ISSN: 2350-0328

International Journal of AdvancedResearch in Science, Engineering and Technology

Vol. 5, Issue 9 , September 2018

Text Based Synthesis and Generation of Images using Deep Learning

Arun Vignesh.M, Aarthe Jayaprakash, Krishnap Priya.G.B, Dr.R.S.Milton

Dept. of Computer Science and Engineering, SSN College of Engineering, Chennai, India.

ABSTRACT: There is a need to encourage children to learn the art of storytelling. Storytelling unleashes the creativity in young minds. What would be more interesting, would be to help them bring their figments of imagination to life. There is a human condition called synesthesia where a simulation of one cognitive pathway leads to experiences in a second cognitive pathway. The word literally means joined perception. There is something really intriguing about this idea of being able to express a thought or an idea or a concept in two entirely different mediums. Generative adversarial networks have demonstrated that it is possible to create fake but photorealistic looking images after training on an image data. Generative adversarial networks, specifically Deep Convolutional Generative Adversarial Networks, are used to create synthetic images from the given text sentence, such that for even the wildest story, an illustration will be possible. The implications of this ability to synthesize one data type to another will eventually allow us to create all sorts of entertainment with just detailed descriptions. The designers, engineers and scientists could all hash out theoretical concepts with real-time visual feedback on their thought processes. This project aims at synthesis of images from the given text using Deep Convolutional Generative Adversarial Networks.

KEYWORDS: Natural Language Processing, Deep Learning, TensorFlow, Neural Networks, Character recognition, Convolutional Neural Networks.

I.INTRODUCTION

From a given piece of text, we want to generate an appropriate and relevant image in two ways: 1) retrieval of image from a database, and 2) synthesis of image using GAN.

Retrieval of image from a database involves two datasets, one dataset containing images and another containing relevant labels descriptions. A model is trained to plot words from the labels into vector space based on the textual similarity of the words. Words with closeness in meaning appear closer together in the vector space, whereas those with unrelated meanings occur farther apart. While testing, any new description will in turn retrieve an appropriate image from the database. This is done by comparing the textual similarity of the test label with that of every training label. The image associated with with the most similar label is retrieved. Comparisons involved in this process are inefficient as it is time consuming and requires intensive processing. Also, given a description which does not significantly match any training label, an irrelevant image will be retrieved. A method of retrieving a relevant image for almost any input would be more useful. This can be achieved by using Generative Adversarial Networks.

Automatic synthesis of realistic images from text would be interesting and useful, but current AI systems are still far from this goal. However, in recent years generic and powerful recurrent neural network architectures have been developed to learn discriminative text feature representations. Meanwhile, Deep Convolutional Generative Adversarial Networks (DCGANs) have begun to generate highly compelling images of specific categories, such as faces, album covers, and room interiors. In this work, we develop a novel deep architecture and GAN formulation to effectively bridge these advances in text and image modeling, translating visual concepts from text to pixels. We demonstrate the capability of our model to generate plausible images flowers from detailed text descriptions.

II. SIGNIFICANCE OF THE SYSTEM

A. Gensim Model

Gensim [1] is a pure Python library that has two fronts: 1) digital document indexing and similarity search 2) fast, memory-efficient, scalable algorithms for Singular Value Decomposition and Latent Dirichlet Allocation. The connection between the two is unsupervised, semantic analysis of plain text in digital collections.

ISSN: 2350-0328
International Journal of AdvancedResearch in Science, Engineering and Technology
Vol. 5, Issue 9 , September 2018

Gensim was created for large digital libraries, but its underlying algorithms for large-scale, distributed, online SVD and LDA are useful on their own, outside of the domain of Natural Language Processing.

Advantages:

Gensim is fast because of its design of data access and implementation of numerical processing. Gensim processes data in a streaming fashion with memory independence – there is no need for the whole training corpus to reside fully in RAM at any one time so it can process large, web-scale corpora. Efficient implementations: Gensim algorithms process numerical processing in separate Numpy code which handle linear algebra in highly optimized routines in Fortran from the system's BLAS and LAPACK libraries.

Gensim is very well optimized, but also highly specialized, library for doing jobs in the periphery of "WORD2VEC". It offers an easy, surpringly well working and swift AI-approach to unstructured raw texts, based on a shallow neural network. Having deep learning methods available in Python allows the utilization of a multitude of NLP tools available in Python.

Disadvantages:

Although gensim is based on shallow neural network, in order to understand productions, or in getting deeper insights into neural networks, using TensorFlow, which offers a mathematically more generalized model, yet to be paid by some 'unpolished' performance and scalability can be used.

B. Similarity Measures for Text Document Clustering

Clustering [2] is a useful technique that organizes a large quantity of unordered text documents into a small number of meaningful and coherent clusters, thereby providing a basis for intuitive and informative navigation and browsing mechanisms. Partitional clustering algorithms have been recognized to be more suitable as opposed to the hierarchical clustering schemes for processing large datasets. Anna Huang performs an experiment to identify which method of computing similarity of text is the best. Different similarity measures are listed such as: Euclidean distance, Cosine Similarity, Jaccard Coefficient and Pearson Correlation Coefficient.

