A Simple Survey of AIGC

1st Yichi Zhang

Zhejiang University
Hangzhou, China
yichics02@gmail.com

***Abstract*—AIGC refers to Artificial Intelligence Generated Content, which is generated automatically or collaboratively using artificial intelligence techniques in response to user input or demand, including text, images, audio, video, and other forms of content. In this survey, the basic principles of Transformer and diffusion models are presented. The GPT and BERT models based on Transformer are introduced, as well as three different types of diffusion models and their practical applications. Finally, the security and robustness issues of AIGC models are discussed.**

***Index Terms*—AIGC, Transformer, Diffusion, Security, Robustness**

I. Introduction

This survey will cover the following contains: First, we will focus on explaining the basic principles of the Transformer [1], followed by a discussion of the details of language models (LLMs) that use the Transformer, such as BERT [2], GPT [3]–[6], etc. We will also cover some techniques for training and optimizing these models. Finally, we will discuss the security and robustness of GPT models, respectively, based on recent research findings.

II. Transformer

Generative models have a long history in artificial intelligence, dating back to the 1950s with the development of Hidden Markov Models (HMMs) [7] and Gaussian Mixture Models (GMMs) [8]. These models generated sequential data such as speech and time series. However, it wasn't until the advent of deep learning that generative models saw significant improvements in performance. In early years of deep generative models, different areas do not have much overlap in general. In natural language processing (NLP), a traditional method to generate sentences is to learn word distribution using N-gram language modeling [9] and then search for the best sequence. However, this method cannot effectively adapt to long sentences. To solve this problem, recurrent neural networks (RNNs) [10] were later introduced for language modeling tasks, allowing for modeling relatively long dependency. This was followed by the development of Long Short-Term Memory (LSTM) [11] and Gated Recurrent Unit (GRU) [12], which leveraged gating mechanism to control memory during training. These methods are capable of attending to around 200 tokens in a sample [13], which marks a significant improvement compared to N-gram language models.

Meanwhile, in computer vision (CV), before the advent of deep learning-based methods, tra- ditional image generation algorithms used techniques such as texture synthesis [14] and texture mapping [15]. These algorithms were based on hand-designed features, and were limited in their abil- ity to generate complex and diverse images. In 2014, Generative Adversarial Networks (GANs) [16] was first proposed, which was a significant milestone in this area, due to its impressive results in various applications. Variational Autoencoders (VAEs) [17] and other methods like diffusion generative models [18] have also been developed for more fine-grained control over the image generation process and the ability to generate high-quality images.

The advancement of generative models in various domains has followed different paths, but eventually, the intersection emerged: the transformer architecture [1]. The Transformer was initially developed to handle NLP tasks, but it has also been applied to tasks in computer vision. Its various variants for different tasks quickly achieved state-of-the-art performance. In the field of NLP, many prominent large language models, e.g., BERT and GPT, adopt the transformer architecture as their primary building block. In CV, Vision Transformer (ViT) [19] and Swin Transformer [20] later takes this concept even further by combining the transformer architecture with visual components, allowing it to be applied to image based downstreams. Except for the improvement that transformer brought to individual modalities, this intersection also enabled models from different domains to be fused together for multi-modal tasks. One such example of multimodal models is CLIP [21]. CLIP is a joint vision-language model that combines the transformer architecture with visual components, allowing it to be trained on a massive amount of text and image data. Since it combines visual and language knowledge during pre-training, it can also be used as image encoders in multimodal prompting for generation. In all, the emergence of transformer based models revolutionized AI generation and led to the possibility of large-scale training.

A. Model Architecture of Transformer

Here is a diagram of the Transformer model architecture. We will now explain this architecture by focusing on several modules: Encoder-Decoder [22], Self-Attention, Position-wise Feed-Forward Network, Embeddings, and Softmax.

