JUNE 18-22, 2023
CVPR
VANCOUVER, CANADA

NAR-Former: Neural Architecture Representation Learning towards Holistic Attributes Prediction

Yun Yi[1], Haokui Zhang*, Wenze Hu, Nannan Wang*, Xiaoyu Wang

Paper tag: **TUE-PM-343**

西安电子科技大学 XIDIAN UNIVERSITY
Xi'an, China

intellifusion 云天励飞
Shenzhen, China

哈爾濱工業大學(深圳) HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN
Shenzhen, China

*Corresponding authors.
[1]This work was done while Yun Yi was an intern at Intellifusion.

Overview

JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

Modeling and learning the representation of neural networks can be used to predict their attributes of themselves without running the actual estimation procedures, thus improving the efficiency of network design and deployment.

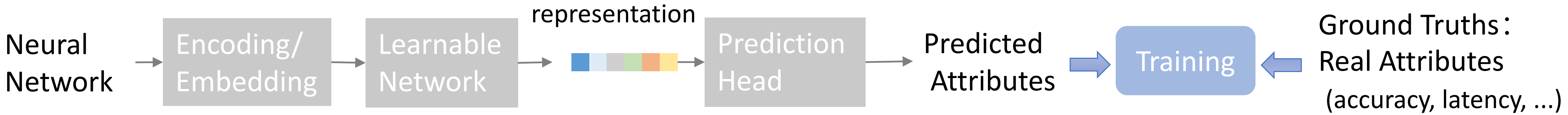
In this paper, in oder to learn general and reasonable representation of neural network, we propose:

- a simple and effective neural network encoding approach to tokenize both operation and topology information of a neural network node into a sequence;
- a multi-stage fusion transformer to learn feature representations;
- an information flow consistency augmentation and an architecture consistency loss to facilitate efficient model training.

