

Rochester Institute of Technology

RIT Digital Institutional Repository

Theses

2-2020

Cross-modal data retrieval and generation using deep neural networks

Premkumar Udaiyar
pxu4114@rit.edu

Follow this and additional works at: <https://repository.rit.edu/theses>

Recommended Citation

Udaiyar, Premkumar, "Cross-modal data retrieval and generation using deep neural networks" (2020). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

Cross-modal data retrieval and generation using deep neural networks

By

Premkumar Udaiyar

February 2020

A Thesis Submitted
in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Computer Engineering

Committee Approval:

Dr. Raymond Ptucha, Department of Computer Engineering

Dr. Alexander Loui, Department of Computer Engineering

Dr. Andres Kwasinski, Department of Computer Engineering

R·I·T | KATE GLEASON
College of ENGINEERING

Department of Computer Engineering

Acknowledgments

I am deeply grateful to Dr. Raymond Ptucha for the constant support throughout my degree completion. I would also like to thank Dr. Alexander Loui and Dr. Andres Kwasinski for accepting to join my thesis committee. I am thankful to Dr. Shagan Sah for laying down the foundation for us to build upon and guiding us. I would also take this opportunity to thank my family members and friends for being with me in my ups and downs and encouraging me to do more than I can.

Abstract

The exponential growth of deep learning has helped solve problems across different fields of study. Convolutional neural networks have become a go-to tool for extracting features from images. Similarly, variations of recurrent neural networks such as Long-Short Term Memory and Gated Recurrent Unit architectures do a good job extracting useful information from temporal data such as text and time series data. Although, these networks are good at extracting features for a particular modality, learning features across multiple modalities is still a challenging task. In this work, we develop a generative common vector space model in which similar concepts from different modalities are brought closer in a common latent space representation while dissimilar concepts are pushed far apart in this same space. The developed model not only aims at solving the cross-modal retrieval problem but also uses the vector generated by the common vector space model to generate real looking data. This work mainly focuses on image and text modalities. However, it can be extended to other modalities as well. We train and evaluate the performance of the model on Caltech CUB and Oxford-102 datasets.

Contents

Cross-modal data retrieval and generation using deep neural networks.....	1
Contents	4
List of Figures	6
List of Tables	8
Acronyms.....	9
Chapter 1.....	10
Introduction.....	10
1.1 Introduction.....	10
1.2 Motivation.....	11
1.3 Contributions.....	11
Chapter 2.....	12
Background.....	12
2.1 Deep Learning.....	12
2.2 Convolution Neural Networks	12
2.3 Recurrent Neural Network	16
A. Long Short-Term Memory	18
B. Gated Recurrent Unit	20
2.4 Generative Adversarial Network (GAN)	21
A. StackGan [28]	23
2.5 Cross Modal Retrieval	24
2.6 Loss Functions	25
A. Triplet Loss Function.....	25
B. Contrastive Loss Function	27
C. Softmax and Cross-Entropy Loss	28
Chapter 3.....	29
Methodology	30
3.1 Common Vector Space Network	30
3.2 Generative Network.....	32
3.3 Loss functions	34
Chapter 4.....	35
Implementation	35
4.1 Datasets.....	35
A. Caltech CUB Dataset	35

B. Oxford-102 dataset.....	36
4.2 Implementation	37
4.3 Evaluation metrics	38
A. Mean Average Precision (mAP)	38
B. Inception score	38
Chapter 5.....	40
Results and Analysis.....	40
5.1 Results.....	40
A. Retrieval output (Caltech CUB dataset).....	40
B. Generative output (Caltech CUB dataset).....	40
C. Retrieval output (Oxford-102 dataset)	45
D. Generative output (Oxford-102 dataset)	45
5.2 Analysis	50
A. Analysis for oxford-102 dataset.....	50
B. Analysis for Caltech CUB dataset	53
C. Analysis for captioning model	56
Chapter 6.....	58
Conclusions.....	59
6.1 Conclusions.....	59
6.2 Future Work.....	59
Bibliography	59

List of Figures

Figure 1 Illustration of CNN. An input image is passed to multiple convolution and pooling layers which is then passed to fully connected layers and softmax function to output the probability for each class.	14
Figure 2 Convolution operation. I - input image, K – Filter kernel, $I*K$ – convolution output [50].....	15
Figure 3 Max-pool operation. Left – input, Right – max-pool output [52].	15
Figure 4 Avg-pool operation. Left – input, Right – avg-pool output [53].....	16
Figure 5 Recurrent Neural Network model which takes in input from previous hidden state and current input for predicting the next outcome [54].....	17
Figure 6 Internal structure of Long Short-Term Memory [55].....	18
Figure 7 Internal structure of Gated Recurrent Unit [56].	20
Figure 8 Basic generative adversarial network [57].	21
Figure 9 StackGan model [28].....	23
Figure 10 Working of triplet loss [58].	26
Figure 11 Negative mining [59].....	28
Figure 12 shows the transformation of the data before and after training in the vector space [60].	31
Figure 13 CVS Network	31
Figure 14 Generative network	32
Figure 15 Model architecture.....	33
Figure 16 examples from Caltech cub dataset. [61]	36
Figure 17 Examples from Oxford-102 dataset. [61]	37
Figure 18 Sampled input images from the dataset.	41
Figure 19 input: image vector, output: image.	43
Figure 20 input: text vector, output: image.....	44
Figure 21 Sampled images from Oxford-102 dataset.	46
Figure 22input: image vector, output: image.....	48
Figure 24input: text vector, output: image.....	49
Figure 24 input: text vector, output: image.....	49

Figure 25 Adding captioning model along with generative network.	56
---	----

List of Tables

Table 1 Train and validation split for Caltech CUB dataset.	35
Table 2 Train and validation split for Oxford-102 dataset.	36
Table 3 mAP@50 score for Caltech CUB dataset.	40
Table 4 Inception scores for Caltech CUB dataset.	44
Table 5 mAP@50 score for Oxford-102 dataset.	45
Table 7 Analysis for oxford-102 dataset (good examples).	50
Table 8 Analysis for oxford-102 dataset (bad examples).	52
Table 9 Analysis for caltech CUB dataset (good examples).	53
Table 10 Analysis for caltech CUB dataset (bad examples).	54
Table 11 Analysis for captioning model (Oxford-102 dataset).	57

Acronyms

CNN

Convolution Neural Network

CVS

Common Vector Space

RNN

Recurrent Neural Network

GRU

Gated Recurrent Unit

LSTM

Long Short Term Memory unit

GAN

Generative Adversarial Network

1.1 Introduction

Deep learning models can extract features from the data automatically without any manual adjustment. Deep Neural Networks (DNNs) have shown great ability at performing tasks on a wide range of applications such as image captioning, object detection, and segmentation. A better understanding of cost functions and the amount of training data has helped the neural network model to learn complex structures in images, videos, audios, and texts. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have shown great results in working with image and text data respectively. However, learning common features across the two modalities is still a challenging problem. Karpathy et al. [24] introduced two different networks for solving image captioning problems. They used a CNN to extract features from images and passed these features into a Long Short-Term Memory (LSTM) network to generate the caption for each image. For the past few years, Generative Adversarial Networks (GAN's) have been successful in generating realistic images from a noise vector. Goodfellow et al. [26] introduced GAN's for generating real-looking images from a noise vector. Since, variants of GAN's solved many real-world problems such as mode collapse [27, 28 29].

The performance of deep learning algorithms is directly proportional to the amount of data available for training. Deep learning algorithms simultaneously learn feature extraction and classification parameters through the process of backpropagation. Converting data from one modality to another is a challenging task. Singh et al. [30] have shown how to convert a sensor data into an image. Zhou et al. [31] have shown how to fuse two modalities for more accurate predictions. Lee et al. [32] fused the data in the final stage of learning method for a medical application.

1.2 Motivation

Vectors are used as an input and output to a deep learning model. These vectors are a bunch of numbers which are learned as we train the model to minimize the loss. The data coming from various modalities (image, text, audio, video.) is first converted into its vector representation before passing on to the deep learning models. Therefore, it is important that the vector is good enough to represent the data properly.

The main contribution of this work is to develop a novel architecture which can retrieve data across the image and text modalities while simultaneously generating real looking text and image data using the learned vector representation. In this work, we bring similar concepts closer in the vector space and pull dissimilar concepts far apart. For example, an image of a dog and a text describing a dog would lie closer to each other in the vector space and a text describing a truck would lie far away from the dog image. Simultaneously, the vector representing this data would be used to generate data which is similar to the input data. For example, given a query image about a dog, the model would be able to both retrieve text samples about dogs as well as generate new text samples about dogs. Similarly, given a query text about a dog, the model would be able to both retrieve images about dogs as well as generate new sample images of dogs.

1.3 Contributions

The main contributions of this thesis work can be summarized as:

- Build a model which can retrieve data across the modalities.
- Extend the model to not only perform cross modal retrieval but also generate realistic looking images.
- Try different loss functions and report the performance.

2.1 Deep Learning

Deep learning is a subdomain of machine learning in which algorithms are inspired by the structure of the human brain. Deep learning models are built upon the way the human brain is structured, having several layers and the information passes through each layer to generate an output. Traditional machine learning models are not scalable i.e. the model performance does not improve as we add more data to the training set. On the other hand, deep learning models are scalable. Deep learning model performance increases as you add more data to the training set. The deep learning model can extract features from the data automatically without extensive handcrafted feature engineering. There are a lot of deep learning algorithms out there for different problems such as CNNs and RNNs. CNN's are good at extracting features from image data, while RNN's perform well on temporal data.

2.2 Convolution Neural Networks

CNN's have become the most important tool for extracting features from visual data. CNN's are a modified version of a multi-layer perceptron. CNN's have evolved exponentially since the invention of the VGG-Net [9] model. Szegedy et al. [10] and Kaiming et al. [11] came up with more complex models to increase the accuracy on multiple benchmark datasets such as CIFAR10 [12] and ImageNet [13]. CNN's perform well on static gridded data. However, some researchers use CNN's for temporal data as well. Yoon Kim [20] used CNN for sentence classification where the model takes a sentence as an input and outputs the probability of all the classes. CNN's are also used in other applications such as image segmentation and object detection.

