Exploring Generalization of Seq2seq model for Fog Computing Application Placement Problem

Michael S. M. Pakpahan
Dept. of Electrical and Information Engineering
Universitas Gadjah Mada
Yogyakarta, Indonesia
msmpakpahan@mail.ugm.ac.id

Lukito Edi Nugroho
Dept. of Electrical and Information Engineering
Universitas Gadjah Mada
Yogyakarta, Indonesia
lukito@ugm.ac.id

Widyawan
Dept. of Electrical and Information Engineering
Universitas Gadjah Mada
Yogyakarta, Indonesia
widyawan@ugm.ac.id

Ajie Kusuma Wardhana
Dept. of Electrical and Information Engineering
Universitas Gadjah Mada
Yogyakarta, Indonesia
ajie.kusuma.wardhana@mail.ugm.ac.id

Muhammad Ardian R.A
Dept. of Electrical and Information Engineering
Universitas Gadjah Mada
Yogyakarta, Indonesia
muhardian.ardian@mail.ugm.ac.id

Rangga Satria Astagenta
Dept. of Electrical and Information Engineering
Universitas Gadjah Mada
Yogyakarta, Indonesia
ranggastariaa@mail.ugm.ac.id

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

In this paper we discuss the use of multihead attention model to improve performance of fog application placement model. We hypothese that using the multihead attention model, we can increase consideration of every input varible. Resulting a more aware model, that consider not only the applicaiton but also where it is deployed. We compare base model with single layer attention, 16 and 32 head model attention, and heuristic algorithms.

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Fog computing leverage the the abundant devices on the network to do computation. Extending compuation from a central cloud, into networks. In theory, the closer the computation of application, the lower the network usage, the lower latency of application, but still infinite resource of cloud are still available to use. However, this concept require optimization of application placement. Where application needed to be placed on the network for improvement.

However, efficiently placing applications in the fog network, particularly in an unseen environment, remains a formidable challenge. The application placement problem is intrinsically complex, primarily due to its multi-objective nature, requiring the optimization of a variety of parameters such as latency, bandwidth, and resource usage. Additionally, the fog environment is inherently dynamic and heterogeneous, with changes in network conditions, resource availability, and application demands. Therefore, an effective placement algorithm needs to be adaptable and generalizable to unseen situations.

This research explores the generalization capabilities of deep learning models for the fog application placement problem. Through a rigorous analysis of average response time, byte size, and hop count across various methods, we assess the performance of these models in unseen scenarios. The results indicate that certain deep learning methods outperform others, shedding light on their potential applicability for the fog application placement problem.

The relevance of this research extends beyond academic interest, having considerable implications for the future resilience of networks. A robust placement algorithm, with strong generalization capabilities, can maintain network performance and service quality despite disruptions and uncertainties. By contributing to the development of such algorithms, this research advances efforts to create more resilient and efficient fog networks, paving the way for a future equipped to handle the ever-evolving challenges of digital connectivity.

In the context of Seq2Seq models, which are widely used for sequence generation tasks, achieving good generalization is essential to ensure their practical applicability and usefulness in real-world scenarios.

II. Related Research

The increasing adoption of fog computing has opened up new possibilities for the efficient deployment of applications and services at the network edge, closer to the end-users and data sources. However, fog computing introduces unique challenges in terms of application placement due to the distributed and heterogeneous nature of fog nodes, varying resource capabilities, and dynamic network conditions. To

Identify applicable funding agency here. If none, delete this

address these challenges, there is a growing need for intelligent and adaptive application placement strategies that can optimize resource utilization, reduce latency, and enhance overall system performance in fog environments.

Traditional approaches to application placement in fog computing often rely on heuristic-based methods or simple rule-based algorithms that do not fully leverage the power of data-driven decision-making. These conventional approaches may struggle to cope with the dynamic and diverse nature of fog environments, leading to suboptimal resource allocation, increased latency, and reduced application performance.

