**CHAPTER 1**

**INTRODUCTION**

* 1. **INTRODUCTION**

**1.1.1 DEFINITION OF TERMINOLOGIES USED**

**1.Text-to-Image Synthesis:**

This task involves generating realistic images from textual descriptions. It's a form of artificial intelligence where models learn to understand and visualize text, bridging the gap between language and visual representation.

**2.TensorFlow/Keras:**

TensorFlow is a machine learning library by Google, and Keras is an API integrated into TensorFlow for building neural networks. Keras simplifies model development with high-level abstractions and facilitates rapid prototyping and experimentation.

**3.LSTM (Long Short-Term Memory):**

LSTM is a type of recurrent neural network (RNN) architecture designed to process sequences of data, particularly useful for modeling sequences with long-range dependencies. LSTMs are capable of learning and remembering over long sequences, making them suitable for tasks involving sequential data like text processing.

**4.Tokenization:**

Tokenization is the process of breaking down text into smaller units, such as words or subwords. It converts textual data into a format suitable for processing by machine learning models, enabling efficient text analysis and understanding.

**5.Embedding:**

Embedding involves representing categorical variables, like words in natural language, as continuous vectors in a lower-dimensional space. These embeddings capture semantic relationships between words, enhancing the model's ability to understand language.

**6.Tokenizer:**

A Tokenizer is a tool used to tokenize text, converting it into sequences of integers or tokens. It's a preprocessing step in natural language processing tasks, enabling the transformation of raw text data into a format suitable for neural networks.

**7.Padding:**

Padding is the addition of special tokens or zeros to sequences to ensure uniform length. It's commonly used in sequence-based tasks, such as text processing, to standardize input data dimensions for neural network models.

**8.Model Architecture:**

Model architecture refers to the structure of a neural network, including the arrangement of layers and connections. It defines how data flows through the network and determines the model's ability to learn and generalize from input data.

**9.Loss Function:**

The loss function quantifies the difference between predicted and actual outputs of a model. It's a measure of the model's performance during training, guiding parameter updates to minimize errors and improve predictive accuracy**.**

**10.Optimizer:**

An optimizer is an algorithm used to update the parameters of a neural network during training. It adjusts the model's weights and biases based on gradients computed from the loss function, facilitating convergence towards optimal solutions.

**11.Inference:**

Inference is the process of using a trained model to make predictions on new data. It involves passing input data through the model to generate output predictions, enabling the model to generalize its learned patterns to unseen instances.

**12.Pretrained Model:**

A pretrained model is a model that has been trained on a large dataset and saved. It serves as a starting point for transfer learning or fine-tuning on specific tasks, leveraging learned features to improve performance on related tasks.

**13.GPU (Graphics Processing Unit):**

A GPU is a specialized hardware component designed to accelerate graphics rendering. In deep learning, GPUs are utilized for their parallel processing capabilities, significantly speeding up training and inference tasks compared to traditional CPUs.

**1.1.2. Text encoders**

The encoder in machine learning is a network that takes the input sequence and outputs the feature representations of it. A piece of text consists of a list of words, numbers, special characters and punctuation marks. The text encoder is a block used to encode the raw text into its numerical features.

**1.1.2.a) Text tokenization**

The first step of encoding is text tokenization that transforms the text into tokens. Depending on the requirements of tasks, a tokenizer can be set to filter out the punctuation (or not), unify the letters into lower-cases (or not), etc. Then, the sentences will be split into a bunch of tokens. Sometimes, it is required to have same sequence length for each text. In this case, the sequence of tokens will be either padded with a predefined padding token at the end of sequence or partly removed.

**1.1.2.b) Word embeddings**

The second step is to map each token into its corresponding word embedding. Word embedding methods represent raw words (tokens) as continuous vectors, which can capture lexical and semantic properties of words. Benefiting from using dense and low-dimensional vectors as opposed to symbol representations and one-hot encoding, etc., word embeddings are more efficient in computation and more powerful in generalization. There are multiple ways to learn word embeddings from raw text.

