. **lawful,** i.e., respecting all applicable laws and regulations
2. **ethical,** i.e., respecting ethical principles and values
3. **robust,** i.e., both from a technical perspective while considering its social environment.

More specifically, the guidelines put forward a set of 7 key requirements that AI systems should meet to be deemed trustworthy. A specific assessment list aims to help verify the application of each of the key requirements is shown in Figure 5.1.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

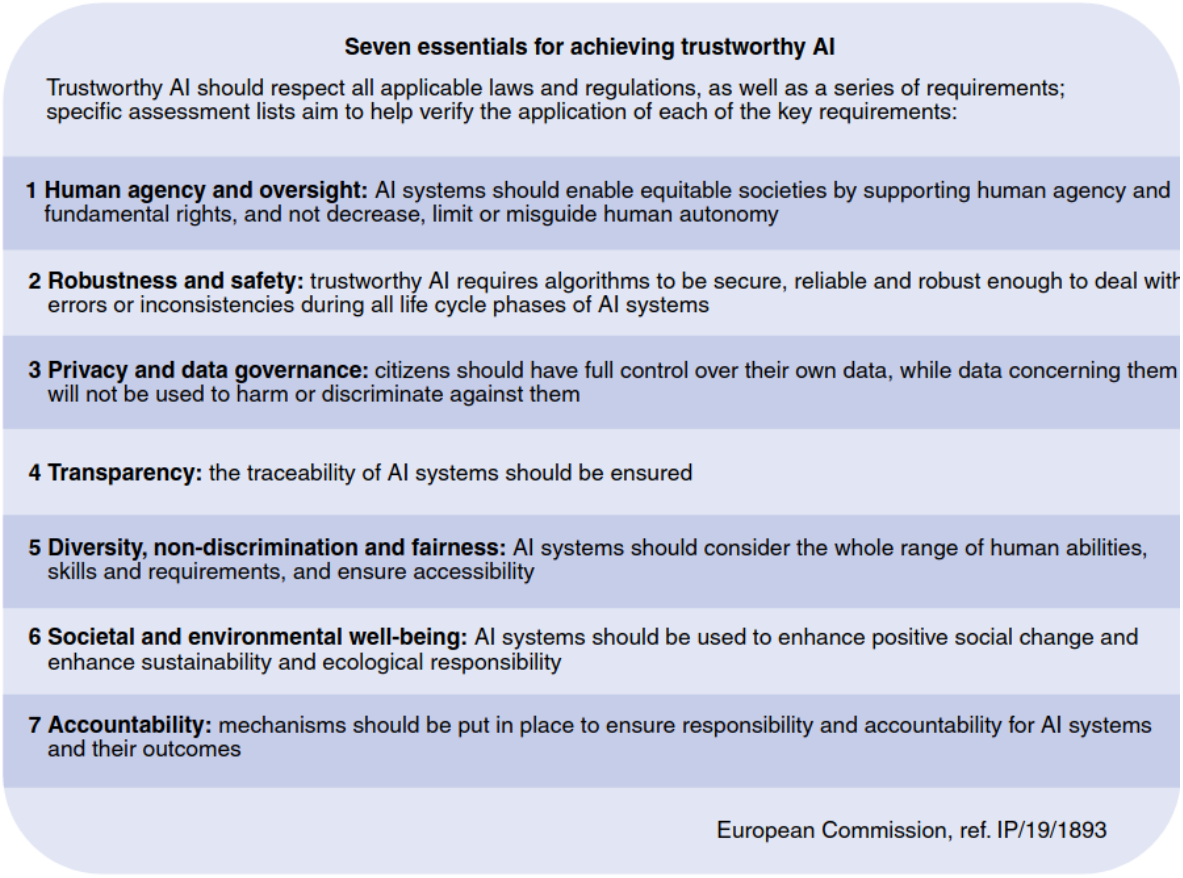

Figure 5.1 - The 7 ethical principles grounding the EU "Ethics guidelines for trustworthy AI" (from [91])

It is important to point out that while these guidelines are meant to inform legislation, they are not being legally enforced, but are part of a broad self-regulatory strategy.

Beside the fact that it might be important to ensure compatibility of an institution's AI strategy with the EU Commission guidelines as they will be the basis for future regulation, these guidelines also server as a useful blueprint pointing out the salient and potentially problematic areas at the intersection of AI and society.

Among the 7 ethical principles of the EU Commission trustworthy AI guidelines, in the previous chapter we already covered aspects that have to do with the key requirement 2, robustness and safety of deployed machine learning models. In the rest of this chapter we will cover issues that touch upon some of the remaining key requirements, specifically as they pertain to training and deploying trustworthy machine learning models. This assumes that anything that has to do with possible challenges at the level of the data (problem specification and evaluation metrics, data understanding, and data preparation) has been addressed (as done in the previous chapter of this document), and we are ready to tackle the next step in the machine learning lifecycle [93], which is *modeling*.

5.3 Trustworthiness machine learning modeling

A useful mental model for a trustworthy machine learning modeling workflow includes the following three parts [94] and is depicted in Figure 5.2:

1. pre-processing the training data

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

2. training the model with a machine learning algorithm
3. post-processing the model's output predictions.

The trustworthiness issues that can arise in the *pre-processing step* have largely been covered in the previous chapter.

In this section we will address the issues that can arise during the *model training* and *post-processing phase:* lack of explainability, unfairness and adversarial attacks.

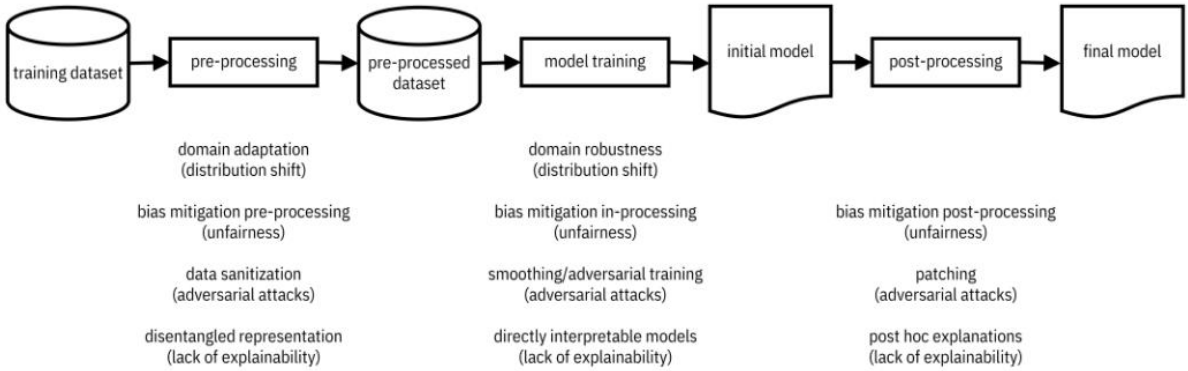

Figure 5.2 - Main parts of trustworthy machine learning modeling (from [94]).

5.3.1 Explainability and Interpretability

Explainability and interpretability of machine learning models is the aim to let people understand how machines makes their predictions. This can be a surprisingly hard tasks, since many machine learning models generate their results from the data they are fed after a sequence of possibly complex non-linear operations which renders the relation between inputs and outputs difficult to grasp, let alone explain.

This potential difficulty in explaining the mechanism behind the decision of more or less opaque machine learning models motivates to draw a clear distinction between the differential use of the term *interpretability* and *explainability.*

- **Interpretability** refers to models whose predictions can be readily inspected and understood by a domain expert. These models are also referred to *"white box" models* to metaphorically indicate that their operations are "transparent" to the practitioner and their outputs can be readily understood in relation the corresponding inputs.
- **Explainability** on the other hand refers to models that are in and of themselves too complex to be readily explained, as the process by which outputs are related to inputs is unintelligible to the point of being effectively completely opaque. These models are also referred to as *"black box" models*, in order to evoke their opaqueness.

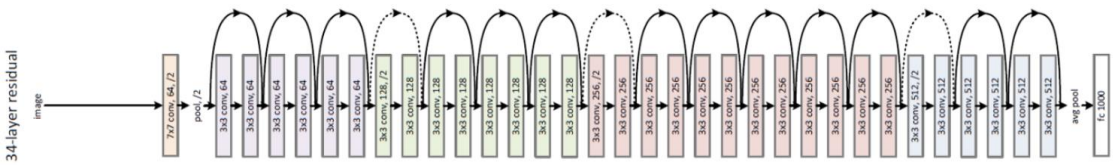

Figure 5.3 - An example of black box model (ResNet). inputs are related to the output by a complex sequence of non-linear operation which makes the mapping unexplainable (from [96]).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

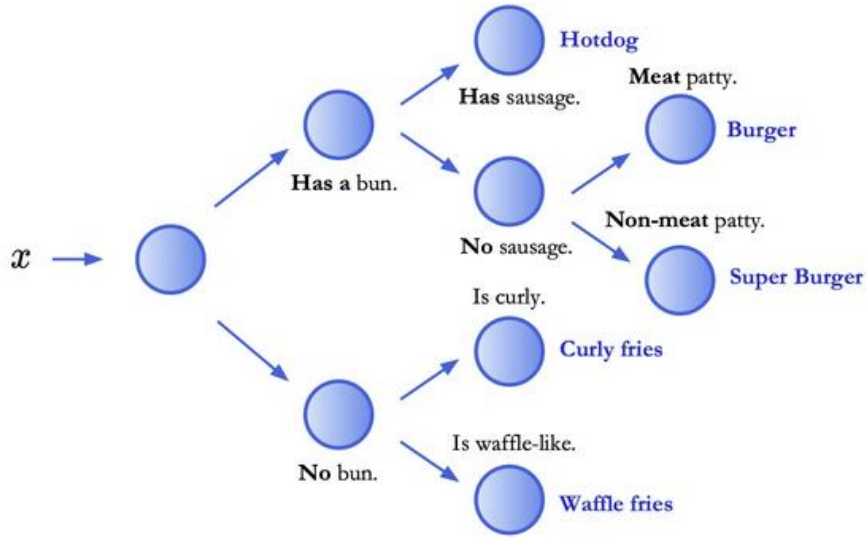

Figure 5.4 - An example of interpretable model. A decision tree is an example of interpretable model: each operation constituting an intermediate decision as to how an input should be classified can be verified or challenged separately (adapted from [95]).

There is a pervasive idea in the field that white box interpretable model, due to their inherent simplicity, are supposedly weaker machine learning modes bound to exhibit lower accuracy. This view is illustrated by a picture shown in the XAI DARPA BAA of 2016 on Explainable Artificial Intelligence (XAI) [97], which depicts a supposed trade-off between interpretability and accuracy (see Figure 5.5).

This view is now being called into question on multiple fronts (see e.g., [98]) for being on one hand excessively simplistic, in part based on the fact that interpretable models have been shown to rival black box models in several domains of interest, but also because generating explanations of black box models without proper input and supervision from domain experts is a practice that risks perpetuating the bad practices and causing the same harms to society that the explainability subfield of trustworthy AI is trying to avoid [98].

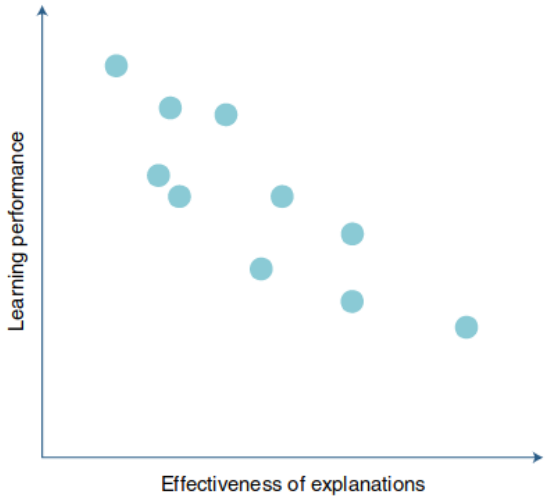

Figure 5.5 - Depiction of the accuracy-interpretability trade-off (from (12), adapted from [97])

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

With these cautionary points in mind, in the next sections of this chapter we will examine methods in the field that have been developed to produce explanations for black box machine learning models. These methods also collectively go under the category of *post-hoc explainability*, since the idea is that they provide *post-hoc* explanations of already existing models and decisions, as opposed to white box models, which are design with explanations already built into their architectures.

5.3.2 Post-hoc explainability

A common difficulty arising when trying to explain machine learning models stems from a fundamentally epistemic hard question, which is, what constitutes a good explanation? Just as people can justify and explain the same event in a myriad of different ways, the functioning of a machine learning model can be explained by many different types of explanations. Different explanations might be better suited to particular goals, domains and consumers of the explanations, or in other words, one explanation does not fit all [99].

There are three dichotomies that delineate the methods and techniques for machine learning explainability depending on the goals [94]:

- **local vs. global explanations**: is the user interested in understanding the machine predictions for individual input data points or in understanding the model overall;
- **exact vs. approximate explanations**: should the explanation be completely faithful to the underlying model or is some level of approximation allowable;
- **feature-based vs. sample-based explanations**: is the explanation given as a statement about the features or is it given by pointing to other data points in their entirety. Feature-based explanations require that the underlying features be meaningful and understandable by the user.

Each one of the choices at these dichotomies can be combined with the other ones, giving a total of 8 combinations corresponding to as many explanation types. Each one of these types typically suit a different kind of *persona* in the process of evaluating the machine learning model based on the provided explanations. Figure 5.6 shows for each type of explanation a type of persona for which it would be interesting, along with some example methods developed in the field that implement that type of explanation. For details on how the mentioned methods are implemented, please refer to reference [94].

Dichotomy 1	Dichotomy 2	Dichotomy 3	Persona	Example Method
local	exact	feature-based	affected user	contrastive explanations method
local	exact	sample-based	regulator	k-nearest neighbor
local	approximate	feature-based	decision maker	LIME, SHAP, saliency map
local	approximate	sample-based	decision maker	prototype
global	exact	feature-based	regulator	decision tree, Boolean rule set, logistic regression, GAM, GLRM
global	exact	sample-based	regulator	deletion diagnostics
global	approximate	feature-based	decision maker	distillation, SRatio, partial dependence plot
global	approximate	sample-based	regulator and decision maker	influence function

Figure 5.6 - The three dichotomies of explanations and their mapping to personas (from [94])

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Despite being a subfield and occasionally controversial subfield of AI (see e.g. [98]), explainable AI is already a vast and thriving ecosystem of methods, algorithms and best practices. It would be pointless to try to be exhaustive here, particularly because the community has started to organize the core material, including code in publicly accessible repositories. One such repository if the IBM Research AI Explainability 360 (AIX360) Toolkit, which can be freely utilized and deployed in ones infrastructure under the Apache-2.0 license (see Figure 5.7).