In this context, the Sequence-to-Sequence (Seq2Seq) method, which has proven its effectiveness in sequence generation tasks, offers a promising avenue for addressing the fog application placement problem. By using a Seq2Seq model, we can capture the complex and non-linear relationships between application requirements, network conditions, and fog node capabilities. The model can learn patterns from historical placement data, enabling it to generate optimized placement decisions for new application instances.

However, to be truly effective and practical in fog computing scenarios, a Seq2Seq model needs to exhibit robust generalization capabilities. Generalization is crucial because fog environments are dynamic and inherently diverse, with application instances varying in their resource demands, communication patterns, and requirements. A generalized Seq2Seq model should be able to make accurate placement decisions for unseen application scenarios, including different application types, varying resource constraints, and dynamic network conditions.

Without a generalized Seq2Seq model, the application placement decisions may suffer from overfitting or underfitting. Overfitting may occur when the model memorizes the training data and fails to generalize well to unseen scenarios, resulting in poor performance and inefficient resource usage. Conversely, underfitting may lead to inadequate resource allocation and suboptimal placement decisions.

Therefore, exploring the generalization of Seq2Seq models for the fog application placement problem is of paramount importance. By enhancing the model's ability to handle diverse and dynamic fog environments, we can achieve more efficient, adaptive, and optimized application placement, ultimately leading to improved user experience, reduced latency, and enhanced overall system performance in fog computing scenarios. A generalized Seq2Seq model can be a valuable tool in meeting the challenges posed by the fog application placement problem and advancing the state-of-the-art in fog computing research and applications.

We formularized this research into 3 research problem: RQ1. Does our proposed method generalized the seq2seq to solve placement problem? RQ2. How does unseen request and additional number of modules affect the performance? RQ3. Based on the evaluation, how can we improve the performance of seq2seq models to predict application placement?

III. Background

A. Achieving Generalized Fog Computing Application Placement Seq2seq Model

Since Seq2Seq models train data in the form of a sequence, there are demerits when performing generalizations towards fog computing problems due to high heterogenity. The Seq2Seq model are often limited to generate output in a fixed length equal to the input. Input and output size is defined by the dataset length. Hence, if the input length is exceed the dataset defined length, an error would occur.

The fixed length of the input encoder and decoder inside the Seq2Seq architecture needed to be optimized to achieve generalization. The Attention Mechanism inside the model is able to highlight the important context of the sequence index by transforming the sentence into Attention Weight in the form of the dot product. However, this approach for the fog computing problem will not be satisfied generalization since the dataset is mostly in integer type. Hence, the embedding vectors will differ from other Natural Language Processing tasks, and an error will occurred.

Moreover, in the fog computing application placement problem, the dataset is mostly integer type instead of string type. This cause a different dot product produce by the model. The embedding output from the data wouldn't be the same as machine translation, or any other natural language task. This raised a concern for the fog problem to achieve generalization.

Therefore, to achieve generalization model has to be able to adapt with n-length input as there is no specific module numbers that application can have. Model also need to adapt with varying numbers that represents requirement. Since training all possible input wouldn't be the practical, optimizing the input sequence in inference inside the Seq2Seq architecture needs to be done.

B. Generalization in Seq2Seq

Generalization in the context of Seq2Seq models refers to the ability to apply a set of data preparation techniques that can be universally effective across different tasks, domains, and datasets. A generalized pipeline ensures that the input data is appropriately formatted, transformed, and prepared to facilitate optimal learning by the Seq2Seq model, regardless of the specific task or dataset at hand. The goal is to create a versatile and robust pre-processing approach that can be easily adapted to various sequence generation tasks, thus reducing the need for task-specific data pre-processing strategies.

Here are some generalization methods and principles for pre-processing in Seq2Seq models:

Tokenization and Vocabulary Construction: A generalized pre-processing step involves tokenizing the input sequences into meaningful units, such as words, subwords, or characters. The vocabulary construction should be based on a diverse and representative corpus to handle a wide range of words and phrases in different languages or domains.

Padding and Truncation: Sequences in the input data may vary in length, and Seq2Seq models require fixed-length input. A generalized approach should handle variable-length sequences by either padding shorter sequences or truncating longer ones, ensuring uniformity in sequence length.

