UNIVERSITY OF UDINE

Mathematics, Computer Science and Physics Department (DMIF)

PhD Course in
Artificial Intelligence and Data Science

PH.D. Thesis in

Generative Models: Image Synthesis, Keyphrase Extraction and Protein Structure Prediction

CANDIDATE	SUPERVISOR
Saida Saad Mohamed Mahmoud	Prof. Giuseppe Serra

Cycle XXXIV- A.Y (2021-2022)

Dedicated to my parents, my sister, my beautiful cat, my
lovely friends, Moon and the willow tree.

Acknowledgements

First and foremost, I would like to praise and thank God, the almighty, who has granted countless blessing, knowledge, and opportunity to me, so that I have been finally able to accomplish the thesis.
Apart from the efforts of me, the success of this thesis depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this thesis. I hardly knows where to start expressing the gratitude but for sure the gratitude goes to all those who have assisted me in the process of completing this Journey. It would be impossible to list all names but several people deserve my sincere and special thanks. I would like to express my sincere gratitude to my supervisor Professor.Giuseppe Serra for the continuous support of my Ph.D. study, for his patience, motivation, and immense knowledge. His guidance helped during the time of research and writing of this thesis. Special thanks to Professor. Federico Fogolari for introducing me to the field of Structural bioinformatics, his advice and guidance have been invaluable for my research. I feel motivated and encouraged every time I attend our group meetings. I would like to thank my family MAMA, BABA and my sister Rasha for their love and support, especially within my last year of graduate studies. Thanks for allowing me to take that opportunity. I am fortunate to be raised by them. Special thank for my Friend Mehanna for his support and words that change my life. I would like to thank my colleagues in Ailab laboratory group for all of their valuable discussions, advice and the interesting discussions that we have had. I would like to special thank Beatrice Portelli for being really good friend to me. She always support me during this journey. I really value all our conversations about many things and allowing me not to give up when the chips were down. She is such beautiful person. I would like to thank Giuseppe Lancioni, Giovanni D'Agostino, Alex Falcon for all the discussion. I would like to thank the daughter of my aunt "Mshaeel" for her support and good words too. She is very beautiful should that close to my soul. I thank my cat "Zaarda" that is really always give me good energy and vibes that help me to continue even when she got sick. I really love you. Special thanks to my Egyptian friends Dahlia, Sally, Nagwan, Marwa and Mariam and the friends that I build during the Journey Silvia, Cristina and Frenschca... Thanks to the PhD Community that choosed me to join that journey that changed my life and helped me to work in topics I like and in place that I fall in love with beautiful UDINE-Italy .

Abstract

Nowadays, Generative models are considered as potentially successful tools for various different tasks with different types of data. Many effective and interesting studies based on Generative models have been introduced, and have achieved good results in various applications even beyond computer vision. Generative models, which can model the probability distribution of the input data, have been intensively studied in numerous application. They have been used in text-guided content generation tools which, are useful and for artists, architects and designers. In addition, They used to improve information retrieval tools which are useful for text summarization and question generation. Moreover, generative models are rapidly gaining the interest of researchers in protein sequence and structure prediction. These models emerged as promising candidates for sequence-data-driven approaches to protein design, and for the extraction of structural and functional information about proteins deeply hidden in rapidly growing sequence databases. Protein structures provide a great level of understanding on how a protein works, and this, in turn, can allow us to create hypotheses about how to affect it, control it, or modify it. For example, knowing a protein's structure could allow you to design site-directed mutations with the intent of changing function. Or you could predict which kinds of molecules are able to bind to that specific protein. A lot of important tasks are tackled using generative models, and in particular Generative Adversarial Networks (GANs) and Transformer-based models (T5) with an increasing trend. In this thesis, I introduce three interesting tasks and explore the capabilities of different generative models that can be used to solve them, in order to better understand GAN-based and Transformer-based architectures of generative models for a better understanding of GANs based architectures and Transformer based architectures. I refer to the importance of generative models in three different domains: Image Synthesis, Keyphrases Extraction and Protein Structure Prediction. In the first task a GAN model is applied for Text to Image Synthesis on Machine Generated Captions. Text to Image Synthesis refers to the process of automatic generation of a photo-realistic image starting from a given text and is revolutionizing many real-world applications. The results of using GANs for creating realistic images from specific distribution given text description have raised many ethical issues along the way. However, there are some dataset which do not contain associated captions. In order to solve this problem, a GAN is trained on machine-generated captions. For the experiments, I choose to use the uncaptioned dataset LSUN bedroom. The results obtained in the study are preliminary but still promising. The second domain is applying generative models to the problem of Keyphrase Generation. Keyphrase Generation is the task of synthesizing keyphrases for a piece of text, that is short pieces of text that convey the main semantic meaning of a document. I introduce a keyphrase generation approach that

makes use of the GAN architecture. In this system, the Generator produces a sequence of keyphrases for an input document. The Discriminator, in turn, tries to distinguish between machine generated and human curated keyphrases. I propose a novel Discriminator architecture based on a BERT pretrained model fine-tuned for Sequence Classification. I trained our proposed architecture using only a small subset of the standard available training dataset, amounting to less than 1% of the total, achieving a great level of data efficiency. The resulting model is evaluated on five public datasets, obtaining competitive and promising results with respect to four state-of-the-art generative models.

The third domain regards the task of predicting the secondary structure of the proteins from their single sequence alignments. The most successful methods currently available, which employ multiple alignments of sequences, can predict the secondary structure of proteins with excellent performance. Predictions based on multiple alignments take advantage of all the amino acids found at a given position and their frequencies, reaching very high accuracy. However, the effect of a single amino acid substitution in a specific protein tends to be hidden by the alignment profile. Many studies have shown the potential of generative Transformer based architecture to increase the capacity of handling sequence data. Accordingly, to better understand the effect of single amino acid in that task, I analyze five different architecture for the task, incluing a model based on a generative Tranformer architecture (T5), discriminative deep learning models and linear models. The generative Tranformer-based architecture (T5) is an encoder decoder transformer pretrained in a text to text denoising generative setting. I compare the performance of the five models using their accuracy in predicting the secondary structure of a protein, but also use the DeepLIFT analysis to assess the effect of each amino acid at each position on the secondary structure prediction. The analysis shows how different network architectures use the information of single protein sequences and highlights their differences with respect to linear models.

Contents

List of Figures

List of Tables

List of Publication

- Marco Menardi, Alex Falcon, Saida S. Mohamed, Lorenzo Seidenari, Giuseppe Serra, Alberto Del Bimbo, Carlo Tasso: Text-to-Image Synthesis Based on Machine Generated Captions. IRCDL 2020 (pp. 62-74).
- Saida S. Mohamed, Gennaro Esposito, Giuseppe Serra, Federico Fogolari: Generalized Born radii computation using linear models and neural networks. Bioinformatics (2020) (pp.1757-1764)
- Giuseppe Lancioni, Saida S. Mohamed, Beatrice Portelli, Giuseppe Serra, Carlo Tasso: Keyphrase Generation with GANs in Low-Resources Scenarios. SustaiNLP@EMNLP 2020 (pp.89-96)
- Giuseppe Lancioni, Saida S. Mohamed, Beatrice Portelli, Giuseppe Serra, Carlo Tasso: Efficient Keyphrase Generation with GANs. IRCDL 2021 (pp.1-12)
- Saida S. Mohamed, Beatrice Portelli, Giovanni d'Agostino, Gianluca Pollastri, Giuseppe Serra, Federico Fogolari: A Comparison of Mutual Information, Linear Models and Deep Learning Networks for Protein Secondary Structure Prediction. Current Bioinformatics (submitted).

Chapter 1

Introduction

Deep learning techniques have achieved impressive accuracy and improved the state-of-the-art in many fields, like speech recognition, visual object recognition, object detection, drug discovery and genomics [66]. Deep learning models consist of a network of multiple processing layers. Data flows in the network to discover intricate structures in large datasets, while backpropagation is used to indicate how the model should change its internal parameters to improve the task for which it has been designed [66]. Deep learning algorithms had the most success in discriminative models, which are a class of models used for statistical classification, i.e. identifying which one of a set of categories a new observation belongs to. Deep learning is a series of machine learning methods based on special forms of neural networks that can conduct both feature extraction and classification in unison and with little human effort. It aims at teaching machines to automate the tasks performed by human sensory systems. One of the most powerful and distinctive aspects of human cognition is the ability to imagine (synthesize) mental objects which are not limited by what is immediately present in reality. The area of machine Learning which aims to endow machines with this same essential capacity to imagine and synthesize new entities is referred to as generative modeling. There are many practical motivations for generative models. One common argument is that humans may use generative models as a way of doing supervised and reinforcement learning while using less labeled data. Generative modeling is the use of artificial intelligence, statistics and probability in applications to produce a representation or abstraction of observed phenomena or target variables that can be calculated from observations. Generative modeling is used in unsupervised machine learning as a means to describe phenomena in data, enabling computers to understand the real world. This artificial intelligent understanding can be used to predict all manner of probabilities on a subject from modeled data. Generative modeling can automatically discover and learn the regularities or patterns in input data in such a way that the model can be used to generate or output new examples that plausibly could have been drawn from the original dataset where is the quality of the generated examples is so close to the original ones and difficult to distinguish. Generative models have achieved tremendous success in just a few years. A Generative model has a powerful way of learning any type of data distribution using unsupervised learning. This means that generative models aim at learning the true data distributions p_{data} of the training set to generate new data points with some variations. But it is not always possible to learn the exact distribution of the data

under analysis either implicitly or explicitly and so we try to model a distribution that is as similar as possible to p_{data}. It means that we can leverage the power of neural networks to learn a function that can approximate the model distribution to the true distribution. One of the strengths of generative models that is used to improve the generalization of deep learning models in the training. To train effective neural networks large amounts of data are needed. In the low-data regime, parameters are underdetermined, and the network generalizes poorly. Data Augmentation alleviates this by using existing data more effectively. However standard data augmentation produces only limited plausible alternative data. Using generative models, generated samples provide a greater diversity of augmentations that allow adversarial training to perform well using an inexpensive approach.
Generative models have been used in various tasks such as information retrieval [4], text classification [100], speech generation [123].
Not only generative Networks are capable of creating entirely new data (such as images, text, or speech), but also they do so in a way where the user can have more control over the type of data that will be generated. While generative models are quite capable of handling different types of problems and achieving the necessary solutions, there are numerous variations of these generative models to accomplish more specific tasks in order to achieve the best possible results. It is a way to get complete control over the type of data that we aim to generate with these simple networks, which might sometimes be required if you train the data on a larger dataset but need to generate a specific type of data (or image) that you are looking for. Those types of networks are called conditional generative models. Conditional generative architectures help to solve various tasks in many applications. For example, in the application of image-to-image translation, we make use of an input image to map it to its respective output image. With the help of a conditional model, we can specify the type of input image that we want as the particular condition and train the output images accordingly with respect to the input image. Given the particular input condition, one or more variations of the respective output can be generated. In addition, the application of the text to image synthesis is similar to the one we worked on in the second chapter of this thesis. Given a text description, the generative network of CGAN can generate images for the particular description [85]. Those conditional generative architectures are extremely useful for some significant applications that help artists and designers to solve various problems like super-resolution, colorization, and artistic style transfer in various tasks. For example,tile art generation [78] where images are made by assembling small pieces of actual tiles. We can also mention the task of generating aesthetically pleasing photography, sometimes termed photographic fine art [86] and the task of painting generation where generative model can be conditioned on different attributes of paintings in order to generate novel paintings that have specified attributes of our choosing [85].
There are many types of generative models that are commonly used as efficient approaches. We are going to focus on two types from those types: generative Adversarial Networks (GAN) and Transformer blocks. Transformer blocks are different types of building blocks that are widely used in neural networks designed for generative problem. In this chapter, I will provide a brief overview of several popular generative models. Based on this, I will then discuss the contributions that have made in this field during my Ph.D. study. This includes the generative models that I introduce for text-to-image translation, keyphrase generation, and protein structure prediction.

1.1 GAN: Generative adversarial networks

Generative adversarial networks(GAN) [32] is a framework for learning generative models through the use of an adversarial process. a GAN works to generate new data which can not be distinguished from real data that is a different learning process from Autoencoders(AEs) which learns to encode the given input (say, an image) and then reconstructs it from the encoding. AEs is a type of artificial neural network used to learn data encoding in an unsupervised manner [62]. The aim of AEs is to learn a lower-dimensional representation (encoding) for higher-dimensional data, typically for dimensionality reduction, by training the network to capture the most important parts of the input image. AEs aim at learning to reconstruct its input by minimizing the loss between original data and compressed data,

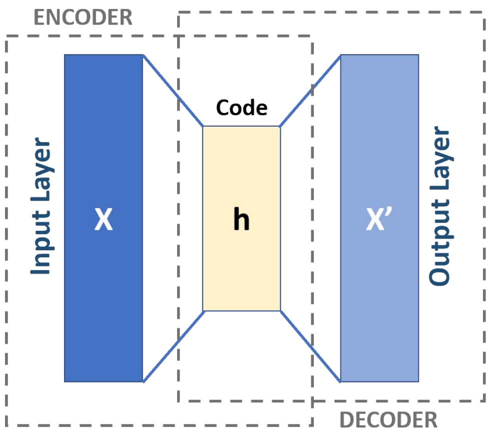


Figure 1.1: Autoencoder Schema Source

To do so, the AEs use two main components: an encode that compresses the input, and a decoder that tries to reconstruct it as shown in Figure 1.1. While GAN consists of two neural networks called the generator (generative model) denoted by G and the discriminator (discriminative model) denoted by D. The adversarial process comprises the training of both the generator G and the discriminator D simultaneously.
The generator G is trained to learn the true data distribution p_{data}. The discriminator D, on the other hand, acts as the adversary of the generator and is trained to distinguish between the real data samples and the samples generated by the generator G. While the discriminator has access to both the real and the fake data, the generator does not have access to the real data, thus it can learn only from the interaction with the discriminator. The error is calculated based on whether the discriminator can distinguish successfully the reals from the fakes. The calculated error is then used to train both the discriminator and the generator. The competition between the generator and the discriminator leads to an improvement to their methods until the fake data are indistinguishable from the reals. Figure 1.2 provides a high-level overview GAN. Consider p_g denote the true distribution over an observed data sample $x \in X$ and p_z denote the input noise's prior. Then, the generator $G(z; \theta_g)$ represents a mapping from the noise space to the data space

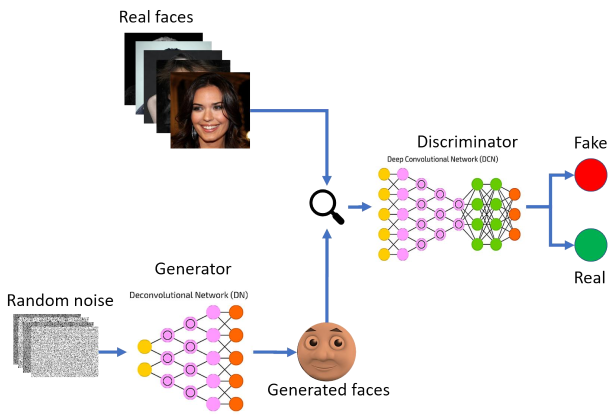


Figure 1.2: GAN ArchitectureSource

and is parameterized by θ_g. The discriminator $D(z;\theta_d)$is also parameterized by θ_d and produces a singular probability value. The higher the output probability the more likely it is that the input to the discriminator was sampled from $p_{data}(x)$. Alternatively, the discriminator D and the generator G play the following min-max game with the objective function $V(G, D)$:

$$min_G max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z)))] \tag{1.1}$$

GANs have become widely used for generative modeling tasks in both computer vision [3, 9] and Natural language Processing (NLP) [135, 148] domains with great success.

1.1.1 Conditional Generative Adversarial Network

On GAN, there is no possibility to control the data samples generated by the generator. Mirza et al [84] proposed a conditional generative adversarial network (CGAN). The proposed CGANs allowed the generated samples from the generator to be conditioned on some supplementary information about the input data items. Many methods use simple conditional variables such as attributes or class labels [18, 90, 129] to represent this information. Other works condition images for the tasks of photo editing [10, 154] domain transfer [119, 50], and super-resolution [67]. GAN can be extended to CGAN by feeding y which represents information about the data denoted by x to both the generator and the discriminator. So, the objective function for CGAN became the following equation:

$$min_G max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x|y)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - D(G(z|y)))] \tag{1.2}$$

1.2 Transformer

We will explain step by step the transformer approach. Since thet Transformer-based framework was built for machine translation task. We start by defining machine translation. **Machine Translation** A sequence-to-sequence task deals with a certain sequence (e.g., words, genes, etc) and its output is a sequence as well. An example of such a problem is a machine translation that gets a sequence of words in English that will be translated into a sequence of Italian words. Some other examples are questions answering, part-of-speech tagging, etc. There are a few ways to solve the translation problem e.g., RNNs, ConvS2S, and Transformers-based models. There are some attempts to solve the machine translation task using recurrent neural networks (RNN) based architectures that have shown the best performance but still they have some problems to cope with long range of dependencies that had to be solved. Because of the problem of their recurrent nature, It is difficult to parallelize each hidden state that depends on the previous one and make it efficient on GPUS. To solve this problem, Vaswani et al. [124] proposed an encoder-decoder model architecture based entirely on an attention mechanism called the Transformer [124]. The Transformer blocks utilize the self-attention mechanism extensively to model global dependencies between the input and output which can be described as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1.3}$$

where Q, K, V in the above function are interpreted as "query", "key" and "value" matrix. d_k is the dimension of K. Dot product attention was not applied directly to the positions. The multihead attention mechanism was also added to the transformer blocks. The Q, K, V matrix are projected into groups, and self-attention is applied within each group. Finally, the outputs of all heads are concatenated and projected. The purpose of the multi-head attention is to make Transformer blocks focus on different information in different subspaces. The encoder part of the model is bidirectional, and it takes the sequence to be translated as input. Also, the decoder uses Transformer blocks and masks over the attentions applied to the decoder to make sure that each position only attends to its prefix. So, in the decoder, the model is unidirectional and it outputs the translated sequence. Additionally, encoder-decoder attention is applied between the Transformer blocks in the decoder. This allows each position in the decoder to attend to the encoder's output. Figure 1.3 shows illustrations of the encoder's and decoder's Transformer blocks. Recently, Transformer-based models [26]started to outperform RNN-based methods in many tasks [74].
Transformers were used as a basis for modeling the language, which was possible by pretraining on large-scale datasets of textual data, and these pretrained models were shown to be quite powerful in zero-shot settings but also when finetuned on downstream tasks.

The Transformers architectures have been and continue to perform well in the range of NLP tasks. It considered as are the basis of NLP that aim at solving many problems like sequence-to-sequence tasks and handling long-range dependencies with ease. Transformer-based pretrained Language models were proven to be potentially useful where pretrained weights can serve as a regularization term to make the model have generalization performance [27]. Recently, Transformers for generative tasks [144, 15] in the hope that the increased expressivity can benefit the generation of complex images. Many tasks in computer vision are using transformers to synthesize pixels in an auto-regressive manner [7], but the slow inference speed limits their practical usage. In the following sections, we introduce some popular Transformers that were mainly useful to apply for the tasks along this thesis (eg. BERT and T5).