Contents

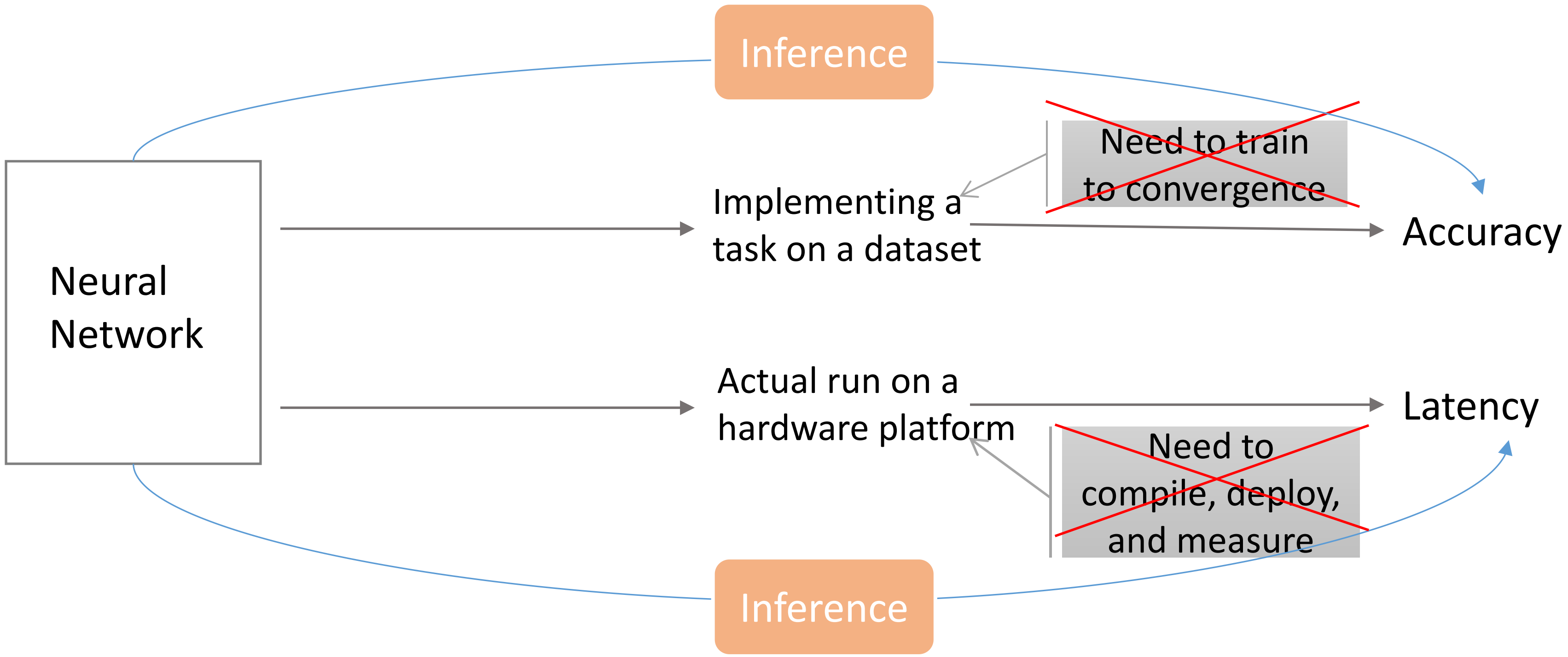
Background

1. What is neural network representation learning



2. Why learn the representation of neurak networks

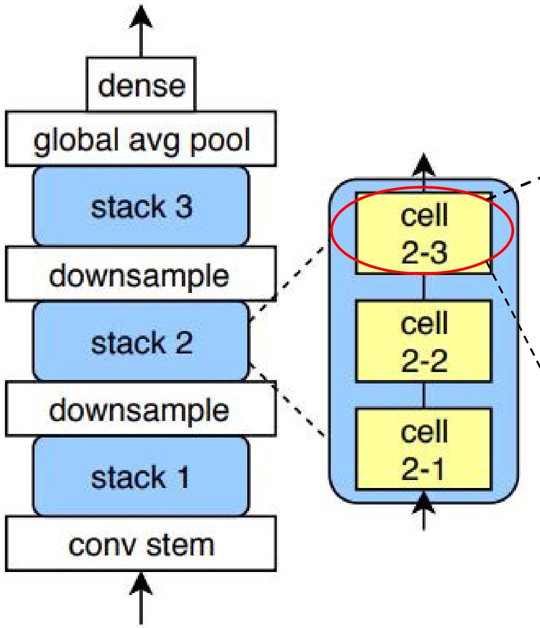
Once the training is finished： Target Neural Network ⇨ Inference ⇨ Network's Attributes



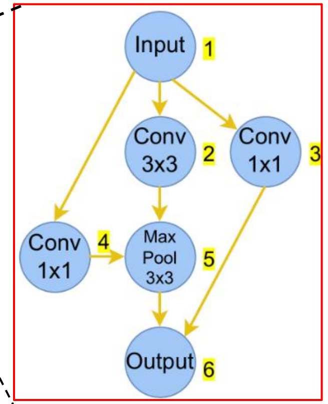
Improving the efficiency of network design and deployment.

Motivation

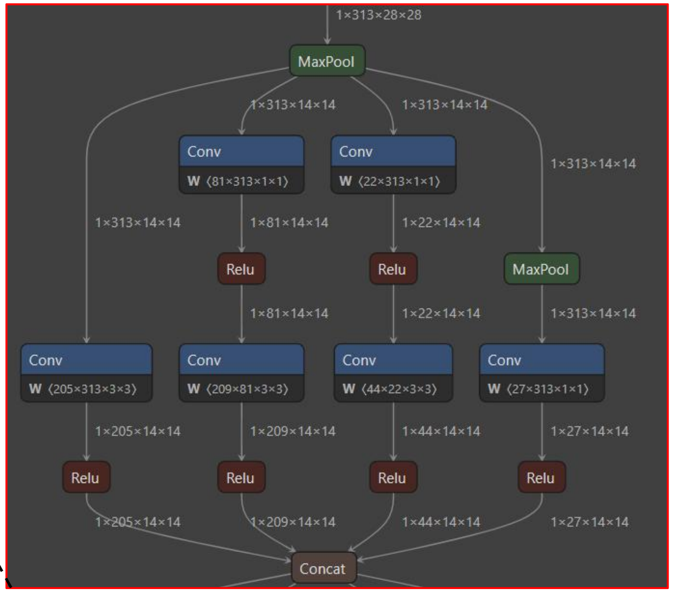
Neural network forms that may need to be encoded in reality:



