Neural Architecture Search based on Cartesian Genetic Programming Coding Method

Xuan Wu¹, Linhan Jia¹, Xiuyi Zhang¹, Liang Chen², Yanchun Liang³, You Zhou¹ and Chunguo Wu*¹
¹Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, College of Computer Science and Technology, Jilin University, Changchun, China
²College of Software, Jilin University, Changchun, China
³Zhuhai Laboratory of Key Laboratory of Symbol Computation and Knowledge Engineering of Ministry of Education, School of Computer, Zhuhai College of Jilin University, Zhuhai, China
*Corresponding Author: wucg@jlu.edu.cn

Abstract—Neural architecture search (NAS) is a hot topic in the field of automated machine learning (AutoML) and has begun to outperform human-designed architectures on many machine learning tasks. Motivated by the natural representation form of neural networks by the Cartesian genetic programming (CGP), we propose an evolutionary approach of NAS based on CGP, called CGP-NAS, for Convolution neural network (CNN) architectures solving sentence classification task. To evolve the architectures under the framework of CGP, the existing key operations are identified as the types of function nodes of CGP, and the evolutionary operations are designed based on Evolutionary Strategy (ES). The experimental results show that the searched architecture can reach the accuracy of human-designed architectures. The ablation tests identify the Attention function as the single key function node and the Convolution and Attention as the joint key function nodes. However, the linear transformations could keep the accuracy of evolved architectures over 70%, which is worthy of investigation in the future.

Key words—Neural architecture search, Cartesian genetic programming, sentence classification

I. INTRODUCTION

The Neural Network is a core technique in modern data-driven artificial intelligence. As an essential component, deep neural networks (DNNs) have surpassed the achievement of former methods in many typical problems and have made excellent solutions to questions in interdisciplinary research. However, the architecture designing of DNNs is limited by the personal knowledge, event by the aesthetics, of designers, which makes it hard to find the global best architectures for a given task. Hence, researchers pay attention to the NAS to relieve the difficulty of architecture design for DNNs. There were many methods proposed to search architecture, and Reinforcement Learning (RL) and Evolutionary Algorithm (EA) were popular methods.

Zoph et al. [1] firstly used the policy gradient algorithm, a RL approach, as the recurrent neural network (RNN) controller to produce architectures of convolutional cells, which are then stacked to compose CNNs. Subsequently, Zoph et al. [2] used the proximal policy optimization as the RNN controller. Baker et al. [3] used Q-learning with the ε-greedy exploration strategy and experience replay to sequentially search for neural architectures. Because of expensive calculations and needs of GPUs, several speed-up methods and efficient solutions were proposed based on the RNN controller. One of them was Efficient Neural Architecture Search (ENAS) [4], in which the controller searched for the best subgraph within a larger graph and shared parameters between subgraphs to accelerate the speed of calculation to 1000 times.

Neuron evolutionary based on EA has a history of 30 years. Gruau [5] proposed Cellular Encoding (CE), which was a grammatical inference process to search neural networks. Yao and Liu [6] proposed EPNnet, which evolved the network architecture and connection weights based on Evolutionary Programming. To evolve neurons, Stanley and Miikkulainen [7] proposed NeuroEvolution of Augmenting Topologies (NEAT), which encoded the neurons into Node genes and Connection genes.

In recent years, scholars used EA to search DNNs rather than neurons. Xie et al. [8] used Genetic Algorithm (GA) to produce and choose better CNNs (GeNet), which required that all convolution operations in the same stage have the same convolution kernel and channel number. Suganuma et al. [9] used the CGP method to encode the CNN architectures (CGP-CNN), which adopted highly functional modules such as convolutional blocks, as the node function in CGP. Different from [9] using graph form of Genetic Programming (GP) [10], Bi et al. [11] used tree form of GP with convolution operators to transform each image into features and then feeds normalized features into a linear support vector machine to perform binary and multiclass image classification. Sun et al. [12] used PSO to search flexible Convolutional autoencoders (FCAE) with chain structure. To search image classifier, Real et al. [13] modified the tournament selection evolution and proposed Aging Evolution which kept as many young individuals as possible.

