Received January 3, 2022, accepted January 19, 2022, date of publication January 26, 2022, date of current version February 16, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3146405

Adversarial Machine Learning in Text Processing: A Literature Survey

IZZAT ALSMADI[1], NURA ALJAAFARI[2], MAHMOUD NAZZAL[3], (Member, IEEE), SHADAN ALHAMED[2], AHMAD H. SAWALMEH[4,5], (Member, IEEE), CONRADO P. VIZCARRA[2], ABDALLAH KHREISHAH[3], (Senior Member, IEEE), MUHAMMAD ANAN[6], (Senior Member, IEEE), ABDULELAH ALGOSAIBI[2], (Member, IEEE), MOHAMMED ABDULAZIZ AL-NAEEM[7], (Member, IEEE), ADEL ALDALBAHI[2], (Member, IEEE), AND ABDULAZIZ AL-HUMAM[2]

[1]Department of Computing and Cyber Security, Texas A&M University, College Station, TX 77843, USA
[2]Department of Computer Science, King Faisal University, Hofuf 31982, Saudi Arabia
[3]Department of Electrical and Computer Engineering, Newark College of Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA
[4]Computer Science Department, Northern Border University, Arar 73222, Saudi Arabia
[5]Remote Sensing Unit, Northern Border University, Arar 91431, Saudi Arabia
[6]Software Engineering Department, College of Engineering, Alfaisal University, Riyadh 11533, Saudi Arabia
[7]Department of Computer Networks and Communications, College of Computer Sciences and Information Technology, King Faisal University, Hofuf 31982, Saudi Arabia

Corresponding author: Izzat Alsmadi (ialsmadi@tamusa.edu)

This work was supported by the Deputyship for Research and Innovation, Ministry of Education, Saudi Arabia, under Project 1120.

ABSTRACT Machine learning algorithms represent the intelligence that controls many information systems and applications around us. As such, they are targeted by attackers to impact their decisions. Text created by machine learning algorithms has many types of applications, some of which can be considered malicious especially if there is an intention to present machine-generated text as human-generated. In this paper, we surveyed major subjects in adversarial machine learning for text processing applications. Unlike adversarial machine learning in images, text problems and applications are heterogeneous. Thus, each problem can have its own challenges. We focused on some of the evolving research areas such as: malicious versus genuine text generation metrics, defense against adversarial attacks, and text generation models and algorithms. Our study showed that as applications of text generation will continue to grow in the near future, the type and nature of attacks on those applications and their machine learning algorithms will continue to grow as well. Literature survey indicated an increasing trend in using pre-trained models in machine learning. Word/sentence embedding models and transformers are examples of those pre-trained models. Adversarial models may utilize same or similar pre-trained models as well. In another trend related to text generation models, literature showed effort to develop universal text perturbations to be used in both black- and white-box attack settings. Literature showed also using conditional GANs to create latent representation for writing types. This usage will allow for a seamless lexical and grammatical transition between various writing styles. In text generation metrics, research trends showed developing successful automated or semi-automated assessment metrics that may include human judgement. Literature showed also research trends of designing and developing new memory models that increase performance and memory utilization efficiency without validating real-time constraints. Many research efforts evaluate different defense model approaches and algorithms. Researchers evaluated different types of targeted attacks, and methods to distinguish human versus machine generated text.

INDEX TERMS Adversarial machine learning, generative adversarial networks, GAN, text generation.

I. INTRODUCTION

There are strong indicators that machine learning (ML) models are amenable to attacks that involve modification of input data intending to cause models' target misclassification. The main reason behind such vulnerability is the adaptive nature of ML models. While their vulnerability to attacks can vary, nonetheless, all ML models are generally susceptible to such attacks. To demonstrate an attack, we can think of an ML model or system that takes and processes some inputs, which

The associate editor coordinating the review of this manuscript and approving it for publication was Kashif Saleem.

results in an initial accurate classification based on original data inputs. Still, it is eventually possible for some malicious actor to artificially construct an input that will result in incorrectly classified instances when tested by the machine learner. Generating such artificial input data or adversarial examples can be done in a variety of ways, including:

- Fast-gradient-sign methods
- One-step target-class methods
- Basic iterative methods
- Iterative least-likely-class methods

Each method has its advantages and drawbacks when used in an actual attack. For example, a fast gradient sign method is a simple and easy way to create adversarial examples. However, it has a lower success rate compared to other methods. This method's success rate is much lower than the others.

After generating adversarial examples, they can be used in two main forms of attack settings: black-box and white-box attacks. Black box attacks occur in cases where the malicious actor has no, or limited information about the specifics of the model and how it works. Conversely, the attacker is assumed to possess full or most of the required knowledge about the ML model and its parameters in a white-box setting.

Adversarial examples are inputs for ML models that allow an attacker to cause the target ML model to make classification/prediction errors. These examples are intentionally designed to make the model output incorrect. In a world ruled by ML systems at a large scale, one exploitative example or sample can have a cascading effect on the model and whatever entity or application using them.

Transferability is a specially important feature of adversarial examples. In this context, most adversarial examples can be transferred from one model to another. Consequently, an adversarial example generated and tested by a certain malicious actor can attack other models, either with the same attacker or by others.

To guard against adversarial input attacks, adversarial training is performed. This concept refers to the process of adding adversarial examples into the data set at the time of training the model. By injecting these adversarial examples into the training set, one trains the model to be resilient against possible attacks since it will know how to handle this kind of input when it is artificially introduced.

Training an ML model with ensemble models (e.g. from different datasets, dataset subsets, features, or model settings) can also be an effective approach against adversarial attacks. There have been several research works that showed the effectiveness of ensemble training in improving the model's resilience against both black-box and white-box attacks.

The idea of transferred attacks mentioned earlier seems particularly difficult to handle or defend against. Still, ensemble training is shown to be an efficient means of increasing the system's resilience against transferred attacks. Nonetheless, there is a continuous arms race between ML defenders and attackers; new attack types have been suggested to increase the success rate of transferred attacks in a black box scenario.

Evaluating the security of ML models can involve the following processes:

- Identifying attack classes to the system
- Measuring the robustness of the system against these attack classes
- Designing and studying defense mechanisms against potential attacks

ML defense strategies can be divided into reactive versus proactive. A huge problem with securing machine learning systems is that adversarial examples are very difficult to detect. In particular, neural networks are susceptible to such adversarial examples [1]. Based on the experiments performed, they can trick ML classifiers. Moreover, it is shown that there is no easy or concrete method to differentiate between an adversarial example and a legitimate one. Accordingly, it can be concluded that the current security measures we possess are not adequate to defend against current and future types of adversarial examples. Before we can think about using ML in security applications and be confident with their performance, we need to design effective defenses against adversarial attacks.

A secure machine learning algorithm is the one that can perform under adversarial conditions. These conditions deal with the possibility that an adversary can design training data to change the system to treat hostile data as legitimate [2]. In [2], the authors exploit these conditions to develop a framework for security analysis. In security analysis, there are two metrics, security goals, and threat models. Security goals deal with two factors, (1) integrity, how well a system can prevent attackers from reaching system assets, and (2) availability, how well a system can prevent attackers from interfering with regular operations. Threat models evaluate two factors: attacker goals/incentives and capabilities. The taxonomy of these factors against learning systems is categorized into three additional categories:

- Influence (causative and exploratory): which describe the capacity of the attacker, and
- Security violation (integrity and availability): the type of breach the attacker causes, and
- Specificity (target and indiscriminate): the attacker's intention.

As text input is discrete, text generator models the problem as a sequential decision-making process. In such a model, the current state is the previously generated characters, words, or sentences. Correspondingly, the action or prediction to make is the next character/word/sentence to be generated. The generative net is a stochastic policy that maps the current state to a distribution over the action space. As one type of generative models based on deep neural networks, a generative adversarial network (GAN) can create new data instances that resemble original training data. Original GAN described by [3] has the following components and work flow:

- Two NNs: a discriminator and a generator. The discriminator's role is as of a simple classifier, that should distinguish real instances (positive examples, from the original training dataset) from fake instances (negative examples) created by the generator.

- The generator tries to fool the discriminator by synthesizing fake instances that resemble real ones. As training progresses, the generator gets closer to producing instances that can fool the discriminator. If the generator is trained very well, the discriminator gets worse at telling the difference between real and fake instances. Generally speaking, the generator module task is harder than that of the discriminator. For the least, the discriminator's job is binary, whereas the generator's job is much more complex.
- The two NNs compete with two different goals. The goal of the discriminator is to distinguish between the real and the fake instances. The goal of the generator is to learn more about the real instances or data and eventually fool the discriminator.
- Both are trained separately. Each one assumes that the other module is fixed at the time of cycle training (to avoid dealing with a moving target that can be more complex). An accuracy of 100% for the generator to generate fake instances indicates an accuracy of 50% for the discriminator.
- As another sign of competition/game between rivals, the generator instances become negative training examples for the discriminator. The discriminator punishes the generator for producing incorrect instances and rewards it for producing correct instances. In return, the generator evolves to make the discriminator punish less and reward more.
- In effect, a good discriminator should not reveal enough information for the generator to make progress.
- In addition to discriminator and generator modules, GANs include the following components:
  - Generator's random input module
  - A generator network that transforms the random input into a data instance
  - Generator loss that punishes the generator for failing to fool the discriminator
  - A back-propagation module that adjusts weights by calculating their impact on the output

In this paper, we evaluate research progresses and trends in adversarial machine learning (AML) text progressing focusing on several subjects including proposed defense mechanisms, text generation models, and metrics. In each section, we identify research trends and challenges that can be used to guide researchers in the respective area. Table 1 lists acronyms used in the text.

We can summarize our contribution in this paper as follows:

- Our main goal of this paper is to provide researchers in one single paper open challenges and research trends in the field of adversarial machine learning for text applications.

TABLE 1. Acronyms and abbreviations.

Acronym	Expansion
ML	Machine learning
GAN	Generative adversarial network
NN	Neural network
AML	Adversarial machine learning
NLG	Natural language generation
RNN	Recurrent neural networks
MNIST	Modified National Institute of Standards and Technology
BLEU	Bilingual evaluation understudy
MLE	Maximum likelihood estimation
MLP	Multilayer perceptron
HMM	Hidden Markov model
GMM	Generalized method of moments
VAE	Variational auto-encoder
CNN	Convolutional neural networks
NLL	Negative log-likelihood
RTN	Recursive transition network
RN	Relational memory
LM	Language model
SGD	Stochastic gradient descent
SAP	Stochastic activation pruning
CLP	Clean logit pairing
LQ	Logit squeezing
AI	Artificial intelligence
KNN	k nearest neighbor
DNN	Deep neural network
NMT	Neural machine translation

- While AML field in image and videos is heavily investigated in literature, AML in text applications is much recent and the volume and depth of contributions are much less in comparison with the previous one.
- Unlike other relevant survey papers in the literature (), we followed a top-down ontological approach to identify major themes, subjects, trends and challenges. This paper typically helps novice researchers in new fields to match their interests and skills with open research areas and challenges in their selected field of study.
- AML text applications are continuously growing. They are also getting more serious. One major example that we evaluated in another pending paper is related to social networks. AML in social networks can be seen in the large number of social bots and trolls that are created at large scales and state-sponsored agencies to impact and influence public opinions.

II. TEXT GENERATION MODELS

Natural language generation (NLG) techniques allow natural language text generation based on a given context. NLG for text can be conducted based on predefined grammar such as the Dada Engine [4], or leverage deep learning neural networks such as recurrent neural networks (RNN) [5]. In this section, we describe some of the popular approaches found in relevant literature in the scope of AML.

In [4], the authors focus on email-based attacks. They show how an attacker uses the NLG approach to masquerade attacks. Then, they evaluate their efficacy with an analysis of within-subjects. Specifically, this work suggests that NLG approaches are useful for experienced email users and users of all genders and ages. Therefore, defenders can use NLG to enhance their system filters.

In [5], a new class of attacks based on deep learning language models such as RNNs is defined. The authors use

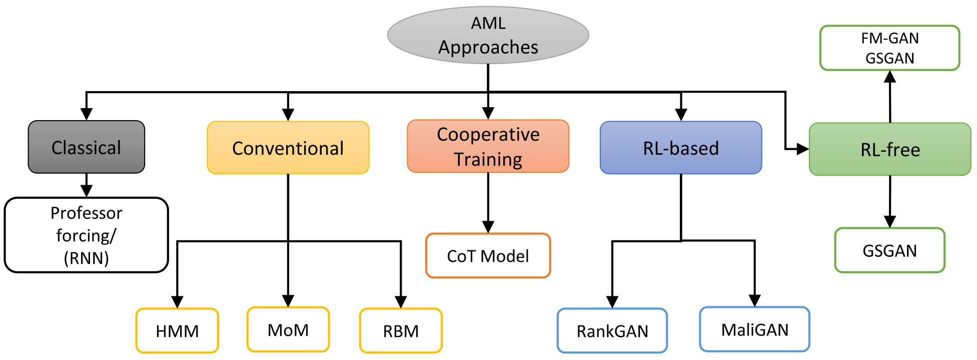


FIGURE 1. Popular approaches for text generation in the scope of AML.

TABLE 2. A comparison of different text generation methods.

Source	Method	Advantage	Limitation
[6]	Texar	Open-source for various text generation	The model does not support various natural language and machine learning applications.
[7]	RELGAN	Relational memory for text generation	Requires model pre-training
[8]	Texygen	A platform to evaluate different text generation model under different metrics	New models update are required and new metrics for improved bench-marking.
[9]	Deep neural network model	Addressing long Chinese review challenge	The model's content for generated texts is not robust compared to human written reviews.
[10]	SeqGAN	SolveS the problem of generating discrete sequence	Implementation of algorithm for better action decision in large scale data.
[11]	LeakGAN	Effective for both long and short text generation scenarios	Discriminator's capacity requires more enhancement
[12]	RankGAN	Can generate high language quality descriptions + It can learn by reference group, no need for training the discriminator	No image synthesis is covered by this model.

RNN to generate an automatic fake review for services and products. These attacks are inexpensive. Therefore, they have the potential to be more scalable. Moreover, they can regulate the produced content rate to avoid the signature trace that makes crowd-sourced campaigns easy to detect. Furthermore, [5] proposes a review platform to demonstrate how a two-phased review generation and customization attack may generate indistinguishable reviews by state of the art detectors. Specifically, a novel approach is developed to defend against RNN-based fake reviews using an RNN-based model's primary vulnerability related to information loss incurred during the training process when fitting a large training dataset.

Figure 1 presents a suggested classification of the popular approaches for text generation in the scope of AML. Also, Table 2 compares different methods for text generation.

A. CLASSICAL APPROACHES

Teacher forcing is a common approach to training RNNs to maximize each token's likelihood from the target sequences given previous tokens in the same sequence [13]. In each time step of training, the model is evaluated based on the target's likelihood, given a ground-truth sequence. Teacher forcing is used for training the generator, which means that the decoder is exposed to the previous ground-truth token.

RNNs trained by teacher forcing should model a distribution that matches the target, where the joint distribution is modeled adequately of RNN models prediction of future steps. The created error when using the model is propagated over each next step, resulting in low performance. A solution to this is training the model using professor forcing. Along this line, the authors in [14] propose a professor/teacher forcing approach, that aims to align generative behavior as closely as possible with teacher-forced behavior. Specifically, they allow the RNN to generate robustly well exceeding the length of the sequences it detected during training. Then, they show that the new training method assists in modeling better long-term dependencies from RNN. Moreover, the professor/teacher forcing scheme can regularize the RNN. This is shown by improving the likelihood of testing on Penn Treebank at the character level, speech synthesis, and sequential MNIST generation. This method's primary motivation is to allow the discriminator to use single-step predictions and behavior statistics. This will force the generator to behave in the same way as in the case of data constraints and when the outputs are generated for sequences that might be much longer than the sequences for the training set.

In text generation, a model is trained over different inputs belonging to different distributions. Specifically, a model may be fed with its predicted data rather than the ground-truth data at inference time. This creates an accumulation in the error referred to as *exposure bias*. This accumulation is known to lead to generating poor samples [15]. To this end, the main advantage of the professor/teacher forcing approach is that it can help to overcome the exposure bias in generated text sequences. On the other hand, a key drawback of this approach is that it is an adversarial training approach; it does not try to render the output distribution indistinguishable from the observed data distribution. As a consequence, high sample quality cannot be guaranteed.

Other language models based on n-grams[1] [16] and feed-forward neural networks [17] are common choice models for text generation in classical approaches. These models usually suffer from two main drawbacks.

1) They are trained to predict the next word in the sequence using the ground-truth words as an input. The sequence is generated by predicting one word at a time and feeding it back as an input at the next step.
2) The word-level loss function is used for the training of these models. A cross-entropy loss is a common option for optimizing the likelihood of the next correct word. Discrete metrics are used to evaluate models' performance, such as bilingual evaluation understudy (BLEU) [18] which is used in language models based on n-grams. It is difficult to train these models to optimize metrics such as BLEU directly for these reasons:
    - It is not distinguishable, and
    - It requires a combinatorial optimization to evaluate which sub-string can maximize the likelihood of the next candidate word.

    We refer the reader to Section V for a comprehensive revision of popular metrics used in the context of text generation.

B. CONVENTIONAL INFERENCE METHODS/MAXIMUM LIKELIHOOD ESTIMATION (MLE)

MLE is used on real data samples, and the parameters are updated directly according to the data samples. This may lead to an overly smooth generative model. The goal is to select the distribution that maximizes the likelihood of generating the data. For practical sample scenarios, MLE is prone to over-fitting or exposure bias issues on the training side. Furthermore, during the inference or generation stage, the error at each time step will accumulate through the sentence generation process [19]. Specifically, the most likely upcoming word at the training process that gives a proper context will be found. Since the text is self-generated, and the upcoming word is found by assuming a valid sequence context, the errors may accumulate through the sentence generation process.

There are four main approaches to address exposure bias: scheduled sampling, reinforcement learning, re-parametrization, and adversarial training.

[1]In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech.

Conventional inference methods typically employ MLP for text generation. Here is a brief revision of several conventional methods.

1) HIDDEN MARKOV MODEL (HMM)

A hidden Markov model (HMM) is a probability graph model that can depict the transition laws of hidden states, and mine the intentional features of data to model the observable variables. The foundation of an HMM is a Markov chain, which can be represented by a special weighted finite-state automaton. The majority of generative models require the using Markov chains [20], [21]. The observable sequence in HMM is the participle of the given sentence in the part-of-speech (PoS) tag while the hidden state is a different PoS.

2) METHOD OF MOMENTS

The method of moments (MoM) or method of learned moments is an early principle of learning [22]. There are situations in which MoM is more advantageous than MLE. One is when MLE is more computationally challenging than MoM [23]. In the generalized method of moments (GMM), in addition to the data and the distribution class, a set of relevant feature functions is given over the instance space [24], [25].

The authors in [23] propose an approach for training large-scale implicit generative models based on MoM. This approach can be mapped to NN models with millions of parameters and provides a learning moments method like asymptotic variance minimized by moment estimators. Moment calculation encounters difficulties in:

- Identifying the millions of moments necessary to learn the model's parameters, and
- Deciding which properties are useful when defining moments

To tackle the first problem, [23] implements a moment network and describes the moments as the hidden units of the network and the gradient output of the network relative to its parameters. On the other hand, to solve the second issue, the asymptotic theory is used to illustrate desired data for moments. Specifically, the asymptotic variance should be reduced for the parameters of the estimated model. It is found that the implicit generative models trained using the proposed MoM algorithm outperform adversarial learning models in generating samples.

