Scaling Multi-Frame Transformers for End-to-End Driving

Anonymous Authors

***Abstract*—Vision-based end-to-end controllers hold the potential to revolutionize the production of Autonomous Vehicles (AVs) by simplifying the implementation of navigation systems and reducing their development costs. However, the large-scale implementation of such controllers faces challenges, such as accurately estimating object trajectories and making real-time decisions. Advanced Deep Learning architectures combined with Imitation Learning provide a promising solution, allowing these controllers to learn from expert demonstrations to map observations directly to vehicle controls. Despite the progress, existing controllers still struggle with generalization and are difficult to train efficiently. In this paper, we introduce CILv3D, a novel video-based end-to-end controller that processes multi-view video frames and learns complex spatial-temporal features using attention mechanisms and 3D convolutions. We evaluate our approach by comparing its performance to CIL++, the previous state-of-the-art controller, and demonstrate significant improvements in the vehicle control accuracy. Our findings suggest that our approach could enhance the scalability and robustness of autonomous driving systems.**

***Index Terms*—Autonomous Vehicles, Transformers, Computer-Vision, Deep Learning**

I. Introduction

Vision-based end-to-end navigation controllers directly compute vehicle controls, including throttle, steering, and braking [12], by utilizing inputs from cameras. These systems have gained popularity due to their simpler implementation and reduced development costs, compared to traditional multi-layer decision-making modules, which fuse inputs from several sensors. However, developing robust controllers of this kind poses significant challenges in the field of Computer Vision, such as handling noisy images, interpreting raw camera inputs, managing variability in driving environments and conditions, and producing real-time vehicle controls.

One notable system that has shown impressive performance in navigation tasks despite these challenges is CIL++ [13]. This system combines cutting-edge Deep Learning methods, such as the ResNet architecture and Transformer Encoders, with Imitation Learning (IL), in order to create a system that performs a wide range of driving tasks and environmental conditions. Moreover, it utilizes only three RGB cameras, eliminating the need for more advanced and expensive sensors like LiDAR, as well as complex sensor fusion pipelines.

Despite the benefits, CIL++ struggles with generalization in complex, densely populated environments with many vehicles and pedestrians. Moreoevr, CIL++ presents instability in steering, making the vehicle to oscillate around the center of its lane. CILRL++ [14] was proposed to address its weaknesses by employing the pretrained CIL++ model as the initial policy for a Deep Reinforcement Learning (DRL) agent, which is then fine-tuned through interaction with simulated environments.

Despite the improved performance, CILRL++ requires significant computational resources and training time to reach optimal results. Our investigation revealed that this stems from the sub-optimal conditions under which CIL++ was developed. First, excessive noise during data collection lead to steering control instabilities. Second, the lack of effective data augmentation techniques reduced the model's generalization capabilities. Third, it lacks environmental awareness, because it employs only the current observations, completely disregarding past information, and thus making it difficult to drive properly. Finally, CIL++ employs ResNet-34 as its feature extractor for image processing, which although it has fast inference time, it is outdated and has shown relatively low accuracy on ImageNet benchmarks [15].

In this paper, we present CILv3D, an improved version of the CIL++ that addresses its weaknesses and improves its driving performance. To achieve this, we first collected a new and improved training dataset using the autopilot from the CARLA 0.9.15 simulator [16]. We also modified the CIL++ architecture to be compatible with sequential inputs, including consecutive vehicle control data and a video sequence for each view, which are further processed by the backbone model and 3D convolutions. Furthermore, we replaced the outdated ResNet-34 backbone model with Uniformer-S64, a more advanced Transformer-based model that extracts features from images effectively, without adding much computational burden. Finally, we examine and suggest the use of several data augmentation techniques during training to further enhance the learning process.

The remainder of this paper is organized as follows. Section II presents all the important literature relevant to this work. Section III describes the CIL++ architecture, while Section IV offers a detailed explanation of the CILv3D approach, which builds on and improves CIL++. Section V presents the simulation environment, the experimental setup, the metrics, as well as presents and discusses the experimental results. Finally, Section VI concludes this work and outlines direction for future research.