1) Encoder-Decoder: The Encoder consists of N layers, each of which contains two sub-layers. The first sub-layer includes a Multi-Head Attention module, and the second sub-layer includes a position-wise fully connected feed-forward

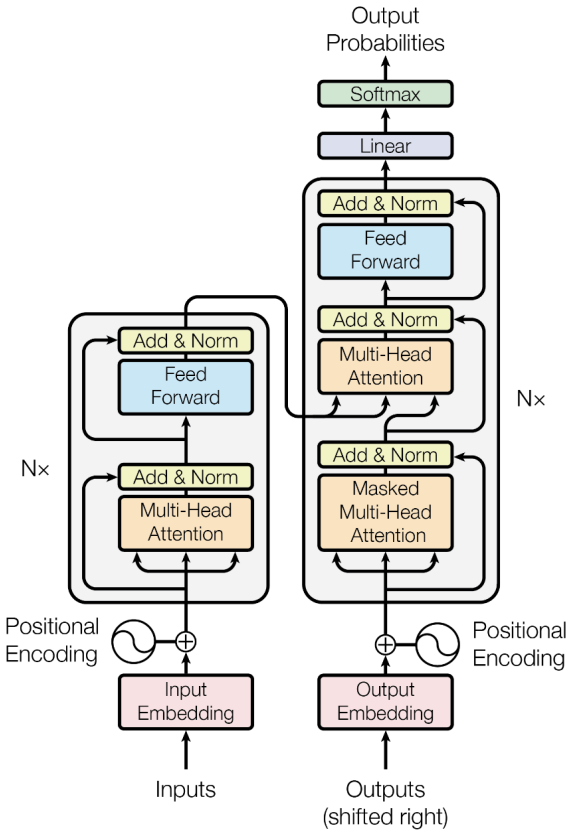

Fig. 1. Model Architecture of Transformer

network. In each sub-layer, we apply a residual connection [23] module, followed by layer normalization [24].

The Decoder also consists of N independent layers, but unlike the Encoder, each layer of the Decoder contains three sub-layers, which has three different kinds of modules: Masked Multi-Head Attention, Multi-Head Attention, and position-wise fully connected feed-forward network. Similarly, we apply the same residual connection and layer normalization after each sub-layer.

This results in the following equation for each sub-layer:

$$Output(x) = LayerNorm(x + SubLayer(x))$$

where $Sublayer(x)$ is the function implemented by the sub-layer itself.

2) Attention: An attention function can be described as mapping a query and a key-value pair to a set of outputs, where all of these quantities are vectors. This allows the output to be viewed as a weighted sum of values, where each value's weight can be seen as calculated based on the query and its corresponding key.

From Fig2, we can see that the Multi-Head Attention used in the model consists of multiple Scaled Dot-Product Attention modules. Therefore, let's focus on Scaled Dot-Product Attention first.

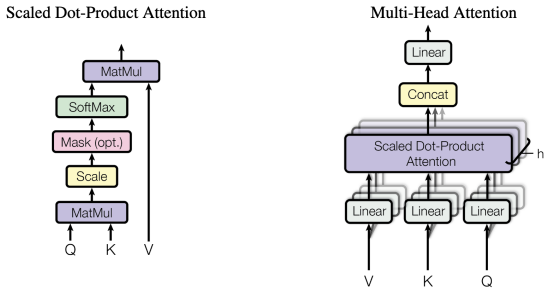

Fig. 2. Attention Mechanism

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Compared to regular dot-product attention, Scaled Dot-Product Attention includes a scaling factor $\sqrt{d_k}$ [25] in the equation to improve the performance of the softmax function on larger datasets.

Next, let's consider how to apply Scaled Dot-Product Attention to Multi-Head Attention. For a set of K, Q, V, each with a d_{model} dimension, they are projected to different dimensions (d_k, d_k, and d_v, respectively). Then, attention calculations are performed in parallel on the projected versions of the Q, K, and V, resulting in an output vector with a dimension of d_v.

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h)W^O$$
$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$
$$W_i^Q \in \mathbb{R}^{d_{model} \times d_k},\ W_i^K \in \mathbb{R}^{d_{model} \times d_k}$$
$$W_i^V \in \mathbb{R}^{d_{model} \times d_v},\ W^O \in \mathbb{R}^{hd_v \times d_{model}}$$

Where h is the number of attention layers that are computing in parallel.

Note that in Fig2, there is also an optional mask module used to mask out certain invalid positions or information when calculating attention scores. This is done to prevent these invalid positions or information from being considered when calculating attention weights.

3) Position-wise Feed-Forward Networks: In each layer of the Encoder or the Decoder, there is a fully connected feed-forward network that is applied separately and identically to each position. This network consists of two layers of ReLU activation functions combined together.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

Of course, different layers may use different parameters, and there may be changes in dimensions caused by modules such as convolutional layers between layers.

4) Embeddings and Softmax: Like other sequence processing models, the Transformer also uses a learnable embedding to convert input and output tokens into d_{model}-dimensional vectors. It also uses a learnable linear transformer and softmax

function to convert the decoder's output into predicted probabilities for the next token.In the embedding layers, multiply those weights by $\sqrt{d_{model}}$.