Other research contributions for AML in MoM or moment matching include the works in [26]–[28]. The work in [26], proposes new training methods used in GANs to train models. The proposed techniques, namely; feature matching, mini-batch features, and virtual batch normalization, are inspired by heuristics to realize the non-convergence problem and improve the semi-supervised learning performance for the generation of samples of the models. On the other hand, [27] proposes a Fisher GAN framework for training generative models based on integral probability metrics (IPMs). In this setting, MoM is employed using the critic function by constraining second-order moments that efficiently discriminates between real and fake distributions. Fisher IPM is used to formulate an algorithm resulting in these attractive features: stability, unconstrained capacity, efficient computational cost, and representation power for semi-supervised learning.

Authors in [29] discussed variational methods of moments (VMM) estimators based on kernel methods or deep learning. Examples of those variations include: Optimally Weighted Generalized Method of Moments (OWGMM), Neural Variational Methods (NVMs).

Authors in [30] proposed an automated algorithm for the efficient Bayesian inference of the parameters of a computationally expensive, first-principles simulation.

Another related work is [28] which provides a solution to the problem of finding a way to select moment conditions appropriate to the learner's hypothesis class. They proposed an adversarial GMM algorithm and uses conditional moment restrictions to learn deep neural networks.

3) RESTRICTED BOLTZMANN MACHINE (RBM)

Restricted Boltzmann machine (RBM) is a two-layer neural network that consists of a visible layer and a hidden layer [31]. This is a generative model capable of learning representations from data. Generative models have evolved from RBM-based models, such as Helmholtz machines (HMs) [32] and deep belief nets [31], to variational autoencoders (VAEs) [33] and GANs.

The authors in [31] derive a fast and greedy algorithm using complementary priors to generate a hybrid model that can learn deep, directed belief networks, one layer at a time. In this model, the top two layers generate an undirected associative memory. At the same time, the other remaining layers generate a directed acyclic graph that converts the representations in the associative memory into measurable variables.

C. COOPERATIVE TRAINING METHOD (CTM)

In CTM, a language model is trained online to offer a target distribution for minimizing the divergence between the real data distribution and the generated distribution [15], [34]. CTM is used as an alternative to MLE training in AML. Specifically, it is used to train a secondary model by fitting the average of this model close to the real distribution and the generator distribution average.

As another example work, [15] propose a cooperative training method that uses language models. The resulting adversarial text generator distribution is efficiently shaped, and the mode collapse will be slowed down. Thus, this leads text generation towards a more favorable trade-off in quality and diversity.

1) RL-BASED VERSUS RL-FREE TEXT GENERATION

GAN models were originally developed for learning from a continuous distribution. However, the discrete nature of text input handicapped the use of early GANs. In GANs, a reinforcement learning algorithm is used for policy gradient, to get an unbiased gradient estimator for the generator and obtain the reward from the discriminator [35].

The performance of RL-based text generation largely depends on the design of the reward ([36]). Previous methods directly consider the rewards as the hand-crafted metrics between ground-truth sentences and generated sentences. The generator is updated based on policy-gradient methods. As a result, collecting rewards in the generation process is critical, [37]. In [38], authors proposed an RL-based algorithm dedicated to learning a word-based textgeneration task, which does not rely on a pre-training phase while scaling to large vocabularies.

D. RL-BASED TEXT GENERATION

Reinforcement learning (RL) is a technique used to train an agent to perform certain tasks. Due to its generality, reinforcement learning is studied in many disciplines. More recently, GAN models that use a discriminating module to guide the training of the generative module as a reinforcement learning policy have shown promising results in text generation [11].

One of the main goals of adversarial learning is to reduce/eliminate exposure bias. Usually, the natural language outputs are discrete. Therefore, there is no well-defined gradient to direct a GAN through the training process. Various solutions have been proposed to overcome this challenge, e.g., the SeqGAN model [10] is one of the effective methods used to handle the discrete problem in text GAN. The RL policy gradient method is used in this model for updating system weights.

The main challenge facing GAN text generation using RNN is the decision of the operation selection at the generator's output. Therefore, it is challenging to use the back-propagation for generator training.

Many approaches based on RL can be used to address this issue, such as:

- The reinforce algorithm and policy gradients
- The actor-critic approach
- The Gumbel-softmax approximation
- Operating in the continuous space for the generator's output

A recent work in [12] develops a novel adversarial learning approach based on GAN, namely RankGAN, to generate high-quality language descriptions. RankGAN is used as a relative ranker for ranking sets of human-written and machine-written sentences by an adversarial approach. In this setting, the machine-learned ranking optimization problem is used to train the discriminator. Herein, two NN models are exploited in the proposed AML network, a ranker, and the generator. The ranker is trained to rank machine-written sentences lower than human-written sentences relative to a human-written reference sentence. As a result, the generator is trained to produce sentences that puzzle the ranker, thus the machine-written sentences are ranked higher than human-written sentences relative to the reference.

One of the main challenges that face text generation via GAN is the discrete state space problem. The authors in [39] propose a text generation method via GAN that addresses the aforementioned challenge by allowing the discriminator to deal with continuous-valued distribution outputs. In this setting, the discriminator receives a sequence of probabilities from the generator for each token in the vocabulary, as well as a sequence of one-hot vectors from the real data distribution. Specifically, an empirical investigation for text generation is provided by applying GANs to discrete state spaces.

Another approach that can be used to tackle the discrete data challenge in GAN is proposed by [40]. The authors propose a maximum-likelihood augmented discrete GAN (MaliGAN). Herein, a new GAN training objective is introduced. This objective avoids the stability problem that arises when the discriminator output is used as a direct reinforcement learning reward. Furthermore, a normalized maximum likelihood optimization is developed as a target. In order to meet this objective, significance sampling and variance reduction techniques are used.

There are several models of RL, some of which are applied to sentence generation, e.g., actor-critic algorithm and deep Q-network [41]–[43].

In general, RL methods consist of two types; 1) Value-based methods such as Q-learning and deep Q-networks (DQN) which outperform other models in the continuous and stochastic environment, and 2) Policy-based methods such as: reinforce and policy gradient. More recently, the actor-critic approach is developed by merging the two above-mentioned types in an attempt to maximize the benefits of both value-based and policy-based approaches while avoiding their shortcomings. Also, [43] uses this method by training the NN to generate sequences. This method can be used to overcome the log-likelihood training approach limitations, which are constrained by the disparity between their testing and training modes.

1) THE ACTOR-CRITIC APPROACH ADVANTAGE

The employment of the actor-critic approach provides better machine translation tasks compared with the MLE and reinforce approaches. On the other hand, [42] proposes using DQN for sequence-to-sequence learning, which iteratively decodes the output sequence, employing an encoder-decoder long short-term memory (LSTM) network for automatic internal states approximation and DQN actions formulation. An attention technique is combined with the RL's exploration strategy to allow the decoder LSTM network to learn from the generated texts during this phase. This approach provides better performance than the sentence regeneration task, specifically while decoding unknown sentences. Moreover, this approach tackles the problem of decoding sequences with the variable-length size and the previously not known sentences.

RL-based approaches face several challenges:

- Optimization challenge: they may yield high-variance gradient estimates [44], [45].
- They may get trapped or converge to a sub-optimal local minimum.
- They may have a huge state-action space, leading to large portions of the space still unexplored.
- In general, the generated sentences' quality is poor due to the above-aforementioned reasons.

E. RL-FREE GANs FOR TEXT GENERATION

Researchers try also to find efficient approaches not based on RL for training text GANs to address RL challenges. Examples of RL-alternative models are.

- Latent space-based solutions
- Continuous approximation of discrete sampling

For GANs in the RL-free category, GSGAN, [46] and TextGAN, [45] use the Gumbelsoftmax and soft-argmax trick, respectively, to deal with discrete data, [47].

These models apply a simple soft-argmax operator, or Gumbel-softmax trick to provide a continuous approximation of the discrete distribution on text. Examples of research efforts in this category include: TextGAN [45], Gumbel-Softmax GAN (GSGAN) [44], [46], [48], FM-GAN [35], GSGAN [46], and RelGAN [7].

Sequence generation for discrete elements is one of the main limitations of GANs. This challenge can be tackled using different distributions rather than multinomial discrete distribution such as Gumbel-softmax distribution. Gumbel-softmax is a continuous distribution used for sample approximation of the multinomial discrete distribution. The work in [46], uses this method based on RNN to generate sequences. Moreover, [45] presents another approach to overcome the generation of the text for discrete elements and alleviate the mode collapse problem associated with GAN training. This is achieved through a framework to generate text (TextGAN) through GAN, where the LSTM network is employed as a generator and a convolutional neural networks (CNN) is a discriminator. In this method, the kernel-based discrepancy metric is used for matching the distributions of the high-dimensional latent feature between the real and synthetic sequences. On the other hand, [35] proposes a new approach, namely feature mover's distance (FMD), to generate text in GAN, inspired by optimal transport theory. Specifically, FMD is used as a discrepancy metric for matching the high-dimensional latent feature distributions between the real and synthetic sequences.

1) ADVANTAGES OF THE FMD APPROACH

This approach leads to a discriminatory critic and an easy-to-optimize objective that can overcome the mode collapse and the brittle training problems in current methods.

The study in [7] proposes a new GAN approach to generate text called RelGAN. This approach contains the following main components.

- A relational-based memory generator for longer-range dependency in text modeling,
- Gumbel-softmax relaxation to train GANs on discrete data, and
- Multiple embedded representations in the discriminator give more informative signals for the generator updates.

RelGAN has the following Advantages:

1) RelGAN outperforms the current models in terms of sample quality and diversity.
2) RelGAN can control the trade-off between sample quality and diversity via a single adjustable parameter.
3) RelGAN is the first approach that allows GANs to generate realistic text with Gumbel-Softmax relaxation.

Table 3 presents a summary of the popular approaches for GANs in the scope of text generation.

F. LONG VERSUS SHORT TEXT GENERATION

The literature body in this area differentiates between the generation of short texts (e.g. less than 20 words) and that of long text. Application areas of each one can be different accordingly. The majority of publications focused on short text generation as it seems to be less challenging. Different challenges are discussed in the literature especially in long text generation. One challenge is the sparse reward issue, in which a scalar guiding signal is only available after an entire sequence has been generated. Furthermore, the non-informative scalar guiding signal of the generated text is another challenge, where the intermediate text structure information is not available during the text generation process [11], [41], [51].

The main disadvantage of the sparse reward problem is making the training sample inefficient [52]. Still, model-based RLs have been proposed recently to solve problems with extremely sparse rewards [53]. Regarding text-based GANs, policy gradient methods usually provide reward functions that sample a token per timestep without considering their surroundings are unfeasible in long sequences, [54].

The hierarchy paradigm in the text generation process can address the scalar guiding signal in the sparse reward challenge. Specifically, the text generation task is decomposed into different sub-tasks according to the hierarchical structure to make the model learn the process easier [11], [51]. Also, [11] proposes a new framework to handle the above-mentioned long text generation challenges. The so-called LeakGAN model is proposed to provide more valuable information from the discriminator to the generator using recent hierarchical reinforcement learning advances. This model uses the manager and worker modules. The generator combines such informative signals into all generation steps via a high-level Manager module that extracts features from currently generated words and results in a latent vector that leads the low-level Worker module to generate the next word. While LeakGAN is proposed for long text generation, it proved to improve the performance in short text generation applications as well, [55].

G. SUPERVISED VERSUS UNSUPERVISED TEXT GENERATION

Supervised text generation is a method to employ the existing ML algorithms that rely on large labeled data sets. RNN and CNN are two famous approaches in supervised learning wherein large data sets contain millions of labeled data. Many works use these approaches to generate texts in a supervised fashion [43], [56]–[60].

The current classifiers with small-sized training data sets cannot perform as expected compared to the large labeled data sets. Besides, the process of obtaining a large data set is time-consuming and expensive, so it's essential to find a new approach to construct a large data set in an optimized and inexpensive way. GANs seem to be a promising solution for expanding data-sets and overcoming the small data-set sizes [61]. Therefore, GANs employed the idea of adversarial training for text generation to generate a more realistic text. Text generation in supervised learning is formulated as a supervised problem. Specifically, in a particular sentence, the terms/words in the sentence can be seen as the input features while the next term/feature is the target.

The main challenges facing supervised learning can be broken down into the following points.

- The majority of supervised learning methods require large-scale and labeled data sets whose construction is costly and time-consuming.
- The predictions to generate the next character/word/sentence in the supervised model are only based on the last few inputs. Therefore, errors may rapidly accumulate and there will be a small chance to improve from earlier mistakes.

In recent literature, unsupervised text generation models are proposed to overcome challenges in the supervised models. RNNs with LSTM cells is one of the leading approaches for unsupervised models [10], [62]–[65]. We refer the reader to Section VII for an elaborate revision of recent memory-based ML architectures. Unsupervised text can be generated from explainable latent topics [66], structured data [67], [68], or knowledge graphs (KGs) [59], [65], [69], [70]. The authors in [62] propose an unsupervised model using LSTM with RNN to generate complex and realistic sentences containing long-range structures. They use the LSTM memory in RNN architecture to better store and access data compared to the conventional RNN. Also, [64] proposes a new unsupervised neural generative model for text generation. This combines VAEs and holistic attribute discriminators to effectively imposition semantic structures. This model resolves the challenge that comes from the discrete nature of text samples. Moreover, it also solves the challenge for controllable text generation related to learning disengaged latent representations.

TABLE 3. A summary of popular approaches for GANs in the scope of text generation.

Text GAN Approach (AML)	Reference	Algorithm	Advantages	Drawbacks
Classical approach	[14]	Professor forcing	Helps to overcomes the exposure bias in generated text sequences.	It does not try to render the output distribution indistinguishable from the observed data distribution. As a consequence, high sample quality may not be guaranteed.
Conventional inference/MLE	[20], [21] [23] [31]	1-HMM 2-MoM 3-RBM	1-It is a probability graph model, it can depict the transition laws of hidden states, and mine the intentional features of data to model the observable variables 2-Well-known method for learning underlying distribution. 3-Applied to evaluate model parameters using the observed data, and less computational cost than HMM	- MLE is prone to over-fitting/exposure bias issues on the training set. - MLE could not represent the actual data distribution complexity. - Doesn't perform well in high-dimensional spaces
RL-based generation	[12] [39] [40] [41] [11]	RankGAN Apply GANs in discrete state MaliGAN Actor-critic DQN	1-Solves the non-differentiability of the operation selection issue at the generator's output. Back-propagation training can be used for generator. 2- Tackles the discrete data challenge in GAN	1- It may yield high-variance gradient estimates 2- Traped or converge to a sub-optimal local minimum. 3- Having huge state-action space, leading to large portions of the space still unexplored. 4- The generated sentences' quality is poor due to the above reasons.
RL-free generation	[49] [46] [50] [7]	TextGAN GSGAN FM-GAN RelGAN	Used to address the RL- based challenges.	N.A.

III. MACHINE LEARNING ALGORITHMS FOR TEXT GENERATION

It has been proven that RNN architectures are well-suited for text generation tasks [71]. This section addresses machine learning algorithms for text generation. We discuss the shortcoming of RNNs, i.e., long text generation where LSTM and gated recurrent unit (GRU) are introduced to solve such a problem. We also discuss the difference between various text generation models. Furthermore, we highlight different metrics used for evaluating text generation such as beam search, greedy search, sequence-to-sequence models, knowledge enhancement methods, recursive transition network, relational memory, and scheduled sampling.

The state-of-the-art text generation models are based on RNNs. RNNs have difficulties when constructing, generating, and representing long texts. However, they are robust when generating local coherent text. Therefore, several papers such as [6]–[8], [71]–[76] discuss various deep learning RNN algorithms in light to this shortcoming. Improved versions of RNN, namely, LSTM and GRU are implemented in an attempt to solve long-term dependencies in text.

LSTM has been used for sentence decoding in VAE. In [6], the authors introduce Texar which is an open-source project for various text generation models. In one model, they use a conventional LSTM RNN decoder. They also used a transformer decoder. The test set perplexity and sentence–level negative log-likelihood (NLL) with the same parameter size, are evaluated for both decoders. Experiments show a significant improvement of transformer VAE over traditional LSTM VAE. Sentence segmentation into short clauses for translation by the model is presented in [77]. Overall, this approach enhances the translation of long sentences.

The authors in [7], introduce RELGAN which is a relational memory for text generation architecture to replace LSTM. This is due to LSTM's limitations such as the fact that the discriminator might be more capable than the generator and can differentiate between real versus non-real samples. There might not be enough generator capacity to distribute data based on modes due to mode crashes in current GANs. Also, there may be the poor performance of current GANs in terms of generating a long sentence. Experiments show that RELGAN outperforms LSTMs (seqGAN, RankGAN, LeakGAN) in terms of long sequence and BLEU scores. Moreover, RELGAN defeats MLE which is used as a baseline, and all other GANs in terms of the generated text samples.

Texygen [8] is a platform of text generation models. It includes metrics to measure the quality, consistency, and diversity of generated text. Models such as seqGan, MaliGAN, RankGAN, LeakGAN, TextGAN, and MLE are evaluated in terms of BLEU score on training data, BLEU score on test data, and self-BLEU score. Experiments show that LeakGAN outperforms the rest of the baseline models on the trained data. LeakGAN achieves better generalization capacity in terms of tested data where MaliGAN attains the lowest BLEU score compared to other models. Furthermore, less diverse documents are generated by all the models on the original trained data. This shows that LeakGAN and TextGAN undergo mode collapse problems when compared to the rest of the models. On the other hand, MLE and MaliGAN can generate outputs with high diversity.

A model consisting of an encoder-aligner decoder based on RNN with LSTM is presented in [78]. The encoder encodes over-determined events by applying a bidirectional LSTM-RNN. Furthermore, to generate natural language descriptions, the model uses an LSTM decoder of the selected records. The performance of their model is analyzed on the benchmark datasets ROBOCUP and WeatherGOV. In comparison to primary WeatherGOV, their model achieves better results in terms of F-1, sBLEU, and cBLEU quality metrics. Other assessment tools also include beam filter, ablation analysis, qualitative analysis, and domain independence using ROBOCUP.

In [9], the authors address the problem of generating long reviews within a neural network context. They focus on an encoder-decoder framework by examining various ways for text review generation. A multi-layer perceptron (MLP) encoder is used along with an RNN encoder with LSTM.

The performance evaluation of GANs based on RNNs with Gumbel-softmax output to generate discrete elements is presented in [46], where the model is based on LSTM RNN. The evaluation of the generative and discriminatory losses is shown throughout training. More recently, SeqGAN [10], which is a sequence generation framework to solve the problem of GANs discrete sequence generation. Experiments show that this model outperforms other baselines such as MLE, SS, PG-BLEU in terms of the NLL metric. Furthermore, LeakGAN is proposed in [11] for tackling the problem of long text generation. The model uses reinforcement learning to generate better quality information from the discriminator to the generator. This model is compared to MLE, seqGAN, and RankGAN using evaluation metrics such as NLL, BLEU, and human rating scores.

RankGAN [12] is an adversarial learning framework for creating high-quality language descriptions. This model is one of its kind to learn by relative information. The model is compared to different synthetic data methods such as MLE, PG-BLEU, and seqGAN in terms of NLL and BLEU scores. The result shows that RankGAN outperforms the state-of-the-art methods in terms of NLL and BLEU scores. Also, [79] presents a text generation model called MaskGAN. MaskGAN is compared and evaluated in terms of the ability in conditional language generation, perplexity of samples, model collapse, and human evaluation to other models. The power of large RNNs based on Hessian-free optimizer through applying them to character levels is studied in [80].

Classical approaches to text generation include template-based, rule-based, n-gram-based, and log-linear-based models. Rule-based techniques are grammar-based methods with structured rules written based on accumulated knowledge. Template-based approaches can be as simple as replacing words of users' choices by their synonyms [81]–[84].

N-gram models are widely used in NLP tasks such as text generation. In an n-gram approach, the last word of the n-gram (i.e. to be predicted) can be inferred from the other words that already appear in the same n-gram [85].