Architecure in NAS-Bench-101



cell architecture



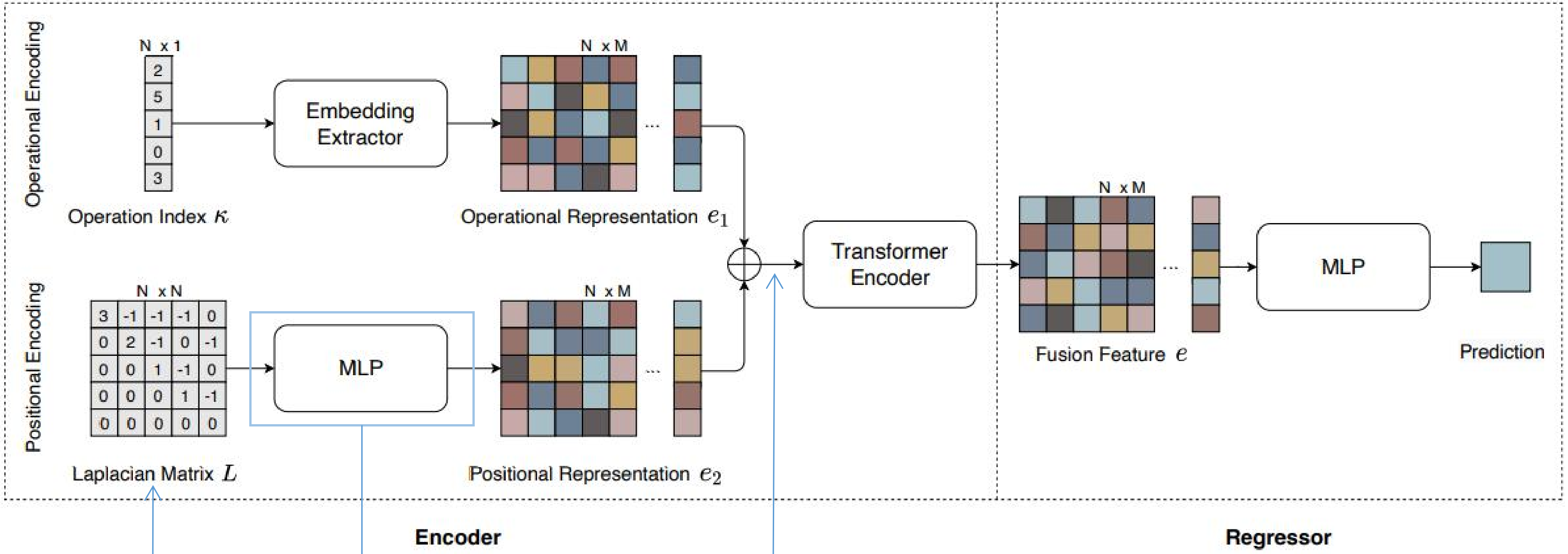
entire deep neural network

Need a unified effective scheme that can encode inputs of various structures and scales

Motivation

The existing mothod that also try to introduce the transformer to neural network representation learning:

Thanks to the powerful capabilities of the transformer, this method achieves promising performance.
But, ...



Need to be paded

Not a pure sequence

The transformer's ability to handling sequence inputs is not fully utillzed.

[1] Shun Lu, Jixiang Li, Jianchao Tan, Sen Yang, and Ji Liu. Tnasp: A transformer-based nas predictor with a selfevolution framework.

Motivation

JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

Others...

Is there a more gradual and reasonable method ?

Existing methods generate final representations by directly summing or averaging the features of all nodes, which are very concise and popular approaches.
However, these approaches may lose information during the rapid compression process.

How to make better use of the data at hand ?

The amount of training architecture-attribute data pairs has a significant impact on the effectiveness of the model.
However, attributes of architectures are usually expensive to acquire.

