Assignment WASP SW & Cloud course Module 2

Software Engineering Course 2025

Riccardo Tedoldi
riccardo.tedoldi@astrazeneca.com

1 Introduction

In-silico drug design aims to tackle a complex problem: developing workflows to uncover potent drugs to target specific diseases. Recent advances in deep-generative modeling [1, 2] have created new avenues to model complex data distributions, such as natural language [3], images [4], 3D shapes modeling [5, 6] and, in the context of life sciences, molecular structure generation [7]. Computational techniques that use deep-generative models for molecular structure generation targeting specific receptors or respecting certain properties can be divided into two main categories: string-based [8] and 3D graph-based [9] methods.

In the first category, a SMILES-based model is typically pre-trained on a large set of molecules (e.g., ChEMBL [10], PubChem [11]) and then fine-tuned on a smaller set of molecules relevant to the target of interest (i.e., Dopamine receptor D2 [12]). Finally, strategies such as reinforcement learning [12] are employed during chemical space exploration to steer the model toward favorable regions of the chemical space.

On the other hand, 3D graph-based methods can be trained for unconditional generation [13], but they are often conditioned on protein structures or pockets for conditional generation, given that they operate in 3D [14, 15]. Although these methods have shown outstanding results in recent years, there is still room for improvement in terms of the quality of generation (e.g., conformational energy states [16], conditional generation based on binding affinity and potency [15]), as well as training and inference speed [17]. My research currently focuses on 3D-based generative models, in particular on the development of new algorithms to improve the unconditional generation of molecular structures and the conditional generation that relies on contextual information, such as protein structures or Molecular Dynamics (MD) data.

As a PhD student, I am not involved on a daily basis in the development of production-level code bases with teams of more than 10 people. Instead, I typically work on smaller code bases, developed entirely from scratch either by myself or in collaboration with one or two other researchers.

Especially during the early research stages of a project, I often rely on the code developed by other researchers in the field (e.g. metrics, data processing pipelines, data splits) and then adapt it to my needs. I use GitHub to keep track of the code I develop and for versioning, but most importantly to ensure that the results I obtain at each step are reproducible.

Reproducibility is a key aspect of my research, especially because I work with diffusion models [18] to generate 3D molecular structures. These deep generative modeling frameworks are particularly known for their training instability [19, 20], making reproducibility crucial for publishing. Testing is another important aspect, particularly when training models or running MD simulations on large sets of structures. To address this, I perform training on a reduced dataset, sometimes with fewer training parameters or using toy problems, and I run MD simulations on a limited number of smaller systems.

2 Lecture principles

Requirements: As we are going through an era where not only the "software" per se, but also the weights of the ML models are programmable, new regulations and legal requirements are emerging (AI Act [21]). It is important to refine these requirements according to the application domain. Here, I provide a non-exhaustive list of requirements applicable to the case of 3D molecular structure generation using diffusion models.

Data: Since the weights of the model depend on the training data, it is critical to ensure that the data meet quality requirements and are sufficiently representative, especially for validation and testing. Additionally, during MD simulations, it can be particularly challenging to set up and define the configuration files so that the simulation run properly. This difficulty is sometimes due to a lack of pre-made examples and can be ameliorated by developing standardized processing pipelines to ensure consistency and scoring components to evaluate the representativeness of the data.

Software Design: Often PhD students focus on prototyping ideas rather than on software design (design patterns [22])

Documentation: Documentation is a requirement when publishing the software alongside a paper.

Openness: Openness is an important requirement in the scientific community, especially for PhD students. This involves open-sourcing code and data to ensure maximum transparency of the results obtained.

Explainability/confidence value: In the context of molecular research, several studies are still addressing explainability [23]. This can be a very important factor when it comes to human-in-the-loop approaches.

Model's limitations: The limitations of the model are uncovered by testing it on real protein pockets for binders design. Highlighting these limitations ensures that the model is used as a tool to explore potential candidate molecules or to optimize the existing ones, rather than as an oracle for designing new molecules.

Software Reproducibility and Maintenance: When PhD students release the code along with the paper, they typically move on to the next project and do not maintain the code. Therefore, it is important to provide a make file to re-create the environment to run the code and a README file that explains how to run the code and reproduce the results. However, if someone submits a pull request on GitHub to fix a bug or add a new feature, I am always happy to review it and merge it.