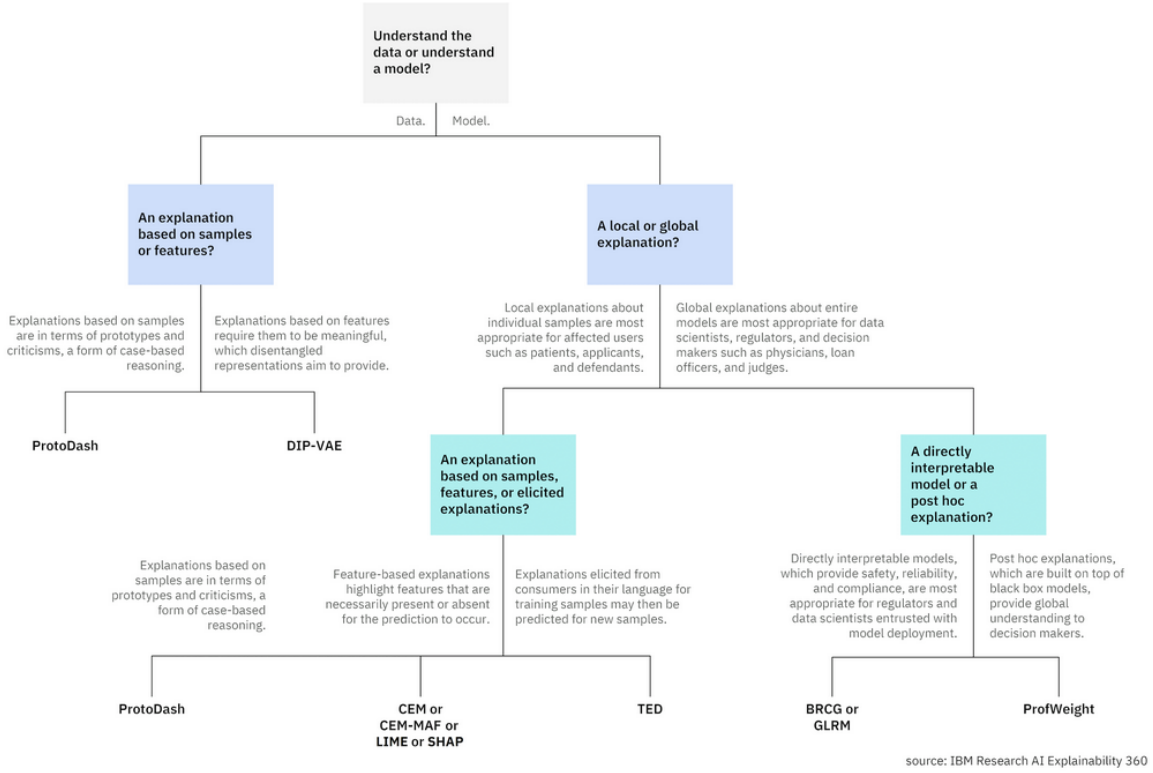

Figure 5.7 - Some of the methods implemented in the IBM Research AI Explainability 360 (AIX360) toolkit publicly available at the URL http://aix360.mybluemix.net (from [99]).

5.3.3 Concept-based interpretability

Recently, post-hoc explainability has been criticized for operating on low-level features such as pixel values, or sensory signals that are combined in unintelligible ways and do not correspond to high-level concepts that humans easily understand, making their explanations correspondingly brittle [100–103].

One way to overcome these limitations that has been proposed is to design machine learning models that generate decisions based on human-understandable categories (i.e., concepts) grounded in domain expertise rather than raw features [100,101,104–111].

For example, to identify a bird species, a model should focus on morphologically meaningful concepts, such as the shape, size and colors of beak, feathers or wings, rather than focusing on raw pixels, and combine them in ways that a domain expert (in this case an ornithologist) would reckon as intelligible to produce a classification.

It was recently demonstrated that this type of concept-based attribution can be achieved together other two properties that are of paramount importance for good explanations, which are *plausibility* and *faithfulness:*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

- **Plausibility**: an explanation is defined as *plausible* if it is convincing to a domain expert using the machine learning model. This means that the explanation is grounded in domain-relevant entities, and is compatible with the mechanisms and phenomena that are known to take place in the domain;
- **Faithfulness**: an explanation is defined as *faithful to a decision* if it truly reflects the reasoning process behind the decision. This means that the same explanation will always need to correspond to the same decision, and conversely different decisions cannot correspond to the same explanations.

The ConceptTransformer model [112] is a model that achieves all of this: it provides concept-based explanations that are plausible by construction and faithful by design. It does this by training a transformer-based model to reproduce the concept explanations of the training dataset and by constraining its structure so that the explanations that it provides mathematically satisfy a formal definition of faithfulness (see Figure 5.8).

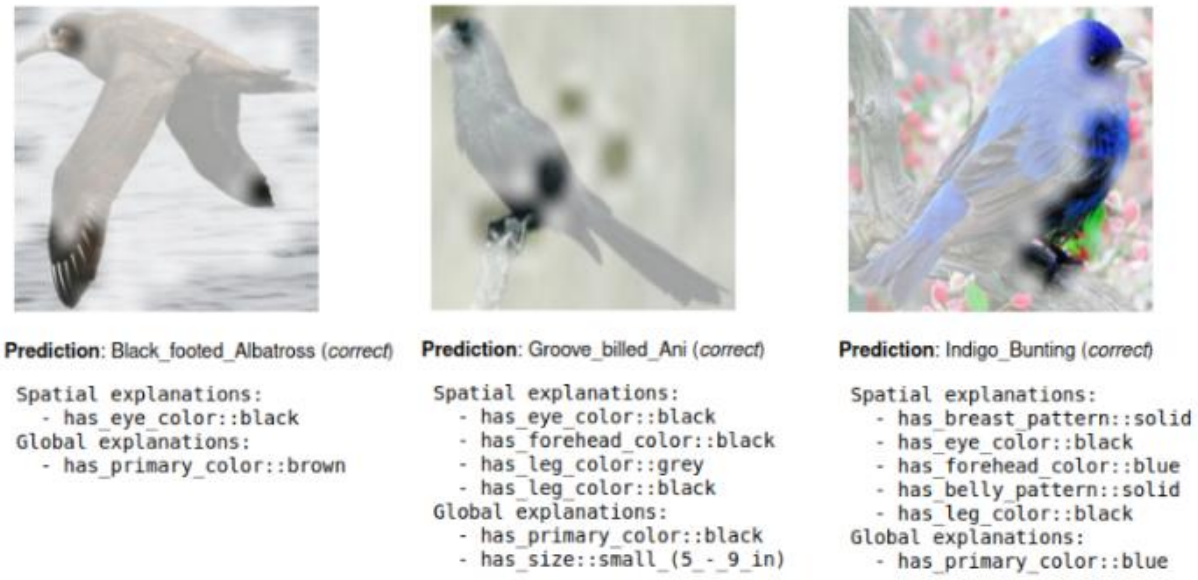

Figure 5.8 - Attention scores provided by the ConceptTransformer model while predicting the species of the birds represented in some input pictures. The attention focuses on relevant spatial location and concept in ways that are guaranteed to be faithful, therefore providing robust explanations (adapted from [112]).

5.3.4 Fairness

The main objective of fairness in AI systems is to eliminate or mitigate the effect of *bias*. Bias often manifests in the form of unfair treatment of different groups of people based on some protected or *sensitive* attributes (e.g., gender, race, ethnicity, etc.).

There are two main types of fairness that are of main concern: 1) group fairness and 2) individual fairness.

- **Group fairness** is the idea that the average behavior of a machine learning model like a classifier should be the same across groups defined by protected attributes. For instance, the probability that an algorithmic decision making system approves a loan should be the same irrespective of the gender of the applicant.
- **Individual fairness** is the idea that individuals similar in their features should receive similar model predictions. Individual fairness includes the special case of two individuals

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

who are exactly the same in every respect except for the value of one protected attribute (this special case is known as counterfactual fairness).

Notice that the conceptual and methodological machinery developed by the fairness in AI community can also be directly reapplied to quantify and counter other forms of bias in machine learning, that do not necessarily have to do with societal injustice. For instance, if instead of focusing on protected attribute one focuses on important structural attributes in the context of civil engineering, then methods that mitigate lack of group fairness can be used to mitigate that the performance of a machine learning model might be different across values of the selected structural attribute, as for instance in the case where a defect detection algorithms has an accuracy that is high for some defects, but excessively low for other defects. This type of scenario can be addressed with the same methods that identify and solve group fairness.

Despite being a relatively new concern in AI, the fairness, accountability, transparency and ethics community in machine learning is a thriving community that has already produced a vast remarkable ecosystem of metrics, methods and algorithms, in parts also to its reliance on inputs from the wider academic and technical traditions concerned with the legal, ethical and socio-technical impact of technology.

For more details oriented to practical applications of fairness in AI we refer the reader to two high-quality software repository that the machine learning community has made freely available.

The first one is the **IBM AI Fairness 360 (AIF360) toolkit** [113] which is available at the url: http://aif360.mybluemix.net, and which provides a comprehensive set of fairness metrics for datasets and machine learning models, explanations for these metrics, and algorithms to mitigate bias in datasets and models.

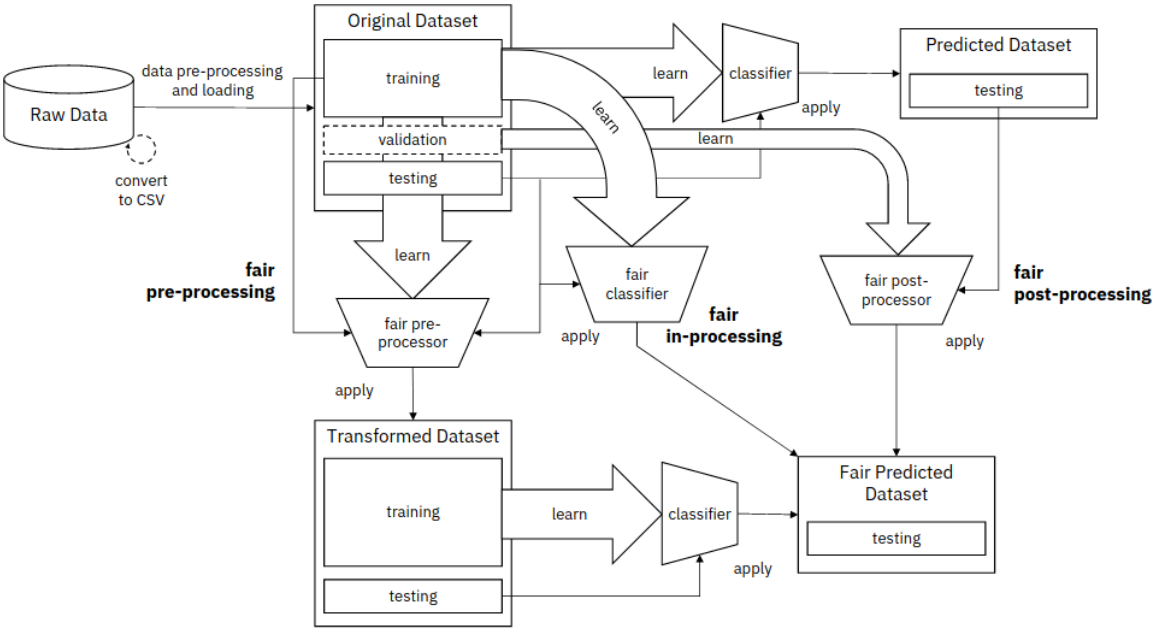

Figure 5.9 - The fairness pipeline of the IBM AIF360 toolkit (from [113])

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

A second very valuable publicly available online resource is the **Microsoft Responsible AI Toolbox** [114] available at the url http:// responsibleaitoolbox.ai, which is a suite of tools provides a collection of model and data exploration and assessment user interfaces that enable a better understanding of AI systems. These interfaces are meant to empower developers and stakeholders of AI systems to develop and monitor AI more responsibly, and take better data-driven actions.

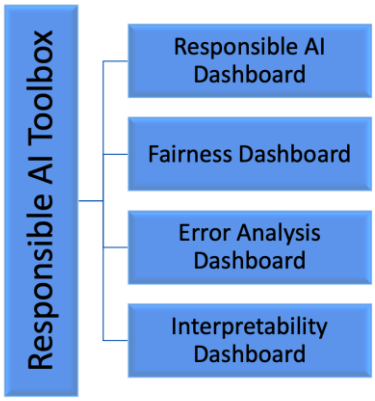

Figure 5.10 - The dashboard that composes the Microsoft Responsible AI Toolbox (from [114])

5.3.5 Guidelines for Human-AI interactions

The topic of Human-AI interactions has several contact points with the fields of interpretability, explainabilty and fairness, but is generally more specifically concerned with the design of end-products that incorporate AI algorithms and maximizing their potential of the end-user. This is in line with a long tradition of the principle of human-AI interaction that have been discussed in the *human-computer interaction community* for at least two decades now.

Central concepts in Human-AI interactions are the concepts of **uncertainty quantification** (specifically how to quantify and communicate the uncertainty that the AI system might have about its internal representations and the outputs that it provides [115,116]), **intelligibility** (which is a human-centered concept that relates to interpretability [117]), and **transparency** (the idea of disclosing information about the whole lifecycle of a system).

A good starting point on the topic is reference [118] which provides a set of very general guidelines for human-AI interaction, which the authors validated through multiple rounds of evaluation and user studies (see Figure 5.11).

It will come with no surprise that the topic of Human-AI interaction is currently in fervent development, particularly due to the recent acceleration of the trend of infusing consumer products and software with AI capabilities. Just as a way of further emphasizing the importance of the topic it is worth mentioning that the recent US National Security Commission report on Artificial Intelligence (NSCAI) has dedicated a whole chapter on the need of “Establishing justified confidence in AI systems” and human-AI interaction and teaming in its report meant to provide recommendations to the President and Congress to “advance the development of artificial intelligence, machine learning, and associated technologies to comprehensively address the national security and defense needs of the United States” [119].

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 958171.

		AI Design Guidelines	Example Applications of Guidelines
Initially	G1	**Make clear what the system can do.** Help the user understand what the AI system is capable of doing.	[Activity Trackers, Product #1] "Displays all the metrics that it tracks and explains how. Metrics include movement metrics such as steps, distance traveled, length of time exercised, and all-day calorie burn, for a day."
	G2	**Make clear how well the system can do what it can do.** Help the user understand how often the AI system may make mistakes.	[Music Recommenders, Product #1] "A little bit of hedging language: 'we think you'll like'."
During interaction	G3	**Time services based on context.** Time when to act or interrupt based on the user's current task and environment.	[Navigation, Product #1] "In my experience using the app, it seems to provide timely route guidance. Because the map updates regularly with your actual location, the guidance is timely."
	G4	**Show contextually relevant information.** Display information relevant to the user's current task and environment.	[Web Search, Product #2] "Searching a movie title returns show times in near my location for today's date"
	G5	**Match relevant social norms.** Ensure the experience is delivered in a way that users would expect, given their social and cultural context.	[Voice Assistants, Product #1] "[The assistant] uses a semi-formal voice to talk to you - spells out "okay" and asks further questions."
	G6	**Mitigate social biases.** Ensure the AI system's language and behaviors do not reinforce undesirable and unfair stereotypes and biases.	[Autocomplete, Product #2] "The autocomplete feature clearly suggests both genders [him, her] without any bias while suggesting the text to complete."
When wrong	G7	**Support efficient invocation.** Make it easy to invoke or request the AI system's services when needed.	[Voice Assistants, Product #1] "I can say [wake command] to initiate."
	G8	**Support efficient dismissal.** Make it easy to dismiss or ignore undesired AI system services.	[E-commerce, Product #2] "Feature is unobtrusive, below the fold, and easy to scroll past...Easy to ignore."
	G9	**Support efficient correction.** Make it easy to edit, refine, or recover when the AI system is wrong.	[Voice Assistants, Product #2] "Once my request for a reminder was processed I saw the ability to edit my reminder in the UI that was displayed. Small text underneath stated 'Tap to Edit' with a chevron indicating something would happen if I selected this text."
	G10	**Scope services when in doubt.** Engage in disambiguation or gracefully degrade the AI system's services when uncertain about a user's goals.	[Autocomplete, Product #1] "It usually provides 3-4 suggestions instead of directly auto completing it for you"
	G11	**Make clear why the system did what it did.** Enable the user to access an explanation of why the AI system behaved as it did.	[Navigation, Product #2] "The route chosen by the app was made based on the Fastest Route, which is shown in the subtext."
Over time	G12	**Remember recent interactions.** Maintain short term memory and allow the user to make efficient references to that memory.	[Web Search, Product #1] "[The search engine] remembers the context of certain queries, with certain phrasing, so that it can continue the thread of the search (e.g., 'who is he married to' after a search that surfaces Benjamin Bratt)"
	G13	**Learn from user behavior.** Personalize the user's experience by learning from their actions over time.	[Music Recommenders, Product #2] "I think this is applied because every action to add a song to the list triggers new recommendations."
	G14	**Update and adapt cautiously.** Limit disruptive changes when updating and adapting the AI system's behaviors.	[Music Recommenders, Product #2] "Once we select a song they update the immediate song list below but keeps the above one constant."
	G15	**Encourage granular feedback.** Enable the user to provide feedback indicating their preferences during regular interaction with the AI system.	[Email, Product #1] "The user can directly mark something as important, when the AI hadn't marked it as that previously."
	G16	**Convey the consequences of user actions.** Immediately update or convey how user actions will impact future behaviors of the AI system.	[Social Networks, Product #2] "[The product] communicates that hiding an Ad will adjust the relevance of future ads."
	G17	**Provide global controls.** Allow the user to globally customize what the AI system monitors and how it behaves.	[Photo Organizers, Product #1] "[The product] allows users to turn on your location history so the AI can group photos by where you have been."
	G18	**Notify users about changes.** Inform the user when the AI system adds or updates its capabilities.	[Navigation, Product #2] "[The product] does provide small in-app teaching callouts for important new features. New features that require my explicit attention are pop-ups."