5) Positional Encoding: Since the Transformer does not contain convolutional or recurrent components, position encoding is required to incorporate positional information of the input sequence. Here, the position encoding uses the same dimension as the embedding to enable addition of the two. Given that there are various forms of position encoding, both learnable and fixed, the Transformer uses sine and cosine functions of different frequencies to encode position information.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

where pos is the position and i is the dimension.

B. Training of Transformer

In the original Transformer paper, the model was trained on the standard WMT 2014 English-German dataset using the Adam optimizer with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. The learning rate was assigned based on a formula, where the number of $warmup\_steps$ was set to 4000.

$$lrate = \frac{min(step\_num^{0.5}, step\_num \cdot warmup\_steps^{-1.5})}{\sqrt{d_{model}}}$$

In addition, to prevent overfitting during training, several regularization techniques were applied, such as Residual Dropout with $P_{drop} = 0.1$ and label smoothing with a value of $\epsilon_{ps} = 0.1$. The use of these techniques allowed the Transformer to achieve better performance after training.

III. BERT

BERT [2], which stands for Bidirectional Encoder Representations fromTransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial taskspecific architecture modifications.

A. Model Architecture of BERT

The model architecture of BERT is a multi-layer bidirectional Transformer encoder based on the Transformer architecture. Compared to the original Transformer architecture, BERT uses a bidirectional structure, which will be explained in more details later.

In addition, BERT also employs different structures for input and output representations. To adapt to different downstream tasks, the input representation should be able to handle sequences consisting of multiple sentences as single token. Therefore, BERT uses WordPiece embeddings for this purpose.

In BERT, the first token of each sequence is a special classification token ([CLS]). Sentences pairs are packed into a single sequence. There are two methods to differentiate between different sentences within a sequence. Firstly, a special separator token ([SEP]) is used to separate different sentences. Secondly, a learnable embedding is added to each token to indicate which sentence it belongs to. For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings, that's what Fig3 shows.

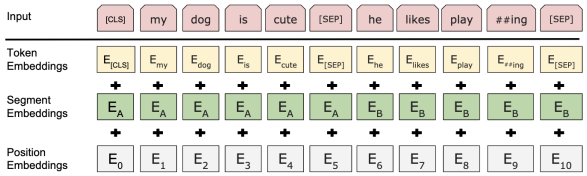

Fig. 3. Input Representation of BERT

B. Training of BERT

Training BERT is a crucial part of its development, and it involves several techniques, including Unsupervised Feature-based Approaches, Unsupervised Fine-tuning Approaches, and Transfer Learning from Supervised Data. The training process is divided into two parts.

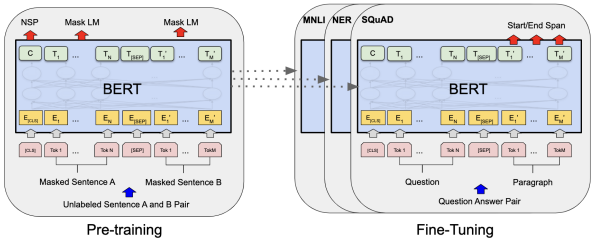

Fig. 4. Training of BERT

The first part is Pre-Training, during which BERT is trained to perform two tasks.Task 1 is Masked Language Modeling (MLM), where 15% of the wordpiece tokens are randomly masked and replaced with a special token ([MASK]). BERT is then trained to predict these masked tokens.Task 2 is Next Sentence Prediction (NSP), where BERT is given a pair of sentences A and B and is trained to predict whether sentence B follows sentence A.

The second part is fine-tuning, where we use a small amount of labeled data to train BERT for specific downstream tasks. Because of the properties of the Transformer architecture, BERT can model downstream tasks by adjusting its inputs and outputs. During fine-tuning, BERT uses bidirectional self-attention mechanisms to encode text and to coordinate and unify its inputs and outputs for the specific task at hand.

IV. GPT

GPT [3] (Generative Pre-trained Transformer) is a Transformer-based pre-trained language model developed by OpenAI. The development of GPT can be divided into four versions. GPT-1 was pre-trained on a large corpus of text and was one of the first models to demonstrate excellent performance in natural language processing tasks. GPT-2 is an improved version of GPT-1, pre-trained on larger datasets with more parameters, and can generate more natural and fluent text. GPT-3 is a much larger pre-trained language model with 175 billion parameters, achieving state-of-the-art results on various natural language processing tasks. As of today, GPT-4 is currently under development, expected to have even larger model parameters and better support for multimodal input to handle more complex tasks.

