Understanding Data Pre-processing with a Hybrid Diagram Integrating ER and Data Flow Notation

Robin Varghese
University of Houston[§]
USA

Carlos Ordonez
University of Houston
USA

***Abstract*—In a data science project, a significant effort is spent pre-processing databases, text and image data sets, before a machine learning model can be computed. Moreover, the analytic process is iterative, requiring a team of data scientists to add and remove features (attributes) from the target data set going all the way back to data sources. However, such programming effort is carried out without a data model behind, which results in redundant and inconsistent data sets and source code difficult to extend and maintain. On the other hand, the ER model has a proven track record to design databases before developing source code. Heeding the importance of data-centric models to store and analyze data, we propose a hybrid diagram (FLOWER=FLOW+ER) mixing data objects and processing flow, under the CRISP standard. Specifically, our novel diagram mixes ER entities (data objects) and processing steps (function calls). We explain how diagram creation can be partially automated by parsing source code. We present two case studies illustrating how our proposed hybrid diagram works in two challenging problems from biomedical engineering and computer vision: (1) detecting and classifying spikes in biomedical signals and (2) labeling identified objects in images. We argue our diagram can reduce code development time, enhance team collaboration and eliminate redundant data. We hope our work will motivate research bridging database design, project management and machine learning.**

I. Introduction

In most data science projects most code development time is spent pre-processing data sets because data sets come from diverse sources, they have different structure, they come in different file formats and they are not integrated. Hence data data scientists need to collect, integrate, clean, merge, aggregate, and transform data files before they can perform analysis. This is achieved using many powerful libraries (mainly in Python these days), but the programming effort remains significant. Such data transformations create many intermediate files, tables, even matrices, in a disorganized manner. This problem becomes more difficult when pre-processing big data beyond alphanumeric data sets (i.e. traditional statistical or data mining analysis). Today the AI analytic challenge generally involves a combination of plain data files, databases, text, images and even video.

ER diagrams have a proven track record to represent data structure and relationships, beyond databases. The ER diagram strengths are generality, flexibility, and intuitive visual representation. On the other hand, flow diagrams [8], [12], despite being old, remain the main mechanism to visualize major components or main processing steps of a software system, but they are less useful to understand complex algorithms. Several closely related works on ER diagrams [7], [4], flow diagrams [2], [13], and data transformations [12], [10] have been done by other researchers.

This work proposes combining them, but with a "data-centric" angle. We propose a hybrid diagram (FLOWER), to assist data scientists in big data pre-processing in a broad "Variety" sense, the 3rd V in Big Data [3].

II. Definitions

Let $E = \{E_1, ..., E_n\}$, be a set of n entities, linked by relationships. Thinking of entities as vertices in a graph G, G must be connected. Given the evolution of ER and ER design tools we follow modern UML notation, where entities are represented by rectangles and relationships are shown by lines, with crowfeet on the "many" side (also called :N side); We assume there exists an identifying set of attributes for each entity (i.e. a primary key, a file name, a variable name, an object id). Intuitively, entities correspond to nouns (real-life objects or items) and relationships to verbs (actions).

Diagram notation extension: We extend relationships with an arrow in the middle, indicating direction of data flow. Such arrow is labeled with function names or expressions parsed from the source code, instead of verbs like traditional ER design. This flow direction is interpreted as input and output, going from source entities going to destination entities (transformed somehow). We argue this is a minor, yet powerful, change that enables navigating data sets in a data lake or complex program, providing a data-oriented flow.

III. Related Work

We are not the first to propose diagrams to understand data pre-processing, but to the best of our knowledge we are the first to attempt to extend modern UML diagram notation for ER with a minor flow symbol, instead of proposing alternative notations. Previous work proposed a framework (analytic component architecture) and a method (steps) to generate diagrams from source working on databases and plain big data files (logs, csv, SQL files) [9]. Furthermore, this idea has been expanded to include typical database processing in Python (Pandas) as

[§]Contact: rsvarghese99@gmail.com, Department of Computer Science, University of Houston, Houston TX 77204, USA

