6Former: Transformer-Based IPv6 Address Generation

1st Qiankun Liu
Institute for Network Sciences and Cyberspace
Tsinghua University
Beijing, China
lqk20@mails.tsinghua.edu.cn

2nd Xing Li
Institute for Network Sciences and Cyberspace
Tsinghua University
Beijing, China
xing@cernet.edu.cn

***Abstract*—Active network scanning in IPv6 is hindered by the vast address space of IPv6. Researchers have proposed various target generation methods, which are proved effective for reducing scanning space, to solve this problem. However, the current landscape of address generation methods is characterized by either low hit rates or limited applicability. To overcome these limitations, we propose 6Former, a novel target generation system based on Transformer. 6Former integrates a discriminator and a generator to improve hit rates and overcome usage scenarios limitations. Our experimental findings demonstrate that 6Former improves hit rates by a minimum of 38.6% over state-of-the-art generation approaches, while reducing time consumption by 31.6% in comparison to other language model-based methods.**

***Index Terms*—IPv6, network scanning, Transformer**

I. Introduction

Active network scanning is an essential means of network research. It can be used for network service measurement, network topology discovery, network security, vulnerability analysis, and so on. Active network measurement has made great progress in IPv4, which is due to the large-scale and rapid search under IPv4 addresses.

With the exhaustion of IPv4 addresses, the Internet has inexorably turned to IPv6. According to Google's data, nearly 37% of users visit their websites through IPv6 and the number is gradually increasing [1]. The adoption rate of IPv6 on Alexa's top 1 million websites has exceeded 19.9%. Various trends show that IPv6 is being deployed on a large scale and widely used [2]. Faced with such a large-scale deployment, users do not know much about the actual application of IPv6 address space. This is because IPv6 has a huge address space, and large-scale detection like IPv4 cannot be realized in IPv6. It takes millions of years to detect the address space of IPv6 by brute-force.

To mitigate this challenge, researchers often gather a set of known active IPv6 addresses, commonly referred to as seeds, and learn their salient features to generate candidate addresses with higher probability of being active for subsequent scanning. Despite the previous research efforts in active IPv6 address detection, the comprehensive global discovery of active IPv6 addresses still presents a set of challenges, particularly in the following dimensions:

1) **Low hit rates** Despite attempts made by existing methods at both statistical and semantic levels, they only leverage a subset of the inter-address and intra-address information available. This limited exploration has resulted in a relatively low hit rate in address generation.
2) **Limited applicability.** The efficiency of current approaches to active IPv6 address probing is constrained in regions where seed addresses are absent. This is attributed to the necessity for these methods to discern the properties of the seed addresses for generating target addresses with a higher likelihood of being active. Therefore, the scarcity of seed addresses in certain regions poses a challenge to the effective execution of active IPv6 address probing.

We drew inspiration from natural language processing techniques and developed a language model-based approach for address generation. Our proposed system, 6Former, consists of a discriminator and a generator, and utilizes a Transformer architecture in the generator to effectively capture the information contained within addresses. We employed a pre-processing step to construct the language model, and manually curated a series of datasets to facilitate model training and address generation. To fine-tune the generated addresses, we introduced temperature and similarity as adjustable parameters.

Contribution

1) We propose a method for constructing an IPv6 language model and elucidate it from a mathematical perspective.
2) We train an address generation model that can be fine-tuned for different downstream tasks. In this work, we apply the model to generate active IPv6 addresses.
3) We introduce an active address generation system based on generation model structure, which outperforms traditional methods by 38.6% and deep learning methods by 63.9% in hit rate, while achieving a 31.6% reduction in time consumption. We conduct a premilinary analysis on generated addresses.

Roadmap In Section II, we introduce the necessary background knowledge. In Section III, we review related work and highlight our approach. In Section IV, we describe the designation of our method. In Section V, we evaluate our method's performance. We conclude this paper and discuss the future work in Section VI.

II. Background

In this section, we present a concise overview of the fundamental knowledge on IPv6 addresses and Transformer models. This is intended to provide a solid foundation for comprehending the techniques of address encoding and model construction.

A. IPv6 Address

An IPv6 address consists of 128 bits of binary digits, which are divided into eight groups of 16 bits each. Each group is represented by a hexadecimal number derived by converting 4 bits of binary to one digit of hexadecimal,which is called nybble, and the groups are separated by colons. For example, 2001:0db8:85a3:0000:0000:8a2e:0370:7334 is a typical IPv6 address.