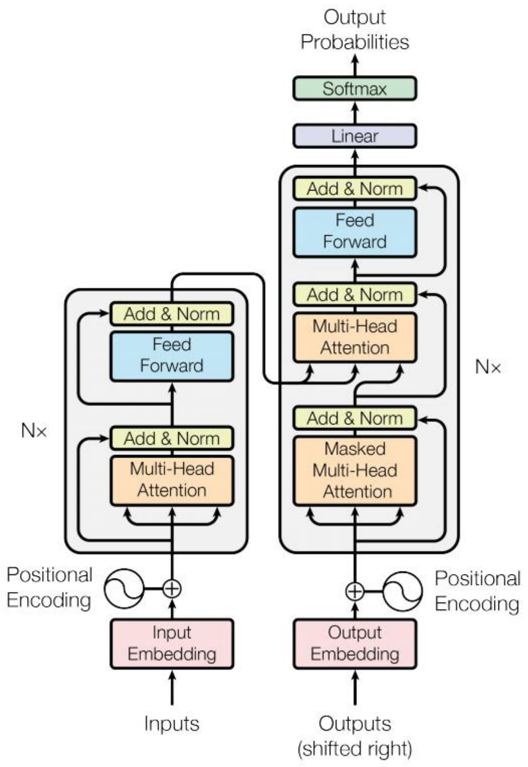


Figure 1.3: Transformer Framework: left half of the figure shows the encoder architecture and the right half shows the decoder architecture.

1.2.1 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT model was introduced by Jacob Devlin et.al [24]. Bert is designed to pretrained deep bidirectional representation from the unlabeled text by jointly working on both the left and right context in all layers. The model architecture of BERT is bi-directional and is almost identical to the encoder part described in Figure 1.3. BERT-base has twelve layers (Transformer blocks), each with twelve self-attention heads and 768 hidden neurons. The input sentences for BERT were tokenized into sub-words. A special token [CLS] was always placed in the head of the input sequence. There was a need for the special token [SEP], which was for marking the end of a sequence if there might be two sentences in the input. The token embedding of the current token, the position embedding of the current position, and the segment embedding of the current sentence were added up to form the input representation of one position as shown in Figure 1.4. BERT is pretrained on the English Wikipedia with $2,500$M and wordsBooksCorpus with 800M words. BERT is pre-trained simultaneously on two tasks [24]: masked language modeling (MLM) and next sentence prediction (NSP). The training loss is simply the sum of the mean MLM likelihood and the mean NSP likelihood. MLM is the task of predicting a percentage of input tokens that are randomly masked by a special token [MASK]. The MLM training objective was chosen over the traditional LM objective because of the bi-directionality of BERT. Hence, BERT is trained using masked language modeling. the prediction result is given by the five hidden vectors corresponding to the masked tokens, that are fed into an output softmax over the vocabulary, as in a standard LM. Then the next step is next sentence prediction (NSP) which is a binary classification task in which these receive pairs of sentences as input and learn to predict if the second sentence in the pair is the subsequent sentence in the original corpus. This training objective helps to understand the relationship between pairs of sentences, which is not directly captured by language modeling but is still very important for many downstream tasks such as question-answering (QA) and natural language inference (NLI).
As a result, the pretrained BERT model can be finetuned with one additional output layer to create as a result models for various tasks classification, language inference, and question answering.

Figure 1.4 is an illustration of how the input representations are computed from the processed tokenized sequences.

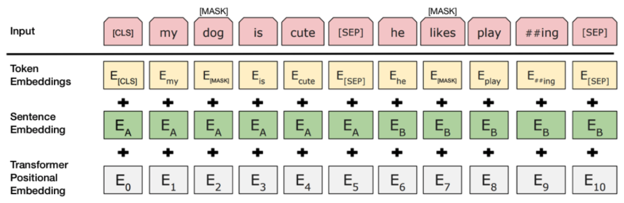

Input
[MASK]
[MASK]
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 1.4: BERT Framework [24]

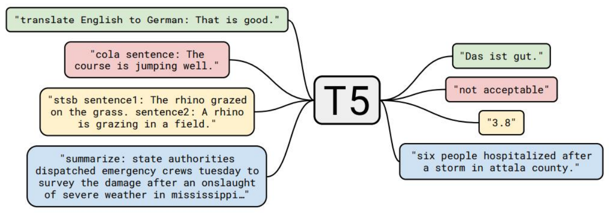

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 1.5: T5 Framework

1.2.2 T5: Text-to-Text Transfer Transformer

Text-to-Text Transfer Transformer (T5) is an encoder-decoder model pretrained on a multi-task mixture of unsupervised and supervised tasks, each of which is converted into a text-to-text format.

T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task. As mentioned before, the transformer model lends itself to effective parallelization of the model training process.

This opened up the possibility of training very large transformers-based models on massive NLP corpus using distributed computational resources.

T5 uses the same basic architecture as proposed in the original transformer paper [124] with some minor variations. It is based on the idea that most problems in NLP can be formulated as text-to-text transformation. In other words, given a sequence of words as input, the model produces other sequences of words as output. Figure [102] shows how the input and output are formulated for performing a variety of tasks using the T5 model. T5 choose to use a denoising pre-training objective as BERT. The pretrained objective was to replace spans of tokens with indexed placeholders and to make the decoder predict the replaced tokens. T5 is a sequence-to-sequence model, and it has an auto-regressive decoder. So, in nature, it coheres with the task of language generation. Some studies showed that pretrained T5 model and fine-tune it for some down-stream task such as question generation performed very well[34, 94].

1.3 Thesis Contributions

Generative models can generate various types of data: In this thesis, we will focus on different applications that cover different types of data text, images and biological data.

1.3.1 Generative Model for Text-to-Image Translation

There is a common problem for training supervised deep learning tasks, which is the need for large-scale datasets with labels /annotations; these annotations are costly, difficult to collect and/or annotate, tedious, error-prone, etc; therefore, unlabelled datasets are quite interesting because it is easier to collect data at scale; yet, many tasks, such as text-to-image synthesis, are difficult to solve without proper guiding/supervision; for example dataset of bedrooms exists and contains $3,000,000$ images[140], it does not contain the associated captions; therefore, techniques to automatically label the data are often studied and researched; in particular, to address this issue, in this paper we propose an approach capable of generating images starting from a given text using CGAN trained on uncaptioned images dataset. In particular, uncaptioned images are fed to an Image Captioning Module to generate the descriptions. Then, the GAN Module is trained on both the input image and the “machine-generated” caption. To evaluate the results, the performance of our solution is compared with the results obtained by the unconditional GAN. For the experiments, we chose to use the uncaptioned dataset LSUN-bedroom. The results obtained in our study are preliminary but still promising. This work was published in [79].

1.3.2 Generative Model for Keyphrase Generation

Keyphrase Generation is the task of predicting keyphrases (KPs), short phrases that summarize the semantic meaning of a given document. Several past studies provided diverse approaches to generate keyphrases for an input document. However, all of these approaches still need to be trained on very large datasets. We introduce BeGan- KP, a new conditional GAN model to address the problem of keyphrase generation in a low-resource scenario. Our main contribution relies in the discriminator’s architecture: a new BERT-based module which is able to distinguish between the generated and human-curated KPs reliably. Its characteristics allow us to use it in a low-resource scenario, where only a small amount of training data are available, obtaining an efficient generator. This work has been published in [64, 64]

1.3.3 Generative Model and different Neural network models for Protein Structure Prediction

We applied Generative model to the task of protein structure prediction from amino acid sequences. Our main contribution, we analyze how generative model and different network architectures for the task of protein structure prediction use the information of single protein sequences and highlights their differences with respect to linear models. In addition, We compare mutual information, the coefficients of two different implementations of a linear model, and two deep learning network architectures. We compare mutual information, the coefficients of two different implementations of a linear model and three deep learning network architectures. For the deep learning algorithms, we use the DeepLIFT analysis to assess the, effect of each amino acid at each position on the secondary structure prediction. Mutual information and linear models quantify direct effects, whereas DeepLIFT applied on deep learning networks is able to quantify both direct and indirect. Our main contribution, we analysis and compare generative and different disrimnative network architectures use the information of single protein sequences bu using DeeLIFT analysis method. This work has been submitted to Current Bioinformatics Journal.

In this domain, we also investigate the generalized born radii prediction task. Generalized Born (GB) models provide, for many applications, an accurate and computationally facile estimate of the electrostatic contribution to aqueous solvation. The computation of generalized born radii is still costly. In our main contribution, we propose an alternative approach using neural network that encodes the physics of the phenomena and the chemical structure of the molecules in model parameters which are learned from examples. We found that Neural networks further improve the accuracy of the predictions with correlation coefficient with "perfect" GB radii. This work had been publish in [75].

1.4 Thesis Outline

In my thesis, we applied different generative model architectures to different types of applications with different types of data (text, images, biological data). Each chapter represents an application that used different deep learning techniques based of generative models. In **Chapter 2**, we applied GANs on text-to-image synthesis application. We proposed an approach with the capable of generating images starting from a given text using conditional GANs trained on an uncaptioned images dataset. For the experiments, we use the uncaptioned dataset LSUN bedroom. In **Chapter 3**, we applied GANS based on Bert framework to keyphraeses generation. Keyphrase Generation is the task of predicting keyphrases: short text sequences that convey the main semantic meaning of a document. In this paper, we introduce a keyphrase generation approach that makes use of a generative adversarial networks (GANs) architecture. generative models. In **Chapter 4** We applied different architectures: generative model, discriminative models and linear models to predict the protein structure given its single sequence alignment. Also, we provide an analysis of the effect of a single amino acid on the structure prediction state. All the materials and datasets for the thesis's projects can be found here `https://github.com/SaidaSaad/My_thesis_Databases`

Chapter 2

Text-to-Image Synthesis Based on Machine Generated Captions using GANS

2.1 Introduction

Text-to-Image Synthesis, also called Conditional Image Generation, is a process that consists in generating a photo-realistic image given a textual description. It is a challenging task and it is revolutionizing many real-world applications. For example, starting from a Digital Library of adventure books it could be possible to enrich the reading experience with computer-generated images of the locations explored in the story, while a Digital Library of recipe books may be enriched with images representing the steps involved in a given recipe. In addition, such images may be used to exploit Information Retrieval systems based on visual similarity. Due to its great potentiality and usefulness, it raised a lot of interest in the research fields of Computer Vision, Natural Language Processing, and Digital Libraries. One of the main approaches used for the text-to-image task involves the use of Generative Adversarial Networks (GAN) [32]: starting from a given textual description, GANs can be conditioned on text [106], [105], [146] in order to generate high-quality images that are highly related to the text meaning. To condition a GAN on text, captioned images datasets are needed, meaning that one (or more) captions must be associated to each image. Despite the large amount of uncaptioned images datasets, the number of captioned datasets is limited. For example, LSUN [141] dataset, which consists in more than 59 million labeled images for each of 10 scene categories and 20 object categories [141]. The LSUN-bedroom dataset contains images from LSUN dataset tagged with the "bedroom" scene category. It contains around $\sim 3,000,000$ images [141], but it does not contain the associated captions. This may lead to a difficulty in training a conditional GAN to generate bedroom images related to a given textual description, such as "a bedroom with blue walls, white furniture and a large bed". In this paper we propose an innovative, though quite simple approach to address this issue as shown in Figure 2.1.

First of all, a captioning system (that we call Image Captioning Module) is trained on a generic captioned dataset and used to generate a caption for the uncaptioned images. Then, the conditional

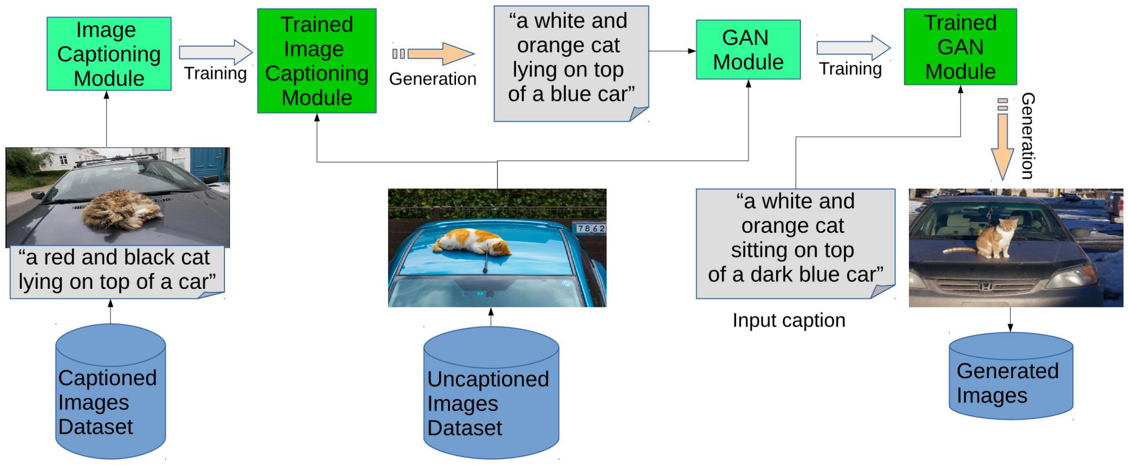


Image Captioning Module
Training
Trained Image Captioning Module
Generation
"a white and orange cat lying on top of a blue car"
GAN Module
Training
Trained GAN Module
Generation
"a red and black cat lying on top of a car"
"a white and orange cat sitting on top of a dark blue car"
Input caption
Captioned Images Dataset
Uncaptioned Images Dataset
Generated Images

Figure 2.1: Our pipeline: captioned images are used to train the Image Captioning Module; uncaptioned images are then captioned through the Trained Image Captioning Module and both the image and the generated captions are used to train the GAN Module; finally, the Trained GAN Module is used to generate an image based on an input caption.

GAN (that we call GAN Module) is trained on both the input image and the "machine-generated" caption. A high-level representation of the architecture is shown in Figure 2.2. To evaluate the results, the performance of the GAN using "machine-generated" captions are compared with the results obtained by the unconditional GAN. To test and evaluate our pipeline, we are using the LSUN-bedroom [141] dataset.

The results obtained in the experiments are very preliminary yet very promising. According to our study, the GAN Module does not learn how to produce meaningful images, with respect to the caption meaning, and we hypothesize that this is due to the "machine-generated" captions we use to condition the GAN Module. The Image Captioning module is trained on the COCO dataset [71], which contains captioned images for many different classes of objects and intuitively this should lead the Image Captioning Module to learn how to produce captions for bedroom images as well. Despite being able to produce the desired captions, we notice that the "machine-generated" captions are often too similar and not detailed for different bedroom images. The last section of the paper proposes some approaches that can deal with these problems.

2.2 Related Work

In 2014, Goodfellow et al. introduced Generative Adversarial Networks (GAN) [32], a generative model framework that consists in training simultaneously two models: a generator network and a discriminator one. The generator network has the task of generating images as real as possible, while the discriminator network has to distinguish the generated images from the real ones. Generative models are trained to implicitly capture the statistical distribution of training data; once trained, they can synthesize novel data samples, which can be used for example in the tasks of semantic

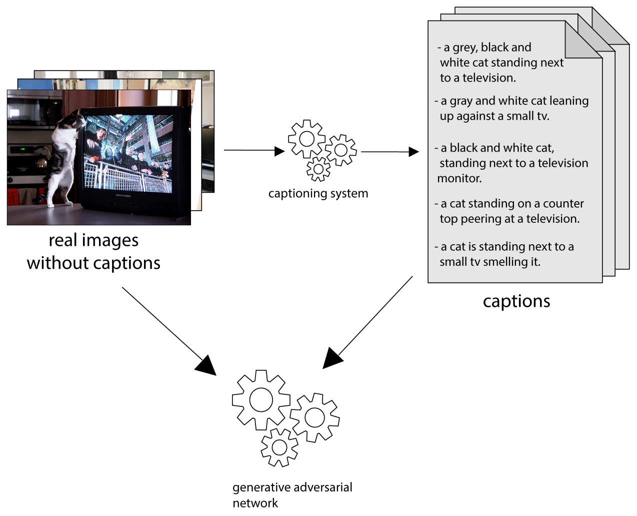

real images without captions
captioning system
- a grey, black and white cat standing next to a television.
- a gray and white cat leaning up against a small tv.
- a black and white cat, standing next to a television monitor.
- a cat standing on a counter top peering at a television.
- a cat is standing next to a small tv smelling it.
captions
generative adversarial network

Figure 2.2: Pipeline: images are fed to a captioning system that outputs its captions. The generated captions and the images are then given as input for training the conditional GAN.

image editing [155] and data augmentation [8]. GANs can be trained to sample from a given data distribution, in such case a random vector is provided as input to the generator. Otherwise, as in the case of text-to-image synthesis, they can be trained conditionally, meaning that an additional variable is provided as input to control the generator output. In certain formulations, the discriminator observes the conditioning variable too, during training. In the literature, several possibilities were tested for the variables used to condition a GAN: attributes or class labels (e.g. [19], [91]), images (e.g. for the tasks of photo editing [155] and domain transfer [51]).
Several methods have been developed to generate images conditioned on text. Mansimov et al. [76] built an AlignDRAW model trained to learn the correspondence between text and generated images. Reed et al. in [107] used PixelCNN to generate images using both text descriptions and object location constraints. Nguyen et al. [88] used an approximate Langevin sampling approach to generate images conditioned on text, but it required an inefficient iterative optimization process. In [106], Reed et al. successfully generated 64×64 images for birds and flowers conditioning on text descriptions. In their follow-up work [105], they were able to generate 128×128 images by using additional annotations on object part locations. Denton et al. in [23] proposed the Laplacian pyramid framework (LAPGANs), which is composed of a series of GANs. A residual image is conditioned at each level of the pyramid on the image of the previous stage to produce an image for the next stage. Also in [57], Kerras et al. use a similar approach by incrementally adding more layers in the generator and in the discriminator. [145] and [146] suggest the use of a so-called sketch-refinement process, where the images are first generated at low resolutions using a GAN conditioned over the textual description, and then refined with another GAN conditioned on both the image generated at the previous step and the input textual description. [45] and [68] infer a semantic label map by predicting bounding boxes and object shapes from the text, and then synthesize an image conditioned on the layout and the text description. A recent work by Qiao et al. [101] uses

a three-step approach where it first computes word- and sentence-level embedding from the given textual description, then it uses the embeddings to generate images in a cascaded architecture, and finally starting from the image generated at the previous step it tries to regenerate the original textual description, in order to semantically align with it. Although several different state-of-the-art architectures may be chosen for the task, such as HDGAN [151] and AttGAN [136], in our pipeline we decided to use StackGAN-v2 [146] as the conditional GAN component, given the availability of its code on GitHub.

Recently, several impressive results [152], [70], [108] were obtained for the Image Captioning (or image-to-text) task, which deals with the generation of a caption describing the given image and the objects taking part to it. It is an important task that raises a lot of interest in the Computer Vision and Natural Language Processing research fields. A recent and comprehensive survey about the task is provided by Hossain et al. in [46]. Some of the approaches used for this task involve the use of Encoder/Decoder networks and Reinforcement learning techniques.