979-8-3503-2445-7/23/$31.00

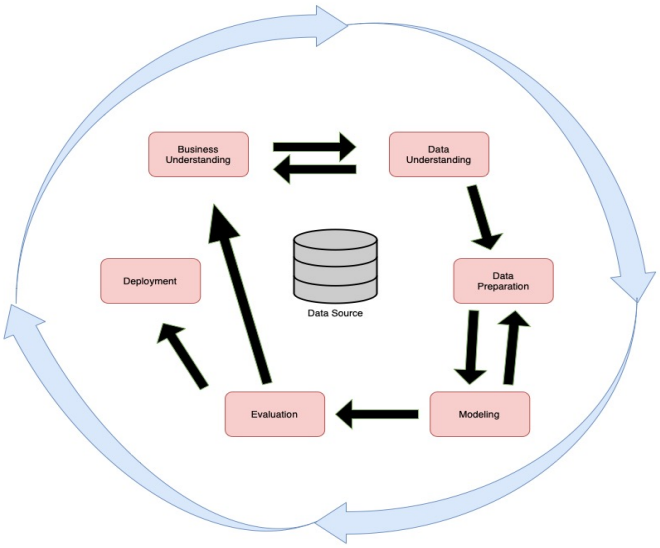

Fig. 1: CRISP-DM main steps.

detailed in [6]. This integration with Python is important given the significant research in the field of data science today. Our contribution builds upon this foundation by extending the work to biomedical signals and image data sets. We manually generate the diagrams to serve as a roadmap for future research.

IV. Proposed FLOW+ER=FLOWER Diagram

A. Cross Industry Standard Process for Data Mining (CRISP-DM)

The Cross Industry Standard Process for Data Mining (CRISP-DM) has been gaining significant attention as the process model designed to provide a structured framework for planning and executing large data mining projects across many domains [11]. As Big Data continues to grow, many businesses are placing greater emphasis on effective project management. CRISP-DM stands out as a leading methodology in this regard.

The CRISP-DM standard, shown in Figure 1, comprises of six major iterative phases:

1) **Business Understanding:** Focuses on understanding the project objectives and requirements from a business perspective. Furthermore, converting this understanding into defining a data mining problem and producing a preliminary plan.
2) **Data Understanding**: Encompasses gathering, describing, and exploring data, understanding its properties, and identifying data quality issues.
3) **Data Preparation**: Here, data is cleaned, transformed, and enriched for modeling. Data preparation is crucial because it can improve model accuracy, efficiency, and scalability.
4) **Modeling:** Different modeling techniques can be applied that most align with the nature of the problem
5) **Evaluation:** The model is evaluated to ensure proper alignment with the predefined business objectives.

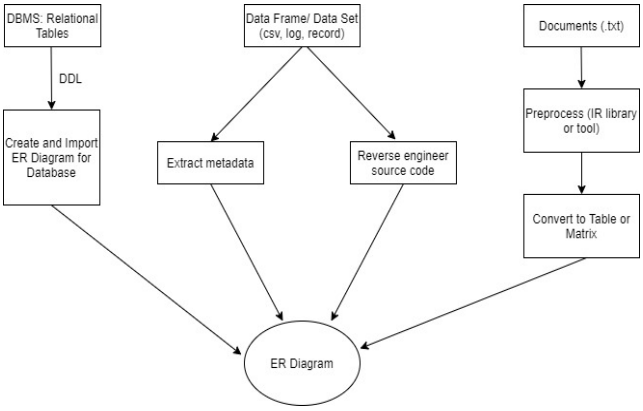

Fig. 2: Example FLOWER Diagram.

6) **Deployment:** This step involves integrating the model into an business or operational environment. The results of the data mining process are made accessible to its respective users.

Our contributions focuses on Data Understanding and especially Data Preparation. We believe providing an intuitive visual understanding of these two phases can greatly aid in the process of CRISP-DM. More importantly, we extend a ”data-centric” perspective on Data Understanding and Preparation in general, benefiting all project managers in Big Data.

B. Building Diagram

First, our solution generates a preliminary ER-Flow diagram as follows. This diagram can be polished and customized by the data scientist. We show this example process in Fig 2.

