



Letter

Evolutionary Neural Architecture Search (NAS) Using Chromosome Non-Disjunction for Korean Grammaticality Tasks

Kang-moon Park ¹, Donghoon Shin ^{2,*} and Yongsuk Yoo ^{3,*}

¹ Department of Computer Science, College of Natural Science, Republic of Korea Naval Academy, Changwon-si 51704, Korea; kmun422@naver.com

² Department of Mechanical Systems Engineering, Sookmyung Women's University, Seoul 04310, Korea

³ Department of Foreign Languages, College of Humanities, Republic of Korea Naval Academy, Changwon-si 51704, Korea

* Correspondence: dhshin@sookmyung.ac.kr (D.S.); yong.yoo@uconn.edu (Y.Y.); Tel.: +82-2-710-9154 (D.S.); +82-55-907-5268 (Y.Y.)

Received: 19 April 2020; Accepted: 14 May 2020; Published: 17 May 2020



Featured Application: Neural Architecture Search (NAS) on linguistic tasks.

Abstract: In this paper, we apply the neural architecture search (NAS) method to Korean grammaticality judgment tasks. Since the word order of a language is the final result of complex syntactic operations, a successful neural architecture search in linguistic data suggests that NAS can automate language model designing. Although NAS application to language has been suggested in the literature, we add a novel dataset that contains Korean-specific linguistic operations, which adds great complexity in the patterns. The result of the experiment suggests that NAS provides an architecture for the language. Interestingly, NAS has suggested an unprecedented structure that would not be designed manually. Research on the final topology of the architecture is the topic of our future research.

Keywords: deep learning; neural architecture search; word ordering; Korean syntax

1. Introduction

In this paper, we apply a modified neural architecture search (NAS) proposed in [1–6] to a grammaticality task for Korean linguistic phenomena. To our knowledge, it is a novel approach to language modeling of Korean linguistic phenomena in terms of automatic neural architecture design. The successful application of deep learning in various fields is due to its automation of pattern finding and powerful performance on difficult problems [7,8]. The fields include image recognition [9,10], natural language processing (NLP) [11,12], game artificial intelligence (AI) [13], self-driving system [14,15], and agriculture [16], among many others. In particular, the deep learning methods have been applied to the fields of psycholinguistics, which attempt to identify the cognitive processing of human languages [17]. This success requires a need for architecture engineering, where more complex neural architectures are designed manually for different tasks. Although open-source toolkits for deep learning have become more various and easy to use [18], the diversity of tasks raises the need for the automation of architecture engineering. NAS, which aims to automate architecture engineering, focuses on automation of engineering architectures, and NAS methods have shown successful results in various tasks including image classification [19,20], object detection [21], or semantic segmentation [22].

NAS can be treated as a subfield of autoML [1] and significantly overlaps with hyper-parameter optimization [23]. Hyper-parameter optimization (HPO) mainly uses Bayesian optimization; however, it cannot generate the topology of neural architecture. The purpose of Bayesian optimization is not on the generation of a topology. By its design, this focuses on parameter optimization of a neural architecture. The optimized neural architecture by the HPO would not change the final topology of the original neural network. The autoML thus contains two steps: the first step is to generate topology by using NAS; the second step is to optimize the hyperparameter of the neural network created by NAS. To generate topology, we need to adopt an evolutionary algorithm. While there are many different kinds of evolutionary algorithms available, the algorithms in question are required to begin from the minimum structure. Here we adopt the variable chromosome genetic algorithm [6], which does not require the minimum structure. This translates into a more efficient tuning method for existing deep learning models to achieve a higher rate of accuracy, as we can start directly from the previous status into a higher accuracy model.

In line with this growing interest in automated architecture engineering, we apply the NAS method to word order patterns found in Korean. Linearization is a process of generating grammatical word orders for a given set of words. Syntactic linearization systems refer to the process of making the grammatical word orders with their syntactic trees. Various word order patterns have been accounted for under syntactic linearization [24,25]. Recently, various language models have been widely tested with various types of word ordering tasks including sentence corrections and word predictions using sequence-to-sequence approaches. They have been compared in terms of their accuracy, and the literature has proven successful on the tasks even without any explicit syntactic information [26]. The Korean language poses an interesting challenge in word order patterns due to two linguistic phenomena: (i) scrambling [4], which permits different ordering of elements, as shown in (1); and (ii) argument ellipsis [5], which permits the missing argument as shown in (2) (the missing argument is crossed in the example). These two properties add complexity into word order patterns, which only exhibit a resulting linear word order that contains underlying linguistic operations.