Figure 5.11 - The 18 human-AI interaction design guidelines proposed in [118], categorized by when they likely are to be applied during interaction with users, along with illustrative application

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

6 Case study: Defect detection and segmentation for automated visual inspection

6.1 Use case description

Numerous civil engineering structures have been constructed over the last century and are critical for transportation, such as airport tarmacs, roads and bridges. For example, bridges are exposed to different kinds of external loads, including traffic and hurricanes, during their life cycle. These external loads cause structural damage to bridges, which may lead to their collapses. In fact, more than 40% of the bridges, roads and tunnels around the world have already surpassed their expected life expectancy. To ensure their safety and serviceability, it is essential to inspect the physical and functional condition of the structure on a regular basis. Consequently, civil engineering structure maintenance has become an important topic of research. Currently, most inspections around the world are conducted as visual inspections, which are the most effective nondestructive methods to assess the physical and functional conditions of a structure (Figure 6.1).

Figure 6.1 - Typical bridge components that are visually inspected on a regular basis.

However, every visual inspection depends on the inspector's subjective evaluation, which could lead to reliability and accuracy issues for the inspection results. In addition to the reliability and accuracy issues, the visual inspection method is problematic in terms of inspector safety, work efficiency, and cost. Typically, experts perform a manual inspection from the ground up by using lifts, ropes and platforms, which can be quite dangerous (Figure 6.2). Additional challenges are imposed by weather conditions, traffic, high costs and the fact that any findings are not easy to document afterwards. As a result, just for the manual inspection of bridge structures alone, more than $50 billion and 2 billion man-hours have been invested.

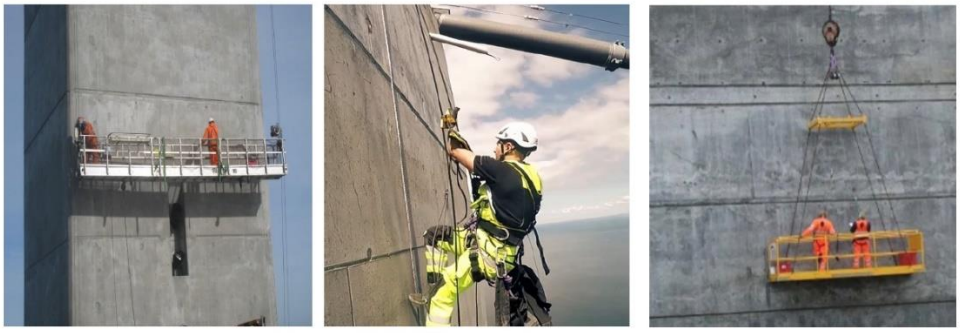

Figure 6.2 - Typical manual visual inspection of a bridge.

The advances in drone technology and its falling costs have recently pushed this laborious process of manual inspection progressively towards automation. Flying drones around a structure and using embedded high-resolution cameras to collect visual data from all angles not

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

only speeds up the inspection process, but it also removes the human from potentially dangerous situations. In addition, thanks to the power of artificial intelligence capabilities, defects can be detected and localized with high precision automatically and presented to the reliability engineer for further analysis.

To address these problems, drone technologies have been recently used to conduct the visual inspection of civil engineering structures. A drone-based inspection system can take high resolution images from all positions around the structure and constant distances, under optimal lighting and weather conditions. These collected images are used by advanced machine learning models to develop a full-fledge automated visual inspection pipeline (Figure 6.3) that provides the following core capabilities:

- high-quality data acquisition;
- detect, segment and assess the severity of defects in the structure;
- measure the size of defects with mm-level precision;
- reconstruct the overall structure without additional CAD models by using instead advanced photo stitching algorithms.

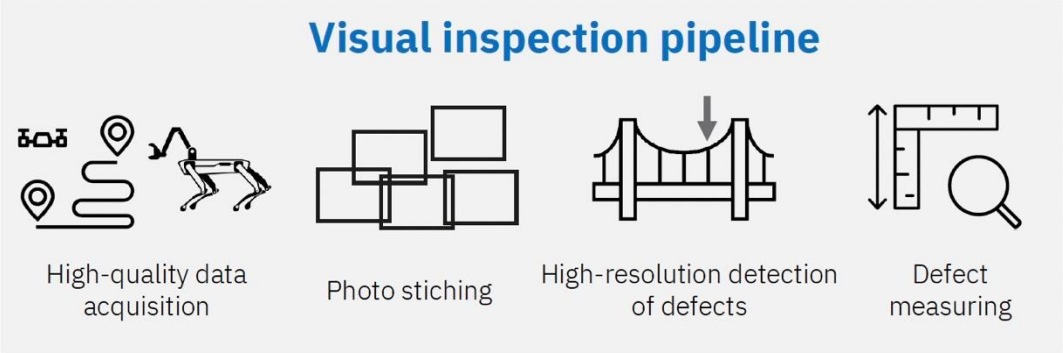

Figure 6.3 - Automated visual inspection pipeline.

6.2 Data and Machine Learning preliminaries

The first requirement in to understand what image capturing strategies for inspection can be used such that high-quality and high-resolution images are collected for further use. While various strategies exist, two particular strategies were used successfully. The standard scan approach consisted of a drone flying over an area and taking pictures in a regular grid. A more sophisticated approach, called high-resolution grid approach, was conducted in such a way that a drone would hover at a constant position but turns its head in various directions to capture multiple images aligned in a grid fashion.

Additionally, the drone flights were executed in order to meet additional requirements:

- Good lightning conditions, therefore performing inspection on days with good weather conditions was important;
- Particular care was taken not to capture images of any irrelevant structures, such as cars, water, ships and so on, since these elements have the potential of confusing the machine learning model;
- Support for cylinder inspections (e.g., cell towers, round piers) was provided by developing three patterns: 1) horizontal circles (images are taken on the move, time efficient), 2) vertical lines (images are taken at pre-calculated points while hovering), 3) horizontal polygons (horizontal lines in a polygon shape around the cylinder);
- Support for grid inspection which allowed the automatic creation of grid inspection patterns by dividing the area into recursive grids.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

Particular to the Sund&Baelt [1] use case was using a Matrix JDI 200 RTK enterprise drone [2] equipped with a multi-sensor ZH20T system which took pictures using the high-resolution grid strategy. For each grid, this included a wide overview picture as well as a full grid of images taken with a tele-objective resulting in high resolution. Over 20000 single images arranged in over 300 such grids capturing the 23 pillars from all angles of the Storebælt East Suspension bridge [3] were collected. At 254 metres above sea level, the East Bridge has a length of 6790 metres and a free span of 1624 metres. Each image has size of 6K x 4K pixels, which amounted to over 20GB of data collected.

The second requirement was, based on the images collected, to understand what types of defects should be detected and segmented within the use case, at which granularity these defects need to be detected and to annotate representative examples of such defects accordingly. Together with civil engineering experts from Sacertis [4] and Sund&Baelt, we defined seven types of defects of interest, as shown in Figure 6.4:

- Cracks
- Cracks with precipitation
- Rust
- Spalling
- Algae
- Net cracks
- Spalling with corroded rebar

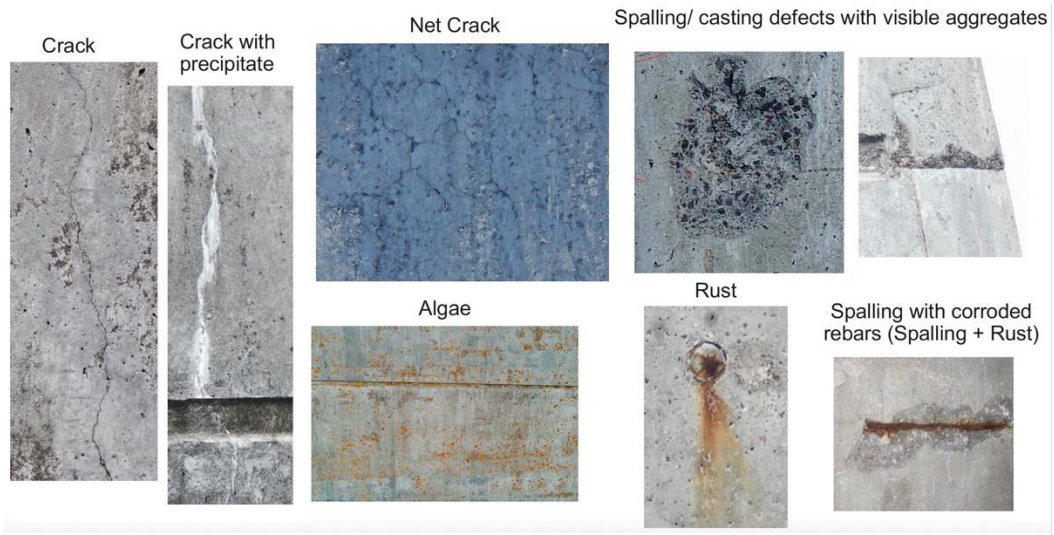

Figure 6.4 - Defects to be detected with the automated visual inspection pipeline.

Next, the use case requirement was to detect these defects with high precision of both localization (i.e., precise location of defect) and classification (i.e., accurate defect categorization). This indicated that the annotations of representative defects need to be done at both bounding box level and fine-grained instance segmentation masks. Especially, the latter is an extremely time consuming manual effort, as human annotators need to draw precise contours around defects that may be potentially very thin or small, such as the case for cracks. Since in every image there exists an arbitrary number of defects present and all defects should be annotated manually, out of the 20000 images collected, a high-quality set of 150 annotated images was developed together with civil engineering experts. This set was later used to assess the performance of the machine learning model in the evaluation stage.

Since detection and segmentation models are typically fully supervised, they require a large amount of training data. The 150 images annotated with domain experts were not enough for any detection and segmentation model to achieve high performance, therefore SuperAnnotate

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

[5] was engaged as an external party to annotate an additional 2500 high-resolution images, based on training and guidelines previously developed in collaboration with experts from Sacertis and Sund&Baelt.

6.3 Model development cycle

The pre-processing step is concerned with extracting a set of images captured via the drone system that are high quality and present a mixture of defects. After the images have been selected, various augmentations and transformations are performed, such as applying vision filters (e.g., blue, sharpness, contrast, etc.), cropping, downscaling, shifting, rescaling, and so on. In the typical case where defects are not equally distributed, oversampling and under-sampling techniques can be applied to provide the detection and segmentation model a more balanced dataset to learn from.

The model selection is primarily dictated by the need to both detect and segment multiple types of defects. With respect to segmentation, there are multiple types of segmentation tasks. Image segmentation involves partitioning images into multiple segments or objects and can be formulated as a classification problem of pixels with semantic label (semantic segmentation) or partitioning of individual objects (instance segmentation). *Semantic segmentation* performs pixel-level labelling with a set of object categories (e.g., person, animal, vehicle or crack, spalling, casting defect) for all pixels of the image, thus it is generally a more difficult task than image classification, which predicts a single label for the entire image. *Instance segmentation* extends semantic segmentation scope further by *detecting and delineating* each object of interest in the image (e.g., partitioning of individual defects). The specific requirement of the automated visual inspection use case was to perform instance segmentation, namely detect and segment every instance of a defect type within an image.

Major deep learning methods for instance segmentation include Fully Convolutional Networks (FCNs), encoder-decoder based models, multi-scale and pyramid network-based models, Regional Convolutional Neural Networks (RCNN) based models, etc. [6]. In FCNs, multiple convolutional layers are employed on the image directly as the feature extractors with the downsampling and upsampling process inside sliding windows, but its efficiency is very low. With R-CNN, the image is preprocessed and thousands of Region of Interests (RoIs) for feature extraction with FCNs are eventually produced. The R-CNN reduces the computational time compared to alternative approaches and improves the accuracy of segmentation. However, it still is computationally demanding. Fast R-CNN and Faster R-CNN are quite different from the conventional R-CNN. The former applies FCNs directly on the RoIs of the feature maps which comes after convolutional process on the original image, but, in Faster R-CNN, a network called Regional Proposal Network (RPN) on these maps is inserted to automatically produce the proposal, thus the speed and accuracy of prediction are improved [7]. However, none of them are applicable for instance segmentation. Mask-RCNN [8] has been proposed to predict the class, bounding box and instance segmentation mask of an object, which made it the best candidate for the machine learning model developed in this use case. Across all defects, the same Mask-RCN model was used with the only difference being in the amount of training data used for each defect and the hyperparameters of the model that are better suited depending on the type of defect to be detected.

The post-processing step is concerned with post-processing which involves the validation and retraining schedules of the machine learning models. Part of post-processing revolves around the standardization of the output format of each module. Arguably the most important aspect of post-processing revolves around extracting value from the model outputs. This value extraction depends on the approach used in the model. For example, in case the main objective is to provide fine-grained segmentation masks for the defects detected, a comparison in the intersection over union against the ground truth annotations is critical. In this case, a variety

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

of thresholds can be used, where the threshold represents the minimum amount of area overlap between the detected mask and the ground annotation. If, on the other hand, the goal is relatively simpler and focused on detecting the presence of a defect within a bounding box area, more relaxed metrics can be used, such as mean average precision.

Finally, the retraining schedule is part of post-processing. This is a crucial part that heavily influences the robustness of the models. This is because a model that is trained using a relatively long period of time might start degrading in performance, and so a period should be set for re-training. This implies that regular drone flights have to be conducted, such that updated images are provided to the model. It is essential that similar flight conditions, distance to the structure and image resolution are maintained.

6.4 Detection and segmentation model

The Mask -RCNN model used for this use case is an extension of Faster R-CNN. A Regional Proposal Network (RPN) is inserted on feature maps to automatically produce RoIs, then a small FCN is applied on each RoI to segment the instance of objects with masks when the classes and bounding boxes of these defects are predicted with the same pipeline as used in Faster R-CNN. In addition, different depth of ResNet and the Feature Pyramid Network (FPN) are combined to extract high-quality feature maps. The framework of Mask R-CNN is shown in Figure 6.5.

