MEMORY-EFFICIENT LLM TRAINING BY RANDOM SUBSPACE PROJECTION AND NATURAL GRADIENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal promptable foundational model for document question answering. Introduce Document-JEPA objective.

1 INTRODUCTION

We introduce a promptable document question answering model where the focus is on the visual segmentation task, where the model is given a document and a question, and the model is expected to segment the document based on that question. The model is pre-trained on a large corpus of documents and OCRs with the Document Joint Embedding Predictive Architecture (D-JEPA) objective. The D-JEPA objective here masks part of the document, and is predicted based on the joint image and text embedding, while a masked text is predicted based on the image embedding.

2 ONLINE NATURAL GRADIENT ALGORITHM FOR LOW-RANK TENSOR GRADIENTS

2.1 INTRODUCTION

Natural gradient methods are optimization algorithms that adjust parameter updates according to the geometry of the parameter space, leading to faster convergence compared to standard gradient descent. They precondition the gradient using the inverse of the Fisher Information Matrix (FIM). However, computing and storing the full FIM and its inverse is computationally infeasible for large-scale models due to their high dimensionality.

To address this challenge, we propose an **online natural gradient algorithm** that operates in a low-rank subspace of the gradient space. By projecting gradients onto this subspace and approximating the FIM within it, we can efficiently compute natural gradient updates without explicit layer-wise information.

2.2 ALGORITHM DESCRIPTION

The algorithm consists of the following key steps:

1. **Low-Rank Gradient Projection**
2. **Gradient History Buffer Maintenance**
3. **FIM Approximation Using Gradient History**
4. **Natural Gradient Computation via Woodbury Identity**
5. **Efficient Computation Using Cholesky Decomposition**

We will explain each step in detail.

2.2.1 STEP 1: LOW-RANK GRADIENT PROJECTION

Given a full-rank gradient tensor $\boldsymbol{g} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, we project it onto a low-rank subspace using Tucker decomposition. The Tucker decomposition approximates $\boldsymbol{g}$ as:

$$\boldsymbol{g} \approx \mathcal{G} = \mathcal{C} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_d \mathbf{U}^{(d)},$$

where:

- $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ is the core tensor.
- $\mathbf{U}^{(i)} \in \mathbb{R}^{n_i \times r_i}$ are the factor matrices (orthogonal matrices).
- $\times_i$ denotes the mode-i tensor-matrix product.
- r_i is the rank along mode i, with $r_i \ll n_i$.

The low-rank representation (transformed gradient) is obtained by:

$$\boldsymbol{g}_{\text{low}} = \text{Transform}(\boldsymbol{g}) = \boldsymbol{g} \times_1 (\mathbf{U}^{(1)})^\top \times_2 (\mathbf{U}^{(2)})^\top \times_3 \cdots \times_d (\mathbf{U}^{(d)})^\top.$$

2.2.2 Step 2: Gradient History Buffer Maintenance

We maintain a buffer of the recent s low-rank gradient vectors to capture local curvature information. Let $\boldsymbol{g}_t$ be the transformed low-rank gradient at iteration t, flattened into a vector:

$$\boldsymbol{g}_t \in \mathbb{R}^k, \quad \text{where } k = r_1 r_2 \cdots r_d.$$

We store the recent s gradients in the matrix $\mathbf{G}$:

$$\mathbf{G} = [\boldsymbol{g}_{t-s+1}, \boldsymbol{g}_{t-s+2}, \ldots, \boldsymbol{g}_t] \in \mathbb{R}^{k \times s}.$$

2.2.3 Step 3: Approximating the Fisher Information Matrix

We approximate the FIM within the low-rank subspace using the outer products of the stored gradients:

$$\mathbf{F} = \lambda \mathbf{I}_k + \mathbf{G}\mathbf{G}^\top,$$

where:

- $\lambda > 0$ is a damping term to ensure numerical stability.
- $\mathbf{I}_k$ is the $k \times k$ identity matrix.