An IPv6 address is composed of a global routing prefix, a local subnet identifier, and an interface identifier (IID). The global routing prefix serves to route traffic to a Local Area Network (LAN), while the configuration of the IID is more flexible to guarantee the uniqueness of the host interface in the local network segment. RFC 7707 categorizes IIDs into various patterns, briefly shown as follows [3]:

1) **Low-Byte.**The most common form of low-byte addresses is that in which all the bytes of the IID are set to zero.
2) **Embedded-IPv4.** The most common form of these addresses is that in which an IPv4 address is encoded in the lowest-order 32 bits of the IPv6 address.
3) **Embedded-Port.**Addresses following this pattern include the service port in the lowest-order byte of the IID and have the rest of the bytes of the IID set to zero.
4) **Randomized.** IID generated through Stateless Address Autoconfiguration (SLAAC) often employs a pseudorandom representation.

The complex address configuration methods need to be classified, and the address generation method needs to utilize these patterns to narrow down the scope of address probing. Some common configuration methods, along with examples and the corresponding proportion used in a network address measurement, are briefly presented in the table I.

TABLE I
Patterns Example and Ratio

Patterns	Example	Ratio(%)
Low-byte	2001:db8::1	26.12
Embedded-IPv4	2001:db8::192:0:2:1	7.43
Embedded-Port	2001:db8::1:80	0.12
Other	2001:db8:85a3::8a2e:0370:7334	66.33

B. Transformer

The Transformer model [4] is a prevalent deep learning architecture in natural language processing, which was originally introduced by the Google research team for various language tasks, including machine translation [5], text generation [6], and question answering systems [7]. Its unique design integrates attention mechanisms, which distinguishes it from prior recurrent neural network (RNN) based models by enabling it to efficiently process longer input sequences, while achieving superior parallel computing performance.

The Transformer's architecture is comprised of an encoder and a decoder, both of which comprise several identical modules known as Transformer modules. Each module incorporates attention mechanisms and fully connected neural networks to learn the input sequence's underlying dependencies, enhancing the model's capacity to extract relevant features. The self-attention mechanism allows the model to selectively concentrate on different input positions and features, further boosting its representational power.

III. Related Work

A. Address Pattern Mining

Various studies have explored the characteristics of IPv6 address space to facilitate the generation of target addresses. RFC 7707 has identified common address configuration schemes and administrator configuration customs. Gasser et al. used entropy clustering to classify known addresses into six addressing pattern categories, which showed a strong correlation with the configuration schemes [8]. Cui et al. proposed IPv62Vec [9] to learn address similarity and explore IPv6 semantics. This method clusters addresses with similar semantic structures in the vector space. These previous works have provided insights into the addressing patterns of the IPv6 space, laying the foundation for generating target addresses in addressing patterns.

B. Target Generation

1) ***Traditional Algorithm***:

- In 2015, Ullrich et al. advanced two more comprehensive premises based on Barnes's work [10]: obtaining a greater number of active addresses, either through acquisition from alternate sources or by synthesizing new addresses from available information. Subsequently, Ullrich et al. [11] proposed a novel feature-based recursive address generation method, which outperformed the conventional brute-force generation scheme by unveiling 50% to 100% more novel addresses. However, it is noteworthy that Ullrich et al.'s approach was still conducted on previous address sets, and no augmentation or expansion of the dataset was undertaken.
- In 2016, Foremski et al. proposed Entropy/IP, an algorithm that leverages empirical entropy and Bayesian networks to mine statistical information from seed addresses [12]. Their work highlights the effectiveness of entropy in extracting information from each address bit and the existence of correlation between address segments that can be exploited.
- Murdock et al. proposed a density-based generative probing scheme, 6Gen, which identifies dense regions in the set of seed addresses without prior knowledge of their structure [13]. The scheme is able to detect more active addresses under a limited detection budget, providing a standard for evaluating subsequent generation schemes.

