National College of Ireland

Image Captioning: Capsule Network vs CNN approach

MSc Research Project
Data Analytics

Jaydeep Deka
Student ID: x18134246

School of Computing
National College of Ireland

Supervisor: Dr. Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Jaydeep Deka
Student ID:	x18134246
Programme:	Data Analytics
Year:	2019
Module:	MSc Research Project
Supervisor:	Dr. Vladimir Milosavljevic
Submission Due Date:	12/12/2019
Project Title:	Image Captioning: Capsule Network vs CNN approach
Word Count:	8214
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	11th December 2019

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Image Captioning: Capsule Network vs CNN approach

Jaydeep Deka
x18134246

Abstract

Given an image, generating a relevant sentence to describe the objects and the activities is an active research area popularly termed as 'Image Captioning'. The problem requires the integration of both computer vision and natural language processing. Different approaches have been proposed over the last decade which used neural networks to achieve state-of-the-art results. Many of the recent researches have used an encoder-decoder framework that uses Convolutional Neural Network (CNN) for image feature extraction. However, CNN fails to retain information on spatial hierarchy and lacks rotational invariance. These drawbacks of CNN are addressed in a comparatively new neural network called Capsule Network (CapsNet). This research takes a novel approach in the implementation of an image captioning solution using CapsNet as the image feature extractor. There are six different models trained using both CapsNet and CNN on the Flickr8k dataset and evaluated using BLEU-(n) scores. The experiments have shown convincing results from CapsNet based solution considering much smaller size than CNN. The BLEU-1 score of the CapsNet based solution is 0.536 compared to 0.534 of the CNN based solution.

Keywords— Image Captioning, Encoder-decoder, CNN, Capsule Network, BLEU-(n), Flickr8k

1 Introduction

Image captioning is an application of machine learning which involves computer vision and natural language processing to describe an image in a sentence. In a study done by Fei-Fei et al. (2007), it was observed that humans are capable of describing an image in words very easily. Developing software capable of generating image description, which is both syntactically and semantically correct is a very complex problem. Different approaches were taken by the researchers from a period of almost a decade ago (Farhadi et al. (2010)). Active researches are going on to develop production-ready solutions. An image captioning solution will enhance the experience of human-computer interaction in many use-cases. The primary benefit of such a system will be to the visually challenged people, where an image captioning solution can be developed for real-time description generation which will assist the person while traveling around. This kind of system can also be helpful in automatic remote surveillance, driver-less cars, and image-retrieval, etc.

In recent years sophisticated algorithms like Convolutional Neural Network(CNN), Recurrent Neural Network(RNN), etc have been able to deliver remarkable performances in several tasks. Motivated from such studies many researchers proposed solutions for image captioning

that use CNN, RNN or LSTM. The encoder-decoder framework proposed in the studies by Vinyals et al. (2014) Kiros et al. (2014) involving CNN and RNN was able to demonstrate significant improvements from the earlier approaches and has been a popular framework thereafter for further research in the task. CNN architectures are specially designed set of neural networks for computer vision as it is capable of learning features of images in hierarchical and shared formation. For sequential tasks like machine translation, sentence generation, RNN architectures are well-suited. In image description generation solutions, a CNN architecture works as the visual feature extractor or the image encoder to extract the features from an image which is then used as the input to an RNN architecture for the generation of a sentence also called the decoder of the framework.

Although CNN is widely used in problems involving image data and remarkable performance is attained in such problems, the implementation algorithm of CNN has two major shortcomings. CNN uses pooling layers for information routing which causes information loss and the second drawback is its incapability in expressing viewpoint invariance. In a study done by Hinton et al. (2011), a new set of neural network representations called Capsule Network was introduced which addressed the drawbacks of CNN implementation architecture. Capsule Network uses dynamic routing among the activity vectors also known as Capsules, which is responsible for preserving the spatial information of the objects. In the last couple of years, capsule network architecture has been applied in different experiments to compare its performance against state-of-the-art solutions which used CNNs. In different studies capsule network solutions outperformed CNN architecture on image datasets like MNIST, CIFAR10, Fashion MNIST both computationally as well as in overall accuracy (Palvanov and Im Cho (2018), Edgar et al. (2018a), Wang et al. (2018)).