2.2.4 Step 4: Computing the Natural Gradient via Woodbury Identity

Directly computing $\mathbf{F}^{-1}$ is computationally expensive for large k. Instead, we use the **Woodbury identity** to efficiently compute $\mathbf{F}^{-1}\boldsymbol{g}_t$:

$$\mathbf{F}^{-1}\boldsymbol{g}_t = \frac{1}{\lambda}\left(\boldsymbol{g}_t - \mathbf{G}\left(\lambda \mathbf{I}_s + \mathbf{G}^\top \mathbf{G}\right)^{-1} \mathbf{G}^\top \boldsymbol{g}_t\right).$$

Let us define:

- $\mathbf{S} = \lambda \mathbf{I}_s + \mathbf{G}^\top \mathbf{G} \in \mathbb{R}^{s \times s}$.
- $\boldsymbol{y} = \mathbf{G}^\top \boldsymbol{g}_t \in \mathbb{R}^s$.

Then, we compute:

1. $\mathbf{S} = \lambda \mathbf{I}_s + \mathbf{G}^\top \mathbf{G}$.
2. Solve $\mathbf{S}\boldsymbol{z} = \boldsymbol{y}$ for $\boldsymbol{z}$.
3. Compute $\mathbf{F}^{-1}\boldsymbol{g}_t = \frac{1}{\lambda}(\boldsymbol{g}_t - \mathbf{G}\boldsymbol{z})$.

2.2.5 STEP 5: EFFICIENT COMPUTATION USING CHOLESKY DECOMPOSITION

To efficiently solve the linear system $\mathbf{S}\boldsymbol{z} = \boldsymbol{y}$, we use **Cholesky decomposition**:

1. Compute the Cholesky factorization of $\mathbf{S}$:

$$\mathbf{S} = \mathbf{L}\mathbf{L}^\top,$$

   where $\mathbf{L}$ is a lower triangular matrix.
2. Solve for $\boldsymbol{u}$ in the forward substitution:

$$\mathbf{L}\boldsymbol{u} = \boldsymbol{y}.$$

3. Solve for $\boldsymbol{z}$ in the backward substitution:

$$\mathbf{L}^\top \boldsymbol{z} = \boldsymbol{u}.$$

2.3 ALGORITHM SUMMARY

At each iteration t, the algorithm proceeds as follows:

1. **Project the Full-Rank Gradient**:
   - Obtain the low-rank transformed gradient $\boldsymbol{g}_t$ using the current factors $\{\mathbf{U}^{(i)}\}$.
2. **Update Gradient History**:
   - If the history buffer is full, remove the oldest gradient.
   - Add $\boldsymbol{g}_t$ to the history buffer $\mathbf{G}$.
3. **Compute Intermediate Quantities**:
   - $\mathbf{S} = \lambda \mathbf{I}_s + \mathbf{G}^\top \mathbf{G}$.
   - $\boldsymbol{y} = \mathbf{G}^\top \boldsymbol{g}_t$.
4. **Solve for $\boldsymbol{z}$**:
   - Use Cholesky decomposition of $\mathbf{S}$ to solve $\mathbf{S}\boldsymbol{z} = \boldsymbol{y}$.
5. **Compute the Natural Gradient**:
   - $\tilde{\boldsymbol{g}}_t = \dfrac{1}{\lambda}\left(\boldsymbol{g}_t - \mathbf{G}\boldsymbol{z}\right)$.
6. **Reshape and Project Back**:
   - Reshape $\tilde{\boldsymbol{g}}_t$ back to the low-rank tensor shape.
   - Optionally, project back to the full parameter space using the inverse transform.

2.4 MATHEMATICAL FORMULAS

2.4.1 LOW-RANK PROJECTION

The transformed low-rank gradient is obtained by:

$$\boldsymbol{g}_{\text{low}} = \boldsymbol{g} \times_1 (\mathbf{U}^{(1)})^\top \times_2 (\mathbf{U}^{(2)})^\top \times_3 \cdots \times_d (\mathbf{U}^{(d)})^\top.$$

2.4.2 FIM APPROXIMATION

We approximate the FIM as:

$$\mathbf{F} = \lambda \mathbf{I}_k + \mathbf{G}\mathbf{G}^\top.$$

2.4.3 NATURAL GRADIENT COMPUTATION

Compute the natural gradient using:

$$\tilde{\boldsymbol{g}}_t = \mathbf{F}^{-1}\boldsymbol{g}_t = \frac{1}{\lambda}\left(\boldsymbol{g}_t - \mathbf{G}\boldsymbol{z}\right),$$

where $\boldsymbol{z}$ solves:

$$\left(\lambda \mathbf{I}_s + \mathbf{G}^\top \mathbf{G}\right)\boldsymbol{z} = \mathbf{G}^\top \boldsymbol{g}_t.$$