More and more methods on NAS were proposed, but most of them were proposed on Computer Vision (CV) problems [14], and focus on encoding CNN. Nowadays, researchers made efforts on NAS in Nature Language Processing (NLP). Ramakanth et al. [15] proposed Flexible and Expressive Neural Architecture Search (FENAS), and the results showed FENAS had the ability of reproducing Long Short-Term Memory

XXX-X-XXXX-XXXX-XXX/$XX.00 ©20XX IEEE
(LSTM) and Gated Recurrent Unit (GRU) structures. Since Transformer [16] has become the state-of-the-art model in NLP, David et al. [17] used Transformer as initial, set a new searching space to fit in NLP problems, and used Tournament Selection to find the best candidate in each generation. Results showed that this EA-based NAS algorithm performed better than the original Transformer, and found a novel model called the Evolved Transformer.

Motivated by the effectiveness of the Transformer on NLP problems and the natural representation of DNN with CGP, this paper proposes the NAS based on CGP coding method to deal with sentence classification task.

The remaining parts of this paper are organized as follows: Section II introduces the related work briefly; Section III proposes the evolutionary approach of NAS based on CGP (CGPNAS); Section IV presents the experimental to evaluate the performance of CGPNAS; and finally, Section V presents the conclusion.

II. RELATED WORK

A. Neural architecture search based on Evolutionary Algorithm

Research on NAS based on EA mainly focused on the following two aspects: encoding method and variation operator. Encoding method converts the phenotype of DNNS into the genotype of DNN and the role of variation operator is to produce new genotypes in each iteration. Except encoding method and variation operator, there are also some studies on survival selection strategy and paternal selection strategy [13], [18].

There are two types of encoding methods: direct and indirect. As a widely used method, the direct encoding method explicitly specifies the genotype and restores neural architecture information. In NEAT [7], genotype was composed of Node genes and Connect genes. Node genes restored node type, including input node, output node and hidden node. Connect genes restored the in-node, the out-node, the weight of the connection, an enable bit and an innovation number. Because all convolution of GeNet is the same, Xie et al. [8] only encoded the connection information of network structure into a binary string, in which 1 indicates that two nodes are connected, and vice versa. Because FCAE is a chain architecture, Sun et al. [12] only encoded node type and node type parameter information into genotype.

The indirect encoding method specifies a generation rule of genotype. CE [5] was a classical of the indirect encoding method. The entire neural network evolved from a single ancestor cell where the evolution’s DNA is stored in a tree structure. The tree structure defines the division method of one cell, which leads to the final topology of the network.

In EA, there are two classic variation operators: Crossover and Mutation. Crossover combines the genotype of two or more parents to get one or more offspring genotypes. Mutation changes the genotype of a parent to get a new genotype. To produce new the genotype, different NAS uses one or all of the two variation operators. CGP-CNN [9] used ES as evolutionary operations, so only mutation is used as variation operator. NEAT [7], GeNet [8] and AmoebaNet-A [13] used crossover and mutation as variation operator.

B. sentence classification task

With the development of deep learning, scholars mainly used CNN, RNN and the Attention mechanism [19] to deal with sentence classification tasks. Kim [20] applied a simple CNN to sentence classification task, and achieved excellent results on multiple benchmarks, compared with traditional machine learning methods and RNN. Hochreiter et al. [21] proposed Long Short-term Memory (LSTM), a special RNN, that can learn long-term dependencies. After that, scholars have improved and promoted LSTM, and they have achieved success in dealing with many NLP problems. Vaswani et al. [22] proposed the Transformer architecture based on attention mechanisms, which could be widely used in NLP task. BERT [23] was a pre-trained architecture by Masked Language Model and Next Sentence Prediction and can be finetuned with just one additional output layer to create the state-of-the-art architecture for a wide range of tasks.

Scholars often combined different methods to solve the sentence classification task. Liu and Guo [24] proposed AC-BiLSTM, combining the Attention mechanism, the convolutional layer and the bidirectional LSTM. Zhang et al. [25] proposed 3W-CNN, combining deep learning methods and traditional feature-based methods. Zhang et al. designed a confidence function to divide the outputs of CNN to 2 parts, the results with strong confidence and the results with weak confidence. The CNN classification results with weak confidence will be reclassify by enhanced model.

III. OUR METHOD

In this section, we propose a novel NAS method based on Cartesian genetic programming, named CGPNAS, to deal with sentence classification tasks.

A. CGP CODING METHOD

CGP is a graph form of GP. CGP was initially proposed to optimize digital circuits [26], and hence, each intermediate node had two inputs. Subsequently, CGP were applied to many problems, such as image processing, molecular docking [27]–[30].