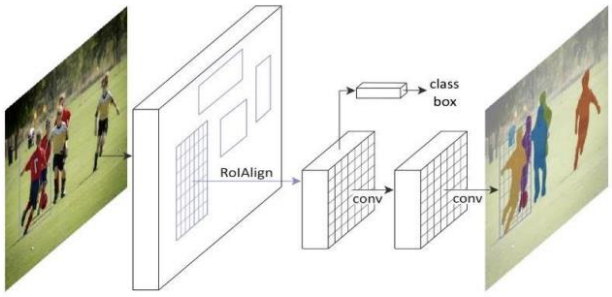

Figure 6.5 - The Mask-RCNN framework for instance segmentation (from [9]).

The Mask-RCNN model was implemented using the Detectron2 library [9], which provides state-of-the-art detection and segmentation algorithms. On top of the Mask-RCNN backbone, additional functionalities were implemented in order to boost the model's overall mean accuracy precision and intersection over union values:

- Tiling – patches are extracted from high resolution images in order to improve detection of smaller defects, all the while retaining the maximum amount of information rather than resizing them;
- Online augmentations – multiple types of augmentations available in albumentations [10], such as pixel-level and spatial-level transformations, including flipping, rotating, cropping, etc., were applied in online fashion to boost the model's generalization capabilities. In particular, spatial-level transformations are adopted to change input images, masks and bounding boxes simultaneously;
- Ensembling – train multiple models and ensemble their results at different confidence thresholds, depending on the defect type.

Figure 6.6 visually shows the improvement in defect detection with the augmented model compared to the baseline.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

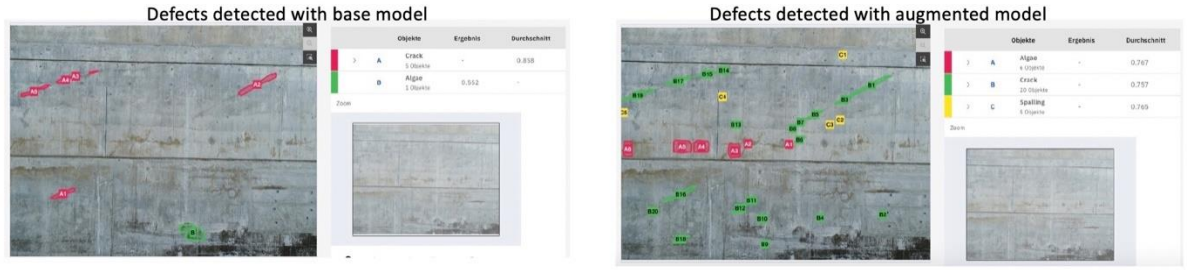

Figure 6.6 - Comparison in defect detection between baseline and augmented models.

The model is deployed in the IBM cloud and using services that support model training, inference and periodic re-training. Detected and segmented defects with the Mask-RCNN model are made available to customers, data scientists and domain experts through the One Click Learning (OCL) developed at IBM.

6.5 Defect detection and segmentation scenario

The OCL platform (Figure 6.7) takes a data-centric approach to focus the user experience around the management of users' assets and provides advanced capabilities for:

- powerful data exploration of large datasets and high resolution images with corresponding detected damage;
- detection, characterization and measurement of damages;
- reconstruction of infrastructure elements and localization of damages with automated image stitching;
- quick extraction of actionable results;
- empowering engineers and infrastructure managers to use pre-trained AI models for everyday inspections.

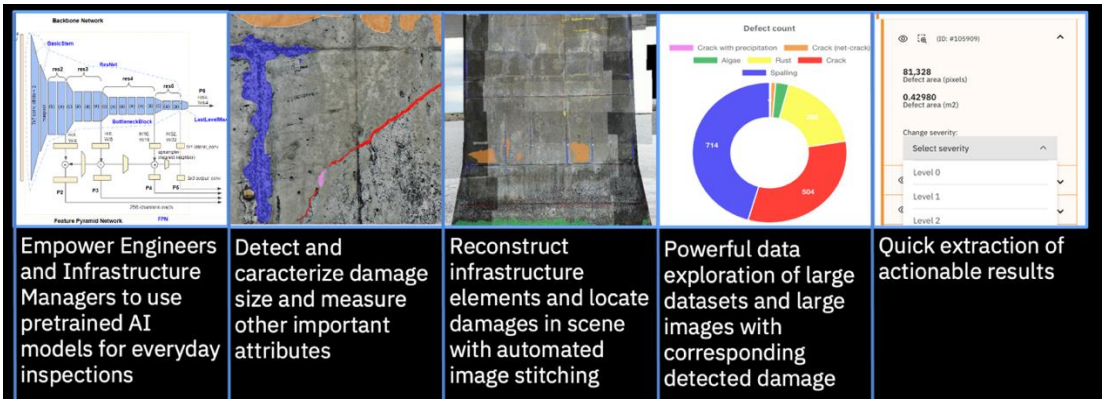

Figure 6.7 - One Click Learning (OCL) platform for automated visual inspection.

First, as seen in Figure 6.8, general statistics are presented to the engineer. On the left side, a progressive view of the bridge that dives into the pillars, their orientations and corresponding images, allow the user to understand the hierarchy of the structure and quickly locate data of interest. On the right side, a summarized view of the images collected during drone inspection and the defects detected and classified during the inference of the AI model is provided. The defects are classified into categories specific to the use case. Below the summary section, more detailed statistics are presented, such as average area (in m^2) per defect type and distribution of defects across the dataset.

Additionally, the platform provides filtering options to enable an intuitive and fast navigation through the defects. Specifically, one can select one or more defect categories and provide

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

ranges of interest for attributes like area (in pixels or m^2), confidence score as resulting from the model inference or severity rating. This will return the set of images satisfying this criteria. The user can navigate through the selected images and visualize all defects detected in a specific image, as seen in Figure 6.9. By hovering with the mouse over a defect, more details of interest are provided, such as type of defect, area in pixels and m^2, prediction score and severity level. While the functionality described so far applies per individual image, it already provides great benefits to reliability engineers. However, decisions around repair and maintenance take into consideration the locations of defects in a bridge structure as well. Powered by image stitching algorithms, OCL provides an overall view of each bridge pillar. These stitched images are reconstructed automatically from the raw high resolution photos taken by the drone, by using image rectification and location reconstruction algorithms, and combine all defects detected in the individual images that were attached together, as shown in Figure 6.10. It is always possible for the engineer to go from the overview image to the underlying raw photos for further detailed inspection (e.g., by clicking on the red highlighted area in Figure 6.10). Intuitive navigation through the raw photos is possible by means of a minimap (shown in Figure 6.11), which allows for an overview of where in the pillar structure each defect is located.

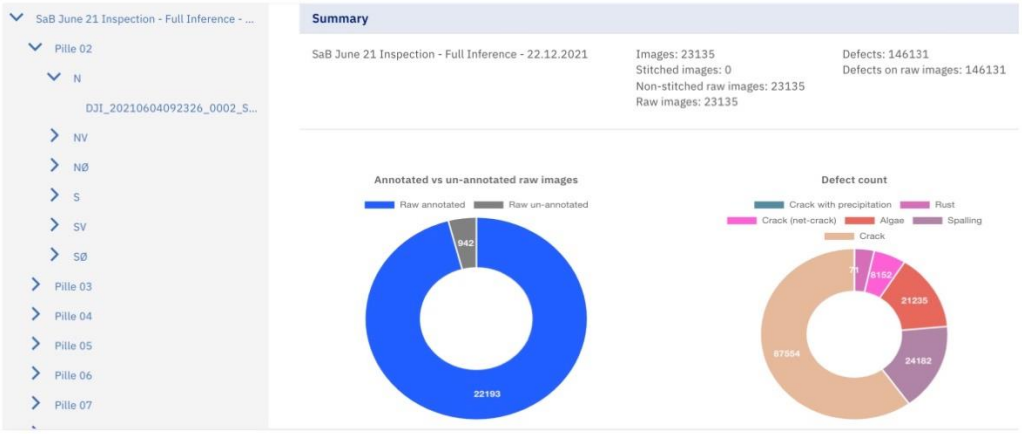

Figure 6.8 - Hierarchical view of assets (left); summary of dataset and associated defects, as detected and classified into multiple categories during AI-model inference.

Figure 6.9 -Visualization of defects in specific image with associated confidence score as computed by the AI-model.

These stitched images are extremely big in size, as they preserve the full resolution of the already detailed raw photos. In order to deliver a smooth navigation, in the civil infrastructure component in OCL, images are divided in multi-resolution tiles and streamed to the browser

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

from cloud object storage on demand. This is just one of the challenges that we had to overcome when dealing with such a big amount of data. Nevertheless, thanks to the flexible architectural design of the platform and the scalability of cloud-based solutions, we are able to deliver a smooth user experience in these complex use cases also.

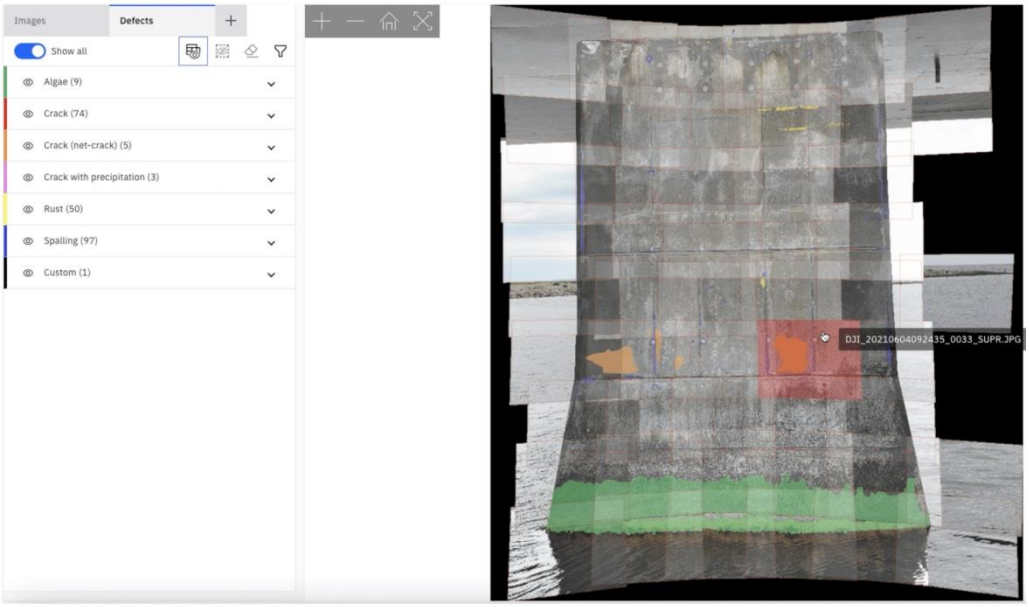

Figure 6.10 - Overall view of a bridge pillar, after image stitching. Summary of defects is provided on the left and different categories are distinguished by color in the stitched image.

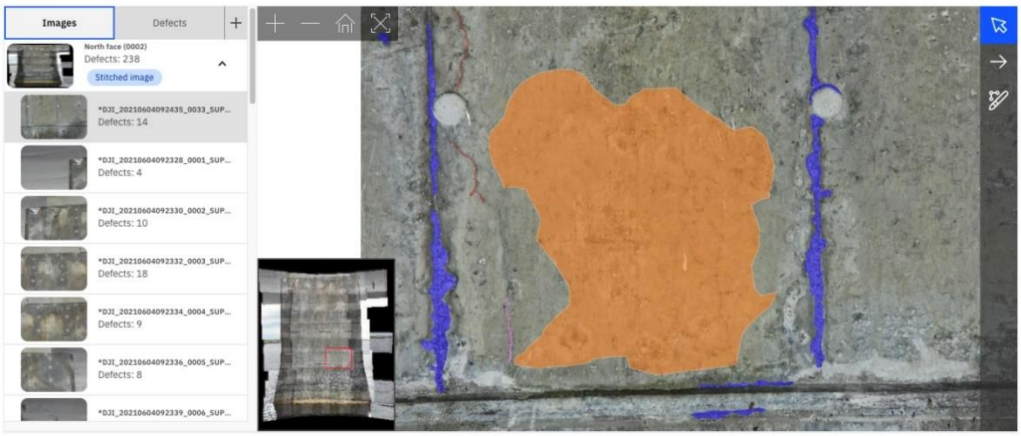

Figure 6.11 - Highlight on a defect; a minimap with the overview stitched image is showing the corresponding location of the defect on the full pillar.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171. © 2021

7 Case study: Continuous monitoring of event streams for wind turbines diagnosis and management

7.1 Use case description

Although in the last years there has been a substantial increase of the power capacity growth of Wind Energy across the EU covering around 11.4% of the EU electricity consumption in 2015, there is still a long way to go in order to achieve the target in 2030 of at least 27% for the share of renewable energy consumed in the EU1. In particular, emerging technologies as Offshore Wind Energy, demand new advanced O&M solutions/tools for improving significantly their return of investment (RoI) and their Levelised Cost of Energy (LCoE) indicator, as well as for performing the reliability and extended lifetime of WTs and farms over the years.

Despite the current O&M strategies to maximise the energy yield, there is a considerable need for the reduction of the O&M costs (one of the main contributors to the Renewable Energy Cost and Performance) to alleviate their impact in LCoE and improve cost competitiveness of offshore wind energy. This challenge can be overcome thanks to the implementation of Condition Monitoring Systems (CMS) and Diagnosis and Prognosis Models to optimise the operation, maximise lifetime and adjust O&M to real performance of the Wind Turbines (WTG and structure). Nevertheless, these technologies must be scaled up and tested under demanding real conditions with the main aim of analysing the improvements achieved in the whole life cycle of the main components of a Wind Turbine (WT) at Wind Farm (WF) scale. These needs have been already identified and prioritized by WINDE EUROPE3 in the near future.

The scenario developed around offshore wind energy is focused on developing a set of sophisticated CMS technologies, both for turbine and structure and advanced failure detection for diagnosis and prognosis integrated in an O&M decision support tool. The advanced CMS analytics have been integrated in an end-to-end system from sensor/models data generation to maintenance decision support. Such connectivity from the edge (sensors/models) via the cloud (integrator) to the enterprise O&M tools allows for a continuous learning expanding CMS models created in test bed environments with field data and observations. These novel concepts/solutions address advanced condition and risk-based approaches based on remote and non-intrusive maintenance, allowing the coordination and dispatching of offshore O&M services and logistics that can assist O&M managers, while the lifetime of the WTs is substantially improved. Advanced O&M tools enable load dependent strategies for individual WTs based on load or stress levels induced by wakes and waves. Ideal service slots and working schedules are calculated through the evaluation of data on each component, fittings, vessels and service staff, including a reliable analysis of the weather forecast. Therefore, the solutions developed in scenario contribute a significant reduction of the O&M costs thanks to both the decrease in operating costs (OPEX) due to the optimization of O&M logistics and a reduction of unavailability and decrease of material costs and number of inspections.

7.2 Data and Machine learning preliminaries

Multiple wind turbine farms that took part in this use case collected several types of data, that were later used in the model development cycles to provide diagnosis and management capabilities.

Most importantly, SCADA (Supervision Control and Data Acquisition) data was acquired from 70, respectively 27 wind turbines operated by two different turbine operators located in Europe, over a span of 28 months from 2016 to 2018. A total of 312 variables were collected

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

from 70 wind turbines, where for each average, min, max and standard deviation values were captured with a 10 min frequency. For the remaining 27 wind turbines, 58 average values, 15 min values, 15 max values and 14 standard deviation values were captured with the same frequency.