The encoder/decoder paradigm for machine translation using recurrent neural networks (RNNs) [20] inspired [56], [125] to use a deep convolutional neural network to encode the input image, and a Long Short-Term Memory (LSTM) [44] RNN decoder to generate the output caption. Given the unavailability of labeled data, recent approaches to the image captioning task involve the use of reinforcement learning and unsupervised learning-based techniques. [152] and [70] use actor-critic reinforcement learning methods, where a "policy network" (the actor) is trained to predict the next word based on the current state, whereas a "value network" (the critic) is trained to estimate the reward of each generated word. These techniques overcome the need to sample from the policy (actors) action space, which can be enormous, at the expense of estimating future rewards. Another approach, used by Ranzato et al. in [103], consists in applying the REINFORCE algorithm [131]. A limitation of this algorithm consists in the requirement of a context-dependent normalization to tackle the high variance encountered when using mini-batches. The approach we are following uses Self-Critical Sequence Training (SCST) [108] which is a REINFORCE algorithm that utilizes the output of its own test-time inference algorithm to normalize the rewards it experiences: doing so, it does not need neither to estimate the reward signal nor the normalization.

2.3 Our Approach

We propose a pipeline whose goal is to generate images by conditioning on "machine-generated" captions. This is fundamental when image captions are not available for a specific domain of interest. Thus, the proposed solution involves the use of a generic captioned dataset, such as the COCO dataset, to make the Image Captioning Module capable of generating captions for a specific domain.

To do so, we want to explore the possibility of using an automatic system to generate textual captions for the images and use them for the training of a Generative Adversarial Network. For achieving our goal, we built a pipeline composed by an Image Captioning Module and a GAN Module, as shown in Figure 2.1. First of all, the Image Captioning Module is trained over a generic captioned dataset to generate multiple captions for the input image. Then, real images are given as input to the Trained Image Captioning Module, which outputs multiple captions for each image.

The generated captions together with the images are then fed to the GAN Module, which learns to generate images conditioned on the "machine-generated" captions. By feeding the GAN with multiple captions for each image, the GAN can better learn the correspondence between images and captions.

In the following sections, we detail the two modules used in our pipeline: the Image Captioning Module and the GAN Module.

2.3.1 Image Captioning Module

The goal of the Image Captioning Module is to generate a natural language description of an image. Good performance in this task is obtained by learning a model which is able to first understand the scene described in the image, the objects taking part to it and the relationships between them, and then to compose a natural language sentence describing the whole picture. Given the complexity of such a task, it is still an open challenge in the fields of Natural Language Processing and Computer Vision. The task of open domain captioning is a challenging task. It requires a fine-grained understanding of the whole entities, attributes and relationships in an image. In our pipeline, we are implementing our Image Captioning Module in a similar way as the one proposed in [108], meaning that we also use a captioning system based on FC models. It has been built using an optimization approach that is called Self-Critical Sequence Training (SCST).

Typical deep learning models used for the Image Captioning task are trained with the "teacher-forcing" technique, which consists in maximizing the likelihood of the next ground-truth word given the previous ground-truth word. This has been shown to generate some mismatches between the training and the inference phase, knows as "exposure bias". Moreover, the metrics used during the testing phase are non-differentiable (such as BLEU and CIDEr), meaning that the captioning model can not be trained to directly optimize them. To overcome these problems, Reinforcement Learning techniques such as the REINFORCE algorithm have been used. SCST is a variation and an improvement of the popular REINFORCE algorithm that, rather than estimating a baseline to normalize the rewards and reduce variance, utilizes the output of its own test-time inference algorithm to normalize the rewards it experiences. This means that it is forced to improve the performance of the model under the inference algorithm used at test time. Practically, SCST has much lower variance than REINFORCE and can be more effectively trained on mini-batches of samples using SGD. Moreover, it has been shown that SCST system has achieved state-of-the-art performance by optimizing their system using the test metrics of the MSCOCO task. Practically, it has been found that SCST has much lower variance, and can be more effectively trained on mini-batches of samples using SGD. Since the SCST baseline is based on the test-time estimate under the current model, SCST is forced to improve the performance of the model under the inference algorithm used at test time. In addition, this encourages training consistency like the maximum likelihood-based approaches except it optimized sequence metrics.

2.3.2 GAN Module

The GAN Module has the major role of learning to generate images by conditioning on the "machine-generated" captions. In particular, we are using StackGAN-v2 [146] as our GAN Module.

StackGAN-v2 consists of a multiple stage generation process, where high-resolution images are obtained by initially generating low-resolution images which are then refined in multiple steps. It consists in a single end-to-end network composed by multiple generators and discriminators in a tree-like structure. Different branches of the tree generate images of different resolutions: at branch i, the generator G_i learns the image distribution p_{G_i} at that scale, while the discriminator D_i estimates the probability of a sample being real. The framework of StackGAN-v2 has a tree-like structure, that takes as input the noise vector $z \sim p_{noise}$. The noise z is transformed in hidden feature layer by layer. The hidden features h_i for each generator G_i are calculated by a non-linear transformation

$$h_0 = F_0(z); \quad h_i = F_i(h_{i-1}, z), \tag{2.1}$$

where h_i represents hidden features for the i^{th} branch, m is the total number of branches, and F_i are modeled as neural networks. The noise vector z is concatenated to the hidden features h_{i-1} as the inputs of F_i for calculating h_i. The generators produce samples at different scales $(s_0, s1, ..., s_{m-1})$ based on the hidden features at different layers $(h_0, h_1, ..., h_{m-1})$.

$$s_i = G_i(h_i), \quad i = 0, 1, ..., m-1, \tag{2.2}$$

where G_i is the generator for the i^{th} branch. Since we are more interested in the conditional case, we are not reporting the loss function used by the generator and the discriminator in the unconditional setting, for which more details can be found in [146].
The discriminator D_i takes a real image x_i or a fake sample s_i as input and is trained to classify them as real or fake by minimizing the cross entropy loss:

$$\begin{aligned} \mathcal{L}_{D_i} = & \underbrace{-\mathbb{E}_{x_i \sim p_{data_i}}[logD_i(x_i)] - \mathbb{E}_{x_i \sim p_{G_i}}[log(1 - D_i(s_i))]}_{\text{unconditional loss}} \\ & \underbrace{-\mathbb{E}_{x_i \sim p_{data_i}}[logD_i(x_i, c)] - \mathbb{E}_{x_i \sim p_{G_i}}[log(1 - D_i(s_i, c))]}_{\text{conditional loss}} \end{aligned} \tag{2.3}$$

where x_i is an image from the true image distribution p_{data_i} at the i^{th} scale, s_i is from the model distribution p_{G_i} at the same scale. While StackGAN-v2 [146] follows the approach of Reed et al. [104] to pre-train a text encoder to extract visually-discriminative text embeddings of the given description, in our case we use Skip-Thought [60], that works at the sentence level, to generate the text embeddings (c in the equations 2.3 and 2.4). Sentences that share semantic and syntactic properties are mapped to corresponding vector representations [60]. The multiple discriminators are trained in parallel each one for a different scale, while the generator is instead optimized to jointly approximate multi-scale image distributions $p_{data_0}, p_{data_1}, ..., p_{data_{m-1}}$ by minimizing the following loss function:

$$\mathcal{L}_G = \sum_{i=1}^{m} \mathcal{L}_{G_i}, \quad \mathcal{L}_{G_i} = \underbrace{-\mathbb{E}_{s_i \sim p_{G_i}}[logD_i(s_i)]}_{\text{unconditional loss}} \underbrace{-\mathbb{E}_{s_i \sim p_{G_i}}[logD_i(s_i, c)]}_{\text{conditional loss}} \tag{2.4}$$

where L_{G_i} is the loss function for approximating the image distribution at the i^{th} scale. The unconditional loss is used to determine whether the image is real or fake, while the conditional loss is used to determine if the image and the condition match.

2.4 Experimental Results

In this section, we present the preliminary results of the experiments involving the proposed pipeline. The Image Captioning Module was trained on the COCO dataset [71], which contains 120,000 generic images tagged with categories and captioned by five different sentences each. The uncaptioned dataset that we considered is the LSUN [141] dataset, which consists in more than 59 million labeled images. From the LSUN dataset, we first select the $\sim 3,000,000$ images tagged with the "bedroom" scene category and from that set a subset of the first 120,000 images is selected: 80,000 are then used to train the GAN and 40,000 as test set. Later on in this paper, the selection of the $\sim 3,000,000$ images tagged with the "bedroom" scene category is called "LSUN-bedroom".

A typical metric used to evaluate both the quality and the diversity of generated images is the Inception Score [113]. Unfortunately, the type of image of the LSUN dataset is very different from those used by ImageNet [146, 22], therefore it has been shown that the Inception Score is not a good indicator for the quality of generated images [146]. So we decided not to report the obtained scores. We performed three experiments over the considered dataset. The first experiment consists in training the GAN Module on the whole LSUN-bedroom dataset ($\sim 3,000,000$ images). This is done because of two reasons: first, it serves as a baseline for the next experiment; second, we compare the results obtained by our computing facilities with the results obtained in [146], since with our graphics card we are limited to a lower batch size of 16. Figure 2.4 shows some examples of generated images, and it is possible to see that the quality of the generated images is similar to those shown in Figure 2.3 [146].

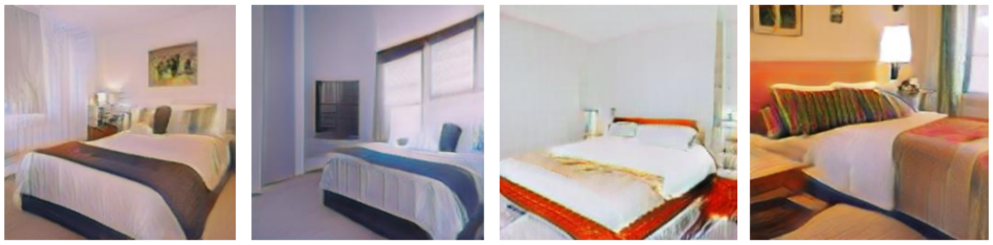


Figure 2.3: Examples of images generated by the StackGAN Module trained on the whole LSUN-bedroom dataset.

To reproduce the results reported in the paper, we used an NVIDIA GTX 1080 8GB machine. It took us around one month to train the GAN Module on the whole LSUN dataset. Because of this, we decided to explore and understand how the GAN Module performs with less training images. In the second experiment, the training of the GAN Module without conditioning is done on a subset of LSUN-bedroom, consisting of 120,000 images. Some of the results obtained in this experiment are showed in Figure 2.5. Although the quality of the generated images is slightly reduced, it is possible to see that the semantic content is still clear and defined.

Finally, to test our pipeline, we used the Image Captioning Module to generate captions for the images contained in the considered subset of the LSUN-bedroom dataset. Then, the GAN Module was trained on these same images and conditioned by the "machine-generated" captions. About

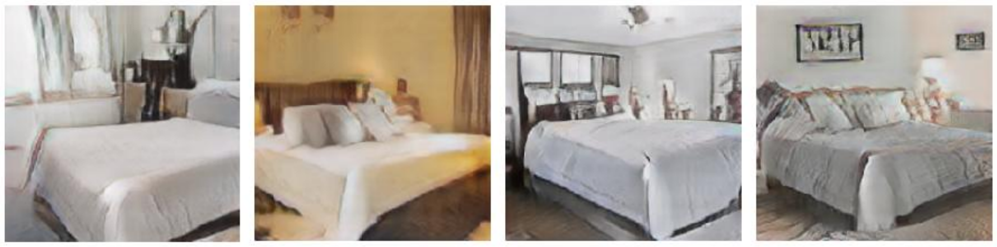


Figure 2.4: Examples of images generated by the GAN Module trained on the whole LSUN-bedroom dataset.

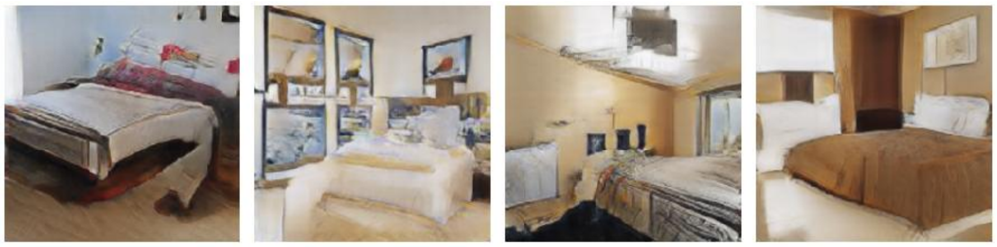
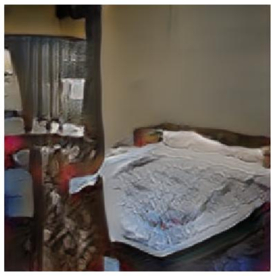


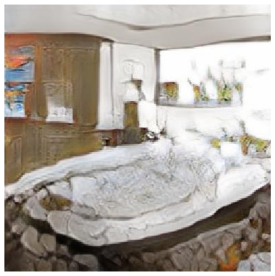
Figure 2.5: Examples of images generated by the GAN Module trained on a part of the LSUN-bedroom dataset.

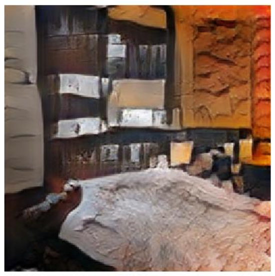
the preliminary results that we obtained, some examples are shown in Figure 2.6. We suspect the problem is due to the similarity of the “machine-generated” captions: the LSUN-bedroom dataset does not come with captions and thus the Image Captioning Module is trained on a generic dataset (COCO) and not for that specific dataset. Because of this, the Image Captioning Module is unable to produce detailed and varied captions for different bedroom images. In addition, usually GANs used noise vector to generate images which always different from each other [32]. In our experiment, the noise vector is taken as input by the model and used to generate an image. Then, the captions are used to yield the embeddings, which are also used as noise by the generator. The fact that the noise is almost always the same could be the cause of the observed problem. We found that the scores for the LSUN-bedroom dataset seem to not fully correlate with the quality of the generated images. As explained in [146], this may be due to the inception score being trained on the inception dataset, and thus it does not work well on datasets with specific types of images. Also, it has to be considered that different datasets get inception scores in different ranges. For this reason, inception scores must not be compared across different datasets.

2.5 Summary

In this chapter, we explored the problem of conditional image generation using Generative Adversarial Networks with machine-generated captions. For this task, we built a pipeline to first generate captions for uncaptioned datasets and then to use the machine-generated captions to condition a


(a) “this bedroom looks very old-fashioned”


(b) “the bedroom has carpeted floor and the walls are papered”


(c) “the room has rich colors and overall looks very modern”

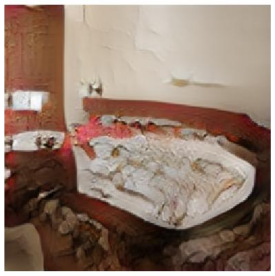

(d) “this small bedroom has matching wooden and glass furniture”

Figure 2.6: Examples of images generated by the GAN Module trained on a part of the LSUN-bedroom dataset and conditioned on “machine-generated” captions.

GAN. To test our pipeline, we run experiments on the LSUN-bedroom dataset, which is a subset of the LSUN dataset containing uncaptioned images of bedrooms, and then compare the generated images in the unconditional setting and in the conditional setting where "machine-generated" captions are used. The results observed in the experiments do not achieve success in the task of conditioning with "machine-generated" captions. So we identify, analyze, and propose possible solutions to the obstacles that need to be overcome.

The Image Captioning Module that we trained on the COCO dataset seems to generate captions too similar to each other. Moreover, the captions we generated lack details and contain some errors. This is probably related to the fact that more diverse and detailed captions are needed during training in order to achieve significant improvements. During a subsequent review of works on captioning, we found a work from Shetty et al. [115] that promises to generate more different captions, instead of variations of the same caption. This result is achieved by using GANs for image captioning instead of other traditional methods. An open question is whether with a bigger dataset the GAN could learn the image-captions correspondence, even when captions are very similar for each image. We believe improving the quality of the generated caption is the main challenge for our method. An hybrid approach could make our proposed method work by making humans write captions on a subset of the dataset, then use the obtained captions to train a captioning system. For generating human captions, crowdsourcing platforms like Amazon Mechanical Turk (AMT) could be used. We are currently working on this idea because it's likely that it will lead to improvements in the quality of generated bedroom images, given that AMT could make it possible to have high-quality and more diverse captions. Moreover, we are also considering the use of the Fréchet Inception distance [42] to evaluate the generated captions and images.

Chapter 3

Keyphrase Generation using Generative adversarial model based on Bert Framework

3.1 Introduction

A Keyphrase (KP) is a piece of text that conveys the main semantic meaning of a document. KPs can be either present (or extractive) or absent (or abstractive): present KPs are exact substrings of the document while absent KPs are not. Their automatic prediction is an important challenge for the community research as KPs are a key component for a wide range of applications such as text summarization [150], opinion mining [6], document clustering [37], information retrieval [53] and text categorization [48]. Historically, the first approaches focused on simply extracting substrings of the text to be used as keyphrases candidates [139, 73, 147]. Recently, the research community has focused on the broader task of Keyphrase Generation [80, 14, 16]. Keyphrase Generation aims to *produce* a set of phrases that summarize the essential information in a given text, as opposed to simply *look for them* in the text. This allows for greater flexibility. Several approaches introduced generative models based on the Encoder-Decoder architecture [80, 14]. This architecture works by compressing the contents of the input (e.g. the text document) into a hidden representation using an Encoder module. The same representation is then decompressed using the Decoder module, which returns the desired output (e.g. a sequence of KPs). The modules are trained jointly to learn the best intermediate representation to perform this mapping.

More recently, an approach based on GAN (Generative Adversarial Networks [33]) architecture has been proposed to address the task [118]. Although all these solutions achieved interesting results, they require a very large amount of data in order to be trained.

Our aim is to improve training efficiency, so that a model can be trained using only small subsets of the data. We focus our research in the generation of present KPs and we propose a new conditional GAN architecture for Keyphrase Generation that can be trained with a relatively small set of samples. The key component of our solution is the Discriminator: a model based on BERT that is able to distinguish between human and machine-generated Keyphrases leveraging on the language modelling information obtained from finetuning in a low-resource scenario.

A Reinforcement Learning (RL) strategy is then used to train the Generator, with rewards evaluated by the Discriminator. This encourages the model to generate more accurate and relevant KPs.

Thanks to the characteristics of our architecture, we are able to use only a small subset of the available data, using less than 1% of them to train our system. Compared to all the previous approaches that needed to be fully trained on large set of training samples, our architecture greatly reduces required resources, while still providing competitive results in the generation of present KPs.

3.2 Related Work

3.2.1 Keyphrase Extraction

Extractive methods aim at identifying Keyphrases in the span of the source text. Most of the algorithms in this field adopt a two steps pipeline to extract KPs. First, given a document, a list of candidates phrases is selected using heuristic methods [128, 65]. Secondly, all candidates are scored against the document. The first step has a considerable impact on the ability of the whole model to correctly identify all KPs, so selecting a sufficiently high number of candidates is of utmost importance. The second step can be done either in a supervised or unsupervised manner [82, 133, 89]. The top-scoring candidates are returned as KPs. Two interesting strategies that differ from the common pipeline approach have been proposed by [120] and [147]. The first method employs two statistical language-based models to extract Keyphrases. The latter introduces a model based on joint layer recurrent neural network to extract Keyphrases from tweets.

3.2.2 Keyphrase Generation

Recently, research has focused on the introduction of methods of text generation to predict Keyphrases. Most of these approaches rely on Encoder-Decoder framework in which the source text is first mapped to an encoded representation, and then decoded to the target text, that is the Keyphrases to predict.

[80] proposed CopyRNN, a RNN-based generative model for KP Generation, which is an Encoder-Decoder model with copy mechanism. [14] proposed CorrRNN model which is a sequence-to-sequence architecture for Keyphrase Generation that captures the correlations among Keyphrases. TG-Net model was introduced by [17] for improving automatic Keyphrase Generation using the information contained in the title of the document. [16] proposed an integrated approach for Keyphrase Generation which is a multitask learning framework that jointly learns an extractive model and a generative model.

