

ANR, Appel à Projets Générique (AAPG 2021)

AI₄CODE Project (ANR-21-CE25-0006)

Deliverables D4.1 and D4.2

How we learned deep learning

What we learned from the projet and the road ahead

Final Report

Editor:	Raphaël Le Bidan (IMT Atlantique)
Deliverable nature:	Public
Due date:	October 31, 2025
Delivery date:	October 24, 2025
Version:	1.0
Total number of pages:	12 pages
Keywords:	

Abstract

This report reflect upon the main outcomes and lessons learned from the research work carried out in WP2 and WP3. It also includes a list of bibliographic references and online resources that proved especially valuable in developing our understanding and expertise in deep learning.

List of Authors

Partner	Author
LAB-STICC/IMTA	Raphaël Le Bidan (raphael.lebidan@imt-atlantique.fr) Elsa Dupraz (elsa.dupraz@imt-atlantique.fr) Charbel Abdel-Nour (charbel.abdelnour@imt-atlantique.fr)
CEA-LETI	Valentin Savin (valetin.savin@cea.fr) Valérien Mannoni (valerien.mannoni@cea.fr)
IMS/INPB	Christophe Jégo (christophe.jego@ims-bordeaux.fr) Camille Leroux (camille.leroux@ims-bordeaux.fr) Romain Tajan (romain.tajan@ims-bordeaux.fr)
IRIT/INP-ENSEEIH	Charly Poulliat (charly.poulliat@enseeiht.fr)
ETIS/CYU	Inbar Fijalkow (inbar.fijalkow@ensea.fr)
Lab-STICC/UBS	Emmanuel Boutillon (emmanuel.boutillon@univ-ubs.fr)

Contents

Introduction	4
1 Learning deep learning	5
1.1 Books	5
1.2 Online courses	6
1.3 Papers and online articles	7
2 Learning for channel coding: Lessons learned and the road ahead	8
Bibliography	12

Introduction

Work Package 4 (WP4) of the AI4CODE project was devoted to the sharing of knowledge and operational experience on machine and deep learning across WPs and Tasks. This WP also aimed at reflecting upon the outcomes from WP2 and WP3 to develop a general expertise and critical thinking on the pros and cons of ML in coding. The Gantt chart of WP4 is shown in Fig. 1.

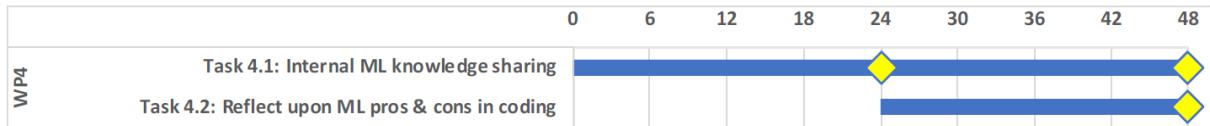


Figure 1: Gantt diagram of WP4

This report is the final deliverable for **Tasks 4.1** and **Task 4.2** of WP4. It is organized into two sections. Section 1 presents some of the most valuable resources that supported our development of deep learning expertise, including materials on state-of-the-art model architectures such as transformers. Section 2 summarizes the key lessons learned from this project, discussing both the potential and the current limitations of machine learning methods applied to channel coding problems.

1 Learning deep learning

Deep learning is a research field that evolves at an extraordinarily rapid pace. ChatGPT is only three years old¹, yet large language models have already become an integral part of our personal and professional lives. By the time new results are presented at conferences, they often already belong to the past. Staying up to date with the latest developments in the field requires following daily updates on platforms such as X (formerly Twitter) and arXiv.

Despite this constant whirlwind of innovation, very few ideas truly stand the test of time. The main neural network architectures—MLP, RNN, CNN/ResNet, and Transformer—have only evolved marginally over the past eight years. Most of the progress has focused on making these models faster and more efficient, in line with improvements in GPU computing power, thereby enabling larger models with greater learning and processing capacity. Perhaps the most notable paradigm shift has been the emergence of hybrid, multi-stage training procedures that combine supervised learning with unsupervised learning and various forms of reinforcement learning.

This is good news for those seeking to develop expertise in deep learning. Despite the field's rapid evolution, it remains possible to acquire solid and lasting skills. However, this demands careful selection of trustworthy sources amid the overwhelming amount of online information, as well as extensive hands-on practice, since deep learning is still a largely experimental science.

This section compiles the bibliographic references, books, research papers, blog posts, videos, and online courses that have been most valuable in developing our understanding of deep learning. As researchers coming from other disciplines, identifying these resources was not always straightforward. We share this list in the hope that it will serve as a useful guide for others embarking on a similar learning path.