In addition to the SCADA data, valuable information regarding work orders for large components, such as component categorization and healthy/unhealthy assessments, as well as elated work orders, such as component replacements were provided. This type of information gives an indication about the health status of a wind turbine and its core components, such as the main bearing, gearbox, generator, converter and transformer. Moreover, it allowed the construction of healthy datasets, which represented the normal baseline behavior of a wind turbine, as well as unhealthy datasets. Both were used further in the training and evaluation of various models for diagnosis and management.

Finally, vibration data was additionally collected and primarily used together with SCADA data for the learning processes of the model (i.e., learning non-linear relationships between input features). Vibration data primarily covers the vibration sensor measurements for the drive train. This is done in order to characterize gears and bearing failures in traditional gearboxes, main bearings and generators. Vibration data is used to characterize the kinematic behavior of the drive train, namely movement patterns, frequency patterns and amplitudes of cyclic signals related with drive train vibration over the time.

7.3 Model development cycle

The pre-processing step is concerned with extracting a feature set out of the datasets available. Since wind turbines have different failure modes, that exhibit significantly different behaviors, these are considered independently. Therefore, the feature extraction step is done based on the failure mode which is being processed. After the features have been selected, they are then processed in a manner that is ingestible by the machine learning model.

This processing primarily involves scaling the features since different features might have different variable spans. i.e., the temperature feature might vary between -10 and 30 degrees Celsius, while the power generated varies from 0 to ~5KW in the case of AD5 135 WT. High values in power generated might have higher impact on the ML model and therefore should be scaled by using tools such as the min-max approach, or by using a standard scaler.

Another major part of pre-processing is dealing with missing data. These can be filled using a forward or backward fill, or some more sophisticated approach such as using non-linear interpolation depending on the criticality of the feature being used. Features with a high count of missing values, or even a relatively lower count of consecutive missing values are generally dropped. Constant value features are also dropped, along with high or low entropy features. This feature dropping exercise improves the robustness of the machine learning models down the line.

The model selection is different for different failure modes since they can differ in the type of data used and quantity of labels. These two factors dictate the usability of approaches such as anomaly detection or regression.

The post-processing step is concerned with post-processing which involves the validation and retraining schedules of the machine learning models. Part of post-processing revolves around the standardization of the output format of each module.

Arguably the most important aspect of post-processing revolves around extracting value from the model outputs. This value extraction depends on the approach used in the machine learning model. For example, an anomaly detection approach using a seq-to-seq autoencoder

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

model outputs reconstructions of the original data. The post-processing of these reconstructions starts with comparison against the original input data to generate a reconstruction error. Furthermore, the reconstruction error by itself would still be insufficient and one would further need to extract a threshold which when exceeded signals a novelty. At this point the novelties are processed to decide if in fact this novelty corresponds to an anomaly or not.

Finally, as mentioned earlier, the retraining schedule is part of post-processing. This is a crucial part that heavily influences the robustness of the models. This is because a model that is trained using a relatively long period of time might start degrading in performance, and so a period should be set for training. For the models to model the latest wind turbines operation regimes (depending on period of year etc.) the models have to be retrained based on a predefined schedule. This schedule is defined for each failure mode separately and is done in a fixed interval manner. For example, a failure mode might require the models to be retrained every 6 months. The duration of the fixed interval is defined by experimentation on a range of values (2, 4, 6 months etc.) and by optimizing for the lowest performance error. Also, another factor for defining this schedule is computational constraints since training can very compute intensive.

7.4 Model selection process

Several suitable machine learning models will be selected per failure mode depending on the data type and label quantity that are made available. Depending on the relevance and similarity, these models are shared between different modules along with the pre-processing and post-processing, for e.g., the same forecasting model can be used for both the main bearing failures and gearbox failures given vibration data in the frequency domain. However, their hyperparameters are different.

Prior art in applying machine learning model to wind turbine failures focuses mostly on the use of neural networks. These models are trained with data samples that might contain multiple values for the same metric measured at different times. Recent developments of neural networks have created significantly more powerful ways to represent time, e.g., to learn time-delayed impact, from multiple input metrics, e.g., Temporal Convolutional Neural networks (TCN) [12], and Long Short Term Memory (LSTM) [13] networks.

Based on the nature of the failures and training data sets, the best suited neural network model will be selected. Furthermore, while neural networks are especially well suited to discover hidden convolutional representations e.g., in images, they also require abundant training data to perform well. Given this, the model selection process that will be followed is visualized below as a decision tree (Figure 7.1).

The most decisive factor when performing model selection is whether or not labels exist for a given failure mode. This dictates the feasibility of performing prognostics which would be the case if a significant ammount of labels is present. Otherwise only diagnostics solutions can be developed. If labels do exist for a given failure mode, the next question would be whether class balance exists within the label set. In other words, whether the minority set which in this use case represents failure events are more or less equal to the majority set which represents normal behaviour of the wind turbines. If this balance exists between labels, then supervised learning can be employed. This would yield the best results in terms of performance, and would allow for prognostic approaches to be considered. For example, a supervised classifier can be trained to predict if in the near future there would be a failure event or not. If there is imbalance between the labels, then based on the amount of available labels, once can check if this imbalance is severe or not. If there is a sufficient amount of minority labels, then one can resort to minority oversampling techniques, a benchmark example of this is SMOTE [14] and some of its more recent variation which can handle high dimensionality and temporal data. At this point one can once again perform supervised learning. Otherwise, if this is not the case,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

that is if the minority samples cannot be augmented, then the only approach which can be used is unsupervised learning.

In the case were labels do not exist for a given failure mode, one can check for the existance of target variables. These can be in the form of an already existing feature, or can be engineered using the expertise of the manufacturers or operators of the wind turbines. An example of this would be the extraction of a health index signal from the fundamental frequency and its harmonics. In this example the operators identify these frequencies of interest based on the relavent litterature on the topic and the machine design.

If these target variables do exist or can be extracted, then a forecasting approach can be employed which would also open up the possibility of performing prognostics for a given failure mode. Otherwise, the only approach that can be considered is unsupervised learning, such as the use of clustering techniques or seq-to-seq autoencoders.

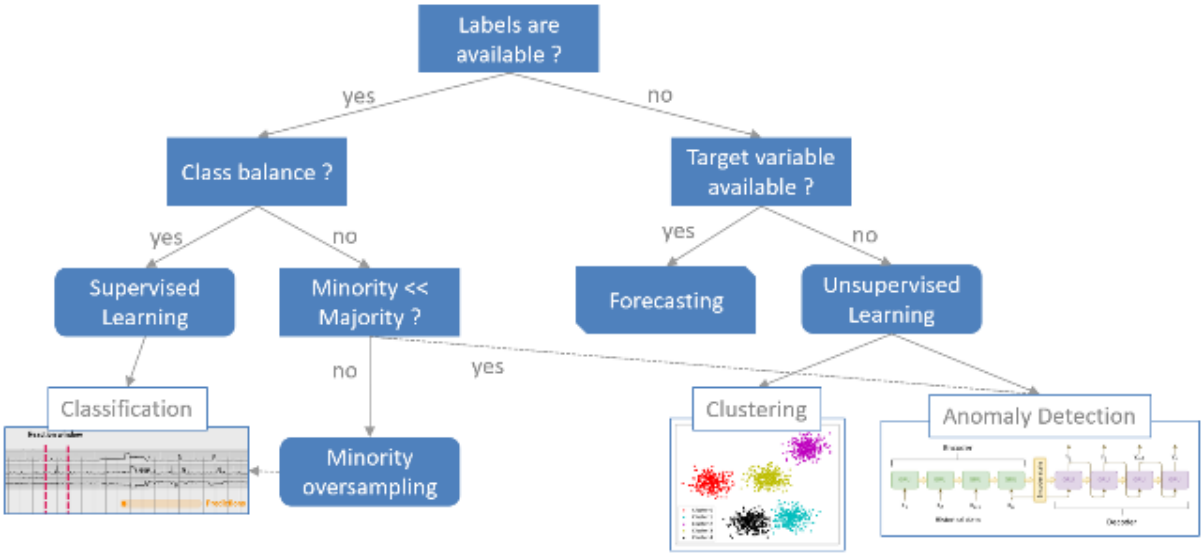

Figure 7.1 - Model selection decision tree

7.5 Forecasting scenario

When vibration data is available, a forecasting scenario can be tackled. By concatenating vibration data in the frequency domain acquired in an irregular fashion based on alarm events within the wind turbine, event series can be generated.

These data are then scaled using a min-max scaler to achieve optimum performance when using the activation functions used within the ML model. After scaling, missing data and constant values are processed in order to improve the model performance and to make it more robust.

These failure modes can be processed using forecasting since a target variable exists as a health index signal. This is extracted from the fundamental frequency and its harmonics which is then fed into the neural network as a target variable, as shown in Figure 7.2.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

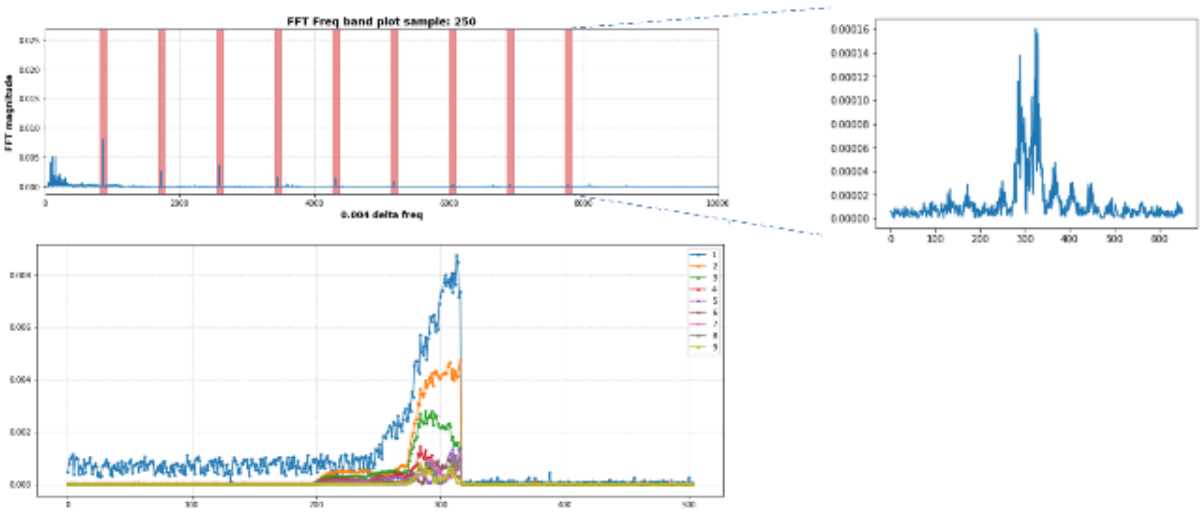

Figure 7.2 - Extraction of machine learning features when dealing with CMS data.

Regarding the modelling, an ML model that can model long term dependencies in the data, and that can handle a large input sample in the time domain is used. For this reason and since the aim is to perform forecasting, a Temporal Convolutional Network (TCN) model is selected. As the name suggests the basis for this model are convolutional operations. This might come as a surprise since sequence modelling is synonymous with the use of Recurrent Neural Networks (RNN) [15]. However, recent evaluations [16] indicate that convolutional architectures outperform recurrent networks on such tasks.

Figure 7.3 shows a representation of the TCN architecture. Mainly this architecture makes use of dilated convolutions which can span across a longer range of time series. This also results in deep networks, i.e., many layers, which can suffer from vanishing gradient problems. Therefore, we use these dilated layers within residual blocks in order to improve the learning process.

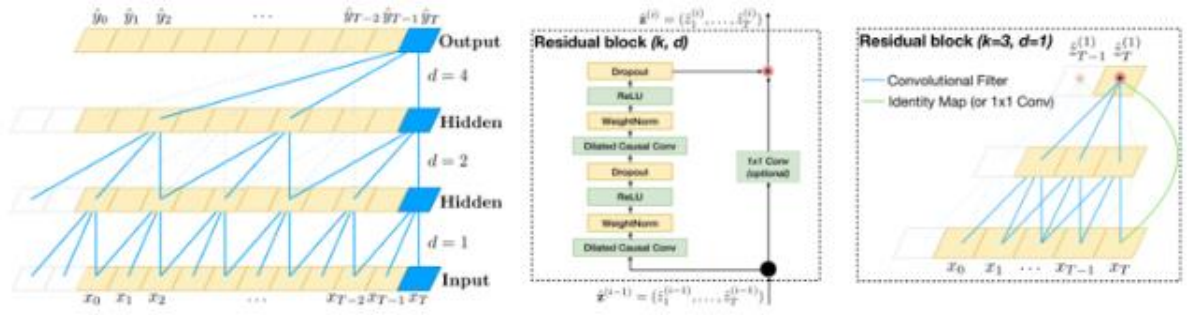

Figure 7.3 - Temporal Convolutional Network (from [12]).

The output of the TCN model is a list of values that predict the health index evolution in the next month. This can be adjusted to give longer predictions with the trade-off of less accuracy the further the predictions are from the last data sample.

If possible, a threshold of failure can be set for this health index. Then the predictions acquired from the model can be compared to that threshold for diagnosing the state of the component. And the predictions can also be used to detect the first hitting time, i.e., the point in time at which the health index reaches or crosses the threshold for the first time. This is then used to extract the remaining useful lifetime (RUL). An output forecast can be seen in Figure 7.4.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

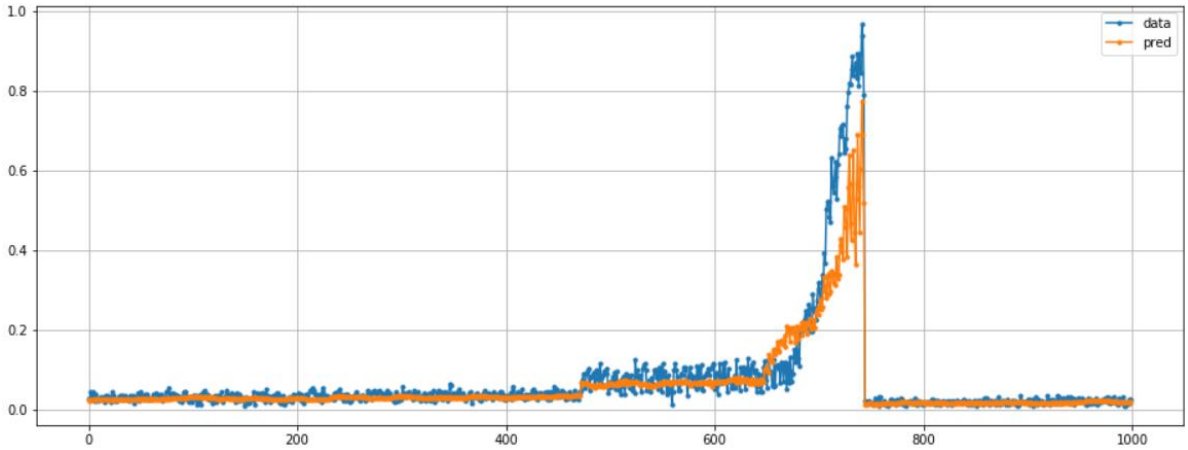

Figure 7.4 - Forecasting module output.

7.6 Anomaly detection scenario

When labels are scarce, an unsupervised approach needs to be used, rather than supervised. This dictates that other types of architectures for the machine learning models are more appropriate, such as a seq-to-seq autoencoder model which utilizes 1D convolutions that span across some k number of timesteps and all the features, as shown in Figure 7.5.

This autoencoder model uses a symmetrical architecture, whereby the encoder part of the network is symmetrical to the decoder. Usually, one can use an embedding size of 256 and a 2-layer encoder/decoder configuration. These parameters are of course subject to change where necessary and so might vary between modules. This model is trained using samples which are 144 timesteps that represent 1 day in time and the adam optimizer [17] for minimizing the L2 loss [18] between the reconstructions and the original data.