Two popular deterministic decoding approaches are beam search and greedy search [86]–[88]. Beam search maintains a fixed-size set of partially-decoded sequences. It is a common search strategy to improve results for several tasks such as text generation, machine translation, and dependency parsing. On the other hand, greedy search selects the highest probability token in each step. Thus, it can be seen as a special case of beam search.

1) SEQUENCE-TO-SEQUENCE MODELS AND KNOWLEDGE ENHANCEMENT METHODS

Sequence-to-sequence models are common architectures for text generation tasks where both the input and the output are modeled as sequences of tokens. In other words, the model converts an input sequence into an output sequence. More specifically, this setting uses two models; the first one encodes the input sequence as a set of vector representations using an RNN. The second RNN then decodes the output sequence step-by-step. Sequence-to-sequence models are commonly trained via maximum likelihood estimation (MLE) [50].

One challenge with sequence-to-sequence models is that the input text alone often does not provide enough knowledge to generate the desired output. Several methods are proposed to enhance the model's knowledge beyond input text such as attention, memory, linguistic features, graphs, pre-trained language models, and multi-task learning. Many of those techniques are listed in [89] and https://github.com/wyu97/KENLG-Reading. One of those particular enhancement techniques is attention, [87] in which an encoder compresses the input text and a decoder with an attention mechanism generates output target word(s). The decoder is bound to generate a sequence of text tokens.

The authors in [4] discuss using an RTN [90] for generating fake content similar in nature to legitimate content. RTN is used to detect simplification constructs. Nodes of the graph are labeled, and arcs may be labeled with either node names or terminal symbols. RNNs are essentially equivalent to an extension of context-free grammars in which regular expressions are allowed on the right side of production.

RTN is used to analyze an input sequence to establish its grammatical structure in relation to a given formal grammar by parsing the syntax of natural language phrases to identify the syntactic structure of a sentence. Besides, they have been used in a variety of research works on grammar development and natural language parsing strategies. Along this line [90] implements a system that provides a general facility for semantic analysis, and one of the key aims of the implementation is to examine the relationship between syntactic and semantic aspects of sentence ''understanding''. In the same research, the use of semantic knowledge to direct parsing, the reduction of the number of blind-alley analysis paths that must be taken, and the ordering of sentence analysis in terms of some measure of ''likelihood'' have all been discussed.

As a key work on parsing, [91] uses and compares several English construction and parsing techniques and their role in sentence generation. Besides, it is worth mentioning that [91] also proposes a prototype system for parsing Arabic to decide on the synthetic correctness of Arabic sentences. Due to the extensive use of grammatical relations, conjunctions, and other constructions, Arabic sentences can be complex and syntactically ambiguous. Therefore, this prototype system considered basic Arabic grammatical structures in its formation. Primarily, it consisted of examining and analyzing the grammar of Arabic language in terms of gender and numeric figures, formulating the rules using context-free grammar, representing the rules using transition networks, constructing a lexicon of words that will be used in sentences, implementing the recursive transition network parser, and testing the structure using actual Arabic sentences.

In the case of text generation, [92] proposes a system for evaluating RTNs defined in a script called "Dada engine". The package includes files that could be used in scripts that describe frequently used features, and include a format-independent method for generating formatted text. Dada Engine scripts store RTN definitions in a form of grammar as a set of rules that generate different sections of the output that aim to produce randomly generated documents in several formats. Reference [4] uses Dada Engine to generate masquerade emails by fine-tuning the grammar of Dada Engine with respect to the original author's main stylistic elements while inducing content deception that simulates masqueraded behavior to fool learning algorithms or people. Several procedures are considered to generate fake emails similar to the original email used in the Dada Engine, from combining original emails to form a dataset to be used to build a structure for generating fake emails, up to writing the email grammar by identifying key features about the compromised account's writing style and to generate other email details such as recipients email, date, time sent, and email subject.

Ideally, it is possible to generate a variety of sentences using only RTNs. However, this process is both inconvenient and time-consuming. Besides, there are certain concerns about how RTNs generate sentences [91], as follows.

- The parser may be unable to assign any rule to the input sentence due to a lexical problem in which some parts of sentences are not available in the lexicon, and
- When the parser fails to produce a rule for an input sentence because its synthetic form is not included in the grammar, the synthetic form of the sentence generated is incorrect.

It is not necessary to obtain a representation of all possible parsings of the input sentence in many applications of natural language analysis and generation. As a response, a non-deterministic algorithm is still needed, as well as the ability to discover any particular parsing. Of course, the effectiveness of this method is contingent on the existence of a system for prioritizing the semantically "most probable" parsing modes [90].

The basic idea of a relational memory is to consider a fixed set of memory slots and allow for interactions between memory slots through using self-attention mechanisms [93]. RM is proposed to record key information of the generation process, for example, record the information from previous generation processes. The goal is to enhance the text generation process through such learning/memory as well as patterns for long text generation. Such RL can provide a stateful, rather than stateless text generation process. Self-attention is also used between the memory slots to enable interaction between them and facilitate long-term dependency modeling [93]. Several relational-based text generations that showed better modeling of longer-range dependencies are described in the literature, [7], [94].

Released by Google, Google LM is a language pre-trained model trained on a billion-word corpus, a publicly available dataset containing mainly news data [95], [96]. It is based on a two-layer LSTM with 8192 units in each layer, [97], [98].

SS is proposed to bridge the gap between training and inference for sequence prediction tasks. It is used to avoid exposure bias [2] in sequence-to-sequence generation [56], [99]. During the inference process of sequence-to-sequence generation, true previous target tokens are unavailable. As a result, they are replaced by tokens generated by the model itself, which may yield a discrepancy between how the model is used at training and inference [56]. One limitation with SS is that target sequences can be incorrect in some steps since they are randomly selected from the ground-truth data, regardless of how the input is chosen [19], [100].

GANs are algorithmic architectures that place two neural networks against each other (hence the term "adversarial") to generate new, synthetic instances of data that can pass as real data.

GANs are designed to produce continuous data and have a lot of success with continuous samples such as images. GANs have been modified to produce discrete data, such as text sequences. A growing buffer of previous states has been used in a variety of other approaches to model sequential information. Some decisions must be taken about the size of the stored past-embedding buffer, whether it should be a rolling window, how computations should be cached and propagated over time, and so on.

GANs are implicit generative or language models (LMs) learned via a competition between a generator network and a discriminator network. The discriminator distinguishes uniquely GANs from other LMs. Particularly for this subject, AML, adversarial training with the discriminator is used in GANs as opposed to training based on solely maximum likelihood and categorical cross-entropy in other LMs. Conventional LMs are not trained in an adversarial manner. Unlike traditional approaches (e.g. teacher forcing, SS), GANs do not suffer from exposure bias [39], [101].

IV. ADVERSARIAL TRAINING TECHNIQUES

Adversarial examples are data points created by adding small perturbations to the input; these perturbations are incomprehensible to the human eye but lead to incorrect model outcomes. It is shown that deep learning models are vulnerable to invasion and poisoning attacks initiated using those examples [3], [102]. There have been several attempts to design algorithms to generate and defend against those examples [103]–[107]. A widely used defense technique is adversarial training, in which the training data set is augmented

[2] See Section II

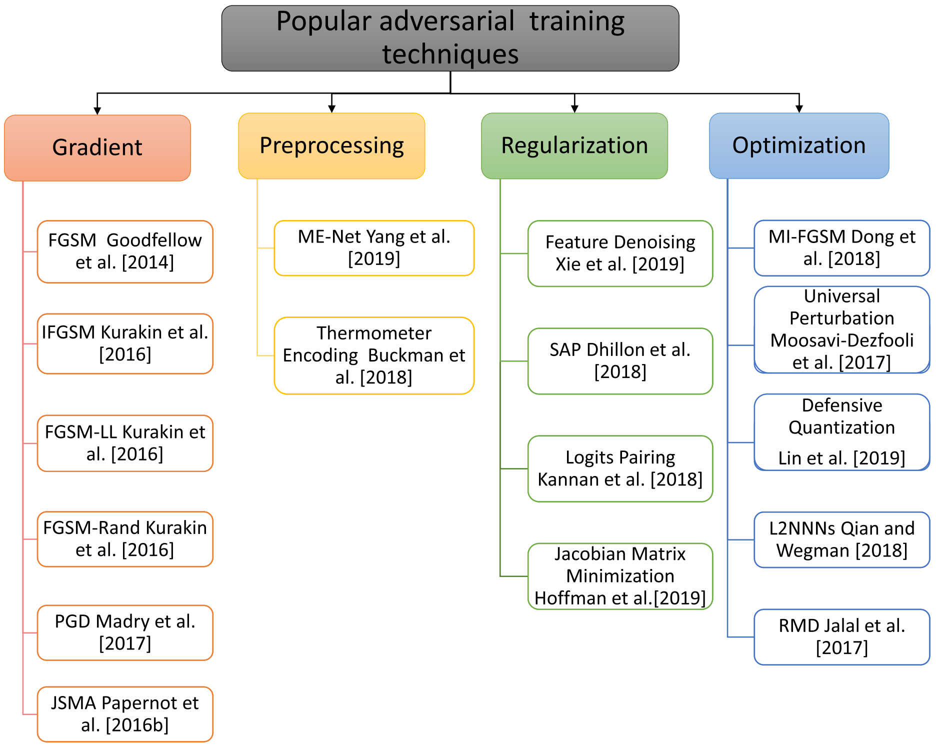

Popular adversarial training techniques
Gradient
Preprocessing
Regularization
Optimization
FGSM Goodfellow et al. [2014]
IFGSM Kurakin et al. [2016]
FGSM-LL Kurakin et al. [2016]
FGSM-Rand Kurakin et al. [2016]
PGD Madry et al. [2017]
JSMA Papernot et al. [2016b]
ME-Net Yang et al. [2019]
Thermometer Encoding Buckman et al. [2018]
Feature Denoising Xie et al. [2019]
SAP Dhillon et al. [2018]
Logits Pairing Kannan et al. [2018]
Jacobian Matrix Minimization Hoffman et al.[2019]
MI-FGSM Dong et al. [2018]
Universal Perturbation Moosavi-Dezfooli et al. [2017]
Defensive Quantization Lin et al. [2019]
L2NNNs Qian and Wegman [2018]
RMD Jalal et al. [2017]

FIGURE 2. **A categorization of popular adversarial training techniques.**

with adversarial examples in the hope that the model will generalize better and be more robust.

Adversarial training techniques can be categorized based on the strategy used, the threat model, and whether the attack using these examples is targeted or non-targeted. Fig. 2 demonstrates a summary of the different adversarial training techniques categorized based on the crafting method. In targeted attacks, adversarial examples are crafted such that a specific label is outputted from the model, while on the non-targeted attacks, the goal is to misclassify the input. Table 4 compares several adversarial training techniques based on the attack target, the threat model, and the number of iterations , and summarizes the limitations of each technique.

Below we explain several adversarial training techniques and showcase their strengths and weaknesses.

A. GRADIENT-BASED METHODS

1) FAST GRADIENT SIGN METHOD (FGSM)

Reference [3] is one of the early papers to explain deep neural networks' behavior in the presence of adversarial examples. It shows a one-shot, white-box method that generates adversarial examples. It is assumed that this method provides a stronger regularization technique compared to the traditional regularization methods in the literature [3], making the models more robust against attacks. As demonstrated in [3] the rule followed to craft an adversarial example is based on an input x, and the goal is to maximize the gradient with respect to input x. The formula is summarized in 1.

$$x^* = x + \epsilon * \text{sign}(\nabla L(\theta, x, y)), \tag{1}$$

where x^* is the adversarial example, x is the original input, y is the original label, ϵ is the size of the perturbation, L is the loss, and θ is the model's parameters. It is shown that this method is fast and efficient in generating the required examples [3], [108] as it only needs to take a step toward the gradient. The problems with FGSM are that it can lead to a label leaking issue [3] [109] and also it has a lower success rate compared to complex models [109].

There have been several attempts to lessen the label leaking problem and to improve the performance of FGSMs, including iterative fast gradient sign method [109], iterative least-likely fast gradient sign method [109], random fast gradient sign method variants [109], and momentum iterative fast gradient sign method [106]. Below we give a brief description of each method and how they work.

2) BASIC ITERATIVE FGSM (IFGSM)

Reference [109] is a variant of FGSM where the adversarial example is crafted by taking multiple steps in the direction of the gradient instead of one, shown in [109] as

$$x_0^* = x,\ x_{t+1}^* = Clip_{x,\epsilon}\{x_t^* + \epsilon * \text{sign}(\nabla_x L(\theta, x_t^*, y_{true}))\}, \tag{2}$$

[3]It is the phenomenon that during adversarial training, the validation errors on adversarial examples are smaller than the validation errors on clean examples. Thus, at the end of the training, models can solve the problem of classifying adversarial examples better than the original problem.

TABLE 4. Adversarial training techniques comparison based on the attack target, the threat model, and the number of iterations.

Technique Name and Paper	Attack Target	Threat Model	Iteration(s)	Limitations
FGSM [3]	Non-targeted	White-box	Single	Insufficient for applications with a large number of classes and works only with highly distinct classes
Basic Iterative FGSM [109]	Non-targeted	White-box	Multiple	The transferability of the attack decreases when the number of iterations increases
FGSM-LL [109]	targeted	White-box	Multiple	Limited transferability of the attack and does not perform well on black-box attacks
FGSM-Rand [109]	Non-targeted	White-box	Multiple	Limited transferability of the attack and does not perform well on black-box attacks
FGSM-momentum [106]	Non-targeted, targeted	White-box, black-box	Multiple	Limited transferability in targeted attacks
PGD [103]	Non-targeted	White-box, black-box	Multiple	High computational requirements and label leaking issues
ME-Net [110]	Non-targeted	White-box, black-box	Single	Extra overhead for preprocessing and degrades model accuracy on clean images
Feature Denoising [111]	Non-targeted	White-box, black-box	Multiple	Improved PGD with overhead work that offers no improvement for clean images and only helps with adversarial examples presence
Universal Perturbations [112]	Non-targeted	White-box	Multiple	Limited control over perturbation generated
SAP [113]	Non-targeted	White-box	Multiple	Limited to feedforward networks, and its performance depends on the balance of the training samples
Thermometer Encoding [104]	Non-targeted	White-box, black-box	Multiple	Suffers from label-leaking issues and requires high computation
Logits Pairing [114]	Non-targeted	White-box, black-box	Multiple	Limited improvement to the model robustness with a complicated process to generating the adversarial examples
DQ [105]	Non-targeted	White-box, black-box	Multiple	Results are data-dependent
L2NNN [115]	Non-targeted	White-box	Multiple	Results are bound to the ℓ_2 norm, and the performance on clean images is degraded
Jacobian Regularization [116]	Non-targeted	White-box	Multiple	Results are highly data-dependent
RMD [117]	Non-targeted	White-box	Multiple	Requires large datasets and many steps until generating acceptable results
JSMA [118]	targeted	White-box	Multiple	Limited to feed-forward networks and requires computational cost for generating the saliency map

where the operator $Clip_{x,\epsilon}$ clips the result x to be within the given ϵ.

IFGSM is a white-box, iterative, non-targeted method believed to produce more harmful examples to the model compared to FGSM. As shown in [109], with a careful selection of ϵ, IFGSM generates better adversarial examples compared to FGSM. The paper also points out that the approach gives higher top-1 and top-5 accuracy than FGSM [109]. Nevertheless, this comes with the extra computational cost for adversarial training. Furthermore, the method transferability drops when increasing the number of training iterations [119].

3) ITERATIVE LEAST-LIKELY FGSM (FGSM-LL) AND ITERATIVE RANDOM FGSM (FGSM-RAND)

[109] are white-box, iterative, targeted approaches to generate adversarial examples. Data is generated by taking multiple steps towards the gradient while minimizing the loss with respect to a target class. The formula of calculating FGSM-LL is presented in [109] as:

$$x_0^* = x, x_{t+1}^* = Clip_{x,\epsilon}\{x_t^* - \epsilon * \text{sign}(\nabla_x L(\theta, x_t^*, y_{LL}))\}. \tag{3}$$

FGSM-LL is called the least likely because the selected label is typically the least likely class predicted by the model. FGSM-Rand follows a similar approach. Still, the target label is chosen randomly. These approaches are created to overcome an issue in FGSM, which is the inability to initiate a targeted attack. Furthermore, both techniques achieve a high success rate with a minimal level of input distortion.

4) PROJECTED GRADIENT DESCENT (PGD) TRAINING

Reference [103] is a training technique that builds on the concept of IFGSM [109]. The authors train and attack deep models by solving a new optimization problem. The algorithm runs a nested loop with the optimization problem formed of two parts; the inner loop has the first one called the inner maximization problem, which has the goal of finding an adversary version of an input x that will have a high loss. On the other hand, the outer loop has the other one called the outer minimization problem, in which the aim is to find the model's parameters such that the adversity loss of a given attack is minimized. The method generates adversarial examples by selecting a point within a small round (ball) from an input image. Then, multi-step IFGSM is run to generate the examples. The authors argue that their method is universal and can be used to train models to defend against any adversary.

PGD is viewed as a robust and strong method for first-order attacks [120], [121]. It is believed to be the state-of-the-art in adversarial training [115], [121]. Nevertheless, due to the method's iterative behavior and the need to take many back-propagation steps to generate the adversarial examples ($\sim$ 10 - 40) [110], [120], the method is expected to have high computational needs. There have been several attempts to improve PGD, including using matrix approximation [110], feature denoising [111], activation pruning [113], and reuse of the training gradients [122].

5) JACOBIAN-BASED SALIENCY MAP APPROACH (JSMA)

Reference [118] is a training technique where the attacker iteratively changes a small part of the input until the model outputs the desired label. The perturbations to the input are crafted using saliency maps. Those maps are created based on the derivative from the model, and they determine which features to perturb to get the desired results from the model. Reference [118] describes the optimization problem for JSMA as:

$$\arg\min_{\delta x} \max ||\delta x|| \quad s.t. F(x+\delta_x) = y^*, \tag{4}$$

where F is the model, δ_x is the perturbation to the input x, and y^* is the target label. This method's main advantages are that it provides adversarial examples with a minimum level of distortion to the input and is independent of the model used for the learning task. However, this approach is limited only to the feed-forward networks. Even though it generates adversarial examples with minimum distortion, it is believed that this method is more computationally expensive compared to other methods [108].

B. PRE-PROCSSING-BASED METHODS

1) ME-Net

Reference [110] is a pre-processing defense technique that aims to mitigate adversarial attacks by adversarial training with matrix estimation (ME). The input data is considered as the noisy versions of the original data, and the intuition behind this method is by reconstructing a cleaner version of the images, the adversarial noise is removed, and the global structure of the data is reserved.

The ME-Net method works as follows. There is a training phase and an inference phase. During training, n masked images are generated for each training image x, wherein for each mask image, a set of random pixels are dropped. Then an ME algorithm is used to reconstruct images. The set of images are used to train the neural network while applying stochastic gradient descent (SGD) or adversarial training. In the inference phase, test images are preprocessed similarly. However, only one masked image is generated via one of the masks used during training. Then it is processed using the same ME used during training. Finally, the reconstructed image is fed to the neural network.

ME-Net can be used to defend against black-box and white-box attacks. Moreover, conventional adversarial training techniques are prone to overfitting and require large training sets to have better generalization [123]. ME-Net lifts this need by its pre-processing work. ME-Net can be combined with any training algorithm because its work is part of the pre-processing step. The method comes with the overhead cost of the estimation. Also, since the input is estimated, the method can degrade the model performance on clean images.