1) For text files like documents, source code we assume they contain strings for words, numbers, symbols and so on. In this case, we utilize a Information Retrieval (IR) library to parse the code or document. This will then allow us to identify the entities such as classes, functions, and variables, along with their attributes including data types and parameters. These elements are then organized into tables or matrices for further analysis.
2) For images we assume the file name is unique and the image has some metadata describing its content (e.g. a jpg file).
3) Automatic data set name and attribute name identification computed and created by Python, R, or SQL code.

The diagram data is stored in two JSON files, where the first file contains the relationships and the second one contains the entities and their attributes. In the transformation module, we define the transformation type and create new transformed entities. Data scientists may perform several transformations discussed above in the source code that generates a temporary entity. In the case of “Merge”, the entity structure may change but the attribute values remain the same, and the ”Aggregation” may use one or more grouping attributes along with or without aggregations (sum, count, avg). In general, aggregations will return numbers, but using only “Group by” will return the attribute values as their types. Mathematical transformations will mostly return derived attributes. Now, the new transformed

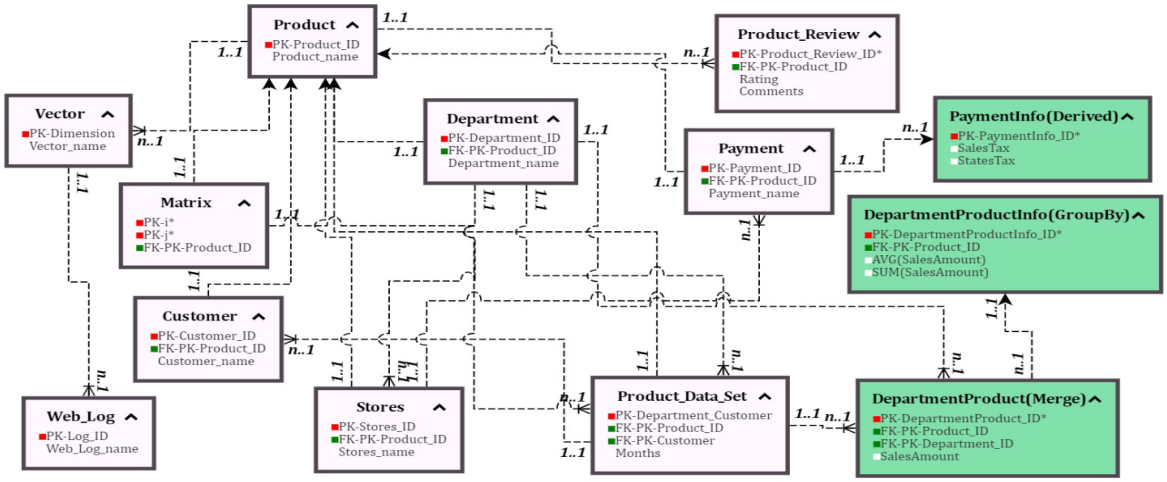

Fig. 3: FLOW+ER diagram for a typical database example.

entities are linked with the original entities using an arrow. After each valid transformation step, we can store the newly generated ER-Flow diagram in JSON files. This ER-Flow diagram can help data scientists to have data-oriented view of the program, navigate source code, reuse functions, and avoid creating redundant data sets.

We generally follow these phases when creating FLOWER based on practical case studies.

1) **Source Entities:** Using a filename as a primary key is valid because filenames are unique identifiers for files within a directory. This ensures that each record associated with a particular filename remains distinct and easily retrievable.
2) **Transformations Entities:** Our case studies examine applications in AI. AI is full of data transformations and often inherit attributes from the data source.
3) **Linking and Relationships:** We link entities together with relationships and are described by the Python functions that perform the transformations.
4) **Draw Diagram:** Given the Source/Transformation entities and their respective relationships, we generate the UML and ER hybrid diagram.
5) **Finalized Data Set:** AI/ML data sets are often written as one final data set and fed through a training loop. We offer two case studies involving one or multiple finalized data sets.
6) **Human In The Loop:** While prior steps can be automated, there are scenarios where human intervention is essential. For instance, in our image-based case study [5], dentists must first annotate the panoramic radiographs using the VGG Image Annotator (VIA).