CNN's consists of four basic components:

- Convolution layer

- Pooling layer
- Activation layer
- Fully connected layer.

Convolution layers perform convolution operations on input images using finite impulse response filters. Pooling layers are used to reduce the size of the input image. Fully connected layers perform nonlinear operations in the network. The last fully-connected layer of the model is used for classification and regression. Both the features and classifiers are trained simultaneously through the process of backpropagation. Backpropagation updates the weights of the network based on the error generated at the output of the network. The early layers of the CNN learn high-level features such as horizontal and vertical edges while later layers learn the low-level features such as colors, shapes, and structures.

The convolution layer typically convolves with the filters and preserves the size of the image given that necessary padding is provided to the input and with a stride of 1. While the pooling layers reduces the size of the input image. Max-pooling and avg-pooling are two regularly used pooling functions. Max-pooling takes the maximum value from the selected frame and avg-pooling takes the average of all the values in the frame. Max-pooling adds more non-linearity to the model as compared to avg-pooling. Fully connected layers are used to convert the convolution filter into a vector form. Parameters learned connecting the last convolution layer to the first fully connected layer are often higher than all other layers combined.

Figure 1 shows the CNN architecture for the image classification task. An input image is passed to multiple convolutions and pooling layers to extract a different level of features. The output of the last convolution layer is passed to a fully connected layer to flatten the features. The last fully connected layer is passed through a softmax function which outputs the probability for each class and the class with maximum probability is assigned to that input. The convolution layers use non-linear activation functions such as

ReLU, sigmoid, tanh, and eakyReLU to learn the complex features in the data. ReLU is perhaps the most widely used activation function.

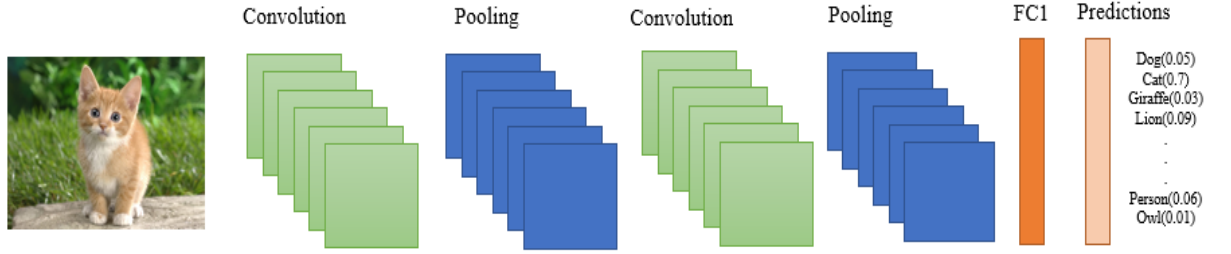


Figure 1 Illustration of CNN. An input image is passed to multiple convolution and pooling layers which is then passed to fully connected layers and softmax function to output the probability for each class.

Equation (2.2.1) demonstrates the convolution operation.

$$M_{(i,j,k)}^l = K_{abc} I_{(r+a,h+b,c)}^{l-1} \quad (2.2.1)$$

Where:

- $I_{(x_r+a,x_h+b,c)}^{l-1}$ – input image at (r, h) convolved with filter size of (a, b)
- K_{abc} – Kernel
- $M_{(i,j,k)}^l$ – Convolution output

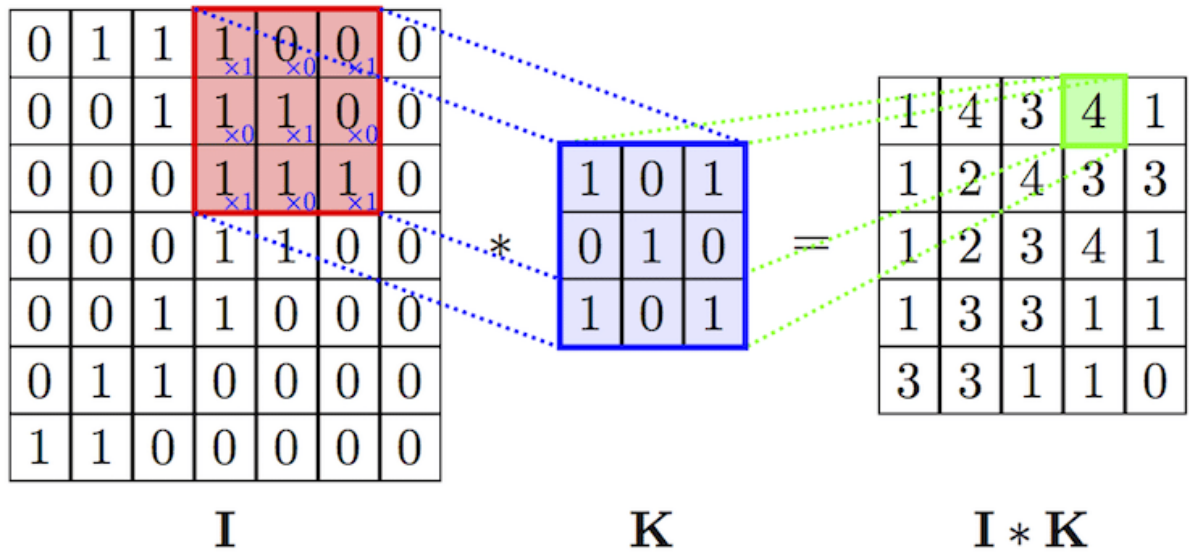


Figure 2 Convolution operation. I - input image, K - Filter kernel, $I * K$ - convolution output [50].

Figure 2 illustrates the working of a convolution operation. Input (I) is convolved with kernel filter (K) to output ($I * K$) convolution map. K is learned during the process of backpropagation. These filters are used to learn low-level and high-level features from the data.

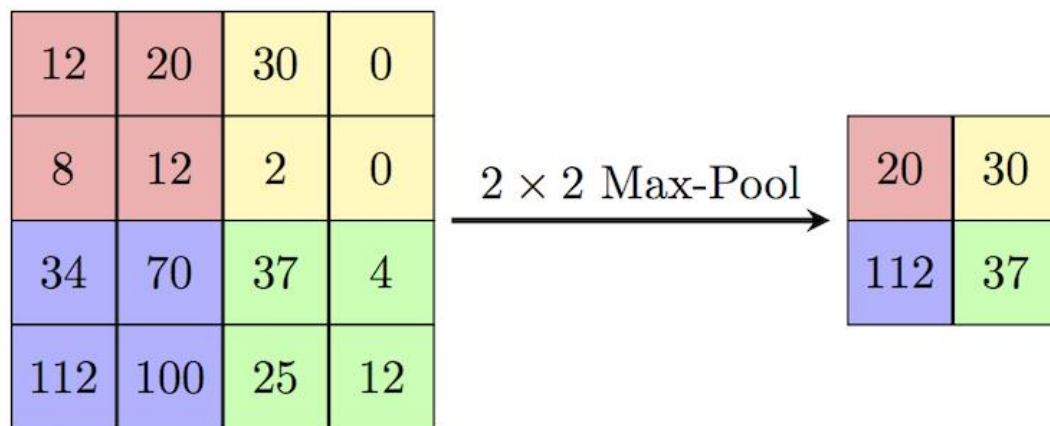


Figure 3 Max-pool operation. Left - input, Right - max-pool output [52].

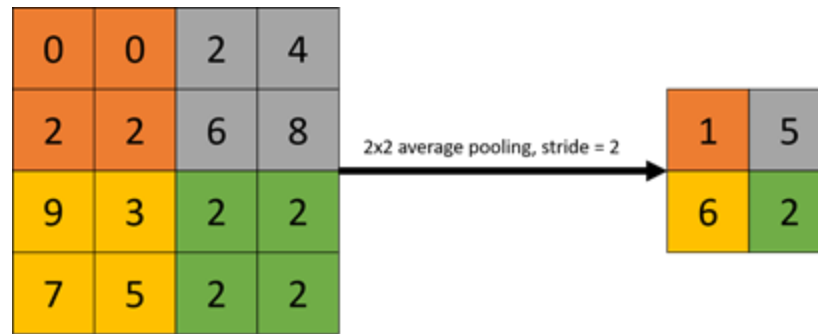


Figure 4 Avg-pool operation. Left – input, Right – avg-pool output [53].

Figure 3 depicts the working of max-pool operation. The maximum of the cell is selected and passed on to the output buffer. Figure 4 depicts the working of avg-pool operation. The average of each cell is passed on to the output buffer.

2.3 Recurrent Neural Network

RNNs are good at extracting features from sequential data such as time series, text, and audio. RNNs consists of an encoder which encodes the data into a vector and a decoder which decodes the vector into a destination data format. RNNs use some information from the previous layer to predict future outcomes. RNNs such as Long-Short Term Memory [21] and Gated Recurrent Unit [22] consists of a memory cell which helps remember previous layer information. RNN's are widely used in applications such as image captioning [17, 18], and video summarization [19].