2.4.4 EFFICIENT SOLUTION VIA CHOLESKY DECOMPOSITION

1. Compute $\mathbf{L}$ such that:

$$\lambda \mathbf{I}_s + \mathbf{G}^\top \mathbf{G} = \mathbf{L}\mathbf{L}^\top.$$

2. Solve for $\boldsymbol{u}$:

$$\mathbf{L}\boldsymbol{u} = \boldsymbol{y}.$$

3. Solve for $\boldsymbol{z}$:

$$\mathbf{L}^\top \boldsymbol{z} = \boldsymbol{u}.$$

4. Compute the natural gradient:

$$\tilde{\boldsymbol{g}}_t = \frac{1}{\lambda}\left(\boldsymbol{g}_t - \mathbf{G}\boldsymbol{z}\right).$$

2.5 NOTES ON IMPLEMENTATION

2.5.1 AVOIDING LARGE MATRIX INVERSIONS

By using the Woodbury identity and Cholesky decomposition, we avoid inverting large matrices of size $k \times k$, where k can be very large.

2.5.2 COMPUTATIONAL EFFICIENCY

- **Matrix-Vector Products**: Preferred over matrix-matrix multiplications for efficiency on GPUs.
- **GPU Computation**: All tensors are kept on the GPU to avoid data transfer overhead.
- **Efficient Linear Algebra**: Utilizing optimized GPU routines for operations like matrix multiplication and Cholesky decomposition.

2.5.3 MEMORY EFFICIENCY

The history size s is kept small (e.g., $s = 10$) to limit memory usage and computational cost.

2.5.4 NUMERICAL STABILITY

The damping term λ ensures that $\mathbf{S}$ is positive definite, allowing for stable Cholesky decomposition.

2.6 CONCLUSION

The proposed online natural gradient algorithm efficiently approximates the natural gradient in a low-rank subspace without requiring explicit layer-wise information. By maintaining a history of recent low-rank gradients and utilizing the Woodbury identity along with Cholesky decomposition, we can compute natural gradient updates efficiently on GPUs. This makes the method suitable for large-scale optimization problems where full natural gradient methods are computationally prohibitive.

3 MODEL

3.1 IMAGE ENCODER

The Hiera model is a novel hierarchical vision transformer designed with a focus on simplicity and efficiency for computer vision tasks. It emerges as a response to the increasing complexity of existing hierarchical vision transformers, such as MViTv2. The Hiera architecture retains the essential components of the ViT framework while eliminating non-essential features that do not contribute to its overall performance, thereby achieving faster operation and competitive accuracy.

Hiera Architecture: Hiera is built upon the foundation of MViTv2, from which the authors carefully removed various specialized components often added in an attempt to enhance performance. Specifically, Hiera eliminates:

1. Convolutions: These have traditionally been used in hierarchical models to impart spatial biases.
2. Shifted or Cross-shaped Windows: These complex mechanisms are designed to improve locality and capture spatial relationships within the data.
3. Decomposed Relative Position Embeddings: These embeddings are typically utilized to encode positional information in a more sophisticated manner.

Instead of these modules, Hiera leverages the Masked Autoencoder (MAE) for strong pretraining, which allows for effective learning of spatial biases. As a result, Hiera comprises of entirely standard ViT blocks, significantly reducing the model's complexity while retaining or even increasing accuracy compared to its predecessor.

Hiera Training Strategy: The training strategy for Hiera centers around the use of the MAE method, which facilitates efficient sparse pretraining. This approach allows Hiera to learn effective spatial representations without relying on complex architecture components. In comparative trials, it was found that when training Hiera with MAE pretraining, the model not only simplifies the architecture but also enhances performance.