Proposed Method: NAR-Former

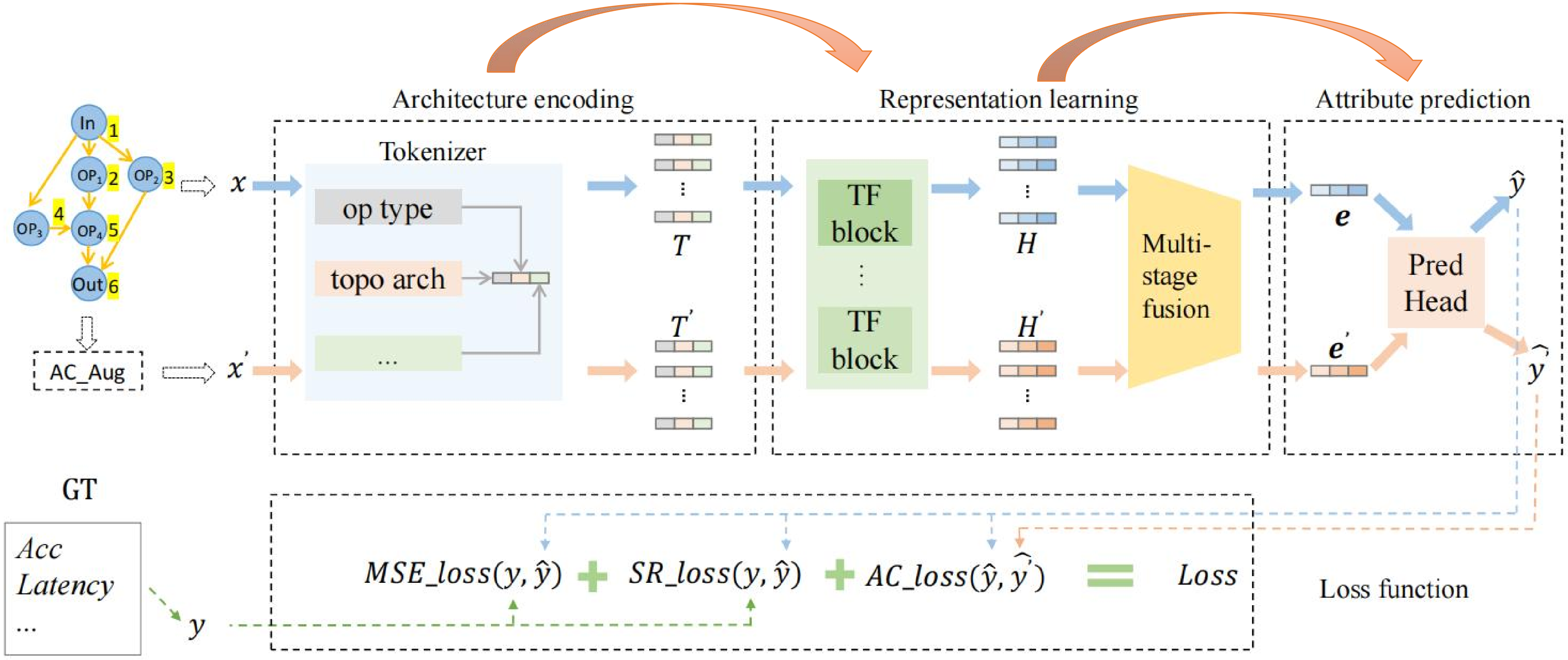
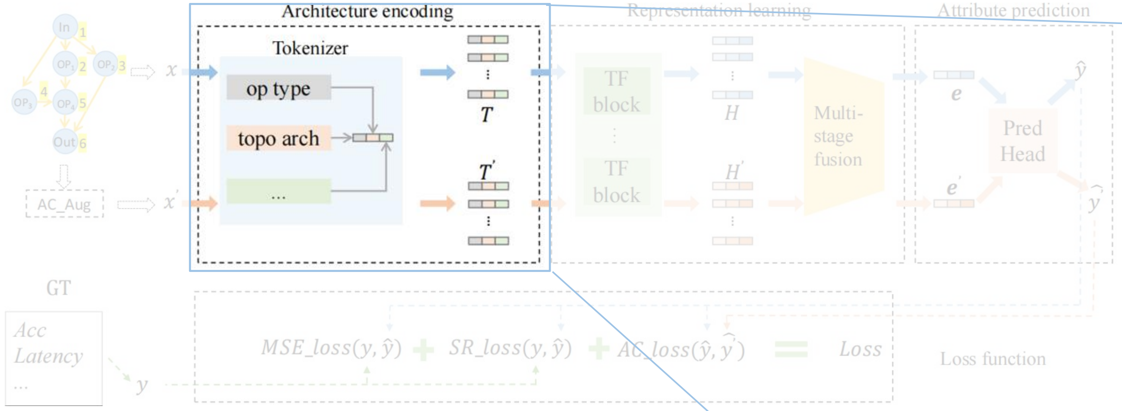