This paper takes a novel approach in developing an encoder-decoder framework for image caption generation using Capsule Network as the visual feature extractor. Considering the time and resource limitation, Flickr8k is used in this research as it is relatively smaller than the other public datasets and popular for rapid prototyping. For evaluation and comparison with CNN based approaches, another model using VGG16 features is also developed. In both the models, the text data is transformed the same way, the final decoder model architecture is kept the same and trained for the same number of epochs. This paper tries to answer the question **whether an encoder-decoder model with Capsule Network as an image feature extractor can outperform a CNN based solution** by evaluating the trained models on BLEU metric.

The paper is organized as follows: The next Section 2 discusses the related literature of different approaches in image captioning and capsule network, followed by the development methodology of this research project in Section 3. Following next, Section 4 explains the detailed end to end implementation stages of the project. Section 6 describes the evaluation of the proposed approach on the chosen metric and discussion on the findings. The conclusion derived from this research and future scope has been discussed in the Section 7.

2 Related Work

In the past few years, many approaches were taken in developing image captioning models using traditional machine learning algorithms like SVM, KNN, etc to complex algorithms like a neural network. The recently proposed methods for image captioning use CNN, RNN and LSTM in an encoder-decoder like framework. This section discusses related literature in the field of image captioning and capsule network which were referred to in developing this research project.

- **Initial approaches in Image Captioning:** The initial studies for this problem drawn inspiration from the researches related to how human perceives the surrounding at its first glance. In a study by Fei-Fei et al. (2007), it was found that humans are extremely quick in deciphering the feature level details. The authors suggested that, for a rich text

generation, extracting the visual features is very important. In the early days of research in this area, Farhadi et al. (2010), Hodosh et al. (2013), Ordonez et al. (2011) followed a retrieval-based approach where the visual features were first extracted and based on that sentences were predicted from a predefined set of sentences. These approaches used 'Meaning Space', an intermediate projection of both images and sentences. To extract features from the images most of the researches used traditional machine learning algorithms like SVM, KNN and highly relied on predefined templates of sentences for prediction. Though these studies helped image captioning gain attention from the researchers, the old approaches became obsolete in the next couple of years. Bai and An (2018) in their survey mentioned that such approaches had to compromise with the fluency of the generated sentences due to hard-coded feature engineering and rigid templates. However, these studies inspired the researchers to execute the same strategies with advanced algorithms for better results.

- **Neural Network based approaches in Image Captioning:** In the subsequent years, the retrieval-based and template-based strategies were augmented and reattempted using deep neural network approaches. Socher et al. (2014), Karpathy et al. (2014) used CNN for visual feature extraction and Dependency-Tree Recursive Neural Network for sentence retrieval. In both of the researches, CNN models pretrained on ImageNet were used and they were able to provide significant performance gain in extracting complex features. Same as earlier approaches, a meaning space or an intermediate space encapsulated the query image and sentence similarities. The features extracted with the help of CNN were used to retrieve relevant sentences from the meaning space. Application of neural network helped in learning complex features that were not possible otherwise with the traditional algorithms and hence enhanced the quality of generated sentences. These studies provided an insight that pretrained models can be used for feature extraction in image captioning tasks.

  Machine translation is a similar kind of task where a source sentence is converted into another sentence in a different language. Motivated by performance gain in machine translation using the multimodal approach, researchers proposed different multimodal solutions for image captioning. Multimodal solutions have used CNN for visual feature extraction while RNN, LSTM based language models to map the extracted image features into a common space. Kiros et al. (2014) first approached image captioning using a multimodal structure. The method introduced a multimodal space of image features and text, and a language model decoder to predict word from that space. This is one of the earlier works that was inspired by neural machine translation and adopted the encoder-decoder framework in image captioning. Kiros et al. (2014) used VGG-19 and LSTM and was able to set new state-of-the-art results in image captioning using the Flickr8k and FLickr30k datasets. Another multimodal approach was taken by Mao et al. (2014), where RNN was used to build the language model. The words were generated based on word prefix and image. The model used deep RNN and CNN which route information through a multimodal layer. This approach was termed as multimodal-RNN and was validated in Flickr8k, Flickr30k, and MS COCO. Ma et al. (2015) used a multimodal CNN (m-CNN) structure where two separate CNN architectures were used to encode image content and to learn image and sentence representation through matching CNN. The matching CNN was used to learn inter-modal relations between semantic fragments of the words and the image. On Flickr30k and MSCOSO dataset the proposed approach could match state-of-the-art performance. Karpathy and Fei-Fei (2017) proposed a multimodal structure that could generate sentence descriptions from images based on regions. This was an early work similar to the attention-based mechanism which focused on learning the inter-modal correspondence between the images and their description. The 'alignment

model' as termed, used CNN and bidirectional-RNN to align both modalities through multimodal embedding. The alignment was further inferred by an m-RNN to generate descriptions based on the image regions. This approach was tested on Flickr8k, Flickr30k, and MSCOCO and was able to outperform the state-of-the-art solutions.