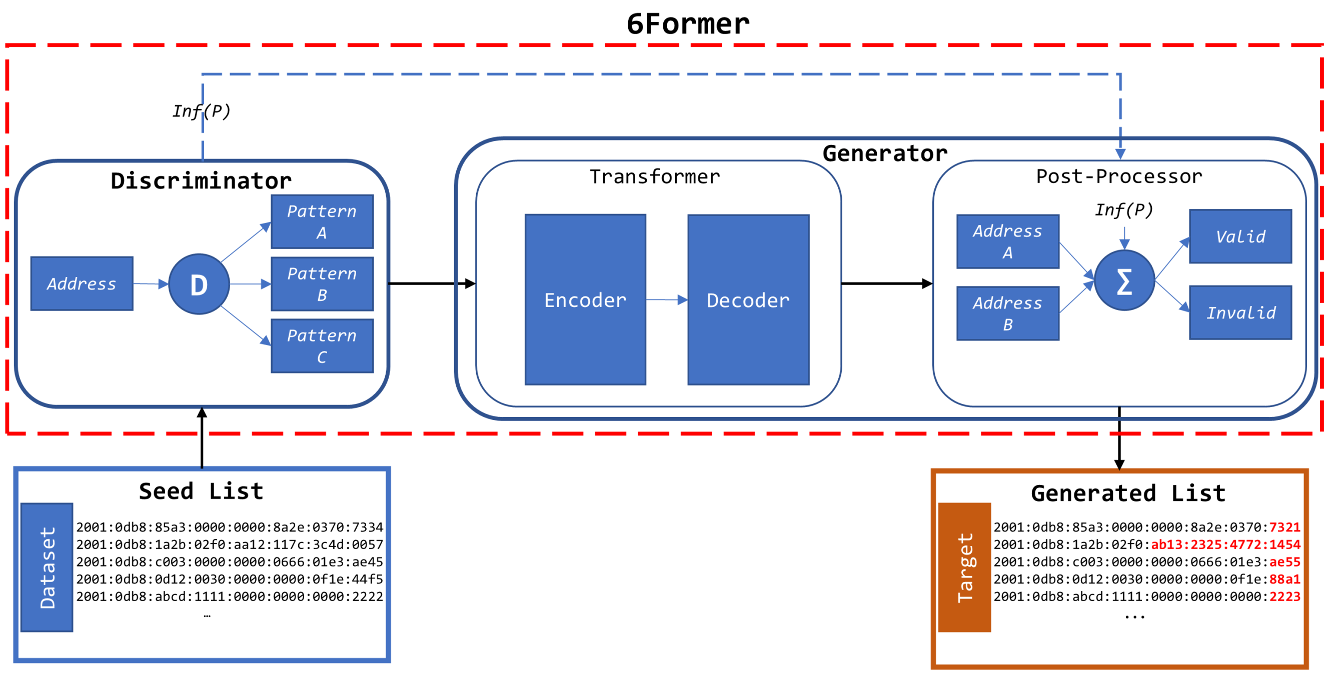
6Former
Inf(P)
Discriminator
Address
D
Pattern A
Pattern B
Pattern C
Generator
Transformer
Encoder
Decoder
Post-Processor
Inf(P)
Address A
Address B
Σ
Valid
Invalid
Seed List
Dataset
2001:0db8:85a3:0000:0000:8a2e:0370:7334
2001:0db8:1a2b:02f0:aa12:117c:3c4d:0057
2001:0db8:c003:0000:0000:0666:01e3:ae45
2001:0db8:0d12:0030:0000:0000:0f1e:44f5
2001:0db8:abcd:1111:0000:0000:0000:2222
…
Generated List
Target
2001:0db8:85a3:0000:0000:8a2e:0370:7321
2001:0db8:1a2b:02f0:ab13:2325:4772:1454
2001:0db8:c003:0000:0000:0666:01e3:ae55
2001:0db8:0d12:0030:0000:0000:0f1e:88a1
2001:0db8:abcd:1111:0000:0000:0000:2223
...

Fig. 1. **Overview of 6Former.** The *black* arrows indicate the flow of addresses.

Evaluation is done using the Hamming distance, and the structure of 6Gen is Agglomerative Hierarchical Clustering (AHC).

- Liu et al. proposed 6Tree, an algorithm that leverages divisive hierarchical clustering (DHC) to construct a spatial tree by partitioning the address vector [14]. Differing from 6Gen, 6Tree utilizes a dynamic address detection method to enhance detection efficiency.
- Song et al. introduced the DET algorithm in 2022 [15], combining the ideas of Entropy/IP and 6Tree. DET uses the DHC architecture to construct a spatial tree and identifies the split point with the smallest entropy. Similar to 6Gen, DET is density-based and assumes that active addresses cluster in high-density areas.