Figure 1. Overview of our NAR-Former. We first encode an input architecture x to a pure sequence T with a proposed tokenizer. A multi-stage fusion transformer is designed to learn a vector representation $\boldsymbol{e}$ from T. x'(optional) is an augmented architecture of x generated by our information flow consistency augmentation. The bottom part shows the loss function used in this paper. SR_loss is designed for learning more accurate sequence ranking. AC_loss is a proposed architecture consistency loss.

Proposed Method: NAR-Former



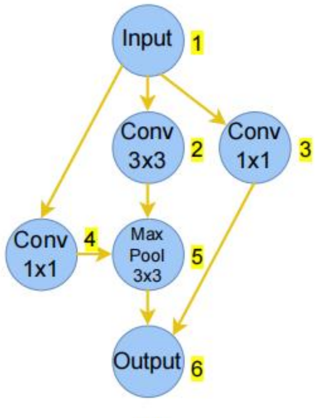
$$s = \frac{2^{L-1} - 2^{0}}{L-1}$$

PE ⇒

$$\boldsymbol{b} = (2^{0}, 2^{0} + s, \cdots, 2^{0} + (L-2)s, 2^{L-1}),$$

$$f(p) = [\sin(b_1 p\pi), \cos(b_1 p\pi), \cdots, \ \sin(b_L p\pi), \cos(b_L p\pi)]\,.$$

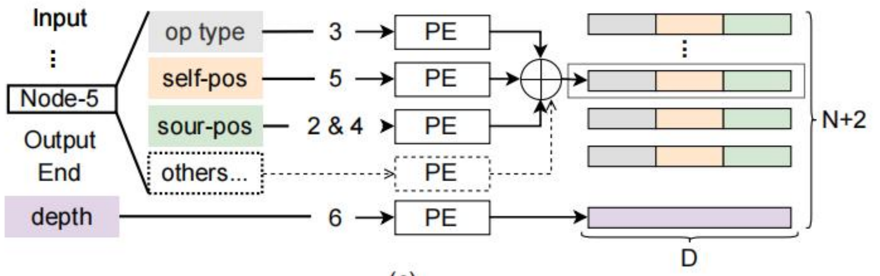
Architecture Encoding Scheme



(a)

Input	0
Conv 1x1	1
Conv 3x3	2
MaxPool 3x3	3
...	...
End	1e5

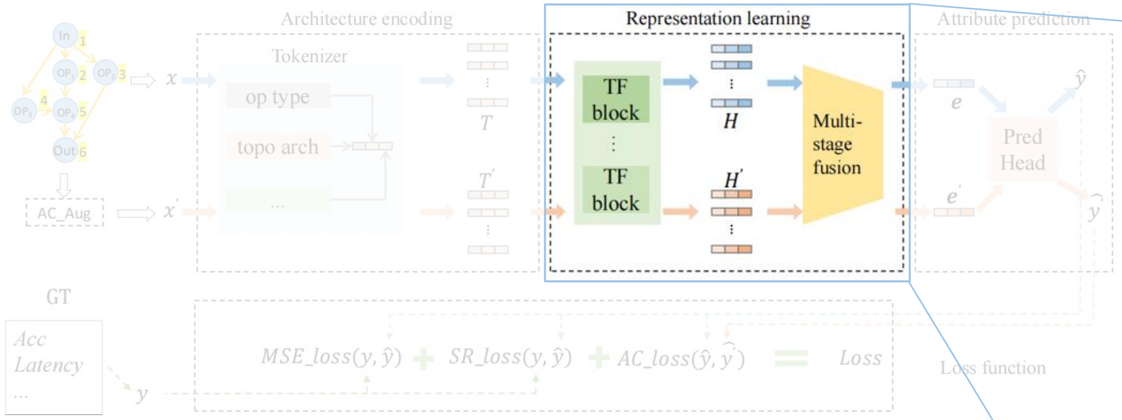
(b)



(c)

Figure 2. (a) An example of architecture with 6 operations($N = 6$). (b) Conversion table from operation categories to indexes. (c) Encoding scheme of our tokenizer.