Two recurrent generative based models, CatSeq and CatSeqD, were proposed by [143]. One of their main characteristics is the ability to determine the appropriate number of Keyphrases for each input document. CatSeq is based on an Encoder-Decoder mechanism, which is used to identify relevant components of the source text (abstracts) and generate KPs (sequence-to-concatenated sequences) [143, 12]. It employed the sequence-to-sequence framework combined with an attention mechanism and pointer softmax mechanisms in the Decoder. CatSeqD introduces the following techniques: orthogonal regularization, which prevents the model from predicting the same word after generating the constant KP separator; semantic coverage, which encodes again the decoded sequences and uses it as a representation of the target phrases. These representations are employed as further input during a self-supervised training phase with the aim of improve the semantic content of the predictions. [12] subsequently proposed a Reinforcement Learning approach with adaptive rewards to improve CatSeq, CatSeqD, CorrRNN and TG-Net generative models, leading to a new version for each of them. These versions are called, respectively, CatSeq-2RF1, CatSeqD-2RF1, CatSeqCorr-2RF1 and CatSeqTG-2RF1.

Recently, [118] proposed a GAN model conditioned on scientific articles for KP Generation. The author uses a CatSeq model to implement the Generator, conditioning it on abstracts of scientific articles. The Discriminator is based on a hierarchical attention mechanism consisting of two GRU layers. The two layers model the relationship between the document and each generated KP to assess whether the KP is synthetic or human in origin.

To the best of our knowledge, no attempts have been made of either extracting or generating KPs in a low-resources scenario, in which only a small amount of the available data samples is used during training. Our proposed architecture, based on a Discriminator that relies on a language model, requires less than 1% of the available training data to achieve good results.

3.3 The Proposed Approach

To generate KPs in a low-resource scenario we propose an approach based on the GAN Framework that we call BeGan-KP. It mainly consists of three components: (1) a conditioned Generator model that produces a set of KPs, (2) a novel BERT Discriminator model that checks if the KPs are fake (generated) or real (human-curated), and (3) the Reinforcement Learning (RL) module that is involved in the training process of the system as a whole (see Figure 3.1).

3.3.1 Notations and Problem Definition

The samples available to train the system are pairs $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ is a document and $\mathbf{y} = (\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^M)$ is the set of M Keyphrases (True KPs) associated to $\mathbf{x}$. Note that both $\mathbf{x}$ and $\mathbf{y}^i$ are sequences of words:

$$\mathbf{x} = x_1, x_2, \ldots, x_L$$
$$\mathbf{y}^i = y^i_1, y^i_2, \ldots, y^i_{K_i}$$

where L and K_i are the number of words of $\mathbf{x}$ and of its i-th KP respectively.

The Generator takes as input $\mathbf{x}$ and outputs $\hat{\mathbf{y}} = (\hat{\mathbf{y}}^1, \hat{\mathbf{y}}^2, \ldots, \hat{\mathbf{y}}^J)$, that is the set of the J predicted KPs for $\mathbf{x}$ (Fake KPs).

The objective is to generate Fake KPs that match exactly the True KPs: $\hat{\mathbf{y}} \equiv \mathbf{y}$.

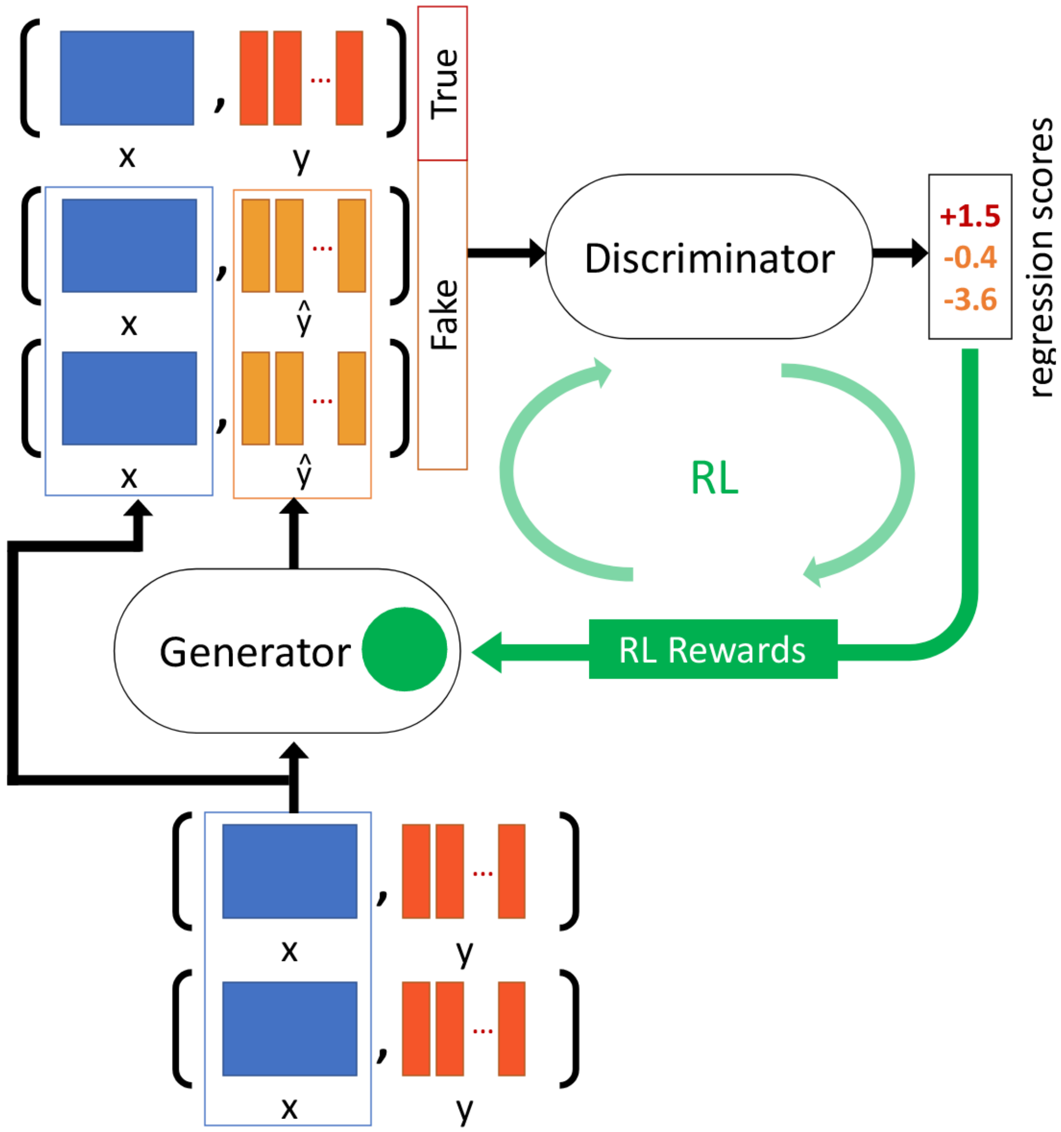


Figure 3.1: Schema of BeGan-KP.

3.3.2 Generator

The Generator G takes as input the document $\mathbf{x}$ and generates as output the sequence of $\hat{\mathbf{y}}$ (Fake KPs).

Following the work of [118] we use the CatSeq model as Generator. It consists in an Encoder-Decoder model in which the Encoder is a bidirectional Gated Recurrent Unit (GRU) and the Decoder is a forward GRU. It is based on CopyRNN by [80].

We choose this component because it embeds some interesting features. It exploits the copying mechanism [35] to deal with long-tail words. These are words which are removed from the vocabulary due to their low frequency but are often topic-specific and therefore good candidates to be KPs. It also introduces the capability of predicting a variable number of Keyphrases for different documents. Furthermore it employs a beam-search strategy during the decoding step, meaning that at each time step the model decodes not just one word (greedy-search) but the top k most probable words. This allows generating more consistent sequences of words.

3.3.3 Discriminator

The Discriminator D receives as input the document $\mathbf{x}$ and a set of Keyphrases. These might be either the True KPs $\mathbf{y}$ or the Fake KPs $\hat{\mathbf{y}}$. Its task is to judge whether the KPs are True or Fake.

We introduce a novel Discriminator based on the language model BERT [25]. Differently from the previous literature, our idea is to exploit the strength of the language model characteristics to classify the quality of the input pair $(\mathbf{x}, \mathbf{y})$. This judgement is given as a *regression score*, which is lower for Fake KPs and higher for True KPs. In this way the regression score can be easily interpreted as the *reward* in the Reinforcement Learning module, giving to the system an inherent clarity. Moreover, different BERT-based models and reward configurations have been tested at an early stage, and the choice of a regression model provided the best results.

The language modelling component is able to achieve a better comprehension of the relationship of the two input sequences, while the robust pretraining allows us to use it efficiently even in a low-resource scenario.

In particular, the Discriminator model consists of four subcomponents (see Figure 3.2) :

- **Input preparation**. The input pairs $(\mathbf{x}, \mathbf{y})$ are tokenized and the tokens are concatenated to be compliant with the general pattern `[CLS]<x>[SEP]<y1><;>...<;><yn>[SEP]`. `[CLS]` and `[SEP]` are special tokens which signal the start of the input and the end of text sequence respectively, `<x>` is the sequence of tokens for the document $\mathbf{x}$, `<yi>` is the sequence of tokens for the KP $\mathbf{y}^i$. Different KPs are separated by semicolon `<;>`. Note that the `[SEP]` token in the center is used to split the input sequence into document and KPs.
- **BERT modelling**. The input sequence is processed by a pretrained BERT model. It generates a word embedding of all the tokens and then passes them through 12 Encoder blocks. As it is basically a positional language model, it returns the last hidden states for each of the initial tokens.
- **Output aggregation**. Each of the outputs of the preceding step can be seen as an highly abstract embedding of the corresponding token. We aggregate the output of all the hidden

states and evaluate their mean to obtain an embedding for the whole input sequence $E = E(\mathbf{x}, \mathbf{y})$. Note that in this way E is not generated using only the output obtained from the `[CLS]` token, but making use of the representations of all the tokens instead. Based on our preliminary experiments as well as literature references [25], this value is considered to represent a better summary of the semantic content of the input.

- **Regression**. E is processed by the regression layer, a fully connected linear classifier, and a regression score is calculated. This is trained to be high for True KPs (human-curated) and low for Fake KPs (artificially generated), and is used as the reward in Reinforcement Learning.

The overall output of the Discriminator is therefore a regression score relative to the combination of input document and the related KPs.

3.3.4 Reinforcement Learning

To overcome the problem of non differentiability of the output layer of our architecture we extend the Reinforcement Learning strategy proposed by [142] in the domain of KP Generation. In particular, we consider the Generator G as an agent whose action a at time step t is to generate a *word* y_t, which is part of the set of predicted KPs $\hat{\mathbf{y}}$ for the document $\mathbf{x}$. In this scenario the Discriminator D plays the role of the environment that evaluates the actions made by G and gives back a reward. Agent G acts following a policy

$$\pi = \pi(y_t|s_t, \mathbf{x}, \theta) \tag{3.1}$$

that is a function representing the probability distribution of y_t given the current state $s_t = (y_1, \ldots, y_{t-1})$, the sequence of words so far generated. The policy function is differentiable with respect to the set of parameters θ of G. Once the agent G generates the predictions, the environment D gives back a reward

$$r_t = f(y_1, \ldots, y_t|\mathbf{x}) \tag{3.2}$$

and moves to the state s_{t+1}. The reward is a quality measure of the action made by the agent G, and depends on the words generated up to the current time step (subset of $\hat{\mathbf{y}}$) given the input document $\mathbf{x}$. The agent G acts to maximize the reward, that is to maximize a differentiable optimization function $J(\theta)$ that gives a measure of the performance of G. According to the policy gradient theorem and the REINFORCE algorithm [132] the gradient of $J(\theta)$ can be expressed as:

$$\nabla J(\theta) = \mathbb{E}_\pi \left[\sum_t r_t \nabla log(\pi(y_t|s_t, \mathbf{x}, \theta)) \right] \tag{3.3}$$

where the sum extends to all the time steps needed to generate the complete sequence $\mathbf{y}$.

The expectation $\mathbb{E}_\pi$ in Equation 3.3 can be approximated using a complete sequence $\hat{\mathbf{y}}$. In order to calculate the cumulative rewards of Equation 3.2 we use the regression score of a complete sequence of generated KPs: $r = D(\hat{\mathbf{y}})$.

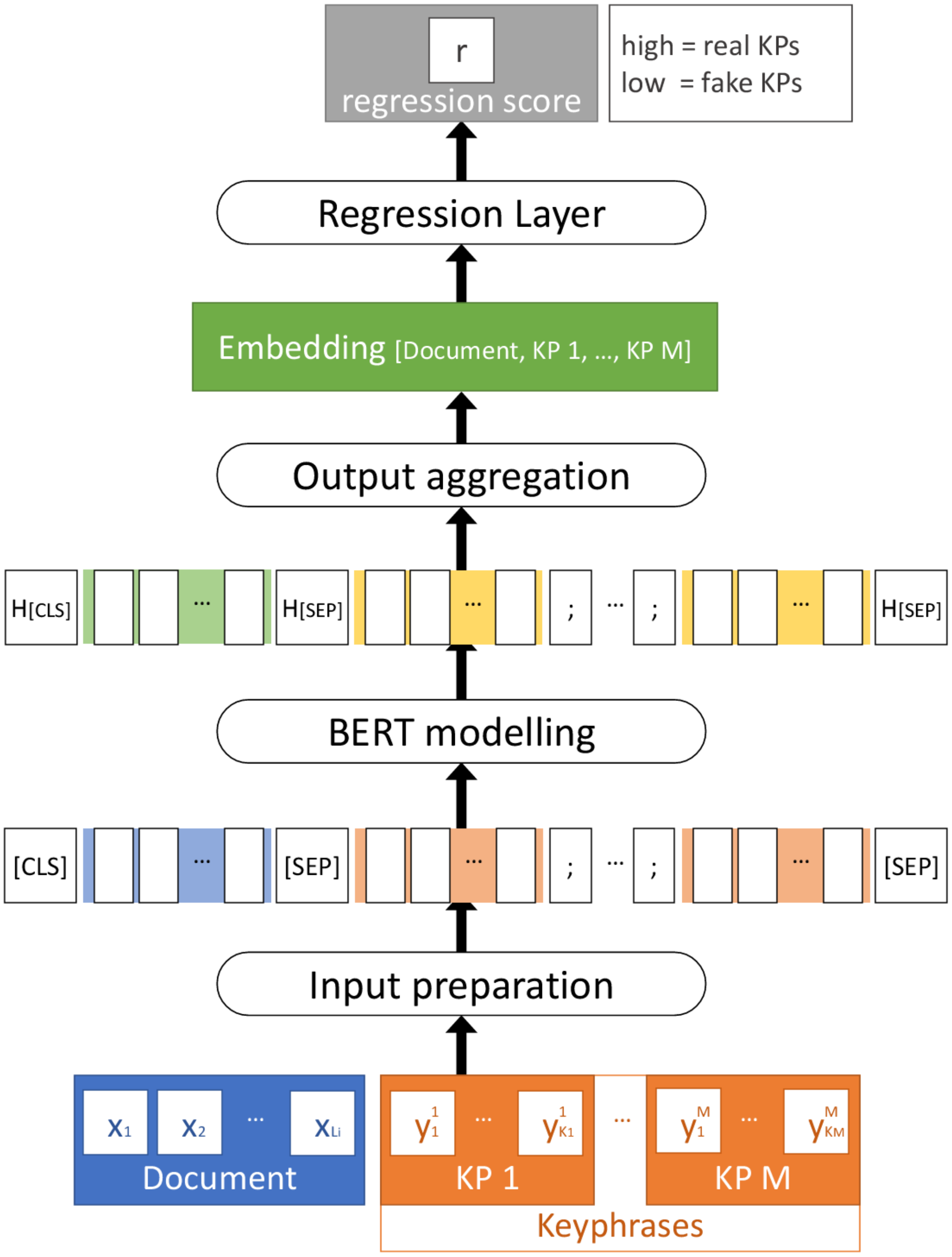


Figure 3.2: Schema of the Discriminator and its four processing phases.

Considering that maximizing the optimization function $J(\theta)$ is equivalent to minimizing its additive inverse, we can define the loss function of G as $L(\theta) = -J(\theta)$ and an estimator of its gradient as:

$$\nabla L(\theta) \approx -\sum_t (r - b)\nabla log(\pi(y_t|s_t, \mathbf{x}, \theta)) \tag{3.4}$$

where the regularization term b is introduced to reduce the variance of the above $\nabla L(\theta)$ estimator. It is essentially the cumulative reward $r = D(\bar{\mathbf{y}})$ where $\bar{\mathbf{y}}$ is a greedy decoded predicted sequence. The aim is to promote rewards that show effective improvements over greedy sequences [109].

3.3.5 GAN Training

The first step is to train a first version G_0 of the Generator using the Maximum Likelihood Estimation (MLE). G_0 is then used to generate the Fake KPs $\hat{\mathbf{y}}$. $\hat{\mathbf{y}}$ and the ground truth $\mathbf{y}$ are used to train the first version of the Discriminator D_0 with Mean Squared Error (MSE) loss:

$$MSE(x, \hat{x}) = \frac{1}{N}\sum_{i=0}^{N}(x_i - \hat{x}_i)^2 \tag{3.5}$$

Starting from Generator G_1 training is performed using RL, so the loss is given by $L(\theta)$ as shown in Section 3.3.4. The training of the Discriminator remains the same. After each training iteration (G_j, D_j), predictions are tested to evaluate the scores $F1@M$ and $F1@5$.

3.4 Experiments and Evaluation

3.4.1 Datasets and Metrics

We compare our solution with state-of-the-art approaches on five datasets which are commonly used in literature:

KP20k [80] It consists of 567,830 titles and abstracts from computer science papers. The usual split consists of 20,000 samples for testing, another 20,000 for validation, while the remaining 527,830 samples are used for training. In our low-resource scenario we only use 2,000 out of the >500,000 training samples.

Inspec [47] The complete dataset is composed of 2,000 abstracts from Computers and Control, and Information Technology disciplines. A subset of 500 samples is used for testing.

Krapivin [63] The original released dataset is composed by 500 complete articles belonging to the domain of computer science. For KP Generation purposes only titles and abstract are used. The first 400 samples in alphabetical order are selected for testing.

NUS [89] A set of 211 scientific publications, all used for testing.

Semeval2010 [58] 288 conference and workshop papers from the ACL Computer Library. 100 are used for testing.

A brief report of main statistics of the test sets used is given in Table 3.1.

	Kp20k		**Inspec**		**Krapivin**		**Nus**		**Semeval**	
	#	%	#	%	#	%	#	%	#	%
Present KPs	66,267	62.91	3,602	73.59	1,297	55.57	1,191	52.26	612	42.41
Absent KPs	39,076	37.09	1,293	26.41	1,037	44.43	1,088	47.74	831	57.59
Total KPs	105,343	100.00	4,895	100.00	2,334	100.00	2,279	100.00	1,443	100.00
Test samples	20,000		500		400		211		100	

Table 3.1: Statistics on test samples for the five datasets.