C. Diagram Elements

Our entity concept is broad: entities can represent a file, matrix, relational table, or dataframe. A file can contain an image, a document or data records. Objects in main memory can be simple data types (integers, reals, string, dates), lists, multidimensional arrays and data frames. In short, our data scope goes well beyond previous analytic approaches focusing only on records with alphanumeric attributes (SQL tables, plain CSV files).

Entities are classified as source (raw) entities, representing raw data, loaded into the Data Lake and Transformation (pre-processing) entities being the output of some tool or programming language (Python, R, SQL).

In our generalized diagram entities represent a set of data elements. In turn, data elements can be records, SQL rows, image pixels, text keywords, video frames. Moreover, data elements can be aggregated bottom up to obtain more abstract entities.

D. Data Transformations

We focus on representing data transformations for big data analytics, including machine learning, graphs, and even text files (documents). However, our diagram does not represent the ”analytic output” such as the parameters of the ML model, IR metrics like precision/recall, graph metrics. We propose these major categories of data transformations:

1) Merge, which splices (joins) multiple entities. We can think of it as a generalized relational join operator ($\bowtie$).
2) Aggregation, which partitions data elements and computes some aggregate function.
3) Mathematical, which represent derived attributes coming from a combination of functions and value-level operators (e.g. equations, arithmetic expression, nested function calls).
4) Filter, which do not transform the data sert, but which is key to focus on the right data.

E. Example

We show an example in Figure 3 where we show our final ER-Flow diagram. We consider an example of a store