Proposed Method: NAR-Former



Multi-Stage Fusion Transformer

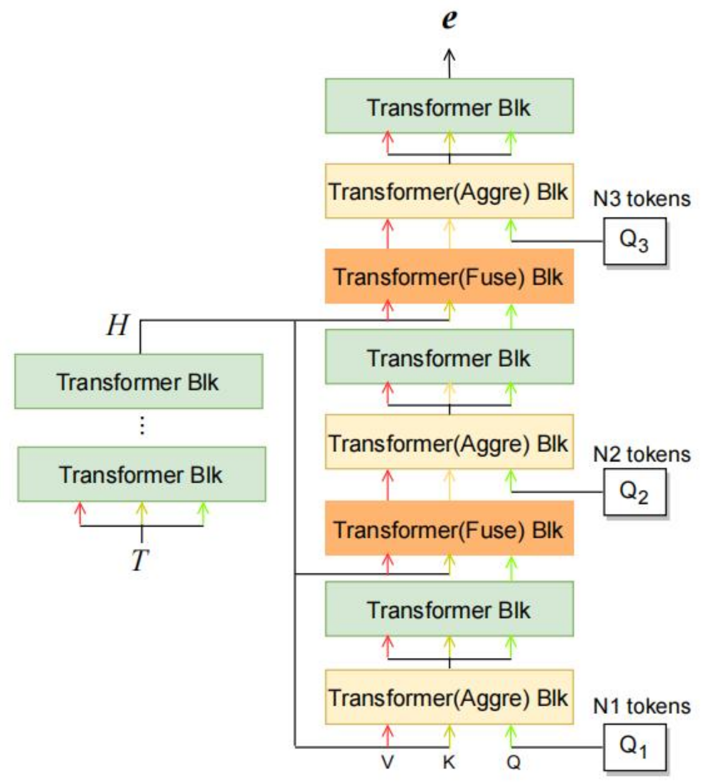
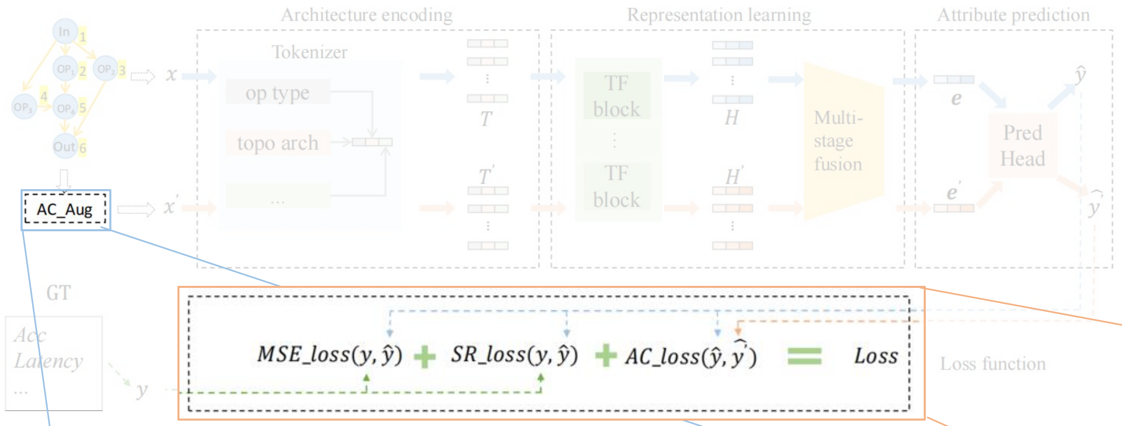


Figure 3. Multi-Stage Fusion Transformer. Token sequence T is first transformed into feature map H by standard transformer blocks, and then gradually fused into a one-token feature vector $\boldsymbol{e}$.