Pragmatic design and development: During code development, especially when adding new features or fixing bugs, it is important to follow version control practices. I typically create new branches, which are merged into the main branch only after testing. This also ensures tracking and reversibility of the changes made, allowing a return to a previous version if needed. For commits, I write meaningful commit messages to ensure that all changes are understandable.

3 Guest-Lecture Principles

Protect private data: As an industrial PhD student at AstraZeneca, I am subject to several restrictions when it comes to sharing the code and weights of the model. The weights cannot be shared publicly because they can be exploited to leak information about the data used for the training.

User oriented design and research development: Currently, I am not working on the development of a new complete 3D molecular design software, but I am focusing instead on the development of new strategies to include new features in the current state of the art models for molecular generation

present in the literature. I am prioritizing the features to be included based on a user-oriented scale and once I have a concrete idea of the functionality, I proceed by identifying the requirements needed to achieve them. Therefore, as a first step, it is fundamental to "understand the problem before you build the solution" and then define it at different levels of abstraction. This last step can be extremely challenging, but talking with people that have already worked on similar problems or have experience in the field can help, resulting in reduced wasted effort.

4 Data Scientists versus Software Engineers

Read chapters "1. Introduction" and "2. From Models to Systems" of the CMU "Machine Learning in Production" book (https://mlip-cmu.github.io/book/01-introduction.html) and then answer:

Q1. **Do you agree on the essential differences between data scientists and software engineers put forward in these chapters? Why or why not?**

I agree with the differences highlighted in the chapters. Data scientists care about the data, exploiting algorithms to generate predictive models and identify patterns and trends. In other words, they focus on which information can be extracted from the data and how to do that. They are not directly concerned with creating a product. It should be noticed that, in many cases, data scientists work in contexts where production considerations are not relevant, because their analyses are directly used within the company, for example to support decision-making [24, 25]. On the other hand, software engineers are less focused on the data itself and more on building a cohesive system that is scalable and "user-friendly", taking into account not only accuracy and robustness but also cost and time [26]. A recent talk of Andrej Karpathy (`https://www.youtube.com/watch?v=LCEmiRjPEtQ`), founding member of openAI and former director of Tesla AutoPilot, highlighted how the software paradigm is shifting from classical programming to querying a neural network. He predicted this transition from software 1.0 to software 2.0 in his blog post (`https://karpathy.medium.com/software-2-0-a64152b37c35`) already on Nov 11, 2017. In this paradigm, a new research branch of software engineering is focusing on this [27], and data scientists and ML scientists can benefit from software like Copilot to quickly write, refactor, document and debug code. In his vision Andrej reports that understand how to use and prompt well it will be key.

Q2. **Do you think these roles will evolve and specialise further or that "both sides" will need to learn many of the skills of "the other side" and that the roles somehow will merge? Explain your reasoning.**

I believe that both trends will occur at the same time, similar to the idea of T-shaped experts discussed in the chapters. It is important that data scientists and software engineers communicate and cooperate together in order to understand how their work fits into the bigger scheme, to remain updated on the changes in the system and to minimize the occurrence of assumptions and misconceptions, which could result in errors and therefore delays [28][29]. This need of constant exchange of knowledge between data scientists and software engineers requires that, despite their different backgrounds and approaches to problems, they have at least a broad understanding of each other's skills. I believe that learning some of the skills of "the other side" will result in a "shared language" that facilitates cooperation and understanding between the two groups. This concept can be translated to my own experience in applying ML to drug design. Having at least a basic knowledge of chemistry helps me to understand the data I'm working with, evaluate the reliability of the obtained outputs and communicate with chemistry experts, allowing me to interpret results and resolve problems more easily. Without this minimum background, I would lack that "shared language" necessary to communicate and understand chemists. Similarly, software engineers' understanding of the type of data used in the ML model, as well as its basic functioning may help them to properly integrate the model in the larger system [29]. However, while a shared understanding between data scientists and software engineers is useful for an effective collaboration, the increasing complexity of the tasks, the growing amount of advance knowledge available, and the competition for the positions translate into a need

of expertise. Therefore, in my view, the roles will not merge but remain distinct: people will specialize in a niche while also acquiring enough understanding of the responsibilities of the other figure, facilitating a smoother communication between the two. For example, Kim et al. [30] analyzed and categorized data scientists at Microsoft into clusters based on the different tasks they perform daily. They identified a group called *moonlighters*, who integrate both data science and engineering roles. They also distinguished other classes of data scientists, each focusing on specific tasks associated with data science.