Advantages:

On average, the Jaccard and Cosine measures are slightly better in generating more coherent clusters, which means the clusters have higher purity scores.

Disadvantages:

The impact of using different document representations on clustering performance is unknown. The combination of the different representations with similarity measures can help give a better picture.

C. Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean [3] studied the quality of vector representations of words derived by various models on a collection of syntactic and semantic language tasks. It was observed that it is possible to train high quality word vectors using very simple model architectures, compared to the popular neural network models (both feedforward and recurrent).

Advantages:

Because of the much lower computational complexity, it is possible to compute very accurate high dimensional word vectors from a much larger data set. Using the DistBelief distributed framework, it should be possible to train the CBOW and Skip-gram models even on corpora with one trillion words, for basically unlimited size of the vocabulary. That is several orders of magnitude larger than the best previously published results for similar models.

ISSN: 2350-0328
International Journal of AdvancedResearch in Science, Engineering and Technology
Vol. 5, Issue 9 , September 2018

Disadvantages:

Long documents are poorly represented because they have poor similarity values (a small scalar product and a large dimensionality). Search keywords must precisely match document terms; word substrings might result in a "false positive match". Semantic sensitivity; documents with similar context but different term vocabulary won't be associated, resulting in a "false negative match". The order in which the terms appear in the document is lost in the vector space representation.
Theoretically assumed terms are statistically independent. Weighting is intuitive but not very formal. Many of these difficulties can, however, be overcome by the integration of various tools, including mathematical techniques such as singular value decomposition and lexical databases such as WordNet.

D. Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval

A model [4] is created that addresses sentence embedding, a hot topic in current natural language processing research, using Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) cells. The LSTM-RNN model sequentially takes each word in a sentence, extracts its information, and embeds it into a semantic vector. Due to its ability to capture long term memory, the LSTM-RNN accumulates increasingly richer information as it goes through the sentence, and when it reaches the last word, the hidden layer of the network provides a semantic representation of the whole sentence. By performing a detailed analysis on the LSTM model, inferences were made:1) The model is robust to noise, i.e., it mainly embeds keywords in the final semantic vector representing the whole sentence and 2) in the model, each cell is usually allocated to keywords from a specific topic. These findings have been supported using extensive examples. As a concrete sample application of this sentence embedding method, evaluation was performed on the important language processing task of web document retrieval where the method outperformed all existing state of the art methods significantly.

Advantages:

LSTM improves the RNN neuron. The range of contextual information is limited in RNN and the Back-Propagation through time does not work properly. This is noticeable in either vanishing or exploding outputs of the network. In literature, this problem is called vanishing gradient problem or exploding gradient. When the network is learning to bridge long time lags, it takes a huge amount of time or does not work at all, because of the vanishing gradient problem. The exploding gradient leads to oscillating weights, which also reduces the quality of the network. In practice, this means when a recurrent network is learning to store information over extended time intervals, it takes a very long time due to insufficient decaying error back flow. The LSTM is designed to overcome the error back flow problems through carousels in their special units. This is all done with still a low computational complexity of O(1) and additionaly the LSTM impoves the RNN with the ability to bridge time intervals.

Disadvantages:

LSTM is slower than other normal activation functions, such as sigmoid, tanh or rectified linear unit. Recently, there has been a work showing that RNN with rectified linear unit with careful initialization of parameters could achieve competitive performances with RNN LSTM on some tasks, including speech. There is a difficulty in training LSTM models. A lot of time and system resources go into training even a simple model.

E. Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks

The authors of the DCGAN focused on improving the architecture of the original vanilla GAN [5]. They found out that:
(1) Batch normalization is a must in both networks.
(2) Fully hidden connected layers are not a good idea.
(3) Avoid pooling, simply stride your convolutions.
(4) ReLU activations are your friend (almost always).

ISSN: 2350-0328
International Journal of AdvancedResearch in Science, Engineering and Technology
Vol. 5, Issue 9 , September 2018

Advantages:

Vanilla GANs could work on simple datasets, but DCGANs are far better.
DCGANs are stable to train. They generate higher quality samples. They are very useful to learn unsupervised image representations. They are used as a baseline to implement other GANs.

Disadvantages:

The quality of samples still need to be further refined. This can be done using a variety of enhancement techniques during training. Training a GAN requires finding a Nash equilibrium of a game. Sometimes gradient descent does this, sometimes it doesn't. It is difficult to find a good equilibrium finding algorithm, so GAN training is unstable compared to VAE or PixelRNN training. Compared to Boltzmann machines, it's hard to guess the value of one pixel given another pixel. GANs are trained to generate all the pixels in one shot.