- (1) a. John-i Mary-lul coahanta.
 John-subject Mary-object like
 'John likes Mary.'
 b. Mary-lul John-i coahanta.
 Mary-object John-subject like
 'John likes Mary.'
- (2) a. (~~John-i~~) Mary-lul coahanta.
 b. John-i (~~Mary-lul~~) coahanta.
 c. (~~John-i~~) (~~Mary-lul~~) coahanta.

Our application of the NAS model to identify word order patterns in Korean yields interesting findings. We show that the patterns in Korean shown above do not require recurrent neural network language models [27] for the desired result. To our knowledge, the application of NAS with a genetic algorithm to language modeling sheds new light on the automated architecture engineering of language models.

This paper is organized as follows: Section 2 introduces NAS with a genetic algorithm suggested in previous research [6]. Section 3 expresses methodology of NAS, Section 4 conducts an experiment on Korean word order patterns. Subsequently, Section 5 contains a discussion and conclusion.

2. Background

2.1. Automated Machine Learning (AutoML)

AutoML is a state-of-the-art deep learning technique that automates the designing of neural network architecture for deep learning. Designing architecture is the most important stage in deep learning, since several types of performance are determined by the design. The majority of the current

architecture is developed manually, which is both a time-consuming and error-prone process [1]. However, as problems for deep learning tasks have scaled up to more challenging and complex tasks, a manual design has become more difficult [28].

The goal of autoML is to find the best performing learning algorithm without human intervention [29]. There are several approaches to autoML: hyper-parameter optimization (HPO), metalearning, and NAS.

HPO aims to find an optimal hyperparameter in a specific neural network. A hyperparameter is a parameter that is necessary for a deep learning process [30]. The optimization of a hyperparameter is very difficult because each deep learning process has different hyperparameters [2,3]. HPO finds these hyperparameters automatically using various methods. Bayesian optimization (BO) is the most famous method for this.

Metalearning can improve the performance of neural networks by using metadata. The challenge for the meta-learning methods is that it must learn from the prior experience in a systematic, data-driven way [2]. However, the final performance of the meta-learning methods is generally lower than the HPO or NAS.

NAS techniques provide an auto-design process for the topology of deep neural network architecture. Recent deep neural networks have been more complex due to their several layers and nodes with links. However, most of the neural network architectures are designed by human experts [2]. Instead, NAS can generate the topology of a neural network architecture automatically. While NAS has similarities with HPO, the difference is that NAS does not focus on parameter tuning.

2.2. Neural Architecture Search (NAS)

NAS has three stages: search space, search strategy, and performance estimation strategy [1,31]. In the search space, we have to define which architectures can be represented. To simplify the size of the search space, it is necessary to incorporate prior knowledge. This stage is highly dependent on human experts. The search strategy is the most important process in NAS; it determines how to explore the search space. The performance estimation strategy evaluates the performance of neural networks to achieve the goal of NAS, which is to find the architecture with the highest performance.

There are many different search strategies to explore the search space of the artificial neural network, such as Bayesian optimization (BO), reinforcement learning (RL), and an evolutionary algorithm (EA). HPO methods employ BO successfully; however, it is not appropriate to generate topology of the neural network due to its goal being within HPO [32,33]. In order to design the topology of the neural network, RL can be considered [34]. RL rewards the agents for their actions distributed by their efficiency. EAs have been adopted by researchers to provide evolving artificial neural networks. The characteristics of the EA methodology is that it generates the topology of neural network architecture with its evolving processes. Since CoDeepNEAT has employed EA for image recognition successfully [28], EA has been a focus of attention in the field. However, EA employs a constructive algorithm which requires a minimum architecture for initialization. In order to overcome this limitation, we will use the variable chromosome genetic algorithm (VCGA) with chromosome non-disjunction [6]. It does not need minimum architecture since it uses a new genetic operation to make the destructive method as well as a constructive method. Since the linguistic dataset we are dealing with has not been tested previously, it does not have any sample of a minimum architecture as well as a middle architecture. VCGA does not care for this issue as it does not require the minimum architecture. It enables us to search the optimal architecture for the dataset. In particular, for this reason, VCGA is used for the current language task since the final result of the experiment can show the topology of the architecture.