- **Encoder Decoder framework:** Similar to the approach byKiros et al. (2014), Vinyals et al. (2014) drawn inspiration from advances in computer vision and machine translation to tackle the image captioning task using an encoder-decoder framework. In machine translation, an RNN works as an encoder to encode the source sentence in a representation vector which is then used in the decoder RNN unit to generate the target sentence. In their approach Vinyals et al. (2014) used a deep CNN in place of the RNN to convert an image to a fixed-length vector which is termed as the 'encoder' part of the model. The model used state-of-the-art networks from both computer vision and language modelling to build the final model. The image features were first extracted using the CNN encoder and then used as the initial input to the language model. The resultant model, termed as 'Neural Image Caption' was fully trained using Stochastic-Gradient-Descent (SGD). The trained model was able to set new state-of-the-art benchmarks on Flickr30k and MSCOCO datasets. Donahue et al. (2015) proposed another model which used recurrent convolutional neural network for learning long-term dependencies. To do that the authors inserted the extracted image features along with the context word to the language model units at each time-stamp. By this way, the resultant model was able to generate long sentences. In a similar approach like Donahue et al. (2015), Jia et al. (2015) inserted the extracted image features into the LSTM cell for word prediction. The authors argued that the decoding of the features needs to balanced between the image content and text data. According to Jia et al. (2015), sometimes the generated sentences turn out to be irrelevant to the images as the image features are inserted into the language model at first and never considered later, which results in sentences drifting away from the main context. The authors implemented three different ways of inserting the image features into the model i.e. inserting the image features as the initial hidden state, together with texts in every timestamp and merging outputs of image feature and encoded text. The resultant models were able to perform on par with state-of-the-art solutions on Flickr8k and Flickr30k. Tanti et al. (2018) in their study, analyzed different approaches on how extracted features should be inserted in an image caption generator model and how it affects the quality of the outcome. Based on the methods already implemented by researchers over the years Tanti et al. (2018) divided the approaches as 'Inject' and 'Merge'.The authors mentioned that how the features are used in the modelling affects the performance of the models.

- **Attention based mechanism:** Incorporating the encoder-decoder framework in image captioning delivered significant results. However, these approaches do not distinguish well enough among the contents in the images. Images being subject to having a wide variety of objects in it, encoder-framework sometimes produces irrelevant sentences as pointed out by Jia et al. (2015). Motivated from the advancement in visual attention mechanism for object detection, similar approaches were also proposed for image captioning. Extending the encoder-decoder approach by Vinyals et al. (2014), the authors Xu et al. (2016) proposed an attention-based mechanism of caption generation. The proposed method developed an attentive model that was able to dynamically look at the image regions which have higher importance on the context of caption generation. Similar to other encoder-decoder frameworks, in this study, CNN was used as an image feature extractor and LSTM was used in the language model. But unlike others, The LSTM units used a context vector that stored a representational vector of the object's location in the context of the image. This allowed the memory units to remember the regions that can have certain objects in the image. This approach by Xu et al. (2016) was able to score .67 in the BLEU-1

score on the Flickr30k dataset. In another approach by You et al. (2016), the authors proposed a hierarchical approach. This approach employed a top-down and bottom-up approach for attending important 'Region of Interest' (ROI). The model weighted the regions based on importance with respect to the sentence. In the top-down phase, the objects in the images are recognized and based on the weights assigned, in the bottom-up phase the sentences are generated. Similar to this approach, Anderson et al. (2018) proposed a method using Faster R-CNN, a very popular algorithm in object detection, for this task to propose image regions. The approach by Anderson et al. (2018) was able to achieve BLEU-1 .802 on MSCOCO dataset. The results suggest that attention-based encoder-decoder approaches have performed remarkably well in image captioning tasks. However, the attention-based mechanisms are computationally expensive as it needs to undergo more complex calculation for region suggestion and weighting.