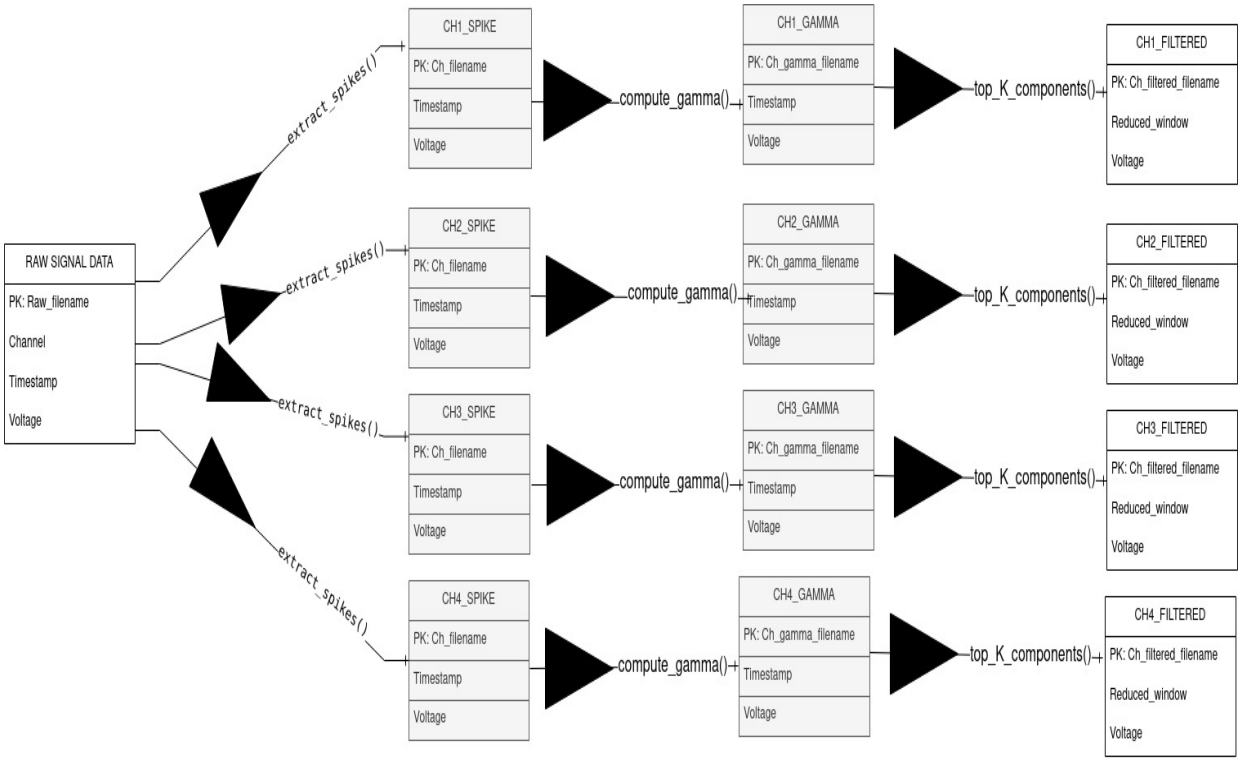

Fig. 4: FLOWER diagram for biomedical signal pre-processing.

for which we show the ER-Flow diagram. In our example the target analytic is a predictive model of product sales considering history sales data, customer information and buyers' opinions. The goal is to produce a data set, which can be used as input for a predictive model like regression, decision trees, SVMs or deep neural networks. Each entity from the original data has an identifying attribute (primary key) and other attributes. From these entities, data scientists can generate new entities by doing data transformations as mentioned above. Popular analytic languages like Python and R, both support data transformations (ex: "Merge", "Group by") in pandas and dpylr libraries respectively. Each of the transformations generates a new entity which is named from the input entities and the transformation type is shown inside the parenthesis as "(TYPE)". The source entities are colored white and the transformed entities are colored green for better understanding. We can see the flow of the transformed entities as they are linked with an arrow from the source entities.

V. Case Studies

To highlight the distinct features of this FLOWER implementation compared to the previous version in Figure 3, we observe the following key differences: The arrows are now labeled with specific function names or expressions, extracted directly from the source code. This contrasts with the traditional ER design, which typically uses verbs. Additionally, the direction of these arrows indicates the flow of data, representing input and output processes. This flow moves from the source entities towards the transformed entities.

We feature two AI/ML examples with FLOWER diagramming. For credibility, we opted for these case studies because they have publicly available source code (APPENDIX) and corresponding published research. In the presented case studies, the source entities are colored white and the transformed entities are colored grey. The data flow of the transformed entities are linked with an arrows and the corresponding Python functions performing the transformations.

A. Biomedical Signals

In this section we present a practical example illustrating how to pre-process a set of biomedical signal data for clustering. [1]. This paper delves into the challenges and advancements in identifying similar patterns in physiological nerve signals collected from micro electrical sensors in animal organs. The primary challenge is discerning these patterns, which appear as spikes within millisecond time-windows amidst high-dimensional data sets, especially with the interference of background electrical noise.

- **Objective:** The main aim is to detect similar patterns in high throughput nerve signal data.
- **Previous Systems:** Earlier methods combined PCA (Principal Component Analysis) and K-means clustering but were slow and required multiple tools.
- **Proposed System:** The paper introduces an integrated system that combines signal filtering, feature engineering, and multidimensional data summarization for a more effective integration of PCA and K-means clustering.