*2) **Deep Learning Algorithm**:*

- Since 2019, machine learning has been applied to address generation. In 2020, Cui et al. [16] introduced deep learning into address generation with their proposed GCVAE model, which uses the variational autoencoder (VAE) [17] to generate new addresses. The GC component in GCVAE refers to the thresholded convolutional neural network introduced by Dauphin et al. [18], which is effective in learning the structure in text data. Cui et al. reported that GCVAE can increase the generation rate by 1.52-1.85 times compared to Entropy/IP. However, it should be noted that the hit rate of Entropy/IP is still higher than that of 6GCVAE. While 6GCVAE is a promising attempt of using deep learning for address generation, its performance is not yet satisfactory.
- Hou et al. were the first to introduce the concept of reinforcement learning to IPv6 active address detection with the development of 6Hit [19]. The algorithm dynamically allocates probing expenditures based on feedback for each area and optimizes subsequent detection direction to high-density areas. To address the issue of potentially losing results with the DHC architecture, 6Hit introduced the concepts of space node slicing and space re-division. Despite these improvements, 6Hit was only able to achieve a hit rate of 11.5% in large-scale detections.
- Cui et al. introduced 6VecLM [9], which utilizes Word2Vec [20] to learn address relationship and generate similar addresses. The limitations of this approach stem from its considerable time investment.
- Cui et al. attempted address generation using a generative adversarial model (GAN) in 2021 [21]. Adversarial generative models are a type of deep learning model that has achieved great success in computer vision. They proposed 6GAN, which integrates seed address classification and confrontation set training. According to their results, 6GAN can exceed 6Tree's hit rate by 1.03-1.33 times. However, the test dataset used is much smaller than the scale of other models, which is a point of debate.

IV. 6Former

A. Overview of 6Former

6Former is an innovative target generation architecture that leverages address pattern recognition. Its architecture, as illustrated in Fig. 1, comprises of multiple components, including a discriminator and an address generator. The discriminator classifies addresses into different patterns and transmits the pattern information to the post-processor. The address generator comprises two essential components: a model that integrates a Transformer encoder and decoder to construct the address model and generate addresses, and a post-processor that performs address validity assessment on the output generated by the model.

B. Discriminator

To begin with, the primary task of the discriminator is to discern among the address patterns. This discernment process can be naturally bifurcated into two categories: first determining the patterns and then segmenting the addresses accordingly, or initially clustering the addresses and subsequently designating the address classes as patterns. The former methodology has the benefit of establishing a limited number of pre-existing patterns. Conversely, the latter approach is more refined but may entail a greater number of patterns that necessitate further learning.

In our research deployment, we adopted the former approach, owing to our empirical observation that deep learning neural networks can unearth more latent information. Consequently, even a rough classification can achieve satisfactory results.

We implemented the classification method outlined by RFC7707 [3] and utilized the addr6 tool from ipv6toolkit to execute the pattern matching. We then classified the seed addresses according to the matching results.

C. Generator

Given a sequence of length T consisting of x_1,x_2,...x_T, where x_T can be regarded as the observation at time step t, the objective of a language model is to estimate the joint probability of the sequence $P(x_1, x_2, ..., x_T)$. In the context of probability theory, the joint probability distribution of a sequence of random variables $x_1, x_2, ..., x_T$ can be computed using the product rule: $P(x_1, x_2, ..., x_T) = \prod_{t=1}^{T} P(x_t|, x_1, ..., x_{t-1})$. In a first-order Markov process, the joint probability distribution can be approximated as a product of conditional probabilities $P(x_1, x_2, ..., x_T) = \prod_{t=1}^{T} P(x_t|x_{t-1})$, assuming that the probability of the first variable x_1 is independent of the initial state x_0. Recurrent neural networks (RNNs) [22] utilize hidden state models to capture the state of the previous t-1 time steps, leading to a conditional probability distribution $P(x_t|, x_1, ..., x_{t-1}) \approx P(x_t|h_{t-1})$. However, the RNN decoder suffers from the vanishing gradient problem and has the same contextual variable in each decoding step, which limits its ability to model long- and short-term dependencies effectively.

To address these limitations, the attention mechanism is introduced, which weights the output based on input positions. Specifically, the attention mechanism computes attention weights for each input position, and the output is a weighted sum of the encoder outputs. This allows the decoder to selectively focus on different parts of the input sequence, and learn more flexible and complex dependencies. In the following subsections, we will provide a detailed description