Model	**Kp20k**		**Inspec**		**Krapivin**		**Nus**		**Semeval**	
	F1@M	*F1@5*	*F1@M*	*F1@5*	*F1@M*	*F1@5*	*F1@M*	*F1@5*	*F1@M*	*F1@5*
catSeqD [143]	-	**0.348**	-	0.276	-	**0.325**	-	0.374	-	**0.327**
catSeqCorr-2RF1 [12]	0.382	0.308	0.291	0.240	0.369	0.286	0.414	0.349	0.322	0.278
catSeqTG-2RF1 [12]	**0.386**	0.321	0.301	0.253	0.369	0.300	**0.433**	**0.375**	**0.329**	0.287
GAN [118]	0.381	0.300	0.297	0.248	**0.370**	0.286	0.430	0.368	-	-
BeGan-KP (our approach)	0.318	0.309	**0.383**	**0.356**	0.332	0.317	0.388	0.366	**0.329**	0.319

Table 3.2: Results of present keyphrases for five datasets. Our approach is BeGan-KP.

All datasets are preprocessed following [12]: duplicate papers are removed from KP20k, and for each document the list of KPs is sorted in order of appearance in the document. Digits in the input texts are replaced with the special token `<digit>`.

Results are evaluated using $F1$ score. In particular $F1@5$ and $F1@M$ are employed: the first is calculated considering only the top 5 high scoring KPs, the second is computed taking into account all the predictions.

All sample documents are annotated with human curated KPs. Of the above mentioned datasets, only KP20k is used for training; all the others are used only for testing and evaluation. Note that the strength of the language model of our Discriminator allows us to use only a small subset of the data samples during training: the whole architecture has been trained with a subset of 2,000 samples instead of the >500,000 used by the other state-of-the-art approaches.

3.4.2 Implementation Details

The initial MLE model G_0 is trained with a batch size of 12 and Adam optimizer [59]; during RL training, batch size is 32. The Discriminator is trained with a batch size of 3 and AdamW optimizer [72]. The pretrained BERT model is the base uncased version, with 12 layers, 12 attention heads, and hidden size of 768. The maximum input length after tokenization is fixed to 384 tokens. We use the implementation provided in the python library transformers by huggingface [134]

Training and experiments have been executed on a PC with a GeForce RTX 2080 GPU, 11GB.

3.4.3 Experimental Results

Our proposed solution BeGan-KP, trained on 2,000 samples, has then been compared with the following state-of-the-art approaches: CatSeqD [143]; CatSeqCorr-2RF1 and CatSeqTG-2RF1 [12], and GAN [118]. The results of our tests are shown in Table 3.2.

First, we can note that BeGan-KP achieves results competitive with the best performing techniques, even using a limited set of samples (all the other approaches were trained on the whole KP20k).

Looking at the results in detail, we obtain by far the best performance for Inspec both in $F1@5$ and $F1@M$.

Our approach has other good results in $F1@5$ metrics, specifically in Krapivin and Semeval2010 where our values are only slightly lower than the best. Since $F1@5$ is calculated considering the 5 predictions with the highest score, we can say that our model is capable of producing high quality Keyphrases reliably, and of outperforming or at least matching other best-performing models in this specific task. This confirms the strength and consistency of our architecture. In addition, we obtain the best $F1@M$ score for Semeval2010. Note that Semeval2010 is a demanding test dataset as it is the smallest of the five, and the gross amount of KPs to predict is the lowest (612 present KPs out of a total of 1,443), leading to a great variance in the output. Finally, consider that in Equation 3.3 the expectation of the policy function is evaluated using only one complete sequence $\hat{\mathbf{y}}$, inducing a high variance in the ∇J. This is a general issue of Reinforcement Learning applied to GANs for text generation and generally leads to unstable training process and slow convergence [142]. Thanks to the capability of the language model embedded in our architecture, in our experiments the training process shows a quick convergence in terms of number of training iterations. In fact, the reported results have been achieved at the second iteration (G_2 generator).

3.5 Summary

In this chapter we introduced an approach to the task of extractive Keyphrase Generation in a low-resources scenario, BeGan-KP. It is based on the GAN framework with a novel BERT based Discriminator model, trained by mean of the Reinforcement Learning paradigm. It has been tested on five public datasets showing performances competitive with state-of-the-art approaches while using less than 1% of the available training data, achieving a great training efficiency.

Chapter 4

Protein Structure Prediction based on Generative, Discriminative and Linear Models

4.1 Introduction

The three-dimensional structure of a protein is determined by its amino acid sequences and it is key to its functional mechanisms. In addition, its prediction is a major challenge for biologists [2, 112]. In recent years the astonishing success of the AlphaFold prediction systems based on multiple alignments and deep learning [54] essentially solved the problem. However, understanding the effect of single amino acid substitutions on the stability of the structure or on its interactions still remains a difficult problem. In order to do this, it is important to estimate the effect of each amino acid at each position on the secondary structure of the central residue.

Here we address the prediction of secondary structure from single sequences to characterize and compare different approaches. We aim at understanding how the different predictive approaches leverate this information coming from neighboring amino acids to predict the local structure structure of the protein.

Protein secondary structure prediction is one of the classical problems in bioinformatics and it has been addressed extensively using many methods.

The interest in predicting secondary structure stemmed from different reasons:

1. identifying secondary structures provides an idea about overall structural categories;
2. secondary structure plays an important role in determining how proteins fold; [153, 92, 98]
3. since the practical experiments that are used to determine protein structure were and are costly and time-consuming, it appears convenient to develop prediction methods. In addition, the number of known sequences is at least 1,000 times bigger than those of examined structures [138], a fact that calls for predictive methods.

Although various techniques were introduced for predicting protein secondary structure from multiple sequence alignments, there have been relatively few attempts to improve the performance of secondary structure predictions from single sequences. The prediction of secondary structure from single sequence challenge, as reviewed by Rost and Sander [111], started with Pauling and Corey in 1951, when they predicted helical and sheet conformations for protein polypeptide backbone. In addition, they focused on predicting structural properties of proteins like backbone dihedral angles, leveraging this information for secondary structure prediction [96, 97]. After these early works, a first generation of prediction methods has been put forward based on local amino acids' propensities to adopt a particular secondary structure, like e.g. the Chou and Fasman methods [21]. The second generation of algorithms was based on information theory instead, considering the effects of amino acids and pairs of amino acids within a window of 11 to 21 adjacent residues. Among these methods the Garnier-Osguthorpe-Robson method was among the best available ones, together with its subsequent refinements [30, 31, 29].

Later on, evolutionary information and the introduction of novel neural network architectures led to an outstanding increase in the accuracy of the predictions [110, 99]. The interested reader will find details and further references in the papers cited in this work and in recent reviews [121].

More recently novel deep learning methods have been applied to the problem of single sequence prediction of secondary structure.

Heffernan et al. [39] proposed LSTM-BRNNs (SPIDER3-Single), a single-sequence-based model using long short-term memory bidirectional recurrent neural networks to solve the task. The model can predict multiple one-dimensional (1D) structural properties with relatively high accuracy, especially for non homologous sequences.

The ProteinUnet architecture has been introduced as an alternative way to SPIDER3-Single for sequence-based prediction of protein secondary structure. The model is based on the U-NET architecture which consists of blocks placed symmetrically as contracting and expanding paths [61].

The deep neural network architectures described above learn patterns from protein sequences and succeed in giving more accurate predictions than traditional methods. Still, the way this is done is hidden in the connections and parameters of the network, which acts as a black box. It is somehow frustrating that the learning machine uses the patterns in the sequences in the best ways, but does not disclose this information to the user.

In recent years several methods have been developed to relate the predictions of the model architecture to its input features (e.g. DeepLIFT [116]), making it possible to assess the interplay of input variables in determining the output. These input-output relationships may be compared in principle with the mutual information of input variables and/or pairs of input variables and the output, and with linear models thus providing a glimpse of how the network is using input information.

Here we use different approaches to the task of predicting protein secondary structure from protein single sequences:

i) a linear model solved exactly for coefficients minimizing the mean square prediction error;

ii) a linear model implemented as a single-layer feedforward neural network;

iii) a neural network based on Bidirectional Long Short-Term Memory (Bi-LSTM) architecture;

iv) a neural network based on the Transformer framework with a novel BERT-based model;

v) a neural network based on the Encoder-Decoder framework, utilizing a T5 model.

Using different predictive models, based on really different and recent architectures gives us the opportunity to compare extensively performances and understanding how the single position information is linked to the output.

We compare the different predictive approaches in terms of the feature importance. This is identified with the absolute magnitude of the coefficients for the linear models, whereas for neural networks it is assessed by propagating activation differences in the protein single-sequence with respect to a reference activation, using the DeepLIFT (Deep Learning Important FeaTures) [116] method.

The comparison of the models among themselves and with the mutual information may help in rationalizing the way deep neural network architectures use the information coded in protein single-sequences.

4.2 Materials and methods

4.2.1 Datasets and Metrics

The dataset of structures used in this work was obtained by downloading non-redundant sequences (at less than 40% identity) from the culling server Pisces [127] with resolution of < 4.0A and R-factor < 1.0. The set entails 27,316 sequences. Additionally splitting sequences at breaks in connectivity of the corresponding PDB file, resulted in 35,964 sequences with maximum length of 2,166 amino acids.

This dataset was split into a Training set (TR33964) consisting of 33,964 different chains and a test dataset (TS2000) consisting of 2,000 different chains. Each subsequence with no breaks was divided in subsequences of length 17, centered around the position for which we aim to predict the secondary structure (see Figure 4.1). This results in 5,672,821 samples for the Training set and 490,612 samples for the Test set, where each sample consists in a subsequence of 17 amino acids (8 preceding and 8 following the central residue) and a label indicating the secondary structure of the protein at the position of the central residue.

Note that the dataset we use in this work is larger than the ones used in the papers describing SPIDER3-Single and ProteinUnet. This does not influence the overall performance of the models, as preliminary experiments showed that the same model developed on TR33964/TS2000 or on the SPIDER3-Single dataset reached the same classification accuracy. At the same time, a larger dataset enables us to perform a more in-depth analysis of the models' predictions.

The secondary structure of the central residue was assigned using the DSSP databank, [122] which outputs for each residue one of eight possible states [55]. The eight states were further clustered in three main classes as detailed in the following section.

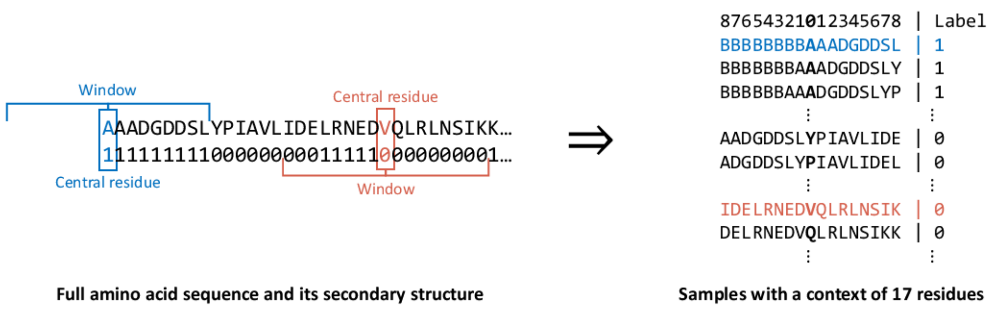


Figure 4.1: Example of the process that splits the full protein sequence in subsequences with a context window of 17 residues/amino acids.

4.2.2 Input Features and Model Output

The input for each of the predictive models is one feature vector for each of the 17 amino acids in the subsequence. In the case of the linear models and the LSTM, the feature vector is a one-hot encoding of the amino acid (size 20), while in the case of BERT and T5 it is a word embedding generated by the model itself. As BERT and T5 accept plain text as input, they are given the sequence of 17 amino acids directly as characters.

We did not augment the data with any other single sequence features, such as BLOSUM62 substitution scores [41] and/or physicochemical properties [38, 40] of the amino acid. These features did not offer any improvement as shown in [38].

Each model predicts the secondary structure state for the amino acid at the center of the subsequence (see Figure 4.1). The secondary structure state is represented by one of the possible eight secondary structure states, grouped as follows:

1. the three helical states: 3_{10}-helix (G), α-helix (H), and π-helix (I);
2. the two extended hydrogen-bonded states: β-bridge (B) and β-strand (E);
3. the three coil types: high curvature loop (S), β-turn (T), and coil (C).

These eight states are converted into three states using the following assignment: states G, H, and I to Helix (H); B and E to Extended (E); and S, T, and C to Coil (C).

We designed our networks and linear models to independently predict the 3-state secondary structure state.

4.2.3 Mutual information

Before analyzing the models, we want to provide a reference assessment of the influence of single (or pair) of amino acid positions on the secondary structure state of a central residue in 17-residue subsequence. In order to do this, we consider as a measure of information:

- the entropy of the 17 positions and the secondary structure state;

- the mutual information of each pair of position and state;
- the three-variable mutual information for each pair of positions and the secondary structure state.

Entropy and mutual information are defined as follows.

Each position in the 17-residue window may assume 21 possible values: 20 amino acids plus the "absence of amino acids", used as padding for the initial and final positions in the sequence. The central position can only assume 20 values, as we require the presence of central amino acid associated to a secondary structure state. The latter can be one of three possible states.

In the following equations, we reserve index 0 for the secondary structure state variable and indices 1 to 17 for amino acid position variable.

The entropy of each variable x_i is estimated from $\{n_{i\ell}\}$, that is the counts of occurrences of its possible values $\{v_\ell\}$:

$$\hat{S}(x_i) = -\sum_{\ell=1,\dots,ns_i} \frac{n_{i\ell}}{\sum_{\ell=1,\dots,ns_i} n_{i\ell}} \log\left(\frac{n_{i\ell}}{\sum_{\ell=1,\dots,ns_i} n_{i\ell}}\right) \tag{4.1}$$

where ns_i is the number of possible values (states) of variable x_i.

The joint entropy of two variables is computed as:

$$\hat{S}(x_i, x_j) = -\sum_{\substack{\ell=1,\dots,ns_i \\ m=1,\dots,ns_j}} \frac{n_{ij;\ell m}}{\sum_{\substack{\ell=1,\dots,ns_i \\ m=1,\dots,ns_j}} n_{ij;\ell m}} \log\left(\frac{n_{ij;\ell m}}{\sum_{\substack{\ell=1,\dots,ns_i \\ m=1,\dots,ns_j}} n_{ij;\ell m}}\right) \tag{4.2}$$

where $n_{ij;\ell m}$ is the count of simultaneous occurrences of value i assuming the value v_ℓ and variable j assuming the value v_m.

The mutual information between two variables is estimated as:

$$\hat{MI}(x_i, x_j) = \hat{S}(x_i) + \hat{S}(x_j) - \hat{S}(x_i, x_j) \tag{4.3}$$

Note that

$$\hat{MI}(x_i, x_i) = \hat{S}(x_i) \tag{4.4}$$

In order to quantify also the effect of pairs of positions on the secondary structure state we consider the three-variable joint entropy and mutual information. The joint entropy of three variables is defined similarly as above:

$$\hat{S}(x_i, x_j, x_k) = -\sum_{\substack{\ell=1,\dots,ns_i \\ m=1,\dots,ns_j \\ n=1,\dots,ns_k}} \frac{n_{ijk;\ell mn}}{\sum_{\substack{\ell=1,\dots,ns_i \\ m=1,\dots,ns_j \\ n=1,\dots,ns_k}} n_{ijk;\ell mn}} \log\left(\frac{n_{ijk;\ell mn}}{\sum_{\substack{\ell=1,\dots,ns_i \\ m=1,\dots,ns_j \\ n=1,\dots,ns_k}} n_{ijk;\ell mn}}\right) \tag{4.5}$$

The mutual information of three-variable is defined as [77]:

$$\hat{MI}(x_i, x_j, x_k) = \hat{S}(x_i) + \hat{S}(x_j) + \hat{S}(x_k) - \hat{S}(x_i, x_j) - \hat{S}(x_i, x_k) - \hat{S}(x_j, x_k) + \hat{S}(x_i, x_j, x_k) \quad (4.6)$$

When one of the variables is the secondary structure state, the three-variable mutual information may be used to assess the information which is gained on the secondary structure when the amino acids at two positions are known simultaneously as opposed to when they are known independently. The latter case is implemented in linear models as those discussed here, whereas the former is implemented in machine learning algorithms.

When the three-variable mutual information involving the secondary structure variable is written in terms of two-variable (albeit also composite) mutual information:

$$\hat{MI}(x_i, x_j, x_0) = \hat{MI}(x_i, x_0) + \hat{MI}(x_j, x_0) - \hat{MI}(\{x_i, x_j\}, x_0) \quad (4.7)$$

It is apparent that any additional effect coming from the combination of x_i and x_j, with respect to the separate effect of x_i and x_j will translate into a negative three-variable mutual information.

4.2.4 Linear Models

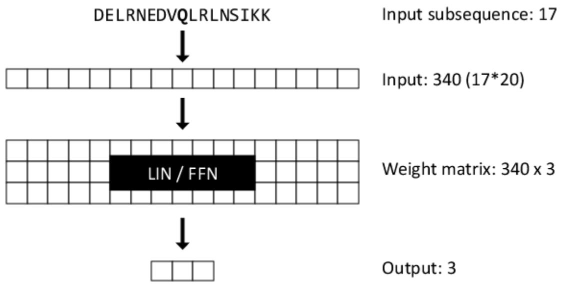


Figure 4.2: Schema of the LIN and FNN architectures.

Multilinear regression (LIN)

A linear model (LIN) was implemented in an adhoc program to find the best 340 x 3 linear coefficients (17 one-hot vectors of size 20 to fit the one-hot vector of length 3). A linear system of the form $A\vec{x} = \vec{y}$ was solved by pseudo-inversion, i.e. $\vec{x} = (A^tA)^{-1}A^ty$, which is the solution that minimizes the sum of square errors in prediction. Due to matrix size, the model was fitted on only 136000 randomly selected input samples. However, repeated experiments showed that the coefficients found were essentially independent of the set of inputs chosen.

Feedforward network (FFN)

We implemented a simple feedforward neural network (FFN) model, consisting of one linear layer and no activation function. As for LIN, the input of FFN was the concatenation of the 17 one-hot vectors of the subsequence, and the output was a one-hot vector of size 3. The model was trained on the whole Training set for 50 epochs, with stochastic gradient descent, using a batch size of 128 samples, a learning rate of 1e-3 and Mean Square Error as a loss function.

Figure 4.2 shows a schema of both the LIN and FFN model.

4.2.5 Bi-LSTM neural network (LSTM)

A Bidirectional LSTM (Bi-LSTM) is a variant of the LSTM model [43] which scans the input sequence in both directions: forward, from the start to the end of the sequence, using a Forward LSTM layer made of forward cells (Figure 4.3, light blue layer); and backwards, from the end to the start of the sequence, using a Backward LSTM layer made of backward cells (Figure 4.3, red layer). At time t, a Bi-LSTM has two different active cells: a forward cell $\overrightarrow{c}$ and a backward cell $\overleftarrow{c}$. The forward cell uses the current input $\overrightarrow{x_t}$ and the state of the previous cell ($\overrightarrow{c_{t-1}}$) to calculate its internal cell state ($\overrightarrow{c_t}$), and hidden state ($\overrightarrow{h_t}$). In Figure 4.3, the cell states flow left-to-right, while the hidden states are output from the bottom of the cells. At the same time, the backward cell uses its current input $\overleftarrow{x_t}$ and the state of the previous backward cell ($\overleftarrow{c_{t-1}}$) to calculate its internal cell state ($\overleftarrow{c_t}$), and its hidden state ($\overleftarrow{h_t}$). Cells states of the Backward LSTM flow from right-to-left, while hidden states are output from the bottom of the cells. The choice of using a Bi-LSTM instead of a simple LSTM network to predict the secondary structure is justified by the fact that such typology of neural network has achieved promising results while also beating the performance obtained with simple LSTMs, as shown in [39]. This is due to the ability of the Bi-LSTM to use more contextual information, as is it able to read the sequence both ways.