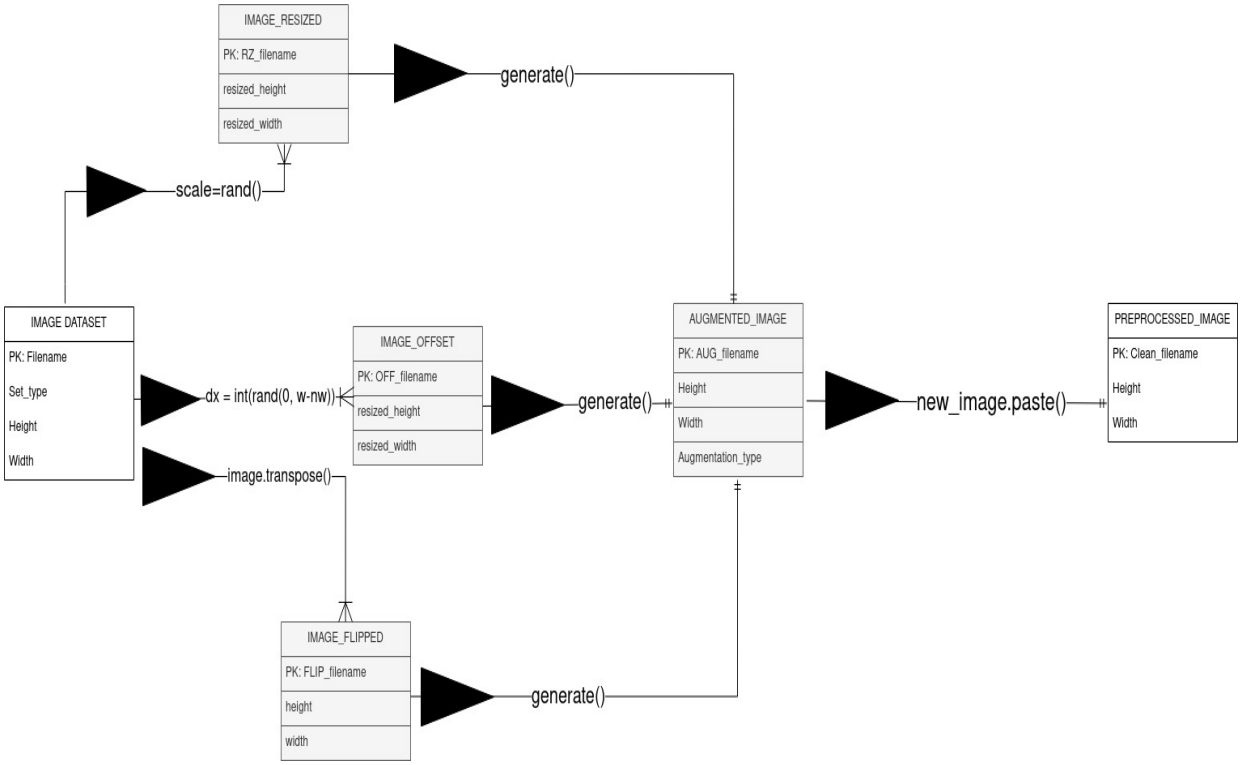

Fig. 5: FLOWER diagram for image pre-processing.

- **Applications:** The ultimate goal is to associate signal patterns with specific physiological functions, potentially leading to innovative medical treatments via nerve stimulation.
- **Contribution:** The research offers an efficient method to analyze multiple continuous signals over time, detecting signal patterns across them using machine learning techniques.
- **Implementation:** The entire system is implemented in Python, a popular language in Big Data and Data Science.

The transformations applied source (raw) images are as follows:

1) **Computing Correlations Between Channels:**
   - Correlations are computed based on split raw data sets ({D1, D2, ..., DM}).
   - An incremental algorithm computes the correlation matrix using a summarization matrix formed by multiplying the combined raw data set with its transpose (e.g. $D_1 D_1^T$, ..., $D_M D_M^T$).
2) **Filter Noise And Detect Spikes:** Noise is filtered and spikes are detected from the correlated channels.
3) **Reduce Dimensions to $\hat{d}$ Dimensions:** Original variable values are retained instead of using principal components.

B. Images

This section presents an example illustrating how to pre-process image files for a computer vision problem, aiming at identifying and labeling objects [5]. The paper presents a novel method for the automated diagnosis of periodontitis bone loss using panoramic radiographs. Addressing the current challenges in diagnostic accuracy with these radiographs, the authors introduce a two-stage convolutional neural network named PDCNN. Their approach, tested on a comprehensive data set, achieves superior accuracy compared to other contemporary models, promising a more efficient and accurate diagnostic tool for this prevalent oral disease.

- **Objective:** The primary goal is to automate the diagnosis of radiographic bone loss (RBL) in panoramic radiographs.
- **Previous Systems:** Existing deep learning methods for this task faced challenges in diagnosis accuracy and implementation.
- **Proposed System:** The paper introduces the PDCNN, a two-stage periodontitis detection convolutional neural network.
- **Applications:** The developed system is intended for use in medical settings to make RBL diagnosis using panoramic radiographs more efficient and accurate.
- **Contribution:** The research introduces a novel method for RBL detection that outperforms existing state-of-the-art models, providing more accurate and efficient results.
- **Implementation:** The proposed PDCNN and associated techniques, is implemented in Python.