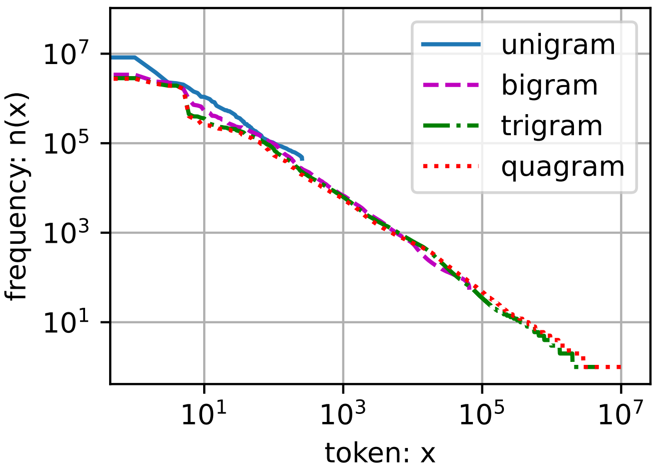

Fig. 2. Frequency for tokens

of how the Transformer model with attention mechanism is used to generate IPv6 addresses.

1) Word Building: We propose a methodology for modeling IPv6 addresses, which involves pre-processing the address set by tokenizing the address strings and constructing a vocabulary that maps tokens to numerical indices. The resulting numerical indices represent a sequence of observations that are amenable to further analysis and modeling.

Specifically, we employ a tokenization scheme that combines every two nybbles into one token, reducing the sequence length to 16 and yielding a vocabulary size of 256. This tokenization approach balances the trade-off between sequence length and vocabulary size.

The constructed vocabulary exhibits a Zipfian distribution, shown in Fig.2, as evidenced by the approximately linear relationship between word frequency and their rank in a log-log plot. As a consequence, traditional statistical approaches that rely on frequency counts and smoothing may leading to biased estimates of tail words. Therefore, more sophisticated modeling techniques that account for the long tail nature of the vocabulary distribution are required.

2) Model Building: We have opted for the encoder-decoder architecture of the Transformer, a state-of-the-art neural network model in natural language processing. The encoder component of this architecture is composed of stacked attention modules, forming a series of identical layers with two sub-layers each. The first sub-layer is responsible for multi-head self-attention, while the second sub-layer implements a position-wise feedforward network. The input sequence is first transformed into an embedding representation and then augmented with positional encoding before being fed into the encoder layer. The output of each encoder layer is a vector that represents each position of the input sequence.

The decoder component of the Transformer architecture shares a similar structure with the encoder, but it includes an additional sub-layer between the two existing ones, the encoder-decoder attention layer. In this sub-layer, the query is derived from the output of the previous decoder layer, while the key and value are obtained from the entire decoder output. During training, the masked multi-head decoder self-attention sub-layer ensures that the query and key/value both come from the output of the previous decoder layer, whereas during prediction, the output sequence is generated one position at a time. This important distinction enables the model to perform robustly in both training and prediction scenarios.

3) Model Training: To begin with, the dataset is partitioned into training, validation, and testing sets. In order to ensure that the model can effectively learn patterns, a training set of one million addresses is generated manually with diverse patterns. The preprocessed address sequence is then segmented into input and target sequences, with each having a fixed length of 16.

Subsequently, the segmented sequence is passed into a Transformer model. Due to the lack of the temporal structure of recurrent neural networks (RNNs) in the Transformer model, sine and cosine functions, which are widely used for positional encoding, are selected to preserve positional information. Moreover, a multi-head attention mechanism with 5 attention heads is employed to facilitate the model in capturing inter-token relationships. Multi-head attention maps the input into multiple subspaces to compute attention vectors, which are subsequently concatenated to form multiple attention vectors. As for the loss function, cross-entropy is adopted to measure the difference between predicted and true results.

4) Target Generation: Upon completion of model training, the trained model can be employed to generate addresses. Active addresses are selected as seeds from the IPv6 Hitlist and pre-processed before being fed into the model. The model generates a probability distribution for each digit, and during the sampling phase, the temperature parameter t is utilized to balance the trade-off between accuracy and diversity. The effect of temperature can be mathematically described by the following equation 1.

$$Pr(i) = \frac{e^{logP(i)^{\frac{1}{t}}}}{\sum_{j=1}^{n} e^{logP(j)^{\frac{1}{t}}}} \tag{1}$$

D. Post-Processor

After sampling and decoding the address into an IPv6 address, a similarity calculation, shown as equation 2, is conducted to ensure that the generated address falls within the pre-specified threshold of similarity to the input address. If the similarity measure fails to meet the threshold criteria, the generated result is discarded. Subsequent experimentation has confirmed its efficiency in significantly enhancing the quality of the generated results.

$$similarity = \sum_{i=0}^{15} dist(ori_i, gen_i) \tag{2}$$

V. Evaluation

A. Experimental Setup

1) Dataset: The dataset utilized in our experiments was sourced from the IPv6 Hitlist, a public repository of IPv6 addresses collected and made available by Gasser et al [8]. We extracted a dataset of 6 million IPv6 addresses generated on February 20th, 2023, for use in our analyses.