2) INTRODUCING NON-LINEARITY

to the models such as *thermometer encoding* is believed to help defend against the attacks [104]. Thermometer encoding can help with the discretization of the input where it discretizes the input such that the magnitude of the variable is represented while the relative distance information is preserved [104]. For example, representing the value 4 in a vector of 5 elements, the encoding will be [0, 1, 1, 1, 1]. Formally, for an index $i \in 1, \ldots, k$, the thermometer vector $\xi(i)_l \in R^k$ is defined in [104] as

$$\xi(i)_l = \begin{cases} 1 & \text{if } l \text{ is} \geq i, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

As shown in [3], models converge to mostly-linear solutions; consequently, a small change in the input will result in a big change in the output, making the models vulnerable to adversarial attacks. Thus introducing non-linearity to the models will make them more robust against adversarial attacks. The proposed idea is shown effective against white-box attacks such as [3] and [103] trained on MNIST [124]. Nevertheless, it is shown less effective on larger, more complex datasets such as CIFAR-100 [125]. The method can be adapted to black-box attacks as well, but it shows a small improvement in the defense. The method depends on [3] and [103] known to have label-leaking issues or require high computation, respectively. A possible improvement to this method is to try coupling it with methods that solve the stated issues. Another possible research track is to try exploring the usage of this method with textual input where the encoding is almost an essential part of the pre-processing and observe the effectiveness of using non-linear functions to defend against the attacks.

C. REGULARIZATION-BASED METHODS

1) FEATURE DENOISING

Reference [111] is an adversarial training approach that uses the training procedure in [103]. It is also similar to [110] in the sense that they deal with the adversarial perturbations as noise to the original data. The authors propose adding noise blocks as intermediate layers in CNNs; these blocks are typically trained end-to-end along with the neural network using adversarial training [103]. The denoising blocks consist of a denoising operation followed by a 1×1 convolution and an identity skip connection.

Four different denoising operations are suggested in the paper, namely, non-local means, bilateral filter, mean filter, and median filter. The non-local means generate a new feature map of input x by calculating a weighted average of features in all spatial locations Γ as shown in [111] as

$$y_i = \frac{1}{L(x)} \sum_{\forall j \in \Gamma} f(x_i, x_j) \cdot x_j, \tag{6}$$

where $f(x_i, x_j)$ is a weighting function and $L(x)$ is a normalization function.

Bilateral filter is a simple modification to the non-local mean where the neighborhood is a local region to x rather than pre-selected spatial locations. The formula of this filter is described in [111] as:

$$y_i = \frac{1}{L(x)} \sum_{\forall j \in \Omega(i)} f(x_i, x_j) \cdot x_j. \tag{7}$$

Mean filtering simply performs an average pooling with a stride of one. Average pooling [126] is a function that down-samples the output of a previous layer by taking the average of a rectangular portion of the feature map then reducing that potion to the average. Median filter works in a similar manner to the mean filter, but the median is taken over a local region. Calculating the median filter can be done using the formula given in [111] as:

$$y_i = \text{median}\{\forall j \in \Omega(i) : x_j\}, \tag{8}$$

where $\Omega(i)$ is the local region for calculating the median.

Although feature denoising seems to be useful in the setting of adversarial attacks, there are no benefits shown on clean data. Thus it can be viewed as an overhead work that can only help in the presence of adversarial examples.

2) STOCHASTIC ACTIVATION PRUNING (SAP)

Reference [113] is a method that can be applied to pre-trained networks as a defense mechanism against adversarial attacks. The method runs in a similar manner to the game theory setting where the attacker's goal is to maximize the loss of the defender with respect to the perturbed input, and the defender's goal is to minimize the model loss. It is a white-box method with the assumption that the adversary will craft adversary examples using the method in [3]. The optimization problem of this method is presented in [113] as:

$$\pi^*, \rho^* = \arg \min_{\pi} \max_{\rho} \mathrm{E}_{p\sim\pi, r\sim\rho}[J(M_p(\theta), x + r, y)], \tag{9}$$

where ρ is attacker policy, and π is the defender policy.

During forward propagation, a stochastic number of nodes in each layer are dropped out. Then the remaining nodes are scaled up to maintain the dynamic range of the activation in the network's layers. The advantage of this technique compared to other adversarial training techniques is the maintenance of the model accuracy. This can be achieved if the number of samples used in training is carefully selected and balanced, then the pruned model will have comparable accuracy to the original model. Another advantage is that this method can be used on pre-trained models trained to generalize well on the original task.

3) LOGITS PAIRING

Reference [114] is a regularization method that can improve models' robustness against adversarial attacks if paired with adversarial training. The basic idea of the method is to boost logits from two images to be similar. The penalization term added to the loss during the model training is demonstrated in [114] as:

$$\alpha L(J(x), J(x')), \tag{10}$$

where α is a penalization coefficient determining the strength of the pairing, $J(x)$ is the vector of logits of model J based on input x, and L is the loss function. There are three types of logit pairing proposed in the paper, clean logit pairing (CLP), logit squeezing (LSQ), and adversarial logit pairing (ALP). CLP pairs two clean images x and x'. These images are randomly picked and can have different labels. LSQ works similarly to CLP, but the loss penalizes the norm of the logits. ALP pairs a clean image x with its adversarial version x', and the model is trained to output the same label for both images. The paper claims that adding this regularization can improve the model robustness against white-box and black-box attacks. Yet, [127] shows that CLP and LSQ complicate the generation of adversarial examples with no improvement in the model's robustness. It also shows that combining ALP with adversarial training can improve the robustness of the model.

4) JACOBIAN REGULARIZATION

is another method that can be used to defend against adversarial attacks which incorporates the minimization of the norm of the Jacobian matrix to the training of the model. As demonstrated in [116], one way to enhance model robustness is to increase the classification margins of a network which can be achieved by minimizing the square of the Frobenius norm of the input-output Jacobian. The norm computation using the method of [116] can be found as:

$$\|J(x)\|_F^2 = \sum_{i,c}[\frac{\partial f_c}{\partial x_i}(x)]^2, \tag{11}$$

where f is the classification function, and c is classes in the model.

The exact computation of the norm can grow exponentially with the number of classes, thus computationally it is very costly. The authors show a theoretically guaranteed alternative to doing the exact computation which is using the expectation of an unbiased estimator. The square of the norm can be calculated using random vector projections sampled from the output. Since this is a regularization method, it should be easily added to the loss of any model and architecture. The method also has some theoretical guarantees in terms of performance and cost. Moreover, it can also be combined with adversarial training to strengthen the defense.

D. OPTIMIZATION-BASED METHODS

1) MOMENTUM ITERATIVE FAST GRADIENT SIGN METHOD (MI-FGSM)

Reference [106] is a method that combines the idea of iterative FSGM with the addition of a momentum. An adversarial example is generated using iterative FGSM by maximizing the loss with respect to the input gradient. Then it is tested on an ensemble of models. To speed up and stabilize the gradient descent algorithm performance in training, the authors use the momentum method [128]. It stores a velocity vector in the gradient direction of the loss function over iterations [106]. The vector which holds the history of the gradients helps tackling mini obstacles the algorithm might encounter such as local minima and maxima [129]. The method's formula is shown in [106] as:

$$\begin{aligned} f_{t+1} &= \epsilon \cdot f_t + \frac{\nabla_x L(x_t^*, y)}{\|\nabla_x L(x_j^*, y)\|_1} \\ x_{t+1}^* &= x_t^* + \alpha \cdot \text{sign}(f_{t+1}), \end{aligned} \tag{12}$$

where x is the sample, x_t^* is the adversarial example at t, L is the loss function of the classifier, y is the ground-truth label, and ϵ is a decay factor. One can see that (12) is the application of the sign gradient method [109] to generate x_{t+1}^*.

Model performance can be improved if an adversarial example could fool multiple models i.e. the examples have a high transferability level: [106], [130]. To test this idea, [106] proposes the *ensemble in logits* method. In this method, models are ensembled by averaging their logits as in (13), and the loss will be calculated as in (14).

$$l(x) = \sum_{k=1}^{K} w_k l_k(x), \tag{13}$$

where $l_k(x)$ are the logits of the k−th model and w_k is the ensemble weight.

$$J(x, y) = -1_y \cdot \log(\text{softmax}(l(x))), \tag{14}$$

where 1_y is the one-hot encoding of y. The method is designed so that the adversarial examples have a high transferability level. Therefore, it can be used to adversely train models against white-box and black-box attacks.

2) UNIVERSAL ADVERSARIAL PERTURBATIONS

Reference [112] is a method that builds on the concepts of the generalization of adversarial perturbations an adversarial example generated to fool a certain model can be used to fool another [3]. The authors show that their generated perturbations generalize well across data points and models. The goal is to find a perturbation vector v that can be used to fool most of the data points in the data set while satisfying the following constraints:

1) $\|a\|_p \leq \alpha$, and
2) $P_{x\sim\mu} = (k(x+v) \neq k(x)) \geq 1-\gamma$,

where α is the parameter that controls the magnitude of the perturbation γ is the control parameter for the fooling rate, and μ is a sample from the data distribution. The algorithm runs iteratively by working with a sample of data μ. Every point in μ is assumed to be placed in a classifier region, which outputs the correct label.

A minimal adversarial perturbation ∇v_i is crafted for one of the data points x_i such that the model outputs a label outside x_i's classifier region. ∇v_i is then projected to a ball with radius α to ensure it satisfies the size constraint. The proposed universal perturbation has several advantages. However, the main one is the reduced amount of computation. The crafted perturbation is image-agnostic. Therefore it can be directly added to an unseen sample without the need to compute or optimize a new perturbation, and it will produce a new adversarial example. The method is tested across several models to check if it is model-agnostic, and it showed relative success in the aspect.

3) QUANTIZATION

References [131]–[135] is an optimization technique applied to neural networks to decrease memory and computation costs. Empirical experiments in [105] show that vanilla quantization can remove small perturbations and increase network robustness against adversarial attacks. Per contra, if the input has large perturbations, the error is amplified, and the network shows poor performance and makes more errors.

4) DEFENSIVE QUANTIZATION (DQ)

Reference [105] is a neural network quantization method designed to improve the networks' efficiency and robustness at the same time. The authors suggest transforming the features to low-bit representation while controlling the Lipschitz constant to be small enough to the point where the error does not amplify. This can strengthen the robustness and lower the computational costs. The control of the Lipschitz can be achieved by adding a regularization term to the overall loss of the model described in [105] as (15), where $\mathcal{L}_{CE}$ is the original loss, and I is the identity matrix. Essentially, they are forcing the filters to be orthogonal to each other.

$$\mathcal{L} = \mathcal{L}_{CE} + \frac{\beta}{2} \sum_{W_l \in \mathcal{W}} \|W_l^T W_l - I\|^2. \tag{15}$$

The method combined with adversarial training or other defense techniques can be used to defend against white-box and black-box attacks and added the computations cost are illustrated as non-expensive computations [105].

5) L2-NON-EXPANSIVE NEURAL NETWORKS (L2NNNs)

Reference [115] are a class of well-condition neural networks built to reduce the effect of input perturbations on the neural network's output. Similar to [105], LNNNs use the concepts of restricting the Lipschitz constant of a network to defend against adversarial attacks. However, L2NNN gives more freedom to the weights of the model, namely, an upper bound is given rather than the condition of orthogonality between them as shown in [115] as:

$$\psi(W^T W) \leq b(W) \triangleq \min(r(W^T W), r(WW^T)), \tag{16}$$

where $r(M) = max_i \sum_j |M_{i,j}|$.

For a single L2NNN, the classifier's output will be affected if the perturbations to the input do not exceed the $\mathcal{L}_2$-norm of $g(x)\ /\sqrt{2}$. To achieve this, several techniques are suggested, including an updated loss function that maximizes the confidence gap between the largest and second-largest logits. They also suggest using two-sided ReLU, and norm-pooling instead of ReLU, tanh, and average pooling. The aforementioned techniques are believed to preserve the distance better than the latter ones. Preserving the distance and maximizing the average confidence gap will help in the overall robustness against adversarial attacks. The model is trained using the loss function in [103] which also improves the robustness. One issue with this method is that adversarial examples are bounded to the ℓ_2 norm, which is a limited attack model. Another issue is that the performance on clean images is degraded.

6) ROBUST MANIFOLDS DEFENSE (RMD)

Reference [117] introduces a different approach than the previous ones, which builds a model called Spanner. Spanner is a deep neural network that has low dimensional inputs and outputs an approximation of the data set. The concept of Spanner is similar to the generative models found in the literature, such as generators in GANs [20] and decoders in auto-encoder [33].

An overpower attack, which is a new type of attack designed for generative models is proposed in [117]. The problem is formulated as a min-max game, where the adversity is the max player and it searches for two adversarial images generated (via generator G) using two latent variables (z, z') such that $\|G(z) - G(z')\| \leq \gamma$. This means that they have similar features, but they are classified using classifier M differently, i.e., $M(G(z))$ and $M(G(z'))$ are different. One advantage of using this method is reducing the search space for an adversary example, the latent space (z, z') is much smaller than the original input space. The experiments in the paper demonstrate a boost of performance to the state-of-the-art methods to defend against bounded white-box attacks. It is also noted that combining this method with that of [103] during training helps in improving the performance. The problem with this method is the complexity of training the generator G. Generally speaking, generative models need large datasets and many steps until Spanner outputs acceptable results.

V. TEXT GENERATION METRICS

Recently, with increasing the interest in text generation with several applications such as translation, question answering, summarization, and more, there is a need for evaluating the generated text. Text generation is a challenging task because of the importance of considering several constraints such as linguistic structure, grammar, the meaning of the text, and semantic connections. Evaluation metrics shall measure the model quality from different perspectives. Mainly, there are two methods for evaluation; human evaluation and automatic metrics. Human evaluation is more accurate. Still, it is expensive and time-consuming where days can be spent just evaluating part of the corpus.

Automatic metrics are introduced to find cheaper and faster methods than human-based evaluation. An automatic metric is often based on two sentences a candidate and a reference, and they return a score that indicates the similarity between sentences. Several studies and researches discussed automatic metrics for different applications. With all the metrics developed, still, there is no agreed-upon metric for evaluation.

In this section, several categories of text generation metrics are discussed and compared showing the challenges and future work for evaluation metrics. Table 5 provides a summary of text generation metrics.

A. CONTRIBUTIONS IN THE CONTEXT OF TEXT GENERATION METRICS

Several text generation metrics are proposed where they are classified based on applications, usage, evaluating method, and more. categories and metrics are summarized as follows.

1) DOCUMENT SIMILARITY-BASED METRICS

BLEU [18] is one of the most used metrics for automatic evaluation of text generation. It is a precision-based metric that evaluates based on the overlapping between the generated text-based and the reference text. This is achieved by computing a score of a range between 0 and 1 where 1 indicates the best score where the generated text matches the reference text. The BLEU Score [18] is counting the number of n-gram matched between the texts where n-gram of sizes 1 to 4 with the coefficient of brevity penalty

$$BP = \begin{cases} 1 & \text{if c > r} \\ e^(1 - r/c) & \text{if } c \leq r \end{cases}$$

TABLE 5. A summary of text generation metrics.

Metric	Property	category	Fields *
BLEU [18]	Precision n-grams over-lapping	Document similarity	• Documentation generation • Machine translation
Embedding Similarity (Emb-Sim) [8]	Cosine similarity of word embedding	Document similarity	• Documentation generation • Machine translation
Okapi BM25	Retrieval model [136]	Document similarity	• Machine Translation
Word Mover's Distance [137]	Dissimilarity of word em-bedding	Document similarity	• Summarization
Log-likelihood (NLL) [3]	Minimizing NLL	Likelihood	• Documentation generation
Perplexity [138]	Minimizing distance of probabilities	Perplexity	• Machine Translation
Inception Score (IS) [26]	Prediction of the class probabilities	Inception Score (IS)	• Machine Translation • Documentation generation
Frechet Inception Distance (FID) [139]	Distance between Gaus-sian distribution including real-world samples	Frechet Inception Distance (FID)	• Machine Translation
Distinct-n [140]	Percentage of n-grams	N-gram based metrics	• Summarization
ROUGE [141]	n-gram recall	ROUGE metrics	• Documentation generation • Summarization
METEOR [142]	n-gram	METEOR	• Documentation generation • Machine Translation
Maximum Mean Discrepancy (MMD) [143]	Distance of means of dis-tributions	Metrics for GANs	• Machine Translation • Documentation generation
Test on Real (TSTR) [143]	Training of the generated instances	Metrics for GANs	• Machine Translation • Documentation generation
Kullback-Leibler divergence (KL) [144]	Divergence between two probability distributions	Divergence based Metrics	• Machine Translation • Documentation generation

$$BLEU = BP * \exp(\sum_{n=1}^{N} W_n \log P_n), \tag{17}$$

where c is the total length of generated text and r is the sum of effective reference length. Also, P_n indicates n-gram precision using n-grams until length N and positive weights W_n where it is usually selected as a uniform weight. BLEU is an effective metric. However, it does not consider the meaning of the sentences and sentence structure and does not work well with human judgments.

Embedding similarity (EmbSim) [8] is a metric inspired by BLEU [18] by comparing the word embeddings instead of comparing words by words. With EmbSim, for each word embedding, the cosine similarity is calculated with other words. Then, a similarity matrix is generated. EmbSim is calculated as follows [8].

$$EmbSim = \log(\sum_{i=1}^{N} \cos(W_i' - W_i)/N), \tag{18}$$

where W defines the similarity matrix of real data and the similarity matrix W' of generated data with i-th column. Also, N denotes the total number of words.

Word mover's distance (WMD) [137] is a metric that calculates the dissimilarities between documents. The metric aims that the embedded words of a document reach the minimum amount of distance for another document. It is based on word embedding to calculate the distance even if there are no common words. WMD retrieves the vectors from word2vec word embedding model and then uses a normalized bag of words (nBOW) to represent the weight and importance. In WMD, d and d' are the two documents in the nBOW representation, and i and j are the number of words in the documents, respectively. Also, $T_{ij} \geq 0$ shows how much of word i in d travels to word j in d'. The minimum cumulative cost [137] of moving d to d' is obtained as follows.

$$\min_{T \geq 0} \sum_{i,j=1}^{n} T_{ij} c(i, j), \tag{19}$$

subject to:

$$\sum_{j=1}^{n} T_{i_j} = d_i \quad \forall i \in \{1, \ldots..n\}. \tag{20}$$

$$\sum_{i=1}^{n} T_{i_j} = d_i' \quad \forall j \in \{1, \ldots..n\}. \tag{21}$$

2) LIKELIHOOD-BASED METRICS

Reference [3] apply a log-likelihood (NLL) term to minimize the negative of the training loss function for the binary classification task (logistic regression). The log-likelihood term ensures that the maximum value of the log of the probability occurs on the same original probability function. The system uses FGSM for adversarial training. In this setting, for training a single model to recognize labels $y \in \{-1, 1\}$ and the input $x \in \mathbb{R}^d$, the adversarial objective of logistic regression [3] is to minimize the following.

$$\mathbb{E}_{xypdata}\zeta(y(\in \|w\|_1 w^T x - b)), \tag{22}$$

where weights w and bias b are defined in $w^T\ x - b$ as where the predictions are scaled by the signed distance between x and the classification boundaries defined by the model. Also, the soft plus loss is defined as $\zeta(z) = \log(1+e^{-z})$ and the sign of the gradient is $-\operatorname{sign}(w)$ that implies $w^T\ \operatorname{sign}(w) = \|w\|_1$.

3) PERPLEXITY

perplexity [138] measures a model's certainty of its predictions. The model aims to minimize cross-entropy and perplexity to achieve the goal by following probability distributions. The method measures the distance between probabilities in discrete distributions [138] as follows.

$$PPL = 2^{(H(^P,Q))}, \tag{23}$$

where

$$H(P, Q) = -\sum_x P[X = x]l \log Q[X = x], \tag{24}$$

where H is the cross-entropy that calculates the distance between probabilities in P which contains the word distribution of the actual data and Q presents the predicted output of word's probability.