A. GPT-1

GPT-1 [3] implemented a semi-supervised approach that combines unsupervised pre-training and supervised fine-tuning to accomplish language understanding tasks. This approach enables a large amount of training to form a global transformation, and only minor adjustments are needed to adapt to a wide range of downstream tasks.

1) Model Architecture of GPT-1: GPT-1 uses a Transformer Decoder [26] as its primary architecture, which has been proven to have strong performance on various tasks. However, depending on the specific task, we can replace or fine-tune different modules of the model, such as its input and output, to better adapt to the task at hand.

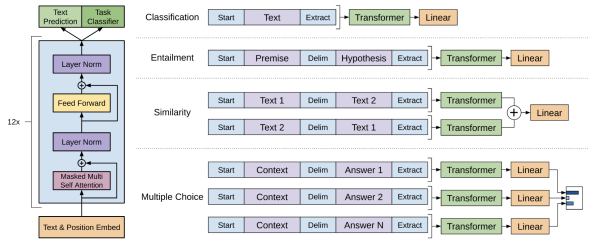

Fig. 5. Model Architecture of GPT-1 & The Task-specific for different tasks

As GPT use a Transformer Decoder, this model applies multi-headed self-attention operation over the input context tokens, followed by position-wise feedforward layers, to produce an output distribution over target tokens.

$$
\begin{aligned}
h_0 &= UW_e + W_p \\
h_l &= transformer\_block(h_{l-1}) \ \forall i \in [1, n] \\
P(u) &= softmax(h_n W_e^T)
\end{aligned}
$$

where $U = (u_{-k}, ..., u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

2) Training of GPT-1: The training of GPT-1 is divided into two parts.

The first part being unsupervised pre-training.In this process, GPT is given a set of unlabeled corpus tokens $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ and utilizes a standard language model to maximize the following likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, ..., u_{i-1}; \Theta)$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].

The second part is supervised fine-tuning, which involves adjusting the model parameters for a specific task after unsupervised pre-training. Assuming there is a labeled dataset $\mathcal{C}$, where each input has an associated label. The input is processed by the model to obtain an activation value h_l^m, which, when multiplied by W_y, can predict the label y. This allows us to maximize the following objective $L_2(\mathcal{C})$:

$$
\begin{aligned}
P(y|x^1, x^2, ..., x^m) &= softmax(h_l^m W_y) \\
L_2(\mathcal{C}) &= \sum_i \log P(y|x^1, x^2, ..., x^m)
\end{aligned}
$$

Adding a language model as an auxiliary objective during supervised fine-tuning can improve learning by (a) enhancing the generalization ability of the supervised model, and (b) accelerating convergence. Therefore, we perform a maximization operation on the following objective $L_3(\mathcal{C})$:

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda L_1(\mathcal{C})$$

where λ is weight.

B. GPT-2

GPT-2 [4] largely follows the architecture of its predecessor GPT-1, with only minor modifications. Layer normalization is moved to after the input of each sub-block, and additional layer normalization is added after the final self-attention module. A modified initialization is used to account for accumulation on the residual path with model depth, where the weights of residual layers are scaled by a factor of $1/\sqrt{N}$ at initialization, with N being the number of residual layers. The vocabulary is expanded to 50,257, and the context size is increased from 512 to 1024 tokens. A larger batch size of 512 is also used.

GPT-2 was trained on the WebText dataset, and in the original paper, the authors evaluated GPT-2's performance on various tasks, including Language Modeling, Children's book test, LAMBADA, Winograd Schema Challenge, Reading Comprehension, Translation, and Question Answering. As the number of parameters increased, GPT-2 exhibited remarkable abilities.

C. GPT3

The model size and amount of data used for training in GPT-3 [4] have significantly increased. GPT-3 has 96 layers, each with 96 attention heads, for a total of 175 billion parameters, which is 116 times more than GPT-2. GPT-3 was trained on a 45 terabytes text dataset selected from Common Crawl, containing 500 billion words, which is 10 times larger than the dataset used for GPT-2.

GPT-3's main selling point is few-shot learning, which means it can complete various downstream tasks with only a few examples as input, without fine-tuning or gradient updates. GPT-3 has demonstrated outstanding performance on several natural language processing tasks, even surpassing some specially designed models.