To evaluate the performance under limited conditions, often referring to situations where seed density is limited, we constructed a sparse seed set. We employed the k-nearest neighbor (kNN) clustering algorithm [23] to partition the seed address set into clusters, and subsequently reduced the number of addresses in each cluster by 90%. Finally, a seed address set of size 50,000 was obtained by means of random sampling.

2) Verification Method: We have devised a verification method for the generated results, wherein we employed XMap [24] to conduct scanning of the generated addresses. To ensure precision and rigor, we employed a diverse range of protocols, including ICMPv6, TCP/80, TCP/443, among others. The active status of an address was confirmed if it responded during the scan. In order to mitigate any potential errors stemming from singular measurements, we conducted multiple measurements at various time intervals, such as 3 and 7 days. The address was deemed active if it responded at least once during the designated time period, while its continuity across all measurements resulted in its classification as ”3d-active” or ”7d-active,” based on the respective interval.

3) Evaluation Metric:

a) **Hit Rate.** The hit rate can be defined as the ratio of active addresses generated by the model to the total number of addresses generated, and serves as a measure of the model's accuracy in predicting active addresses.

$$r_{Hit} = \frac{N_{Hit}}{N_{total}} \tag{3}$$

b) **Generation Rate.** The generation rate can be defined as the ratio of active addresses generated by the model that are not present in the seed set, to the total number of addresses generated. This measure can be used to assess the diversity of the model's predictions.

$$r_{Gen} = \frac{N_{Gen}}{N_{total}} \tag{4}$$

4) Model Setting: We deployed our model on an NVIDIA 3080Ti GPU, with a Transformer architecture consisting of 3 layers to balance training efficiency and result quality. The number of attention heads was set to 5, and a temperature of 0.01 was used. To demonstrate the effectiveness of post-processing, we denoted the method without post-processing as 6Former-. Two post-processing thresholds were used: (0.5, 0.65) for Randomized and (0.9, 0.99) for the rest.

5) Baselines:

a) **Traditional Design Algorithms.** We selected Entropy/IP [12], 6Gen [13], 6Tree [14], and DET [15] as benchmark algorithms for our study. Entropy/IP, 6Tree, and DET employ entropy information for address space partitioning and spatial tree construction. We used open-source code of Entropy/IP and 6Tree and reproduced the code of 6Gen and DET as baselines.

TABLE II
PERFORMANCES IN 500K PROBES

					Probe = 500k	
Category	Method	N_Hit	N_Gen	r_Hit	r_Gen	time
Traditional	Entropy/IP	16053	11351	3.21%	2.27%	\
	6Gen	54635	9773	10.93%	1.95%	\
	6Tree	63802	21159	12.76%	4.23%	\
	DET	152844	20689	30.57%	4.14%	\
Deep Learning	6GAN	69058	24001	13.81%	4.80%	\
	6GCVAE	49333	16855	9.87%	3.37%	\
	6VecLM	129262	27701	25.85%	5.54%	8h12m
Our Approach	6Former-	155259	36102	31.05%	7.22%	4h44m
	6Former	211843	38892	**42.37%**	**7.78%**	5h37m

TABLE III
PERFORMANCES WITH LIMITED SEEDS IN 50K PROBES

				Probe = 50k	
Category	Method	N_Hit	N_Gen	r_Hit	r_Gen
Traditional	6Gen	1531	843	3.06%	1.69%
	6Tree	2893	1056	5.79%	2.11%
	DET	5881	3412	11.76%	6.82%
Deep Learning	6GAN	5786	609	11.57%	1.22%
	6GCVAE	3563	1286	7.13%	2.57%
	6VecLM	10418	3101	20.84%	6.20%
Our Approach	6Former-	14891	4802	29.78%	9.60%
	6Former	19009	5004	**38.02%**	**10.01%**

b) **Deep Learning Approaches.** We considered 6GCVAE [16], 6VecLM [9], and 6GAN [21] as deep learning-based algorithms for address generation. 6GCVAE leverages the VAE architecture, 6GAN employs reinforcement learning algorithms, and 6VecLM utilizes Word2Vec for word embeddings. We used open-source code for 6VecLM and reproduced the code for 6GCVAE and 6GAN following the algorithm descriptions, and adopted these approaches as baselines.

