COM-304 nano4M Project Guidelines

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

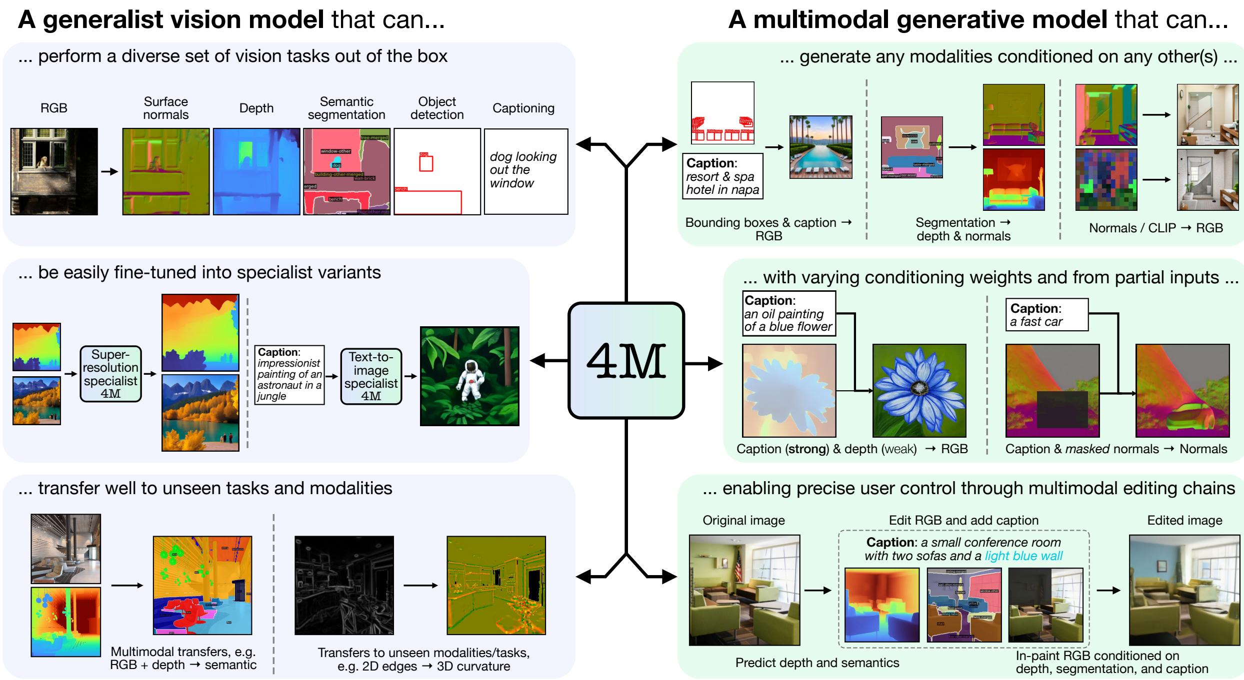

Figure 1: nano4M: In your project you will implement and train a minimal version of a massively multimodal masked model, inspired by 4M [11, 1] and nanoGPT [7]. You will learn what makes current large language and multimodal models work from a practical standpoint.

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 [11, 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[11, 1], inspired by `nanoGPT` [7]. 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 [15, 18] and masked image generation [3, 2, 9]. 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 provide a selection of potential extensions to give you an idea of possible directions you will work on in the second project half. You can use them as inspiration for your proposal. We give rough estimates for the difficulty and effort required through the number of points, but not all examples will list them. Especially when the implementation effort is subject to a broad range of possible interpretations, consult your TAs and propose a number of points that, in your best estimate, is a fair amount. The suggested number of points below may still be subject to change depending on your proposed instantiation of those extensions.