III. ALGORITHMS FOR TEXT TO IMAGE GENERATION

The various algorithms for text to image generation depends on whether it is static image retrieval or synthetic image synthesis. In this work, we are interested in translating text in the form of single-sentence human-written descriptions directly into image pixels. However, deep convolutional and recurrent networks for text have yielded highly discriminative and generalizable (in the zero-shot learning sense) text representations learned automatically from words and characters. These approaches exceed the previous state of-the-art using attributes for zero-shot visual recognition.

A. Word Embedding

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. They are a distributed representation for text that is perhaps one of the key breakthroughs for the impressive performance of deep learning methods on challenging natural language processing problems. It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems. One of the benefits of using dense and low-dimensional vectors is computational: the majority of neural network tool kits do not play well with very high-

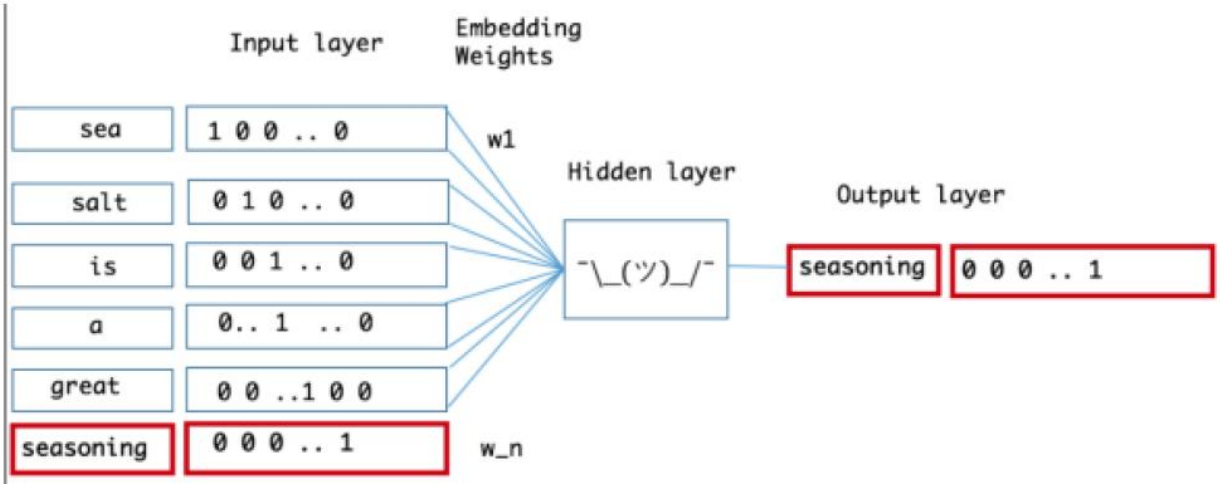

Figure 3.1: Word Embedding

dimensional, sparse vectors. The main benefit of the dense representations is generalization power: if we believe some features may provide similar clues, it is worthwhile to provide a representation that is able to capture these similarities. Word embedding methods learn a real-valued vector representation for a predefined fixed sized vocabulary from a

ISSN: 2350-0328
International Journal of AdvancedResearch in Science, Engineering and Technology
Vol. 5, Issue 9 , September 2018

corpus of text. The learning process is either joint with the neural network model on some task, such as document classification, or is an unsupervised process, using document statistics. There are three techniques that can be used to learn a word embedding from text data.

B. GloVe

The Global Vectors for Word Representation, or GloVe, algorithm is an extension to the word2vec method for efficiently learning word vectors, developed by Pennington, et al. at Stanford. Classical vector space model representations of words were developed using matrix factorization techniques such as Latent Semantic Analysis (LSA) that do a good job of using global text statistics but are not as good as the learned methods like word2vec at capturing meaning and demonstrating it on tasks like calculating analogies (e.g. the King and Queen example above). GloVe is an approach to marry both the global statistics of matrix factorization techniques like LSA with the local context-based learning in word2vec. Rather than using a window to define local context, GloVe constructs an explicit word context or word co-occurrence matrix using statistics across the whole text corpus. The result is a learning model that may result in generally better word embeddings. GloVe is a new global log-bilinear regression model for the unsupervised learning of word representations that outperforms other models on word analogy, word similarity, and named entity recognition tasks.

C. Gensim

Gensim was developed and is maintained by the Czech natural language processing researcher Radim Rehurek and his company RaRe Technologies. It is not an everything- including- the- kitchen- sink NLP research library (like NLTK); instead, Gensim is a mature, focused, and efficient suite of NLP tools for topic modeling. It supports an implementation of the Word2Vec word embedding for learning new word vectors from text.