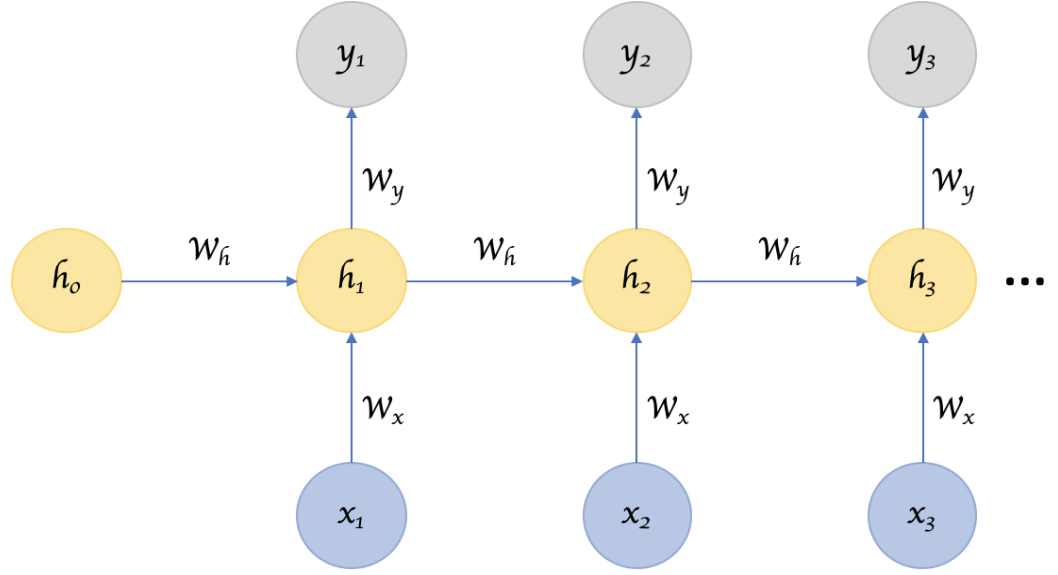


Figure 5 Recurrent Neural Network model which takes in input from previous hidden state and current input for predicting the next outcome [54].

Figure 5 shows the basic RNN which takes input from the previous time step and current input to predict the next outcome. $h_0, h_1, h_2, h_3, \dots, h_t$ are the inputs to the time steps $t = 0, 1, 2, 3, \dots, t$ which are used along with $x_1, x_2, x_3, \dots, x_t$ to predict the output $y_1, y_2, y_3, \dots, y_t$.

$$h_t = f(W^{hh}h_{t-1} + W^{hx}x_t) \quad (2.3.1)$$

$$h_t = \tanh(h_t) \quad (2.3.2)$$

$$y_t = W^{hy}h_t \quad (2.3.3)$$

The above equations describe the working of RNN. Equation (2.3.1) calculates the current hidden state using current input and previous hidden state. In (2.3.2), it is passed through a tanh function to constrain the output values in the range of -1 and 1 which adds non-linearity in the model. Finally, (2.3.3) calculates the final output by multiplying with the weight matrix. The weights are updated for each time step by backpropagation using the error calculated for that time step.

RNNs do a great job at extracting features from temporal data but they sometimes face the problem of vanishing gradients. The vanishing gradient problem occurs when the value of gradients exponentially decreases (with repeated multiplies of values less than 1.0) as it reaches to the early layers. To tackle this problem, LSTM's and GRU's are used instead.

A. Long Short-Term Memory

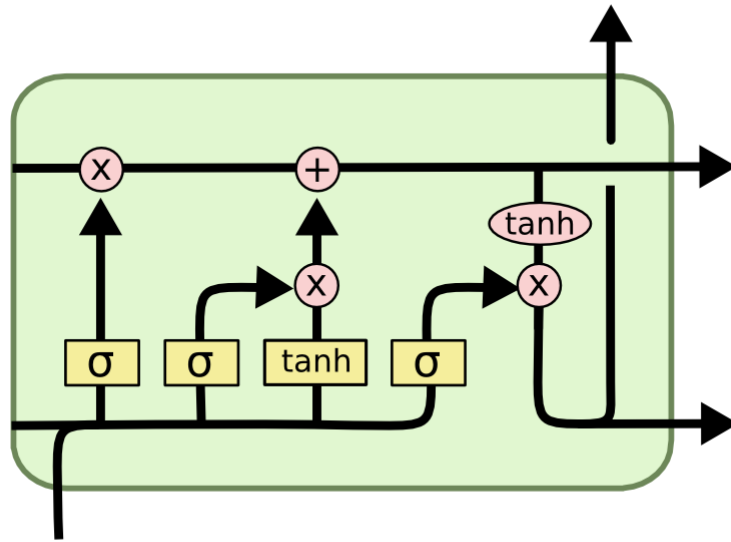


Figure 6 Internal structure of Long Short-Term Memory [55].

Figure 6 shows the internal structure of the LSTM. LSTM's are used for problems containing temporal data. LSTM's solve much of the vanishing gradient problem which is observed in basic RNN's. LSTM's come with a memory cell that helps remember the previous information for longer time steps. They were introduced by Hochreiter and Schmidhuber [21]. Several other works use this method and have refined it.

$$i_t = \sigma(x_t U^i + h_{t-1} W^i) \quad (2.3.5)$$

$$f_t = \sigma(x_t U^f + h_{t-1} W^f) \quad (2.3.5)$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o) \quad (2.3.6)$$

$$\hat{C}_t = \tanh(x_t U^g + h_{t-1} W^g) \quad (2.3.7)$$

$$C_t = \sigma(f_t * C_{t-1} + i_t * \hat{C}_t) \quad (2.3.8)$$

$$h_t = \tanh(C_t) * o_t \quad (2.3.9)$$

Equations (2.3.5) – (2.3.9) describe the working of LSTM. Input gate i calculates the new information that is to be stored. Forget gate f tells the information that is not important for the model to store. Output gate o is used to provide activation to the final output of the LSTM block. σ represents the sigmoid function. It outputs values between 0 and 1. The sigmoid function determines the percentage of information to be passed through the gate. C is the internal memory unit which is used to store the previous information. C_t grabs the information from previous hidden state and current input to calculate current hidden state output. The final hidden state output is calculated by combining C_t and output gate. This hidden state is a vector representation of the data which is then further used for a variety of applications such as classification and captioning.

B. Gated Recurrent Unit

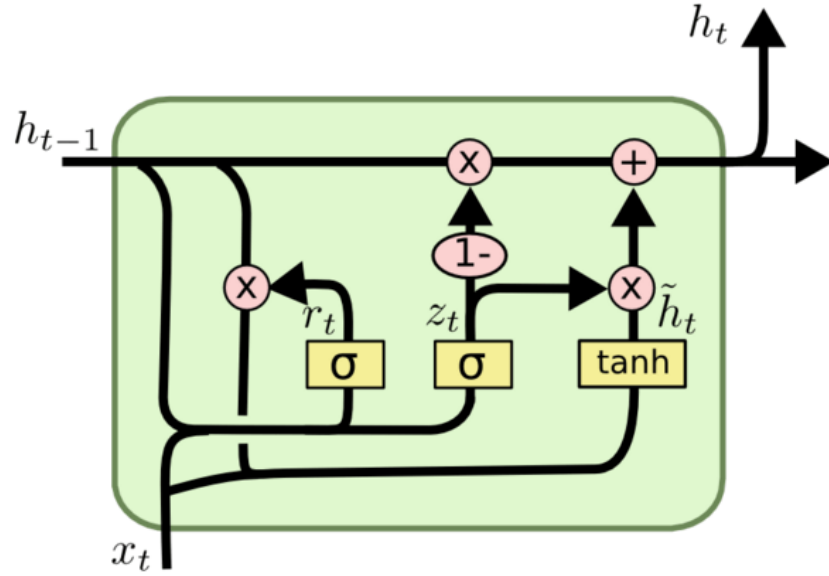


Figure 7 Internal structure of Gated Recurrent Unit [56].

Figure 6 shows the internal structure of the GRU network. The GRU was first introduced by Chung et al. [22]. GRU is a variant of LSTM. Unlike LSTM, GRU has two gates (reset and update gates). GRU doesn't have a memory unit. It outputs the entire hidden unit without any control. GRU is less complex and computationally more efficient as compared to LSTM.

$$z_t = \sigma(x_t U^z + h_{t-1} W^z) \quad (2.3.10)$$

$$r_t = \sigma(x_t U^r + h_{t-1} W^r) \quad (2.3.11)$$

$$\hat{h}_t = \tanh(x_t U^h + (r_t * h_{t-1}) W^h) \quad (2.3.12)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{h}_t \quad (2.3.13)$$

Equations (2.3.10) – (2.3.13) describe the mathematical working of GRU. Update gate z_t is calculated using the current input and the previous hidden state output. Reset gate r_t is calculated to decide how much information about past information to forget. Both, update and reset gate use sigmoid functions to constrain the output between 0 and 1. \hat{h}_t calculates the current memory that is to be passed to final output. h_t calculates the final memory at the time step t .

2.4 Generative Adversarial Network (GAN)

Recently, GAN's have shown great performance at generating realistic looking images from a noise vector. GAN's were first introduced by Goodfellow et al. [26]. Follow on work [27, 28, 29] popularized GAN with valuable refinements. Unlike a conventional neural network, GAN's use two networks competing with each other. The model learns to generate data from the training distribution using a 2-entity game. The two entities are a generator and a discriminator. These two networks fight with each other throughout the training process.

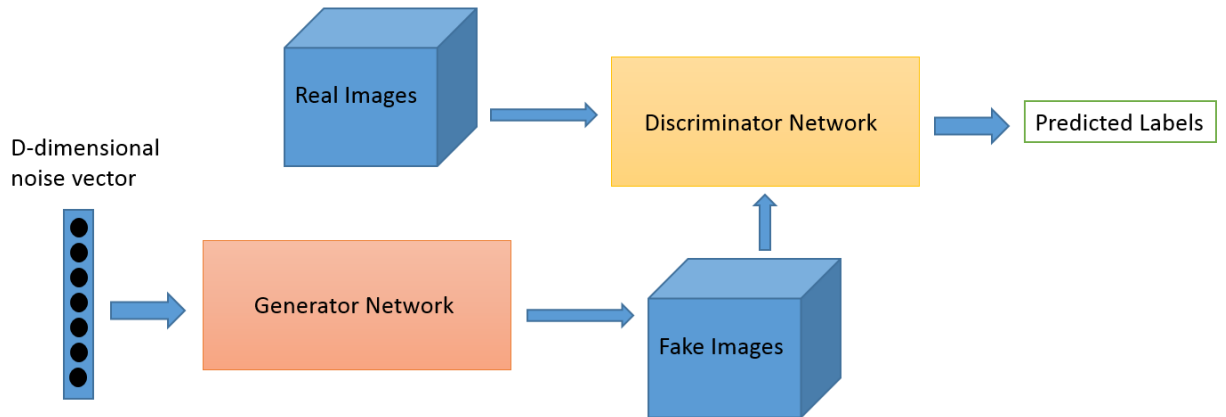


Figure 8 Basic generative adversarial network [57].