3. Methodology

We used the modified NAS that is originally proposed in [6] in this experiment. It uses genetic algorithms, where several genetic operators make different offsprings from their parents. The proposed

method adopts a special genetic operation, called the chromosome non-disjunction, in order to allow the destructive searching not possible in conventional auto-design artificial neural networks (ANN) architectures [6]. The chromosome non-disjunction operation provides the variability for designing an ANN architecture. Consequently, our approach does not need to start from the minimum architecture. Instead, the designer should define the initial ANN architecture.

Figure 1 shows the system configuration of a modified NAS. It consists of a genetic algorithm (GA) generator and a group of neural networks (NN) generators. The genetic operator has three operations: (i) a cross-over operation that blends information of parents to make various but parent-like offspring; (ii) a mutation operation that changes a random gene to make it different from parents; and (iii) a non-disjunction operation [6] that makes two offspring where one has less information and the other one has more information.

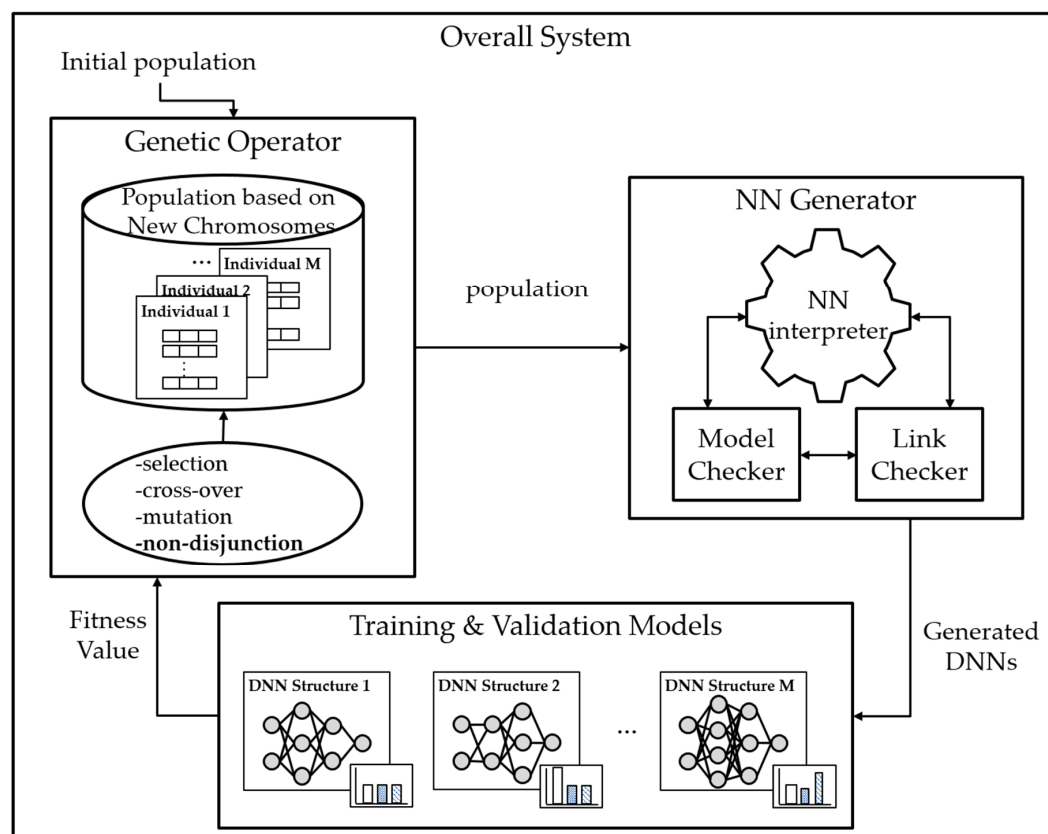


Figure 1. Overall methodology.

The GA operator has a role in controlling the population. It makes new generations of individuals which present ANN architectures. The genetic operator consists of selection, cross-over, mutation, and non-disjunction. The selection operator selects the appropriate individuals which survive from the application of the fitness function as defined in Equation (1). The cross-over operator mixes the chromosomes between the two parents. The mutation operator mutates a part of the information of the parents. The non-disjunction can add or discard the layers or the connections.