The transformations applied to the source (raw) images are as follows:

1) **Data Splitting:**
   - The data set is divided into training, validation, and

test sets in a 7:1:2 ratio.

2) **Data Augmentation:**

- Images from the training set are randomly resized.
- Images are offset.
- Images are horizontally flipped.
- These augmented images are then placed onto a gray background of size 512×512 to counteract overfitting.

ACKNOWLEDGMENT

We would like to acknowledge Elijah Mitchell, Nabila Berkani, and Ladjel Bellatreche for their contributions in preliminary efforts and research that laid the groundwork for this study.

REFERENCES

[1] Sikder Tahsin Al-Amin, Robin Varghese, David Lloyd, Maria A. Gonzalez-Gonzalez, Mario I. Romero-Ortega, and Carlos Ordonez. Discovering similar spike patterns in high dimensional biomedical signals. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 4337–4345, 2022.

[2] Carlo Batini, Enrico Nardelli, and Roberto Tamassia. A layout algorithm for data flow diagrams. *IEEE Trans. Software Eng.*, 12(4):538–546, 1986.

[3] Xin Luna Dong and Divesh Srivastava. Big data integration. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 1245–1248. IEEE, 2013.

[4] Gaoyang Guo. An active workflow method for entity-oriented data collection. In *Advances in Conceptual Modeling - ER 2018 Workshops Emp-ER, MoBiD, MREBA, QMMQ, SCME, Xi'an, China, October 22-25, 2018, Proceedings*, 2018.

[5] Zhengmin Kong, Hui Ouyang, Yiyuan Cao, Tao Huang, Euijoon Ahn, Maoqi Zhang, and Huan Liu. Automated periodontitis bone loss diagnosis in panoramic radiographs using a bespoke two-stage detector. *Computers in Biology and Medicine*, 152:106374, 2023.

[6] Elijah Mitchell, Nabila Berkani, Ladjel Bellatreche, and Carlos Ordonez. Flower: Viewing data flow in er diagrams. In *Big Data Analytics and Knowledge Discovery: 25th International Conference, DaWaK 2023, Penang, Malaysia, August 28–30, 2023, Proceedings*, page 356–371. Springer-Verlag, 2023.

[7] Jonathan Mugan, Ranga Chari, Laura Hitt, Eric McDermid, Marsha Sowell, Yuan Qu, and Thayne Coffman. Entity resolution using inferred relationships and behavior. In *IEEE International Conference on Big Data*, pages 555–560. IEEE Computer Society, 2014.

[8] C. Ordonez and J. García-García. Managing big data analytics workflows with a database system. In *IEEE/ACM CCGrid*, pages 649–655, 2016.

[9] Carlos Ordonez, Sikder Tahsin Al-Amin, and Ladjel Bellatreche. An er-flow diagram for big data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5795–5797, 2020.

[10] Minh Pham, Craig A. Knoblock, and Jay Pujara. Learning data transformations with minimal user effort. In *IEEE International Conference on Big Data (BigData)*, pages 657–664, 2019.

[11] Christoph Schröer, Felix Kruse, and Jorge Marx Gómez. A systematic literature review on applying crisp-dm process model. *Procedia Computer Science*, 181:526–534, 2021.

[12] Merlijn Sebrechts, Sander Borny, Thomas Vanhove, Gregory van Seghbroeck, Tim Wauters, Bruno Volckaert, and Filip De Turck. Model-driven deployment and management of workflows on analytics frameworks. In *IEEE International Conference on Big Data*, pages 2819–2826, 2016.

[13] Petia Wohed, Wil M. P. van der Aalst, Marlon Dumas, Arthur H. M. ter Hofstede, and Nick Russell. Pattern-based analysis of the control-flow perspective of UML activity diagrams. In *Conceptual Modeling - ER 2005*, volume 3716, pages 63–78, 2005.

APPENDIX

The source code for each case study can be found at the following github repositories:

- Biomedical Signals
  https://github.com/RobinVar/pygammasignal
- Images
  https://github.com/PuckBlink/PDCNN