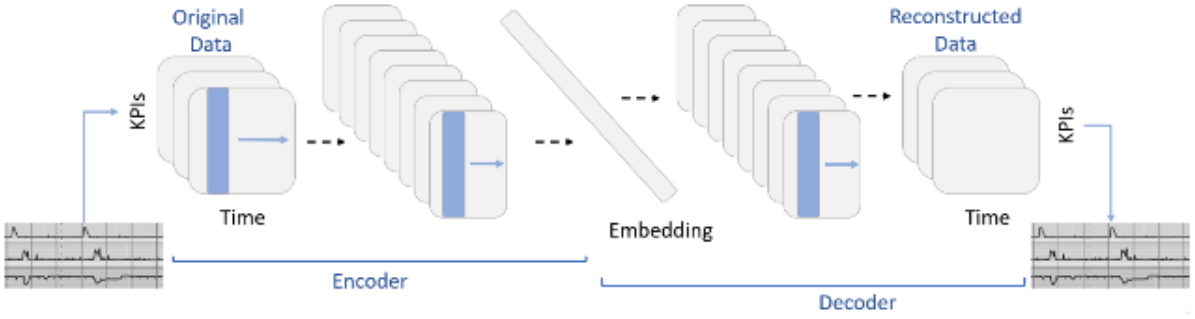

Figure 7.5 -1D CNN seq-to-seq autoencoder model.

To detect abnormal behavior, one classic approach is to compute the reconstruction error of the model by measuring the difference between the input and reconstructions. Then these errors which are computed on a feature basis are summed up to generate an error signal. The average error and standard deviation of that signal is extracted using the training data. This is then used to flag novel events.

The output of the model is the Aggregated reconstruction error over all features and a table with start and end times of anomalous events. A sample aggregate reconstruction error signal can be seen in Figure 7.6.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

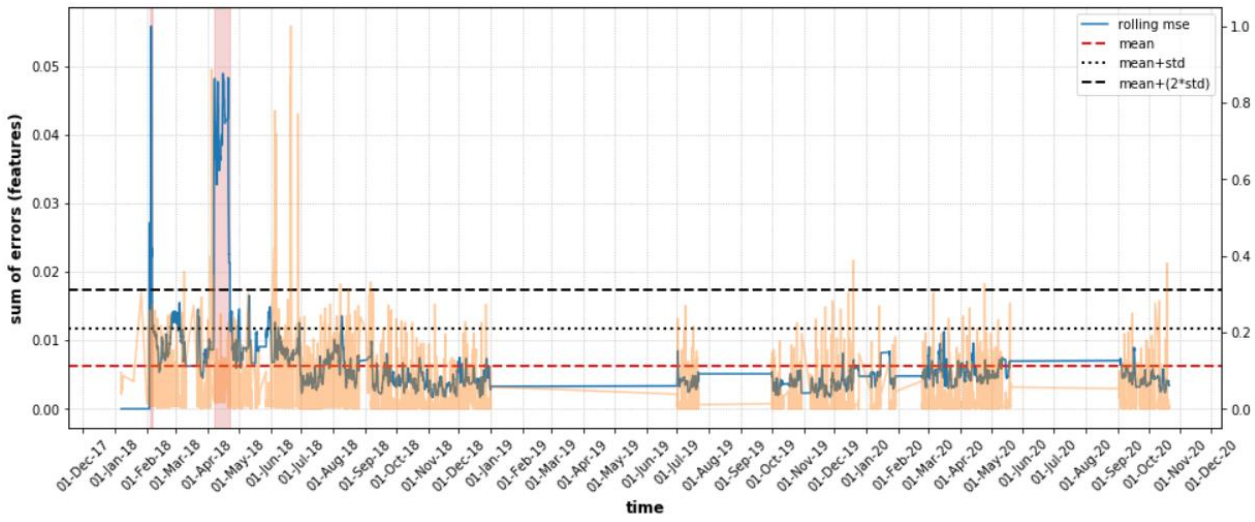

Figure 7.6 - Anomaly detection module output.

The yellow background signal is the aggregate error signal. This is then smoothed to achieve the signal represented in blue which will be considered for anomaly detection. Also, in the figure we have horizontal lines representing the mean reconstruction error and 1 and 2 standard deviations away from that. If at any point the smoothed error signal breaches the 2 standard deviation threshold, we consider that event to be a novelty as shown using the red highlight in the figure. If these novel events occur in sequence for some predefined n number of timesteps, then this is considered as an anomaly. The rest of the point anomalies are discarded as false positives. This is further post-processed in order to assess the performance of the model given ground truth data, i.e., computing precisions and recall and so on. This of course depends on the module and availability of ground truth data.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

8 Case study: Intelligent Neural Weight in Motion System with High Accuracy for automatic penalties on over-loaded vehicles [MOS]

8.1 Use case description

Static and dynamic vehicle weighing systems are used to prevent the degradation of infra-structure caused by excessive axle or total weight of heavy vehicles. Both parameters are limited to ensure proper levels of road safety. Negative impact of overloading on the safety of traffic cannot be omitted, since a huge number of overloaded vehicles are still present. Weight in-motion systems have been studied as an optimization tool for bridge assessment – in the first place it was possible to model the traffic load on the bridge. The effort has been put into reducing the number of overloaded vehicles. Data aggregated and transmitted after weighing procedure are shared with the Road Transport Inspectorate or other authorities to manage the vehicles which break the weight limits. To avoid excessive expenditures on road maintenance different static and dynamic weighing systems should be implemented [120].

One of the Weight-in-Motion platform which has been developed under the Intelligent Devel-opment Program in 2018 in Poland is a very precise WIM software – NeuroCar WIM, which is a part of the large platform – NeuroCar for Intelligent Transportation Systems utilized for vari-ous purposes (e.g., management of the parking spaces, management of the road lightning).

Architecture of the WIM-System can be mostly described as two parts – measuring stations and central systems for managing the stations and data captured (Figure 8.1).

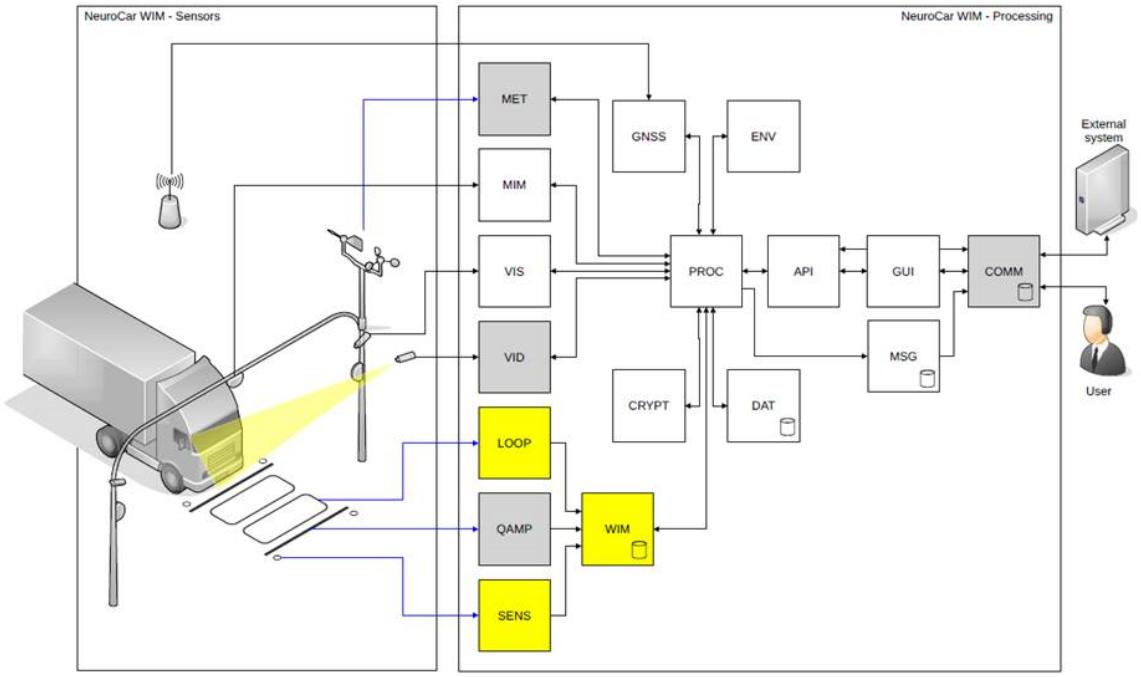

Figure 8.1 - Organization of the modules in the measurement station [121].

The measuring station is equipped with:

- camera for monitoring of the selected lane,
- measuring camera with an infrared light source,
- quartz sensor system in the road pavement,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

- inductive loops in the road pavement,
- supporting structure to cover the width of the carriageway
- computing terminal consisting of the hardware (computer, controller, transformer),
- software – NeuroCar – for data processing and transmission algorithms.

Software NeuroCar utilizes the concept of microservices and the protocol based on open data formats. Communication is based on per-to-per connections (no data bus) in either IPv4 or IPv6 infrastructure.

8.2 Data machine learning algorithm

One of the subsystem utilized in the WIM architecture of the NeuroCar WIM consists of the Vehicle Identification module, which enables automatic identification of the vehicles based on the images generated by a camera. Video identification subsystem core consist of the software built on artificial neural network technology which uses DSP techniques.
The use of neural networks enable automatic recognition of:

- model of the vehicle,
- color of the vehicle,
- vehicle classes,
- content of the registration plate,
- country of origin.

Photo sequences supplied by a video camera from the observation point are subjected to the image processing analysis to detect the presence of the vehicles. Input data for neurons are images in the form of registration plates. The neural network recognition process is divided into three phases (Figure 8.2):

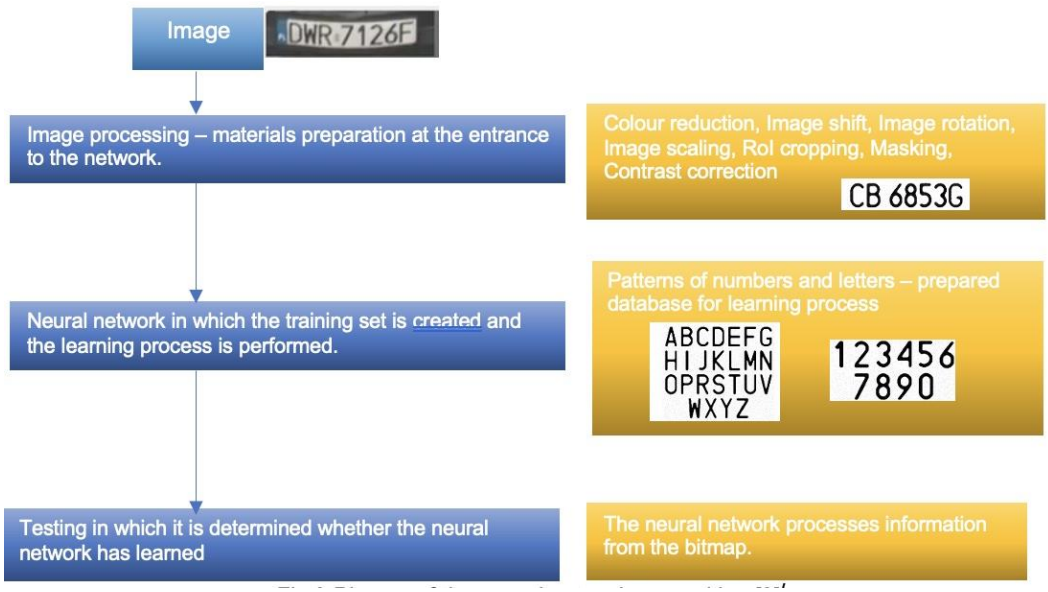

Figure 8.2 - Phases of the neural network recognition [122].

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

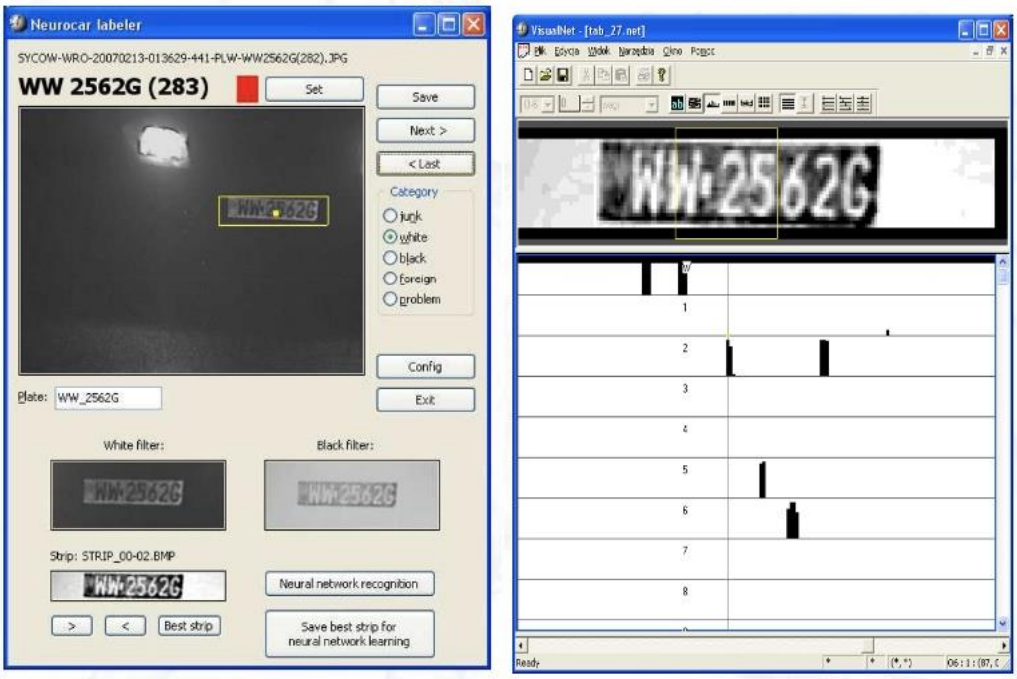

Figure 8.3 - NeuroCar deep neural network classifier [123].