4) INCEPTION SCORE (IS)

The inception score (IS) [26] is one of the main metrics used to evaluate the quality of generative models for different instances especially within images. IS uses a pre-trained model such as the *Inception v3* model to predict the class probabilities for each generated instance. Moreover, the instances will have conditional probabilities where the instance with one classified class is considered as high quality where conditional probabilities should have low entropy. IS [26] is calculated as follows.

$$\exp(\mathbb{E}_x KL(p(y|x)\|p(y))), \tag{25}$$

where $p(y|x)$ indicates the conditional probability for each image and $p(y)$ is the marginal probability as the average of the conditional probabilities. After that KL divergence (KL) is used to measure the difference between the conditional and marginal probability distributions for each instance and then calculates the average for all classes to get the IS value.

5) FRECHET INCEPTION DISTANCE (FID)

Frechet inception distance (FID) [139], [145] is a metric that is built to enhance the evaluation of IS to include more real-world samples. In FID, the generated samples are embedded into the feature space provided by the model. The generated samples are considered as multivariate Gaussians where the mean and covariance of embedding of real and generated data are calculated. FID finds the distance between two distributions for the evaluation of the quality of the generated samples. FID Score is computed as follows.

$$d^2((m, c)(m_w, c_w)) = \|m - m_w\|_2^2 + Tr(C + C_w - 2(CC_w)^{\frac{1}{2}}). \tag{26}$$

The above-mentioned FID measure finds the square distance between the distributions of real-world data and generated samples where $\|m - m_w\|_2^2$ calculates the sum of the difference between the mean of the real data and generated samples. Also, C and C_w refer to the covariance metrics for the real and generated samples.

6) N-GRAM BASED METRICS

In [140], distinct-n calculates the percentage of the distinct n-grams in all the n-grams. Distinct-1 and Distinct-2 have been used to measure the degree of the unigram and bigram diversity. The metric is used for n-gram distinct word embedding.

7) ROUGE METRICS

ROUGE metrics [141] are mostly used for text generation with several variants such as ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE uses the recall of the number of words or n-grams in the references that appearing in the generated text or candidate text. ROUGE-N [141] is calculated as follows.

$$\text{ROUGE} - N = \frac{\sum\limits_{s \in \text{References}} \sum\limits_{\text{Gram}_n \in S} \text{Count}_{\text{Match}}(\text{Gram}_n)}{\sum\limits_{s \in \{\text{References}\}} \sum\limits_{\text{Gram}_n \in S} \text{Count}(\text{Gram}_n))}. \tag{27}$$

ROUGE counts the maximum number of n-grams occurring in a candidate text ($\text{Count}_{\text{Match}}(\text{Gram}_n)$) where n indicates the length of n-grams.

8) METEOR

METEOR [142] (metric for evaluation of translation with explicit ordering) is a metric used to evaluate text generation by computing a score based on word-to-word matches between the generated text and the reference. METEOR creates a word alignment in the reference and generated text where every word in each string maps to the closest word in the other string and the alignment is incrementally generated by sequences of word mappings. The word mapping defines all possible word matches and the largest word mappings selected. The computation of the METEOR score [142] is as follows.

$$Score = Fmean * (1 - Penalty), \tag{28}$$

where

$$Fmean = \frac{10PR}{R + 9P}, \tag{29}$$

and

$$Penalty = 0.5 * (\frac{\#chunks}{\#unigrams, matched})^3, \tag{30}$$

where P and R stand for n-gram precision and recall, respectively. Chunks are created by grouping the unigrams in the reference into the number of chunks where the longer the n-grams created fewer chunks. The score is between 0 and 1 and a higher score indicates better matching between the reference and generated text.

9) METRICS FOR GANs

Traditional probability-based LM metrics, for recurrent GAN and recurrent conditional GAN [143], maximum mean discrepancy (MMD) and train on synthetic, test on real are used for evaluation. Maximum mean discrepancy (MMD) is defined as the distance between the means of the two distributions. In GAN, the MMD finds the squared difference of the statistics between the two sets of samples (samples of real data and those generated by GAN). The kernel K is defined as: $k : x \times y \leftarrow R$, and the samples $x_i(i = 1)^N, x_i(j = 1)^M$ where MMD2 [143], is defined as follows.

$$\begin{aligned}\widehat{MMD_u^2} &= \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} K(x_i, x_j) \\ &\quad - \frac{2}{mn} \sum_{i=1}^{n} \sum_{j=1}^{m} K(x_i, y_j) + \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} K(y_i, y_j).\end{aligned} \tag{31}$$

For train on synthetic, test on real or reverse case [143], the data is generated from GAN is used to train the model and tested on a held-out set of true examples where generated data shall have labels. This method will generate a new feature based on the generated data.

10) DIVERGENCE-BASED mETRICS

Kullback-Leibler divergence (KL) [144] is a widely used tool for evaluating GANs. KL measures the divergence between two probability distributions P and Q, and it quantifies the distance between two probabilities distributions. KL [144] is defined as follows.

$$KL(P|Q) = \sum_i P_i \log \frac{P_i}{Q}, \tag{32}$$

where P represents the data distribution and Q indicates the prediction probability distribution. In KL, a relative entropy of 0 indicates that the two distributions are matched.

Another divergence-based metric is Self-BLEU [8] used in evaluating the diversity of the generated data. Self-BLEU score is calculated for each generated sentence by considering other generated sentences as reference. A higher Self-BLEU score indicates less diversity of the document.

B. TEXT GENERATION: RESEARCH CHALLENGES AND FUTURE RESEARCH TRENDS

Text generation is a challenging task which makes the evaluation using metrics more challenging. Some of the challenges in text generation metrics are:

- Text Generation is an open-ended task which means that the system can generate multiple generated texts for the same user input which makes the evaluation harder.
- The accuracy of the metric is affected by the nature of its application area.
- The accuracy of the metric is affected by the size of real data (corpus).
- Considering the meaning, structure, and semantics of the generated text is an outstanding challenge.

VI. TEXT GENERATION DATASETS

This section highlights the main datasets used in a general NLP context.

NLP and ML have come a long way in recent years and have become a fascinating field of research. Text generation or natural language generation (NLG) is one of the few developments that arose in response to the need for advanced NLP methods/techniques that use computational linguistics and ML knowledge to automatically generate natural language texts that meet conversational requirements. NLG can be achieved by training a computational model with ML heavily depending on data; it is the most important component that allows training algorithms to function, from training to tuning to model selection and testing. For NLG researchers, there are many datasets freely accessible and can be used in their research experiments. A list of dataset resources for NLP research is provided below. Besides, Table 6 summarizes how datasets from these resources can be used in NLP research.

- Aclweb.org (Natural Language Generation Portal): This portal contains lists of datasets and corpora used in natural language generation research. Maintained by ACL SIGGEN since 2005. (https://aclweb.org/aclwiki/Data_sets_for_NLG)
- Paperswithcode.com: A community of project-based at Facebook AI Research. The majority of datasets have been annotated by the *Papers with Code* team and its involved group. Data from other sources, such as NLP-progress, EFF AI metrics, SQuAD, and RedditSota, is also included. (https://paperswithcode.com/task/data-to-text-generation)

TABLE 6. A listing of available dataset for NLP research.

Dataset Resources	Usage	Reference
Aclweb.org	• Text Generation • Sentiment Analysis • Text Summarization • Image Captioning	https://aclweb.org/aclwiki/Data_sets_for_NLG
Paperswithcode.com	• Text Generation	https://paperswithcode.com/task/data-to-text-generation
Project-awesome.org	• Text Generation • Text Analysis/ Evaluation	https://project-awesome.org/tokenmill/awesome-nlg\#datasets
Github.com/niderhoff/nlp-datasets	• Text Generation • Text Analysis/ Evaluation	https://github.com/niderhoff/nlp-datasets
LIONBRIDGEAI	• Text Generation • Sentiment Analysis	https://lionbridge.ai/datasets/the-best-25-datasets-for-natural-language-processing/
Machine Learning Mastery	• Text Generation • Text Analysis/ Evaluation Image Captioning • Translators • Text Summarization	https://machinelearningmastery.com/datasets-natural-language-processing/
KDnuggets™	• Text Generation • Text Analysis/ Evaluation	https://www.kdnuggets.com/tag/datasets

- Project-awesome.org: Includes a list of NLG applications and techniques, as well as links to different systems, approaches, scientific papers, and learning materials collected by Piscis Magnus. (https://project-awesome.org/tokenmill/awesome-nlg#datasets)
- Github.com/niderhoff/nlp-datasets: A list of free/public domain datasets with text data for use in Natural Language Processing (NLP). This also provides a dataset of Arabic newspaper articles extracted from a number of Saudi news sites online. (https://github.com/niderhoff/nlp-datasets)
- LIONBRIDGEAI: Contains a list of free online NLP datasets that can be used for sentimental research, voice recognition, chatbots, text generation, and other machine learning applications. Also, it provides a high-quality and multilingual dataset for machine learning. (https://lionbridge.ai/datasets/the-best-25-datasets-for-natural-language-processing/)
- Machine learning mastery: J. Brownlee published "Datasets for Natural Language Processing" in September 2017 on the Machine Learning Mastery website, which provides a list of common datasets for NLP. (https://machinelearningmastery.com/datasets-natural-language-processing/)
- KDnuggets: G. Shapiro and M. Mayo's website about AI, analytics, big data, data mining, data science, and ML. The platform also has a large number of NLP datasets that can be used in general NL tasks and research. (https://www.kdnuggets.com/tag/datasets)

1) TEXT GENERATION DATASETS FOR GAN RESEARCH

The latest developments in text generation have culminated in the use of large-scale datasets and GAN models trained from beginning to end, without explicitly specifying the order and the context of the generated text. This area of research encompasses a wide range of topics in text generation and analysis, all of which use datasets to train and evaluate GAN models for text generation research. In addition to the dataset resources mentioned in Table 6, a few datasets/benchmarks used specifically for GAN text generation research papers are presented below and summarized in Table 7.

TABLE 7. A listing of available datasets for text generation research using GAN.

Dataset	Usage	Metrics	Reference
MS COCO Image Caption	Text Analysis/Evaluation	BLEU-1 BLEU-2 BLEU-3 BLEU-4 CIDEr Meteor	[146] [147]
EMNLP2017 WMT	Text Generation Text Analysis/ Evaluation	BLEU-1 BLEU-2 BLEU-3 BLEU-4	[11]
WeiboDial	Text Generation Text Analysis/ Evaluation	N.A.	[148]
Chinese Poem Dataset	Text Generation Text Analysis/ Evaluation	BLEU-2	[149]
CUB captions	Text Generation	N.A.	[150]

- MS COCO Image Captions: A dataset that includes human-generated captions for images containing multiple objects in their natural context. The dataset is created using images from the original MS COCO dataset used in the training, validation, and testing sets. Using Amazon's Mechanical Turk a total of 1,026,459 human-generated image captions are obtained, with 413,915 captions for 82,783 images used for training, 202,520 captions for 40,504 images used for validation and 379,249 captions for 40,775 images for testing. Since captions created by humans can differ significantly, creating an image caption dataset presents many challenges although even if two captions are very different, a person may judge them to be the same. Developing successful automated assessment metrics strongly associated with human judgment is still a challenging job [146]. Ever since the creation of the MS Coco image caption dataset, numerous research projects have used it to evaluate their models using different evaluation metrics such as BLEU-1, BLEU-2, and Meteor [147].

- EMNLP2017 WMT: This dataset contains text from papers taken from a variety of European online news sources. Along this line, [11] uses this dataset as a long text corpus in the LeakGAN text generation framework to test and generate long text. Words with a frequency of less than 4,050 are removed from the dataset, as well as sentences containing these words. Since the focus is to generate long text, sentences with a length of less than 20 are removed also. After preprocessing the dataset, it contains 5,742 terms and 397,726 sentences. A random sample of 200,000 sentences is used as a training set and 10,000 sentences as a test set to assess the framework, with BLEU 1–5 scores as evaluation metrics.
- WeiboDial: A data set from Weibo, a microblogging application launched by Sina Corporation focuses on user relationships to communicate, disseminate and get information similar to Twitter. Sina Weibo has grown to become one of China's top two social media sites with free data collection applications that can be used to collect real-time user post data for analysis, data visualization, and academic research. Reference [148] used the WeiboDial dataset to train their proposed model for a chatbot profile detector used to generate comprehensible responses in their text generation research. The data set included 9, 697, 651 post-response pairs from Weibo used to train the model decoders, as well as 7000 pairs for validation. The classifier trained on Weibo social data to detect chatbot profiles and generate responses is fairly accurate. In addition to this dataset, the following link provides access to other existing Weibo data sets, which are primarily used for academic study. (Retrieve from https://ocean.sagepub.com/blog/how-researchers-around-the-world-are-making-use-of-weibo-data).
- Chinese Poem Dataset: A Chinese poem corpus that consists of 284,899 poems obtained from several online resources. 78,859 of these are used for training and evaluating the model proposed in [149] for Chinese poem generation research based on RNN by learning representations of individual characters and their variations inside and across poem lines. The model performs both content selection and surface realization.
- CUB captions: Contains 11,788 images in the dataset, representing 200 different bird species. Each species is linked to a Wikipedia article and categorized according to scientific classification (order, family, genus, and species). An online field guide is used to compile the list of species names and a Flickr image search is used to gather the images. The bounding box, part position, and attribute labels are all labeled on each image.
- Image Captioning dataset: Proposed a tool for collecting massive datasets of images from the internet, which could be useful in replacing the MS COCO or Flickr datasets used in most previous studies. It can also be used in generating a caption indistinguishable from human-written captions. This can be achieved by adding semantic implementation to provide a simple way to insert emotion into the existing image captioning scheme [147].

VII. MEMORY-BASED MODELS

CNNs, RNNs, LSTMs, and GRUs are some of the most popular memory-based neural networks that have emerged. This section reviews the structure, characteristics, and use cases of ML models commonly used in NLP applications.

CNN is a type of neural network originally designed to map image data to output variables. Fig. 3 shows the basic architecture of a CNN network. CNN's are typically made up of a convolutional layer, a pooling layer, and a fully connected layer. The convolutional layer is traditionally a multi-dimensional matrix and could be a one-dimensional sequence. This layer analyses the input using parallel filters in order to extract different features of interest. The output is sent to the pooling layer to reduce the size of the feature map. After that, the fully connected network is used as a classification layer for computing the score of each class from the extracted features. Next, the scores of the respective classes are calculated. Finally, the classifier gives output for the corresponding classes based on the highest score.

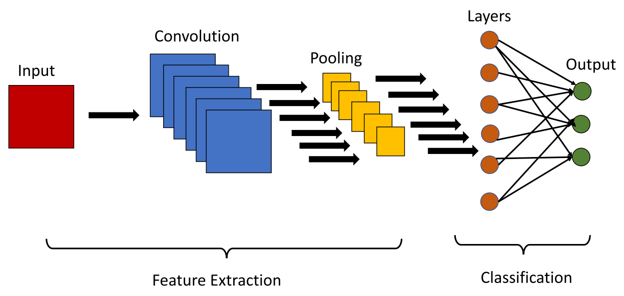


FIGURE 3. The basic architecture of a CNN.

Major applications in medical image analysis, recognition, and biometric detection use CNN-based models. The CNN architecture allows for learning the position and extraction of features in a variety of images. However, CNN's structure has the limitation of fixed-size inputs and fixed-size outputs. Later, it is used in NLP-related applications and analyzing sequential inputs, such as text, speech, and videos.

RNN is a type of neural network essentially used for processing sequential data. It simply uses feedback loops in the recurrent layer in order to allow for maintaining information over time. This lets the previous information in the sequence produce the current output. Fig. 4 illustrates the

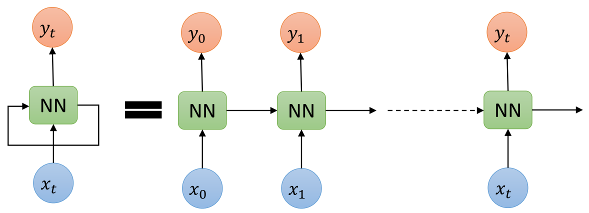


FIGURE 4. The basic structure of an RNN with a loop.

basic structure of RNN where x_t is a vector of inputs at a time slice t, h_t is the hidden vector sequence for time slices 1 to t. The loop allows information to be passed from one step in the network to the next one.

RNNs are useful for sequence generation applications such as machine translation, as well as other models where the input is not a sequence such as in image captioning. Moreover, RNN works well with sequence perdition problems. Sequence prediction problems involve different combinations of inputs and outputs such as *one-to-many, many-to-one, and many-to-many* (a.k.a. sequence-to-sequence). Still, a drawback of RNNs is their poor performance when the learning sequences have long-term temporal dependence [56]. For example, processing text to do prediction using RNN may overlook important information from the beginning. Specifically, RNNs have a difficulty in carrying information from early steps to later ones due to an issue called vanishing gradients. Gradients are values used to update NN weights. The vanishing gradient issue occurs when the gradient becomes smaller as it back propagates through time. Accordingly, [56] proposes a curriculum learning process to train RNNs to produce a sequence of tokens given some input. This leads to performance improvements using a short training period. The proposed model is an example of a scheduled sampling process for sequence prediction using RNN. In essence, this is a curriculum learning approach for sequence prediction tasks where the system explores more options during the training task.

As an alternative to RNNs, LSTM models are effective approaches in the field of sequential modeling methods. LSTMs are special types of RNN that overcome the limitations of RNN by introducing memory cells within four neural network layers in their structure instead of a single one. Figures 5 and 6 illustrate the basic structures of RNNs and LSTMS. In these figures, where the horizontal line at the top represents the cell state. LSTM models provide a mechanism to both store and discard the information saved about the previous steps. This happens while limiting the accumulated error using Constant Error Carousels [151]. LSTM removes or adds information to the cell state using input gate, and forget gate, respectively [152]. Likewise, a GRU is introduced to solve similar types of problems such as LSTMs. In terms of their architecture, LSTMs and GRUs are types of RNN that use special units in addition to standard units. GRUs can

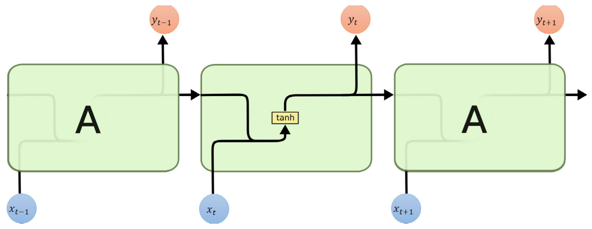


FIGURE 5. The basic structure of an RNN [156].

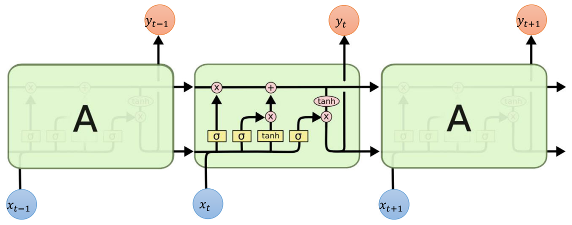


FIGURE 6. The basic structure of an LSTM [156].

better learn long-term patterns [153], [154]. The design of the forget gate is the core of the LSTM and GRU models [155].

GRUs are similar to LSTMs, but they have a simplified structure. Fig. 7 illustrates the structure of LSTM and GRU. GRUs use fewer gates which makes the model faster. They make use of a memory cell to store the value of previous words in the long sequences. A set of gates are is used to control the flow of information in the network and to learn which inputs in the sentence are important to store in the memory units. Therefore, GRU requires fewer network parameters. Nevertheless, LSTM provides better performance than GRU, if one has enough data and computational power [157].