II. Related Work

In 2017, the CARLA team [16] released the CARLA simulator, a realistic driving simulator that enables the development and testing of navigation algorithms. Along with it, they conducted a comparison of various methods for developing

navigation controllers in CARLA simulator, including end-to-end approaches, such as IL and DRL. These methods were evaluated on several driving scenarios, with IL demonstrating superior performance in most of them. However, in that work, the CARLA team used raw RGB pixels as the controller's states, instead of effective state representations, making the learning process slow and challenging for both IL and DRL approaches.

In a recent study by [19], the authors introduced another DRL-based controller that outperformed the initial CARLA team's DRL approach. More specifically, the authors employed several feature extraction methods, such as the bird's-eye-view technique and Convolutional layers, which effectively extracted comprehensive set of features from the RGB images. Despite the improved performance, their controller was only reliable in simple scenarios with low traffic density and without the presence of pedestrians.

The first notable work in end-to-end autonomous driving was introduced by the authors of CIRLS architecture [17], which employed a Deep Neural Network to estimate vehicle controls. CIRLS was trained on a limited dataset consisting of pairs of RGB images and expert vehicle control demonstrations, collected using an older version of CARLA's autopilot module. The model's architecture consists of a frozen ResNet-34, which is used to extract feature maps from the input frames, and a Fully-Connected (FC) network, which is processes these features and predicts the expert's controls, including steering, throttle, and braking. While this approach surpassed the DRL-based controllers and presented promising driving skills, the authors acknowledged that it suffers from overfitting, dataset biases due to several autopilot mistakes, and generalization challenges.

A more recent version of the CIRLS architecture, named CIL++ [13], addressed several key limitations of its predecessor and improved overall performance. First, the authors utilized a different autopilot system to construct the training dataset, named Roach [20], which proved to be more smooth and accurate than the original CARLA's autopilot. During dataset collection, the authors injected artificial noise into the vehicle's control inputs to enhance generalization and reduce overfitting. Second, they employed three RGB cameras (left, front, and right) to capture a wider horizontal field of view around the vehicle. Lastly, they integrated a Transformer encoder after ResNet-34's feature extraction to better emphasize important features from each frame, resulting in improved vehicle control predictions. Although CIL++ achieved state-of-the-art performance in several complex environments within the CARLA simulator, it still faces generalization challenges and instabilities due to sub-optimal training conditions.

The CILRL++ approach [14] attempts to address the instabilities of CIL++ by integrating IL with DRL. In this approach, the CIL++ model is fine-tuned using a DRL algorithm called Phasic Policy Gradient (PPG) [21], along with a custom reward function designed to focus the DRL agent on rectifying the weaknesses of the CIL++ model. To enable fine-tuning with DRL, the authors employed a KL-Divergence loss, which address the Catastrophic Forgetting problem, where a DRL policy model forgets previously learned skills when undergoing new training. Although this approach achieves state-of-the-art performance across several scenarios within the CARLA simulator, it requires extensive training time and computational resources to effectively mitigate the Catastrophic Forgetting problem.

To conclude our literature review, various attempts have been made to develop end-to-end driving systems, including both IL and DRL, with most of them facing generalization issues in complex environments, as well as training difficulties. In our work, we aim to address these challenges by a) improving the data collection and training pipeline and b) introducing an improved network architecture.

III. Background

CIL++ is an end-to-end vision-based controller that operates using frames captured from three RGB cameras (left, front, right) of the host vehicle. Its architecture leverages a transformer encoder, which serves as a mid-level attention mechanism for processing the observed images. Similar to CIRLS, it employs a frozen ResNet-34 network as its backbone model for extracting useful feature maps from the input images. However, instead of passing these features directly to the fully connected (FC) network, it enhances them by combining the extracted feature maps with the control and positional embeddings, as illustrated in Figure 1. These enriched features then pass through Self-Attention blocks of the transformer encoder, which further refines the information before being fed into the final FC network. Finally, the FC network estimates the desired steering and acceleration, which is used to navigate the vehicle.

This architectural improvement enables CIL++ to achieve state-of-the-art performance in several challenging towns within the CARLA simulator. However, despite these advances, CIL++ still encounters generalization issues and instabilities, especially when applied to unseen environments in CARLA simulator.