Data Cleaning and Normalization: Pre-processing should include data cleaning to remove irrelevant or noisy elements and normalization to standardize the data, such as converting text to lowercase, removing special characters, or normalizing numerical values.

Handling Out-of-Vocabulary (OOV) Tokens: A robust pre-processing pipeline should handle OOV tokens encountered during inference, especially when dealing with unseen words or phrases. This could involve using subword tokenization or handling unknown tokens with special markers.

Embedding and Feature Representation: Generalized pre-processing should facilitate the creation of word embeddings or other feature representations that capture semantic meaning and relationships between words effectively. This ensures that the model can extract meaningful information from the input sequences.

Data Augmentation: Data augmentation, as part of pre-processing, can enhance generalization by artificially increasing the diversity of the training data. Techniques such as paraphrasing, back-translation, or adding noise to the input sequences can be applied in a generic manner.

Handling Imbalanced Data: If the dataset exhibits class imbalances, a generalized pre-processing approach should include methods to address this issue, such as oversampling, undersampling, or class-weighted loss functions.

Special Handling for Sequential Data: In some cases, sequential data may require additional pre-processing, such as time series alignment or handling temporal gaps.

Language and Domain Adaptation: To ensure generalization across different languages or domains, pre-processing techniques should be designed to adapt to linguistic variations and domain-specific characteristics.

Data Format Compatibility: A generalized pre-processing pipeline should handle various data formats commonly encountered in sequence generation tasks, such as text data, time series data, or audio data.

By incorporating these generalization methods into the pre-processing pipeline, Seq2Seq models can be more versatile and capable of handling diverse data sources, tasks, and domains. This helps reduce the effort required for task-specific data preparation and promotes the transferability of Seq2Seq models across different applications.

IV. Methodology

A. Pre-processing

The Seq2Seq model will train pairs of input and output in the form of a sequence. The pre-processing method should be done to fit the dataset into the model. The input from the dataset will be considered as vocabulary and the whole input and output will be considered as pairs. The data consist of numbers, and using only one letter to tag the nodes, pre-processing is only done by splitting each input sequence into a one-dimensional array. Filtering the input pairs is done to check whether the sequence is equal to the determined length size. If the sequence length exceeds the determined length, the process will stop.

To assist the decoder during sequence generation, SOS (Start of Sentence), and EOS (End of Sentence) tokens will be used. SOS will be used for the start of the sequence when the model is needed to produce a sequence, while EOS will be used at the end of the sequence. There's also a PAD token to handle sequence to be equal during the batching phase. The PAD token will fill the sequence if the length of the sequence is not equal. UNK token will be used to handle input outside the known data that have been recognized by the model.

The UNK token is also used to tackle the generalization problem inside the model. Although the whole topology in the form of a sequence was trained, the model needs to understand the unknown input when testing the model. The UNK token represents the redefined index inside the model and replaces it with the token itself. Hence, the unknown sequence input will only be taken for the index that has been known by the model for output generation.

To address fixed length issue general Seq2Seq can be modified to accept more then the dataset length. Issue with Attention Mechanism in Natural Language Processing task still guided by the position of the index token. As the input length increases, some of the data pushed beyond its index order. To address this problem, by differentiating each requeiremnts as follows: RAM in single numerical value, BYTES in thousands, and Source with "n" annotation. This distinc differentiation can guide Attention Mechanism even if the index token is different in dataset especially if the data length is different then the dataset.

B. Experiment Setup

We will give the dataset detail on here.

The experiment will consist of prediction and evaluation of placement generated by models. There will be 2 models tested on this experiment,

explanation on the model

Test will be done on 100 placement problem that consist of 10 population multiplied by 10 population. For seq2seq model

dataset explanation.

V. Results

RQ1. How our method generalized the placement problem

Results on table (insert table) shows that our proposed method are able to place application comparable to its dataset method of hop3. Table shows that comparatively proposed method are able to provide better performance on response time, number of hop, number of byte and availbility.