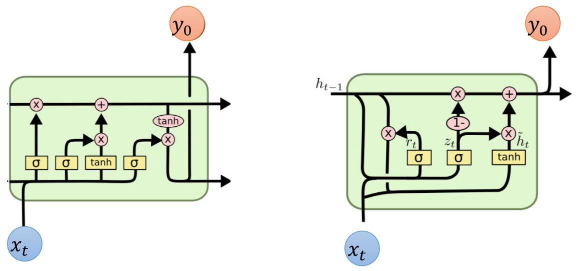


FIGURE 7. The structure of LSTMs and GRUs [156].

Reference [158] illustrates an example of an LSTM application for automated music generation by combining raw and symbolic audio models. The raw audio models represent the realistic sound and feel of the music, while the symbolic models represent the complexity, structure, and long-range dependency of the generations. This model can inspire many future applications.

Tables 8 and 9 summarize the differences between popular memory-based models and list the advantages and disadvantages of these models.

TABLE 8. A summary of advantages and drawbacks of memory-based models.

Model	Advantages	Drawbacks
CNN	- Used in classification and prediction applications. More suitable for processing image and video data - Captures the spatial features - Supports parameter sharing - Used to reduce number of parameters needed to train without sacrificing performance	- Suffers from gradient vanishing and exploding problems - Doesn't capture the spatial features
RNN	- Performs well with NLP especially sequence prediction problems - Useful for solving temporal data problems (i.e. time series) - Supports parameter sharing	- Suffers from vanishing gradient problems - Training an RNN is difficult - Cannot process very long sequences - Not suitable for processing image and video data input. - Not appropriate for tabular datasets data input - Poor performance with time series forecasting problems
LSTM	- Overcomes the issues of training a recurrent network and vanishing gradient problems of RNN - Suitable to classify, process and predict time series - Provides better performance than RNN, if enough data and computational power are available - Captures the dependency across time sequences in the input vector - Looks at long sequences of inputs without increasing the network size	- Slow to train - Poor performance with time series forecasting problems
GRU	- Similar to LSTM but uses a simplified structure without the need for memory cells - Faster to train than LSTM - Effective in learning long-term dependencies - Captures the dependency across time sequences in the input vector - Generally used with long sequence training samples while expect a quick and decent accuracy	- Slow convergence - Low learning efficiency

TABLE 9. A summary of differences between popular memory-based models.

Model	Parameter sharing	Spatial relationship	Temporal information processing	Vanishing	Exploding Gradient	Best for
CNN	No	Yes	Yes	No	Yes	Images
RNN	Yes	Yes	No	Yes	Yes	Sequences
LSTM	Yes	Yes	No	Yes	No	Sequences
GRU	Yes	Yes	No	Yes	No	Sequences

VIII. DEFENSE AGAINST NLP ADVERSARIAL ATTACKS

As discussed earlier, the discrete nature of text creates difficulties in generating adversarial examples in NLP applications. There is a research body on NLP attacks and defenses at character, word, and sentence levels. Table 10 revises popular NLP adversarial attacks, attack granularities, and defense strategies.

TABLE 10. NLP adversarial attacks, attack granularity, and defense strategies.

NLP Adversarial Attacks	Attack Granularity	Defense Strategy	References
Reading comprehension	Word-level	Stanford Question Answering Dataset (SQuAD) scheme where systems are evaluated on adversarially-chosen inputs	[164]
Text classification	Word-level	Learning to DIScriminate Perturbations (DISP) framework to define and modify malicious perturbations	[168]
Text classification	Word-level	Black-box population-based optimization algorithm	[169]
Text classification	Word-level	Insertion, modification, and removal	[170]
Text classification	Word-level	DANCin SEQ2SEQ	[171]
Neural Machine Translation	Word-level	A combination of projected gradient method, group lasso, and gradient regularization	[172]
Neural Machine Translation	Character-level	White-box optimization, targeted attacks, evaluation method, and training	[173]
Dialogue systems	Word-level	White-box method, gradient-based approach guided by translation loss to generate adversarial examples	[174]
Dependency parsing	Sentence-level	GAN to train an explanation generator to produce explanations for features used in latent variables	[175]

Current adversarial attacks can be roughly be divided into three categories: white-box attacks, black-box, and gray-box attacks, according to whether the data, model architecture, and parameters of the target are accessible. In black-box attacks (also called zero-knowledge attacks), no or very limited information about the target model is accessible. For example, a certain number of model queries (i.e. oracle queries) are granted.

Some of the defenses, [11], [34] are shown to be quite robust against black-box attacks. In gray-box attacks/limited knowledge attacks, partial knowledge about the model under attack (e.g., type of features, or type of training data) is assumed. On the other side, is white-box (perfect-knowledge) attacks. Those exploit model internal information. They assume complete knowledge of the targeted model, including its parameter values, architecture, training method, and in some cases its training data.

These are examples of defense against specific attacks in an NLP context. Below is a concise revision of these algorithms and approaches, their rationale, and how they evolved.

- Dirichlet neighborhood ensemble, a randomized smoothing method for training a model against substitution-based attacks [159]
- Adversarial training as a defense method, [160]–[162]
- Increasing model's robustness by adding perturbations on word embedding [3]
- Certified defenses: aim to provide guarantees of robustness to some specific types of attacks [163], [164]
- Defensive distillation: Defensive distillation can take an arbitrary NN and increase its robustness, reducing the success rate of attacks' ability [165]
- Defense through randomization [166], [167]

An example work on adversarial evaluation in NLP is [164]. A scheme called the Stanford question answering dataset (SQuAD) is proposed for reading comprehension tasks then answer questions on paragraphs from Wikipedia. The authors are doubtful that current systems have correct language understanding and reasoning capabilities. To improve adversarial evaluation, the authors suggest the need for new strategies for training models.

In the context of text classification, [168] proposes a framework, called learning to discriminate perturbations (DISP), to identify and modify malicious perturbations, which accordingly helps in blocking NLP adversarial attacks. The model is represented against adversarial attacks without changing the model structure and the training procedure. DISP has three parts: perturbation discriminator, embedding estimator, and hierarchical navigable small-world graphs. The perturbation discriminator classifies a group of perturbed tokens. The framework organizes an efficient k nearest neighbor (KNN) search across a hierarchical taxonomy toward translating each of the embedding vectors to a suitable token for changing the related perturbed word.

Another work on text generation is [169] which uses a black-box population-based optimization algorithm. The authors produce semantically and syntactically similar adversarial examples versus models trained on sentiment analysis task as well as the Stanford natural language inference (SNLI) textual entailment task. They validate the examples, which are correctly categorized through human evaluators and like the original via a human study. Defense against such adversarial attacks is performed using adversarial training. However, it fails to yield any robustness, demonstrating the strength and diversity of the produced adversarial examples.

From a deep neural network (DNN) perspective, [170] introduces a technique for crafting adversarial samples against DNN-based text classifiers. Rather than simply overlapping the perturbation and the original input, the authors' design three perturbation tactics (insertion, modification, and removal) and establish the natural language watermarking method to elaborately dress up a given text to generate an adversarial example. The proposed technique carries out either white-box or black-box attacks to various adversarial scenarios. For the model whose implementation can be freely and completely examined by the adversaries, the cost gradients of the input are calculated to effectively decide what, where, and how to insert, modify or remove. Then, if the objective model cannot be directly analyzed, some occluded tests are produced to probe it and obtain the needed usable knowledge.

In a binary classification context, [171] proposes a technique to generate adversarial text examples which fool or increase the probability of misclassification. Text classifiers are now arguably approximately of the most organized machine learning models, used in high-impact domains ranging on or after spam classification to medical record analysis. On the contrary, several standard methods used to generate adversarial samples have largely focused on image classifiers, and depend on gradient-based, visually imperceptible changes to current images which do not simply apply to discrete and semantically meaningful text sequences.

2) MACHINE TRANSLATION

Considering the applications of machine translation, [172] proposes a framework, called Seq2Sick, for generating adversarial attacks against sequence-to-sequence models. Seq2Sick uses non-overlapping and targeted keyword attacks. The framework achieves a high success rate (84% -100%) in both types of attacks with one or two keywords. This approach employs group LASSO [4] regularization and the projected gradient method to simultaneously search all replacement positions as opposed to the majority of approaches, which are based on greedy search. The authors evaluate the robustness of the proposed model on multiple applications such as text summarization and machine translation.

Another work on neural machine translation (NMT) is [173] which proposes a white-box character-based NMT system for generating adversarial attacks. The proposed approach is built on top of the HotFlip method [176] for generating adversarial examples. The robustness of the approach is further improved with faster adversarial training. The authors propose a two-level type of attack: a controlled attack aiming to remove a specific word from the output, and a targeted attack aims to replace a specific word with another one. The proposed system outperforms its black-box counterparts. Both [172] and [173] use gradient-based estimate to rank adversarial manipulations, then they search for adversarial examples using greedy search or beam search. Overall, Seq2Sick achieves a high success rate of almost 100%, while [173] is less as successful.

3) DIALOGUE SYSTEMS

The authors in [174] improve the robustness of NMT models compared to baseline methods using adversarial examples generated by a white-box, guardian-based method guided by translation loss from the inputs of an NMT model. This approach allows access to the parameter level of the attacker. Both [174] and [173] use white-box methods. However, the former applied it at the word level while the latter applied it at the character level. Generally speaking, latent variables in DNNs are uninterpretable. Hence, it is difficult to explain how adversarial perturbations influence features in these variables leading to final misclassification.

4) DEPENDENCY PARSING

[175] proposed interpretability and diagnostic method, called InterpretGAN, to understand and diagnose vulnerability attacks. The method applies GAN to train an explanation generator to produce explanations for features used in latent variables. Interpretability results can be further used to improve model performance.

[4]Least absolute shrinkage and selection operator, which is an algorithm for sparse approximation.

A. BLACK VERSUS WHITE-BOX ATTACKS

Considering long text generation, [11] proposes an algorithmic framework called LeakGAN based on adversarial training. The proposed method addresses the sparsity and non-informativeness issues of the scalar reward signal in previous GAN solutions ([12], [39], and [40]) by leaking of the feature that has been extracted by the discriminator, as the step-by-step guiding signal, which guides the generator to produce better long text.

[11] conducts experiments based on synthetic and real data, which include long, mid-length and short text. Dealing with synthetic data, LeakGAN showed a lower NLL than previous GAN solutions using sequence lengths set to 20 and 40. Dealing with real data, the text in EMNLP2017 WMT News, COCO Image Caption, and Chinese Poems had been used as the long, mid-length, and short text corpus, respectively. using all these datasets, the proposed method showed better results compared to previous GAN models in terms of BLEU statistics [18] and the human Turing test.

[34] proposes using randomization at inference time to mitigate adversarial attacks' effects. The authors use two randomization procedures: random padding, which pads zeros around the input images in a random manner, and random resizing, which resizes the input images to a random size. In this regard, the impact of randomization on the genratyion of adversarial exmaples is analyzed. Results show that adversarial examples hardly transfer between different randomization patterns, particularly for iterative attacks. Furthermore, the randomization layers of the proposed technique are compatible with different network structures and adversarial defense methods. Thus, they can work as basic components for defense against different adversarial examples.

A literature review on the special effects of different types of attacks on machine learning systems and their defenses against these types of attacks is presented in [2]. This work also proposes the a spam detection system called SpamBayes. The authors then argue on possible attacks against this system and quantify their success rates. There proposed framework offers a good foundation for analyzing attacks on machine learning systems and developing defense mechanisms against them.

From another perspective, [159] proposes a method called Dirichlet neighborhood ensemble to generate virtual sentences by mixing the embedding of the original word in the input sentence with its synonyms. By training on these virtual sentences, the proposed model improves the robustness against word substitution-based attacks.

Reference [159] tests a new defense method over experimental results conducted using several model architectures (bag-of-words, CNN, LSTM, and attention-based) on several data sets. These results shows that the proposed method achieves better performance on clean and adversarial examples compared with other defense techniques. Furthermore, [159] finds that one can enable the embedding of a set of similar words to be updated together in a coordinated way. This idea is then exploited in improving the robustness of NLP models against adversarial attacks.

There has been an increasing interest in extending adversarial training for the purpose of defense. Along this line, [160] extends adversarial and virtual adversarial training to the text domain by occupying perturbations to the word embedding in an RNN rather than to the original input itself. A major finding in this work is pointing out that adversarial and virtual adversarial training have good regularization performances in sequence models on text classification tasks. Another finding is that adversarial and virtual adversarial training enhance not only classification performance but also the quality of word embedding.

Another research trend considers developing our understanding of the evolution of adversarial examples, their reasoning, and how they work. As an example, [161] discusses the interpretability of adversarial training based on adversarial perturbation applied to tasks in the NLP field. Specifically, [161] controls the directions of perturbations toward the locations of existing words in the word embedding space. Consequently, the authors could directly restructure each input with perturbations to an actual text by reflecting the perturbations to be the replacement of words in the sentence while maintaining or even improving the task performance.

In an attempt at developing better understanding of model's behavior under attack, [161] shows that an attacker can positively generate reasonable adversarial texts and interpretable visualizations of perturbations in the input embedding space, which they assume will help researchers analyzing a model's behavior. The presented results confirm that the proposed technique maintains or improves the state-of-the-art performance obtained by baseline methods, AdvT-Text and VAT-Text, in well-studied sentiment classification, grammatical error detection, and category classification benchmark datasets. From another perspective, [162] reports that adversarial training considerably improves the performance of state-of-the-art models for various language understanding tasks. Along this line, an adversarial training algorithm, known as FreeLB (Free Large-Batch), is proposed. This algorithm adds adversarial perturbations to word embeddings and minimizes the resultant adversarial loss around input samples. Besdies it can improve Transformer-based models (BERT, RoBERTa, and ALBERT) on several datasets and achieve the new state-of-the-art on GLUE and ARC benchmarks. This algorithm is experimentally verified to score higher robustness in the embedding space compared to natural training, and better generalization capability.

It is noted that the early attempts to explain adversarial examples focus on overfitting and nonlinearity. Conversely, [3] argues that the main cause of neural networks' vulnerability to adversarial perturbation is their linearity, which is supported by experimental results showed in the paper while explaining the highest interesting fact about them, which is their generalization across architectures and training sets. Furthermore, this view leads to developing a simple and fast technique for generating adversarial examples. Using this method to create examples for adversarial training [3] reduces the test set error of a maxout network on the MNIST dataset.

Another line of research considers studying system's robustness against predefined classes of adversarial attacks. Along this line, [163] considers text classification under synonym replacements or character flip perturbations. This is achieved by modeling these input perturbations as a simplex and then using interval bound propagation. The resulting models show only a minor difference in terms of nominal accuracy. However, they present considerably enhanced verified accuracy under perturbations and it came with an efficiently computable formal guarantee on worst case adversaries.

The used standard accuracy metrics show that reading comprehension systems are achieving rapid progress. However, they do not evaluate these system's capability to understand language. To evaluate the real language understanding abilities of the reading systems, [164] proposes an adversarial evaluation method for the Stanford question answering dataset (SQuAD). The proposed method examines whether reading systems can answer questions about paragraphs that contain adversarial inserted sentences, which are generated to confuse computer systems without changing the right answer or misleading humans. This work considers defensive distillation as a defense mechanism. Int his setting, one uses the logits of the (possibly backdoored) trained model to retrain for a benign one, using benign data.

The experimental results presented in [164] show that no published open-source model is robust against adversarial sentences. Furthermore, a major contribution in this work is showing that defensive distillation does not meaningfully increase the robustness of neural networks. This observation is experimentally verified on three types of attack algorithms, which are successful on both un-distilled and distilled neural networks with 100% probability. Nearly at the same time, [165] proposes adversarial attacks that overcome defensive distillation, demonstrating that these attacks could be used to evaluate the potential defense efficiency. The generated attacks from the proposed algorithm are tailored to three types of distance metrics that used previously in the literature. Then, they, when compared to previous adversarial example generation algorithms, the generated attacks are regularly much more effective.

A recent research trend considers merging multiple defense techniques to improve the resilience of ML models against attacks. As an example, [167] proposes a new defense method, named as Random SelfEnsemble (RSE), by combining two concepts: ensemble and randomness. This is done by adding a new "noise layer" that merges input vector with randomly generated noise. Next, this new layer is inserted before each convolution layer of a deep network. Experimental results show that this method makes the network more resistant to adversarial attacks. Moreover, RSE slightly affects test accuracy when no attack is performed on natural images.

Following the line or randomization, a new defense technique referred to as randomized smoothing has been emerging recently. This technique delivers better-certified accuracy because it permits the utilization of larger networks and it does not constrain the expressively of the base classifier. In this regard, [166] uses randomized smoothing to train state-of-the-art certifiably $\mathcal{L}_2$-robust ImageNet classifiers. As an example, one of the trained classifies scores 49% provable top-1 accuracy under adversarial perturbations with $\mathcal{L}_2$ norm less than 127/255. Reference [166] also show that even on smaller-scale datasets, they can challenge approaches to certified $\mathcal{L}_2$ robustness. An important fidnign in this work is showing that randomized smoothing is a promising direction on future research for adversarial robust classification.

As a final remark, a comparison of the mentioned papers in this section is presented in Table 11.

IX. SUMMARY AND RESEARCH TRENDS

In this paper, recent literature in adversarial machine learning for text generation tasks is summarized. We noticed a continuous expansion in the applications, models, and algorithms. This paper can serve as an introduction to this field and readers may need to follow through with some of the researchers and references we referred to based on their focuses or interests. Our goal is to present a one-stop source for researchers and interested readers to learn the basic components, explore the main research trends, and envision future research in this field.

The applications of machine-based automatic text-generation span many domains and use cases. Automatic text generators replace humans for saving time, effort, and resources. However, they are vulnerable to be malicious in some cases. This is especially the case when such generators try to masquerade as humans. Popular recent examples include the usage of Trolls and Social-Bots to act as regular citizens in online social networks. The goal of such trolls/bots is to create politically manipulated text that can influence public opinions [177].

A. RESEARCH CHALLENGES AND FUTURE RESEARCH TRENDS

Now that we have revised key approaches for attack and defense ML models for text generation, it is time to explore and envision possible areas of research to improve the knowledge circle in this domain. We categorize these into the scopes of GAN text generation, text generation metrics, memory-based ML models associated used in NLP settings, defense against NLP adversarial attacks, and adversarial training, as detailed below. Besides, Table 12 summarizes outstanding directions for future work.

1) GAN TEXT GENERATION

1) Exploring new NLP application areas and contemporary use cases for GANs. An example along this line is non-goal-oriented dialog systems, with no specific training and assessment criteria.

TABLE 11. A compression of the main surveyed papers in this subsection.

Research Paper	Black/White/Gray Attacks	Types of Attacks its dealing with (ex. substitution-based attacks)	Method	Important Notes
Improving NLP tasks by Adversarial Training				
[11]	White Attacks	N/A	N/A	- Improve text generation quality using adversarial training
[160]	White Attacks	N/A	Applying adversarial and virtual adversarial training to the text domain by applying perturbations to the word embedding in an RNN	- Improve text classification performance & quality of word embedding using adversarial training
[161]	White Attacks	N/A	Improving NLP tasks by restricted the directions of perturbations toward the locations of existing words in the word embedding space	- Improving NLP tasks using adversarial training
[162]	White Attacks	N/A	Proposing a novel adversarial training algorithm, called FreeLB to improve performance of models for many language understanding tasks	- Enhanced adversarial training for language understanding
Improving NLP/Neural Networks robustness against Adversarial Attacks				
[34]	Black Attacks	Single-step and iterative attacks	using randomization-based mechanism (Random Resizing & Random Padding) to mitigate adversarial effects	- None
[3]	N/A	N/A	N/A	- The authors argued that the primary cause of neural networks' vulnerability to adversarial perturbation is their linear nature
[166]	Black Attacks	Adversarial Perturbations	Increase classified adversarial robustness via randomized smoothing	- Increasing ImageNet classifiers robustness against adversarial perturbations
[167]	White Attacks	N/A	Adding a new defense algorithm called Random Self-Ensemble (RSE) by combining two important concepts: randomness and ensemble	- Improving the robustness of deep robust neural networks against adversarial attacks
Adversarial Attacks in NLP/Neural Networks				
[159]	Black Attacks	Substitution-based attacks	Creating virtual sentences by mixing the embedding of the original word in the input sentence with its synonyms	- Virtual sentences by mixing the embedding of the original word in the input sentence with its synonyms
[163]	White Attacks	Synonym Replacements & Character Flip Perturbations	Modifying the conventional loglikelihood training objective to train models	- None
[164]	White Attacks	Distracting Sentences	Creating adversarial examples by adding distracting sentences	- Proposing adversarial evaluation scheme for evaluating reading comprehension systems
[165]	White Attacks	N/A	Creating white attacks adversarial examples	- Proposing adversarial examples to evaluate the robustness of neural networks
Identifying Attacks against ML Systems				
[2]	N/A	N/A	N/A	- Identifying & analyzing attacks against ML systems - Evaluating ML systems for security Apps

TABLE 12. A table summary of horizons for future work.