D. GPT4

GPT-4 [5]'s main selling point is multimodal learning, which means it can accept both images and text as input, and output text. GPT-4 can generate descriptive or creative text based on the content of the images, and it can also generate relevant images based on the content of the text.

V. Security and Robustness of LLM

As the use of GPT (Generative Pre-trained Transformer) models becomes more widespread, people are beginning to realize the importance of addressing their robustness and security issues.

Robustness issues mainly refer to the ability of GPT models to perform in the face of noise, attacks, and other challenges. For example, how will the model handle input text that contains noise, typos, or interfering phrases? How will it perform in the face of targeted attacks, such as targeted adversarial sample attacks? These are all robustness issues that need to be considered.

Security issues mainly relate to the potential harm that GPT models may cause during their application. For instance, if GPT models are used to generate misleading statements, malicious ads, fake news, etc., they could have a negative impact on society. At the same time, if unauthorized individuals gain access to GPT models, they may use them for malicious attacks, phishing, and other illegal activities.

Therefore, researching the robustness and security of GPT models is essential to ensure their reliability and safety. This research includes but is not limited to adversarial sample training, robustness evaluation, model interpretability analysis, privacy protection, and other areas. Only by continuously researching the robustness and security issues of GPT models can we better address future challenges and risks, safeguard the reliability and security of GPT models in various fields, and promote their more extensive use.

Shortly after the launch of ChatGPT, many papers analyzing the security and robustness of GPT models emerged. For example, paper [27] tested the performance of GPT models and previous LLM models from the perspectives of adversarial attacks and OOD, to see if there was significant improvement. Through the testing in paper, we found that GPT models showed relatively good performance on many security testing datasets. However, there are still issues such as vulnerability to attacks and inadequate guarantees of the authenticity of generated content.

Next, we will start with the lifecycle of an LLM (Large Language Model) and attempt to analyze the vulnerabilities that may exist in LLMs and the methods for discovering these vulnerabilities.

A. Vulnerabilities

At the beginning of this section, we first need to discuss the inherent issues that exist in LLMs. These issues cannot be fixed by the LLM itself, but can be addressed through expanding the training dataset and adjusting the model structure.

Performance Issues: Unlike traditional software systems, the same input in an LLM does not always result in the same output. Performance issues in LLMs can be divided into two parts: factual errors and reasoning errors.

Factual errors refer to when the output of an LLM contradicts the truth. When we ask an LLM a simple common sense question, the answer we receive may be far from the truth. [28] noted poor performance of ChatGPT in the legal and scientific fields due to factual errors.

Reasoning errors are a particularly interesting problem, as they indicate that LLMs are more prone to errors when performing logical reasoning tasks. This is because LLMs do not reason themselves, but generate a probabilistic answer based on the way they were trained. When a question is similar to what the LLM was trained on, it will have a higher accuracy [29].

Backdoor Attack: Backdoor attacks aim to inject harmful knowledge to cause errors in LLMs. Common backdoor attacks include poisoning the training data during the training phase or modifying the parameters of the model. Backdoor attacks only trigger when the input prompt to the model contains a corresponding trigger, which causes the LLM to exhibit the erroneous behavior that the attacker expects.

Let's first consider how to design a backdoor trigger. [30] introduced three different types of backdoor triggers that operate at the character, word, and sentence levels. [31] introduces two novel backdoor triggers, one that uses homophones to attack and another that applies dynamic sentence attacks.

Next, let's consider how to implant a pre-designed trigger into a model. [32] introduced a backdoor attack on pre-trained NLP models that does not require any task-specific labels. [33] introduces a method called RIPPLe for implementing backdoor attacks on fine-tuning datasets, while also implementing a technique called Embedding Surgery to enhance the survival rate of the backdoor.

In addition, there are many more complex forms of backdoor attacks. For example, [34] implemented backdoor attacks in more complex downstream NLP tasks, such as NMT and QA, by replacing the designed questions, resulting in harmful responses to users. [31] introduced backdoor attacks on text-based image generation tasks for the first time, using a teach-

student method to implant the backdoor into a pre-trained text encoder.