IV. Methodology

Our methodology focuses on a video-based end-to-end approach, which can be structured into three core modules. The first module involves constructing diverse driving scenarios and establishing the dataset collection process. The second module involves the design of the video-based neural network architecture, which generates the desired vehicle controls. Building upon the CIL++ framework, this architecture incorporates advanced transformers and temporal dependencies, allowing the neural network to process not just the current frame but also the context from previous frames, thus improving the understanding of dynamic driving environments. The third module involves the training of the neural network, which includes the preprocessing of the collected data and the data augmentation techniques. This module aims to reduce overfitting and enhance the model's ability to generalize and

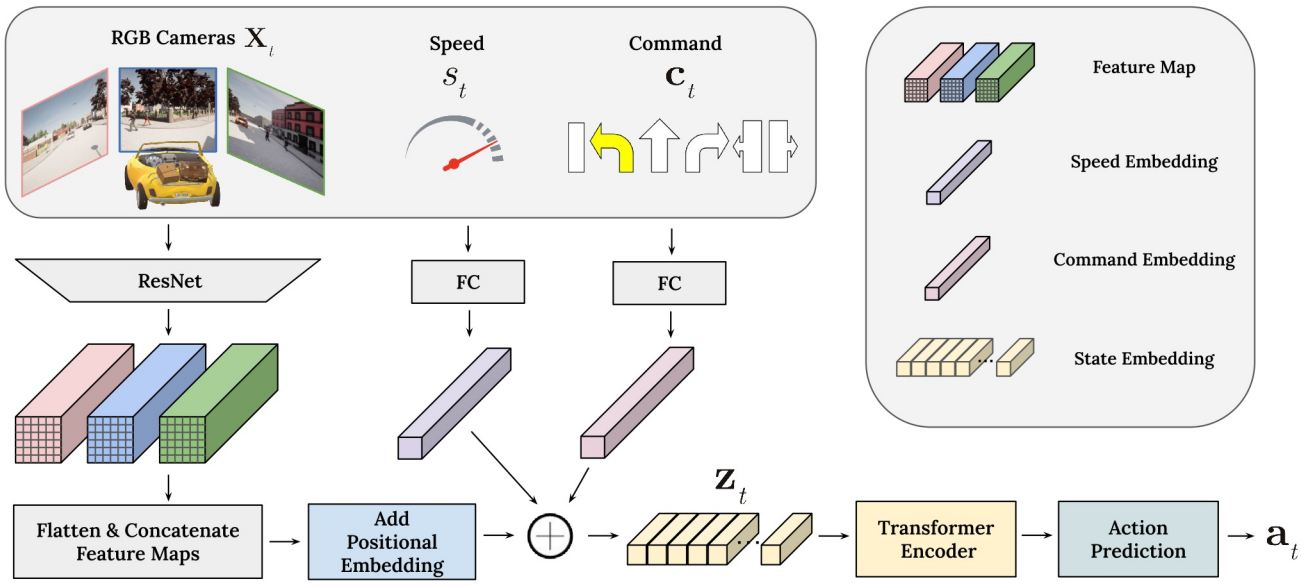

Fig. 1. Overview of the multi-view CIL++ architecture.

perform well in unseen scenarios, ensuring a more robust and reliable navigation system.

A. Dataset Collection

Although the authors of CIL++ provide the datasets used to train their model, we identified two main issues in their methodology.

1) Autopilot Expert: First, they used Roach, an outdated autopilot system for the CARLA simulator, which used to outperform CARLA's earlier autopilot system and making more smooth decisions. However, since then, CARLA's autopilot has significantly improved, reducing its perception and decision-making errors. The updated CARLA autopilot now also utilizes internal simulator data, such as the precise location and direction of every vehicle and pedestrian within the simulation. Instead, Roach relies solely on top-view cameras to observe the traffic around the vehicle, making it a less reliable autopilot. Additionally, CARLA's autopilot is more user-friendly and requires less fine-tuning. This makes CARLA's current autopilot more effective for collecting expert demonstrations

2) Vehicle Control Noise: Another issue we identified during the dataset collection process of the CIL++ approach was the introduction of artificial noise during the vehicle's navigation. Specifically, at each simulation step, the authors added random, small amounts of noise to Roach's control outputs, including acceleration, steering, and speed. Although adding noise is a well-known regularization technique to help reduce overfitting, in this case, it had a negative effect on the CIL++ system. The controller exhibited a tendency to cause the vehicle to oscillate around the center of the lane, struggling to maintain steady steering. This occurred because, during the dataset collection process, the vehicle did not consistently navigate in a straight line at the center of the lane. As a result, the controller learned these deviations, leading to instability and an inability to consistently stay centered in the lane. To avoid this issue, we apply both noise and data augmentation only during the training process, as described later in this Section.