For NAS problems, CGP uses a two-dimensional grid as the phenotype of neural networks, as shown in Fig. 1, which is a natural presentation of neural networks due to the topological similarity between CGP and neural networks. The function nodes represent basic functions of the neural networks, such as Convolution, Batch normalization, Pooling, Attention and so on. The links represent the data flow. CGP restricts the links of nodes in the same grid columns, which is corresponding to the usual restriction of connections in neural networks. In addition, CGP usually set a max stride of connection between columns, called “levels-back”, which can increase or reduce the size of searching space.

Fig. 2-a is the coding structure, called genotype of DNN encoded with CGP. For the genotype of DNN, each gene corresponds to a node in Fig. 1 and is composed of three sub-components. The first sub-component, shown as a rectangle,
represents an operation of DNN, such as Convolution, ReLU and so on, and the corresponding parameters. The two successive sub-components, shown as squares, represent the serial numbers of its two input nodes, respectively. Fig. 2-b is the intermediate phenotype of the DNN with both inactive and active links in dashed and solid arrows, respectively. Borrowing the words in genetics, some function nodes encoded in genotype, e.g., Node 6 and Node 8 shown in Fig. 2-a, may not be used as input for subsequent nodes, called inactive nodes [9]. Fig. 2-c is the final phenotype of the DNN with only active links in solid arrows.

![Phenotype of DNN encoded by CGP](image)

**Fig. 1.** Phenotype of DNN encoded by CGP

**B. Function Node Design**

The function nodes used in this paper include Convolution, Attention, Linear, Sum, ReLU, Layer Normalization [31] and Gated Linear Units (GLU) [32]. The function nodes, the number of input nodes and candidate parameter value sets are shown in TABLE I. In addition, n odes can have up to two Gated Linear Units (GLU) [32]. The function nodes, the number of input nodes and candidate parameter value sets are shown in TABLE I. In addition, nodes can have up to two Gated Linear Units (GLU) [32]. The function nodes used in this paper include Convolution, Attention, Linear, Sum, ReLU, Layer Normalization [31] and Gated Linear Units (GLU) [32]. The function nodes, the number of input nodes and candidate parameter value sets are shown in TABLE I. In addition, nodes can have up to two Gated Linear Units (GLU) [32]. The function nodes used in this paper include Convolution, Attention, Linear, Sum, ReLU, Layer Normalization [31] and Gated Linear Units (GLU) [32]. The function nodes, the number of input nodes and candidate parameter value sets are shown in TABLE I. In addition, nodes can have up to two Gated Linear Units (GLU) [32].

![TABLE I. Function Nodes and Their Candidate Parameter Values](image)

**TABLE I. Function Nodes and Their Candidate Parameter Values**

| Function Type | Number of Input Nodes | Parameter Name | Value Set of Parameter |
|---------------|-----------------------|----------------|------------------------|
| Convolution   | 1                     | Channel, Kernel, Padding | [16, 32], [1, 3, 5], SAME |
| Attention     | 1                     | Head           | [4, 8, 16]              |
| Linear        | 1                     | Channel        | [32, 128]               |
| Sum           | 2                     |                | -                      |
| ReLU          | 1                     |                | -                      |
| Layer Normalization | 1   |                | -                      |
| GLU           | 1                     |                | -                      |

Real et al. [19] applied the attention mechanism to NLP for the first time. We use multi head attention to process the information of word vectors. The Attention nodes take \( b \times l \times d \) tensors as inputs and produce \( b \times l \times d \) tensors.

The function of Sum nodes is to merge two branches in the word vector dimension to enable the evolved architectures have multiple branches rather than a chain only. When two branches with different dimension numbers of word vectors are going to be merged, the smaller word vector should be filled with 0 at its end to force it into the same size as the larger one. Hence, the Sum nodes take two tensors with \( b \times l \times d_1 \) and \( b \times l \times d_2 \) dimensions, respectively, as inputs and produce \( b \times l \times d' \) tensors as outputs, where \( d' \) is the larger one of \( d_1 \) and \( d_2 \). Although Sum nodes have two input nodes formally, it is allowed that the Sum nodes received the same input two times from a single precursor node of Sum, such as Node 1 and Node 5, shown in Fig. 2-a.

Ba et al. [31] proposed a normalization method for RNN, named Layer Normalization, which is normalized in the channels and features of samples. The nodes of Layer Normalization take \( b \times l \times d \) tensors as input and outputs, so is ReLU. The Linear nodes represent a linear transformation and take \( b \times l \times d \) tensors as inputs and produce \( b \times l \times d' \) tensors as outputs.