Figure 8 shows the basic GAN model. The generator's job is to generate fake images from a D-dimensional noise vector while the discriminator's job is to identify if its input is real or fake. Both the networks are in a constant battle where the generator tries to fool the discriminator and the discriminator's job is not to be fooled. The generator tries to learn the training distribution and generates images that are close to training images. One problem that is commonly faced by GAN models is mode collapse. Mode collapse occurs when the generator network discovers the same or a series similar looking images which fools the discriminator, and thus outputs these images frequently. To tackle this problem, Zhang et al. [28] introduced conditioning augmentation. Conditioning augmentation adds smoothness to the latent distribution. Smoothness removes the clusters formed in the distribution which are used by the generator to move from mode to mode to fool the discriminator. Arjovsky et al. [27] used a different loss function to train the model. They used a distance function to calculate the distance between the training distribution and the generated distribution and this distance is then used to train the model through the process of backpropagation.

$$\text{Loss}_D = \log(D(x)) + \log(1 - D(G(z))) \quad (2.4.1)$$

$$\text{Loss}_G = \log(1 - D(G(z))) \quad (2.4.2)$$

Equation (2.4.1) shows the loss function used to train the discriminator and (2.4.2) shows the loss function used to train the generator. $D(x)$ is the output of discriminator given real input and $D(G(z))$ is the output of discriminator given the generated image as an input. Both the generator and discriminator are trained to output $D(x)$ close to 1 and output $D(G(z))$ close to 0.

A. StackGan [28]

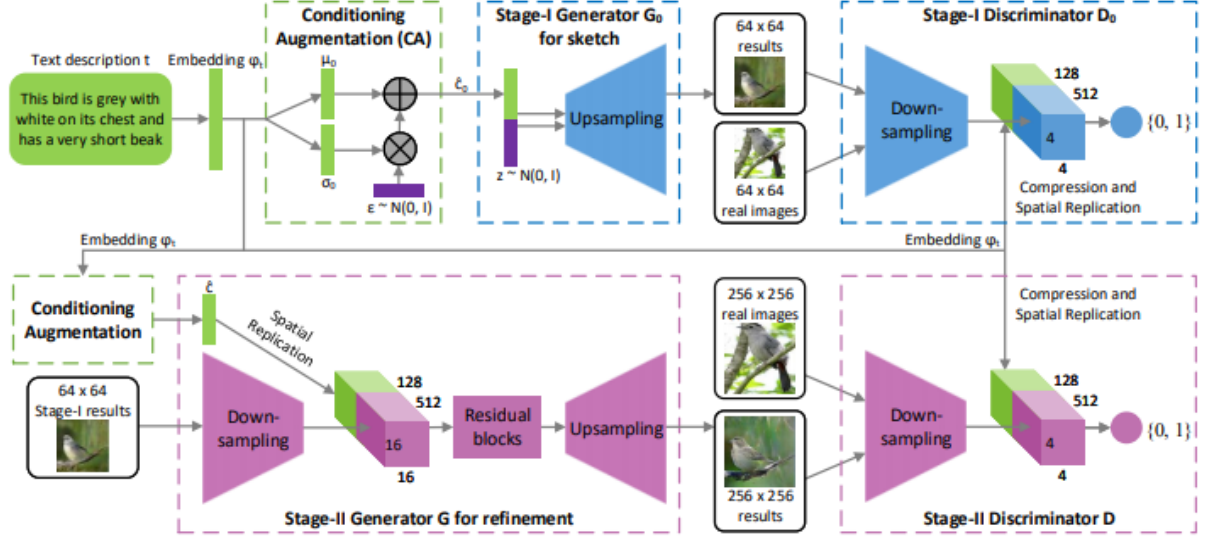


Figure 9 StackGan model [28].

Zhang et al. [28] developed a model called StackGan. StackGan takes text as an input and outputs an image related to the text. The StackGan model introduced conditional augmentation to tackle the problem of mode collapse. The generator uses the conditional input and is forced to generate a variety of images. These images are both required to fool the discriminator as well as meet the conditional requirements. The conditional requirements may, for example be to generate an image such that when this image is passed through an automatic captioner, the generated caption matches the input (conditional) caption.

StackGan consists of two stages of learning. Figure 9 shows the architecture of StackGan model. The stage-I of StackGan sketches the shape and color of the object based on the text input. The stage-II takes the output from the stage-I along with the text as an input, and outputs realistic-looking image.

2.5 Cross Modal Retrieval

Retrieving data across multiple modalities is a challenging task. One solution is to map all modalities to a common space. Different modalities come with different structures and mapping them into the same space can be tricky at times. Ultimately, we have to convert all the data points from different modalities into a vector of the same dimensions so that we can compare data across the modalities. The problem of cross-modal retrieval has been extensively studied by researchers in the past few years.

Traditional approaches use a latent space to compare the data belonging to different modalities. Canonical Correlation Analysis (CCA) [33] was used to maximize the correlation between the data from different modalities.

The cross-modal retrieval problem has been tackled by [2, 3] with different loss functions and different model architectures. Lee et al. [1] computed the similarity between image regions and words in sentences to find the overall similarity between image and text modalities. [4, 5, 6] show cross-modal retrieval by using category information. Zhen et al. [7] included three loss functions while training which helped the model to learn inter- and intra-modal discrepancies. Specifically, they use label information from the image and text data to learn discriminative features and use weight sharing to learn inter-modal discrepancies in the common space. Sanakoyeu et al. [8] jointly divided the embedding space and data into k subparts and learn different metric distances for all of them. They show retrieval, clustering and re-identification tasks using one model.

Recent work uses different models and loss functions to get better results. Xu et al. [34] used the adversarial loss function to minimize the distance between the data vector and built a model to map the 4096-dimensional vector to 200 dimensions using multiple multi-layer perceptrons. Zhen et al. [35] combined three different networks to retrieve data. They used a linear classifier that classifies each instance of the data to a class, a modality

invariance loss to minimize the error within modality, and intermodal loss to minimize error across modalities.

An attention mechanism was introduced to solve the image captioning problem [36] to correlate the words and the image regions which in turn boosts the performance of the model. This was also adopted in cross-modal retrieval. Lee et al. [37] used attention mechanisms to correlate the image regions generated by mask-RCNN [38] with each word in the sentence.

2.6 Loss Functions

One of the important aspects of creating a deep learning model is to define an appropriate loss function for a task. Metric learning loss functions map similar concepts closer to each other and maximizes the distance between dissimilar concepts. These loss functions form pairs of positive and negative samples which is then used to calculate a loss depending on the task.

Steps involved in metric learning models are as follows (using image and text modalities as an example):

- Extract features from images and texts using CNN and LSTM respectively.
- Further convert these features into embeddings by learning fully connected layers.
- Formulate positive and negative pairs based on the class of the data

The following are some of the metric learning loss functions.

A. Triplet Loss Function

The triplet loss was introduced by Schroff et al. [14]. It compares an anchor input with both a positive input and a negative input. The distance between the anchor and positive input is minimized, and the distance between the anchor and negative input is maximized. The triplet loss ensures that the distance between negative inputs is at least a margin away from the positive inputs.

$$L_c = \frac{1}{2N} \sum \max(0, |f_a^i - f_p^i|^2 - |f_a^i - f_n^i|^2 + \alpha) \quad (2.6.1)$$

Where

- N is the number of samples
- f_a^i is the anchor
- f_p^i is positive input
- f_n^i is negative input
- α is the margin

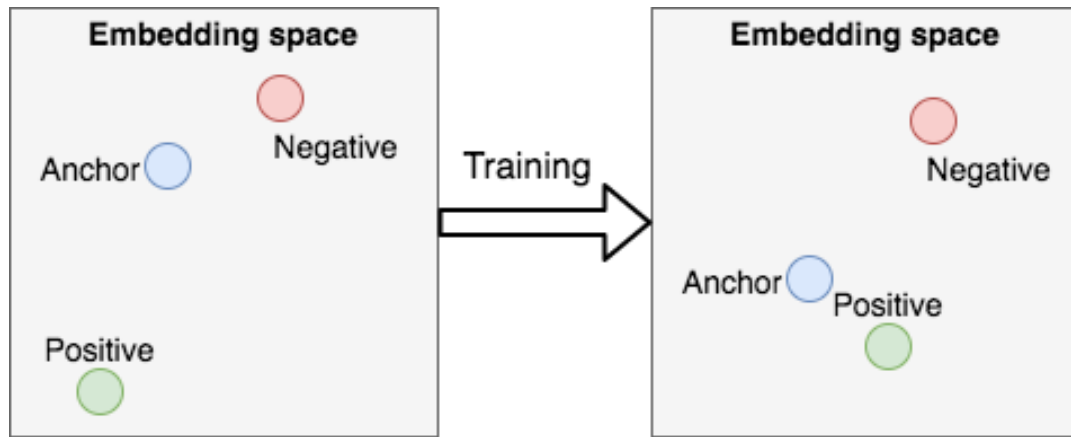


Figure 10 Working of triplet loss [58].

Figure 10 shows the working of triplet loss. Before training, the negative sample can be closer to anchor as shown on the left side of the figure. After training, the negative sample is at least α away from the positive sample.

We can modify this loss functions to account for the classes which are closely related to each other to lie closer to each other in the vector space. This can be done by having low values of alpha for the classes such as person and policeman or car and bus and so on. This will allow the similar looking classes to lie closer in the vector space.