It also provides tools for loading pre-trained word embeddings in a few formats and for making use and querying a loaded embedding.

class gensim.models.word2vec.Word2Vec()

Bases: gensim.models.baseany2vec.BaseWordEmbeddingsModel

The model can be stored/loaded via its save() and load() methods, or stored/loaded in a format compatible with the original word2vec implementation. Initialize the model from an iterable of sentences. Each sentence is a list of words (unicode strings) that will be used for training. Parameters: sentences (iterable of iterables) – The sentences iterable can be simply a list of lists of tokens, but for larger corpora, consider an iterable that streams the sentences directly from disk/network. See BrownCorpus, Text8Corpus or LineSentence in word2vec module for such examples. If you don't supply sentences, the model is left uninitialized – use if you plan to initialize it in some other way. sg (int 1, 0) – Defines the training algorithm. If 1, skip-gram is employed; otherwise, CBOW is used. Size (int) – Dimensionality of the feature vectors. window (int) – The maximum distance between the current and predicted word within a sentence.

Alpha (float) – The initial learning rate. min alpha (float) – Learning rate will linearly drop to min alpha as training progresses. seed (int) – Seed for the random number generator. Initial vectors for each word are seeded with a hash of the concatenation of word + str(seed). Note that for a fully deterministically-reproducible run, you must also limit the model to a single worker thread (workers=1), to eliminate ordering jitter from OS thread scheduling. (In Python 3, reproducibility between interpreter launches also requires use of the PYTHONHASHSEED environment variable to control hash randomization). min count (int) – Ignores all words with total frequency lower than this. max vocab size (int) – Limits the RAM during vocabulary building; if there are more unique words than this, then prune the infrequent ones.

Every 10 million word types need about 1GB of RAM. Set to None for no limit. Sample (float) – The threshold for configuring which higher-frequency words are randomly down sampled, useful range is (0, 1e-5). workers (int) – Use these many worker threads to train the model (=faster training with multicore machines). hs (int 1,0) – If 1, hierarchical softmax will be used for model training. If set to 0, and negative is non-zero, negative sampling will be used. negative (int) – If > 0, negative sampling will be used, the int for negative specifies how many "noise words" should be drawn (usually between 5-20). If set to 0, no negative sampling is used. cbow mean (int 1,0) – If 0, use the sum of the context word vectors. If 1, use the mean, only applies when cbow is used. hashfxn (function) – Hash function to use to

ISSN: 2350-0328

International Journal of AdvancedResearch in Science, Engineering and Technology

Vol. 5, Issue 9 , September 2018

randomly initialize weights, for increased training reproducibility. iter (int) – Number of iterations (epochs) over the corpus. trim rule (function) – Vocabulary trimming rule, specifies whether certain words should remain in the vocabulary, be trimmed away, or handled using the default (discard if word count < min count). Can be None (min count will be used, look to keep vocab item()), or a callable that accepts parameters (word, count, min count) and returns either gensim.utils.RULE DISCARD, gensim.utils.RULE KEEP or gensim.utils.RULE DEFAULT. Note: The rule, if given, is only used to prune vocabulary during build vocab() and is not stored as part of the model. sorted vocab (int 1,0) – If 1, sort the vocabulary by descending frequency before assigning word indexes. batch words (int) – Target size (in words) for batches of examples passed to worker threads (and thus cython routines).(Larger batches will be passed if individual texts are longer than 10000 words, but the standard cython code truncates to that maximum.) compute loss (bool) – If True, computes and stores loss value which can be retrieved using model.get latest training loss(). callbacks – List of callbacks that need to be executed/run at specific stages during training.

D. PRINCIPAL COMPONENT ANALYSIS FOR PLOTTING WORD VECTORS

The sheer size of data in the modern age is not only a challenge for computer hardware but also a main bottleneck for the performance of many machine learning algorithms. The main goal of a PCA analysis is to identify patterns in data; PCA aims to detect the correlation between variables. If a strong correlation between variables exists, the attempt to reduce the dimensionality only makes sense. In a nutshell, this is what PCA is all about: Finding the directions of maximum variance in high-dimensional data and project it onto a smaller dimensional
subspace while retaining most of the information.

- Standardize the data.
- Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix, or perform Singular Vector Decomposition.
- Sort eigenvalues in descending order and choose the k eigenvectors that correspond to the k largest eigenvalues where k is the number of dimensions of the new feature subspace.
- Construct the projection matrix W from the selected k eigenvectors.
- Transform the original dataset X via W to obtain a k-dimensional feature subspace Y.

E. Deep Convolutional Generative Adversarial Networks

To build a DCGAN, we create two deep neural networks. Then we make them fight against each other, endlessly attempting to out-do one another. In the process, they both become stronger. Let's pretend that the first deep neural network is a brand new police officer who is being trained to spot counterfeit money. It's job is to look at a picture and tell us if the picture contains real money. Since we are looking for objects in pictures, we can use a standard Convolutional Neural Network for this job. But the basic idea is that the neural network that takes in an image, processes it through several layers that recognize increasingly complex features in the image and then it outputs a single value—in this case, whether or not the image contains a picture of real money.

This first neural network is called the Discriminator: Now let's pretend the second neural network is a brand new counterfeiter who is just learning how to create fake money. For this second neural network, we'll reverse the layers in a normal ConvNet so that everything runs backwards. So instead of taking in a picture and outputting a value, it takes in a list of values and outputs a picture. This second neural network is called the Generator: So now we have a police officer (the Discriminator) looking for fake money and a counterfeiter (the Generator) that's printing fake money. Let's make them battle!