![Fig. 2. DNN Coding Method (Different nodes have different number of inputs.)](image)

**Fig. 2.** DNN Coding Method (Different nodes have different number of inputs.)

**C. Evolution Strategy Design**

As a genetic programming method, CGP usually uses the \((1 + \lambda)\) Evolutionary Strategy (ES) to update and select the population, meaning that one parental individual and \( \lambda \) offspring individuals compete to survive into the next generation. Through mutation operation and adaptive selection, the population evolves towards the optimal goal. According to \([9]\), there are two kinds of mutations in \((1 + \lambda) – ES\), including forced mutation and neutral mutation. The mutation to generate offspring is called the forced mutation, and mutation on parents only is called the neutral mutation, which could contribute new nodes for the next iteration. Both forced mutation and neutral mutation are point mutation, which means that function and connection of node randomly
change to valid values according to mutation rate. To overcome the local optimal traps and balance exploration and exploitation, we double the initial mutation rate for the late 25\% iterations.

The \( \lambda \) offspring individuals are produced by the parental individual of the current generation through the forced mutation. If the \( \lambda \) offspring individuals are all worse with regards to their fitness than their parental individual, the inactive nodes of the parent are mutated by neutral mutation, and the \( \lambda \) offspring individuals are eliminated. Otherwise, the offspring individual with the highest fitness become the parental individual of the next iteration. The pseudocode is described as follows:

1. Create a parent
2. Evaluate the fitness of parent
3. While iteration < maximum iteration
4. Double the mutation rate for late 25\% iterations
5. \( \lambda \) offspring are produced by forced mutation.
6. Evaluate the fitness of \( \lambda \) offspring individuals
7. If the \( \lambda \) offspring individuals are all worse than the parent
8. Mutate the inactive nodes of parent
9. Else offspring with the best fitness become the new parent for the next iteration
10. End

The accuracy of sentence classification task corresponding to each architecture is taken as the individual fitness. The neutral mutation acts on inactive nodes, it does not change the fitness of parent in this iteration, so we do not need evaluate the fitness of the parent after the neutral mutation.

IV. EXPERIMENT

In this section, we first introduce parameters of CGP in our experiment. And then we compare the searched architecture with the classical methods. Finally, we present the ablation testing to analyze the impact of function nodes on the searched architecture.

The following datasets are used in our experiments:
- SST-5 [33], Stanford Sentiment Treebank, is marked with very positive, positive, neutral, negative and very negative 5 labels. SST-2 [33], Binary labeled version of Stanford sentiment treebank, is marked with positive and negative. MR [34] is extracted from Rotten Tomatoes web site pages where movie reviews are marked with positive and negative. IMDB [35] is a large movie review dataset, which is marked with positive and negative. AG_news [36] is extracted by ComeToMyHead website, which is marked with World, Sports, Business and Sci/Tech 4 labels.

Taking the time consumption of architecture search into account, we try to use the small values for the max sentence length, the word vector dimension and the max training epoch. The final performance of architectures fully trained is reported. Hence, the max sentence length is set as 50 for SST-2, SST-5, MR and Ag news, while 400 for IMDB since its average sentence length is 8 times larger than the other datasets. For all experimental datasets, the word vector dimension and the max epoch for architecture search are set uniformly as 300 and 50, respectively.

A. Parameter settings of CGP

The CGP parameters are shown in Table I. Initially, we set the CGP grid as \( 5 \times 20 \) and use a relatively large number of columns size to generate deep architectures. To leverage searching space complexity and models’ generalization ability, Levels-back is set to 3. The number of active nodes in CGP is at least 10 and at most 60. To enhance exploration, Offspring Size is set to 4.

| Parameter                  | Values |
|----------------------------|--------|
| Rows Size                  | 5      |
| Columns Size               | 20     |
| Levels-back                | 3      |
| Activate nodes number     | [10, 60] |
| Mutation rate              | \{0.1,0.2\} |
| Offspring Size             | 4      |
| Maximum iteration          | 1000   |

To balance exploration and exploitation, we set the initial mutation rate of the early 75\% iterations as 0.1, and double it into 0.2 for the late 25\% iterations. However, the mutation rate of SUM function nodes should be larger than that of other functional nodes, otherwise the searched architecture is easy to be a single chain. Hence, with trail experiments, we set the mutation rate of SUM function nodes as 0.2 for early iterations and 0.4 for late iterations.