5 Paper analysis

Paper 1: Generating and Verifying Synthetic Datasets with Requirements Engineering

Core idea(s) of the paper: As diffusion models scale well with model and dataset size, the paper propose to tackle the problem of data scarcity by generating synthetic datasets (with diffusion models) that however must match predefined requirements. To check those requirements, the authors suggest to use a verification workflow that aims to ensure a good trade-off between generation quality and requirement satisfaction. In their pipeline, they employ YOLOv8 (a well known model for object detection) to verify if the objects in the prompt are present in the generated images. They also manually inspected the results with human evaluation. The method is promising, as they were able to generate and verify some high-quality images that satisfy the requirements. However, the limitations are also clear: the generation quality with diffusion models is still not perfect, the automated verification of slightly blurred images is still challenging for the object detection model and, the current state of the art detection models are not yet able to detect all possible objects or tend to misclassify them. Therefore, the verification model is a bottleneck of the entire pipeline.

Relation to my research: The focus of my research is on 3D molecular generation of small molecules using diffusion models. Therefore, I work with a different type of data from the one used in the paper. When working on molecular generation, the aim is to produce valid and stable molecules in low energy conformations. Even when the task is more challenging, because for example you have to generate molecules that bind to a specific target or have specific properties (e.g., solubility, lipophilicity, toxicity), it is possible to use MD simulations/docking. Although computationally expensive and not always fully accurate, this approach is generally more straightforward than evaluating images. The generated images are indeed easy to evaluate just by visual inspection on MNIST, but the automated evaluation of the generation quality is not trivial, especially for more complex images. I am well aware of this because I am quite used to training diffusion models on images before testing the method on molecules, as this reduces the architectural constraints and the computational resources needed for work on small sets like CIFAR-10 [31] or MNIST [32].

AI-intensive project for a proposal: I think that the idea of generating synthetic data with diffusion models is very interesting, especially to counterbalance data scarcity, even if I am concerned about the automated evaluation of the generated data quality, since none of the current captioning or object detection models are perfect. Still, I would propose to apply this idea to the generation of molecular structures with diffusion models, where the requirements could include, for example, the presence of specific functional groups or substructures. These aspects can be automatically valuated by RDKiT and are crucial for chemists working on the design of new molecules. For targeted generation, I would still use docking/MD simulations to evaluate, for instance, the binding affinity of the generated molecules to the target and the human-in-the-loop approach. In addition, I would also work on developing standardized evaluation approaches and data sheets for reporting [21].

Long-term AI-engineering tweak: I think that building foundations to evaluate synthetic data generation is as relevant as developing new models. For instance, in drug discovery, not all the molecules generated by the model are really useful, even if valid, novel and stable. They may indeed not meet the parameters of drug-likeness or not synthetizable. Moreover, building well-defined

benchmarks to evaluate these models is crucial to compare different methods and lead the field in the right direction. One of the aspects I am working on is conformational diversity metrics, trying to answer questions such as: "How diverse are the conformations generated for a molecule?" and "Is this diffusion model better than others in generating different conformational states?". These are still open questions in the field, and I believe that building solid foundations to evaluate these models is as important as developing new architectures. A currently established strategy exploits QM-based software to obtain minimal energy states for a given molecular structure and then compares the generated conformations to these reference states. However, this approach is very expensive and not always feasible when using large sets of molecules. Therefore, in my research project I work on the development of a more efficient strategy for evaluating the conformational diversity of a set of conformers. Another important aspect, especially for targeted ligand generation within a protein pocket, is to evaluate whether the generated molecules are stable in the pocket and maintain key interactions with the protein. This can be tested with MD simulations in combination with human expertise. However, also this is computationally expensive and therefore not optimal in the long term. In my opinion, if at a certain point generative models will be able to generate molecules with the desired properties (e.g., binding affinity, solubility, lipophilicity, toxicity) for a specific protein targets, then it will be important also to have intuitive interfaces that allow medicinal chemists to explore the samples generated by these models and evaluate them following a human-in-the-loop approach.

Paper 2: Investigating the Impact of Solid Design Principles on Machine Learning Code Understanding

Core idea(s) of the paper: The authors of the paper claim that it is useful for data scientists to adopt the SOLID design principles from software engineering when writing ML code. The resulting code is cleaner, more readable and, therefore, easier to understand for other developers who may need to manage it. The code becomes more maintainable, reusable, modular, and flexible compared to code that doesn't follow the SOLID principles. This is important for the engineering of AI systems because other data scientists or software engineers can modify, update, extend or transfer the code with less effort, in shorter time, and without the need to drastically change the source code. This means a more maintainable and sustainable ML code.