The chromosomes are organized with two types: neuron type and connection type. The neuron type of chromosome contains the information of the neuron or layer. It presents the unique number of layers, the numbers of inputs, activation function, stride, etc. The connection type of chromosome contains the information of the connection between two layers with the unique number of them. We refer readers to Park et al [6] for further details on the algorithm of generating NN using chromosomes.

The NN generator designs neural architectures from chromosomes in individuals and evaluates them. The deep neural architecture generator interprets chromosomes with a node checker and link

checker. Every individual has several chromosomes, which means layers and links. However, not all individuals are runnable and learnable. Some have broken connections, and some have unmatched input/output. The model checker and link checker classify these inappropriate neural architectures. The NN architecture evaluator implements training and testing with deep learning. Users must prepare preprocessed training data and testing data. Then, fitness values, which are calculated by NN generators, are sent to the GA operator.

The accuracy of the experiment is calculated by fitness function. Fitness function (Equation (1)) is defined as follows:

$$fitness = (1 - loss) - R * \frac{num_{layer}}{num_{avg}} * (2 * (1 - loss) - 1) \quad (1)$$

where R means the coefficient of dependency rate of the number of layers between 0 to 1. Num_{layer} means the number of layers and num_{avg} means the average of the number of layers.

4. Experiment

4.1. Distribution of Grammatical Sentences in Four-Word Level Sentences in Korean

We have created four-word level sentences in Korean that contain seven syntactic categories: noun phrase, verb phrase, prepositional phrase, adjective phrase, adverbs, complementizer phrase, and auxiliary phrases, which result in 2401 combinations. To confirm the grammaticality of each combination, we have consulted the Sejong Corpus and two linguists, and the distribution of grammaticality is given below. To our knowledge, the data used in this experiment present a novel approach since we have added two syntactic operations that are not visible in the Sejong Corpus. In Table 1, one example of sentences containing scrambling and ellipsis is presented. Although the input slots are counted as four, the underlying sentence may contain more than four words, as shown in the table. This is because few of them can be elided (deleted words), and the orders can be changed. One of the example sentences is given in Table 1. To our knowledge, the data used in this experiment present a novel approach since we have added two syntactic operations that are not visible in the Sejong Corpus.

Figure 2 shows the grammaticality of the dataset; it consists of 113 sentences (circles) that are grammatical. Figure 1 consists of five dimensions, including colors. The X-axis refers to the first slot of four words, and Y and Z refer to the second and the third slot, respectively. The color spectrum refers to the fourth-word slot. The O/X represents grammaticality, which is presented as an output value of the artificial neural network.

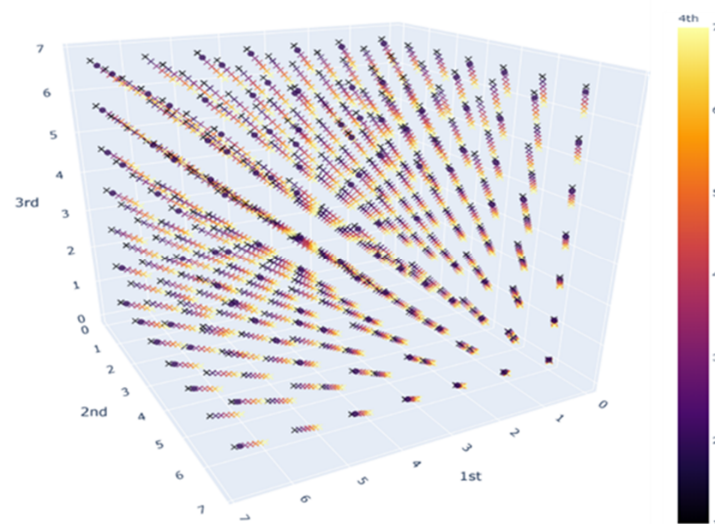


Figure 2. Distribution of input data.

Although our experiment is limited to the four-word level, the Korean language adds a great deal of complexity into patterns due to its underlying syntactic operations. For example, the sentence in Table 1 counts as grammatical even though more than four words are present underlyingly.

Table 1. An example of input data.