Hiera achieves remarkable efficiency:

1. It is up to **2.4×** faster on images than MViTv2.
2. For image classification, Hiera-L is reported to be **3×** faster to train than a supervised MViTv2-L model while achieving superior accuracy.
3. Remarkably, it performs effectively even in smaller configurations, demonstrating strong capabilities where convolutions typically dominated earlier designs.

To adapt the Hiera model for 1024x1024 images, several adjustments would be necessary:

1. Input Rescaling: Ensure that the input layers of the model can handle the larger 1024x1024 images. This might require modification of any layers or preprocessing steps that specifically target a 224x224 resolution.
2. Adjusting Patch Size: The model's patch size may need to be recalibrated to appropriately process the larger images effectively. A larger patch size can enhance efficiency given the increased input size.
3. FLOPs and Memory Considerations: Analyze and possibly upgrade the computational resources (e.g., GPU memory) since higher resolutions will significantly increase the number of computations (FLOPs) and memory usage. Adjusting the batch size may also be necessary.
4. Network Architecture: Depending on the model's performance with the larger size, modifications to the number of blocks, channels, or heads in the Hiera architecture may be necessary. For instance, it could involve scaling up certain configurations (similar to Hiera-B+ or Hiera-L) to maintain performance.
5. Fine-tuning Pretraining: If the model was pretrained on smaller images, it may need to undergo a new pretraining phase on the 1024x1024 images or include an additional fine-tuning step specifically for this resolution to optimize its performance at the new scale.

6. Evaluation Metrics: Adjust evaluation protocols and metrics to appropriately assess model performance at the 1024x1024 resolution, ensuring that you are capturing the accuracy, speed, and efficiency of the model in this new context.

By implementing these adjustments, the Hiera model can be effectively utilized for processing 1024x1024 images while aiming to maintain or improve performance.

Reference: (Ryali et al., 2023, Hiera Model)

In this work, we use a different training strategy, known as the I-JEPA.

I-JEPA Training Strategy: The training strategy for the Image-based Joint-Embedding Predictive Architecture (I-JEPA) is distinct from traditional methods like Masked Autoencoders (MAE) due to its focus on abstract representation learning without reliance on hand-crafted data augmentations.

1. **Architecture Overview:** I-JEPA uses a Vision Transformer (ViT) for encoding and predicting representations. The architecture includes a context encoder and a target encoder, with the purpose of predicting representations of various target blocks from a single context block within the same image.
2. **Predictive Learning:** In I-JEPA, the training objective revolves around predicting the representations of target blocks based on the context block. Unlike generative methods that reconstruct inputs at the pixel level, I-JEPA makes predictions in abstract representation space, focusing on high-level semantic features.
3. **Multi-block Masking Strategy:** The model employs a multi-block masking strategy that enables learning from larger, informative context blocks, which helps in capturing more spatial context and reducing the focus on pixel-level details.
4. **Training Efficiency:** One of the key advantages of I-JEPA is its efficiency; it converges in approximately five times fewer iterations compared to traditional pixel reconstruction methods. This allows I-JEPA to achieve strong performance with significantly less computational effort and training duration.
5. **Adaptability:** I-JEPA scales effectively when trained on larger datasets and benefits from larger model sizes, improving downstream performance even when fine-tuning or adapting the model for specific tasks.

Comparison with MAE:

1. **Data Augmentation Reliance:** MAE relies heavily on hand-crafted data augmentations, processing multiple views of each image during pretraining, while I-JEPA requires only a single view. This gives I-JEPA a scalability advantage and simplifies the training process.
2. **Learning Paradigm:** MAE's approach is generative and focuses on reconstructing masked portions of images at the pixel level, while I-JEPA's focus is on predicting high-level semantic representations in a more abstract sense, which improves the quality of learned features.
3. **Performance:** Empirical evidence indicates that I-JEPA outperforms MAE in several benchmarks, including linear probing on ImageNet-1K and semi-supervised tasks, while also requiring fewer training epochs and less computational power.
4. **Training Dynamics:** Although I-JEPA might introduce some overhead from computing targets in representation space (approximately 7% slower per iteration), its reduced overall training time results in significant computational savings in practice compared to MAE.
5. **Conclusion:** In summary, I-JEPA represents a novel approach to self-supervised learning focused on semantic representation while improving efficiency and performance over traditional methods like MAE by leveraging a predictive approach to learning without extensive reliance on data augmentations.
6. **ImageNet-1K Linear Evaluation:** I-JEPA significantly improves linear probing performance on the ImageNet-1K benchmark compared to MAE. I-JEPA achieves 77.3% Top-1 accuracy with a ViT-H/14 model over 300 epochs, while MAE achieves 71.5% Top-1 accuracy with a ViT-H/14 model over 1600 epochs. This indicates that I-JEPA can achieve superior performance with considerably less training time.