B. Computational Complexity

CGPNAS’s search space roughly depends on Rows size and Columns size because of the existence of inactive nodes. IMDB’s average sentence length is 8 times larger than the other datasets, so is the most time-consuming dataset in search process. The whole search process on IMDB dataset takes 4 GPU days on Nvidia 2080Ti. As a result, there are 44 million parameters in the model searched on IMDB dataset.

C. Comparison with other algorithms

We compare our method CGPNAS with the following classic architectures: TextCNN [20], Transformer [22], BERT [23], The Evolved Transformer [37], AC-BiLSTM [24], 3W-CNN [25], FENAS [15]. Glove.840B.300D word embedding is applied when pretrained word embedding is required in model.

We train the CNN model [20] with Adam Optimizer for 50 epochs. We train a 6 layers Transformer encoder [22] with the Adam optimizer for 50 epochs. Attention heads number is set to 6 and the learning rate is set to 0.00005. We follow the official guide from [38] to finetune the BERT-Base-Uncased model [23] for downstream tasks. Adam optimizer is applied for 50 epochs and the learning rate is set to 0.00001. We use the searched network from [37], training a 6 layer Evolved Transformer encoder with a linear layer to perform classification task at last. The optimization setting is the same as previous Transformer.

In TABLE III., the items with symbol “*” are the results from the original papers due to lack of public source codes. As an example, the searched architecture on IMBD dataset is shown in Fig. 3. The results show that CGPNAS reach the accuracy of human-designed architectures and CGPNAS(GloVe) performs better than TextCNN, Transformer and The Evolved Transformer. Compared with CGPNAS, CGPNAS(GloVe) improves the accuracy 8\% at
most for SST2 dataset. The Evolved Transformer is a searched architecture of machine translation task, so it may not be suitable for the sentence classification task. On AG_news dataset, CGPNAS(GloVe) achieves the best accuracy. The experiments show that the SST-5 dataset is currently a challenging problem for all involved methods. The AC-BiLSTM performs best on SST-5 dataset due to its bidirectional LSTM units.

**TABLE III. COMPARISON OF DIFFERENT ALGORITHMS**

| Dataset     | Architecture | TextCNN | Transformer | BERT | The Evolved Transformer | AC-BiLSTM* | 3W-CNN* | FENAS* | CGPNAS | CGPNAS(GloVe) |
|-------------|--------------|---------|-------------|------|-------------------------|------------|---------|--------|--------|---------------|
| SST2        | TextCNN      | 0.812   | 0.855       | 0.915| 0.769                   | 0.883      | 0.806   | 0.866  | 0.729  | 0.806         |
| SST5        | Transformer  | 0.372   | 0.365       | 0.423| 0.385                   | 0.489      | 0.423   | -      | 0.372  | 0.423         |
| MR          |              | 0.713   | 0.746       | 0.821| 0.717                   | 0.832      | 0.823   | -      | 0.704  | 0.766         |
| IMDB        |              | 0.840   | 0.863       | 0.912| 0.873                   | 0.918      | -       | -      | 0.850  | 0.860         |
| Ag_news     |              | 0.817   | 0.853       | 0.892| 0.812                   |            | -       | -      | 0.888  | 0.911         |

**D. Ablation study**

To investigate the key component that has remarkable influence on models’ ability, the ablation testing is presented in this section. The function nodes that we test are Convolution and Attention. We remove each function nodes individually to measure the impact of each single function.

It can be seen from 0 that even if the Convolution node is removed (Fig. 3-a), the accuracy is almost unchanged. However, if the Attention function node is removed (Fig. 3-b), the accuracy drops by 8%. The experimental results show that the Attention function node is vital for the searched architecture. If we remove Conv and Attention function nodes simultaneously (Fig. 3-c), the accuracy drops by 15%. Hence, the joint of Convolution and Attention nodes plays the most important role in the evolved architecture. While it is also noted that even if all Convolution and Attention nodes are removed, the accuracy is kept at more than 70%, depending on the combination of Linear, Layer Normalization, Sum and Full. It can be known that the architecture shown in Fig. 4-c performs mainly the linear transformation from its input. The detailed mechanism is worth of investigating in the future.