3) Dataset: We collected a diverse dataset of approximately 80,000 expert demonstrations of driving data using CARLA 0.9.15. The dataset captures a wide range of driving scenarios across all available CARLA maps, including varying weather conditions, traffic densities, and urban layouts. Each demonstration includes:

- Three synchronized camera views (left, center, right), denoted as l_t, f_t, r_t respectively.
- Vehicle state information, which is a vector (v_t) that includes the acceleration, steering, speed and command of the vehicle, denoted as a_t, s_t, u_t, c_t respectively.

The cameras are strategically positioned to cover the left, center, and right views, collectively providing the agent with a 180-degree HFOV. This setup mimics the visual range of human drivers and ensures comprehensive environmental perception [13]. The center camera is responsible for the front view while the other two are responsible for covering the agents peripheral view. This camera layout is crucial for the agent, in order to be able to perform tasks such as lane changing, avoiding obstacles or stopping at traffic lights and crossing pedestrians.

To ensure diversity in our dataset, we developed a randomization script that dynamically adjusts environmental conditions such as weather patterns, pedestrian presence and their behavior. Additionally, for each scenario, we varied the number and types of vehicles on the road, assigning each one random attributes like color, size, and destination. This

approach is designed to create a robust dataset that captures a wide range of driving conditions, enhancing the model's ability to generalize across diverse environments.

B. Network Architecture

Our network architecture is composed of four parts: Spatio-Temporal encoding, state embedding, transformer encoder, and action prediction network. The entire architecture is illustrated in Figure 2. At time t, the current input of the neural network consists of a set of video sequences $SEQ_t = L_t, F_t, R_t$ of size $N+1$, which each video sequence including the current and past N captured frames from each camera, and the vehicle control sequences, denoted as V_t. More specifically, L_t, F_t and R_t are the video sequences from left, front and right view respectively, while V_t is a matrix which includes the current and the past N vehicle state information vectors v_t.

1) Vehicle State Representation: To extract temporal dependencies from the vehicle state V_t, we apply 1D convolutions with 512 filters and kernel size of $N+1$. To produce the final State Embeddings, we then apply a linear layer with 12,800 units. These State Embeddings are then combined with the Spatio-Temporal Embeddings generated from each video sequence to form a unified representation for further processing.

2) Spatio-Temporal Encoding: Each frame is an 3-channel RGB image containing raw pixel values in range $[0, 255]$. The dimensions of each frame are $W \times H = 224 \times 224$, where W and H represent the width and height respectively. Acceleration is representred as a normalized real value $a_t \in [-1.0, 1.0]$, where negative values indicate braking, while positive values indicate acceleration. Similarily, Steering is also a normalized real value $s_t \in [-1.0, 1.0]$, with negative values indicating that the steering wheel is turning left, zero meaning that the vehicle is driving in a straight line, and positive indicating a right turn. Speed is real value $u \in [0, 100]$. Finally, the command c is a navigation command, which can take one of 6 distinct values: *Lane Follow, Turn Left, Move Forward, Turn Right, Change Lane Left, Change Lane Right*, as illustrated in Figure 1.

To extract spatial information from each frame, we replace the frozen ResNet-34 architecture used by CIL++ with a frozen UniFormer-S64 [22] backbone model, which produces feature maps of size $B \times 7 \times 7 \times 512$, where B denotes the batch size of the input. To capture the temporal dependencies within the video sequence, we apply a 3D convolution layer with 512 filters and kernel size of $(N+1, N+1, N+1)$. This generates the Spatio-Temporal Embeddings of the model, resulting in output of size $B \times 5 \times 5 \times 512$, which is then flattened into a vector of 12,800 features for further processing. For each flattened embedding vector corresponding to the left, front and right camera sequences, we add their respective positional Embeddings along with the vehicle State embeddings. Finally, these Embeddings are, we concatenated vertically, resulting in a tensor of $B \times 3 \times 12800$, which is passed into a Transformer Encoder.

3) Transformer Encoder: The Transformer Encoder consists of K connected Transformer Blocks, with each block producing M head units, where K and M are user-defined settings. The final Transformer Block outputs a flattened vector of size M, which is passed into the Action Prediction Model, which is a Fully-Connected network. The final layer of the Action Prediction Model is a linear layer with 2 units, corresponding to the desired outputs: the next step acceleration and steering, denoted as a_{t+1} and s_{t+1} respectively.