Poisoning and Disinformation: Poisoning attacks are quite effective in attacking deep neural networks because most models are fine-tuned on many public datasets that often contain many untrusted documents and websites, making it easy for attackers to inject harmful data. [35] demonstrates how the poisoning attack can render a spam filter useless. Even if only 1 percent of the data is maliciously modified, the spam filter can easily fail. [36] introduces two poisoning attacks, showing us how to modify a small amount of data to achieve good attack results, and experiments were conducted on various popular datasets.

Prompt Injection: The Prompt Engineering Guide on github introduced several methods of using prompt injection to cause LLM misalignment. In these attacks, the attacker uses the prompt to make the LLM generate offensive and inappropriate content. [37] provides two ways to attack using prompts, one called goal hijacking, which aims to change the potential target of the original prompt into a targeted one, and the other called prompt leaking, which endeavors to retrieve information from private prompts.

[38] explores the programmatic behavior of LLMs, demonstrating that classical security attack methods can also be used to attack the defense systems of LLMs. Instruction-based LLMs can generate convincing malicious content under malicious prompts. [39] proposes an attack method of assigning a persona to LLMs. For example, prompting an LLM to s̈peak like Muhammad Aličan significantly increase the toxicity of its generated content. [40] develops a black-box framework for non-structured image and text generation, applying black-box methods to explore adversarial prompts.

[41] claims that in previous versions of ChatGPT, personal private information could be directly extracted through prompts. Furthermore, they have developed a method that can change ChatGPT's restrictions, allowing it to answer user queries unethically. [42] has proposed a novel indirect prompt injection method, indicating that prompt injection risks may occur not only among attackers but also among users, developers, and automated data processing systems.

Else: In fact, there are many vulnerabilities in LLMs that we have not yet encountered and are difficult to control. For example, there have been issues with the leakage of some users' information during the operation of ChatGPT, which highlights the importance of maintaining LLM application systems. In addition, we have found that LLMs may exhibit some obvious biases during operation, leading to answers with a certain subjective tendency. In fact, the ChatGPT model categorizes itself as an ENFJ personality type in the Myers-Briggs Type Indicator test. These potential issues may cause harm to LLMs in the future.

B. Verification

This section discusses if and how more rigorous verification can be extended to work on LLM-based machine-learning tasks. So far, the verification or certification of LLMs is still an emerging research area. This section will first provide a comprehensive and systematic review of the verification techniques on various NLP models. Then we discuss a few pioneering black-box verification methods that can be workable on large-scale language models. These are followed by a discussion on how to extend these efforts towards LLMs and a review of the efforts to reduce the scale of LLMs to increase the validity of verification techniques

Verification on Natural Language Processing Models: As discussed in previous sections, an attacker could generate millions of adversarial examples by manipulating every word in a sentence. However, such methods may still fail to address numerous unseen cases arising from exponential combinations of differ- ent words in a text input. To overcome these limitations, another class of techniques has emerged, grounded in the concept of "certification" or "verification" [43], [44]. For example, via certification or verification, these methods train the model to provide an upper bound on the worst-case loss of perturbations, thereby offering a certificate of robustness without necessitating the exploration of the adversarial space [45]. By util- ising these certification-driven methods, we can better evaluate the model's robustness in the face of adversarial attacks [46].

Black-box Verification: Most existing verification techniques have specific requirements for deep neural networks, but with the increasing complexity and scale of LLMs, black-box methods have become an inevitable trend for verification [47]–[49]. In the black-box setting, adversaries can only query the target classifier without knowing the underlying model or the feature representations of inputs. Several studies have explored more efficient methods for black-box settings, although most of the current approaches focus on vision models [47]–[49]. Subsequently, an anytime algorithm was developed to approximate global robustness by iteratively computing lower and upper bounds [50]. Recently, some researchers have attempted to develop black-box verification methods for NLP models, although these methods are not scalable to LLMs. For example, one study introduced a framework for evaluating the robustness of NLP models against word substitutions [51].

Robustness Evaluation on LLMs: With the widespread adoption of LLMs, people have begun to analyze the robustness of these models. Cheng et al. [52] proposed a seq2seq algorithm based on a projected gradient method, and introduced innovative loss functions for targeted keyword attacks and non-overlapping adversarial examples. Weng et al. [53], [54] presented a self-verification method that leverages the conclusion of CoT as a condition for constructing a new sample, and then tasks the LLM with re-predicting the original conditions that have been masked. This approach allows for the calculation of an explainable verification score based on accuracy. However, until now, there has been limited research on verifying LLMs.