7. **Low-Shot ImageNet-1K (1% Labels):** I-JEPA also outperforms MAE in this scenario, demonstrating better adaptation for image classification using only 1% of the available labels. I-JEPA requires fewer pretraining epochs compared to MAE while still achieving competitive results.
8. **Model Efficiency:** Although I-JEPA is about 7% slower per iteration due to additional overhead from computing targets in representation space, it converges in roughly 5 times fewer iterations than MAE. This leads to significant computational savings in practice.
9. **Fine-Tuning on Full ImageNet:** During fine-tuning, I-JEPA achieves a Top-1 accuracy of 87.1%, which is just 0.7% lower than MAE's 87.8% accuracy. However, I-JEPA is trained for 5.3 times fewer epochs, indicating better efficiency in utilizing training resources.

Overall, I-JEPA consistently shows enhanced performance across various benchmarks while minimizing reliance on extensive epochs of training, making it a more efficient alternative to MAE in self-supervised learning for image representations.

Reference: https://arxiv.org/pdf/2301.08243

3.2 TEXT ENCODER

Large decoder-only language models (LLMs) are the state-of-the-art models on most of today's NLP tasks and benchmarks. Yet, the community is only slowly adopting these models for text embedding tasks, which require rich contextualized representations. In this work, we introduce LLM2Vec, a simple unsupervised approach that can transform any decoder-only LLM into a strong text encoder. LLM2Vec consists of three simple steps: 1) enabling bidirectional attention, 2) masked next token prediction, and 3) unsupervised contrastive learning. We demonstrate the effectiveness of LLM2Vec by applying it to 3 popular LLMs ranging from 1.3B to 7B parameters and evaluate the transformed models on English word- and sequence-level tasks. We outperform encoder-only models by a large margin on word-level tasks and reach a new unsupervised state-of-the-art performance on the Massive Text Embeddings Benchmark (MTEB). Moreover, when combining LLM2Vec with supervised contrastive learning, we achieve state-of-the-art performance on MTEB among models that train only on publicly available data. Our strong empirical results and extensive analysis demonstrate that LLMs can be effectively transformed into universal text encoders in a parameter-efficient manner without the need for expensive adaptation or synthetic GPT-4 generated data.

LLM2Vec is a novel unsupervised approach designed to transform any pre-trained decoder-only large language model (LLM) into a robust text encoder capable of generating rich contextualized representations suitable for text embedding tasks. The transformation process comprises three key steps:

1. **Enabling Bidirectional Attention:** This initial step replaces the standard causal attention mechanism of decoder-only LLMs with a bidirectional attention matrix, allowing each token in the input sequence to access information from all other tokens, including those that come after it. This adjustment aims to enhance the model's ability to incorporate contextual information across the entire sequence.
2. **Masked Next Token Prediction (MNTP):** Following the activation of bidirectional attention, the model undergoes training with a task focused on predicting masked tokens within input sequences. This training helps the model adapt to utilizing the newly available bidirectional context effectively.
3. **Unsupervised Contrastive Learning:** The final step employs an unsupervised contrastive learning approach, which helps the model learn more effective representations of sequences. This is achieved by creating two different representations of the same input sentence and performing a contrastive learning task to optimize the similarity of these representations.

Through these steps, LLM2Vec effectively enhances the capabilities of decoder-only LLMs, enabling them to produce high-quality, universal text embeddings without the need for labeled data or extensive adaptation. The methodology has been demonstrated to outperform traditional encoder-only models on various NLP tasks, particularly on benchmarks like the Massive Text Embeddings Benchmark (MTEB).

