COM-304 nano4M Project Guidelines

https://github.com/EPFL-VILAB/com-304-4M-project
EPFL, Lausanne, Switzerland
Spring 2025

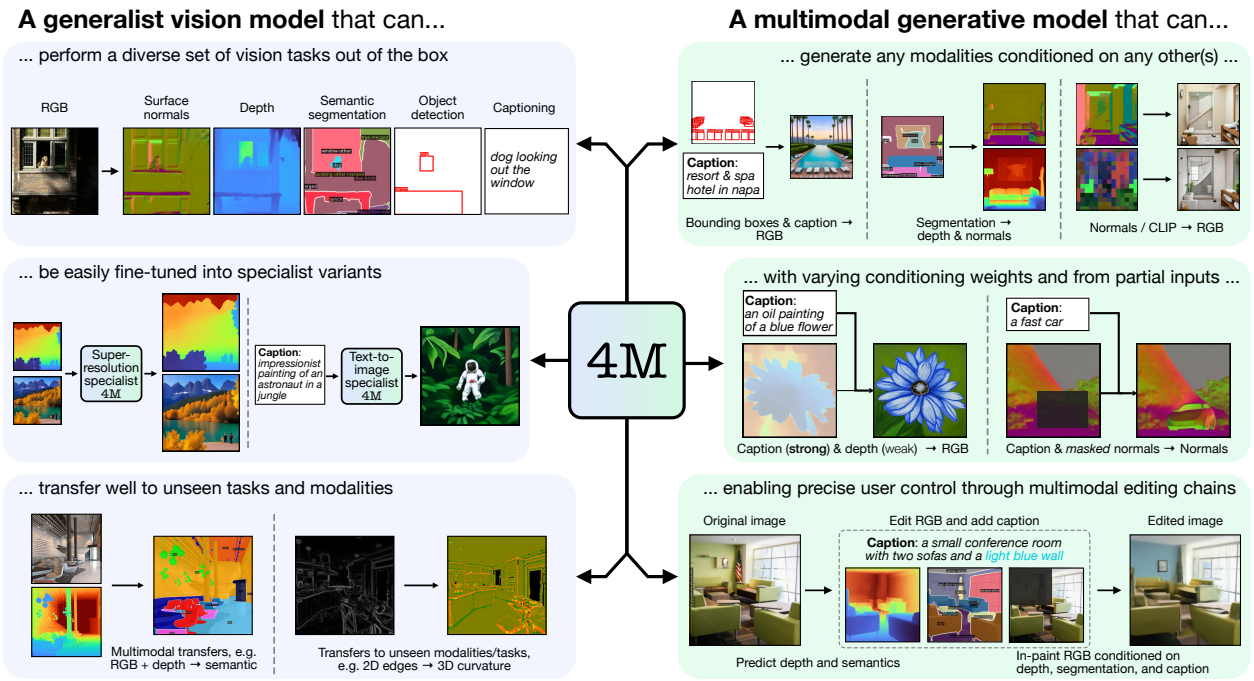

Figure 1: `nano4M`: In your project you will implement and train a minimal version of a massively multimodal masked model, inspired by 4M [7, 1] and `nanoGPT` [4]. You will learn what makes current large language and multimodal models work from a practical standpoint.

1 Introduction

The general goal of the project is to understand the building blocks that make up modern (multimodal) large language models (LLMs) like GPT-4o, and assemble them to build a minimalist multimodal model we call `nano4M`, inspired by 4M [7, 1]. Although simplified, the principles taught in this project are directly applicable to training real-world large multimodal models.

The project consists of two parts:

- **Building `nano4M`:** In the first half of the project, you will build a code base for training massively multimodal models, like 4M. First, you will be guided to implement and train an LLM from scratch, learn how to train Transformer-based image generation models, and finally make them multimodal. See section 2 for more information.
- **Extensions:** You will implement a set of extensions of your liking. We will propose a list of extensions of various difficulties, but you can also propose your own. Each extension will be worth a certain number of point, depending on its difficulty and implementation effort, and you will need to complete extensions worth a total of 100 points. See section 3 for more information.