B. Contrastive Loss Function

The contrastive loss was invented by Hadsell et al. [39]. This loss tries to maximize the distance between the negative and positive samples. The goal is to push the negative sample at least margin away from the positive sample. This loss function is mostly used for instance-level retrieval where we don't have class information.

$$L_c = \frac{1}{2N} \sum ((y)d^2 + (1 - y) \max(\text{margin} - d, 0)^2) \quad (2.6.2)$$

Where

- y is either 0 or 1, 1 if the both the samples are similar
- d is the euclidean distance between the samples
- N is the number of samples

Negative samples can be categorized into three types:

- Hard negatives
- Semi-hard negatives
- Easy negatives

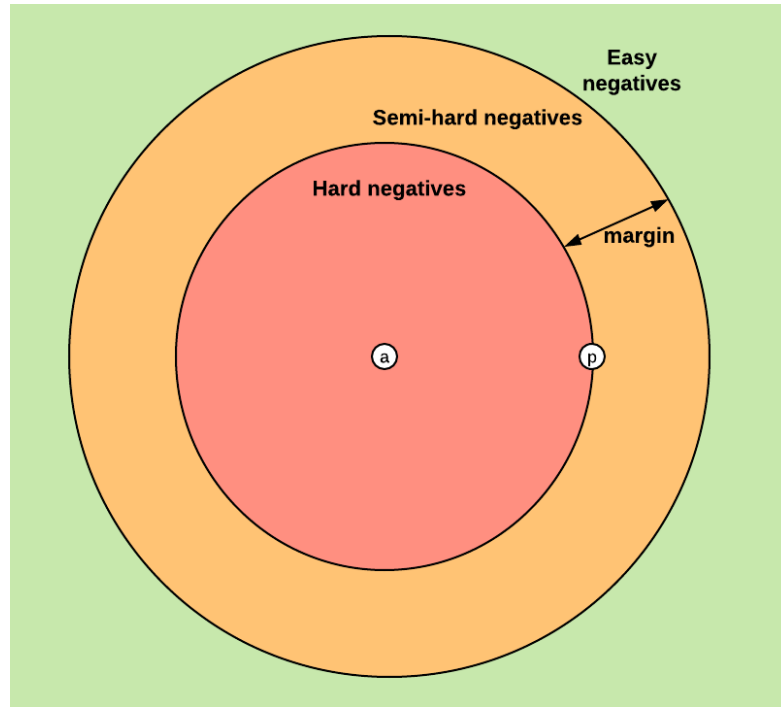


Figure 11 Negative mining [59].

Figure 11 shows the distribution of negative samples around the anchor and the positive sample. Hard negative samples lie close to the anchor sample as compared to the positive sample. Semi-hard negative samples lie within the margin and positive sample. Easy negative samples lie far away from the anchor sample and those samples do not contribute much to the overall loss. Hard negative takes the negative sample in each batch which is very close to the anchor. Since the hard negative is close to the anchor, it generates a high loss and higher gradients which in turn can make the training unstable. Semi-hard negatives produce good loss values to backpropagate and this strategy is often used to train the model.

C. Softmax and Cross-Entropy Loss

Softmax is used before most of the loss function. Softmax takes N dimensional array as input and outputs values between 0 and 1 which can be added up to 1.

Equation (2.6.3) is used to calculate cross-entropy loss.

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^N e^{a_k}} \quad \forall j \in 1 \dots N \quad (2.6.3)$$

Equation (2.6.4) is used to calculate softmax probability.

Cross-Entropy loss is used to observe the performance of the model when classifying the object as belonging to two or more classes. The output of this loss is 0 when classification is done perfectly and gradually increases when samples diverges from the actual label.

$$H(y, p) = - \sum_i y_i \log p_i \quad (2.6.4)$$

Where:

- y – label
- p – output probability

The aim of this thesis is to create a multi-task model which can perform cross modal retrieval and image generation at the same time. A lot of work has shown good performance on the cross modal retrieval problem, and independently, recent methods have been shown to generate realistic-looking images. However, it is difficult to find works that have created a unified model to solve both of these problems. We have created a model such that given an input image, the model will retrieve the closest text to that image and also generates a similar-looking image. Likewise, given a text input, the model will retrieve the closest image to that text and also generate an image that well aligns with the input text.

The model is comprised of two networks:

- Common Vector Space Network
- Generative Network

3.1 Common Vector Space Network

The goal of the Common Vector Space (CVS) network is to bring similar concepts from different modalities closer in the vector space and maximize the distance between dissimilar concepts. Figure 10 shows the before and after training visualization of data points of different modalities. Different shapes are the data from different modalities and different colors are different data points. These are vector representations of data points that are projected onto the lower dimensional CVS. As we can see from the figure, the data is projected at random onto the vector space before training. However, as we train the model, the data belonging to different modalities conveying the same meaning are projected closer in the vector space.

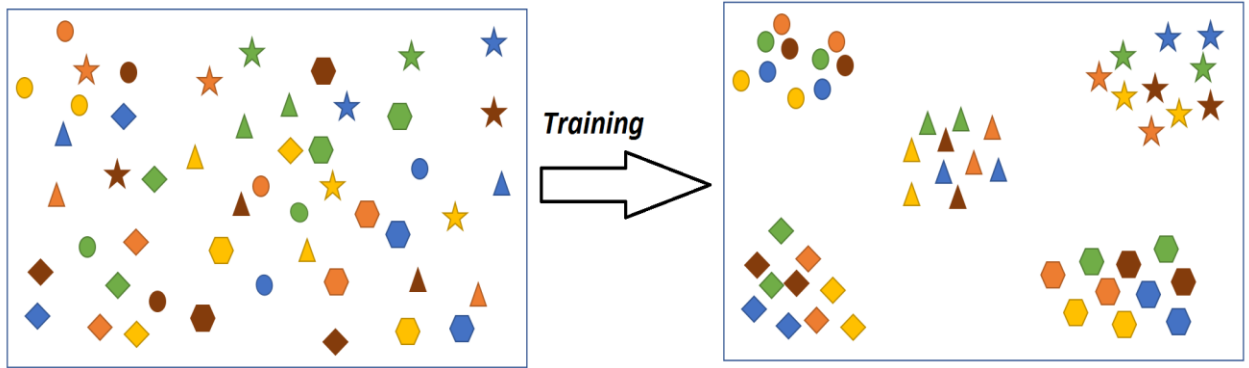


Figure 12 shows the transformation of the data before and after training in the vector space [60].

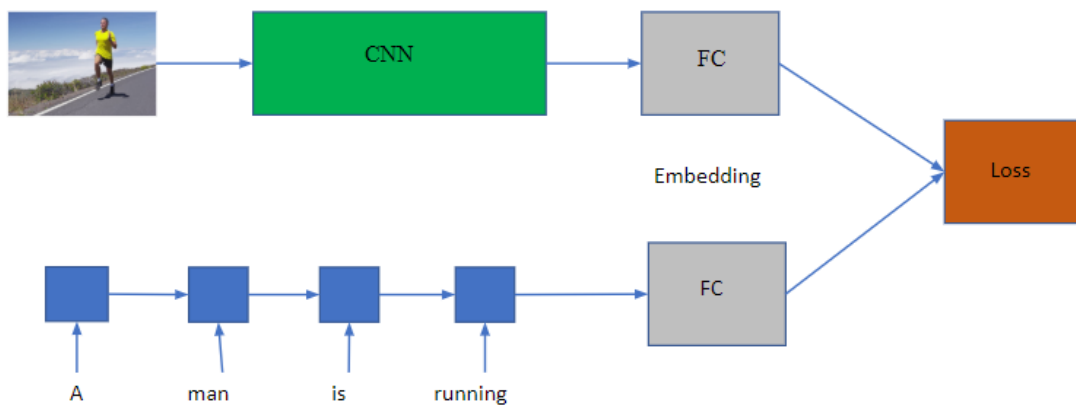


Figure 13 CVS Network.

Figure 13 shows the structure of the CVS network. The CVS network [23] takes image and text samples as input and converts them into their respective feature representations. A CNN is used for extracting features from the image data and an RNN is used to extract features from the text data. The output of these feature networks is then fed to multiple fully connected layers to bring the inputs into a latent vector representation. These CVS vectors have the property such that similar concepts (irrespective of input modality) are brought together and dissimilar concepts are pulled far apart.

The similarity between the two vectors is calculated by using a distance function. Usually, distance functions used are Euclidean distance and cosine distance.

3.2 Generative Network

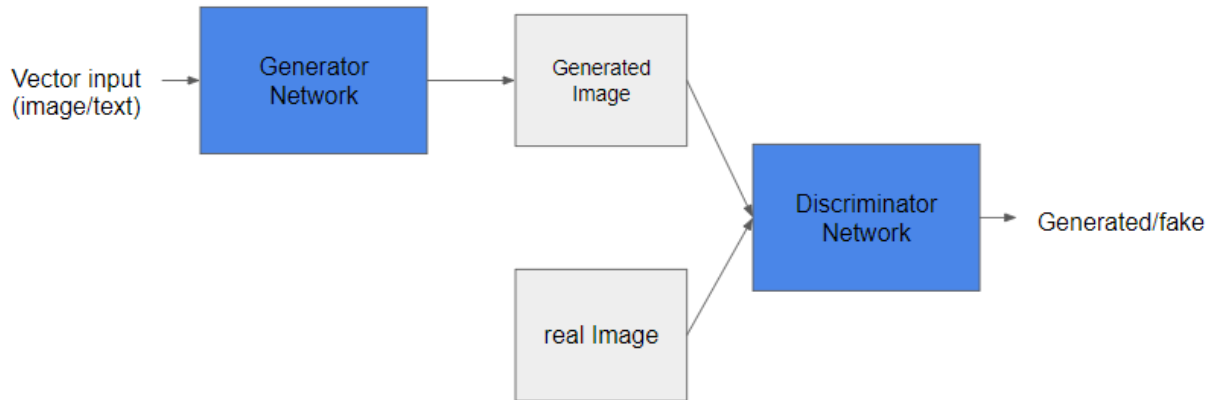


Figure 14 Generative network

Figure 14 shows the structure of a generative network. The input to the network is a vector (vector input) which is generated by the CVS network. This input is then fed to the generator network to generate an image. Generated images along with a real image from the dataset are passed through the discriminator network to calculate the loss. This loss is backpropagated to update the weights of the network.