3.3 How MLM Works

3.3.1 Input Preparation

- Given a sentence or a sequence of tokens, a certain percentage of tokens (typically 15%) are randomly selected to be masked.
- For each selected token:
  - 80% of the time, it is replaced with the special token `[MASK]`.
  - 10% of the time, it is replaced with a random word from the vocabulary.
  - 10% of the time, it remains unchanged.
- This mixture of masking, random replacement, and unchanged tokens helps the model to not only predict masked words but also to understand the context when the original word is present or when an incorrect word is given.

3.3.2 Model Training

- The sequence with masked tokens is fed into the model, which is trained to predict the original tokens that were masked out.
- The model learns to leverage the context provided by the surrounding tokens to make these predictions.

3.3.3 Loss Calculation

- The model calculates the loss only on the positions where the tokens were masked, comparing its predicted tokens against the actual tokens.
- This loss is then used to update the model's weights during training.

3.3.4 Why MLM is Important

- **Contextual Understanding**: By predicting missing words in a sentence, the model learns to understand the context of words and their relationships within a sentence. This helps the model to generate rich, contextualized embeddings.
- **Bidirectional Training**: Unlike traditional language models that predict the next word in a sequence (and thus are unidirectional), MLM allows the model to consider context from both directions (left and right of the masked token). This bidirectional training is key to BERT's success in various NLP tasks.
- **Generalization**: The random masking process helps the model generalize better by preventing it from relying too heavily on any specific tokens during training.

3.3.5 Variations and Applications

- **Dynamic Masking**: In models like RoBERTa, dynamic masking is used, where the masking pattern changes in each training iteration rather than being fixed. This ensures that the model sees a variety of masked tokens during training, which can improve generalization.
- **Fine-Tuning**: After pre-training with MLM, the model can be fine-tuned on specific downstream tasks (like classification, question answering, etc.) where it applies its learned contextual knowledge to perform these tasks effectively.
- **Extensions**: MLM has been extended or modified in various ways for different models or tasks, but the core idea remains to help the model understand and predict words based on their context within a sequence.

3.3.6 Summary

Masked Language Modeling is a pre-training task where certain tokens in a sentence are masked, and the model learns to predict these tokens using the surrounding context. This task helps the model to develop a deep understanding of the language, enabling it to perform well on a wide range of NLP tasks after fine-tuning.

The text encoder is a BERT model knowledge distilled from a bidirectional version of an open source LLM, further pre-trained for Masked Language Model (MLM) on a large OCR corpus.

3.4 JOINT ARCHITECTURE

The image encoder and the text encoder are combined using a cross-attention mechanism. The combined model is pre-trained on a large corpus of documents and OCRs for Masked Language Model (MLM), Masked Image Model (MIM) and Document Joint Embedding Predictive Architecture (D-JEPA) objectives. The D-JEPA objective here masks part of the document, and is predicted based on the text encoder, while a masked text is predicted based on the image encoder.

Then we have a prompt encoder, which is the CLIP model. The prompt encoder is used to encode the question, and the question is used to prompt the model to segment the document.

The mask decoder efficiently maps the image embedding, prompt embeddings, and an output token to a mask. The mask decoder is a transformer decoder block followed by a dynamic mask prediction head. The modified decoder block uses prompt self-attention and cross-attention in two directions (prompt-to-image embedding and vice-versa) to update all embeddings. After running two blocks, we upsample the image embedding and a MLP maps the output token to a dynamic linear classifier, which then computes the mask foreground probability at each image location. With one output, the model will average multiple valid masks if given an ambiguous prompt.

The model is then fine-tuned on the promptable visual segmentation task.

It also includes a data engine which improves model and data via user interaction.

3.5 STYLE

Papers to be submitted to ICLR 2024 must be prepared according to the instructions presented here.

Authors are required to use the ICLR LaTeX style files obtainable at the ICLR website. Please make sure you use the current files and not previous versions. Tweaking the style files may be grounds for rejection.

3.6 RETRIEVAL OF STYLE FILES

The style files for ICLR and other conference information are available online at:

`http://www.iclr.cc/`

The file `iclr2024_conference.pdf` contains these instructions and illustrates the various formatting requirements your ICLR paper must satisfy. Submissions must be made using LaTeX and the style files `iclr2024_conference.sty` and `iclr2024_conference.bst` (to be used with LaTeX2e). The file `iclr2024_conference.tex` may be used as a "shell" for writing your paper. All you have to do is replace the author, title, abstract, and text of the paper with your own.