B. Target Generation

1) Normal Scenario: The Table II exhibits the generation rate and hit rate of various algorithms after 500,000 probes, and reports the training and generation times for deep learning-based approaches. Our method demonstrates a marked improvement over traditional address generation algorithms, including Entropy/IP and 6Gen. Despite being the best-performing traditional algorithm, DET still falls short of our approach in terms of hit rate. Among deep learning method, the performance of 6GAN and 6GCVAE does not meet expectations, possibly because the hyperparameters we reproduce are not the optimal hyperparameters. Compared to 6VecLM, another language model-based approach, our method outperforms in both generation speed and hit rate. Comparing 6Former- and 6Former, it can be found that the added discriminator and post-processor have increased the hit rate by 35.49%. Fig. 3 shows the hit rates of Entropy/IP, DET, 6VecLM and 6Former.

It is worth noting that the results generated by deep learning methods may contain duplicate addresses, the de-duplication is performed before conducting the experiments.

Overall, our approach demonstrates clear superiority over traditional and deep learning-based algorithms for address

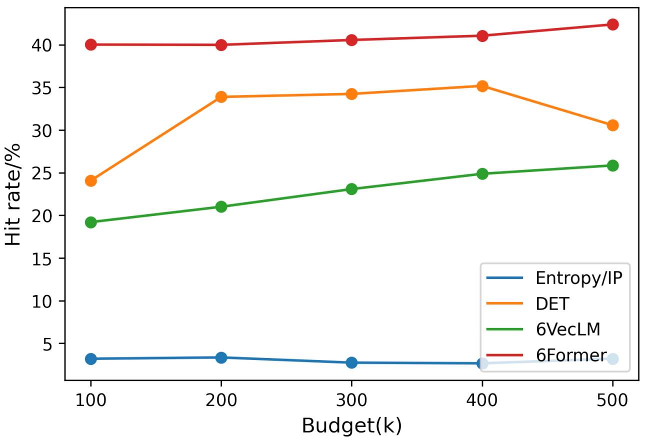

Fig. 3. Hit rate over 500k probes

generation, indicating its efficiency and potential for practical applications.

2) Limited Scenario: The performance of various methods under limited seed addresses, including hit rate and generation rate, is presented in Table III. The results evince that deep learning methods vastly outperform traditional methods in terms of hit rate due to the neural network's capacity to memorize all addresses within the training set during the training process. Traditional methods, on the other hand, underperform due to the sparse address space constructed from sparse addresses, thus resulting in insufficiently dense intervals even with repeated segmentation. Conversely, the ratio of generation rate to hit rate of traditional methods significantly improves due to the reduced density of the seed set. Statistical analysis indicates that the hit rate and generation rate of 6Former in limited seed sets are almost the same as those in normal seed sets, hence addressing the constraints faced by traditional methods in limited application scenarios.

C. Analysis of Active Addresses

In order to gain a deeper understanding of the IPv6 address configuration landscape, we conducted an analysis of the IID (Interface IDentifier) types of active address assignments. Specifically, we utilized the addr6 tool to partition the IID portion of 200,000 generated active addresses into distinct categories.

As is shown in Table IV, our findings indicate that 81.14% of the active addresses were observed to be active for a period of 3 days, while 65.79% remained active for 7 days. Additionally, the Low-Byte and Embedded-IPv4 categories were found to comprise 16.25% and 14.06% of the active addresses, respectively. The remaining categories consisted of Randomized and EUI-64 addresses.

It is worth noting that, across the longitudinal comparison, the proportions of each address category remained relatively stable, without any discernible trend towards a decrease in any specific category. These results provide valuable insights into the global configuration landscape of IPv6 addresses and can

TABLE IV
IIDs Pattern Analysis

	IPs	Low-Byte	Embedded-IPv4	Other
1d-active	200k	32.5k(16.25%)	28.1k(14.06%)	139.4k(69.69%)
3d-active	162k(81.14%)	30.1k(15.05%)	25.5k(12.75%)	106.4k(53.20%)
7d-active	132k(65.79%)	24.2k(12.13%)	19.4k(9.76%)	87.8k(43.90%)

inform the development of more effective address allocation and management strategies.

VI. Conclusion

In this study, we present 6Former, an efficient system for generating active IPv6 addresses. To achieve both high efficiency and accuracy in address generation, 6Former employs a combined architecture of discriminator and generator. 6Former first pre-classifies addresses and employs a language model for character-level language modeling within addresses. The learning and generation processes are conducted using the Transformer's encoder-decoder architecture. Experimental results demonstrate that 6Former outperforms existing methods by a minimum of 38.6% at hit rate and 31.6% reduction at time consumption.