We implemented a Bi-LSTM model with a hidden state size of 64. The last hidden states of the forward and backward LSTMs are concatenated (size 128) and passed to a small postprocessing network to project them to the output space. The postprocessing network is formed by a linear layer (128x32), a ReLU activation function [117], and another linear layer (32x3). A schema of the network is shown in Figure 4.3. The input of the LSTM is a sequence of 17 one-hot vectors representing the amino acids, while the output is a one-hot vector of size 3. Similarly to the FFN model, LSTM was trained on the whole Training dataset for 50 epochs with stochastic gradient descent, using a batch size of 128, learning rate of 1e-1 and Mean Square Error as a loss function.

4.2.6 BERT-based neural network (BERT)

BERT [26] (Bidirectional Encoder Representations from Transformers) is a Transformers-based architecture that has revolutionized many tasks in the Natural Language Understanding field. It is a large language model, composed of a series of stacked encoder blocks, aimed at creating deeply contextualized word embeddings. Each encoder processes the whole input sequence and transforms it into an embedding sequence that can be either used for inference or fed into another encoder block for further processing. This model is commonly used in the following way:

- A domain of interest is chosen, together with a large corpus that can be used to train the model.

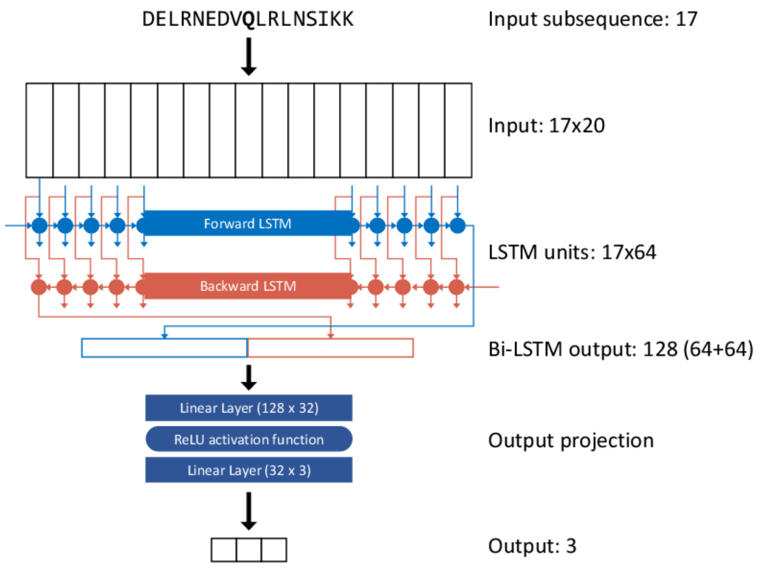


Figure 4.3: Schema of the proposed Bi-LSTM architecture.

- The BERT model is initialized from scratch and *pretrained* on the large corpus, to learn domain-specific correlations between the words/tokens in the given samples. The pretraining is usually performed using Masked Language Modeling.
- The pretrained model is extended with new untrained layers and *finetuned* (i.e., trained) on a new specific task in the same domain of interest.

In the last years, a great variety of pretrained BERT-based models have been created an publicly released, such as PubMedBERT [36] for the biomedical domain, LEGAL-BERT [11] for legal documents and CodeBERT [28] for source code.

We implemented a BERT-based model and trained it from scratch on the domain of protein encoding and secondary structure prediction. The basic BERT model is usually composed of 12 encoder blocks and can read sequences up to 512 words. Preliminary experiments showed that such a large network size led to a low performance in our task. For this reason, we created a model with only 4 encoder blocks and a maximum sequence length of 64 words. The embedding size of the model is 768 (standard BERT embeddings).

The model was *pretrained* on 20% of the full protein sequences in the training set with the task of Masked Language Modeling: given an input sequence, a random amino acid was "masked" (i.e., replaced with the special token [MASK]) and the model was asked to guess the correct amino acid to fill the gap. The pretraining was carried out for 50 epochs, with a batch size of 2048. The vocabulary of the model only includes the characters for the 20 amino acids (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y) plus five special tokens needed for the model's functioning ([CLS], [PAD], [SEP], [UNK], [MASK]).

The pretrained model was then extended with an untrained linear layer for *finetuning*. The linear layer projects the word embedding of the central amino acid (size 768) to the output space (3 states). Figure 4.4 is a schema of the final model, which takes as input the string-representation of the amino acids subsequence, embeds it, runs it trough a stack of encoders and then projects the embedding to the three possible output states. BERT was finetuned on the training set subsequences for 20 epochs, with a batch size of 128 and a learning rate of 1e-6.

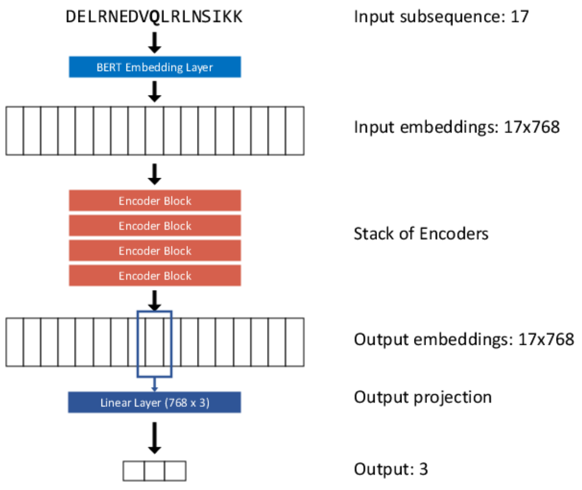


Figure 4.4: Schema of the implemented BERT architecture.

4.2.7 T5 neural network (T5)

T5 [102] is a text-to-text encoder-decoder model pretrained on a multi-task mixture of unsupervised and supervised tasks. It is composed of a stack on encoder blocks (similarly to BERT), which output an embedding of the input sequence, and a stack of decoder blocks, which transform the embedding representation back into plain text. Figure 4.5 shows a schema of the working of T5. The dimension of the embeddings is 512, and the stacks of encoders/decoders consists of six blocks each. The output of the model is a string containing one of the three words "Helix", "Coil" or "Extended". T5 was finetuned on the training set subsequences for 9 epochs, with a batch size of 256 and a learning rate of 1e-5.

4.2.8 DeepLIFT analysis

DeepLIFT is a method to that can be used to analyze the inner workings of a neural network, drawing relations between its input features and the resulting output. In particular, its aim is to determine the "contribution" that each input feature has in generating a specific prediction.

The DeepLIFT method requires the user to supply the network with a "reference" input (i.e., an example of an empty sequence or background noise) and a real input (e.g., one of the 17-amino acid subsequences). The algorithm then compares the output produced by the network on the reference

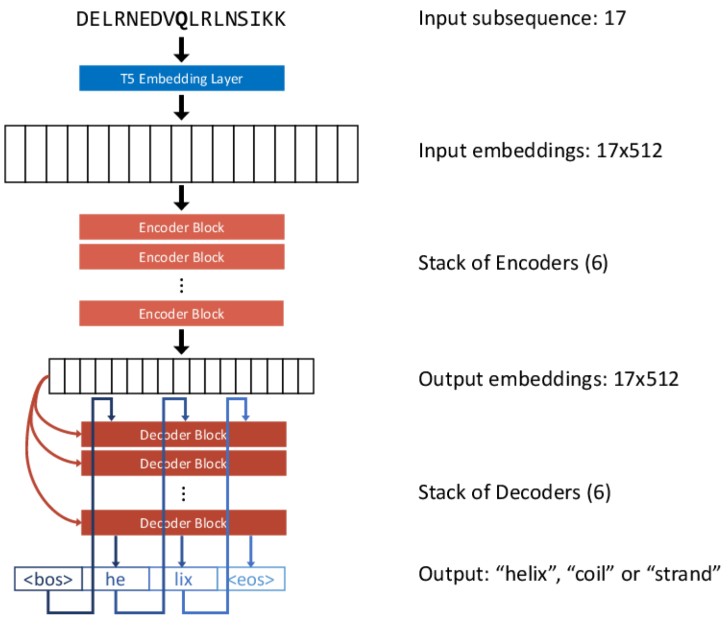


Figure 4.5: Schema of the T5 architecture.

input with the output obtained using the real input sequence. It then attributes the change of the output to the previous layer in the neural network, using gradients and network activations to determine how much each neuron contributed to the output change. The algorithm keeps traversing the network backwards (from outputs to inputs), calculating the impact that every neuron had in obtaining the final output. The final values calculated for the input features are called *attributions* or *attribution scores*.

We used the DeepLIFT algorithm on all neural network architectures (both linear and non-linear). In order to observe the patterns that the network learned for the three kinds of protein secondary structure, we calculated the attributions separately for the sets of samples belonging to the three output classes. We used an "empty" sequence as a reference value for the method, that is 17 repetitions of the value used to pad the input sequences (i.e., the "B" character in Figure 4.1). More specifically, we used an all-zero vector for the models that employ one-hot encoding, and the "padding" special token for the BERT and T5 models).

The attribution scores can be used to gain insights on the patterns used by the neural networks to determine the protein secondary structure, and can be compared to visualizations such as the absolute magnitude of the coefficients for the linear models, or the values of the Mutual Information for a position-state pair.

4.2.9 Implementation

All the methods were implemented in the environment containing Python 3.7, Pytorch [95] accelerated by CUDA 10.1 and CUDA 11.2.

4.3 Results and discussion

4.3.1 Mutual information

The mutual information was computed as described in the Methods section.

Entropy The single variable entropy was almost identical for all of the 17 positions, with the exception of a slight increase towards the ends. The cause of the increased entropy for the tails of the sequence is that the terminal residues can also assume the value "absence of amino acid". Although the "absence of amino acid" occurs with a low frequency, this additional value is never present at the central position, and increasingly more frequent moving towards the end of the sequence. This leads the entropy to increase in a similar pattern, from 2.89 (central position) to 2.94 at the tails of the distribution, in a logarithmic fashion.

Two-variable Mutual Information (position-position) When considering pairs of positions, the two-variable mutual information is dominated by the correlation of the absence of amino acid for windows entailing terminal residues, e.g. because the absence of amino acids at a position before the central one implies the absence of amino acids at all preceding positions.

Two-variable Mutual Information (position-state) The most interesting result obtained with two-variable mutual information is the one calculated between each input position and the predicted state. We computed it taking into account all possible states and considering only one state at the time, i.e. categorizing for example Helix and non-Helix states. The position-state mutual information for each position is reported in Figure 4.6.

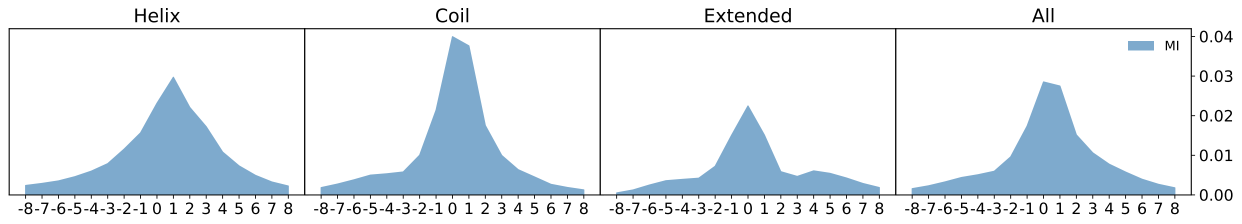


Figure 4.6: Two-variable Mutual information (position-state) of each position along the 17-residue window around the position for which secondary structure is considered. From left to right: mutual information obtained categorizing Helix states, Coil states, Extended states, and the overall Mutual Information.

The mutual information highlights the direct influence each position has on the secondary structure state. As expected, the mutual information is bell-shaped around the central position (the position for which the secondary structure is considered). When the single states, Helix, Coil and Extended are considered, some differences are apparent. For Helices the curve has a larger width than for Coils and Extended structures, consistent with the spread in local interactions. Another interesting feature is that for Coils there is a peak at the central position (as expected), but also at the following one, consistent with the disruptive effect some amino acids have on other secondary structures. For Extended structures the mutual information appears less consistent with the non-local interactions that stabilize this secondary structure element. However, the interpretation of the plot is not straightforward: as stated before, we compute the mutual information comparing Extended and non-Extended samples. Because of this, the mutual information implicitly takes into account the characteristics of Helices and Coils too (grouped together in the non-Extended class), and the plot highlights which positions are more important to distinguish between these two groups.

The position-state mutual information should provide a good reference for linear models which, albeit only additively, take into account the effect that each amino acid at each position has on the secondary structure of the central positions.

Three-variable Mutual Information (position-position-state) Another piece of information which is not caught by linear models is the additional effect of pairs of positions. This is captured by the three-variable (position, position, secondary structure) mutual information, computed as described in Methods section. This three-variable mutual information is directly related to how much information is gained by knowing the residue at two position simultaneously with respect to knowing them separately. Contrary to the two-variable mutual information, the three-variable mutual information can be positive (if the pieces of information provided by the two variables on the third one are not independent) or negative (when the two variables are cooperative or anti-cooperative). The corresponding heatmaps for all the states together and for the separate state are reported in Figure 4.7.

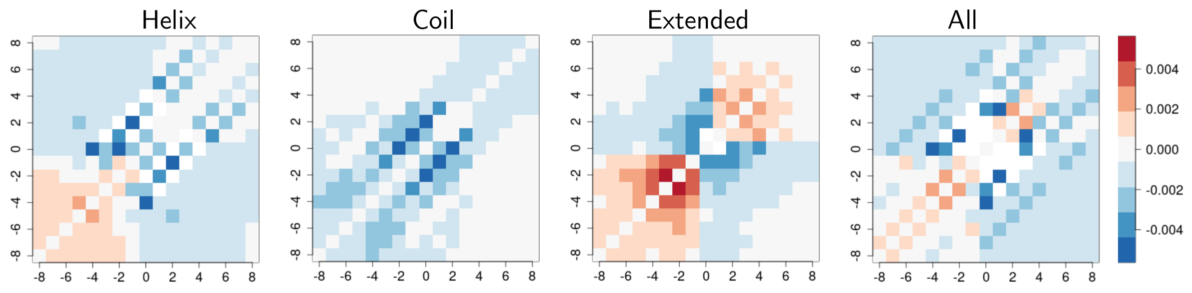


Figure 4.7: Three variable mutual information for each pair of positions and the secondary structure state of the central residue. The information along the diagonal corresponds to the information reported in Figure 4.6 and has been set to zero, for better readability of the maps. Heatmaps refer to: Helices, Coils, Extended structures and all secondary structures.

As for the two-variable mutual information, the heatmaps for the single states are calculated considering a single state (e.g., Helix vs non-Helix), so they also implicitly contain information from other states. Although interpretation is not obvious, patterns are different among different secondary structure elements, such as the negative peak at (0,-4) for Helices, or the interactions parallel to the diagonal for Coils, showing that there are (albeit small) cooperative effects which involve a large number of pairs of positions.

In order to provide an illustration of this important point we consider only Helix and Extended

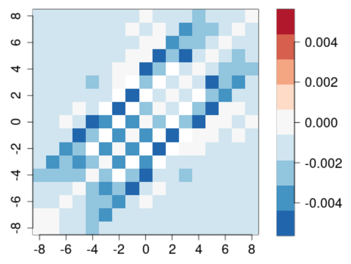


Figure 4.8: Three variable mutual information for each pair of positions and the secondary structure state of the central residue with only Helix and Extended structures are considered. The information along the diagonal has been set to zero, for better readability of the maps.

structures. The three variable mutual information is reported in Figure 4.8 and displays negative (informative) values along diagonals running parallel to the main diagonal, in particular at distances ±1, ±3, and ±4. The map is thus reporting about features determining both Helix and Extended structures as said above.

It is clear that the information reported in these heatmaps (resulting from the joint effect of pairs of positions) is not modelled by linear models, whereas neural networks should be able to reproduce even complex dependencies of secondary structures on pairs of amino acids. The mutual information computed here offers a convenient way to compare predictive methods.

4.3.2 Linear models

Multilinear regression (LIN)

A linear model has been implemented as a set of 136,000 overdetermined equations in 340 variables (17 positions times 20 elements of the 1-hot vector for each position), which has been solved to find the solution leading to minimum sum of square errors by pseudoinversion as detailed in the Methods section.

In order to compare the results with the mutual information results, the 340 coefficients were grouped 20 by 20, and we calculated the average of the absolute value of the 20 coefficients as a measure of the overall importance of that position.

The results are reported in Figure 4.9 where the different profiles for Helix, Coil and Extended structures are apparent, together with the silhouette of the mutual information, represented by the dashed line.

The plots have a bell shape, which is wider for Helices and Extended structures than for Coil structures. Similarly to what is seen in the mutual information, the bell shape for the Extended structures is less defined and noisier.

The distribution of the linear coefficients broadly parallels that of the mutual information (reported in Figure 4.6), although the difference between the central peak and the tails is more accentuated. The linear correlation of $-\bar{c}_j log(\bar{c}_j)$ of the former quantity with the mutual information of each position with the secondary structure of the central residue is 0.90.

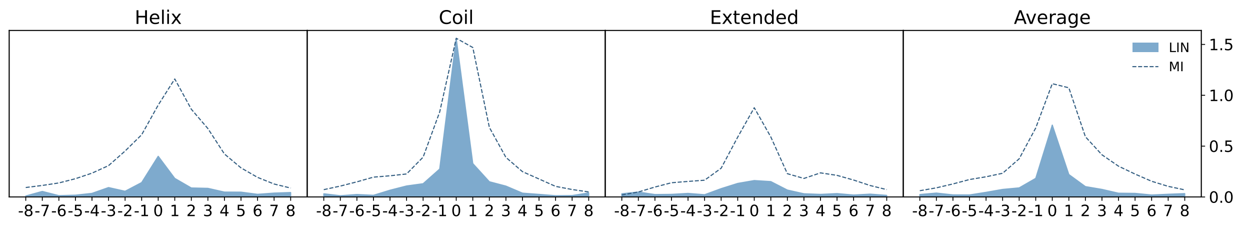


Figure 4.9: Weights of LIN for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average. For each position we plot the average of the absolute values of the 20 coefficients that encode the amino acid.

Feedforward network (FFN)

The linear model implemented using neural networks was analyzed in two ways: considering its weights, similarly to what was done for the LIN model, and using the DeepLIFT algorithm to determine the attribution scores of each input position.

Figure 4.10 displays both plots for the three output classes. We observe that the weight distribution is more similar to the mutual information than to the other linear model (LIN). It is interesting to observe such a huge difference in the weights of the model, since FFN was trained without using any weight normalization. We assume that the weight distribution of FFN is largely due to its weight initialization and some characteristics of the training samples. In particular, the weights of the neural network are first initialized to random values, sampled from a continuous uniform distribution between $-\frac{1}{\sqrt{340}} \approx -0.054$ and $\frac{1}{\sqrt{340}} \approx 0.054$, and then adjusted during training. We can see that 0.05 (after absolute value) is also the pivotal number to distinguish informative and non-informative positions in Figure 4.10 top. This is likely because the model's weights are modified only if they contribute to a wrong prediction, while they are left unchanged if they do not impact the prediction negatively. Since most of the values in the tail positions have low impact on the model's output (see mutual information), the model initially lowers their weights during the first epochs, but then only focuses on updating the weights related to the central region, which contain more information and bring greater gains in accuracy. This explains why most of the tail positions have weights close to 0.05: they have low impact on the model's accuracy, so their values remain similar to the ones set during the random initialization.