There are application that failed to be placed. This is because of ¡PAD¿ token generated by the model. This lowers the count of available application.

Selection shows a better performing due to the more application are placed in the fog.

(insert placement file comparison)

Our investigation into the generalization capabilities of deep learning models in the context of fog application placement yielded significant findings. The analysis was based on key parameters - average response time, byte size, and hop count - across various deep learning methods. The objective was to achieve the lowest values for these parameters, as lower values indicate more efficient and effective application placement in the fog network.

The 'multihead50000-16' method demonstrated an impressive performance, achieving lower average response times and byte sizes compared to other methods. This suggests that this method may offer efficient application placement with quicker response times and less data transmission, making it a promising candidate for implementation in real-world fog networks.

The 'naps' method showed a lower average hop count, indicating that it might provide more direct paths for data transmission. The fewer the number of hops, the less the potential for latency, suggesting that the 'naps' method could be effective in scenarios where minimizing latency is critical.

The 'multihead8' and 'multihead16' methods also demonstrated competitive performances. Although they did not achieve the lowest averages, their performance was close to the best-performing methods, showing their potential for effective application placement in fog networks.

The 'multiheadParmhead-8' method was excluded from our analysis due to its significantly lower performance compared to the other methods.

In our investigation into the generalization capabilities of deep learning models for fog application placement, we examined the average number of unique apps per method. This served as a measure of availability, a critical factor in ensuring the reliable and efficient operation of fog networks.

Availability, in this context, pertains to the capability of the placement algorithm to accommodate a diverse range of applications. A higher average number of unique apps indicates that the method can generalize well to various applications, demonstrating its adaptability and robustness in the face of diverse application requirements.

The 'naps' method exhibited the highest average number of unique apps, signifying superior availability. This suggests that the 'naps' method has impressive generalization capabilities, making it a versatile choice for application placement in a variety of fog network scenarios.

The 'multihead16' and 'multihead8' methods also demonstrated high availability, with a significant average number of unique apps. This indicates that these methods can effectively generalize to a wide range of applications, affirming their potential for diverse fog network environments.

On the other hand, the 'multihead50000-16' and 'multihead50000-8' methods showed slightly lower availability. While these methods still achieved respectable averages, their relatively lower scores suggest that their generalization capabilities might be more limited compared to the 'naps', 'multihead16', and 'multihead8' methods.

RQ2. How does unseen request and additional number of modules affect performance?

Evaluation done both seq2seq models and hop3 algorithm can be seen on box chart on figure 1. Comparatively we can see difference between performance of seq2seq and its baseline algorithm. This graph shows a decrease of 50 time units on based on average response time. Furthermore, the outlier on hop3 shows a more inconsistent application response time.