Towards Smaller Models: Due to the large number of parameters in current LLMs, which can reach billions, verification becomes difficult, and thus, we may need to use smaller LLMs. One approach is model compression, such

as quantization [55]–[57], but directly applying quantization techniques on LLMs can lead to performance degradation. Therefore, we still need other techniques to achieve this goal [58]–[60]. In addition to quantization techniques, low-rank adaptation [61] has been shown to significantly reduce the number of parameters while maintaining model performance.

VI. Conclusion

In this survey, we explored the basic structure and training principles of the Transformer model, with a focus on representative models such as BERT and GPT. We investigated the security and robustness issues of LLMs, and identified their major vulnerabilities. Furthermore, we examined various methods to verify LLMs.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. I, II

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. I, III

[3] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018. I, IV, IV-A

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019. I, IV-B, IV-C

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020. I, IV-D

[6] OpenAI, "Gpt-4 technical report," *arXiv*, 2023. I

[7] K. Knill and S. Young, "Hidden markov models in speech and language processing," *Corpus-based methods in language and speech processing*, pp. 27–68, 1997. II

[8] D. A. Reynolds *et al.*, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009. II

[9] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Advances in neural information processing systems*, vol. 13, 2000. II

[10] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, no. 3. Makuhari, 2010, pp. 1045–1048. II

[11] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012. II

[12] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE, 2017, pp. 1597–1600. II

[13] U. Khandelwal, H. He, P. Qi, and D. Jurafsky, "Sharp nearby, fuzzy far away: How neural language models use context," *arXiv preprint arXiv:1805.04623*, 2018. II

[14] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. IEEE, 1999, pp. 1033–1038. II

[15] P. S. Heckbert, "Survey of texture mapping," *IEEE computer graphics and applications*, vol. 6, no. 11, pp. 56–67, 1986. II

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020. II

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013. II

[18] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019. II

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020. II

[20] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022. II

[21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763. II

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014. II-A

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. II-A1

[24] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016. II-A1

[25] D. Britz, A. Goldie, M.-T. Luong, and Q. Le, "Massive exploration of neural machine translation architectures," *arXiv preprint arXiv:1703.03906*, 2017. II-A2

[26] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," *arXiv preprint arXiv:1801.10198*, 2018. IV-A1

[27] J. Wang, X. Hu, W. Hou, H. Chen, R. Zheng, Y. Wang, L. Yang, H. Huang, W. Ye, X. Geng *et al.*, "On the robustness of chatgpt: An adversarial and out-of-distribution perspective," *arXiv preprint arXiv:2302.12095*, 2023. V

[28] X. Shen, Z. Chen, M. Backes, and Y. Zhang, "In chatgpt we trust? measuring and characterizing the reliability of chatgpt," *arXiv preprint arXiv:2304.08979*, 2023. V-A

[29] H. Liu, R. Ning, Z. Teng, J. Liu, Q. Zhou, and Y. Zhang, "Evaluating the logical reasoning ability of chatgpt and gpt-4," *arXiv preprint arXiv:2304.03439*, 2023. V-A

[30] X. Chen, A. Salem, D. Chen, M. Backes, S. Ma, Q. Shen, Z. Wu, and Y. Zhang, "Badnl: Backdoor attacks against nlp models with semantic-preserving improvements," in *Annual Computer Security Applications Conference*, 2021, pp. 554–569. V-A

[31] L. Struppek, D. Hintersdorf, and K. Kersting, "Rickrolling the artist: Injecting invisible backdoors into text-guided image generation models," *arXiv preprint arXiv:2211.02408*, 2022. V-A, V-A

[32] L. Shen, S. Ji, X. Zhang, J. Li, J. Chen, J. Shi, C. Fang, J. Yin, and T. Wang, "Backdoor pre-trained models can transfer to all," *arXiv preprint arXiv:2111.00197*, 2021. V-A

[33] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pre-trained models," *arXiv preprint arXiv:2004.06660*, 2020. V-A

[34] S. Li, H. Liu, T. Dong, B. Z. H. Zhao, M. Xue, H. Zhu, and J. Lu, "Hidden backdoors in human-centric language models," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 3123–3140. V-A

[35] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia, "Exploiting machine learning to subvert your spam filter." *LEET*, vol. 8, no. 1-9, pp. 16–17, 2008. V-A

[36] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning web-scale training datasets is practical," *arXiv preprint arXiv:2302.10149*, 2023. V-A

[37] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," *arXiv preprint arXiv:2211.09527*, 2022. V-A