Jane-i	Yepputako	John	Maryekey	Cipeysey	Malhassta
Jane	pretty	John	Mary	home	said
	1st input	2nd input		3rd input	4th input
“At home, John said to Mary that Jane is pretty.”					

4.2. Experiment Setups

We ran the experiment of NAS on Korean grammaticality judgment data. The initial neural network of our NAS model is shown in Figure 3. This neural network consists of one input layer, one hidden layer, and one output layer. This hidden layer has five nodes and Rectified Linear Unit (ReLU) as an activation function. The loss of the initial neural architecture model is about 0.270816. Parameters of these experiments are shown in Table 2.

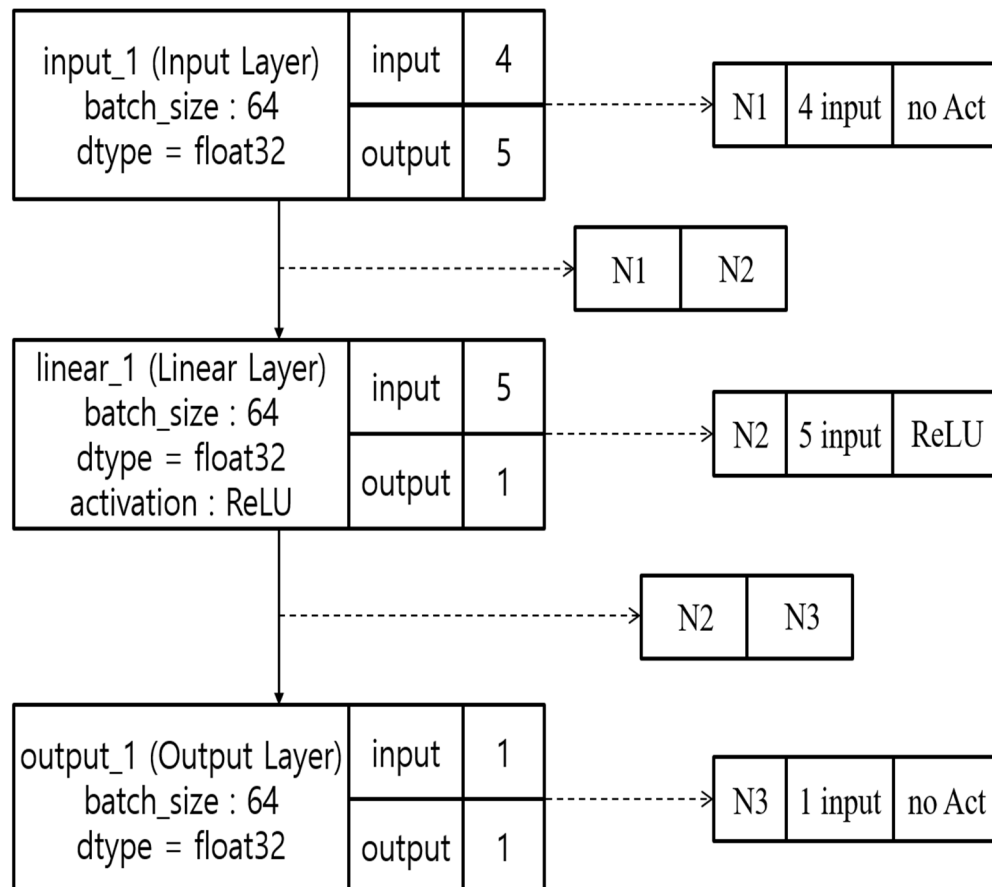


Figure 3. Initial architecture.