We plan to release our model as a pre-trained language model which can be fine-tuned for various usage scenarios. We are also exploring the potential of augmenting language models with other prominent deep learning models, such as the Diffusion Model.

Acknowledgment

Thanks to Zhang Songyu and Li Rui for their kind help, and thanks to reviewers for their comments and feedback. This research is supported by the R&D project of Tsinghua-Telecom Next Generation Internet Research Center, No. 2022930033.

References

[1] Google, "Ipv6 adoption statistics.." https://www.google.com/intl/en/ipv6/statistics.htmltab=ipv6-adoption, 2022.

[2] W3Techs, "Usage statistics of ipv6 for websites." https://w3techs.com/technologies/details/ce-ipv6, 2022.

[3] F. Gont and T. Chown, "Network reconnaissance in ipv6 networks," tech. rep., 2016.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[5] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. F. Wong, and L. S. Chao, "Learning deep transformer models for machine translation," *arXiv preprint arXiv:1906.01787*, 2019.

[6] Y. Huang, H. Xue, B. Liu, and Y. Lu, "Unifying multimodal transformer for bi-directional image and text generation," in *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 1138–1147, 2021.

[7] X. Zhao, F. Xiao, H. Zhong, J. Yao, and H. Chen, "Condition aware and revise transformer for question answering," in *Proceedings of The Web Conference 2020*, pp. 2377–2387, 2020.

[8] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczyński, S. D. Strowes, L. Hendriks, and G. Carle, "Clusters in the expanse: understanding and unbiasing ipv6 hitlists," in *Proceedings of the Internet Measurement Conference 2018*, pp. 364–378, 2018.

[9] T. Cui, G. Xiong, G. Gou, J. Shi, and W. Xia, "6veclm: Language modeling in vector space for ipv6 target generation," in *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part IV*, pp. 192–207, Springer, 2021.

[10] R. Barnes, R. Altmann, and D. Kerr, "Mapping the great void: Smarter scanning for ipv6," *Proc. CAIDA AIMS-4*, 2012.

[11] J. Ullrich, P. Kieseberg, K. Krombholz, and E. Weippl, "On reconnaissance with ipv6: a pattern-based scanning approach," in *2015 10th International Conference on Availability, Reliability and Security*, pp. 186–192, IEEE, 2015.

[12] P. Foremski, D. Plonka, and A. Berger, "Entropy/ip: Uncovering structure in ipv6 addresses," in *Proceedings of the 2016 Internet Measurement Conference*, pp. 167–181, 2016.

[13] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson, "Target generation for internet-wide ipv6 scanning," in *Proceedings of the 2017 Internet Measurement Conference*, pp. 242–253, 2017.

[14] Z. Liu, Y. Xiong, X. Liu, W. Xie, and P. Zhu, "6tree: Efficient dynamic discovery of active addresses in the ipv6 address space," *Computer Networks*, vol. 155, pp. 31–46, 2019.

[15] G. Song, J. Yang, Z. Wang, L. He, J. Lin, L. Pan, C. Duan, and X. Quan, "Det: Enabling efficient probing of ipv6 active addresses," *IEEE/ACM Transactions on Networking*, 2022.

[16] T. Cui, G. Gou, and G. Xiong, "6gcvae: Gated convolutional variational autoencoder for ipv6 target generation," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 609–622, Springer, 2020.

[17] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[18] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning*, pp. 933–941, PMLR, 2017.

[19] B. Hou, Z. Cai, K. Wu, J. Su, and Y. Xiong, "6hit: A reinforcement learning-based approach to target generation for internet-wide ipv6 scanning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10, IEEE, 2021.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[21] T. Cui, G. Gou, G. Xiong, C. Liu, P. Fu, and Z. Li, "6gan: Ipv6 multi-pattern target generation via generative adversarial nets with reinforcement learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pp. 1–10, IEEE, 2021.

[22] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[23] P. Soucy and G. W. Mineau, "A simple knn algorithm for text categorization," in *Proceedings 2001 IEEE international conference on data mining*, pp. 647–648, IEEE, 2001.

[24] X. Li, B. Liu, X. Zheng, H. Duan, Q. Li, and Y. Huang, "Fast ipv6 network periphery discovery and security implications," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 88–100, IEEE, 2021.