- **Capsule Network:** In all of the recent studies discussed above, CNN is the most commonly used visual feature extractor. CNN has performed remarkably well on visual data problems. CNN learns the features of images through abstraction. However, Hinton et al. (2011), Sabour et al. (2017) argued that despite CNN being so powerful, it has two major drawbacks.Hinton et al. (2011) mentioned that CNN lacks rotational invariance and fails to understand spatial relationships. The reason being to route information from lower layers to higher-level layers CNN uses pooling layers which only considers the maximum value out of a region. Though this operation helps CNN to increase the 'field of view', pooling layers cause information loss and hence it is not able to capture the relationship of the features in an image. In their study, Hinton et al. (2011) proposed a new concept with a different architecture of a neural network, where the presence of features are presented through vectors called Capsule. The new network was named as Capsule Network. However, from 2011 till a couple of years back, Capsule Network did not gain so much of attention as it was just a theoretical concept until 2017. In a recent study by Sabour et al. (2017) from Google Brain, the authors proposed a fully working Capsule Network which used Dynamic Routing for learning of its parameters. The authors mentioned that a capsule, which basically contains a group of neurons outputs activity vector to provide the probability of the presence of an entity or a part of it. The solution was tested using the MNIST dataset, which is for digit recognition. From the experiments of Sabour et al. (2017) it was found that the Capsule Network solution performs better than the existing CNN based solution even when there is an overlapping in the dataset. Edgar et al. (2018b) implemented a Capsule Network architecture to check how it performs on complex data like CIFAR10. The authors used an ensemble of seven different models which is lighter than the baseline CNN model and it was able to surpass the validation accuracy of the baseline model by 1.85%. In another study by Rajasegaran et al. (2019), the authors introduced a deeper capsule network 'DeepCaps1' which used a novel dynamic routing algorithm. The proposed architecture could beat the state-of-the-art results on CIFAR10, Fashion MNIST, and SVHN while reducing the size of the model by 68% than its CNN counterpart. Dynamic routing being computationally expensive, the authors avoided stacking up multiple Capsule units and introduced skip connections in between the capsule units for faster training and inference.

- **Inference:** From the above discussion, we have seen how the encoder-decoder framework has been implemented in the task image captioning in different approaches. The current state-of-the-art solutions for image description use CNN architectures and have been able to achieve much better results than the earlier approaches. However, as observed from the discussed literature in the last section Capsule Network has been able to show its potential in the field of computer vision. The proposed methods are lighter and faster than CNN. This makes us believe that in an encoder-decoder framework a lighter and

faster Capsule network can be an alternative for CNN as a visual feature extractor.

3 Methodology

For research and development of data mining projects, there are three methodologies widely followed i.e. Knowledge Discovery in Databases (KDD), Sample Explore Modify Model and Assess (SEMMA) and Cross-Industry Standard Process for Data Mining (CRISM-DM). As the objective of this project is to evaluate the performance of Capsule Network to CNN as an encoder, the KDD framework has been considered as the best-suited methodology. Unlike CRISP-DM where Deployment is the last stage, the KDD framework is limited to Evaluation only. Figure 1 illustrates the KDD framework modified in the context of this project. **The colored blocks signify the novelty part of this project.**