1.1 Books

The 2016 Deep Learning book from Ian Goodfellow, Yoshua Bengio, and Aaron Courville is still widely cited and remains of interest, particularly for the historical perspective it provides. However, it is no longer the most suitable reference for those wishing to train or update their knowledge in deep learning. Here are a few more compelling options.

Book	Description
Deep Learning: Foundations and Concepts	Christopher Bishop <i>Pattern recognition and machine learning</i> (2006) has been a longstanding classic in the field. This new textbook published in 2024 is a revised edition that include new chapters relating to contemporary architectures and techniques such as transformers, normalizing flows, diffusion models, or graph neural networks.
Reinforcement Learning: An Introduction	The 2020 second edition of Richard Sutton and Andrew Barto still remain the classic textbook and the gold reference for any researcher in reinforcement learning.

¹The first very first public release of ChatGPT was launched in november 2022.

Book	Description
Deep learning for coders with Fastai and PyTorch	A 2020 book by Jeremy Howard and Sylvain Gugger. Yes, this book is 5-year old, which may sound like an eternity in the deep learning timeline. But most of its content still holds today. This is the companion book to the Fast.ai online course cited in the next sub-section. Both are complementary. If you just need one book to get you started, this is the one (or the next below).
Deep learning with Python	Another classic, by François Chollet and Matthew Watson, now in its 3rd edition (2025). This book, like the previous one, teaches good practices and helps readers become quickly operational. It also takes the time to explain the main architectures — why they were designed in a certain way and why they perform so well. It is an essential read, perfectly complementing the previous reference.

1.2 Online courses

Several high-quality online courses are available that not only provide in-depth coverage of the basics but also provide an efficient way to acquire solid hands-on experience in deep learning.

Online course	Description
Andrej Karpathy YouTube channel	Karpathy is an experienced and well-recognized researcher in deep learning, as well as an outstanding educator. His video series <i>Neural Networks: From Zero to Hero</i> is a widely regarded reference for anyone starting in the field. As are his introductory videos on large language models.
Fast.ai Practical Deep Learning for Coders	Jeremy Howard's Practical deep learning for coders course Part 1 and Part 2 is a free course designed for people with some coding experience who want to learn how to apply deep learning and machine learning to practical problems. These worldwide-recognized lectures have proven to be a valuable resource for acquiring good practices in deep learning and, more importantly, for understanding what truly works — and why — with references to several foundational research papers. This course has been instrumental in building the skills for the project.

1.3 Papers and online articles

Instead of compiling an inevitably incomplete and loosely organized list of papers, we chose to reference a few curated reading lists that collect key articles and situate their contributions within context — in terms of both problem category and scientific significance. We wish we had known about these websites earlier.

Online resource	Description
AI Canon	For a gentle guided tour of the evolution of AI and a survey of the most influential papers and contributions in deep learning up, the AI Canon reading list is a must. A special emphasis is placed on large language models and diffusion models.
The 2025 AI Engineer Reading List	AI Canon only covers deep learning research up to 2023. So much happened since then. The 2025 AI Engineer Reading List fills the holes in a brilliant, comprehensive way, with a practical focus, and with all modalities covered (language, vision, speech, code).
Elicit Machine Learning Reading List	Another very comprehensive and well-organized paper reading list, compiled by Elicit to help their new employees learn background in machine learning, with a focus on language models (but not only, e.g. world models and AlphaGo are covered as well).
Eugene Yan's language modeling reading list	People interested in transformer architectures and LLMs may find this reading list very helpful.
The Machine Learning Engineering Open Book	Wants to get serious about training or serving large models at scale? This online book by Stas Bekman is a rich source of information, intended primarily for experienced engineers and researchers.

2 Learning for channel coding: Lessons learned and the road ahead

The AI4CODE project was built around four inter-related objectives:

1. Explore how machine learning (ML) can contribute to improving the state of the art (SoA) in decoding forward-error correction (FEC) codes [WP3]
2. Investigate how ML can improve current knowledge and practice in FEC code design [WP2]
3. Learn from the machine [WP2 & WP3]
4. Develop a general expertise and critical thinking on ML algorithms and their applications to coding theory and practice [WP4].

The present section tackles the last one, objective 4, by reflecting upon the work carried out in WP2 and WP3 and its outcomes, in relation with research objectives 1–3.