**TABLE IV. ABLATION TESTING OF CGPNAS**

| Serial number | Accuracy on IMDB | Remarks                      |
|---------------|------------------|------------------------------|
| a             | 0.846            | Remove Convolution node      |
| b             | 0.773            | Remove Attention node        |
| c             | 0.703            | Remove Attention and Convolution |
| d             | 0.850            | The full architecture obtained by CGPNAS |

**E. Transfer learning**

**Fig. 3. The evolved architecture on IMDB**

**Fig. 4. Architectures of ablation testing**

**V. CONCLUSION**

CGP can optimize the structure and parameters of the model at the same time and the phenotype of CGP is similar with neural networks, so it's very suitable for neural architecture search. The results show that the CGPNAS can reach the accuracy of human-designed architectures. According to the ablation testing, the attention mechanism is very important for sentence classification tasks. The future work may include more types of function nodes and design efficient methods for evaluating performance.

**ACKNOWLEDGMENT**

This work is supported by the National Natural Science Foundation of China (61876069 and 61972174), the Jilin Natural Science Foundation (20200201163JC), and the
Science and Technology Planning Project of Guangdong Province (2020A0505100018), the Guangdong Key-Project for Applied Fundamental Research (2018KZDXM076).

REFERENCES

[1] B. Zoph and Q. V. Le, “Neural Architecture Search with Reinforcement Learning,” 2017. [Online]. Available: https://openreview.net/forum?id=r1Ue8H6cx

[2] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning Transferable Architectures for Scalable Image Recognition,” in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun. 2018, pp. 6587–6597. doi: 10.1109/CVPR.2018.00907.

[3] B. Baker, O. Gupta, N. Naik, and T. Darrell, “Designing Neural Network Architectures using Reinforcement Learning,” 2017. [Online]. Available: https://openreview.net/forum?id=S1e2cVqee

[4] H. Pham, M. Y. Guan, B. Zoph, and Q. V. Le, “Efficient Neural Architecture Search via parameter Sharing,” in 35th International Conference on Machine Learning, ICML 2018, 2018, vol. 9, pp. 6522–6531.

[5] F. Gruau, “Cellular encoding as a graph grammar,” in IEEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives, 1993, p. 171-1710.

[6] G. Liu and J. Guo, “Bidirectional LSTM with attention mechanism and convolutional layer for text classification,” Neurocomputing, vol. 337, pp. 325–338, Apr. 2019, doi: 10.1016/j.neucom.2019.01.078.

[7] A. B. Gar mendia-Doval, S. D. Morley, and S. Juhas, “Post Docking Filtering Using Cartesian Genetic Programming,” in Artificial Evolution, Berlin, Heidelberg, 2004, pp. 189–200. doi: 10.1007/978-3-540-24621-3_16.

[8] A. B. Garmendia-Doval, J. F. Miller, and S. D. Morley, “Cartesian Genetic Programming and the Post Docking Filtering Problem,” in Genetic Programming Theory and Practice II, U.-M. O'Reilly, T. Yu, R. Riolo, and B. Worzel, Eds. Boston, MA: Springer US, 2005, pp. 225–244. doi: 10.1007/0-387-23254-0_14.

[9] J. Rothermich and J. Miller, “Studying the Emergence of Multicellularity with Cartesian Genetic Programming,” in Late Breaking Papers at the Genetic and Evolutionary Computation Conference (GECCO-2002, 2002, pp. 397–403.

[10] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, “Language Modeling using Gated Convolutional Networks,” in Proceedings of the 34th International Conference on Machine Learning, International Convention Centre, Sydney, Australia, Aug. 2017, vol. 70, pp. 933–941. [Online]. Available: http://proceedings.mlr.press/v70/dauphin17a.html

[11] A. Vaswani et al., “Attention is All You Need,” in Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 2017, pp. 6000–6010.

[12] Y. Zhang, J. Zhao, and Y. LeCun, “Character-level Convolutional Networks for Text Classification,” in Advances in Neural Information Processing Systems, 2015, vol. 28, pp. 649–657. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/250e98b51c7f37f308e6b496c7a9a2f-Paper.pdf

[13] D. So, Q. Le, and C. Liang, “The Evolved Transformer,” in Proceedings of the 36th International Conference on Machine Learning, Jun. 2019,
[38] T. Wolf et al., “Transformers: State-of-the-Art Natural Language Processing,” in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, 2020, pp. 38–45. doi: 10.18653/v1/2020.emnlp-demos.6.