The scheme of structure of the subsystem for image recognition, where neural networks have been implemented is presented below (Figure 8.4):

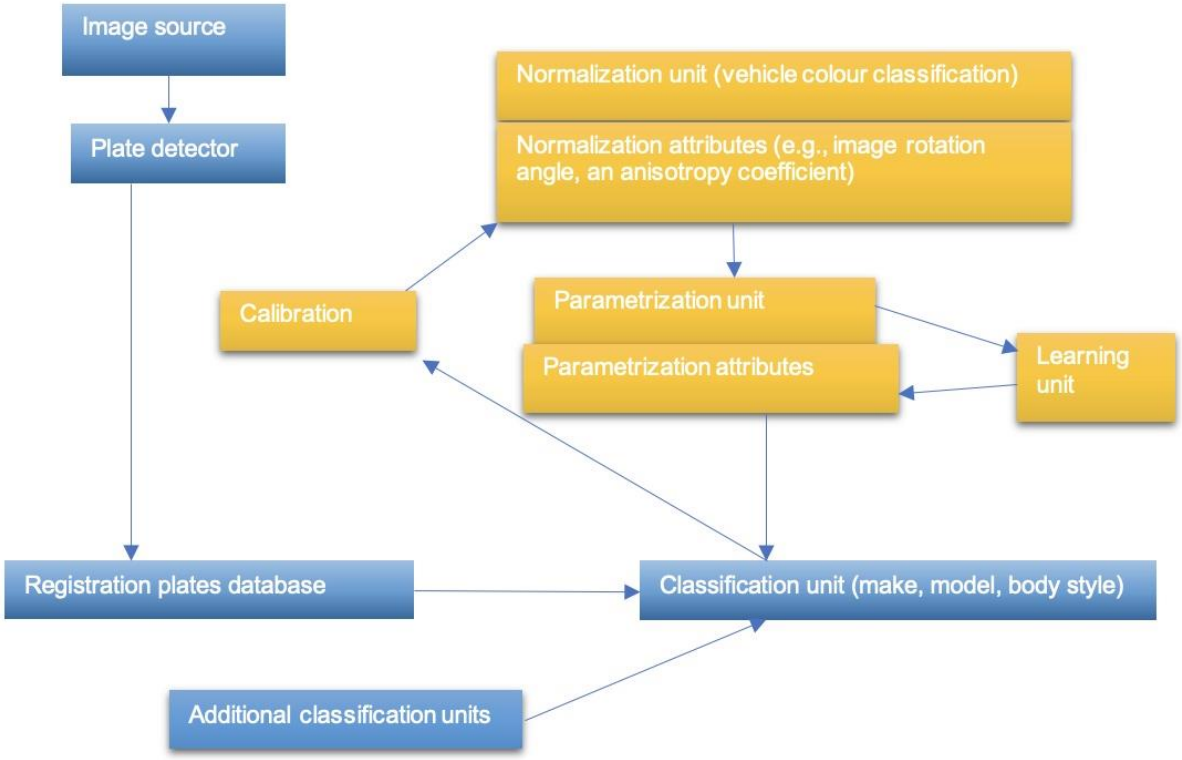

Figure 8.4 - Image recognition subsystem [124].

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

8.3 Utilization scenario

The platform NeuroCar WIM has been implemented as a monitoring system for the Road Inspectorate authorities (General Directorate for National Roads and Motorways in Poland) in different national regions. According to the Neurosoft Sp. z o.o. company website portfolio the platform has been successfully applied on 36 lanes in Poland and the average number of overloaded vehicles in the month from one measuring station is 1200 [125].

The user interface depends on the license of the software and needs – can be extended with additional functionalities. Graphical user interface allows to view measurement data, configuration changes and turn on/off the activities (Figure 8.3). Interface has a form of a web application, which can be opened on tablets, computers and mobile phones. Inside the platform there are all measurement points of the system implemented, which can be easily tracked through (Figure 8.5).

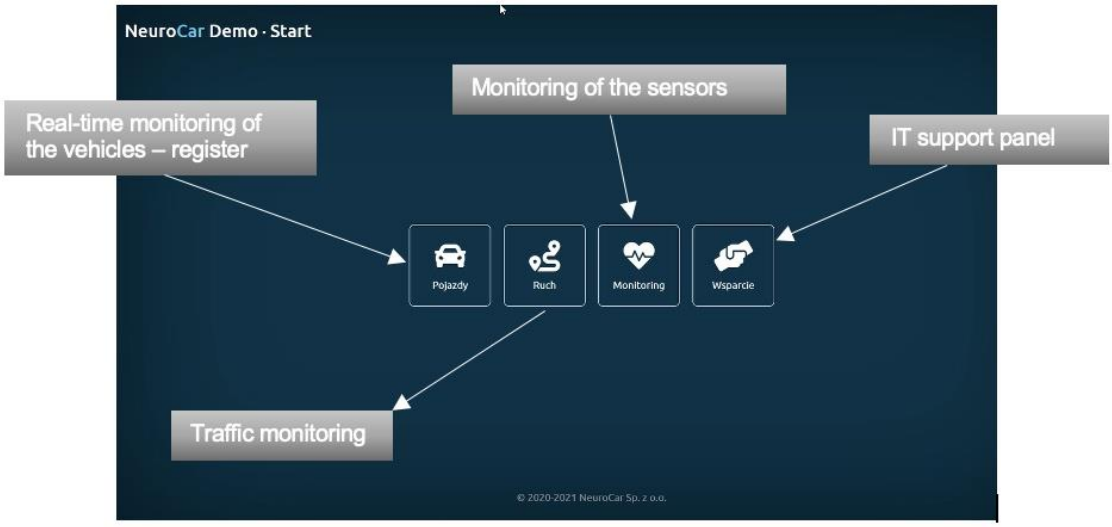

Figure 8.5 - Initial panel of the platform.

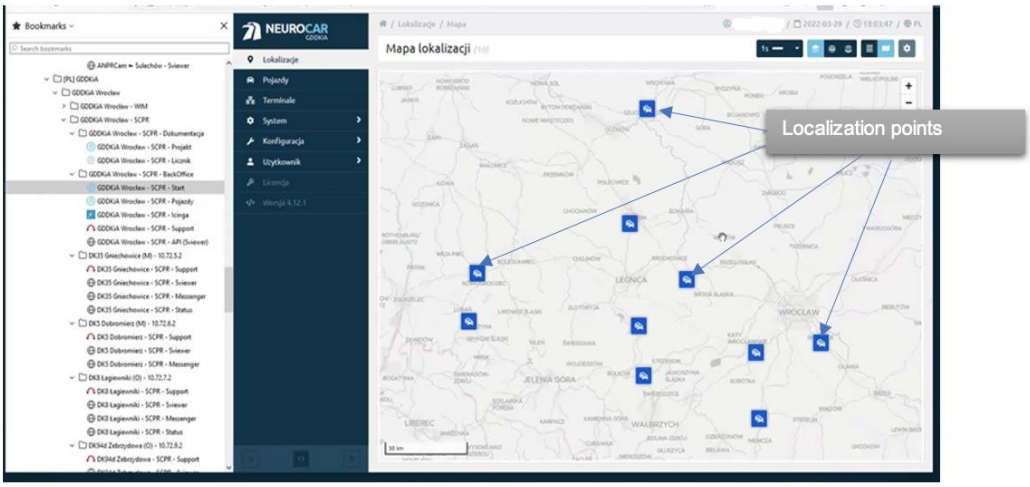

Figure 8.6 - Localization map of the measurement stations (example).

In the 'Traces' tab there is a real-time database of all vehicles passing by the measuring points with the exact time and other identification data (Figure 8.6). The list can be changed to the overview with an outline of the vehicles (Figure 8.7) and their sketches (Figure 8.8).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171. © 2021

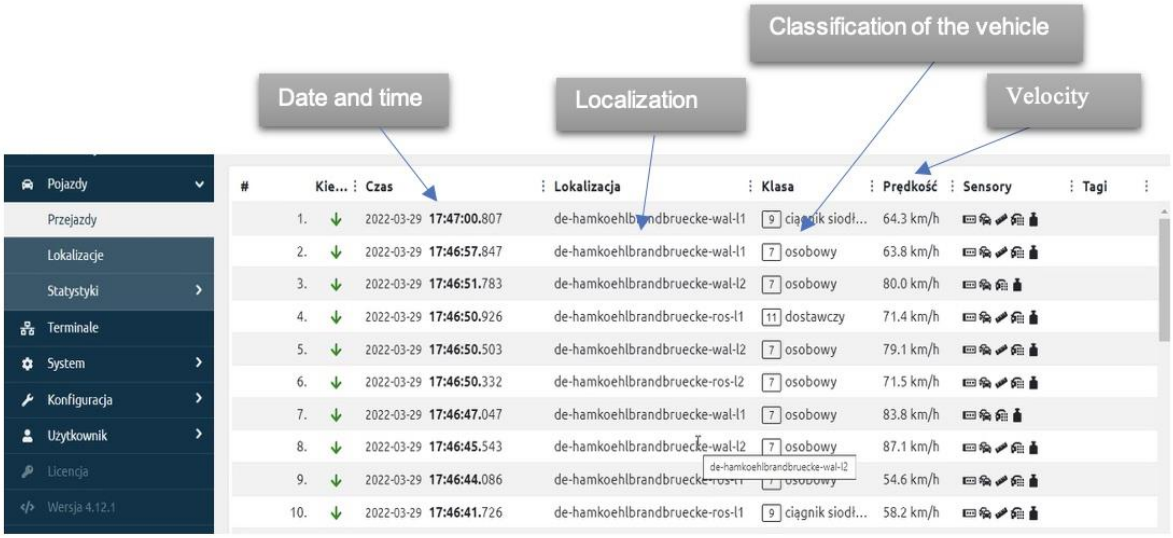

Figure 8.7 - Register of the vehicles in a form of a list.

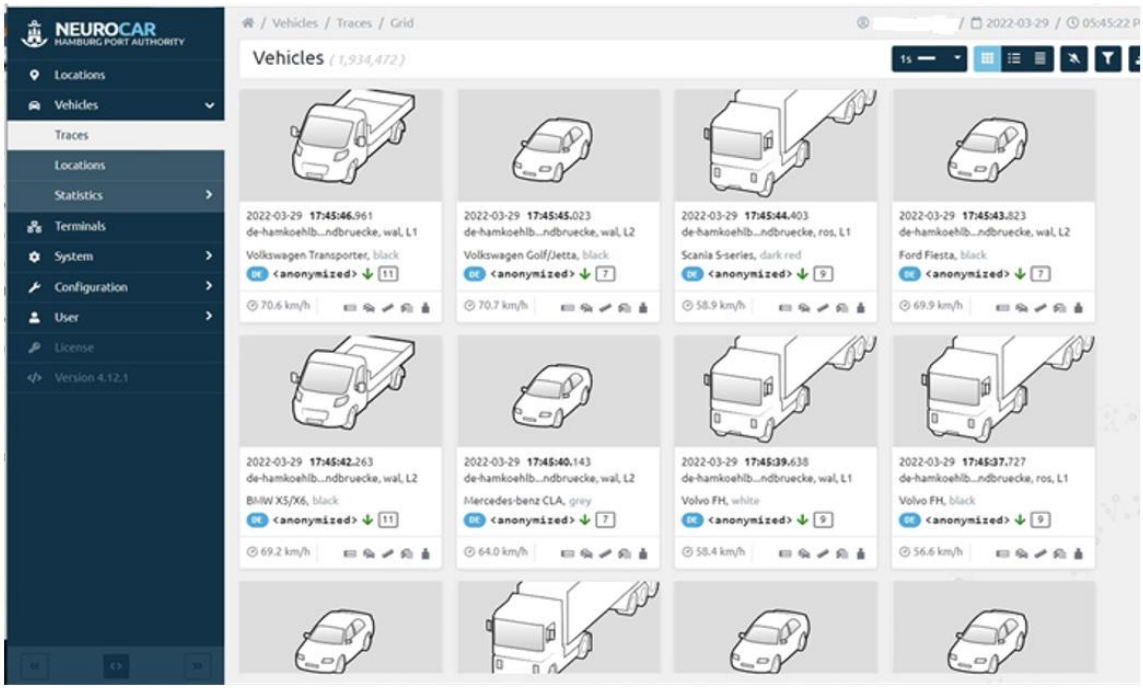

Figure 8.8 - Register of the vehicles with the body sketches.

The database contains details with the captured photography, register plate and all the necessary information to identify vehicles that violate traffic regulations and load limits (Figure 8.9).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

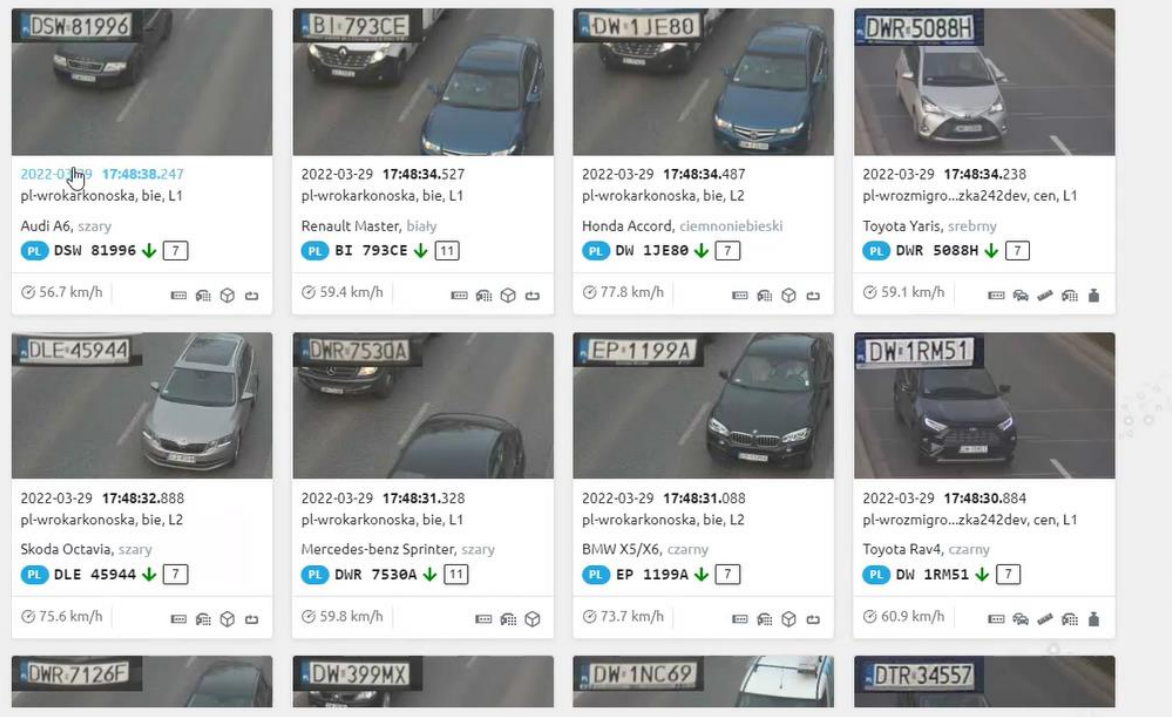

Figure 8.9 - Register of the vehicles passed by the lane in the selected period of time.

In NeuroCar platform there is also module called '*Measure-in-Motion'* which analyses the data signals from the LIDAR system mounted above the measuring point. '*Measure-in-Motion'* module allows to:

- detect the vehicle in the measurement field,
- detect the direction of movement,
- detect the 3D body of the vehicle (Figure 8.10),
- calculate the height profile (detection of exceeding the permissible height),
- classification of vehicles with accordance to the 3D model.

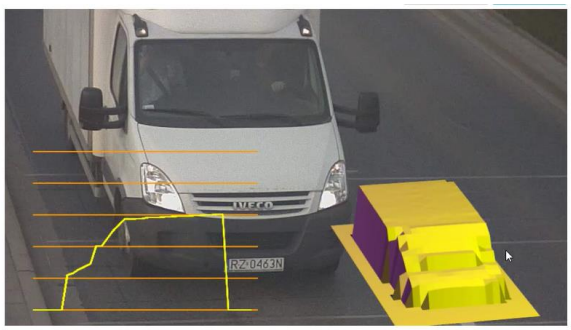

Figure 8.10 - 3D model of the vehicle

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171. © 2021

9 Bibliography

[1] Sund&Baelt, https://sundogbaelt.dk/

[2] Matrix DJI 200 RTK, https://www.dji.com/ch/matrice-200-series

[3] Storebælt East Suspension bridge, https://en.wikipedia.org/wiki/Great_Belt_Fixed_Link

[4] Sacertis, https://sacertis.com

[5] SuperAnnotate, https://superannotate.com

[6] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey", arXiv preprint arXiv:2001.05566, 2020.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards realtime object detection with region proposal networks," in Advances in Neural Information Processing Systems, pp. 91–99, 2015.

[8] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969, 2017.

[9] Y. Bai, A. Yilmax, and H. Sezen, "End-to-end Deep Learning Methods for Automated Damage Detection in Extreme Events at Various Scales", in International Conference on Pattern Recognition, 2020.