4) Uniformer-S64: The UniFormer-S64 is a recent version of the UniFormer architecture designed for efficient image classification [22]. This architecture combines Convolutional Neural Networks with Self-Attention mechanisms by unifying them into a transformer architecture. The architecture integrates local Multi-Head Self-Attention (MHRA) in the shallow layers, which reduces computational burden, and global MHRA in the deeper layers, enabling it to capture global token relations more effectively. One of the model's strengths is the balance it strikes between performance and computational efficiency, with fewer parameters and FLOPs compared to other large transformer models, making it a good option for real-time end-to-end systems.

In our study, we applied Grad-CAM (Gradient-weighted Class Activation Mapping) [23] to compare the key regions of images that UniFormer-S64 and ResNet34 identified as important during their classification tasks within the CARLA simulator. The produced Grad-CAM heatmaps highlight the pixels or areas that a model considers more crucial for making predictions, offering insights about the quality of the produced feature maps of each model. The Grad-CAM visualizations in Figure 3 reveal that UniFormer-S64 assigns greater importance to features that are more critical for identifying objects in the simulated environment compared to ResNet34, such as traffic lights, as well as other vehicles and their surrounding areas. This suggests that the UniFormer architecture is better at prioritizing key details, leading to more precise feature representation, enabling the navigation controller to output more precise vehicle controls.

C. Training

For our preprocessing and training setup, we apply several transformations to the three video sequences and the vehicle state. Since the UniFormer-S64 model is pretrained on ImageNet, we apply the ImageNet normalization, using a mean of $[0.485, 0.456, 0.406]$ and a standard deviation of $[0.229, 0.224, 0.225]$ to standardize the input pixels. The acceleration and steering inputs are already normalized within a range of -1 to 1, but speed, which ranges from 0 to 100, are normalized using Min-Max scaling to convert it into the range of 0 to 1, and thus avoiding biases during the decision-making process. The control commands, which consist of six distinct values, are one-hot encoded, resulting in a vector of size 6 for each input.

To reduce overfitting and improve the model's generalization, we introduced Gaussian noise to the timeframe data. Additionally, we applied various data augmentation techniques to each frame of the video sequences. After experimenting with several methods, we selected the augmentations that worked