The bell-shaped curve for the Coil class is wider for the FFN model, and it displays two peaks around position -2 and 1. This behaviour loosely imitates the shifted peak of the mutual information, but the presence of a double peak is probably an effect of the stochastic nature of the training algorithm.

We can also note that the weights and DeepLIFT attributions of the FFN model create remarkably similar representations, showing that the DeepLIFT representations computed for the following neural networks can be used to draw similarities between the models.

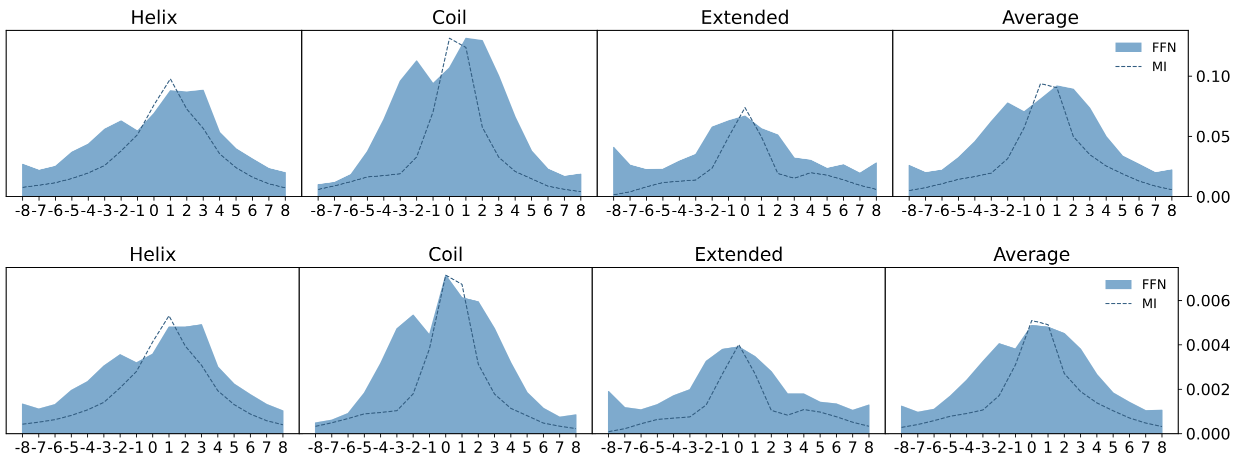


Figure 4.10: Weights and DeepLIFT attributions of the FFN model for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average. Top: weights of the FFN, for each position we plot the average of the absolute values of the 20 coefficients that encode the amino acid. Bottom: DeepLIFT attributions for each position.

4.3.3 Bi-LSTM neural network (LSTM)

For the LSTM neural network, we report the attributions calculated with the DeepLIFT algorithm for the samples belonging to the three output classes (Figure 4.11).

Comparing the attribution scores with the mutual information in Figure 4.6, we see that the bell curves are even more similar than the ones produced by the two linear models. In particular, the LSTM attribution score distribution for the Extended class resembles the mutual information scores very closely, and the distribution for the Helices peaks at position 1 (same as the mutual information).

It is surprising that a model containing non-linearities is better at replicating the mutual information patterns than the two linear models previously considered. This reflects the fact that linear models poorly reproduce the single-position state mutual information, and additionally that the average absolute value of the 20 coefficients (instead of e.g. the root mean square, or its logarithm) at each of the 17 positions might be not the best quantity to compare with mutual information.

4.3.4 BERT-based neural network (BERT)

Figure 4.12 reports the attributions scores of our BERT-based implementation on the three states. In this case, the shapes of curves for the three output classes are very similar to each other: they all have a strong peak at position 0, and the attribution score gradually decrease on both sides in a mostly symmetrical way.

Compared with the mutual information, the attributions scores are more evenly distributed among the positions. This can be explained by the densely connected structure of the stacked encoder blocks, which encourages the even spread of information along all positions of the sequence.

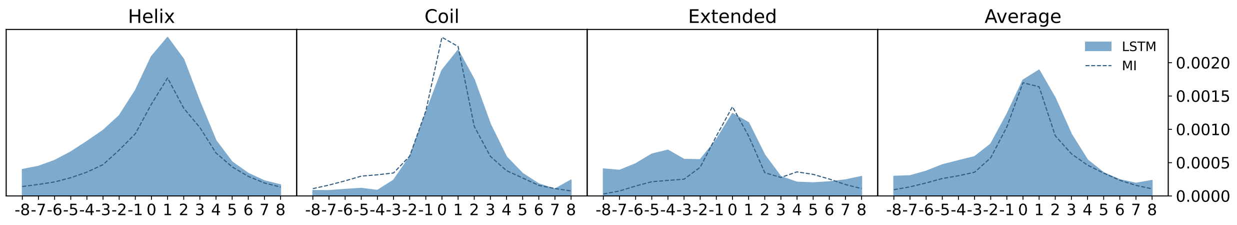


Figure 4.11: DeepLIFT attributions of the LSTM model for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average.

It is also interesting to note that the attributions for Extended structures are generally higher than the ones of Coils, which is significantly different from the behaviour of all the previous models. This might indicate a better ability in recognizing Extended patterns, which leads to stronger activations.

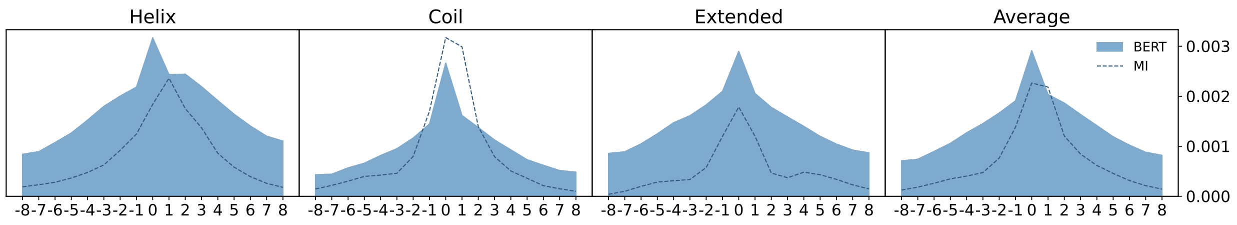


Figure 4.12: DeepLIFT attributions of the BERT model for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average.

4.3.5 T5 neural network (T5)

Finally, Figure 4.13 reports the DeepLIFT attributions for the T5 model. The shapes of the attribution plots are very similar among each other, and behave similarly to what was seen for the BERT model, descending symmetrically on both sides. They all have a central peak at position 0-1, which is more pronounced for the coil class. Contrary to BERT, the attributions scores are high for all classes.

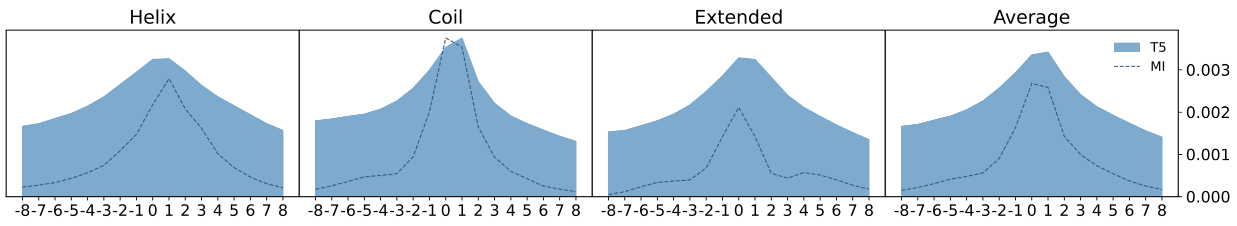


Figure 4.13: DeepLIFT attributions of the T5 model for each position in the 17-amino acid window, for the three output states (Helix, Coil, Extended) and their average.

4.3.6 Comparison of input features effects

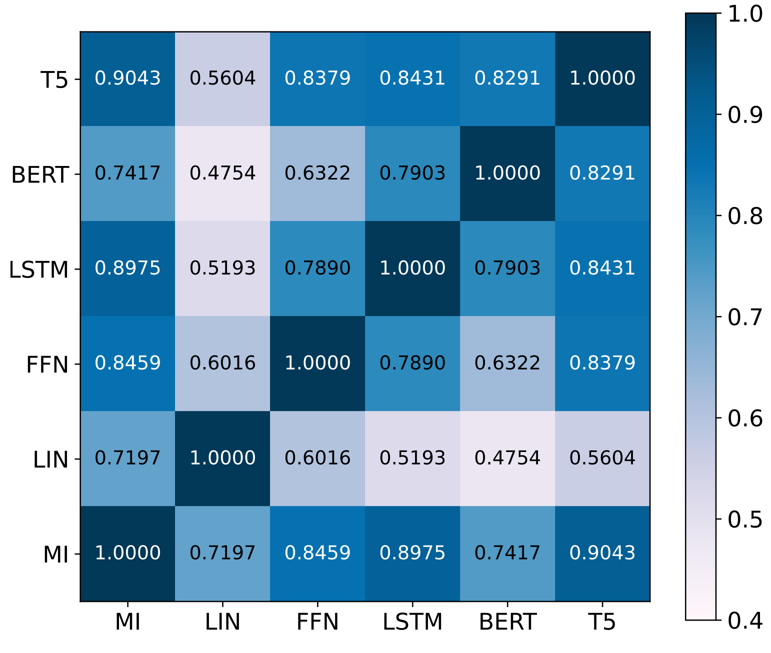
After we separately analyzed the way the models use their input features, we perform a quantitative evaluation of their similarity using the Pearson ansd Spearman Correlation Coefficient [5, 87] , which measures the linear correlation between two sets of data.

We calculated the correlation between the weights/attribution scores of each pair of models and the position-state mutual information. For each pair of models we compare the concatenation of the three sets of 17 values computed for each of the output classes, in order to measure the similarities between how to the models treat each output state.

Figure 4.14 shows the correlation matrix between all the models and the mutual information. The Pearson Correlation Coefficient confirms some of our previous observations: the models that most closely resemble the score distribution of the mutual information are the LSTM and T5, followed by the FFN, BERT and finally LIN.

The linear model LIN is the least similar to the other models, due to the large difference in magnitude between the central peak and the tails, and also the high value of the peak for the Coil class (see Figure 4.9). However, it still has a high pearson correlation with the mutual information (0.72) and the other linear model (0.60 FFN). In addition, It has high spearman correlation with the mutual information (0.783) and the other linear model (0.78 FFN).

In general, the attribution scores of all the neural networks have high correlation (over 0.63) among each other, and T5 has a very high correlation with all the other models.



	MI	LIN	FFN	LSTM	BERT	T5
T5	0.9043	0.5604	0.8379	0.8431	0.8291	1.0000
BERT	0.7417	0.4754	0.6322	0.7903	1.0000	0.8291
LSTM	0.8975	0.5193	0.7890	1.0000	0.7903	0.8431
FFN	0.8459	0.6016	1.0000	0.7890	0.6322	0.8379
LIN	0.7197	1.0000	0.6016	0.5193	0.4754	0.5604
MI	1.0000	0.7197	0.8459	0.8975	0.7417	0.9043

Figure 4.14: Pearson Correlation coefficient between the models

4.3.7 Performance and comparison of the predictive models

We compare the performance of all the classification models on both the train and test datasets.

The performance is computed using accuracy (percentage of samples with the correct predicted secondary structure), but we also report the confusion matrices on the test set for a more in-depth analysis.

	Train	**Test**
LIN	62.62%	62.09%
FFN	62.72%	62.25%
LSTM	69.88%	69.15%
BERT	70.24%	70.15%
T5	69.04%	68.35%

Table 4.1: Performance evaluation of the analyzed models for protein secondary structure prediction.

Looking at Table 4.1 we can observe that, for each model, train and test accuracy are really close. This indicates great generalization capabilities for all classifiers and no signs of overfitting.

Comparing the linear models (LIN and FFN) with the non-linear ones (LSTM, BERT, T5), we can see that the latter consistently outperform the former ones. The non-linear models surpass the linear ones of up to 8 points in train and test accuracy. In particular, BERT proves to be the most accurate model, with a test accuracy of 70.15%, while the best linear model reaches 62.25%.

Interestingly, the test accuracy of the non-linear models is inversely correlated to their Pearson Coefficient with mutual information (bottom row in Figure 4.14). For example, BERT has the highest test accuracy, but its attribution scores were the least correlated ones with the mutual information. On the other hand, T5 had the highest correlation with mutual information, but it has the worst performance on the test set. This is probably because deep neural networks gain performance by capturing complex non-linear relationships between pair of amino acids (or patterns in even higher dimensions), which leads to their attributions scores to look different from the two-variable mutual information, which only captures linear dependencies between a single amino acid and the output. As we observed in Figure 4.7 with the three-variable mutual information, the information gained from pairs of positions is not negligible, so it is reasonable for non-linear models to exploit them for more accurate predictions.

To sum it up:

- we can infer how accurate a linear model is based on how much their attribution score distribution *resembles* the mutual information. This is because the more they are similar to the mutual information, the more they are exploiting all linear dependencies between input and output.

- we can infer how accurate a non-linear model is based on how much their attribution score distribution *differs* from the mutual information. This is because the weights of the model need to deviate from that distribution to focus on higher-dimension correlations between different input positions. If the attributions of a non-linear model resemble mutual information, they are probably not making the most out of the data.

Looking at the the confusion matrices in Figure 4.15, we can see that the main difference in predictive power between linear and non-linear models comes from the classification of Extended structures.

The confusion matrices of LIN and FFN show that only 35-37% of the Extended structures are correctly identified as such (lower right corner). The non-linear models improve their identification by 20 points reaching 54.87% with T5. This shows that neural networks containing non-linearities are better-suited to encode the non-local interactions needed to identify Extended structures and distinguish them from Helices.

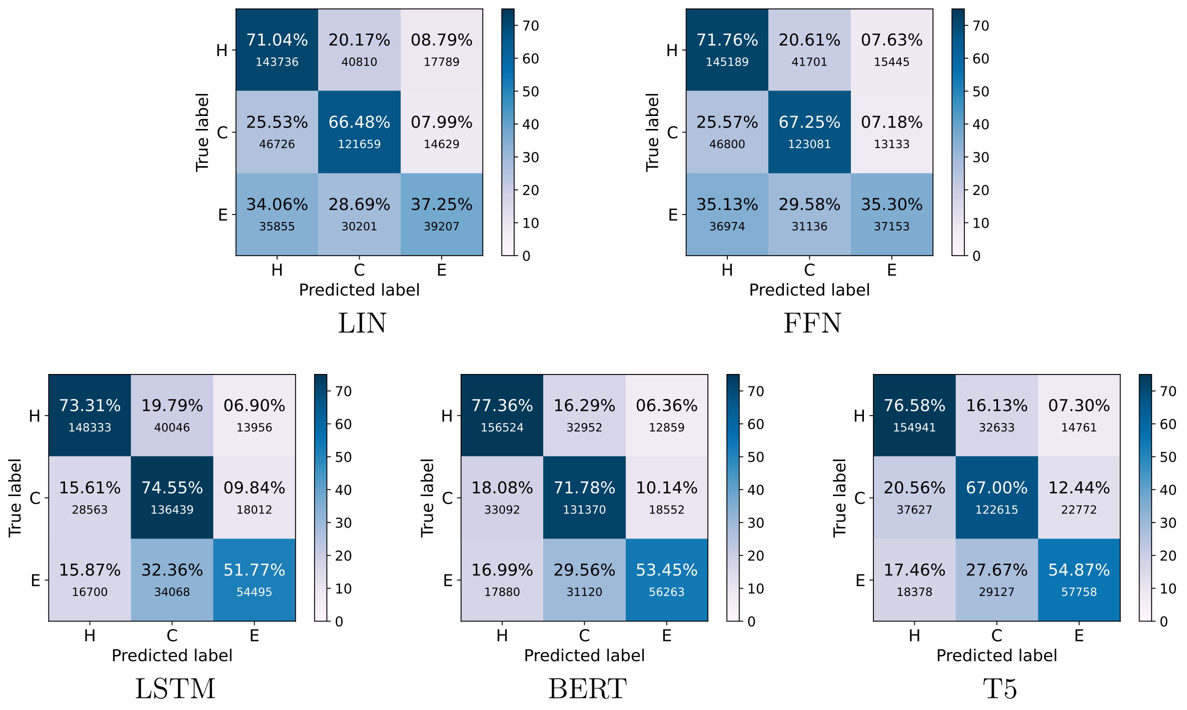


Figure 4.15: Confusion matrices of the analyzed models.

4.4 Summary

In this chapter, we introduce an analysis and comparison between T5 generative model, discriminative models and liner models for the task of predicting protein structure. We aim at understanding how the single-sequence secondary structure prediction of a residue is influenced by the residues surrounding it, in a window of 17 residues. We also investigate how different prediction methods use single-sequence information to output the predicted state.

Chapter 5

Conclusion and Future Work

The aim of the thesis was to explore the capabilities of deep generative models in a wide range of downstream tasks, generating different kinds of data, including computer vision tasks, such as visual content generation [69, 93, 137] and object recognition [49, 83, 126]; natural language processing tasks, such as keyphrase generation [81, 149, 13]; and generation tasks based on biological sequences [52, 1, 114].

During my PhD, I worked on several topics concerning generative models. The contributions presented in this thesis are based on the following two deep generative models: the GAN framework and the Transformer based architecture. I summarize the thesis contributions in the following list:

1. Building a new pipeline to solve a text to image synthesis task for unlabeled datasets by applying the CGA framework. The pipeline is able to first generate captions for unlabeled datasets, and then to use the "machine-generated" captions to train CGAN. The proposed framework introduces possible solutions to the obstacles of the text to image translation task for uncaptioned dataset, that still need to be overcome.

2. Introducing an approach to the task of present keyphrase generation in a low-resources scenario. The proposed BeGan-KP is based on the GAN framework and Transformer based architectures. It is composed of a CGAN framework combined with a novel BERT-based Discriminator model, trained by mean of the Reinforcement Learning paradigm. It has been tested on five public datasets showing performances competitive with state-of-the-art approaches while using less than 1% of the available training data, achieving a great training efficiency.

3. Proposing and analyzing five different frameworks for the task of protein structure predictions from their single sequence alignment. Several models have been analyzed, including generative models, discriminative deep neural networks and linear models. Our findings suggest that the linear model might be more suited than non-linear ones to model the single position mutual information with the secondary structure of the central residue. However, non-linear models will always perform better than linear ones in accuracy, because they can exploit more information.

Here I conclude by briefly summarizing in points the the main accomplishments of my PhD work:

- I worked on task of applying CGAN in text to image synthesis for achieving the goal of generating images conditioned on "machine-generated captions" . We introduced a pipeline composed by a captioning system and GAN. First, real images are given as input to a captioning system, which outputs multiple captions for each image. The generated captions together with images are then fed to the GAN that learns to generate images conditioned on textual descriptions. By feeding the GAN with multiple captions for each image, the GAN can better learn the correspondence between images and captions. For the experiments we used the COCO dataset and the LSUN-bedroom dataset.