[10] Detectron2, https://github.com/facebookresearch/detectron2

[11] albumentations, https://albumentations.ai/

[12] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal Convolutional Networks: A unified approach to acrtion segmentation", arXiv preprint arXiv:1608.08242, 2016.

[13] S. Hochreiter, and J. Schmidhuber, "Long short-term memory", in Neural Computation, vol. 9(8), pp. 1735-1780, 1997.

[14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique.", in Journal of artificial intelligence research, vol. 16, pp.321-357, 2002.

[15] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", arXiv peprint arXiv:1808.03314, 2018.

[16] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling", in arXiv preprint arXiv:1803.01271, 2018.

[17] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization", in International Conference on Learning Representations, 2015.

[18] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss Functions for Image Restoration with Neural Networks", arXiv preprint arXiv:1511.08861v3, 2018.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

[19] COCO dataset, https://cocodataset.org/

[20] OpenCV, https://opencv.org/

[21] PyTorch, https://pytorch.org/

[22] TensorFlow, https://www.tensorflow.org/

[23] SuperbAI, https:// https://www.superb-ai.com/

[24] databricks, https://databricks.com

[25] Neptune AI, https://neptune.ai/

[26] H. Zhang, M. Cisse, Y. N. Dauphin, D.-L. Paz, “mixup: Beyond empirical risk minimization”, in International Conference on Learning Representations, 2018.

[27] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, Y. Yoo, “CutMix: Regularization strategy to train strong classifiers with localizable features”, arXiv preprint arXiv:1905.04899v2, 2019.

[28] D. Dwibedi, I. Misra, and M. Herbert, “Cut, Paste and Learn: Surprisingly easy synthetis for instance detection”, in International Conference on Computer Vision, 2017.

[29] L. McInnes, J. Healy, and J. Melville, “UMAP: Uniform Manifold Approximation and Projection for Dimensions Reduction”, arXiv preprint arXiv:1802.03426, 2018.

[30] L. J. P. van der Maaten, and G. E. Hinton, “Visualization high-dimensional data using t-SNE”, in Journal of Machine Learning Research, vol. 9, pp. 2579-2605, 2008.

[31] J. Lever, M. Krzywinski, and N. Altman, “Principal component analysis”, in Nature Methods, vol. 14, pp. 641-642, 2017.

[32] Git, https://git.com

[33] Mercurial, https://www.mercurial-scm.org/

[34] Amazon S3, https://aws.amazon.com/s3/

[35] Apache Kafka, https://kafka.apache.org/

[36] DVC, https://dvc.org/

[37] docker, https://www.docker.com/

[38] TensorBoard, https://www.tensorflow.org/tensorboard

[39] Pearson correlation coefficient, https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

[40] A. Ng, “Feature selection, L1 vs. L2 regularization, and rotational invariance”, in International Conference on Machine Learning, 2004.

[41] LDA, https://en.wikipedia.org/wiki/Linear_discriminant_analysis

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 958171.

[42] L. K. Saul, and S. T. Roweis, “An introduction to locally linear embedding”, in Journal of Machine Learning Research, 2001.

[43] Kullback-Leibler divergence, https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

[44] CVAT, https://github.com/openvinotoolkit/cvat

[45] K. K. Maninis, S. Caelles, J. Pont-Tuse, and L. van Gool, “Deep Extreme Cut: From extreme points to object segmentation”, in Computer Vision and Pattern Recognition, 2018.

[46] Supervisely, https://supervise.ly/

[47] Hasty, https://hasty.ai

[48] Labelbox, https://labelbox.com/

[49] v7 Darwin, https://www.v7labs.com/

[50] Diffgram, https://diffgram.com/main/

[51] DataTorch, https://datatorch.io/

[52] LabelMe, http://labelme.csail.mit.edu/Release3.0/

[53] VoTT, https://vott.z22.web.core.windows.net/

[54] mlflow, https://mlflow.org/

[55] Tableau, https://tableau.com

[56] Microsoft Power BI, https://powerbi.microsoft.com/

[57] Apache Hadoop, https://hadoop.apache.org/

[58] Apache Spark, https://spark.apache.org/

[59] NoSQL databases, https://en.wikipedia.org/wiki/NoSQL

[60] J. Dean, and S. Ghemawat, “MapReduce: Simplified data processing on large clusters”, in Symposium on Operating System Design and Implementation, 2004.

[61] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Sgarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O’Malley, S. Radia, B. Reed, and E. Baldeschwiele, “Apache Hadoop YARN: Yet Another Resource Negotiator”, in Symposium on Cloud Computing, 2013.

[62] Naïve Bayes classifiers, https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[63] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines”, IEEE Intelligent Systems and their Applications, vol. 4, 1998.

[64] J. R. Quinlan, “Induction of decision trees”, in Machine Learning, vol. 1, pp 81-106, 1986.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 958171.

[65] L. Breiman, “Random forests”, in Machine Learning 45, pp 5-32, 2001.

[66] T. Chen, and C. Guestrin, “XGBoost: A scalable tree boosting system”, in Knowledge Discovery and Data mining, 2016.

[67] A.Patle, and D. S. Chouhan, “SVM kernel functions for classification”, in International Conference on Advances in Technology and Engineering, 2013.

[68] W. Y. Loh, “Classification and regression trees”, in Data Mining and Knowledge Discovery, 2011.

[69] E. Rasmussen, “Gaussian processes in machine learning”, In Lecture Notes in Computer Science, 1998.

[70] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise”, in Knowledge Discovery and Data Mining, 1996.

[71] k-means clustering, https://en.wikipedia.org/wiki/K-means_clustering

[72] Hierarchical clustering, https://en.wikipedia.org/wiki/Hierarchical_clustering

[73] V. Tresp, “Mixture of Gaussian processes”, in Advances in Neural Information Processing Systems, 2000.

[74] S. Singla, B. Nushi, S. Shah, E. Kamar, and E. Horvitz, “Understanding Failures of Deep Networks via Robust Feature Extraction”, In Computer Vision and Pattern Recognition, 2021.

[75] D. Hoiem, Y. Chodpathumwan, and Q. Dai, “Diagnosing error in object detectors”, in European Conference on Computer Vision, 2012.

[76] D. Bolya, S. Foley, J. Hays, and J. Hoffman, “TIDE: A general toolbox for identifying object detection errors”, in Computer Vision and Pattern Recognition, 2020.

[77] A. Borji, and S. M. Iranmanesh, “Empirical upper-bound in object detection and more”, arXiv preprint arXiv:1911.12451, 2019.

[78] COCO Analysis Toolkit, COCO Analysis Toolkit: http://cocodataset.org/#detection-eval

[79] Apache Airflow, https://airflow.apache.org/

[80] PREFECT, https://www.prefect.io/

[81] TensorFlow Lite, https://www.tensorflow.org/lite

[82] Kolmogorov-Smirnov test, https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test

[83] Y. Shen, Y. Xiong, W. Xia, and S. Soatto, “Towards backward-compatible representation learning”, in Computer Vision and Pattern Recognition, 2021.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 958171.

[84] S. Yan, Y. Xiong, K. Kundu, S. Yang, S. Deng, M. Wang, W. Xia, and S. Soatto, “Positive-congruent training: towards regression-free model updates”, in Computer Vision and Pattern Recognition, 2021.

[85] S. Soltan, H. Khan, and W. Hamza, “Limitations of knowledge distillation for zero-shot transfer learning”, in EMNLP Workshop on Simple and Efficient Natural Language Processing, 2021.

[86] R. Duggal, H. Zhou, S. Yang, Y. Xiong, W. Xia, Z. tu, and S. Soatto, “Compatibility-aware heterogeneous visual search”, in Computer Vision and Pattern Recognition, 2021.

[87] Mattu JA Jeff Larson,Lauren Kirchner,Surya. Machine Bias [Internet]. ProPublica. 2016 [cited 2022 Apr 14]. Available from: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[88] Heaven WD. Predictive policing algorithms are racist. They need to be dismantled. [Internet]. MIT Technology Review. 2020 [cited 2022 Apr 14]. Available from: https://www.technologyreview.com/2020/07/17/1005396/predictive-policing-algorithms-racist-dismantled-machine-learning-bias-criminal-justice/

[89] Larrazabal AJ, Nieto N, Peterson V, Milone DH, Ferrante E. Gender imbalance in medical imaging datasets produces biased classifiers for computer-aided diagnosis. Proc Natl Acad Sci U S A. 2020 Jun 9;117(23):12592–4.

[90] O’Neil C. Weapons of math destruction: How big data increases inequality and threatens democracy. Broadway Books; 2016.

[91] Floridi L. Establishing the rules for building trustworthy AI. Nat Mach Intell. 2019 Jun;1(6):261–2.

[92] European Commission. Communication: Building Trust in Human Centric Artificial Intelligence | Shaping Europe’s digital future [Internet]. 2019 [cited 2022 Apr 14]. Available from: https://digital-strategy.ec.europa.eu/en/library/communication-building-trust-human-centric-artificial-intelligence

[93] Wirth R, Hipp J. CRISP-DM: Towards a standard process model for data mining. In: Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining. Manchester; 2000. p. 29–40.

[94] Varshney KR. Trustworthy machine learning. Chappaqua, NY, USA: Independently Published; 2022.

[95] Wan A. Making Decision Trees Accurate Again: Explaining What Explainable AI Did Not [Internet]. The Berkeley Artificial Intelligence Research Blog. 2020 [cited 2022 Apr 14]. Available from: http://bair.berkeley.edu/blog/2020/04/23/decisions/

[96] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770–8.

[97] DARPA. Broad Agency Announcement, Explainable Artificial Intelligence (XAI). DARPA-BAA-16-53. 2016;7–8.

[98] Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell. 2019 May;1(5):206–15.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 958171.

[99] Arya V, Bellamy RK, Chen P-Y, Dhurandhar A, Hind M, Hoffman SC, et al. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. arXiv preprint arXiv:190903012. 2019.

[100] Kim B, Wattenberg M, Gilmer J, Cai C, Wexler J, Viegas F, et al. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). arXiv:171111279 [stat] [Internet]. 2018 Jun 7 [cited 2020 Sep 23]; Available from: http://arxiv.org/abs/1711.11279

[101] Alvarez-Melis D, Jaakkola TS. Towards Robust Interpretability with Self-Explaining Neural Networks. arXiv:180607538 [cs, stat] [Internet]. 2018 Dec 3 [cited 2020 Jun 3]; Available from: http://arxiv.org/abs/1806.07538

[102] Kindermans P-J, Hooker S, Adebayo J, Alber M, Schütt KT, Dähne S, et al. The (un) reliability of saliency methods. In: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. Springer; 2019. p. 267–80.

[103] Su J, Vargas DV, Sakurai K. One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation. 2019;23(5):828–41.

[104] Barbiero P, Ciravegna G, Giannini F, Lió P, Gori M, Melacci S. Entropy-based Logic Explanations of Neural Networks. arXiv:210606804 [cs] [Internet]. 2021 Jun 22 [cited 2021 Jun 24]; Available from: http://arxiv.org/abs/2106.06804

[105] Ghorbani A, Wexler J, Zou JY, Kim B. Towards automatic concept-based explanations. Advances in Neural Information Processing Systems. 2019;32.

[106] Yeh C-K, Kim B, Arik S, Li C-L, Pfister T, Ravikumar P. On completeness-aware concept-based explanations in deep neural networks. Advances in Neural Information Processing Systems. 2020;33:20554–65.

[107] Koh PW, Nguyen T, Tang YS, Mussmann S, Pierson E, Kim B, et al. Concept bottleneck models. In: International Conference on Machine Learning. PMLR; 2020. p. 5338–48.

[108] Goyal Y, Feder A, Shalit U, Kim B. Explaining Classifiers with Causal Concept Effect (CaCE). arXiv:190707165 [cs, stat] [Internet]. 2020 Feb 28 [cited 2020 Sep 18]; Available from: http://arxiv.org/abs/1907.07165

[109] Kazhdan D, Dimanov B, Jamnik M, Liò P, Weller A. Now you see me (CME): concept-based model extraction. arXiv preprint arXiv:201013233. 2020;

[110] Li O, Liu H, Chen C, Rudin C. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2018.

[111] Chen C, Li O, Tao C, Barnett AJ, Su J, Rudin C. This Looks Like That: Deep Learning for Interpretable Image Recognition. arXiv:180610574 [cs, stat] [Internet]. 2019 Dec 28 [cited 2021 Jan 11]; Available from: http://arxiv.org/abs/1806.10574

[112] Rigotti M, Miksovic C, Giurgiu I, Gschwind T, Scotton P. Attention-based Interpretability with Concept Transformers. In 2022 [cited 2022 Feb 24]. Available from: https://openreview.net/forum?id=kAa9eDS0RdO

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.

[113] Bellamy RKE, Dey K, Hind M, Hoffman SC, Houde S, Kannan K, et al. AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias [Internet]. 2018. Available from: https://arxiv.org/abs/1810.01943

[114] Microsoft. Responsible AI Toolbox [Internet]. Microsoft; 2022 [cited 2022 Apr 15]. Available from: https://github.com/microsoft/responsible-ai-toolbox

[115] Guo C, Pleiss G, Sun Y, Weinberger KQ. On Calibration of Modern Neural Networks. arXiv:170604599 [cs] [Internet]. 2017 Aug 3 [cited 2022 Apr 15]; Available from: http://arxiv.org/abs/1706.04599

[116] Niculescu-Mizil A, Caruana R. Predicting good probabilities with supervised learning. In: Proceedings of the 22nd international conference on Machine learning - ICML '05 [Internet]. Bonn, Germany: ACM Press; 2005 [cited 2022 Apr 15]. p. 625–32. Available from: http://portal.acm.org/citation.cfm?doid=1102351.1102430

[117] Vaughan JW, Wallach H. A human-centered agenda for intelligible machine learning. Machines We Trust: Getting Along with Artificial Intelligence. 2020;

[118] Amershi S, Weld D, Vorvoreanu M, Fourney A, Nushi B, Collisson P, et al. Guidelines for Human-AI Interaction. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems [Internet]. New York, NY, USA: Association for Computing Machinery; 2019 [cited 2022 Apr 15]. p. 1–13. (CHI '19). Available from: https://doi.org/10.1145/3290605.3300233

[119] NSCAI. Final Report of the National Security Commission on Artificial Intelligence [Internet]. NSCAI. 2021 [cited 2022 Apr 15]. Available from: https://www.nscai.gov/

[120] B. G. Monika Wasilewska, "2019 Signal Processing Symposium (SPSympo)," in *Convolutional neural network based vehicle classification* , Wroclaw, Poland , 2019.

[121] Neurosoft, "NeuroCar API – Documentation – Version 4," 10 04 2021. [Online]. Available: https://gitlab.com/ncar-tools/04/api.

[122] W. Dobrosielski, "Using artifical neural networks in image rcognition," in *Applied Computer Sciences, Materials, Volume 2* , Bydgoszcz, Uniwersytet Kazimierza Wielkiego, Instytut Techniki, Poland , 2010, pp. 45-56.

[123] Neurosoft, "Artificial intelligence - Innovative Solutions," in *Artificial intelligence for road safety and public order* , Wroclaw, 2008.

[124] T. D. C. D. P. M. Michal Karkowski, "Method of Vehicle Identification and a System for Vehicle Identification". United States Patent US 2015/0294174 A1, 15 October 2015.

[125] Neurosoft Sp. z o.o. , "Products/ Neurocar/ Weight in Motion," Neurosoft Sp. z o.o. , 2022. [Online]. Available: https://en.neurosoft.pl/products/neurocar/neurocar-weight-in-motion/. [Accessed 15 April 2022].

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958171.