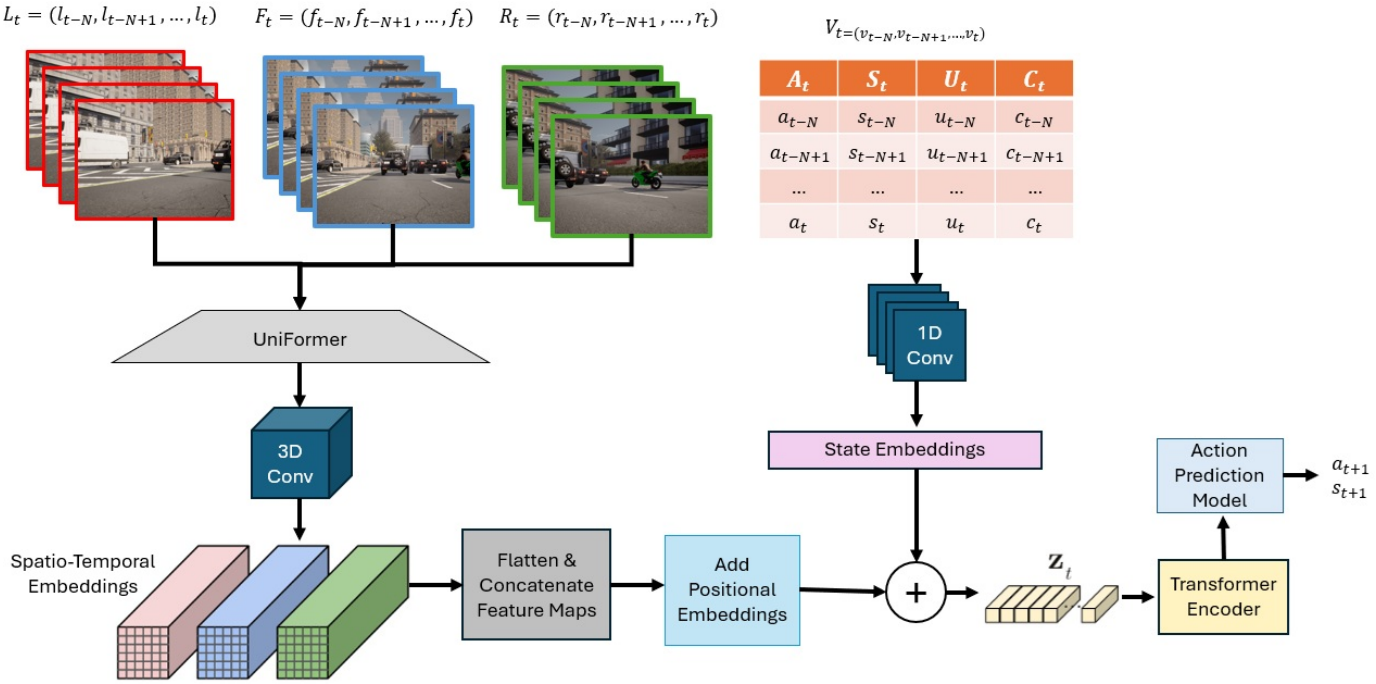

Fig. 2. Overview of the multi-frame CILv3D architecture.

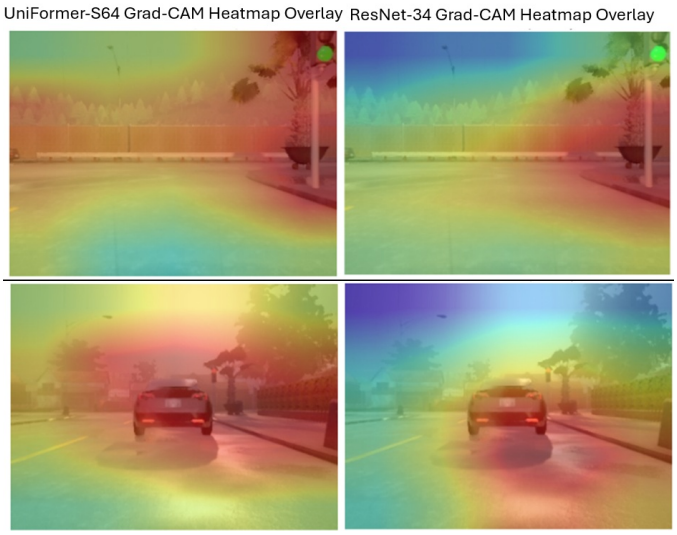

Fig. 3. Grad-CAM visualizations of UniFormer-S64 and ResNet-34. Red and orange areas indicate regions of high importance, while green and blue areas reflect less significance. UniFormer can even identify small details inside an image, such as the vehicle in the left edge of the top image.

best for our model: rescaling the images to 224x224 (which is the input size for UniFormer), adjusting brightness and contrast, applying Gaussian noise, and using blurring. Certain augmentations were discarded as they negatively affected the model's performance and slowed-down the training. For instance, center cropping removed important visual information, rotation was irrelevant because the vehicle's cameras remain steady, color jittering caused confusion by changing the colors of critical elements like traffic lights and lane markings, and horizontal flipping was impractical as the steering wheel is always on the left side.

V. Experiments & Discussion

A. Experimental Setup

B. Metrics

C. Hyperparameter Tuning

D. Results

E. Discussion

VI. Conclusion & Future Work

A. Conclusion

B. Future Work

C. Maintaining the Integrity of the Specifications

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

VII. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections

VII-A to VII-H below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [7].

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. 4", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

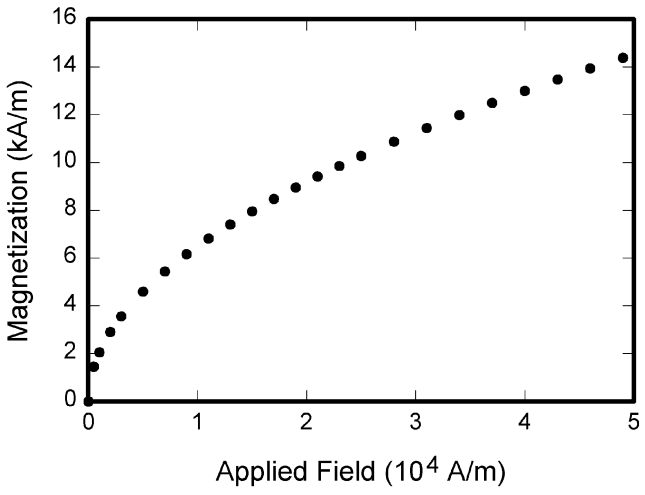

Fig. 4. Example of a figure caption.