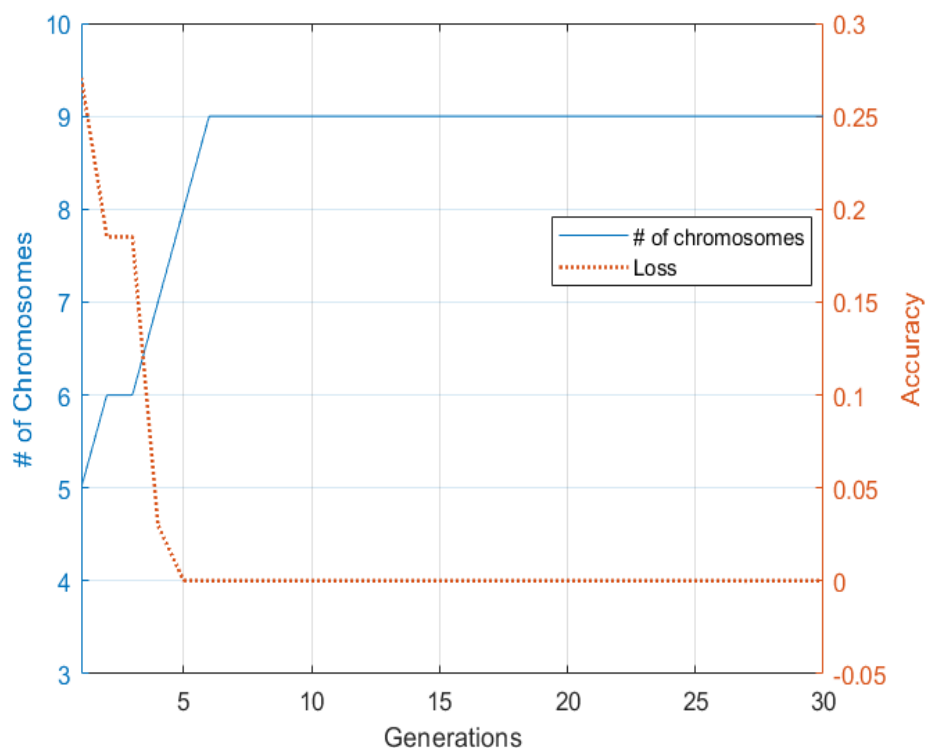
Table 2. Parameters of experiment.

Parameter	Value
Population	50
Generations	30
Mutant rate	0.05
Cross-over rate	0.05
Non-disjunction rate	0.1
Learning rate	0.01
Criterion	MSELoss

4.3. Experiment Results

Figure 4 shows the entire evolution process of the experiment. The number of chromosomes means the complexity of the neural architecture. It starts with five chromosomes within three layers and grows to nine chromosomes. The loss of an auto-designed neural architecture starts from 0.270816 to 0.000096. The final architecture has a linear layer with five nodes and five links to the output layer from a linear layer, as shown in Figure 5. It added a connection to the output layer from the hidden layer every evolution step and converged after generation 6. The resulting topology we have in Figure 5 is rather interesting. While there has to be one-to-one correlation between the input and output, the five outputs of the hidden layer are added into the one input of the output layer. In other words, the five outputs are identical.

The final result of the experiment shows that the initial neural architecture with the loss of 0.270816 is successfully evolved into the converged neural architecture with a loss that is close to 0. In each evolution stage, we have noticed the addition of links into the output layer by one. While the mechanism behind the addition needs to be explored, the final result with the loss that is close to 0 means the NAS can successfully find a neural architecture for the Korean language data.