Proposed Method: NAR-Former



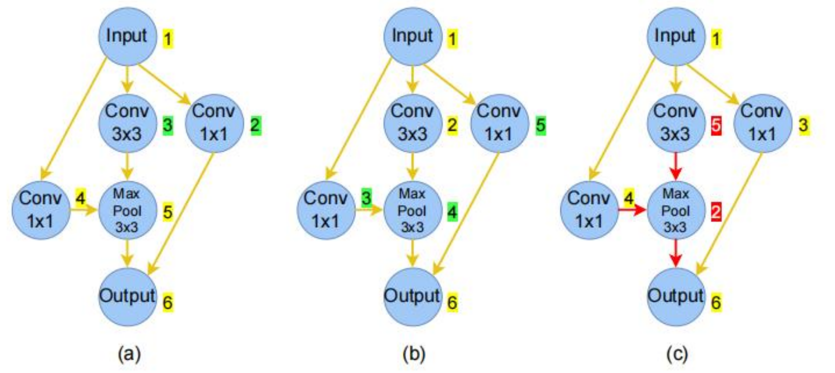


Figure 4. Isomorphic examples of Fig. 2(a). (a) and (b) are augmented architectures that maintain the consistency of information flow, while (c) is not.

Information Flow Consistency Augmentation

$$\mathrm{Loss} = \mathrm{MSE\_loss} + \lambda_1 \mathrm{SR\_loss} + \lambda_2 \mathrm{AC\_loss}.$$

$$\mathrm{AC\_loss}(\widehat{y}, \widehat{y'}) = \sum_{i=1}^{M} |\widehat{y_i} - \widehat{y'_i}|$$

Architecture Consistency Loss

$$\mathrm{SR\_loss}(y, \widehat{y}) = \sum^{M} [(\widehat{y}_{I(i)} - \widehat{y}_i) - (y_{I(i)} - y_i)]$$

Sequence Ranking Loss

$$\mathrm{MSE\_loss}(y, \widehat{y}) = \sum_{i=1}^{M} (\widehat{y}_i - y_i)^2$$

Loss Function

Experiments

JUNE 18-22, 2023 CVPR VANCOUVER, CANADA

Table 1. Results on NAS-Bench-101 [40](depth=2∼7). Kendall's Tau is calculated using predicted accuracies and ground-truth accuracies. Three different proportions of the whole dataset are used as the training set. "SE" refer to self-evolution proposed by TNASP [23] to improve prediction performance.

Backbone	Method	Training Samples 0.1% (424)	0.1% (424)	1% (4236)	5% (21180)
		Test Samples 100	all	all	all
CNN	ReNAS [39]	0.634	0.657	0.816	-
LSTM	NAO [24]	0.704	0.666	0.775	-
	NAO+SE	0.732	0.680	0.787	-
GCN	NP [36]	0.710	0.679	0.769	-
	NP + SE	0.713	0.684	0.773	-
	CTNAS [4]	0.751	-	-	-
Transformer	TNASP [23]	0.752	0.705	0.820	-
	TNASP + SE	0.754	0.722	0.820	-
	NAR-Former	**0.801**	**0.765**	**0.871**	**0.891**

↑4.7%(avg)

Table 2. Results of on NAS-Bench-201 [10](depth=8). Kendall's Tau is calculated using predicted accuracies and ground-truth accuracies. Three different proportions of the whole dataset are used as the training set. "SE" refer to self-evolution proposed by TNASP [23] to improve prediction performance.

Backbone	Model	Training Samples (781) 5%	(1563) 10%	(7812) 50%
LSTM	NAO [24]	0.522	0.526	-
	NAO + SE	0.529	0.528	-
GCN	NP [36]	0.634	0.646	-
	NP + SE	0.652	0.649	-
Transformer	TNASP [23]	0.689	0.724	-
	TNASP + SE	0.690	0.726	-
	NAR-Former	**0.849**	**0.901**	**0.947**

↑16.7%(avg)

Experiments

Table 3. Performance of searched architectures using different NAS algorithms in DARTS [19] space on CIFAR-10 [16]. † denotes using cutout [9] as data augmentation.