The formatting instructions contained in these style files are summarized in sections 4, 5, and 6 below.

4 GENERAL FORMATTING INSTRUCTIONS

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing of 11 points. Times New Roman is the preferred typeface throughout. Paragraphs are separated by 1/2 line space, with no indentation.

Paper title is 17 point, in small caps and left-aligned. All pages should start at 1 inch (6 picas) from the top of the page.

Authors' names are set in boldface, and each name is placed above its corresponding address. The lead author's name is to be listed first, and the co-authors' names are set to follow. Authors sharing the same address can be on the same line.

Please pay special attention to the instructions in section 6 regarding figures, tables, acknowledgments, and references.

There will be a strict upper limit of 9 pages for the main text of the initial submission, with unlimited additional pages for citations.

5 HEADINGS: FIRST LEVEL

First level headings are in small caps, flush left and in point size 12. One line space before the first level heading and 1/2 line space after the first level heading.

5.1 HEADINGS: SECOND LEVEL

Second level headings are in small caps, flush left and in point size 10. One line space before the second level heading and 1/2 line space after the second level heading.

5.1.1 HEADINGS: THIRD LEVEL

Third level headings are in small caps, flush left and in point size 10. One line space before the third level heading and 1/2 line space after the third level heading.

6 CITATIONS, FIGURES, TABLES, REFERENCES

These instructions apply to everyone, regardless of the formatter being used.

6.1 CITATIONS WITHIN THE TEXT

Citations within the text should be based on the `natbib` package and include the authors' last names and year (with the "et al." construct for more than two authors). When the authors or the publication are included in the sentence, the citation should not be in parenthesis using `\citet{}` (as in "See Hinton et al. (2006) for more information."). Otherwise, the citation should be in parenthesis using `\citep{}` (as in "Deep learning shows promise to make progress towards AI (Bengio & LeCun, 2007).").

The corresponding references are to be listed in alphabetical order of authors, in the REFERENCES section. As to the format of the references themselves, any style is acceptable as long as it is used consistently.

6.2 FOOTNOTES

Indicate footnotes with a number[1] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).[2]

6.3 FIGURES

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction; art work should not be hand-drawn. The figure number and caption always appear after the figure. Place one line space before the figure caption, and one line space after the figure. The figure caption is lower case (except for first word and proper nouns); figures are numbered consecutively.

Make sure the figure caption does not get separated from the figure. Leave sufficient space to avoid splitting the figure and figure caption.

You may use color figures. However, it is best for the figure captions and the paper body to make sense if the paper is printed either in black/white or in color.

[1] Sample of the first footnote

[2] Sample of the second footnote

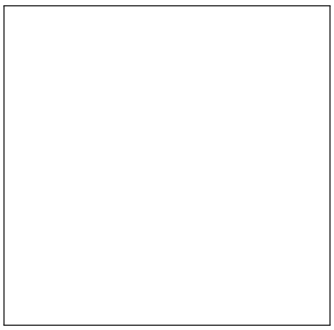

Figure 1: Sample figure caption.

Table 1: Sample table title

PART	DESCRIPTION
Dendrite	Input terminal
Axon	Output terminal
Soma	Cell body (contains cell nucleus)

6.4 TABLES

All tables must be centered, neat, clean and legible. Do not use hand-drawn tables. The table number and title always appear before the table. See Table 1.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

7 DEFAULT NOTATION

In an attempt to encourage standardized notation, we have included the notation file from the textbook, *Deep Learning* Goodfellow et al. (2016) available at `https://github.com/goodfeli/dlbook_notation/`. Use of this style is not required and can be disabled by commenting out `math_commands.tex`.