ISSN: 2350-0328
International Journal of AdvancedResearch in Science, Engineering and Technology
Vol. 5, Issue 9 , September 2018

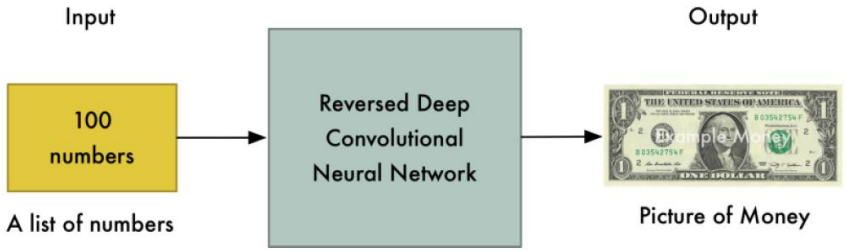

Figure : Generator

In the first round, the Generator will create pathetic forgeries that barely resemble money at all because it knows absolutely nothing about what money is supposed to look like:

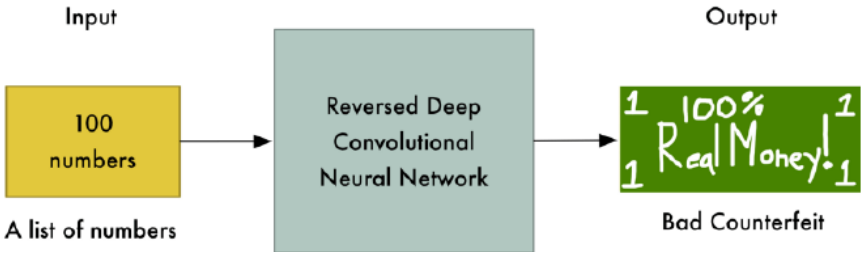

Figure: Generator creates a bad counterfeit

But right now the Discriminator is equally terrible at it's job of recognizing money, so it won't know the difference:

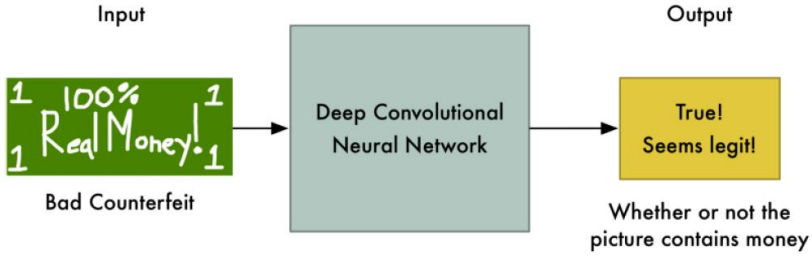

Figure : Discriminator is unable to distinguish the counterfeit and recognizes it as real.

At this point, we step in and tell the Discriminator that this dollar bill is actually fake. Then we show it a real dollar bill and ask it how it looks different from the fake one. The Discriminator looks for a new detail to help it separate the real one from the fake one. For example, the Discriminator might notice that real money has a picture of a person on it and the fake money doesn't. Using this knowledge, the Discriminator learns how to tell the fake from the real one. It gets a tiny bit better at its job:

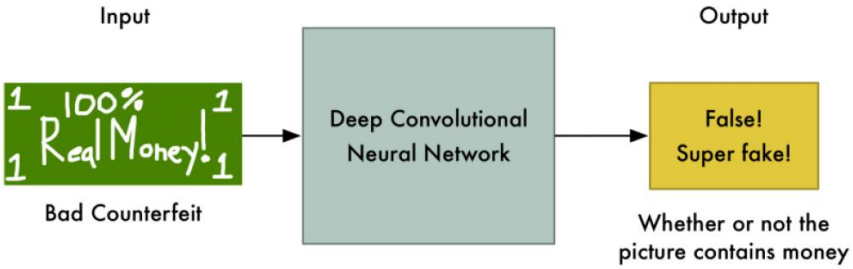

Figure: Generator takes n feedback to create more realistic images

ISSN: 2350-0328
International Journal of AdvancedResearch in Science, Engineering and Technology
Vol. 5, Issue 9 , September 2018

Now we start Round 2. We tell the Generator that its money images are suddenly getting rejected as fake so it needs to step up its game as shown in the Figure. We also tell it that the Discriminator is now looking for faces, so the best way to confuse the Discriminator is to put a face on the bill:

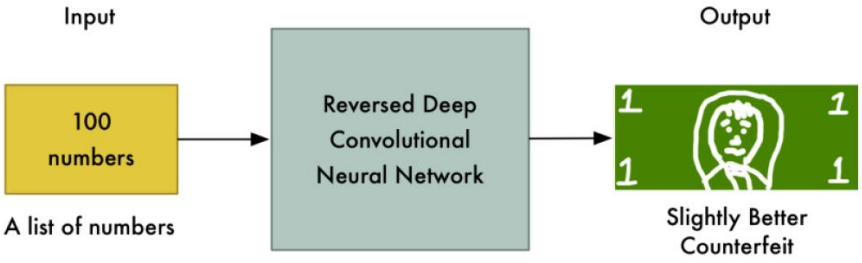

Figure: Discriminator uses features to distinguish between real and fake images so slightly better counterfeits are perceived as real images by the discriminator.