- *Add span-masking for sequence-like modalities **(20 points)**:* nano4M will be able to work with sequence-like modalities, like captions, through standard masking. In 4M [11], however, they used span-masking [14] for these modalities, which allowed them to generate masked-out regions in sequences in an autoregressive (rather than masked) manner. In this extension, you are tasked to extend nano4M to work with non-2D modalities, such as captions, using span-masking. This means you need to implement the masking logic, as well as make the decoder autoregressive for those modalities.

- *Add a new modality / task* ***(20 points):*** Extend `nano4M` to additional data types. This could be any modality you can extract from the existing ones, through known functions (e.g. edge detection), or off-the-shelf neural networks (e.g. feature maps or pseudo labels). Investigate how multimodal alignment, reconstruction, and generation evolve when integrating these diverse modalities into the model. Depending on the modality, this may be paired with implementing and training your own modality-specific tokenizer, or you can experiment with repurposing an existing tokenizer to encode your new modality.

- *Super-resolution* ***(20 points):*** Similar to adding a new modality, super-resolution can be performed as a token-to-token mapping problem, too. In this case, your task is to jointly train on low-resolution images (e.g. 224x224 pixels = 14x14 tokens), and high-resolution images (e.g. 448x448 pixels = 28x28 tokens), teaching the model to map from one to the other.

- *Tokenization* ***(20 points):*** 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* ***(5 points):*** 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.

- *Transfer evaluation* ***(20 points):*** Evaluate transfer learning performance when shifting between domains or extending to previously untrained modalities. For example, study how training on one set of modalities transfers to a held-out modality.

- *Implement classifier-free guidance* ***(20 points):*** Train your model with classifier-free guidance [5, 4] in mind, i.e. by randomly dropping the conditioning a small percentage of the time. Evaluate the inference performance under various classifier-free guidance scales.

- *Use μP* ***(20 points):*** Add μP [19] to nano4M and demonstrate if you can transfer hyperparameters swept at a small scale to a larger scale. Demonstrate basin alignment between runs using different learning rates and model widths, and compare to a non-μP baseline.

- *Architecture modifications* ***(5-10 points each):*** Perform various architecture modifications, such as replacing the MLP by SwiGLU [16], replacing the learned absolute positional embeddings by Fourier or RoPE [17], studying better weight initialization strategies, etc...

- *Add a different optimizer, e.g. muon* ***(10 points):*** Recent investigations into new optimizers, such as muon [6, 10], are yielding impressive results. How does nano4M perform if you train it with those instead of AdamW?

- `nano4M` *speed run* ***(10+ points, depending on effort):*** Inspired by `nanoGPT` speedruns [8], how fast (in terms of tokens seen or wall-clock time) can you train `nano4M` to a pre-determined validation loss? This may be paired with adding alternative optimizers, like muon, or implementing certain architecture modifications. You may also want to look into making your model compatible with `torch.compile`, and adding FlexAttention [13].

- *Paper (re-)implementation* ***(variable number of points):*** 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.

Open-ended research topics:

- *Multimodal reasoning & test-time compute* ***(variable number of points)****:* Recent advances like o1, o3, r1 have shown remarkable benefits of scaling inference compute (instead of just scaling pre-training compute). How can you extend this to multimodal models?
- *Scaling laws:* This is subject to the available compute and the breath of the investigation, but can you come up with some small-scale scaling laws for multimodal masked models? (*Note: We expect compute to be on the low end for this, and would not recommend choosing this topic, but we leave it here as inspiration.*)
- *Mechanistic interpretability study of multimodal models* ***(variable number of points)****:* Instead of scaling up, let's scale down to a 1-2 layer Transformer and thoroughly investigate the inner workings of the model from a mechanistic interpretability angle, e.g. similar to [12].

4 Evaluation

Your project and models will be evaluated along several complementary dimensions, based on the quality of it's delivery, implementation, results, and the performance of the proposed extensions.