**Figure 4.** Evolution process of experiment.

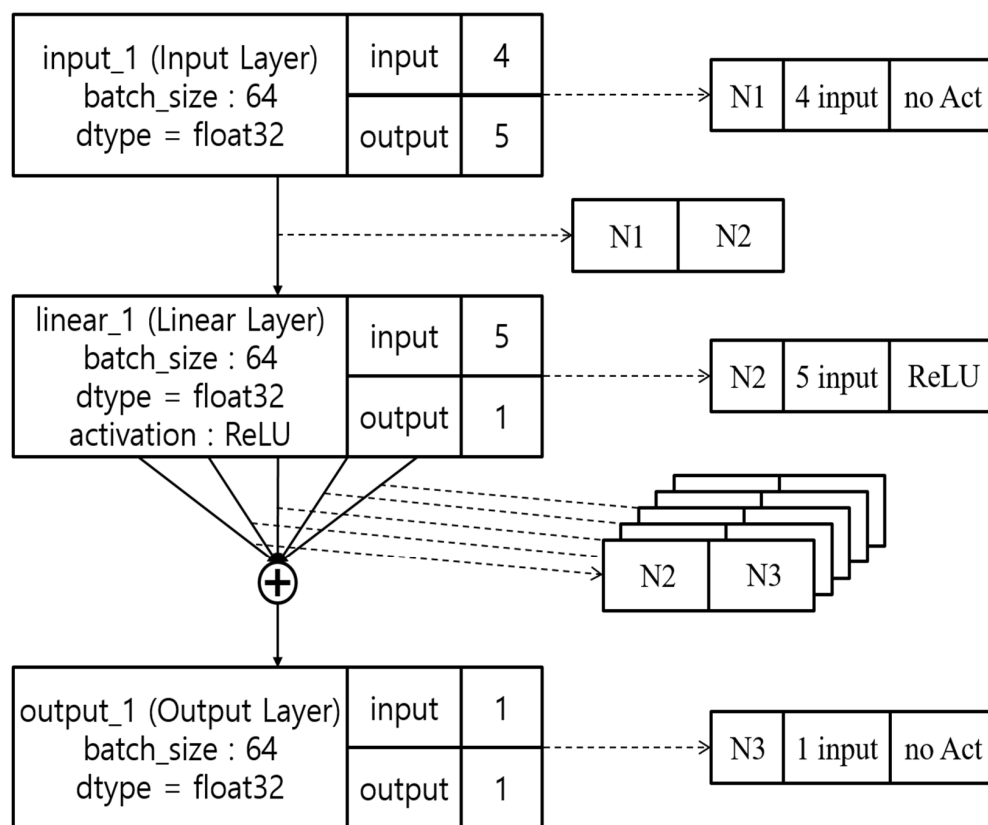


Figure 5. Final architecture of experiment.

5. Discussion and Conclusions

The results of this experiment show that it is plausible to apply the NAS method to linguistic data. In spite of the complexity of the data we have used, the NAS method has been successful in the automation of architecture designing. The experiment sheds new light on language modeling that is generally focused on replicating word order patterns under recurrent neural networks. Furthermore, this work contributes to the field of computational psycholinguistics, since the resulting model could be associated to the block box problem of the generated language models. The result above shows that a simple linear layer can learn the complex patterns in the linguistic data. However, the result of the paper is somewhat limited due to the small amount of the dataset. In further researches, the expanded dataset will be employed, and this will enable the direct comparison between the current NAS and RNN. Furthermore, a modified NAS that can generate RNN will be also researched.

However, we need to improve the data with more inputs. We predict the number of inputs would not affect the topology of the final architecture. In addition to this, cross-linguistic studies that involve more than one language are in order for future research.

Furthermore, the final topology is rather intriguing in that it is not suggested anywhere but only found in the automation of architecture designing. We speculate this structure to be language specific, yet we need to expand our dataset as well as the methodology. This fork-like structure in Figure 4 needs to be investigated in various aspects; this will be done in future research.