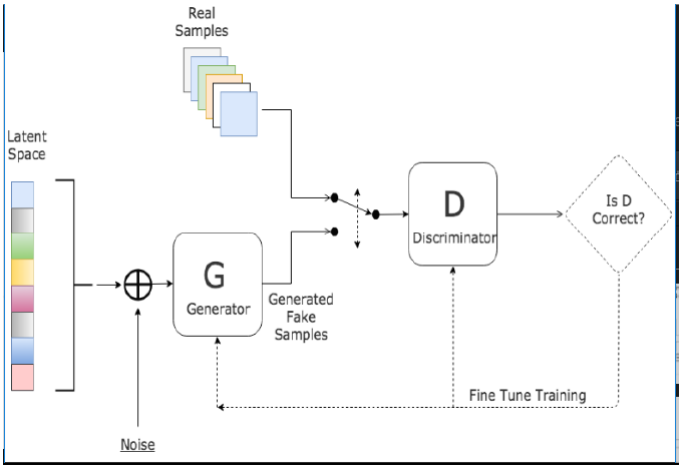

Figure: Generative Adversarial Networks

And the fake bills are being accepted as valid again! So now the Discriminator has to look again at the real dollar and find a new way to tell it apart from the fake one as shown in the Figure. This back-and-forth game between the Generator and the Discriminator continues thousands of times until both networks are experts. Eventually the Generator is producing near-perfect counterfeits and the Discriminator has turned into a Master Detective looking for the slightest mistakes. At the point when both networks are sufficiently trained so that humans are impressed by the fake images, we can use the fake images for whatever purpose we want.

VI. LSTM ENCODER-DECODER

The Encoder-Decoder LSTM is a recurrent neural network designed to address sequence-to-sequence problems, sometimes called seq2seq. Sequence-to-sequence prediction problems are challenging because the number of items in the input and output sequences can vary. For example, text translation and learning to execute programs are examples of seq2seq problems. One approach to seq2seq prediction problems that has proven very effective is called the Encoder- Decoder LSTM. This architecture comprises two models: one for reading the input sequence and encoding it into a fixed-length vector, and the other for decoding the fixed-length vector and outputting the predicted sequence. The use of the models in concert gives the architecture its name of Encoder- Decoder LSTM designed specifically for seq2seq problems.

ISSN: 2350-0328

International Journal of AdvancedResearch in Science, Engineering and Technology

Vol. 5, Issue 9 , September 2018

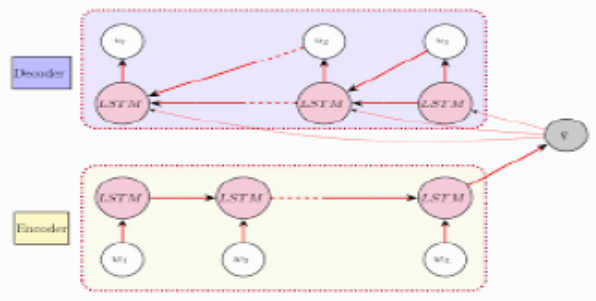

Figure : LSTM Encoder and Decoder.

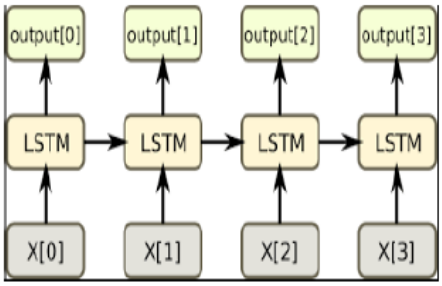

Figure : RNN with LSTM

IV. IMPLEMENTATION OF THE PROPOSED SYSTEM

In this work we are interested in translating text in the form of single-sentence human-written descriptions directly into image pixels. For example, "this small bird has a short, pointy orange beak and white belly" or "the petals of this flower are pink and the anther are yellow". Traditionally this type of detailed visual information about an object has been captured in attribute representations - distinguishing characteristics the object category encoded into a vector. Fortunately, deep learning has enabled enormous progress in both subproblems - natural language representation and image synthesis - in the previous several years, and we build on this for our current task. However, one difficult remaining issue not solved by deep learning alone is that the distribution of images conditioned on a text description is highly multimodal, in the sense that there are very many plausible configurations of pixels that correctly illustrate the description. The reverse direction (image to text) also suffers from this problem but learning is made practical by the fact that the word or character sequence can be decomposed sequentially according to the chain rule; i.e. one trains the model to predict the next token conditioned on the image and all previous tokens, which is a more well-defined prediction problem. Our main contribution in this work is to develop a simple and effective GAN architecture and training strategy that enables compelling text to image synthesis of bird and flower images from human-written descriptions. We mainly use the Oxford-102 Flowers dataset along with five text descriptions per image we collected as our evaluation setting. Our model is trained on a subset of training categories, and we demonstrate its performance both on the training set categories and on the testing set, i.e. "zero-shot" text to image synthesis. In addition to birds and flowers, we can apply our model to more general images and text descriptions in the MS COCO dataset.