Quality of the solution

1. **Hypothesis-driven research**: Strong projects articulate clear hypotheses or research questions and then design experiments to test them systematically.
2. **Technical Rigor**: Your method and results should be described in enough detail, using figures and sample outputs where applicable, to allow for proper evaluation. Compare your proposed method to appropriate external and self-baselines and perform controlled ablations to demonstrate the impact of your key design choices. Claims in your report about your method/design choices should be supported by quantitative and/or qualitative evidence.
3. **Results**: Model outputs (e.g., quantitative analyses and qualitative examples) must convincingly demonstrate your claims. If you were unable to achieve the results you worked towards and expected, demonstrate an understanding of why that may be the case and what steps you would take to make it work.

Quality of the delivery

1. **Problem Framing & Motivation**: Motivate concisely what you are solving and why it matters.
2. **Clarity & Structure**: The report should be well organized into abstract, introduction, related work, method, experiments, and conclusion sections. See the report guidelines below for more details. Be precise in your writing and avoid purple prose.
3. **Visuals**: Use figures and plots with informative captions to demonstrate your results and to visually convey how your method works. Figures and captions should be able to stand on their own, without requiring extensive back-and-forth between the main text and the figures.

Code verification

1. **Reproducibility**: Submitted code will be evaluated for reproducibility, ensuring that reported results can be independently verified.
2. **Documentation and Clarity**: Provide complete and clear documentation, including a self-contained README, listing all dependencies, setup instructions, and execution details. Ensure that your code is well-organized and thoroughly commented to facilitate easy verification.

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 an opportunity for you to use what you have learned in terms of the implementation and training complexity of multimodal models such as `nano4M` to update your plan for the extensions from the Project Proposal. We will be expecting this updated plan to be realistically achievable in the time left to develop the extensions to the base `nano4M` model. 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, including both a very brief summary of the notebooks and any other steps taken to prepare for implementation of the extensions if any.
- Whether you believe that your original plan for extensions as outlined in the Project Proposal is achievable or not, justifying this belief in terms of expected data needs, compute requirements, training costs etc. needed to achieve your original plan.
- An updated description of the extensions that you plan to implement, together with a more concrete explanation of how this plan will be achieved. This should include a tentative but clear 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 GitHub 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 Presentation Guidelines

Each group will have **5 minutes** to present their system with another **10 minutes** set aside for the demo and any questions. You should expect the audience to have some general understanding of the base `nano4M` model, approximately equivalent to the content covered in the lectures. Hence, the greatest focus should be on your implemented extensions. A general outline for the final presentation is highlighted below:

- Introduction & Motivation: Similar to the project proposal guideline, why is your project useful? What problem is your project addressing?
- Method Overview: Give a high level explanation of your method and model. A method figure can be useful to illustrate to the rest of the class what you have built.
- Technical Section: Highlight one or two key aspects of your design. These can be parts you found interesting or particularly difficult to implement.
- Results: Show the results of the extensions you proposed to tackle, as well as any interesting results or takeaways you had from the project (eg. videos, images, plots, numbers, etc).

We expect each team member to present a portion of their presentation.

5.4 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] Oran Gafni, Adam Polyak, Oron Ashual, Shelly Sheynin, Devi Parikh, and Yaniv Taigman. Make-a-scene: Scene-based text-to-image generation with human priors. *ArXiv*, abs/2203.13131, 2022.

[5] Jonathan Ho. Classifier-free diffusion guidance. *ArXiv*, abs/2207.12598, 2022.

[6] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

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

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

[9] 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.

[10] Jingyuan Liu, Jianling Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Meng Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training. 2025.

[11] 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.

[12] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[13] PyTorch Team: Horace He, Driss Guessous, Yanbo Liang, Joy Dong. FlexAttention: The Flexibility of PyTorch with the Performance of FlashAttention, August 2024.

[14] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2019.

[15] 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.

[16] Noam M. Shazeer. Glu variants improve transformer. *ArXiv*, abs/2002.05202, 2020.

[17] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[18] 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.

[19] Greg Yang, J. Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub W. Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *ArXiv*, abs/2203.03466, 2022.