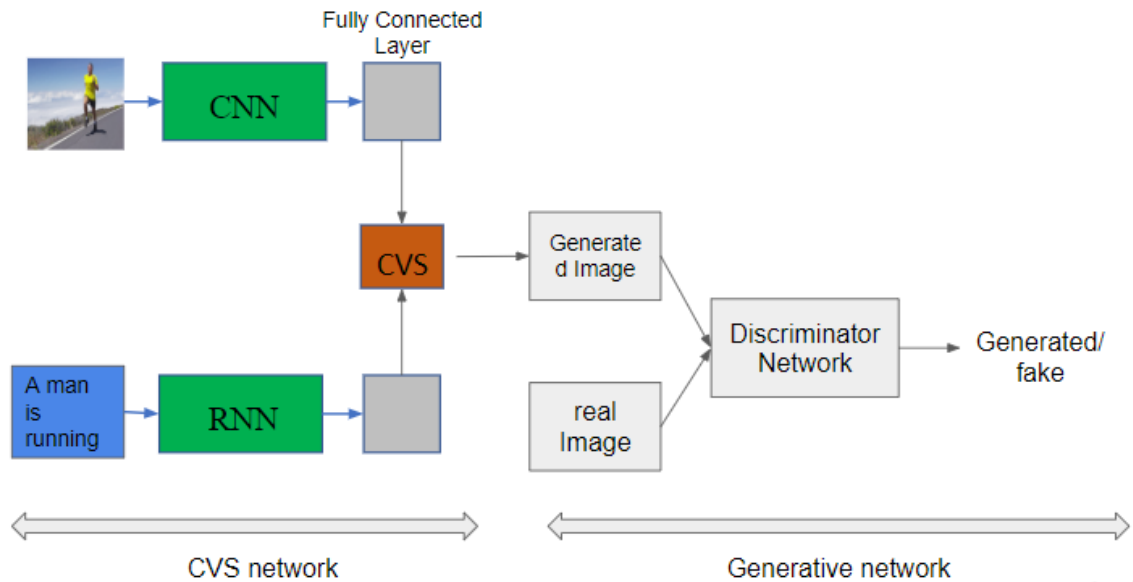


Figure 15 Model architecture

Figure 15 shows the overall architecture. The input to the model is a pair of images and text. These data are passed through the feature extraction network whereby the image is passed through the CNN model and text is passed through the RNN model. Once the feature is generated, it is passed through multiple fully connected layers before it is given to the loss function. The Loss function ensures that the data belonging to the same class is mapped closer to each other and the data belonging to a different class is mapped far apart from each other. The vector output of the last fully connected layer of both the data is passed to the generator network to generate a realistic-looking image. The generated image along with the real image is then passed to the discriminator network to classify the image as real or fake.

This work is mostly inspired by Peri et al. [33]. They use a captioning model after the CVS network to generate captions for the images and paraphrase the sentence input. We take this architecture a step further and use generative model along with the captioning model to generate realistic looking images.

3.3 Loss functions

The CVS network is trained using the triplet, cross-entropy and GAN loss functions. Triplet loss is used to bring positive pairs closer to each other and maximize the distance between negative pairs as discussed in section 2.6.A. Cross-entropy loss is used to maximize the inter-class distance as discussed in section 2.6.C. The overall loss is a combination of all three losses:

$$L_{cvs} = L_{tp} + L_{ce} + \alpha L_{GAN} \quad (3.3.1)$$

Where

- L_{cvs} : Loss for CVS network
- L_{tp} : (2.6.1) triplet loss
- L_{GAN} : (2.4.1) $Loss_D$ + (2.4.2) $Loss_G$
- α : scaling factor

$$L_{GAN} = \beta L_{G-txt} + \gamma L_{G-img} \quad (3.3.2)$$

Where:

- L_{G-txt} : Adversarial loss using text vector as an input
- L_{G-img} : Adversarial loss using image vector as an input

The generative network is trained by adding the adversarial loss generated by image and text vectors.

4.1 Datasets

We use datasets which have multiple modalities. We are only dealing with images and texts in our experiments.

A. Caltech CUB Dataset

The Caltech CUB dataset [16] has images of 200 different bird species. The dataset comes with 6033 images and 10 sentences associated with each image. Corresponding sentences describe the features of the bird in the image such as color, appearance, and shape. Figure 16 shows example images from the Caltech CUB dataset.

Table 1 Train and validation split for Caltech CUB dataset.

	Caltech CUB dataset
Train, Test	8855, 2933
Categories	150, 50



Figure 16 examples from Caltech cub dataset. [61]

B. Oxford-102 dataset

The Oxford-102 dataset [15] has 102 different flower categories. Each class consists of 40 and 258 samples. Each image is associated with five sentences describing the features of the flower in the image.

Table 2 Train and validation split for Oxford-102 dataset.

	Oxford-102 dataset
Train, Test	7034, 1155
Categories	102



Figure 17 Examples from Oxford-102 dataset. [61]

4.2 Implementation

This work is implemented in Tensorflow and training on Nvidia GPUs. We use pre-trained Resnet-152 [41] architecture to extract features from the images. The last layer before softmax is used as a feature vector for the images. The size of the image vector is 2048. We use the pre-trained SkipThought [40] model to extract features from texts. The size of the text vector is 4096. We use two fully connected layers after the feature extraction of size 2048. The size of the image text vectors are fixed as they are extracted from a pre trained networks. We experimented with the size of the fully connected layers after the feature extraction part such as 512, 1024 and 2048 and, found 2048 to be giving good results. The reason being 2048 vector can hold more contextual information as compared

to low dimension vector. This can be seen from the results. We use StackGan model to generate images from the vector input.

4.3 Evaluation metrics

Evaluation metrics allow us to evaluate the performance of the model given the dataset. We use two metrics to evaluate the overall performance of the model.

- Mean Average Precision (mAP) – to evaluate the retrieval performance
- Inception score – to evaluate the generated images

A. Mean Average Precision (mAP)

We use mean average precision (mAP) to evaluate the performance of our model.

Equation (4.3.1) describes the formula to calculate the

$$AP = \frac{1}{N} \sum_{n=1}^K (p(r).rel(r)) \quad (4.3.1)$$

Where:

- N is the number of samples retrieved
- K is the number of queries
- $p(r)$ is precision at r
- $rel(r)$ is a flag indicating if the result retrieved is a match or not

B. Inception score

Evaluating the performance of the generative model is a difficult task. We use inception score [43] to evaluate the performance of the generative model quantitatively.

$$I = \exp(E_x D_{KL}(p(y|x)||p(y))) \quad (4.3.2)$$

Where:

- x denotes generated image

- y is the label predicted by inception model [42]
- D_{KL} is KL divergence

Good models should generate diverse images that are meaningful. Therefore, the KL divergence between the marginal probability distribution $p(y)$ and $p(y|x)$ should be high. We take the pre-trained model for COCO [44] dataset. We then fine-tune the model for Caltech CUB and Oxford-102 datasets. We take a large number of samples to evaluate the model (30k randomly selected samples). The inception score calculation involves passing generated images to the inceptionv3 model to calculate conditional probability for each generated images $p(y|x)$. The marginal distribution is calculated using the average of conditional probability for all the images $p(y)$.

$$D_{KL} = p(y|x)(\log(p(y|x)) - \log(p(y))) \quad (4.3.3)$$

Equation (4.3.3) is used to calculate the KL divergence between $p(y|x)$ and $p(y)$.

The inception score ranges from 1.0 to the number of classes in the dataset.

Chapter 5

Results and Analysis

In this section we will discuss the performance of the model discussed in Chapter 3 on the datasets discussed in Chapter 4.

5.1 Results

A. Retrieval output (Caltech CUB dataset)

We calculate mAP@50 for both images to text retrieval and text to image retrieval. For example, given an image input, the model will retrieve the closest text to the image from the dataset and given a text input, the model will retrieve the closest image to the text from the dataset. We perform zero-shot retrieval on the Caltech cub dataset. We train the model on 150 categories and evaluate the performance of the model on the remaining 50 unseen categories. The goal of this test to see the robustness of the model on the unseen categories. The numbers inside the square brackets next to the two G-CVS entries represent the embedding dimensions used.

Table 3 mAP@50 score for Caltech CUB dataset.

Method	img2txt	txt2img
Bow [63]	44.1	39.6
Word2Vec [64]	38.6	33.5
Word CNN [65]	51.0	43.3
Word CNN-RNN [66]	56.8	48.7
GMM-HGLMM [67]	36.5	35.6
Latent Co-attention [68]	61.5	57.6
CVS+AA [70]	58.9	56.2
G-CVS (ours) [1024]	54.5	53.2
G-CVS (ours) [2048]	55.6	54.8

B. Generative output (Caltech CUB dataset)

The vector output of the CVS network is passed to the generative network. Figure 18 shows the randomly sampled images from the dataset. Figure 19 shows the generated images using the image vector and Figure 20 shows the generated images using text vector.