Relation to my research: The application of SOLID principles would be useful in my research at different levels. In my project, I had to use pieces of code already written by other members of my department and adapt them to my own work. This required a significant effort because I had to modify different parts of the code to resolve errors caused by breakages generated in unrelated sections. A more modular code with low coupling would have made easier to integrate and reuse the code developed by others in my own model. Moreover, the generative model for molecular design that I am working on is intended to be used by researchers working in drug discovery, who may need to modify the code to adapt it to their scope. By applying SOLID principles to my code, I can facilitate maintainability and understandability, making it easier for others to use and extend it. Additionally, I am planning to open-source the code along with the publication of the paper, and therefore the more usable and readable the code is, the more likely it is that other researchers will be able to use it.

AI-intensive project for a proposal: I propose a larger AI-intensive project for developing an interface for models that generates molecules. This package would be useful for researchers working in drug discovery, who aim to design molecules, as every one can integrate his/her model and use a standardized process to evaluate it. In my current research, I am developing a generative model for molecular design, without considering target-specific (protein) information. In this way my interface could serve as a starting point to push forward the models in the field, and as new metrics are proposed we continuously update them. This can be beginning for also more complex applications that integrates target-based constraints in order to generate molecules optimized with respect to the target structure and properties. By applying the SOLID design principles, the ML code may become more modular, maintainable and adaptable, enabling other researchers to modify and extend the code according to

their needs. For example, the physical-chemical properties of the molecule may be influenced by the localization of the target. The code developed in accordance with the SOLID principles could make the tool more easily customized and reusable by other research groups (under MIT licence!!).

Long-term AI-engineering tweak: The paper highlights the difficulty of writing ML code that is clean and understandable also by people other than the original developer and proposes to address this problem by adopting the SOLID design principles. Therefore, for what concern my own project, the first step to tackle the problem of limited comprehensibility will be to ensure that my code follows these principles. This will support modularity, maintainability and flexibility of the code. Additionally, to make the code more easily accessible and usable for users and other developers, it could be useful to provide clear documentation with the most relevant information, organized in a way that makes it easy to examine. Also adding comments directly in the code to clarify the role of each module and to briefly explain the most complex parts of the code can facilitate its management. Finally, it may be helpful to introduce some example tests to use both as a demonstration of how the system works and to check how the output changes after modifying the original code. I believe that by applying all these considerations to my code, it will result more accessible and understandable to others.

6 Research Ethics & Synthesis Reflection

To find the papers, I initially conducted a search on Google Scholar using keywords such as "Software Engineering for AI", "Software Engineering and Data Science", and "Differences between Software Engineering and Data Science ". I prioritized papers that were published more recently and the most cited ones. During this initial screening I noticed that only a small fraction of the articles were really related to the topic I was looking for. Therefore, I then started to explore the papers cited in the most significant paper I had already identified and read, allowing me to identify additional works on the topic. The talk by Andrej Karpathy appeared instead on my YouTube homepage a few weeks ago and, since I found not only very interesting but also relevant for the discussion, I included it. To ensure originality, I based my answers primarily on my own understanding and reflections on the topic. Since I had already some prior knowledge of the differences between data scientists and software engineers, I started from the idea that I had based on what I already knew of the topic, and then I corroborated, expanded and elaborates my initial opinion based on what I found in the literature. In this way, the papers I consulted were used not to replace my reasoning but rather to refine and contextualize my opinion. To acknowledge the authors, whose works I relied on in order to write and elaborate my answers, I cited their papers in my text.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[5] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts, 2022.

[6] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions, 2023.

[7] Ross Irwin, Alessandro Tibo, Jon Paul Janet, and Simon Olsson. Efficient 3d molecular generation with flow matching and scale optimal transport. *CoRR*, abs/2406.07266, 2024.

[8] Josep Arús-Pous, Simon Viet Johansson, Oleksii Prykhodko, Esben Jannik Bjerrum, Christian Tyrchan, Jean-Louis Reymond, Hongming Chen, and Ola Engkvist. Randomized smiles strings improve the quality of molecular generative models. *Journal of Cheminformatics*, 11(1), November 2019.