2 Building nano4M

In this first part of the project, you will build nano4M, a minmal version of 4M[7, 1], inspired by nanoGPT [4]. You will approach this goal in several stages. Detailed instructions will be handed out as the project unfolds.

Building a large language model. Your first task will be to implement an LLM from scratch and train it on a simple dataset to perform text generation. To this end, you will build a causal ("decoder-only") Transformer architecture, and the necessary infrastructure to train it on a text dataset. You will also learn how to perform autoregressive inference.

Building an image generation model. Now that you built and trained an LLM, we will adapt the codebase to support autoregressive [8, 9] and masked image generation [3, 2, 6]. You will learn how to build an "encoder-only" Transformer, about image generation with discrete token-based Transformer models, masked training objectives, and alternative decoding schemes to standard autoregressive generation.

Building a multimodal model. Finally, we will combine what we have learned into a multimodal model, inspired by 4M. This model will be trained on an aligned multimodal dataset with a multimodal masking objective, which will result in "any-to-any" generation capabilities like the ones shown in fig. 1 (e.g., chained multimodal generation, performing classical vision tasks out-of-the-box, in-painting, etc.). We will provide you pre-trained multimodal tokenizers, as well as the pre-tokenize dataset. You will learn how to implement "Encoder-Decoder" Transformer architectures and train them with a cross-modal masked prediction objective. The codebase implemented at the end of this will be the base for implementing a number of exciting extensions, as shown in the next section.

3 Extension

You will choose and/or propose a set of extensions to the basic nano4M to get the full mark. Each extension will count for a certain number of points, and your task for the second half of the project is to complete extensions with a total summed worth of at least 100 points.

Beyond the proposed set of extensions, we encourage you to come up with novel ideas. The more creative the extension, the better. Creativity can appear in various forms, e.g., in terms of problem selection, solution formulation, implementation, experimentation, or demonstration. We will cast a wide net and will be flexible with any meaningful efforts, so focus your energy on doing something interesting and less on worrying about grades. If you propose your own extension, we will ask you to propose how many points you think it is worth, and discuss with the TAs.

Below we already provide an initial selection of extensions to give you an idea of possible directions you will work on in the second project half. A complete list with detailed descriptions and amount of points awarded will be distributed before the project proposals.

- *Add a new modality / task:* Extend nano4M to additional data types (video, speech, audio, 3D, etc.). Investigate how multimodal alignment, reconstruction, and generation evolve when integrating these diverse modalities into the model.
- *Tokenization:* Current off-the-shelf tokenizers might not yield optimal reconstruction performance for certain modalities. Instead, train specialized tokenizers and compare reconstruction, generation, and alignment performance against the default tokenizer.
- *Inference strategy:* Compare iterative decoding vs. one-pass decoding. Experiment with advanced sampling methods (top-k, nucleus, etc.) and explore controlled generation. Measure trade-offs in speed, reconstruction quality, and sample diversity.

- *Zero-shot & transfer evaluation:* Assess the model's ability to handle unseen tasks or new modalities at inference time. Evaluate transfer learning performance when shifting between domains or extending to previously untrained modalities.
- `nano4M` *speed run:* Inspired by `nanoGPT` speedruns [5], how fast (in terms of tokens seen or wall-clock time) can you train `nano4M` to a pre-determined validation loss?
- *Architecture extensions:* Move beyond the standard encoder-decoder transformer, or even discrete tokenization. Investigate alternative architecture and training choices to improve efficiency, scalability, and representation quality in multimodal tasks.
- *Data & augmentation:* Experiment with different masking strategies in the dataloader. Scale to larger training sets (e.g., more RGB images) and leverage strong off-the-shelf networks for pseudo labeling other modalities. Study how these augmentations affect learning stability and performance.
- *Paper (re-)implementation:* pick an exciting and relevant paper published in a top-tier conference (e.g., in CVPR, ICLR, E/ICCV, NeurIPS) and re-implement it (or part thereof). Note that not any paper will do, and it should be an *exciting* one.