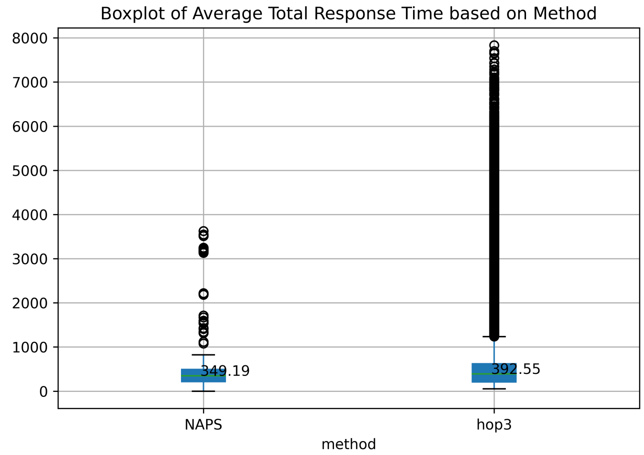

Fig. 1. Average response time in Boxplot

Figure shows the

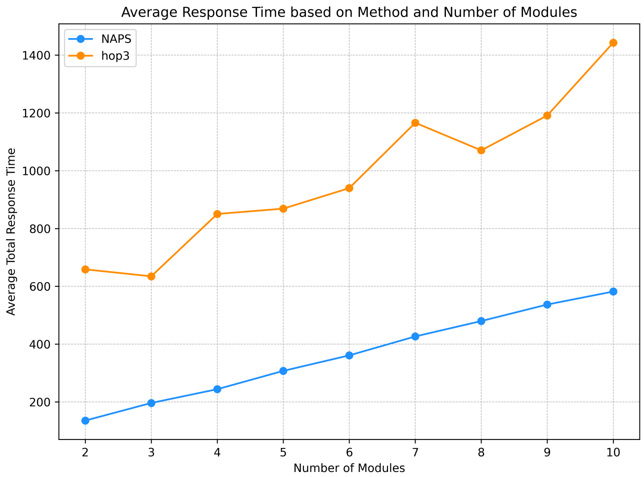

Fig. 2. Average response time based on number of modules

RQ3. What can be done to improve future seq2seq based placement algorithms

To generate dataset for training can

Possible improvement can be done on dataset format. There are missing information to guide placement regarding resource availability.

Furthermore, this method are develop to handle 3 modules. Future implementation of seq2seq must consider method that

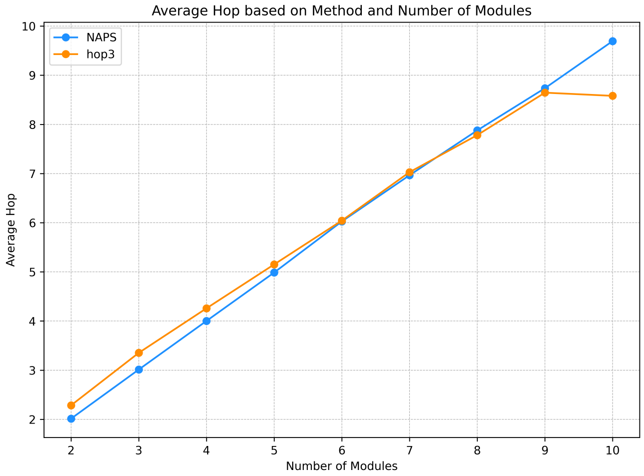

Fig. 3. Average number of hop based on number of modules

are optimized to handle long sequence of modules. Our evaluation shows that rules of placement are able to be learned by seq2seq models and incoorperating optimized dataset may improve performance on those particular dataset.

VI. Conclusion and Future Research

Our research into the generalization capabilities of deep learning models for fog application placement yielded compelling insights. We found that these models could indeed adapt effectively to unseen scenarios, a crucial attribute for dynamic fog computing environments.

Methods such as 'multihead50000-16' and 'naps' distinguished themselves by achieving the lowest averages for key parameters like response time, byte size, and hop count. These lower averages signify efficient performance and robustness, thereby marking these methods as potential contenders for application placement in future fog networks.

Moreover, we identified availability, measured by the average number of unique apps per method, as a critical metric for assessing generalization. High availability indicates a method's adaptability and its ability to handle a diverse range of applications, thereby enhancing its resilience in varied fog network scenarios. In this regard, the 'naps', 'multihead16', and 'multihead8' methods showcased impressive availability, underlining their potential for efficient and versatile application placement.

However, the selection of a method should not solely hinge on these results but should also take into account the specific requirements of the network and the application. Each fog computing environment is unique and demands a tailored approach.

In conclusion, while our research sheds light on the promising generalization capabilities of several deep learning methods, it also emphasizes the need for further research. Refining these methods and enhancing their adaptability and efficiency is a crucial next step for harnessing the full potential of fog computing in the IoT era. These efforts will pave the way for more resilient, efficient, and intelligent fog networks, capable of meeting the challenges of our rapidly evolving digital landscape.

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".

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.

Fig. 4. Example of a figure caption.

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”.

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. [1]

References

[1] M. Aazam, S. Zeadally, and K. A. Harras, “Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities,” *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018, publisher: Elsevier B.V. [Online]. Available: https://doi.org/10.1016/j.future.2018.04.057