Preamble: Although not stated explicitly in the above-mentioned objectives, the first and foremost underlying goal of AI4CODE was, for all research teams involved in the project, to build the required knowledge and practical skills in machine learning deep learning, so as to be in position to apply those techniques in a meaningful manner to channel coding problems. Four years later it is fair to acknowledge that this objective has been met. Yet the harsh truth is that learning takes time. Furthermore, the straightforward application of leading architectures from computer vision, natural language processing, or time-series modeling to error-correcting code design or decoding problems typically fails to produce satisfactory outcomes. As a result, several of the research directions initially proposed in the project could not be fully explored — notably, the design of short LDPC codes suitable for belief propagation decoding and the automatic adaptation of the decoder to unknown channels. Conversely, we investigated topics not originally included in the project plan, such as applying machine learning to multi-user detection in multi-antenna (MIMO) systems. Quoting the first sentence of Haruki Murakami’s book *What I Talk About When I Talk About Running*, “Pain is inevitable but suffering is optional”. Having access to the bibliographic resources listed in the Section 1 from the very beginning would have saved us significant time. Let us now review some of the main lessons learned from the investigations carried out in WP2 and WP3.

Learning to code: The work conducted in this project on the application of machine learning to code construction—specifically for polar codes—has primarily focused on supervised approaches. While certain learned codes demonstrate measurable performance gains, these improvements remain modest. This limitation stems from the supervised learning process itself. Supervised learning requires explicit targets. Accordingly, models for FEC code construction or optimization needs to be trained on a predefined, inherently imperfect set of codes; if better codes were already known, there would be no need to search for them. Consequently, such models tend to generalize poorly beyond their training set. Such an approach is not creative enough. More exploratory directions, including reinforcement learning and unsupervised learning, appear considerably more promising. A notable example is a recent result involving several members of the AI4CODE group, but achieved within another research project: the construction of flexible, 5G-compatible quasi-cyclic LDPC codes through gradient-based optimization of the parity-check matrix solely driven by decoding performance [1]. It is somewhat surprising that

FEC code construction has not yet undergone the generative AI transformation. However, we expect such a shift to be made soon, particularly with the rapid progress in discrete diffusion models.

Learning to decode: One of the project’s primary objectives (Objective 1) was to show that machine learning can help decode short codes with performance approaching that of optimal Maximum Likelihood Decoding (MLD), while achieving greater efficiency than conventional methods such as Ordered Statistics Decoding (OSD), Belief Propagation (BP), or Chase-2 decoding. The initial strategy adopted a *model-based* paradigm, enhancing existing decoding algorithms with AI components. Two distinct approaches were investigated, both applied to short LDPC codes. In each case, the outcome was positive: learning-based augmentation enabled a reduction in average decoding complexity at equivalent performance levels compared to expert algorithms. This observation raised the question of whether building upon existing expert algorithms introduces an inductive bias strong enough to constrain the benefits that AI can provide. To examine this hypothesis, we decided to explore *model-free* approaches, where decoding is carried out entirely by a learned model without relying on algorithmic priors. This was not initially planned in the original scientific plan. The key finding from this investigation is that, while MLD-level performance can indeed be approached, sometimes very closely, it comes at the expense of a large number of trainable parameters. Therefore, again, the computational cost remains significantly higher than that of practical expert decoders, and of *model-based decoders* as well. In summary, machine learning can indeed enhance the decoding of short codes, but to date, with the approaches that we have explored, the computational cost remains too high for practical deployment.

Model-free vs model-based approaches: Model-based decoders produce simpler architectures, yet their performance appear to be somewhat limited by that of the expert algorithms they augment. In contrast, model-free decoders exhibit greater potential but face structural challenges that limit both their efficiency and scalability—that is, their ability to decode longer codes. In both cases, we believe that substantial foundational work is still required to develop neural architectures intrinsically adapted to the channel decoding problem. It seems in particular that embedding more expert knowledge into the architecture itself could help, such as equivariance to codeword translation. Our work on multi-user MIMO detection using message-passing graph neural networks indeed suggests that integrating domain knowledge can lead to simpler, more effective architectures. It appears, therefore, that the field of channel coding is still far from facing the “bitter lesson” [2] learned in other areas of deep learning.

Learning from deep learning best practice: An unexpected finding in our decoding studies was that insufficient attention was paid to both the training data and training methodology. Channel decoding differs significantly from other deep learning application domains, such as computer vision, in that the training of decoder models relies entirely on synthetic data. As much data as needed can be generated through Monte Carlo simulations. Consequently, training is almost always performed on data generated on demand. While this may appear highly advantageous — notably because it greatly reduces the risk of overfitting — on-demand data generation also presents certain drawbacks. First, it tends to slow down convergence, since the model is constantly exposed to new data. This typically leads to longer training processes and larger overall data volumes. Moreover, we have shown that this approach does not always allow training on the most relevant targets, which can ultimately degrade model performance. One of the outcomes of the project was to demonstrate that, by adopting standard deep learning best practices — namely the use of fixed training datasets — it is possible to achieve significantly better performance with far less data, thanks to full control over the dataset composition (in particular, the selection of training samples). Other techniques that have demonstrated their

effectiveness in deep learning, such as data augmentation, can likewise be successfully applied in this context. The importance of good training data has too often been overlooked in previous works.