One is using an embedding layer, which is actually a weight matrix. One dimension of it is the size of the embedding space, while another dimension is the number of tokens in the corpus. It is initialized with small random numbers and can be learned jointly with a neural network in a supervised way using backpropagation for a specific task like text classification. By this means, it may require a large amount of data for training and thus can be slow, but it is able to learn embeddings targeted to the specific text and the specific task simultaneously. Besides, various algorithms are proposed to learn word embeddings more efficiently. Word2Vec is a prediction-based supervised model developed by Tomas Mikolov, et al [10]. It is alterable by choosing one of the following components: the Continuous Bag-of-Words model (CBOW) and the SkipGram model. CBOW learns the embeddings by predicting target word (e.g. ’mat’) based on context words (’the cat sits on the’).

On the contrary, the skipgram model predicts context words given the target word. A context window (a window of neighboring words) will shift through the text and each shift is a training example. Since this approach is computationally-efficient, it facilitates the learning of larger-dimension word embeddings with much larger text dataset. Nevertheless, the usage of context window restricts the learning from the global statistics of the text.

The unsupervised learning algorithm Global Vectors (GloVe) constructs a global word-word co-occurrence matrix to capture the global statistics using matrix factorization techniques and also leverages the predition-based method in Word2Vec, resulting in generally better word embeddings. Apart from using these techniques to learn word embeddings from scratch, their pre-trained embeddings are available for reusing in other language-related tasks in static or updated way to possibly improve the performance and reduce training time. Finally, text embeddings can be constructed based on word embeddings for phrases, sentences or paragraphs. This can be done by simple operations on vectors and matrices, such as unweighted averaging. Some methods incorporate recurrent neural networks (RNNs) and/or convolutional neural neworks (CNNs) to model the sentences, etc. These two types of networks will be briefly introduced in the next two subsections.

**1.1.3 Recurrent Neural Networks (RNNs):**

Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed to process sequential data by maintaining a hidden state or memory vector. Unlike feedforward neural networks that process each input independently, RNNs utilize feedback loops, allowing them to capture temporal dependencies and patterns in sequential data. This recurrent structure enables RNNs to handle inputs of variable length and model time-dependent relationships.

The key feature of RNNs is their ability to maintain a hidden state that evolves over time as the network processes each element of a sequence. At each time step, the hidden state is updated based on the current input and the previous hidden state, allowing the network to retain information about past inputs. This hidden state serves as a form of memory, enabling the network to capture long-range dependencies and contextual information in sequential data.

However, traditional RNNs have limitations, such as difficulties in capturing long-term dependencies and the vanishing gradient problem during training. To address these issues, more advanced variants of RNNs, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), have been developed. These architectures incorporate specialized mechanisms, such as memory cells and gating units, to better capture long-range dependencies and mitigate gradient-related issues.

**1.1.3.a) Utilization of RNN** **in Fashion Design**

In the context of text-to-image synthesis, RNNs are likely used to process textual descriptions or prompts. Specifically, RNNs may be employed as part of the text encoder to convert raw text into numerical representations or embeddings. This involves tokenizing the text, mapping tokens to word embeddings, and encoding the sequence of embeddings using an RNN architecture.

The RNN component of the text encoder allows the model to capture sequential dependencies and contextual information in the textual descriptions. By maintaining a hidden state that evolves over the course of processing the text, the RNN can effectively capture the semantics and nuances of the input text, enabling the model to generate accurate and coherent images based on the textual prompts.

**1.1.4 Convolutional neural networks**

Convolutional neural networks (CNNs), especially deep convolutional networks have been widely used in the field of computer vision for image classification, object detection, instance segmentation, etc. CNNs can take of shape information in the input image by applying a series of filters to the raw pixels to extract and learn higher-level features. The hidden layers of a CNN typically consist of convolutional layers, non-linear layers (i.e. activation functions), pooling layers, fully connected layers and normalization layers.