GAN Text Generation	Text Generation Metrics	Memory-based DNN Models	Defense Measures	Adversarial Training
New applications	Automated metrics	New ML models	Exploring attack types	Cost efficiency
Exploiting word embeddings	Human judgement association	Memory utilization	Better adversarial examples	Benchmarking
Creating latent representations	Multilingual metrics	Better sampling decisions	New defense approaches	Applicability from computer vision
Developing universal perturbations	Using ML	Error mitigation	Improving GAN discriminators	Data and model complexity
			Identifying model vulnerabilities	Robustness and scailability

2) The exploitation of word and sentence embeddings to provide an extra pre-training stage for the machine learning model. This includes embeddings from large databases such as those of Google and conditional generation cases such as dialogue generation
3) Using conditional GANs to create latent representation for writing types. This usage will allow for a seamless lexical and grammatical transition between various writing styles.
4) Developing universal text perturbations to be used in both black- and white-box attack settings.

2) TEXT GENERATION METRICS

1) Developing successful automated assessment metrics
2) Associating such metrics with human judgement [146]
3) Developing multilingual text generation metrics
4) Successfully making use of ML to develop, train and deploy quality metrics having desirable properties, including the aforementioned points

3) MEMORY-BASED DNN MODELS

1) Designing and developing new memory models that increase performance and memory utilization efficiency without validating real-time constraints
2) Exploring better sampling decisions to minimize back-propagation of errors, enriching knowledge from external memory inherited from a knowledge base (top-down) can contribute to designing appropriate networks for emerging real-world problems.

4) DEFENSE AGAINST NLP ADVERSARIAL ATTACKS

1) Exploring different types of targeted attacks especially those that target more than one word, with different types of constraints and characterizations.
2) Investigating the direction of generating more natural adversarial examples dispensing with word replacements
3) Developing new defense approaches with improved performance through adversarial training
4) Reintroducing a true GAN-style discriminator, updated during training, to distinguish between human and machine-generated instances.
5) Complex classification tasks and model architectures might yield similar explainable insights. Such an approach might offer a useful outline for identifying important target model vulnerabilities and limitations.

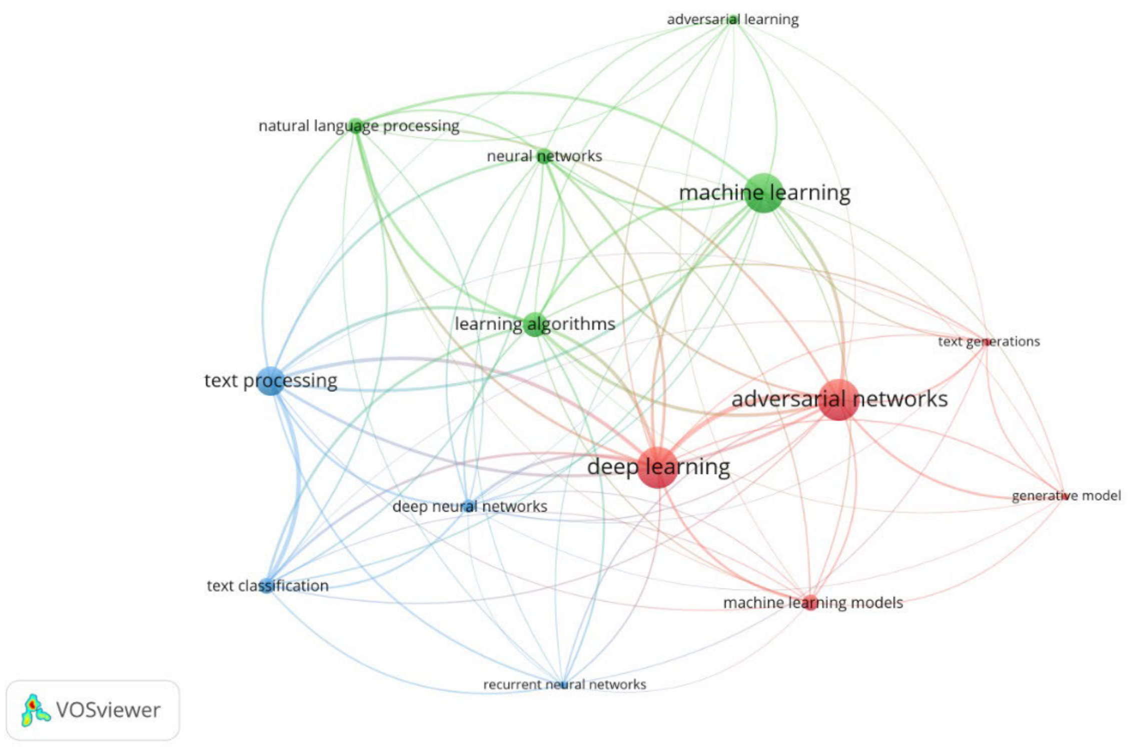


FIGURE 8. Machine learning for text generation tasks related keywords.

5) ADVERSARIAL TRAINING

Nowadays, adversarial training is considered to be the state-of-the-art defense against several adversarial attacks [178]. Despite its robustness, the method has some shortcomings, highlighted below.

1) Computational cost: Most of the mentioned techniques require high computational needs, typically much more extensive than what the model requires to train naturally [49]. A potential track of research is to find new training techniques that require less computational requirements and accelerate the training process. As an example, [122] suggests free adversarial training, which optimizes SGD passes such that during the backward pass, the model's gradient is computed with respect to the network parameters. Simultaneously, the gradient of the loss with respect to the input is computed [122]. The paper shows an interesting level of speed-up compared to the original training model [103] with comparable accuracy. Another example work is [49] where the authors propose a new training method that reduces the number of passes required to generate an adversarial example to one. Yet, even with the improvement of run time, it is still notably slower than regular training [121]. Other papers suggest using FGSM [3] with random initialization is sufficient to defend against PGD [103] attacks and with a much lower computational cost. This approach can be further improved by coupling it with several efficient training procedures shown in [179] such as mixed-precision arithmetic [180], and cyclic learning rate [181].
2) Lack of benchmark algorithms: Adversarial training techniques are generally tested and compared based on empirical experiments. Therefore, a possible improvement is to provide rigorous and theoretical guarantees to their effectiveness against the attacks.
3) NLP vs computer vision: The vast majority of adversarial attack and defense techniques are mainly concerned with computer vision applications. Thus, they can not be arbitrarily applied to NLP domains. In view of this limitation, an seemingly beneficial area of research is to investigate how to exploit the research outcomes already attained in computer vision and accustom them to be suitable for NLP application areas.
4) Size of training dataset and models complexity: The majority of the discussed methods are not tested on large datasets such as ImageNet, neither on complex models [122].
5) This suggests the need for investigating robustness against adversarial attacks on more complex and larger datasets.

6) RESEARCH TRENDS VISUALIZATION

Many studies have been conducted in adversarial machine learning for text generation problems, with various focus

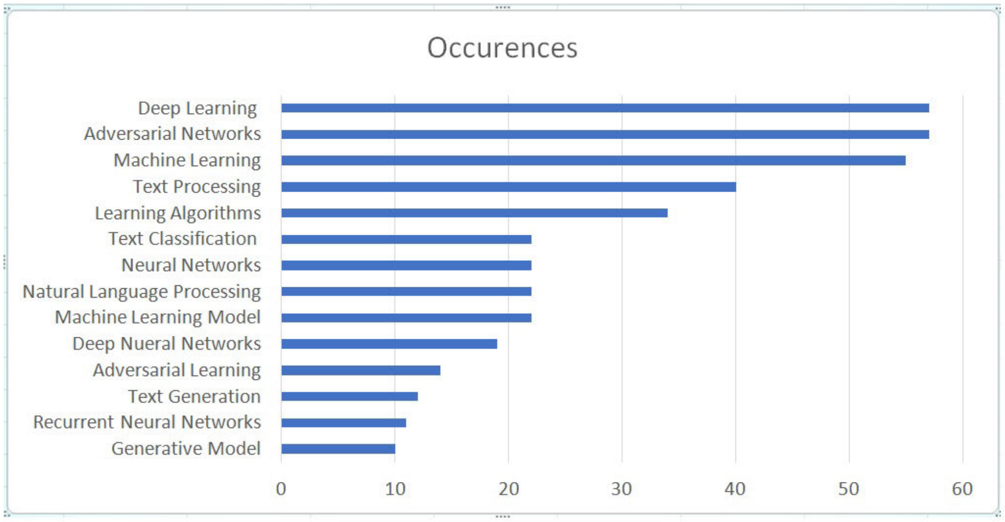


FIGURE 9. Related keywords ranking in machine learning for text generation tasks.

and subjects, where finding and reviewing relevant articles is a time-consuming process while preparing a research article. Keywords are used by most electronic search engines, databases, and journal websites to determine whether to show a research article to interested readers. One of the primary functions of keywords in a research paper is to assist other researchers in locating articles relevant to their research topic, where the article's keywords define the field, sub-field, topic, and research concern. Similarly, in research articles, it is essential to incorporate the most relevant keywords to help other authors find your paper and improve article citation. Therefore, we produced a graph using the indexed keywords from the research articles reviewed in this survey extracted from the Scopus database. This graph illustrates the most prevalent keywords to assist future researchers in finding relevant publications in online resources present machine learning for text generation research topics. Figure 8 shows the machine learning for text generation tasks related keywords graph.

To this end, it is interesting to consider the tasks of applications in the papers surveyed in this review. For this purpose, we show the most repetitive keywords appearing these references in Figure 8. The figure reveals that the following are the most frequently appearing keywords: (1) Deep learning, (2) Adversarial networks, (3) Machine learning, (4) Text processing, and (5) Learning algorithms. Furthermore, we present the keyword ranking in Figure 9.

In text processing, adversarial machine learning creates fierce competition among researchers. As more researchers decide to pursue this field of study, the number of available information increases. We expect that by listing the most used keywords in this field, future researchers will be able to find relevant articles to use in their research on adversarial machine learning in text processing.

REFERENCES

[1] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, "HiDDeN: Hiding data with deep networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 657–672.

[2] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[4] S. Baki, R. Verma, A. Mukherjee, and O. Gnawali, "Scaling and effectiveness of email masquerade attacks: Exploiting natural language generation," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 469–482.

[5] Y. Yao, B. Viswanath, J. Cryan, H. Zheng, and B. Y. Zhao, "Automated crowdturfing attacks and defenses in online review systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1143–1158.

[6] Z. Hu, H. Shi, B. Tan, W. Wang, Z. Yang, T. Zhao, J. He, L. Qin, D. Wang, X. Ma, Z. Liu, X. Liang, W. Zhu, D. S. Sachan, and E. P. Xing, "Texar: A modularized, versatile, and extensible toolkit for text generation," 2018, *arXiv:1809.00794*.

[7] W. Nie, N. Narodytska, and A. Patel, "RelGAN: Relational generative adversarial networks for text generation," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–20.

[8] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu, "Texygen: A benchmarking platform for text generation models," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 1097–1100.

[9] H. Zang and X. Wan, "Towards automatic generation of product reviews from aspect-sentiment scores," in *Proc. 10th Int. Conf. Natural Lang. Gener.*, 2017, pp. 168–177.

[10] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.

[11] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[12] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," 2017, *arXiv:1705.11001*.

[13] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.

[14] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4601–4609.

[15] H. Yin, D. Li, X. Li, and P. Li, "Meta-CoTGAN: A meta cooperative training paradigm for improving adversarial text generation," in *Proc. AAAI*, 2020, pp. 9466–9473.
[16] M. Federico, M. Cettolo, F. Brugnara, and G. Antoniol, "Language modelling for efficient beam-search," *Comput. Speech Lang.*, vol. 9, no. 4, pp. 353–380, 1995.
[17] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proc. AISTATS*, vol. 5, 2005, pp. 246–252.
[18] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. 40th Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2001, pp. 311–318.
[19] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," 2015, *arXiv:1511.06732*.
[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.
[21] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process.*, vol. 35, no. 1, pp. 53–65, Jan. 2017.
[22] K. Pearson, "Asymmetrical frequency curves," *Nature*, vol. 48, no. 1252, pp. 615–616, Oct. 1893.
[23] S. Ravuri, S. Mohamed, M. Rosca, and O. Vinyals, "Learning implicit generative models with the method of learned moments," 2018, *arXiv:1806.11006*.
[24] L. P. Hansen, "Large sample properties of generalized method of moments estimators," *Econometrica, J. Econ. Soc.*, vol. 50, no. 4, pp. 1029–1054, 1982.
[25] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.
[26] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," 2016, *arXiv:1606.03498*.
[27] Y. Mroueh and T. Sercu, "Fisher GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2513–2523.
[28] G. Lewis and V. Syrgkanis, "Adversarial generalized method of moments," 2018, *arXiv:1803.07164*.
[29] A. Bennett and N. Kallus, "The variational method of moments," 2020, *arXiv:2012.09422*.
[30] J. Na, J. H. Bak, and N. V. Sahinidis, "Efficient Bayesian inference using adversarial machine learning and low-complexity surrogate models," *Comput. Chem. Eng.*, vol. 151, Aug. 2021, Art. no. 107322.
[31] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 1960.
[32] J. A. Fodor and Z. W. Pylyshyn, "Connectionism and cognitive architecture: A critical analysis," *Cognition*, vol. 28, nos. 1–2, pp. 3–71, Mar. 1988.
[33] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.
[34] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," 2017, *arXiv:1711.01991*.
[35] L. Chen, S. Dai, C. Tao, H. Zhang, Z. Gan, D. Shen, Y. Zhang, G. Wang, R. Zhang, and L. Carin, "Adversarial text generation via feature-mover's distance," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4666–4677.
[36] P. Yu, R. Zhang, Y. Zhao, Y. Zhang, C. Li, and C. Chen, "SDA: Improving text generation with self data augmentation," 2021, *arXiv:2101.03236*.
[37] R. Zhang, "Towards uncertainty and efficiency in reinforcement learning," Ph.D. dissertation, Dept. Comput. Sci., Duke Univ., Durham, NC, USA, 2021.
[38] A. M. Donati, G. Quispe, C. Ollion, S. Le Corff, F. Strub, and O. Pietquin, "Learning natural language generation from scratch," 2021, *arXiv:2109.09371*.
[39] S. Rajeswar, S. Subramanian, F. Dutil, C. Pal, and A. Courville, "Adversarial generation of natural language," 2017, *arXiv:1705.10929*.
[40] T. Che, Y. Li, R. Zhang, R. D. Hjelm, W. Li, Y. Song, and Y. Bengio, "Maximum-likelihood augmented discrete generative adversarial networks," 2017, *arXiv:1702.07983*.
[41] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.
[42] H. Guo, "Generating text with deep reinforcement learning," 2015, *arXiv:1510.09202*.
[43] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," 2016, *arXiv:1607.07086*.
[44] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," 2016, *arXiv:1611.00712*.
[45] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 4006–4015.
[46] M. J. Kusner and J. M. Hernández-Lobato, "GANS for sequences of discrete elements with the Gumbel-softmax distribution," 2016, *arXiv:1611.04051*.
[47] L. Chen, "Learning deep models via optimal transport distance," Ph.D. dissertation, Dept. Electron. Comput. Eng., Duke Univ., Durham, NC, USA, 2021.
[48] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2016, *arXiv:1611.01144*.
[49] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Accelerating adversarial training via maximal principle," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 227–238.
[50] L. Chen, Y. Zhang, R. Zhang, C. Tao, Z. Gan, H. Zhang, B. Li, D. Shen, C. Chen, and L. Carin, "Improving Sequence-to-Sequence learning via optimal transport," 2019, *arXiv:1901.06283*.
[51] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "FeUdal networks for hierarchical reinforcement learning," 2017, *arXiv:1703.01161*.
[52] Y.-L. Tuan and H.-Y. Lee, "Improving conditional sequence generative adversarial networks by stepwise evaluation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 4, pp. 788–798, Apr. 2019.
[53] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 16–17.
[54] G. H. de Rosa and J. P. Papa, "A survey on text generation using generative adversarial networks," *Pattern Recognit.*, vol. 119, Nov. 2021, Art. no. 108098.
[55] M. A. Elaziz, A. Dahou, L. Abualigah, L. Yu, M. Alshinwan, A. M. Khasawneh, and S. Lu, "Advanced metaheuristic optimization techniques in applications of deep neural networks: A review," *Neural Comput. Appl.*, vol. 33, no. 21, pp. 14079–14099, 2021.
[56] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1171–1179.
[57] O. Vinyals and Q. Le, "A neural conversational model," 2015, *arXiv:1506.05869*.
[58] S. Wiseman, S. M. Shieber, and A. M. Rush, "Challenges in data-to-document generation," 2017, *arXiv:1707.08052*.
[59] R. Bhowmik and G. de Melo, "Generating fine-grained open vocabulary entity type descriptions," 2018, *arXiv:1805.10564*.
[60] R. Puduppully, L. Dong, and M. Lapata, "Data-to-text generation with content selection and planning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 6908–6915.
[61] Y. Li, Q. Pan, S. Wang, T. Yang, and E. Cambria, "A generative model for category text generation," *Inf. Sci.*, vol. 450, pp. 301–315, Jun. 2018.
[62] A. Graves, "Generating sequences with recurrent neural networks," 2013, *arXiv:1308.0850*.
[63] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang, and A. L. Yuille, "Single-shot object detection with enriched semantics," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5813–5821.
[64] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," 2017, *arXiv:1703.00955*.
[65] M. Schmitt, S. Sharifzadeh, V. Tresp, and H. Schütze, "An unsupervised joint system for text generation from knowledge graphs and semantic parsing," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 7117–7130.
[66] Y.-S. Wang, Y.-N. Chen, and H.-Y. Lee, "TopicGAN: Unsupervised text generation from explainable latent topics," in *Proc. ICLR*, 2018, pp. 1–10.
[67] M. Schmitt, S. Sharifzadeh, V. Tresp, and H. Schütze, "An unsupervised joint system for text generation from knowledge graphs and semantic parsing," 2019, *arXiv:1904.09447*.
[68] O. Sheffer, O. Castel, and R. Landau. (2020). *Going GREAN: A Novel Framework and Evaluation Metric for the Graph-to-Text Generation Task.* Accessed: Jul. 1, 2021. [Online]. Available: http://www.cs.tau.ac.il/ joberant/teaching/nlp_fall_2019_2020/past_projects/grean.pdf