Learning from the machine: One of the central goal of the AI4CODE project was not to replace expert knowledge by black-box algorithms, but ultimately to learn from the machine. This indeed happened but not in the way initially expected. Inspection of the polar codes designed with the help of learning, or of the trained decoder models did not provide us with theoretical hindsight that could ultimately translate into improved code design tools or decoders. The situation was quite different during the search for CCSK sequences robust to truncation. The analysis of the best solutions discovered through evolutionary algorithms gradually revealed the existence of an underlying mathematical structure. Exploiting this structure then made it possible to construct optimal sequences, called C_4 -sequences [3], for the problem under consideration, which were subsequently patented. Another interesting and unforeseen result was the rediscovery of permutation decoding when we attempted to implement TTA (test-time augmentation) on our models — a well-known deep learning practice used to improve model performance at inference. This observation suggests that, beyond the models themselves, deep learning may have further contributions to make to channel coding. For instance, operations such as dropout used to reduce overfitting share similarities with the erasure operation performed in a Self-Corrected Min-Sum decoder [4], which can be regarded as as form of decoder regularization technique. It is therefore reasonable to think that some concepts or practices initially developed for deep learning could inspire new algorithms for channel coding.

Perspectives: At the end of this project, we believe it is important to emphasize that deep learning still holds considerable potential for channel coding. In particular, its application to the design of efficient error-correcting codes remains largely underexplored, except for feedback channels — a topic that deep learning has truly revitalized and in which steady progress has been made over the past four years (see e.g. [5] for an example recent highlight). The relevance of deep learning for decoding remains more uncertain, due to the current lack of suitable architectures capable of matching expert-designed algorithms in terms of complexity, or of significantly outperforming them in terms of performance. Such architectures have yet to be discovered. Nevertheless, we consider this to be a highly promising research direction, albeit with substantial uncertainty as to when a genuine breakthrough may occur. The question of interpretability in the decision-making process will, however, remain a key challenge. In the meantime, large language models (LLM) have undergone spectacular progress. We now have models capable of performing highly complex mathematical reasoning and excelling at computer code generation. Such tools open up new ways of conducting research, particularly when it comes to discovering new algorithms — for instance, to design more efficient FEC codes or to approach optimal MLD decoding with fewer computations than the best algorithms currently known. The solution explored in the AI4CODE project consisted either in augmenting existing heuristics with an AI-based layer or in replacing them entirely with a black-box model. We have seen that, to date, neither of these approaches is fully satisfactory, as both raise issues of interpretability. All these observations have therefore led us to submit a new PRC proposal to the ANR AAPG 2026 call, entitled Televolve. The Televolve project promotes a different path: searching for better, explicit algorithms, in an automated way, by leveraging the combination of generative AI and program synthesis. Program synthesis consists of automatically generating programs by using a search algorithm, for example an evolutionary algorithm, to explore a large space of candidate programs. The search stops when a program is found that matches the required specifications. Compared with black-box models, program synthesis produces human-readable code, offering significant advantages in terms of interpretability, debuggability, verifiability, and ease of deployment. Here, deep learning is used in combination with program synthesis either

to make the search more efficient, as done for example by the LLM at the core of evolutionary coding agents like AlphaEvolve from Google DeepMind [6], or to supplement the search with specialized sub-modules that can learn from the data, for instance the value and policy networks driving a deep reinforcement learning loop. It is part of Televolve’s ambition to demonstrate — through concrete and compelling examples — that automated algorithm discovery, empowered by generative AI, can accelerate scientific progress in research areas such as channel coding, where algorithmic design plays a central role.

Bibliography

- [1] E. Bodji, B. Jahan, C. Jégo, and E. Boutillon, “Quasi-cyclic gradient learning : une méthode d’optimisation des codes QC-LDPC de la 5G,” in *28ième édition du colloque GRETSI*, Aug. 2025.
- [2] R. Sutton, “The bitter lesson,” 2019, available online: <http://www.incompleteideas.net>.
- [3] E. Boutillon, “Constellations cross circular auto-correlation c4-sequences,” *IEEE Transactions on Communications*, vol. 72, no. 12, pp. 7664–7673, 2024.
- [4] V. Savin, “Self-corrected min-sum decoding of ldpc codes,” in *2008 IEEE International Symposium on Information Theory*. IEEE, 2008, pp. 146–150.
- [5] W. Lai, Y. Shao, Y. Ding, and D. Gündüz, “Variable-length feedback codes via deep learning,” in *ICC 2025-IEEE International Conference on Communications*. IEEE, 2025, pp. 3864–3869.
- [6] A. Novikov *et al.*, “AlphaEvolve: A coding agent for scientific and algorithmic discovery,” *Google DeepMind white paper*, May 2025.