Author Contributions: Conceptualization, K.-m.P. and Y.Y.; methodology, K.-m.P.; software, K.-m.P.; validation, Y.Y. and D.S.; formal analysis, K.-m.P.; investigation, Y.Y.; resources, Y.Y.; data curation, Y.Y.; writing—original draft preparation, K.-m.P. and Y.Y.; writing—review and editing, Y.Y. and D.S.; visualization, K.-m.P.; supervision, Y.Y.; project administration, K.-m.P.; funding acquisition, D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the 2020 National Research Projects of Naval Research Center, Republic of Korea Naval Academy, and by the Sookmyung Women’s University Research Grants (1-2003-2008).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Elsken, T.; Metzen, J.H.; Hutter, F. Neural architecture search: A survey. *arXiv* **2018**, arXiv:1808.05377.
2. Hutter, F.; Kotthoff, L.; Vanschoren, J. *Automated Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2019.
3. Adam, G.; Lorraine, J. Understanding neural architecture search techniques. *arXiv* **2019**, arXiv:1904.00438.
4. Saito, M. Some Asymmetries in Japanese and Their Theoretical Implications. Ph.D. Thesis, NA Cambridge, Cambridge, UK, 1985.
5. Kim, S. Sloppy/strict identity, empty objects, and NP ellipsis. *J. East Asian Linguist.* **1999**, *8*, 255–284. [[CrossRef](#)]
6. Park, K.; Shin, D.; Chi, S. Variable chromosome genetic algorithm for structure learning in neural networks to imitate human brain. *Appl. Sci.* **2019**, *9*, 3176. [[CrossRef](#)]
7. Wang, T.; Wen, C.-K.; Jin, S.; Li, G.Y. Deep learning-based CSI feedback approach for time-varying massive MIMO channels. *IEEE Wirel. Commun. Lett.* **2018**, *8*, 416–419. [[CrossRef](#)]
8. Hohman, F.; Kahng, M.; Pienta, R.; Chau, D.H. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Trans. Vis. Comput. Graph.* **2018**, *25*, 2674–2693. [[CrossRef](#)] [[PubMed](#)]
9. Li, A.A.; Trappey, A.J.; Trappey, C.V.; Fan, C.Y. E-discover state-of-the-art research trends of deep learning for computer vision. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 1360–1365.
10. Han, X.; Laga, H.; Bennamoun, M. Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *1*, 1. [[CrossRef](#)] [[PubMed](#)]
11. Lopez, M.M.; Kalita, J. Deep learning applied to NLP. *arXiv* **2017**, arXiv:1703.03091.
12. Young, T.; Hazarika, D.; Poria, S.; Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Comput. Intell. Mag.* **2018**, *13*, 55–75. [[CrossRef](#)]
13. Justesen, N.; Bontrager, P.; Togelius, J.; Risi, S. Deep learning for video game playing. *IEEE Trans. Games* **2019**, *12*, 1. [[CrossRef](#)]
14. Hatcher, W.G.; Yu, W. A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access* **2018**, *6*, 24411–24432. [[CrossRef](#)]
15. Simhambhatla, R.; Okiah, K.; Kuchkula, S.; Slater, R. Self-driving cars: Evaluation of deep learning techniques for object detection in different driving conditions. *SMU Data Sci. Rev.* **2019**, *2*, 23.
16. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [[CrossRef](#)]
17. Linzen, T.; Dupoux, E.; Goldberg, Y. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Trans. Assoc. Comput. Linguist.* **2016**, *4*, 521–535. [[CrossRef](#)]
18. Rebortera, M.A.; Fajardo, A.C. An enhanced deep learning approach in forecasting banana harvest yields. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 275–280. [[CrossRef](#)]
19. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8697–8710.
20. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. Regularized evolution for image classifier architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 4780–4789.
21. Zoph, B.; Cubuk, E.D.; Ghiasi, G.; Lin, T.-Y.; Shlens, J.; Le, Q.V. Learning data augmentation strategies for object detection. *arXiv* **2019**, arXiv:1906.11172.
22. Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 801–818.
23. Feurer, M.; Hutter, F. Hyperparameter optimization. In *Automated Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 3–33.
24. Zhang, Y.; Clark, S. Syntax-based grammaticality improvement using CCG and guided search. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Edinburgh, UK, 27–29 July 2011; pp. 1147–1157.

25. Liu, Y.; Zhang, Y.; Che, W.; Qin, B. Transition-based syntactic linearization. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, CO, USA, 31 May–5 June 2015; pp. 113–122.
26. Schmaltz, A.; Kim, Y.; Rush, A.M.; Shieber, S.M. Adapting sequence models for sentence correction. *arXiv* **2017**, arXiv:1707.09067.
27. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, CA, USA, 5–10 December 2013; pp. 3111–3119.
28. Miiikkulainen, R.; Liang, J.; Meyerson, E.; Rawal, A.; Fink, D.; Francon, O.; Raju, B.; Shahrzad, H.; Navruzian, A.; Duffy, N. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*; Elsevier: Amsterdam, The Netherlands, 2019; pp. 293–312.
29. Wong, C.; Houlsby, N.; Lu, Y.; Gesmundo, A. Transfer learning with neural automl. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 8356–8365.
30. Wicaksono, A.S.; Supianto, A.A. Hyper parameter optimization using genetic algorithm on machine learning methods for online news popularity prediction. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 263–267. [\[CrossRef\]](#)
31. Weng, Y.; Zhou, T.; Li, Y.; Qiu, X. NAS-Unet: Neural architecture search for medical image segmentation. *IEEE Access* **2019**, *7*, 44247–44257. [\[CrossRef\]](#)
32. Kandasamy, K.; Neiswanger, W.; Schneider, J.; Poczos, B.; Xing, E.P. Neural architecture search with bayesian optimisation and optimal transport. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 2016–2025.
33. Ma, L.; Cui, J.; Yang, B. Deep neural architecture search with deep graph bayesian optimization. In Proceedings of the 2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI), Thessaloniki, Greece, 14–17 October 2019; pp. 500–507.
34. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).