[38] D. Kang, X. Li, I. Stoica, C. Guestrin, M. Zaharia, and T. Hashimoto, "Exploiting programmatic behavior of llms: Dual-use through standard security attacks," *arXiv preprint arXiv:2302.05733*, 2023. V-A

[39] A. Deshpande, V. Murahari, T. Rajpurohit, A. Kalyan, and K. Narasimhan, "Toxicity in chatgpt: Analyzing persona-assigned language models," *arXiv preprint arXiv:2304.05335*, 2023. V-A

[40] N. Maus, P. Chao, E. Wong, and J. Gardner, "Adversarial prompting for black box foundation models," *arXiv preprint arXiv:2302.04237*, 2023. V-A

[41] H. Li, D. Guo, W. Fan, M. Xu, and Y. Song, "Multi-step jailbreaking privacy attacks on chatgpt," *arXiv preprint arXiv:2304.05197*, 2023. V-A

[42] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "More than you've asked for: A comprehensive analysis of novel prompt injection threats to application-integrated large language models," *arXiv preprint arXiv:2302.12173*, 2023. V-A

[43] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Towards verified artificial intelligence," *arXiv preprint arXiv:1606.08514*, 2016. V-B

[44] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*. Springer, 2017, pp. 3–29. V-B

[45] A. Sinha, H. Namkoong, R. Volpi, and J. Duchi, "Certifying some distributional robustness with principled adversarial training," *arXiv preprint arXiv:1710.10571*, 2017. V-B

[46] I. Goodfellow and N. Papernot, "The challenge of verification and testing of machine learning," *Cleverhans-blog*, 2017. V-B

[47] M. Wicker, X. Huang, and M. Kwiatkowska, "Feature-guided black-box safety testing of deep neural networks," in *Tools and Algorithms for the Construction and Analysis of Systems: 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I 24*. Springer, 2018, pp. 408–426. V-B, V-B

[48] M. Wu, M. Wicker, W. Ruan, X. Huang, and M. Kwiatkowska, "A game-based approximate verification of deep neural networks with provable guarantees," *Theoretical Computer Science*, vol. 807, pp. 298–329, 2020. V-B, V-B

[49] P. Xu, W. Ruan, and X. Huang, "Quantifying safety risks of deep neural networks," *Complex & Intelligent Systems*, pp. 1–18, 2022. V-B, V-B

[50] W. Ruan, M. Wu, Y. Sun, X. Huang, D. Kroening, and M. Kwiatkowska, "Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance." International Joint Conferences on Artificial Intelligence Organization, 2019. V-B

[51] E. La Malfa, M. Wu, L. Laurenti, B. Wang, A. Hartshorn, and M. Kwiatkowska, "Assessing robustness of text classification through maximal safe radius computation," *arXiv preprint arXiv:2010.02004*, 2020. V-B

[52] M. Cheng, J. Yi, P.-Y. Chen, H. Zhang, and C.-J. Hsieh, "Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 3601–3608. V-B

[53] Y. Weng, M. Zhu, S. He, K. Liu, and J. Zhao, "Large language models are reasoners with self-verification," *arXiv preprint arXiv:2212.09561*, 2022. V-B

[54] Y. Weng, M. Zhu, F. Xia, B. Li, S. He, K. Liu, and J. Zhao, "Neural comprehension: Language models with compiled neural networks," *arXiv preprint arXiv:2304.01665*, 2023. V-B

[55] E. Frantar and D. Alistarh, "Optimal brain compression: A framework for accurate post-training quantization and pruning," *arXiv preprint arXiv:2208.11580*, 2022. V-B

[56] P. Feng and Z. Tang, "A survey of visual neural networks: current trends, challenges and opportunities," *Multimedia Systems*, vol. 29, no. 2, pp. 693–724, 2023. V-B

[57] M. Nagel, R. A. Amjad, M. Van Baalen, C. Louizos, and T. Blankevoort, "Up or down? adaptive rounding for post-training quantization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7197–7206. V-B

[58] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "Zeroquant: Efficient and affordable post-training quantization for large-scale transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 168–27 183, 2022. V-B

[59] G. Park, B. Park, S. J. Kwon, B. Kim, Y. Lee, and D. Lee, "nuqmm: Quantized matmul for efficient inference of large-scale generative language models," *arXiv preprint arXiv:2206.09557*, 2022. V-B

[60] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 318–30 332, 2022. V-B

[61] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021. V-B