Convolutional Neural Networks (CNNs) serve as fundamental tools for processing and interpreting visual data based on textual prompts.

**1.1.4.a) Feature Extraction**:

CNNs play a pivotal role in extracting pertinent visual features from input images. These features encapsulate essential visual patterns, shapes, and textures that are instrumental in generating images that align coherently with the provided textual descriptions. Through the process of feature extraction, CNNs analyze the inherent structure and content of the images, enabling them to discern relevant details that contribute to the synthesis process.

**1.1.4.b) Hierarchical Representation Learning**:

CNNs leverage a hierarchical architecture composed of multiple layers, each serving a distinct purpose in the feature extraction process. Beginning with low-level features such as edges and textures, CNNs progressively learn more abstract and complex representations as information propagates through successive layers. This hierarchical learning enables CNNs to capture both fine-grained details and high-level concepts present in the input images, facilitating the generation of visually compelling and contextually relevant images.

**1.1.4.c) Convolutional Filtering:**

At the heart of CNNs are convolutional layers, which apply a series of filters or kernels to the input images. These filters convolve across the image pixels, systematically analyzing local patterns and features within the image. Each filter specializes in detecting specific visual attributes, such as edges, corners, or textures, by convolving across the entire image and extracting relevant information that contributes to the overall feature representation.

**1.1.4.d) Non-linear Activation:**

Activation functions, such as the Rectified Linear Unit (ReLU), introduce non-linearities into the network, enabling CNNs to model complex relationships and capture intricate visual details. By applying non-linear transformations to the output of convolutional layers, activation functions ensure that the network can learn from non-linear data distributions, enhancing its capacity to extract meaningful features from the input images.

**1.1.4.e) Pooling and Downsampling**:

Pooling layers in CNNs play a critical role in downsampling the feature maps generated by convolutional layers. Max pooling or average pooling operations are commonly employed to reduce the spatial dimensions of the feature maps while preserving important information. This process enhances computational efficiency and reduces overfitting by condensing the feature representations and focusing on the most salient features present in the input images.

**1.1.4.f) Semantic Representation**:

Through the collective operations of convolution, activation, and pooling, CNNs acquire semantic representations of the input images. These representations encapsulate meaningful visual features and characteristics that are relevant to the provided textual descriptions. By learning to extract and encode semantic information from the input images, CNNs facilitate the synthesis of coherent and contextually relevant images that accurately reflect the underlying textual prompts.

**1.2 OBJECTIVE**

The demand for fashion products, including apparel and shoes, is on the rise as people increasingly appreciate beauty. With this growing demand, the fashion industry faces the challenge of automating the design process. Artificial Intelligence (AI) emerges as a pivotal technological solution in revolutionizing this industry. This project proposes leveraging advanced AI methods, particularly our chosen methods, to automate the fashion design process.

The objective of this project is to develop a neural network generative model capable of producing high-quality images of fashion items based on textual descriptions. This model will be conditioned on various aspects of the fashion product, such as category, style, color, and material. The system aims to provide users with the ability to input descriptive text and generate corresponding images of fashion items, thereby automating and streamlining the fashion design process. Additionally, the project seeks to assess the limitations and performance of existing state-of-the-art methods in text-to-image synthesis for fashion design and devise new learning frameworks, network architectures, and training data to enhance the quantitative and qualitative performance of the proposed model.

Top of Form

**1.3 SYSTEM SCOPE**

**Scope of the proposed system**

* Conditioning the model on various aspects of fashion products, such as category, style, color , etc.
* Enabling users to input descriptive text to prompt the generation of corresponding fashion item images.
* Incorporating a low-dimensional latent variable for user-controlled exploration of different design variations.
* Continuous improvement and innovation to ensure relevance and impact in the fashion technology landscape.

**1.4 PROJECT DESCRIPTION**

**CHAPTER 2**

**SYSTEM ANALYSIS**