- I presented a system for Keyphrase Generation using the GAN architecture with Reinforcement Learning. Thanks to the characteristics of our approach, I have been able to train the system in a data-efficient way using only a small fraction of the available data. I tested it on five baseline datasets, achieving results that are competitive with some state-of-the-art generative models. At the best of my knowledge, this is the first attempt to train such a complex architecture for the demanding task of Keyphrase Generation in an scenario in which only a small amount of data is available.

- I compared five models (a generative Transformer based model, T5, two discriminative deep learning models and two linear models) on the task protein structure prediction. The task aims to improve the analysis of the positional effect on the predicted structure state. The Generative Transformer based model T5 is a text to text Transformer that has shown competitive results in dealing with sequential data. In this work we have considered the prediction of secondary structure from a single sequence without using evolutionary information. The mutual information analysis shows that single positions and several pairs of positions can help determine the secondary structure of the central residue. The information that can be gained from single positions has a dominant role, however the effect of pairs of positions is often non-negligible. In particular, the three-variable mutual information analysis showed that clear information gains can be achieved by considering pairs of residues jointly, instead of separately.

- As I worked on structural bioinformatics, I also worked in Generalized Born (GB) radii using linear models and neural networks. The key to an efficient use of implicit solvent models is the computation of GB radii, which is accomplished by algorithms based on the electrostatics of in homogeneous dielectric media. The speed and accuracy of such computations is still an issue especially for their intensive use in classical molecular dynamics[130]. In the approach I worked on, we propose an alternative approach that encodes the physics of the phenomena and the chemical structure of the molecules in model parameters which are learned from examples. Here I presented the application of multilinear regression and neural networks model for the computation of GB radii. Overall the two models described here [75], provide an alternative to approaches based on the physics of solvation, optimized to reproduce accurately GB radii, and can thus be used for implementing fast calculation of GB radii and their derivatives and provide a useful reference for other alternative methods.

As future work, I introduce a few potential research directions. As generative models, and more generally, deep learning, is an exciting field to study in, with lots of various tasks. It would be interesting to study the application of various generative models for the fast generation of new, viable protein structures for use in protein design applications and to investigate protein disorder. Also, Protein-Protein interaction. I want to study the possibility of using generative models to generate Proteins interaction.

Bibliography

[1] Bissan Al-Lazikani, Joon Jung, Zhexin Xiang, and Barry Honig. Protein structure prediction. *Current opinion in chemical biology*, 5(1):51–56, 2001.

[2] Christian B Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

[3] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. Cvae-gan: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE international conference on computer vision*, pages 2745–2754, 2017.

[4] Alejandro Bellogín, Jun Wang, and Pablo Castells. Bridging memory-based collaborative filtering and text retrieval. *Information Retrieval*, 16(6):697–724, 2013.

[5] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[6] Gábor Berend. Opinion Expression Mining by Exploiting Keyphrase Extraction. In *IJCNLP*, 2011.

[7] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[8] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 95–104, 2016.

[9] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[10] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[11] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online, November 2020. Association for Computational Linguistics.

[12] Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. Neural Keyphrase Generation via Reinforcement Learning with Adaptive Rewards. In *ACL*, 2019.

[13] Hou Pong Chan, Wang Chen, Lu Wang, and Irwin King. Neural keyphrase generation via reinforcement learning with adaptive rewards. *arXiv preprint arXiv:1906.04106*, 2019.

[14] Jun Chen, Xiaoming Zhang, Yu Wu, Zhao Yan, and Zhoujun Li. Keyphrase Generation with Correlation Constraints. In *EMNLP*, 2018.

[15] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020.

[16] Wang Chen, Hou Pong Chan, Piji Li, Lidong Bing, and Irwin King. An Integrated Approach for Keyphrase Generation via Exploring the Power of Retrieval and Extraction. In *NAACL-HLT*, 2019.

[17] Wang Chen, Yifan Gao, Jiani Zhang, Irwin King, and Michael R. Lyu. Title-Guided Encoding for Keyphrase Generation. In *AAAI*, 2019.

[18] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

[19] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29, 2016*, pages 2172–2180, 2016.

[20] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1724–1734, 2014.

[21] P. Y. Chou and G. D. Fasman. Prediction of protein conformation. *Biochemistry*, 13:222–245, 1974.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[23] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using alaplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494, 2015.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2018.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[27] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

[28] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online, November 2020. Association for Computational Linguistics.

[29] J. Garnier, J. F. Gibrat, and B. Robson. GOR method for predicting protein secondary structure from amino acid sequence. *Methods Enzymol*, 266:540–553, 1996.

[30] J. Garnier, D. J. Osguthorpe, and B. Robson. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *J Mol Biol*, 120(1):97–120, 1978.

[31] J. F. Gibrat, J. Garnier, and B. Robson. Further developments of protein secondary structure prediction using information theory. New parameters and consideration of residue pairs. *J Mol Biol*, 198(3):425–443, 1987.

[32] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Z Ghahramani, M Welling, C Cortes, N D Lawrence, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014.

[34] Khushnuma Grover, Katinder Kaur, Kartikey Tiwari, Parteek Kumar, et al. Deep learning based question generation using t5 transformer. In *International Advanced Computing Conference*, pages 243–255. Springer, 2020.

[35] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL*, 2016.

[36] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *arXiv preprint arXiv:2007.15779*, 2020.

[37] Khaled M. Hammouda, Diego N. Matute, and Mohamed S. Kamel. CorePhrase: Keyphrase Extraction for Document Clustering. In *MLDM*, 2005.

[38] Rhys Heffernan, Kuldip Paliwal, James Lyons, Abdollah Dehzangi, Alok Sharma, Jihua Wang, Abdul Sattar, Yuedong Yang, and Yaoqi Zhou. Improving prediction of secondary structure, local backbone angles and solvent accessible surface area of proteins by iterative deep learning. *Scientific reports*, 5(1):1–11, 2015.

[39] Rhys Heffernan, Kuldip K. Paliwal, James G. Lyons, Jaswinder Singh, Yuedong Yang, and Yaoqi Zhou. Single-sequence-based prediction of protein secondary structures and solvent accessibility by deep whole-sequence learning. *J. Comput. Chem.*, 39(26):2210–2216, 2018.

[40] Rhys Heffernan, Yuedong Yang, Kuldip Paliwal, and Yaoqi Zhou. Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility. *Bioinformatics*, 33(18):2842–2849, 2017.

[41] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[42] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium–supplementary material.

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[44] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[45] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7986–7994, 2018.

[46] MD Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys (CSUR)*, 51(6):118, 2019.

[47] Anette Hulth. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *EMNLP*, 2003.

[48] Anette Hulth and Beáta Megyesi. A Study on Automatically Extracted Keywords in Text Categorization. In *ACL*, 2006.

[49] Katsushi Ikeuchi and Takeo Kanade. Automatic generation of object recognition programs. *Proceedings of the IEEE*, 76(8):1016–1035, 1988.

[50] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[51] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 5967–5976, 2017.

[52] David T Jones, William R Taylor, and Janet M Thornton. The rapid generation of mutation data matrices from protein sequences. *Bioinformatics*, 8(3):275–282, 1992.

[53] Steve Jones and Mark S. Staveley. Phrasier: A System for Interactive Document Retrieval Using Keyphrases. In *SIGIR*, 1999.

[54] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596:583–589, 2021.

[55] Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983.

[56] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. pages 3128–3137, 2015.

[57] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

[58] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. SemEval-2010 Task 5 : Automatic Keyphrase Extraction from Scientific Articles. In *Workshop on Semantic Evaluation*, 2010.

[59] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[60] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3294–3302, 2015.

[61] Krzysztof Kotowski, Tomasz Smolarczyk, Irena Roterman-Konieczna, and Katarzyna Stapor. Proteinunet—an efficient alternative to spider3-single for sequence-based prediction of protein secondary structures. *Journal of computational chemistry*, 42(1):50–59, 2021.

[62] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[63] Mikalai Krapivin, Aliaksandr Autaeu, and Maurizio Marchese. Large Dataset for Keyphrases Extraction. Technical Report DISI-09-055, University of Trento, 2009.

[64] Giuseppe Lancioni, Saida S Mohamed, Beatrice Portelli, Giuseppe Serra, and Carlo Tasso. Keyphrase generation with gans in low-resources scenarios. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 89–96, 2020.

[65] Tho Thi Ngoc Le, Minh Le Nguyen, and Akira Shimazu. Unsupervised Keyphrase Extraction: Introducing New Kinds of Words to Keyphrases. In *Advances in Artificial Intelligence*, 2016.

[66] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[67] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[68] Wenbo Li, Pengchuan Zhang, Lei Zhang, Qiuyuan Huang, Xiaodong He, Siwei Lyu, and Jianfeng Gao. Object-driven text-to-image synthesis via adversarial training. *CoRR*, abs/1902.10740, 2019.

[69] Yitong Li, Martin Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[70] Zhang Li, Sung Flood, Liu Feng, Xiang Tao, Gong Shaogang, Yang Yongxin, and Hospedales Timothy M. Actor-critic sequence training for image captioning. 2017.

[71] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference*, pages 740–755, 2014.

[72] Ilya Loshchilov and Frank Hutter. Fixing Weight Decay Regularization in Adam. *ICLR*, 2017.

[73] Yi Luan, Mari Ostendorf, and Hannaneh Hajishirzi. Scientific Information Extraction with Semi-supervised Neural Tagging. In *EMNLP*, 2017.

[74] Alexandra L'Heureux, Katarina Grolinger, and Miriam AM Capretz. Transformer-based model for electrical load forecasting. *Energies*, 15(14):4993, 2022.

[75] Saida Saad Mohamed Mahmoud, Gennaro Esposito, Giuseppe Serra, and Federico Fogolari. Generalized born radii computation using linear models and neural networks. *Bioinformatics*, 36(6):1757–1764, 2020.

[76] Elman Mansimov, Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[77] Hiroyuki Matsuda. Physical nature of higher-order mutual information: Intrinsic correlations and frustration. *Phys. Rev. E*, 62:3096–3102, 2000.

[78] Naoki Matsumura, Hiroki Tokura, Yuki Kuroda, Yasuaki Ito, and Koji Nakano. Tile art image generation using conditional generative adversarial networks. In *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, pages 209–215. IEEE, 2018.

[79] Marco Menardi, Alex Falcon, Saida S Mohamed, Lorenzo Seidenari, Giuseppe Serra, Alberto Del Bimbo, and Carlo Tasso. Text-to-image synthesis based on machine generated captions. In *Italian Research Conference on Digital Libraries*, pages 62–74. Springer, 2020.

[80] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. Deep Keyphrase Generation. In *ACL*, 2017.

[81] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. Deep keyphrase generation. *arXiv preprint arXiv:1704.06879*, 2017.

[82] Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Text. In *EMNLP*, 2004.

[83] Luca Minciullo, Fabian Manhardt, Kei Yoshikawa, Sven Meier, Federico Tombari, and Norimasa Kobori. Db-gan: Boosting object recognition under strong lighting conditions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2939–2949, 2021.

[84] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[85] Adeel Mufti, Biagio Antonelli, and Julius Monello. Conditional gans for painting generation. In *Twelfth International Conference on Machine Vision (ICMV 2019)*, volume 11433, pages 857–864. SPIE, 2020.

[86] Naila Murray. Pfagan: An aesthetics-conditional gan for generating photographic fine art. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[87] Leann Myers and Maria J Sirois. Spearman correlation coefficients, differences between. *Encyclopedia of statistical sciences*, 12, 2004.

[88] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 3510–3520, 2017.

[89] Thuy Dung Nguyen and Min-Yen Kan. Keyphrase Extraction in Scientific Publications. In *ICADL*, 2007.

[90] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[91] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

[92] S Banu Ozkan, G Albert Wu, John D Chodera, and Ken A Dill. Protein folding by zipping and assembly. *Proceedings of the National Academy of Sciences*, 104(29):11987–11992, 2007.

[93] Junting Pan, Chengyu Wang, Xu Jia, Jing Shao, Lu Sheng, Junjie Yan, and Xiaogang Wang. Video generation from single semantic label map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2019.

[94] Saichandra Pandraju and Sakthi Ganesh Mahalingam. Answer-aware question generation from tabular and textual data using t5. *International Journal of Emerging Technologies in Learning*, 16(18), 2021.

[95] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017.

[96] Linus Pauling and Robert B Corey. Configurations of polypeptide chains with favored orientations around single bonds: two new pleated sheets. *Proceedings of the National Academy of Sciences of the United States of America*, 37(11):729, 1951.

[97] Linus Pauling, Robert B Corey, and Herman R Branson. The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. *Proceedings of the National Academy of Sciences*, 37(4):205–211, 1951.

[98] Kevin W Plaxco, Kim T Simons, and David Baker. Contact order, transition state placement and the refolding rates of single domain proteins. *Journal of molecular biology*, 277(4):985–994, 1998.

[99] Gianluca Pollastri, Darisz Przybylski, Burkhard Rost, and Pierre Baldi. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins: Structure, Function, and Bioinformatics*, 47:228–235, 2002.

[100] Antti Puurula. Scalable text classification with sparse generative modeling. In *Pacific Rim International Conference on Artificial Intelligence*, pages 458–469. Springer, 2012.

[101] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. Mirrorgan: Learning text-to-image generation by redescription. *CoRR*, abs/1903.05854, 2019.

[102] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[103] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016,*, 2016.

[104] Scott E. Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 49–58, 2016.

[105] Scott E. Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *Advances in Neural Information Processing Systems 29, 2016*, pages 217–225, 2016.

[106] Scott E. Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, pages 1060–1069, 2016.

[107] Scott E. Reed, Aaron van den Oord, Nal Kalchbrenner, Victor Bapst, Matt M. Botvinick, and Nando de Freitas. Generating Interpretable Images with Controllable Structure, 2017.

[108] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 1179–1195, 2017.

[109] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-Critical Sequence Training for Image Captioning. In *CVPR*, 2017.

[110] Burkhard Rost. Review: Protein secondary structure prediction continues to rise. *J. Struct. Biol.*, 134:204–218, 2001.

[111] Burkhard Rost and Chris Sander. *Third Generation Prediction of Secondary Structures*, pages 71–95. Humana Press, Totowa, NJ, 2000.

[112] Burkhard Rost, Chris Sander, and Reinhard Schneider. Redefining the goals of protein secondary structure prediction. *Journal of molecular biology*, 235(1):13–26, 1994.

[113] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29, 2016*, pages 2226–2234, 2016.

[114] Yang Shen, Oliver Lange, Frank Delaglio, Paolo Rossi, James M Aramini, Gaohua Liu, Alexander Eletsky, Yibing Wu, Kiran K Singarapu, Alexander Lemak, et al. Consistent blind protein structure generation from nmr chemical shift data. *Proceedings of the National Academy of Sciences*, 105(12):4685–4690, 2008.

[115] Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, and Bernt Schiele. Speaking the same language: Matching machine to human captions by adversarial training. In *IEEE International Conference on Computer Vision, ICCV 2017*, pages 4155–4164, 2017.

[116] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *CoRR*, abs/1704.02685, 2017.

[117] P Sibi, S Allwyn Jones, and P Siddarth. Analysis of different activation functions using back propagation neural networks. *Journal of theoretical and applied information technology*, 47(3):1264–1268, 2013.

[118] Avinash Swaminathan, Raj Kuwar Gupta, Haimin Zhang, Debanjan Mahata, Rakesh Gosangi, and Rajiv Ratn Shah. Keyphrase Generation for Scientific Articles using GANs. In *AAAI*, 2019.

[119] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[120] Takashi Tomokiyo and Matthew Hurst. A language model approach to keyphrase extraction. In *ACL workshop on Multiword expressions*, 2003.

[121] Mirko Torrisi, Gianluca Pollastri, and Quan Le. Deep learning methods in protein structure prediction. *Computational and Structural Biotechnology Journal*, 18:1301–1310, 2020.

[122] Wouter G. Touw, Coos Baakman, Jon Black, Tim A. H. te Beek, E. Krieger, Robbie P. Joosten, and Gert Vriend. A series of PDB-related databanks for everyday needs. *Nucleic Acids Research*, 43:D364–D368, 2014.

[123] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *SSW*, 125:2, 2016.

[124] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[125] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pages 3156–3164, 2015.

[126] Bingxu Wang, Jinhui Lan, and Jiangjiang Gao. Multi-class gan for generating multi-class images in object recognition. *JOSA A*, 39(5):897–906, 2022.

[127] G. Wang and R. L. Dunbrack. PISCES: a protein sequence culling server. *Bioinformatics*, 19:1589–1591, 2003.

[128] Minmei Wang, Bo Zhao, and Yihua Huang. PTR: Phrase-Based Topical Ranking for Automatic Keyphrase Extraction in Scientific Publications. In *ICONIP*, 2016.

[129] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[130] Edmund Taylor Whittaker. *A history of the theories of aether and electricity: from the age of Descartes to the close of the nineteenth century.* Longmans, Green and Co., London, UK, 1910.

[131] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 5 1992.

[132] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 1992.

[133] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. KEA: Practical Automatic Keyphrase Extraction. In *ACM*, 1999.

[134] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv:abs/1910.03771*, 2019.

[135] Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3940–3949, 2018.

[136] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 1316–1324, 2018.

[137] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *European conference on computer vision*, pages 776–791. Springer, 2016.

[138] Yuedong Yang, Jianzhao Gao, Jihua Wang, Rhys Heffernan, Jack Hanson, Kuldip K. Paliwal, and Yaoqi Zhou. Sixty-five years of the long march in protein secondary structure prediction: the final stretch? *Briefings Bioinform.*, 19(3):482–494, 2018.

[139] Hai Ye and Lu Wang. Semi-Supervised Learning for Neural Keyphrase Generation. In *EMNLP*, 2018.

[140] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[141] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. {LSUN:} Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *CoRR*, abs/1506.0, 2015.

[142] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*, 2016.

[143] Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Daqing He, and Adam Trischler. Generating Diverse Numbers of Diverse Keyphrases. *ArXiv:abs/1810.05241*, 2018.

[144] Bowen Zhang, Shuyang Gu, Bo Zhang, Jianmin Bao, Dong Chen, Fang Wen, Yong Wang, and Baining Guo. Styleswin: Transformer-based gan for high-resolution image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11304–11314, 2022.

[145] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[146] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[147] Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter. In *EMNLP*, 2016.

[148] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *International Conference on Machine Learning*, pages 4006–4015. PMLR, 2017.

[149] Yong Zhang and Weidong Xiao. Keyphrase generation based on deep seq2seq model. *IEEE Access*, 6:46047–46057, 2018.

[150] Yongzheng Zhang, A. Nur Zincir-Heywood, and Evangelos E. Milios. World Wide Web site summarization. *Web Intelligence and Agent Systems*, 2004.

[151] Zizhao Zhang, Yuanpu Xie, and Lin Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, pages 6199–6208, 2018.

[152] Ren Zhou, Wang Xiaoyu, Zhang Ning, Lv Xutao, and Li Li-Jia. Deep Reinforcement Learning-based Image Captioning with Embedding Reward. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, CVPR'17, pages 1151—1159, 2017.

[153] Yaoqi Zhou and Martin Karplus. Interpreting the folding kinetics of helical proteins. *Nature*, 401(6751):400–403, 1999.

[154] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pages 597–613. Springer, 2016.

[155] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Computer Vision - ECCV 2016 - 14th European Conference*, pages 597–613, 2016.