[69] R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi, "Text generation from knowledge graphs with graph transformers," 2019, *arXiv:1904.02342*.
[70] Z. Jin, Q. Guo, X. Qiu, and Z. Zhang, "GenWiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 2398–2409.
[71] C. Kiddon, L. Zettlemoyer, and Y. Choi, "Globally coherent text generation with neural checklist models," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 329–339.
[72] O. Abdelwahab and A. Elmaghraby, "Deep learning based vs. Markov chain based text generation for cross domain adaptation for sentiment classification," in *Proc. IEEE Int. Conf. Inf. Reuse Integr. (IRI)*, Jul. 2018, pp. 252–255.
[73] S. Lu, Y. Zhu, W. Zhang, J. Wang, and Y. Yu, "Neural text generation: Past, present and beyond," 2018, *arXiv:1803.07133*.
[74] J. Wang and H. Zhang, "Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6629–6638.
[75] S. Mangal, P. Joshi, and R. Modak, "LSTM vs. GRU vs. bidirectional RNN for script generation," 2019, *arXiv:1908.04332*.
[76] M. M. Moila and T. I. Modipa, "The development of a Sepedi text generation model using long-short term memory," in *Proc. 2nd Int. Conf. Intell. Innov. Comput. Appl.*, Sep. 2020, pp. 1–5.
[77] J. Pouget-Abadie, D. Bahdanau, B. van Merrienboer, K. Cho, and Y. Bengio, "Overcoming the curse of sentence length for neural machine translation using automatic segmentation," 2014, *arXiv:1409.1257*.
[78] H. Mei, M. Bansal, and M. R. Walter, "What to talk about and how? Selective generation using LSTMs with coarse-to-fine alignment," 2015, *arXiv:1509.00838*.
[79] W. Fedus, I. Goodfellow, and A. M. Dai, "MaskGAN: Better text generation via filling in the_," 2018, *arXiv:1801.07736*.
[80] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proc. ICML*, 2011, pp. 1–8.
[81] E. Reiter, "NLG vs. templates," 1995, *arXiv:cmp-lg/9504013*.
[82] K. V. Deemter, M. Theune, and E. Krahmer, "Real versus template-based natural language generation: A false opposition?" *Comput. Linguistics*, vol. 31, no. 1, pp. 15–24, Mar. 2005.
[83] S. Wiseman, S. M. Shieber, and A. M. Rush, "Learning neural templates for text generation," 2018, *arXiv:1808.10122*.
[84] H. Peng, A. P. Parikh, M. Faruqui, B. Dhingra, and D. Das, "Text generation with exemplar-based adaptive decoding," 2019, *arXiv:1904.04428*.
[85] E. M. De Novais, T. D. Tadeu, and I. Paraboni, "Improved text generation using N-gram statistics," in *Proc. Ibero-Amer. Conf. Artif. Intell.* Berlin, Germany: Springer, 2010, pp. 316–325.
[86] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 27, 2014, pp. 3104–3112.
[87] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.
[88] R. Zheng, M. Ma, B. Zheng, and L. Huang, "Speculative beam search for simultaneous translation," 2019, *arXiv:1909.05421*.
[89] W. Yu, C. Zhu, Z. Li, Z. Hu, Q. Wang, H. Ji, and M. Jiang, "A survey of knowledge-enhanced text generation," 2020, *arXiv:2010.04389*.
[90] W. A. Woods, "Transition network grammars for natural language analysis," *Commun. ACM*, vol. 13, no. 10, pp. 591–606, Oct. 1970.
[91] B. M. Bataineh and E. A. Bataineh, "An efficient recursive transition network parser for Arabic language," in *Proc. World Congr. Eng.*, vol. 2, 2009, pp. 1–3.
[92] A. C. Bulhak, "On the simulation of postmodernism and mental debility using recursive transition networks," Monash Univ. Dept. Comput. Sci., Clayton, VIC, Australia, Tech. Rep. CS 96/264, 1996.
[93] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[94] A. Santoro, R. Faulkner, D. Raposo, J. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. Lillicrap, "Relational recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7299–7310.
[95] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," 2016, *arXiv:1602.02410*.
[96] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson, "One billion word benchmark for measuring progress in statistical language modeling," 2013, *arXiv:1312.3005*.
[97] T. Linzen, E. Dupoux, and Y. Goldberg, "Assessing the ability of LSTMs to learn syntax-sensitive dependencies," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 521–535, Dec. 2016.
[98] C. Garbacea, S. Carton, S. Yan, and Q. Mei, "Judge the judges: A large-scale evaluation study of neural language models for online review generation," 2019, *arXiv:1901.00398*.
[99] T. Mihaylova and A. F. T. Martins, "Scheduled sampling for transformers," 2019, *arXiv:1906.07651*.
[100] R. Zheng, Y. Yang, M. Ma, and L. Huang, "Ensemble sequence level training for multimodal MT: OSU-Baidu WMT18 multimodal machine translation system report," 2018, *arXiv:1808.10592*.
[101] G. Tevet, G. Habib, V. Shwartz, and J. Berant, "Evaluating text GANs as language models," 2018, *arXiv:1810.12686*.
[102] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.
[103] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.
[104] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–22.
[105] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," 2019, *arXiv:1904.08444*.
[106] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9185–9193.
[107] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2016, *arXiv:1608.04644*.
[108] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," 2018, *arXiv:1810.00069*.
[109] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, *arXiv:1611.01236*.
[110] Y. Yang, G. Zhang, D. Katabi, and Z. Xu, "ME-Net: Towards effective adversarial robustness with matrix estimation," 2019, *arXiv:1905.11971*.
[111] C. Xie, Y. Wu, L. V. D. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 501–509.
[112] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1765–1773.
[113] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," 2018, *arXiv:1803.01442*.
[114] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," 2018, *arXiv:1803.06373*.
[115] H. Qian and M. N. Wegman, "L_2-nonexpansive neural networks," 2018, *arXiv:1802.07896*.
[116] J. Hoffman, D. A. Roberts, and S. Yaida, "Robust learning with Jacobian regularization," 2019, *arXiv:1908.02729*.
[117] A. Jalal, A. Ilyas, C. Daskalakis, and A. G. Dimakis, "The robust manifold defense: Adversarial training using generative models," 2017, *arXiv:1712.09196*.
[118] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Mar. 2016, pp. 372–387.
[119] J. Zhang and C. Li, "Adversarial examples: Opportunities and challenges," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2578–2593, Jul. 2020.
[120] T.-J. Chang, Y. He, and P. Li, "Efficient two-step adversarial defense for deep neural networks," 2018, *arXiv:1810.03739*.
[121] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," 2020, *arXiv:2001.03994*.
[122] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3358–3369.
[123] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Mądry, "Adversarially robust generalization requires more data," 2018, *arXiv:1804.11285*.
[124] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[125] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.

[126] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*.

[127] M. Mosbach, M. Andriushchenko, T. Trost, M. Hein, and D. Klakow, "Logit pairing methods can fool gradient-based attacks," 2018, *arXiv:1810.12042*.

[128] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.

[129] W. Duch and J. Korczak, "Optimization and global minimization methods suitable for neural networks," *Neural Comput. Surveys*, vol. 2, pp. 163–212, Dec. 1998.

[130] J. Li, W. Monroe, and D. Jurafsky, "Understanding neural networks through representation erasure," 2016, *arXiv:1612.08220*.

[131] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[132] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," 2016, *arXiv:1612.01064*.

[133] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[134] Y. Xu, Y. Wang, A. Zhou, W. Lin, and H. Xiong, "Deep neural network compression with single and multiple level quantization," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.

[135] Q. Jin, L. Yang, and Z. Liao, "Towards efficient training for neural network quantization," 2019, *arXiv:1912.10207*.

[136] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval," in *Proc. 17th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 1994, pp. 232–241.

[137] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, vol. 37, 2015, pp. 957–966.

[138] P. Keukeleire, "Correspondence between perplexity scores and human evaluation of generated TV-show scripts," M.S. thesis, Dept. Comput. Sci. Eng., Delft Univ. Technol., Delft, The Netherlands, 2020.

[139] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6626–6637.

[140] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," 2015, *arXiv:1510.03055*.

[141] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Workshop Text Summarization AC*, Spain, Jul. 2004, pp. 74–81.

[142] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proc. ACL Workshop Intrinsic Extrinsic Eval. Measures Mach. Transl. Summarization*, 2005, pp. 65–72.

[143] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," 2017, *arXiv:1706.02633*.

[144] A. Koochali, P. Schichtel, A. Dengel, and S. Ahmed, "Probabilistic forecasting of sensory data with generative adversarial networks—ForGAN," *IEEE Access*, vol. 7, pp. 63868–63880, 2019.

[145] O. Cífka, A. Severyn, E. Alfonseca, and K. Filippova, "Eval all, trust a few, do wrong to none: Comparing sentence generation models," 2018, *arXiv:1804.07972*.

[146] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick, "Microsoft COCO captions: Data collection and evaluation server," 2015, *arXiv:1504.00325*.

[147] R. Staniūtė and D. Šešok, "A systematic literature review on image captioning," *Appl. Sci.*, vol. 9, no. 10, p. 2024, May 2019.

[148] Q. Qian, M. Huang, H. Zhao, J. Xu, and X. Zhu, "Assigning personality/profile to a chatting machine for coherent conversation generation," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 4279–4285.

[149] X. Zhang and M. Lapata, "Chinese poetry generation with recurrent neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 670–680.

[150] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD birds-200-2011 dataset," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS TR 2011 001, 2011.

[151] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[152] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electron*, vol. 8, no. 3, p. 292, Mar. 2019.

[153] C. Olah, "Understanding LSTM networks," Colah's Blog, Tech. Rep. 1, 2015.

[154] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *CoRR*, vol. abs/1708.02709, pp. 1–31, Aug. 2017.

[155] A. Sun, J. Wang, N. Cheng, H. Peng, Z. Zeng, L. Kong, and J. Xiao, "GraphPB: Graphical representations of prosody boundary in speech synthesis," 2020, *arXiv:2012.02626*.

[156] (2015). *Understanding LSTM Networks*. Accessed: Jun. 11, 2021. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[157] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of CNN and RNN for natural language processing," *CoRR*, vol. abs/1702.01923, pp. 1–7, Feb. 2017.

[158] R. Manzelli, V. Thakkar, A. Siahkamari, and B. Kulis, "An end to end model for automatic music generation: Combining deep raw and symbolic audio networks," in *Proc. Musical Metacreation Workshop 9th Int. Conf. Comput. Creativity*, 2018, pp. 1–6.

[159] Y. Zhou, X. Zheng, C.-J. Hsieh, K.-W. Chang, and X. Huang, "Defense against adversarial attacks in NLP via Dirichlet neighborhood ensemble," 2020, *arXiv:2006.11627*.

[160] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," 2016, *arXiv:1605.07725*.

[161] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," 2018, *arXiv:1805.02917*.

[162] C. Zhu, Y. Cheng, Z. Gan, S. Sun, T. Goldstein, and J. Liu, "FreeLB: Enhanced adversarial training for natural language understanding," 2019, *arXiv:1909.11764*.

[163] P.-S. Huang, R. Stanforth, J. Welbl, C. Dyer, D. Yogatama, S. Gowal, K. Dvijotham, and P. Kohli, "Achieving verified robustness to symbol substitutions via interval bound propagation," 2019, *arXiv:1909.01492*.

[164] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," 2017, *arXiv:1707.07328*.

[165] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[166] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," 2019, *arXiv:1902.02918*.

[167] X. Liu, M. Cheng, H. Zhang, and C.-J. Hsieh, "Towards robust neural networks via random self-ensemble," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 369–385.

[168] Y. Zhou, J.-Y. Jiang, K.-W. Chang, and W. Wang, "Learning to discriminate perturbations for blocking adversarial attacks in text classification," 2019, *arXiv:1909.03084*.

[169] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," 2018, *arXiv:1804.07998*.

[170] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," 2017, *arXiv:1704.08006*.

[171] C. Wong, "DANCin SEQ2SEQ: Fooling text classifiers with adversarial text example generation," 2017, *arXiv:1712.05419*.

[172] 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 *Proc. AAAI*, 2020, pp. 3601–3608.

[173] J. Ebrahimi, D. Lowd, and D. Dou, "On adversarial examples for character-level neural machine translation," 2018, *arXiv:1806.09030*.

[174] Y. Cheng, L. Jiang, and W. Macherey, "Robust neural machine translation with doubly adversarial inputs," 2019, *arXiv:1906.02443*.

[175] H. Zheng, Z. Zhang, H. Lee, and A. Prakash, "Understanding and diagnosing vulnerability under adversarial attacks," 2020, *arXiv:2007.08716*.

[176] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," 2017, *arXiv:1712.06751*.

[177] I. Alsmadi and M. J. O'Brien, "How many bots in Russian troll tweets?" *Inf. Process. Manage.*, vol. 57, no. 6, Nov. 2020, Art. no. 102303.

[178] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineerig*, vol. 6, no. 3, pp. 346–360, 2020.

[179] C. Coleman, D. Narayanan, D. Kang, T. Zhao, J. Zhang, L. Nardi, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, "DAWNBench: An end-to-end deep learning benchmark and competition," *Training*, vol. 100, no. 101, p. 102, 2017.

[180] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," 2017, *arXiv:1710.03740*.

[181] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 464–472.

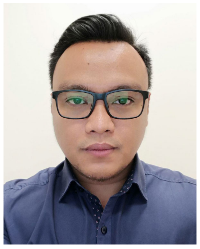
IZZAT ALSMADI received the master's and Ph.D. degrees in software engineering from North Dakota State University, in 2006 and 2008, respectively. He is currently an Associate Professor with the Department of Computing and Cyber Security, Texas A&M University, San Antonio. He has more than 100 conferences and journal publications. He is the Lead Author and an Editor of several books, including *The NICE Cyber Security Framework: Cyber Security Intelligence and Analytics* (Springer, 2019), *Practical Information Security: A Competency-Based Education Course* (Springer, 2018), and *Information Fusion for Cyber-Security Analytics*—Studies in Computational Intelligence (Springer, 2016). His research interests include cyber intelligence, cyber security, software security, software testing, social networks, and software defined networking.

NURA ALJAAFARI received the B.S. degree in computer science from King Faisal University (KFU), in 2015, and the M.S. degree in computer science from the King Abdullah University of Science and Technology (KAUST), in 2018. She is currently a Faculty Member with the College of Computer Science and Information Technology (CCSIT), KFU. Her research interests include deep learning, security and privacy in machine learning, and AI explainability.

MAHMOUD NAZZAL (Member, IEEE) received the B.Sc. degree in electrical engineering from Birzeit University, in 2009, and the M.Sc. and Ph.D. degrees in electrical and electronic engineering from Eastern Mediterranean University, in 2010 and 2015, respectively. He was a Lecturer with the Electrical and Electronic Engineering Department, Eastern Mediterranean University, from 2015 to 2016. He was also a Lecturer with the Izmir University of Economics, from 2016 to 2017. From July 2017 to August 2019, he was a Postdoctoral Researcher with Istanbul Medipol University. Then, he was a Lecturer with Abu Dhabi Vocational Education and Training Institute, United Arab Emirates. His research interests include sparse coding, security in machine learning, and signal processing for wireless communications.

SHADAN ALHAMED, photograph and biography not available at the time of publication.

AHMAD H. SAWALMEH (Member, IEEE) received the B.S. and M.S. degrees in computer engineering from the Jordan University of Science and Technology, Jordan, in 2003 and 2006, respectively, and the Ph.D. degree in computer and communication engineering from University Tenaga Nasional, Malaysia, in 2020. He is currently a Senior Lecturer with the Computer Science Department, Northern Border University, Saudi Arabia. His research interests include wireless communications and emerging technologies, with a focus on unmanned aerial vehicle (UAV) networks and flying *ad-hoc* networks (FANETs).



CONRADO P. VIZCARRA received the master's degree in information technology from the University of Cordilleras, in 2010. He is currently pursuing the Ph.D. degree in information technology with Saint Paul University Philippines. He is currently a Lecturer with the College of Computer Sciences and Information Technology, King Faisal University, where he has been a Faculty Member with the Computer Science Department, since 2014. As a Research Assistant, he has worked with researchers in a variety of computer science areas, including natural language processing on text generation and analysis problems and adversarial physical attack and defense in computer vision applications. Recently, he is appointed as the Supervisor at the Artificial Intelligence and Programming Club, CCSIT Department that offers training and workshop for CCSIT student related to the current trends in AI and programming. His research interests include information technology and system development. He is a member of various department and college level committee, such as Academic Advising, Quality Assurance, Graduation Project, Infrastructure Management, and Students Activity. He is also a member of the accreditation team that focuses on student services and policies at the institutional level.

ABDALLAH KHREISHAH (Senior Member, IEEE) received the B.S. degree in computer engineering from the Jordan University of Science and Technology, in 2004, and the M.S. and Ph.D. degrees in electrical and computer engineering from Purdue University, in 2006 and 2010, respectively. While pursuing his Ph.D. studies, he worked with NEESCOM. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology. His research interests include visible-light communication, green networking, network coding, wireless networks, and network security. He is the Chair of North Jersey IEEE EMBS Chapter.

MUHAMMAD ANAN (Senior Member, IEEE) received the B.S. degree in computer engineering from the King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, in 1997, the M.S. degree in electrical and computer engineering from the University of Missouri–Columbia, USA, in 1999, the M.S. degree in software engineering from Kansas University, USA, in 2003, and the Ph.D. degree in computer engineering and telecommunications networking from the University of Missouri–Kansas City, USA, in 2008. He is currently an Associate Professor of software engineering and the Acting Dean with the College of Engineering, Alfaisal University, Riyadh, Saudi Arabia. His research interests include computer networks, cloud computing, the Internet of Things, software defined networking, future internet architectures, wireless sensor networks, embedded systems, computer architecture, and software engineering.

ABDULELAH ALGOSAIBI (Member, IEEE) received the B.Sc. degree in computer science from the College of Computer Science and Information Technology, King Faisal University, in 2008, and the M.Sc. and Ph.D. degrees in computer science from Kent State University, USA, in 2011 and 2015, respectively. He was an Assistant Professor with the Computer Science Department, College of Computer Sciences and Information Technology (CCSIT), King Faisal University (KFU). He is currently the Chairperson of the Computer Science Department, College of Computer Science and Information Technology, King Faisal University. He has earned more than 1.8 millions SAR research Grant from the Deanship of Scientific Research, King Faisal University, and the Ministry of Education. He is currently working on advanced projects in natural language understanding and ontologies. His research interests include artificial intelligence, semantic web, and natural language processing.

MOHAMMED ABDULAZIZ AL-NAEEM (Member, IEEE) received the B.Sc. degree in computer and information science from the College of Management Science and Planning, King Faisal University, in 2005, the M.Sc. degree in networks and communications (specialization in information security) from Monash University, Australia, in 2009, and the Ph.D. degree in networks and communications (specialization in wireless networks and information security) from Monash University, in 2015. He is currently the Chairperson with the Department of Computer Networks and Communications, King Faisal University. His research interests include wireless networks, network security, machine learning, artificial intelligence, and pattern recognition.

ADEL ALDALBAHI (Member, IEEE) received the B.S. degree in electrical engineering from Virginia Commonwealth University, Richmond, VA, USA, in 2011, and the M.S. and Ph.D. degrees in electrical engineering from the New Jersey Institute of Technology, Newark, NJ, USA, in 2013. He is currently an Assistant Professor of electrical engineering, the Vice Dean of academic affairs, and the Director of the Industrial Relation and Technology Transfer Unit, King Faisal University, Al-Ahsa, Saudi Arabia. His current research interests include visible light communication, millimeter wave networks, and deep learning.

ABDULAZIZ AL-HUMAM received the B.Sc. degree in computer and information science from the College of Management Science and Planning, King Faisal University, in 2005, the M.Sc. degree in computer science from the Faculty of Informatics, Wollongong University, Australia, in 2009, and the Ph.D. degree in computer science from the University of York, U.K., in 2015. His research interests include computer science in general, technology innovation and software engineering, model-driven engineering, requirements engineering, self-adaptive software systems, software reuse and service-based systems, cloud computing, service-oriented architecture, requirements engineering for self-adaptive software systems, and assurance and certification of safety-critical systems engineering.