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, “Title of paper if known,” unpublished.
[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
[8] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” 2013, arXiv:1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114
[9] S. Liu, “Wi-Fi Energy Detection Testbed (12MTC),” 2023, gitHub repository. [Online]. Available: https://github.com/liustone99/Wi-Fi-Energy-Detection-Testbed-12MTC
[10] “Treatment episode data set: discharges (TEDS-D): concatenated, 2006 to 2009.” U.S. Department of Health and Human Services, Substance Abuse and Mental Health Services Administration, Office of Applied Studies, August, 2013, DOI:10.3886/ICPSR30122.v2
[11] K. Eves and J. Valasek, “Adaptive control for singularly perturbed systems examples,” Code Ocean, Aug. 2023. [Online]. Available: https://codeocean.com/capsule/4989235/tree
[12] Tampuu, Ardi and Matiisen, Tambet and Semikin, Maksym and Fishman, Dmytro and Muhammad, Naveed, “A survey of end-to-end driving: Architectures and training methods”, IEEE Transactions on Neural Networks and Learning Systems, vol. 33, pp. 1364–1384, 2020
[13] Xiao, Yi and Codevilla, Felipe and Porres, Diego and Lopez, Antonio M, “Scaling Vision-Based End-to-End Autonomous Driving with Multi-View Attention Learning”, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1586–1593, 2023
[14] V. Kochliaridis, E. Kostinoudis, I. Vlahavas, “Optimizing Pretrained Transformers for Autonomous Driving”, 13th EETN Conference on Artificial Intelligence (SETN), 2024
[15] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian, “Deep residual learning for image recognition”, Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016
[16] Dosovitskiy, Alexey and Ros, German and Codevilla, Felipe and Lopez, Antonio and Koltun, Vladlen, “CARLA: An open urban driving simulator”, Conference on robot learning, pp. 1–16, 2017
[17] Codevilla, Felipe and Santana, Eder and Lopez, Antonio M and Gaidon, Adrien, “Exploring the limitations of behavior cloning for autonomous driving”, Proceedings of the IEEE/CVF international conference on computer vision, pp. 9329–9338, 2019
[18] Lazaridis, Aristotelis and Fachantidis, Anestis and Vlahavas, Ioannis, “Deep reinforcement 1421: A state-of-the-art walkthrough”, Journal of Artificial Intelligence Research, pp. 9329–1471, vol. 69, 2020
[19] Song, Qingpeng and Liu, Yuansheng and Lu, Ming and Zhang, Jun and Qi, Han and Wang, Ziyu and Liu, Zijian, “Autonomous Driving Decision Control Based on Improved Proximal Policy Optimization Algorithm”, Applied Sciences, p. 6400, vol. 13, MDPI, 2023
[20] Zhang, Zhejun and Liniger, Alexander and Dai, Dengxin and Yu, Fisher and Van Gool, Luc, “End-to-end urban driving by imitating a reinforcement learning coach”, Proceedings of the IEEE/CVF international conference on computer vision, pp.15222–15232, 2021
[21] Baker, Bowen and Akkaya, Ilge and Zhokov, Peter and Huizinga, Joost and Tang, Jie and Ecoffet, Adrien and Houghton, Brandon and Sampedro, Raul and Clune, Jeff, “Video pretraining (vpt): Learning to act by watching unlabeled online videos”, Advances in Neural Information Processing Systems, pp.24639–24654, vol. 35, 2022
[22] Li, Kunchang and Wang, Yali and Zhang, Junhao and Gao, Peng and Song, Guanglu and Liu, Yu and Li, Hongsheng and Qiao, Yu, “UniFormer: Unifying Convolution and Self-attention for Visual Recognition”, arXiv preprint arXiv:2201.09450, 2023.
[23] Selvaraju, Ramprasaath R and Cogswell, Michael and Das, Abhishek and Vedantam, Ramakrishna and Parikh, Devi and Batra, Dhruv, “Grad-cam: Visual explanations from deep networks via gradient-based localiza2017tion”, Proceedings of the IEEE international conference on computer vision, pp. 618–626,