A. Generative Adversarial Network

Generative Adversarial Networks (GANs) consist of a generator G and a discriminator D that compete in a twoplayer minimax game: The discriminator tries to distinguish real training data from synthetic images, and the generator tries to fool the discriminator. Concretely, D and G play the following game on V(D;G): minmaxV(D;G) = expx[logD(x)]+expz[log (1-D(G(z)))] Goodfellow et al. (2014) proved that this minimax game has a global optimium precisely when pg = pdata, and that under mild conditions (e.g. G and D have enough capacity) pg converges to pdata. In practice, in the start of training samples from D are extremely poor and rejected by D with high confidence. It has been found to work better in practice for the generator to maximize log (D(G(z)) instead of minimizing log (1-D(G(z))).

B. Deep Symmetric Structured Joint Embedding

To obtain a visually-discriminative vector representation of text descriptions, we follow the approach of Reed et al. (2016) by using deep convolutional and recurrent text encoders that learns a correspondence function with images. The text classifier induced by the learned correspondence function is trained by optimizing the following structured loss: To train the model a surrogate objective is minimized (Akata et al. 2015). The resulting gradients are backpropagated to learn a discriminative text encoder. Reed et al. (2016) found that different text encoders worked better for CUB versus Flowers, but for full generality and robustness to typos and large vocabulary, in this work we always used a hybrid character level convolutional-recurrent network. Intuitively, better image samples should yield a better classifier by providing data augmentation or even the entire training set; classification accuracy can thereby provide a quantitative way to compare GAN models.

ISSN: 2350-0328
International Journal of AdvancedResearch in Science, Engineering and Technology
Vol. 5, Issue 9 , September 2018

C. Visual Semantic Embedding

Our Deep Visual-Semantic Embedding model (DeViSE) is initialized from these two pre-trained neural network models. The embedding vectors learned by the language model are unit normed and used to map label terms into target vector representations. The core visual model, with its softmax prediction layer now removed, is trained to predict these vectors for each image, by means of a projection layer and a similarity metric. The projection layer is a linear transformation that maps the 4,096-D representation at the top of our core visual model into the 500- or 1,000-D representation native to our language model. The model was trained to produce a higher dot-product similarity between the visual model output and the vector representation of the correct label than between the visual output and other randomly chosen text terms.

D. Overall Performance evaluation

Overall Efficiency Improvement:
Compared with GAN-INT, DCGAN achieves 20 percent to 30 percent improvement on the inception score for Oxford-102 Flowers dataset (from 2.66 to 3.20). The better average human rank of our DCGAN also indicates our proposed method is able to generate more realistic samples conditioned on text descriptions.

VI. OUTPUTS

```
sample_sentence = ["the flower shown has red anther red pistil and
bright red petals."] * int(sample_size/ni) + \
                  ["this flower has petals that are yellow, white and
purple and has dark lines"] * int(sample_size/ni) + \
                  ["the petals on this flower are white with a yellow
center"] * int(sample_size/ni) + \
                  ["this flower has a lot of small round pink petals."] *
int(sample_size/ni) + \
                  ["this flower is orange in color, and has petals that
are ruffled and rounded."] * int(sample_size/ni) + \
                  ["the flower has yellow petals and the center of it is
brown."] * int(sample_size/ni) + \
                  ["this flower has petals that are blue and white."] *
int(sample_size/ni) +\
                  ["these white flowers have petals that start off
white in color and end in a white towards the tips."] *
int(sample_size/ni)
```

Figure: Input Text

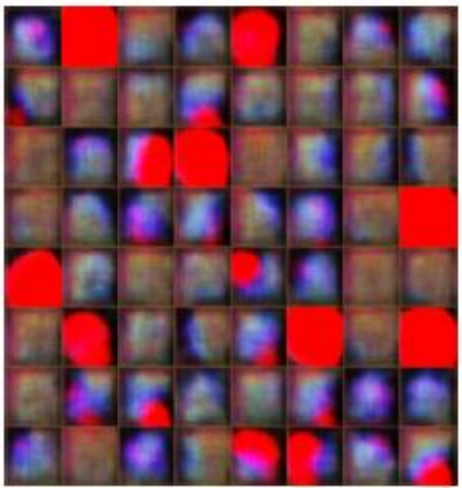

Figure : Output : Training -1

ISSN: 2350-0328
International Journal of AdvancedResearch in Science, Engineering and Technology
Vol. 5, Issue 9 , September 2018