Model	Params (M)	Top1 Acc(%)	No. of archs	Search Cost(G·D)
VGG-19 [43]	20.0	95.10	0	0
DenseNet-BC [13]	25.6	96.54	0	0
Swin-S [22]	50	94.17	0	0
Nest-S [42]	38	96.97	0	0
Ransom search	3.2	96.71	-	-
NASNet-A† [44]	3.3	97.35	20000	1800
AmoebaNet-A† [29]	3.2	96.66	27000	3150
PNAS [18]	3.2	96.59	1160	225
NAONet [24]	28.6	97.02	1000	-
GATES† [26]	4.1	97.42	800	-
ENAS† [27]	4.6	97.11	-	0.5
DARTS† [19]	3.4	97.24	-	4
CTNAS† [4]	3.6	97.41	-	0.3
TNASP† [23]	3.7	97.48	1000	0.3
NAR-Former†	3.8	**97.52**	**100**	0.24

Table 7. Verification of predictor's effectiveness using neural structure search experiments on MobileNet space.

Model	ImageNet Top1(%)	MACs	Search Cost (GPU hours)
OFA [1]	76.00	230M	40
NAR-Former	76.36 ↑0.36%	378M	1.63
NAR-Former	76.90 ↑0.9%	571M	2.00

Mobile setting

Table 6. Latency prediction on NNLQP [20]. "Test Model" denotes the model type that used as test set.

Test Model	Method	MAPE↓	ACC(10%)↑
EfficientNet depth=242	FLOPs	58.36%	0.05%
	TPU [14]	**16.74%**	17.00%
	NNLQP [20]	21.33%	**24.65%**
	NAR-Former	28.05%	24.08%
Nas-Bench-201 depth=112∼247	FLOPs	80.41%	0.00%
	TPU [14]	58.94%	2.50%
	NNLQP [20]	8.76%	67.10%
	NAR-Former	**4.19%**	**95.12%**

Experiments

Table 5. Ablation study. The "Self_ID" represents the way in which the self position information is combined with other information. The number in parentheses indicates the amount of augmented data. All experiments follow the setting of 1% proportion in Sec. 4.2.

Row	Architencture Encoder	Predictor Type	Self_ID	SR_loss	GI Aug [38] (+3812)	Our Aug (+2421)	AC_loss	Kendall's Tau
1	TNASP [23]	Transformer in [23]	-	-	-	-	-	0.8200
2	NP [36]	GCN in [36]	-	-	-	-	-	0.7694
3	Tokenizer	Transformer in [23]	Add	-	-	-	-	0.8416
4	Tokenizer	Transformer in [23]	Concat	-	-	-	-	0.8477
5	Tokenizer	GCN in [36]	Concat	-	-	-	-	0.7953
6	Tokenizer	Multi-stage fusion	Concat	-	-	-	-	0.8481
7	Tokenizer	GCN in [36]	Concat	-	-	✓	-	0.8035
8	Tokenizer	GCN in [36]	Concat	-	-	✓	✓	0.8060
9	Tokenizer	Multi-stage fusion	Concat	✓	-	-	-	0.8495
10	Tokenizer	Multi-stage fusion	Concat	✓	✓	-	-	0.8625
11	Tokenizer	Multi-stage fusion	Concat	✓	✓	-	✓	0.8643
12	Tokenizer	Multi-stage fusion	Concat	✓	-	✓	-	0.8579
13	Tokenizer	Multi-stage fusion	Concat	✓	-	✓	✓	0.8712

Conclusion

We propose an effective neural architecture representation learning framework that are consisted of linearly scaling network encoders, a transformers based representation learning model, and an effective model training method with data augmentations and assisted loss functions.

Experiments show that our framework are capable of improving the accuracy of downstream prediction tasks while overcoming scale limitations on input architectures.

Although not the scope of this work, we believe this framework can also be extended for other down stream tasks, such as predicting the quantization loss or searching for the best mixed precision model inference strategies.

Thanks for your listening!

IIP Lab: https://iip-xdu.github.io

Intellifusion: https://www.intellif.com/

Codes link: https://github.com/yuny220/NAR-Former