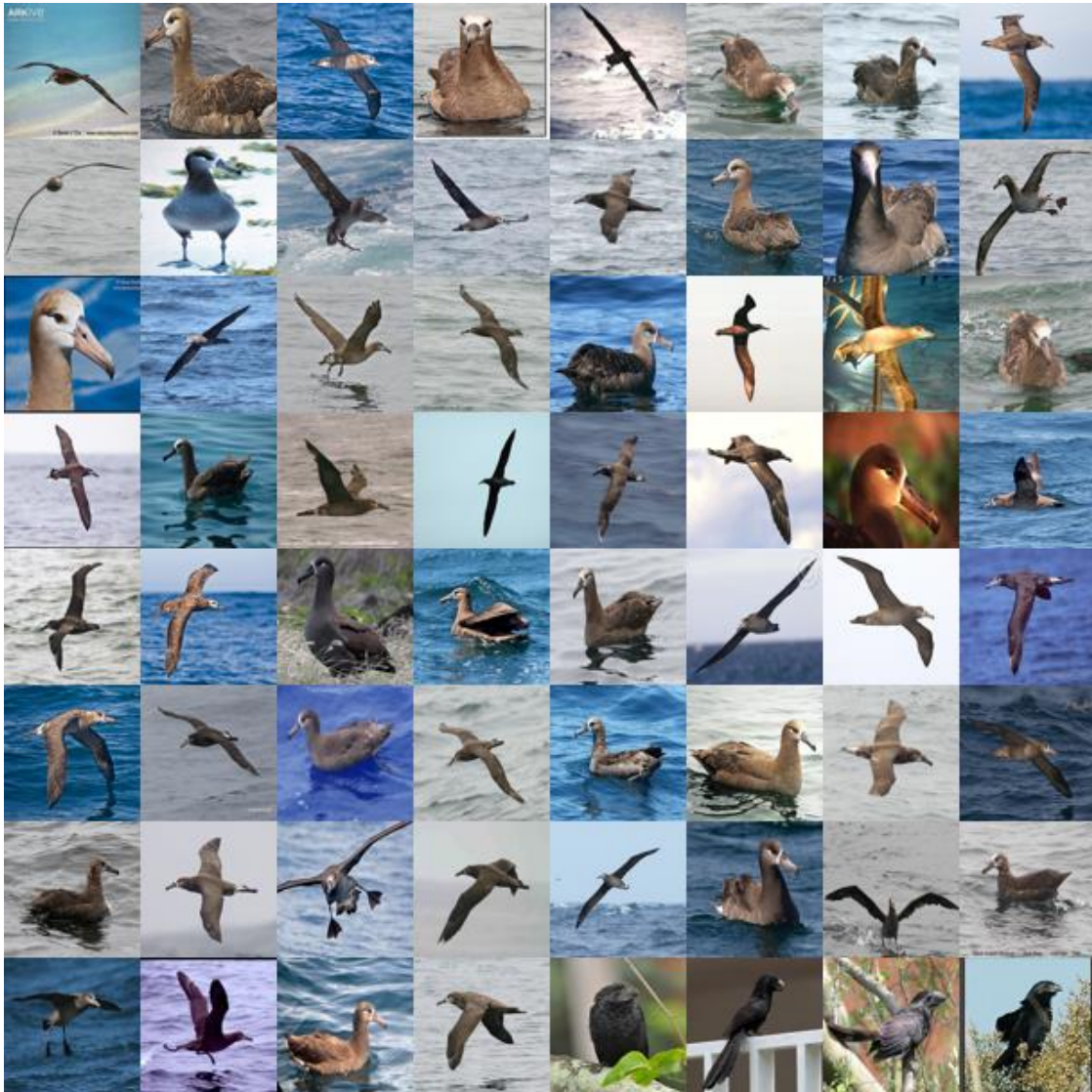


Figure 18 Sampled input images from the dataset.

Captions of top 10 sampled input images:

- 1: a small bird with very long wings, and a large bill.
- 2: a bird with a large, hooked bill, white superciliary and cheek patch, brown crown, and brown body.
- 3: this is a brownish gray bird with large wings and a long, hooked bill.

4: this large seabird is mostly a light brown with a long, hooked bill, dark oval eyes and a white crown and throat.

5: this bird is nearly all gray with a blunt beak.

6: a mostly brown bird with a white face and a brown beak.

7: this bird is nearly all brown with a hooked bill.

8: this bird is predominately grey with a large, curved, grey beak.

9: this bird is almost completely dark gray, it has a light gray crown.

10: a greyish-blue colored bird with a white-grey face, and big blue feet.



Figure 19 input: image vector, output: image.



Figure 20 input: text vector, output: image.

Table 4 Inception scores for Caltech CUB dataset.

Method	Inception score
GAWWN [47]	3.62
StackGAN [28]	3.82
AttnGAN [46]	4.36
G-CVS (ours)	5.36

C. Retrieval output (Oxford-102 dataset)

Table 5 mAP@50 score for Oxford-102 dataset.

Method	img2txt	txt2img
Bow [63]	57.7	57.3
Word2Vec [64]	54.2	52.1
Word CNN [65]	60.7	56.3
Word CNN-RNN [66]	65.6	59.6
GMM-HGLMM [67]	54.8	52.8
Latent Co-attention [68]	68.4	70.1
CVS+AA [70]	66.1	67.1
G-CVS (ours) [1024]	61.5	60.7
G-CVS (ours) [2048]	62.2	61.1

D. Generative output (Oxford-102 dataset)



Figure 21 Sampled images from Oxford-102 dataset.

Captions of top 10 sampled input images:

- 1: this flower has four large rounded light purple petals that are yellow in the middle with stamen.
- 2: the flower has a smooth purple petal with white pollen tubes and yellow anther
- 3: this flower has big purple petals and white filaments with a yellow anther.
- 4: this flower has petals that are dark pink with a light green center.

5: the flower shown has yellow pistil with large pink petals

6: the flower has purple petals with a green center and white pollen tubes

7: this flower has petals that are pink with purple lines

8: leaves are green in color, petals are light pink in color

9: this flower has white petals and yellow pistil as its main features

10: the petals on this flower are a flat purple with a white pistil in the center



Figure 22input: image vector, output: image.



Figure 24 input: text vector, output: image.

The image generated for image vector looks better than the text vector. This can be because the image vector has more insightful information about the image as compared to the text vector which helps the model to generate more realistic looking images.

Table 6 Inception scores for Caltech CUB dataset.

Method	Inception score
GAWWN [47]	-
StackGan [28]	3.26
AttnGan [46]	-
G-CVS (ours)	3.29




5.2 Analysis

In this section, we will analyze the output of the generative models.

A. Analysis for oxford-102 dataset

1) Good examples

Table 7 Analysis for oxford-102 dataset (good examples).

Inputs	Generated image outputs
 <p>this flower has four large rounded light purple petals that are yellow in the middle with stamen.</p>	 







 <p>the flower is pale pink and has a white center and yellow stamen.</p>	 
 <p>the flower with petals that are fused together supporting and surrounding the yellow stamen in the center.</p>	 

Table 7 shows some example output from the Oxford-102 dataset. The left column represents the input pair of image and caption and the right column represents the images generated given the image or text input. The images generated by the generative model are almost similar. This shows that the vectors of the same class projected on the CVS are closely clustered. So, the distance between the vectors is minimal which allows the generator network to generate similar images.

2) Bad examples

Table 8 Analysis for oxford-102 dataset (bad examples).







Inputs	Generated image outputs
 <p>this flower has white petals and yellow pistil as its main features.</p>	 
 <p>the flower is pale pink and has a white center and yellow stamen.</p>	 







Table 8 shows some bad samples generated by the model. In the first case the model wrongly interprets the yellow pistil keyword and uses that to generate a flower that has yellow-colored petals. In the second case the model again misinterpreted the inputs and

generated the image of a different class. These can be because the model failed to extract the contextual features from the image and text.

B. Analysis for Caltech CUB dataset

1) Good examples

Table 9 Analysis for caltech CUB dataset (good examples).

Inputs	Generated image outputs
 <p>this bird has a very long beak is gray and white in color and has very long wings and short little legs.</p>	 
 <p>a small bird that is golden yellow with traces of brown on the wings.</p>	 




 <p>a grey bird with webbed feet and a brown head.</p>	 
---	--

Table 9 shows some example output from Caltech-CUB dataset. The left column represents the input pair of image and caption and the right column represents the images generated given the image or text input. The images generated by the generative model go along with the inputs and the generated images from image and text look similar which shows that the vectors are projected close to each other in the latent vector space.

2) Bad examples

Table 10 Analysis for caltech CUB dataset (bad examples).

Inputs	Generated image outputs
--------	-------------------------







 <p>this large seabird is mostly a light brown with a long, hooked bill, dark oval eyes and a white crown and throat.</p>	 
 <p>light tan colored bird with a white head and an orange beak.</p>	 

Table 8 shows some bad samples generated by the model. In both the cases model completely misinterprets the inputs which leads to poor feature extraction.

C. Analysis for captioning model

We experimented with adding a captioning model along with the generative model. Figure 23 shows the model architecture after adding the captioning model. The image captioning model takes the last layer of the CVS network as input and outputs the caption for both image and text input. Similar to the generative network, the last fully connected layer is used as the feature vector for the inputs. We used the basic image captioning model with one layer of LSTM. This experiment is to show the use of CVS embedding to generate captions for the given inputs.

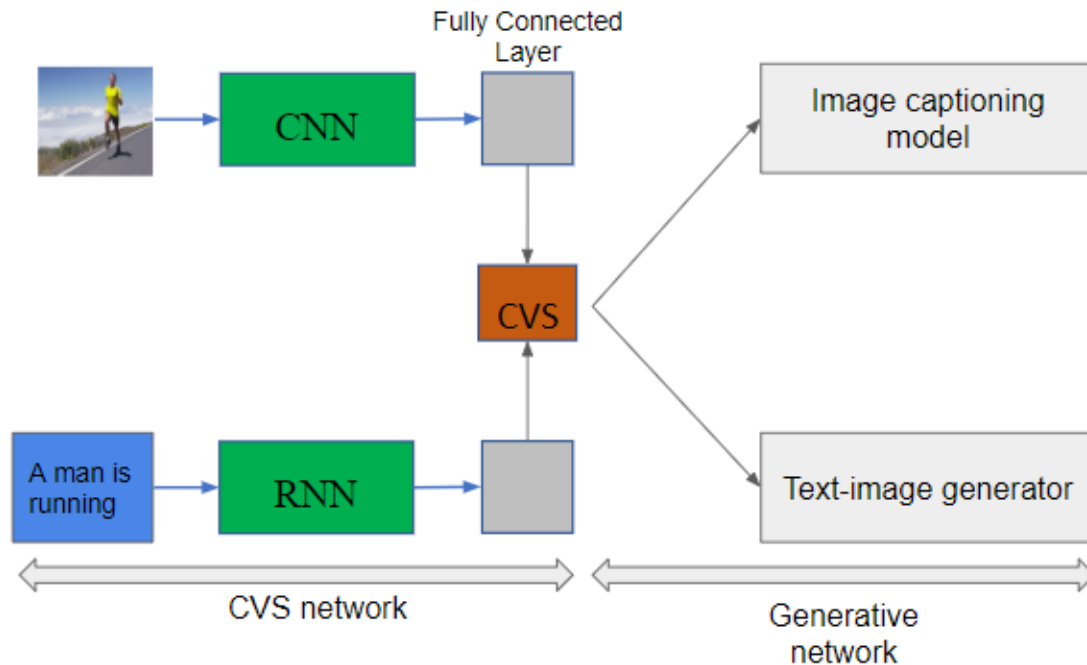


Figure 25 Adding captioning model along with generative network.