Figure: Output : Training -3

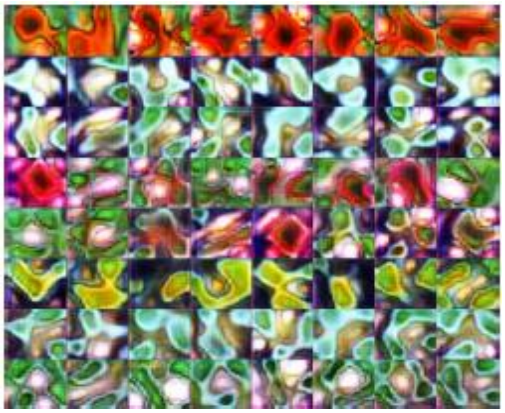

Figure : Output : Training -7

Figure: Output : Training -10

Figure : Output : Training -Final

VI. CONCLUSION AND FUTURE WORK

To conclude, we have discussed, in this report, the two ways of generating images ie, static retrieval of images and synthetic generation of images. Static retrieval involves comparing the vector values and hence precision can be assured , whereas in synthetic image synthesis the accuracy has to be still improved as it is a machine learning model. Training machine learning models on standard synthetic images is problematic as the images may not be realistic enough, leading the model to learn details present only in synthetic images and failing to generalize well on real images. One approach to bridge this gap between synthetic and real images would be to
improve the simulator which is often expensive and difficult, and even the best rendering algorithm may still fail to model all the details present in the real images. This lack of realism may cause models to overfit to 'unrealistic' details in the synthetic images.

A. Enhancement

In particular to GANs we can make the following advancements:

1. A conditional generative model p(xjc) can be obtained by adding c as input to both G and D.
2. Learned approximate inference can be performed by training an auxiliary
   network to predict z given x. This is similar to the inference net trained by the wake-sleep algorithm but with the advantage that the inference net may be trained for a fixed generator net after the generator net has finished training.

ISSN: 2350-0328

International Journal of AdvancedResearch in Science, Engineering and Technology

Vol. 5, Issue 9 , September 2018

3. One can approximately model all conditionals by training a family of conditional models that share parameters. Essentially, one can use adversarial nets to implement a stochastic extension of the deterministic MP-DBM .
4. Semi-supervised learning: features from the discriminator or inference net could improve performance of classifiers when limited labeled data is available.
5. Efficiency improvements: training could be accelerated greatly by devising better methods for coordinating G and D or determining better distributions to sample z from during training.

B. Future Scope

Our imaginations can be fed in as text to this integrated system on the user interface and we can see an visual representation of our imagination. Anything that we can describe as text can be seen as an equivalent image.

1. Children thinking can be improved as their imaginations can be represented as a sequence of images. eg: A fish with hands can be a creative idea of a child.
2. Movie directors can feed in their scripts and see the approximate output of their movie to correct and enhance the movie.
3. Interior designers can check if their design ideas suit that particular room by giving the description of their ideas.

C. Improving realism in images

The goal of "improving realism" is to make the images look as realistic as possible to improve the test accuracy. This means we want to preserve annotation information for training of machine learning models. We learn a deep neural network, which we call 'refiner network', that processes synthetic images to improve their realism.

REFERENCES

[1]. Pradeep Muthukrishnan, Dragomir R. Radev'Algorithms for Information Retrieval and Natural Language Processing tasks', University of Michigan, September 2008
[2]. Chao Shang, Anand Panangadan, and Viktor K. Prasanna,'Event Extraction from Unstructured Text Data',International Conference on Database and Expert Systems Applications DEXA 2015- University of Southern California, 2015
[3]. Frederik Hogenboom, Flavius Frasincar, Uzay Kaymak, and Franciska de Jong, 'An Overview of Event Extraction from Text', Workshop on Detection,Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2011) at Tenth International SemanticWeb Conference (ISWC 2011). Volume 779 of CEUR Workshop Proceedings., CEURWS.org (2011) Pages 48–57, 2011
[4]. Ali Zaidi, 'Text to Image Synthesis Using Stacked Generative Adversarial Networks', Cornell University Library- Stanford University & Microsoft AIR, 2017
[5]. Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, 'Generative Adversarial Text to Image Synthesis', University of Michigan, Ann Arbor, Max Planck Institute for Informatics,International Machine Learning Society (IMLS)- Saarbrucken, Germany, 2016
[6]. Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, 'Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval',57,58 IEEE/ACM Transactions on Audio, Speech and Language Processing(TASLP) Volume 24 Issue 4 Pages 694-707, April 2016
[7]. Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen,'Improved Techniques for Training GANs', NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems Pages 2234-2242, 2016
[8]. 'https://media.readthedocs.org/pdf/gensim/stable/gensim.pdf', 2015
[9]. 'https://machinelearningmastery.com/develop-word-embeddings-pythongensim