Numbers and Arrays

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\mathsf{\mathbf{A}}$	A tensor
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\boldsymbol{I}$	Identity matrix with dimensionality implied by context
$\boldsymbol{e}^{(i)}$	Standard basis vector $[0, \dots, 0, 1, 0, \dots, 0]$ with a 1 at position i
$\text{diag}(\boldsymbol{a})$	A square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$
a	A scalar random variable
$\mathbf{a}$	A vector-valued random variable
$\mathbf{A}$	A matrix-valued random variable

Sets and Graphs

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, \dots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b
$\mathbb{A} \backslash \mathbb{B}$	Set subtraction, i.e., the set containing the elements of $\mathbb{A}$ that are not in $\mathbb{B}$
$\mathcal{G}$	A graph
$Pa_{\mathcal{G}}(\mathrm{x}_i)$	The parents of x_i in $\mathcal{G}$

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
a_{-i}	All elements of vector $\boldsymbol{a}$ except for element i
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
$\mathsf{A}_{i,j,k}$	Element (i, j, k) of a 3-D tensor A
$\mathsf{A}_{:,:,i}$	2-D slice of a 3-D tensor
a_i	Element i of the random vector $\mathbf{a}$

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matrix derivatives of y with respect to $\boldsymbol{X}$
$\nabla_{\mathsf{X}} y$	Tensor containing derivatives of y with respect to X
$\frac{\partial f}{\partial \boldsymbol{x}}$	Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ of $f : \mathbb{R}^n \to \mathbb{R}^m$
$\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x})$ or $\boldsymbol{H}(f)(\boldsymbol{x})$	The Hessian matrix of f at input point $\boldsymbol{x}$
$\int f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral over the entire domain of $\boldsymbol{x}$
$\int_{\mathbb{S}} f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral with respect to $\boldsymbol{x}$ over the set $\mathbb{S}$

Probability and Information Theory

$P(\mathrm{a})$	A probability distribution over a discrete variable
$p(\mathrm{a})$	A probability distribution over a continuous variable, or over a variable whose type has not been specified
$\mathrm{a} \sim P$	Random variable a has distribution P
$\mathbb{E}_{\mathrm{x}\sim P}[f(x)]$ or $\mathbb{E}f(x)$	Expectation of $f(x)$ with respect to $P(\mathrm{x})$
$\mathrm{Var}(f(x))$	Variance of $f(x)$ under $P(\mathrm{x})$
$\mathrm{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$ under $P(\mathrm{x})$
$H(\mathrm{x})$	Shannon entropy of the random variable x
$D_{\mathrm{KL}}(P\|Q)$	Kullback-Leibler divergence of P and Q
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f \circ g$	Composition of the functions f and g
$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\log x$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\dfrac{1}{1+\exp(-x)}$
$\zeta(x)$	Softplus, $\log(1+\exp(x))$
$\|\|\boldsymbol{x}\|\|_p$	L^p norm of $\boldsymbol{x}$
$\|\|\boldsymbol{x}\|\|$	L^2 norm of $\boldsymbol{x}$
x^+	Positive part of x, i.e., $\max(0, x)$
$\mathbf{1}_{\mathrm{condition}}$	is 1 if the condition is true, 0 otherwise

8 FINAL INSTRUCTIONS

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the REFERENCES section; see below). Please note that pages should be numbered.

9 PREPARING POSTSCRIPT OR PDF FILES

Please prepare PostScript or PDF files with paper size "US Letter", and not, for example, "A4". The -t letter option on dvips will produce US Letter files.

Consider directly generating PDF files using `pdflatex` (especially if you are a MiKTeX user). PDF figures must be substituted for EPS figures, however.

Otherwise, please generate your PostScript and PDF files with the following commands:

```
dvips mypaper.dvi -t letter -Ppdf -G0 -o mypaper.ps
ps2pdf mypaper.ps mypaper.pdf
```

9.1 MARGINS IN LATEX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the graphicx package.

Always specify the figure width as a multiple of the line width as in the example below using .eps graphics

```
\usepackage[dvips]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.eps}
```

or

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

for .pdf graphics. See section 4.4 in the graphics bundle documentation (`http://www.ctan.org/tex-archive/macros/latex/required/graphics/grfguide.ps`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*. MIT Press, 2007.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, Jitendra Malik, Yanghao Li, and Christoph Feichtenhofer. Hiera: A hierarchical vision transformer without the bells-and-whistles. *ICML Oral*, 2023. URL `https://arxiv.org/abs/2306.00989`.

A APPENDIX

You may include other additional sections here.