Table 11 shows the output of the captioning model. The left column shows the inputs and the right column is the output captions. This simple model can capture the colors in the image and caption input. The input captions mostly start with ‘this flower’ which is reflected in the generated captions as well.

Table 11 Analysis for captioning model (Oxford-102 dataset).





Inputs	Generated captions
 <p>this flower has white petals and yellow pistil as its main features.</p>	<p>this flower has petals that are white with yellow.</p> <p>this flower has petals that are white with yellow pistil.</p>
 <p>the flower is pale pink and has a white center and yellow stamen.</p>	<p>the flower has petals that are pink.</p> <p>the flower has petals that are white with yellow.</p>

Table 12 shows some bad captions generated by the captioning model. These poor quality captions may be because the model used to generate the captions is a very basic LSTM model with only one layer. More complex models presumably would generate better results.

Table 12 Analysis for captioning model (bad example).

Inputs	Generated captions
 <p>this flower has white petals and yellow pistil as its main features.</p>	<p>this flower has petals that are white with yellow.</p> <p>this flower has petals that are white with yellow pistil.</p>
 <p>the flower is pale pink and has a white center and yellow stamen.</p>	<p>the flower has petals that are pink.</p> <p>the flower has petals that are white with yellow.</p>

6.1 Conclusions

This work develops a unified model which shows the performance on diverse tasks such as cross modal retrieval and image generation. This work exhibits training multiple models for different task in a joint fashion. This thesis work demonstrates the use of CVS for image generation task. Even though the CVS is used to bring the two vectors from different modalities closer to each other in the vector space, this work proves that the vector from the CVS can be used to generate realistic looking images.

6.2 Future Work

We demonstrate the performance of the model on image and text modalities. Further extension of this work is possible. Some possible extensions are:

- Adding complex image captioning model to generate more detailed caption for the image input and paraphrase sentence for the text input.
- Extending this to other modalities such as audio and video and generate images based on the vector generated by the CVS.

Bibliography

- [1] Lee, Kuang-Huei, et al. "Stacked cross attention for image-text matching." Proceedings of the European Conference on Computer Vision (ECCV). 2018.
- [2] Faghri, Fartash, et al. "Vse++: Improved visual-semantic embeddings." arXiv preprint arXiv:1707.05612 2.7 (2017): 8.
- [3] Malinowski, Mateusz, Marcus Rohrbach, and Mario Fritz. "Ask your neurons: A neural-based approach to answering questions about images." Proceedings of the IEEE international conference on computer vision. 2015.
- [4] Rasiwasia, Nikhil, et al. "A new approach to cross-modal multimedia retrieval." Proceedings of the 18th ACM international conference on Multimedia. ACM, 2010.
- [5] Wang, Bokun, et al. "Adversarial cross-modal retrieval." Proceedings of the 25th ACM international conference on Multimedia. ACM, 2017.
- [6] Wei, Yunchao, et al. "Cross-modal retrieval with CNN visual features: A new baseline." IEEE transactions on cybernetics 47.2 (2016): 449-460.
- [7] Zhen, Liangli, et al. "Deep Supervised Cross-Modal Retrieval." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.
- [8] Sanakoyeu, Artsiom, et al. "Divide and Conquer the Embedding Space for Metric Learning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.
- [9] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
- [10] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
- [11] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

- [12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv:1207.0580 [cs], Jul. 2012.
- [13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.
- [14] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "Facenet: A unified embedding for face recognition and clustering," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 815–823.
- [15] Nilsback, Maria-Elena, and Andrew Zisserman. "Automated flower classification over a large number of classes." 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing. IEEE, 2008.
- [16] Welinder P., Branson S., Mita T., Wah C., Schroff F., Belongie S., Perona, P. "Caltech-UCSD Birds 200". California Institute of Technology. CNS-TR-2010-001. 2010.
- [17] Jeffrey Donahue et al., "Long-term recurrent convolutional networks for visual recognition and description," in CVPR, 2015, pp. 2625–2634.
- [18] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Rich Zemel, and Yoshua Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in International conference on machine learning, 2015, pp. 2048–2057.
- [19] S. Sah, S. Kulhare, A. Gray, S. Venugopalan, E. Prud'Hommeaux, and R. Ptucha, "Semantic Text Summarization of Long Videos," in 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), 2017, pp. 989–997.
- [20] Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).
- [21] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

- [22] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).
- [23] Sah, Shagan, et al. "Multimodal reconstruction using vector representation." 2018 25th IEEE International Conference on Image Processing (ICIP). IEEE, 2018.
- [24] Karpathy, Andrej, and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
- [25] Shi, Haichao, et al. "Image captioning based on deep reinforcement learning." Proceedings of the 10th International Conference on Internet Multimedia Computing and Service. 2018.
- [26] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.
- [27] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein gan." arXiv preprint arXiv:1701.07875 (2017).
- [28] Zhang, Han, et al. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.
- [29] Zhang, Han, et al. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.
- [30] Singh, Monit Shah, et al. "Transforming sensor data to the image domain for deep learning—An application to footstep detection." 2017 International Joint Conference on Neural Networks (IJCNN). IEEE, 2017.
- [31] Zhou, Tao, et al. "Feature learning and fusion of multimodality neuroimaging and genetic data for multi-status dementia diagnosis." International Workshop on Machine Learning in Medical Imaging. Springer, Cham, 2017.

- [32] Lee, Sheng Long, Mohammad Reza Zare, and Henning Muller. "Late fusion of deep learning and handcrafted visual features for biomedical image modality classification." *IET Image Processing* 13.2 (2018): 382-391.
- [33] Hardoon, David R., Szedmak, Sandor R., and Shawe-taylor, John R. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16:2639–2664, 2004.
- [34] X. Xu, L. He, H. Lu, L. Gao, and Y. Ji, "Deep adversarial metric learning for cross-modal retrieval," *World Wide Web*, vol. 22, no. 2, pp. 657–672, Mar. 2019.
- [35] L. Zhen, P. Hu, X. Wang, and D. Peng, "Deep Supervised Cross-Modal Retrieval," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 10394–10403.
- [36] P. Anderson *et al.*, "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6077–6086.
- [37] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He, "Stacked Cross Attention for Image-Text Matching," presented at the Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 201–216.
- [38] He, Kaiming, et al. "Mask r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2017.
- [39] Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *Computer vision and pattern recognition, 2006 IEEE computer society conference on*. Vol. 2. IEEE, 2006.
- [40] Kiros, Ryan, et al. "Skip-thought vectors." *Advances in neural information processing systems*. 2015.
- [41] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- [42] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- [43] Salimans, Tim, et al. "Improved techniques for training gans." *Advances in neural information processing systems*. 2016.

- [44] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.
- [46] Xu, Tao, et al. "Attngan: Fine-grained text to image generation with attentional generative adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
- [47] Reed, Scott E., et al. "Learning what and where to draw." Advances in neural information processing systems. 2016.
- [48] Wang, Bokun, et al. "Adversarial cross-modal retrieval." Proceedings of the 25th ACM international conference on Multimedia. 2017.
- [49] Wei, Yunchao, et al. "Cross-modal retrieval with CNN visual features: A new baseline." IEEE transactions on cybernetics 47.2 (2016): 449-460.
- [50] "Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques | Request PDF," ResearchGate. [Online]. Available: https://www.researchgate.net/publication/324165524_Detection_and_Tracking_of_Pallets_using_a_Laser_Rangefinder_and_Machine_Learning_Techniques. [Accessed: 11-Oct-2018].
- [51] <https://github.com/crisbodnar/text-to-image>
- [52] https://software.intel.com/sites/products/documentation/doclib/daal/daal-user-and-reference-guides/daal_prog_guide/GUID-9B434D4F-C723-4191-9A88-69148C75A3F1.htm
- [53] https://embarc.org/embarc_mli/doc/build/html/MLI_kernels/pooling_avg.html
- [54] <https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce>
- [55] <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [56] <https://www.data-blogger.com/2017/08/27/gru-implementation-tensorflow/>
- [57] <https://www.sentiance.com/2018/05/03/venue-mapping/>
- [58] <https://pathmind.com/wiki/generative-adversarial-network-gan>
- [59] <https://omoindrot.github.io/triplet-loss>
- [60] Gopalakrishnan, Sabarish, "Vector Spaces for Multiple Modal Embeddings" (2019). Thesis. Rochester Institute of Technology.

- [61] Li, Ao-Xue, Ke-Xin Zhang, and Li-Wei Wang. "Zero-shot Fine-grained Classification by Deep Feature Learning with Semantics." *International Journal of Automation and Computing* 16.5 (2019): 563-574.
- [62] Peri, Dheeraj Kumar, "Multi-modal learning using deep neural networks" (2018). Thesis. Rochester Institute of Technology. Accessed from.
- [63] Zhang, Yin, Rong Jin, and Zhi-Hua Zhou. "Understanding bag-of-words model: a statistical framework." *International Journal of Machine Learning and Cybernetics* 1.1-4 (2010): 43-52.
- [64] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.
- [65] Zhang, Xiang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification." *Advances in neural information processing systems*. 2015.
- [66] Wang, Jiang, et al. "Cnn-rnn: A unified framework for multi-label image classification." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- [67] Klein, Benjamin, et al. "Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation." *arXiv preprint arXiv:1411.7399* (2014).
- [68] Li, Shuang, et al. "Identity-aware textual-visual matching with latent co-attention." *Proceedings of the IEEE International Conference on Computer Vision*. 2017.
- [69] Peri, Dheeraj, Shagan Sah, and Raymond Ptucha. "Show, Translate and Tell." *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019.