4 Evaluation

Your model will be evaluated based on the quality of it's code implementation, generation results, and the performance of the proposed extensions. *More details about the specifics of the evaluations will be released shortly.*

5 Reports Instructions

During the project, you will need to submit three reports and make the final presentation of your project. Below, you can find instructions for each stage.

5.1 Project Proposal

This proposal will be focused primarily on the extensions that will be made to the model. You should assume that you have access to the LLM, image generation model and nano4M that will be developed in the first half of the project and the extensions will be built on top of those base models.

You should describe each extension you will implement in detail, both at a high level, considering the following questions:

1. **What** is the problem you want to solve?
2. **Why** is it important that this problem be solved?
3. **How** do you solve this problem?

and at a lower level, considering implementation specifics such as:

- Will you need extra data for the model to generalize well with this extension?
- What model architectures will/won't this extension work with?
- How much extra compute might you need to make an extension work?

- And else anything relevant to your specific extensions.

Your description of the extensions should be self-sufficient, so be as specific as possible. You also need to set reasonable goals that you are going to achieve for the progress report to ensure gradual and sufficient progress. Make sure to assess what can go wrong and discuss other options you can try. The proposal document should be **at least one, and at most two pages** long. Please use the template that we will provide on Moodle in the corresponding week.

Additional words of advice:

- Try to communicate and motivate your idea using visuals, diagrams, charts, or any other appropriate tools you see fit.
- Allocate your time well between the progress report and the final project delivery date.

5.2 Progress Report

The Progress report is intended to make sure that projects advance at a uniform and continuous pace. Please provide an **at most two-page long** report that includes, at minimum, a clear description of the following:

- The steps taken so far in implementing the project and deviations from the original proposal, together with explanations.
- A discussion about the problems you encountered, the solutions you explored, and problems that you are likely to encounter as your project progresses.
- A tentative but concrete list of the action items you plan to work on until the final report and how they relate to the project's overall goal.

Please use the template that is provided on Moodle alongside these guidelines when writing your report. One submission per group is sufficient, and submissions should be made by **the same person that submitted the project proposal**.

Apart from the written progress report, you will need to demonstrate the performance of the model as outlined in the basic task (Sec. 2). *This will be an in-person evaluation during the first exercise session after the progress report deadline. We will communicate the exact time and date later.*

5.3 Final Report Guidelines

Report: Your final report should be **4 pages maximum**, excluding references. You can use appendices without any limits on the page number, but make sure that the main material is provided in the main report. Please use the template that is provided on Moodle alongside these guidelines when writing your report. As before, one submission per group is sufficient. We also ask you to submit your **code** with proper running instructions (e.g. a ReadMe file) in a **zip file**. The ReadMe file should be self-contained: specify the packages to install, their version numbers, commands on how to run your code to reproduce your results and the file hierarchy with a description of each file. Please make sure the code is understandable, e.g. through documentation.

Please try to organize your report in the following suggested way for a better understanding.

- **Abstract:** Provide a brief description of your problem, approach, and key results.
- **Introduction:** Describe the problem you are solving, its significance (i.e. why are you solving it?), and how do you solve it. You can organize this section similar to the introduction of your proposal report while being *more concrete and specific*.
- **Related Work:** How is this problem currently solved (if solved)? Discuss the relevant works to your project and pose your approach against them. Indicate the *differences* and *similarities* between your project and these works as clearly as possible.
- **Method:** Explain your approach for solving the problem. Justify the design choices you made and mention other alternatives, if any. Make sure to include figures, diagrams, pseudo-code, etc. to strengthen your case. It is important that your method is explained in an understandable way for a fair evaluation.
- **Experiments:** Discuss your experimental setup in detail. Explain your baselines and justify why you picked them. Support your results with *quantitative and qualitative evaluations* comparing your method to these baselines (e.g. include tables for performance metrics and qualitative figures.). If relevant, perform ablation studies to provide further insight into the inner workings of your method.