[9] Benoit Baillif, Jason Cole, Patrick McCabe, and Andreas Bender. Deep generative models for 3d molecular structure. *Current Opinion in Structural Biology*, 80:102566, 2023.

[10] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, and J. P. Overington. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107, September 2011.

[11] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. Pubchem in 2021: new data content and improved web interfaces. *Nucleic Acids Research*, 49(D1):D1388–D1395, November 2020.

[12] Jiazhen He, Alessandro Tibo, Jon Paul Janet, Eva Nittinger, Christian Tyrchan, Werngard Czechtizky, and Ola Engkvist. Evaluation of reinforcement learning in transformer-based molecular design. *Journal of Cheminformatics*, 16(1), August 2024.

[13] Tuan Le, Julian Cremer, Frank Noe, Djork-Arné Clevert, and Kristof T Schütt. Navigating the design space of equivariant diffusion-based generative models for de novo 3d molecule generation. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[15] Siyi Gu, Minkai Xu, Alexander S. Powers, Weili Nie, Tomas Geffner, Karsten Kreis, Jure Leskovec, Arash Vahdat, and Stefano Ermon. Aligning target-aware molecule diffusion models with exact energy optimization. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[16] Paolo Tosco, Nikolaus Stiefl, and Gregory Landrum. Bringing the mmff force field to the rdkit: implementation and validation. *Journal of Cheminformatics*, 6(1), July 2014.

[17] Artem Moskalev, Mangal Prakash, Junjie Xu, Tianyu Cui, Rui Liao, and Tommaso Mansi. Geometric hyena networks for large-scale equivariant learning. In *Forty-second International Conference on Machine Learning*, 2025.

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[19] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and improving the training dynamics of diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 24174–24184. IEEE, 2024.

[20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[21] Diego Calanzone, Andrea Coppari, Riccardo Tedoldi, Giulia Olivato, and Carlo Casonato. An open source perspective on AI and alignment with the EU AI act. In Gabriel Pedroza, Xiaowei Huang, Xin Cynthia Chen, Andreas Theodorou, Huáscar Espinoza, Nikolaos Matragkas, José Hernández-Orallo, Mauricio Castillo-Effen, Richard Mallah, John A. McDermid, David M. Bossens, Bettina Könighofer, Sebastian Tschiatschek, and Anqi Liu, editors, *Proceedings of the IJCAI-23 Joint Workshop on Artificial Intelligence Safety and Safe Reinforcement Learning (AISafety-SafeRL 2023) co-located with the 32nd International Joint Conference on Artificial Intelligence(IJCAI2023), Macau, China, August 21-22, 2023*, volume 3505 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023.

[22] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns*. Addison Wesley, Boston, MA, October 1994.

[23] Hao Qian, Wenjing Huang, Shikui Tu, and Lei Xu. Kgdiff: towards explainable target-aware molecule generation with knowledge guidance. *Briefings in Bioinformatics*, 25(1), November 2023.

[24] Iqbal H. Sarker. Data science and analytics: An overview from data-driven smart computing, decision-making and applications perspective. *SN Computer Science*, 2(5), July 2021.

[25] Christophe Ley and Stéphane P. A. Bordas. What makes data science different? a discussion involving statistics2.0 and computational sciences. *International Journal of Data Science and Analytics*, 6(3):167–175, February 2018.

[26] Paul Luo Li, Amy J. Ko, and Jiamin Zhu. What makes a great software engineer? In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. IEEE, May 2015.

[27] Iftekhar Ahmed, Aldeida Aleti, Haipeng Cai, Alexander Chatzigeorgiou, Pinjia He, Xing Hu, Mauro Pezzè, Denys Poshyvanyk, and Xin Xia. Artificial intelligence for software engineering: The journey so far and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–27, May 2025.

[28] Grace A. Lewis, Stephany Bellomo, and Ipek Ozkaya. Characterizing and detecting mismatch in machine-learning-enabled systems. In *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, page 133–140. IEEE, May 2021.

[29] Gabriel Busquim, Hugo Villamizar, Maria Julia Lima, and Marcos Kalinowski. *On the Interaction Between Software Engineers and Data Scientists When Building Machine Learning-Enabled Systems*, page 55–75. Springer Nature Switzerland, 2024.

[30] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering*, 44(11):1024–1038, November 2018.

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.(2009), 2009.

[32] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.