  If your project *did not work as expected*; and you instead managed to systematically invalidate an apriori sensible hypothesis; that is also a perfectly meaningful contribution. If this is the case, provide a detailed and sensible analysis that identifies the main modes of failure of your original hypothesis, discusses their potential reasons, and distills what one can learn from them.

  The projects will not be regarded as successful only if they "work"; any project that extracts interesting insights or contributes a signal towards evaluating a meaningful hypothesis will also be regarded as successful. The main evaluation criteria will be the degree of creativity, motivation, and scientific rigor with which you managed your project (e.g., asking interesting and sensible questions, forming and validating hypotheses in a systematic way using the appropriate baselines), rather then the end score you obtained on some pre-defined benchmark.
- **Conclusion and Limitations:** Provide a brief summary of and takeaways from your project. Mention the limitations of your method and how can they be solved. Also mention possible future extensions or other use cases.
- **Individual contributions:** If it is a group project, include an author contribution section explaining the role of each group member throughout the project. You can refer to this exemplary author contribution statement to give you an idea.

Additional suggestions:

- Make sure you proofread your report (or ask an external person to do it, if possible).
- Visuals (figures, tables, etc.) can be more effective at conveying information than writing. But do make sure that your visuals are helpful, e.g. include captions that describe and explain the take home message for your figures and tables, plots are understandable (e.g., with proper labels and readable font size for axes, etc).
- You can include videos as part of your results if it is relevant to your extensions. You can also provide extended image results in your appendix.

Website: In research nowadays it is common to include multiple formats of presenting your project, such as videos, posters and websites. The presentation of this project in particular benefits greatly from displaying a wider range of results in a more accessible way than possible with a report alone. As such, you will also prepare a website / blog post that can display examples of the results of your model. **This website should contain** a short summary of the key takeaways from the project, an extended presentation of what the model is capable of, and what additional capabilities the developed extensions provide the model. You may also choose to add other material such as a video or a demo embedded in the website if you think that may aid the presentation of your project.

Please ensure your website adheres to the following requirements:

- If you use an online report (as opposed to sending the offline webpage package), please use a platform where meeting the deadline can be verified, e.g. use GitHub Pages by creating a repository where the last commit is no later than the deadline.
- Put a corresponding amount of text as to what the page limit for the standard PDF report would allow, i.e. 5 pages excluding references, by using a comparable number of words.

See some examples here (from CS-503 Spring 2023), here, here, and here for inspiration.

Good luck! ✤

References

[1] Roman Bachmann, Oğuzhan Fatih Kar, David Mizrahi, Ali Garjani, Mingfei Gao, David Griffiths, Jiaming Hu, Afshin Dehghan, and Amir Zamir. 4M-21: An any-to-any vision model for tens of tasks and modalities. *Advances in Neural Information Processing Systems*, 2024.

[2] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, José Lezama, Lu Jiang, Ming Yang, Kevin P. Murphy, William T. Freeman, Michael Rubinstein, Yuanzhen Li, and Dilip Krishnan. Muse: Text-to-image generation via masked generative transformers. *ArXiv*, abs/2301.00704, 2023.

[3] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11305–11315, 2022.

[4] Andrej Karpathy. nanoGPT: The simplest, fastest repository for training/finetuning medium-sized GPTs. https://github.com/karpathy/nanoGPT, 2023. Accessed: 2025-02-17.

[5] Jordan Keller. modded-nanogpt: A modified version of nanoGPT with additional features. https://github.com/KellerJordan/modded-nanogpt, 2024. Accessed: 2025-02-17.

[6] Tianhong Li, Huiwen Chang, Shlok Kumar Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. MAGE: Masked generative encoder to unify representation learning and image synthesis. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2142–2152, 2022.

[7] David Mizrahi, Roman Bachmann, Oğuzhan Fatih Kar, Teresa Yeo, Mingfei Gao, Afshin Dehghan, and Amir Zamir. 4M: Massively multimodal masked modeling. In *Advances in Neural Information Processing Systems*, 2023.

[8] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *ArXiv*, abs/2102.12092, 2021.

[9] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.