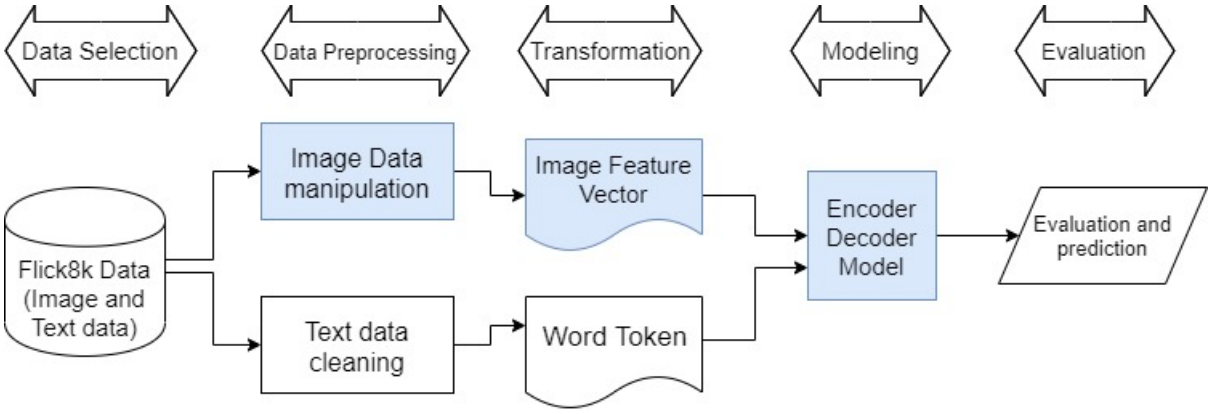


Figure 1: KDD methodology followed in the project

As the problem involves both text and image data the Data Preprocessing and Transformation stages are split. Both types of data are processed separately before fitting into the final model. Before the implementation of any project, it is important to create a road map about the flow, architecture, technologies, and algorithm, etc considering different constraints. This section gives an overview of the selected data, design specification and model structure for the experiments related to the study.

3.1 Data Selection

From the discussed literature in Related Work section, it was observed that Flickr8K, Flickr30K, and MSCOCO are the most used datasets for the development and evaluation of the proposed methods for image captioning. Considering the time and computing resource limitations, Flickr8k has been selected as the experiment dataset for this project. The dataset was published as an outcome of the study done by Hodosh et al. (2013). The authors explained that images were extracted from six different groups in Flickr and annotated using Amazon Mechanical Turk prioritizing the object activities over the context. The dataset contains 8000 images of daily activities of most people and animals described in five sentences each written in English. The dataset is publicly available for download by submitting a request form and allowed to use for non-commercial or academic researches. The approval e-mail is attached in the Appendix.

3.2 Design Specification

The encoder-decoder framework in image captioning draws inspiration from the implementation of neural language models. From the discussion in Related Work, it can be concluded that

encoder-decoder approaches have been able to perform remarkably well on this task. Such a model uses encoders to extract and transform features from the input images and texts and the decoder uses the transformed features for learning the internal representation (Vinyals et al. (2014), Kiros et al. (2014)). Inspired from the convincing results of the previous researches as discussed in the Related Work this project considers the encoder-decoder framework for the development of image captioning solution which uses Capsule Network as an image feature extractor. Figure 2 graphically illustrates the process flow of the development of the application:

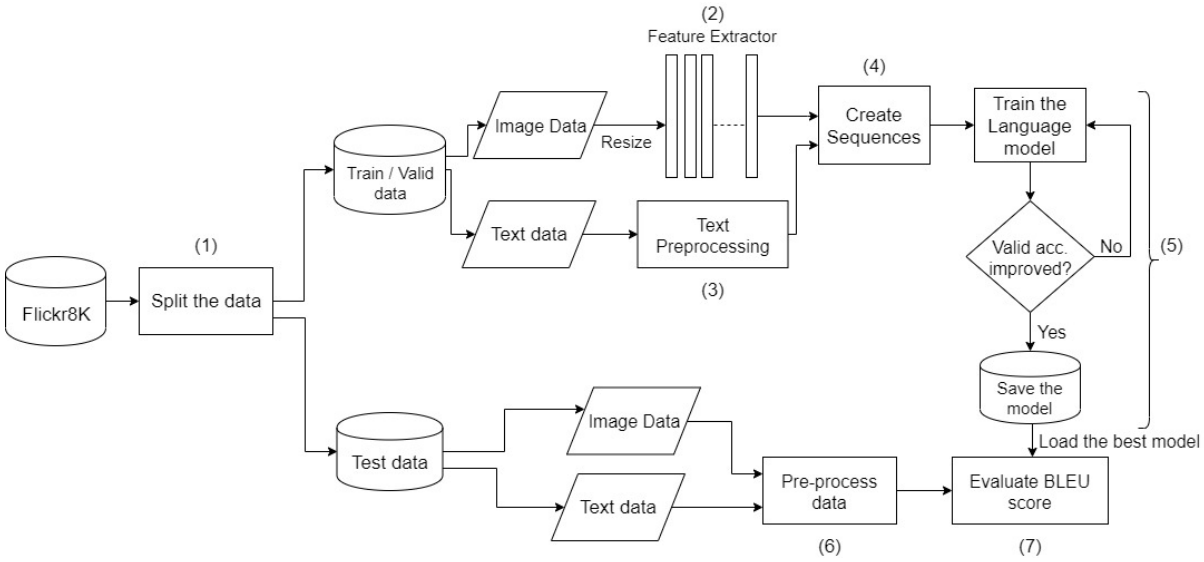


Figure 2: Design flow of the project

The indexes in the Figure 2 represents distinctive phases which are listed below:

1. Divide the dataset into train, validation and test subsets at random without repetition.
2. Extract the features from the images using the image feature extractor (Capsule Network or CNN).
3. Preprocess the text data i.e. converting to lower case, tokenize, padding with 'startseq' and 'endseq' etc.
4. Transform both the extracted features based on the architecture in training the language model.
5. Train the model for the specified epochs and save the model based on the model's performance on unseen data i.e validation dataset.
6. For evaluation, extract the features of the test images and make the predictions using greedy search and beam search.
7. Validate with BLEU score.

The implementation details of the above steps are explained in the Implementation and Evaluation section. Step-5 and Step-6 both undergo the prediction of words for the sentence generation. The process of word prediction follows a similar paradigm as Neural Machine Translation which done either by greedy search or beam search Freitag and Al-Onaizan (2017). In greedy search, in every step, only the most likely word is chosen while in beam search the combinations of different possibilities are considered. Based on the size of the beam length $'l'$

the cumulative probability is calculated and top $'l'$ combinations are kept for further processing. Freitag and Al-Onaizan (2017) explains that the higher the beam length longer is the time taken for sentence generation. This study uses a beam length of only 1 and 2 for the BLEU score evaluation.

3.3 Model Structure and Components

This study tries to evaluate the effectiveness of features extracted from the Capsule Network to the features extracted from CNN in image captioning. Apart from the image feature extraction step, the rest of the procedures follows the same paradigm for unbiased experimentation. The state-of-the-art solutions involving CNN for this task are challenging to reproduce considering the time and resource limitations. Hence, a baseline model using extracted features from CNN is first developed followed by another with extracted features from Capsule Network. In terms of using the image feature vector generated in Step-2 of Figure 2, there are different ways of fitting the processed dataset into the language model. A relative difference in the performance is observed based on how the extracted features are fed into the model. In a study done by Tanti et al. (2018), the authors evaluated two commonly used ways of inserting the extracted features into the language model, they are Inject and Merge architecture. This section focuses on explaining the model structures used and the key components used in the development of this project.

In the Inject architecture, the extracted image features are inserted along with the word into the RNN/LSTM cell as shown in Figure 4, which results in both the image and the text features together to be considered as the input prefix for the generation of next word. The RNN/LSTM cells in this architecture are trained to encode both the image and text features to one vector conditioned for the next generation.

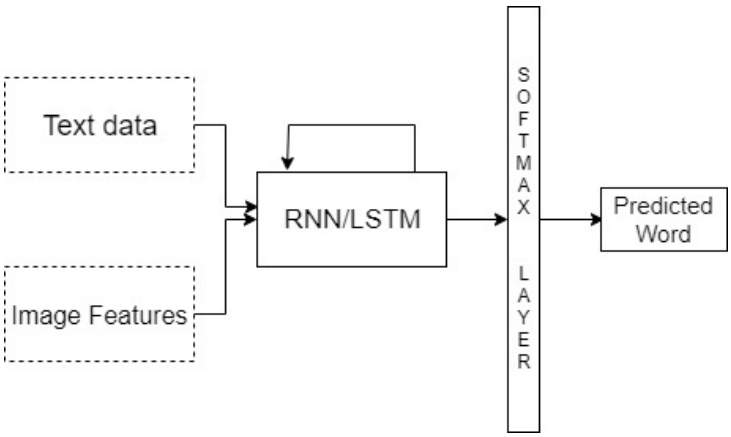


Figure 3: Inject Architecture Tanti et al. (2018)

In contrast to the Inject architecture, the RNN/LSTM cell in the Merge architecture is only responsible for encoding the text data as shown in Figure 5, which is later concatenated with the image features and together treated as an input to the subsequent layers for text predictions. On Flickr8k dataset the experiments in the study by Tanti et al. (2018), the Inject architecture provided relatively better performance than the Merge architecture with VGG19 as the image feature extractor.

For experimentation, both the inject and merge architectures have been developed in this study to find out which performs better in the case of features extracted from Capsule Network in comparison to CNN. The key components used in the development process of the image captioning application are CNN, Capsule Network and LSTM. The functionalities of the structures are explained below:

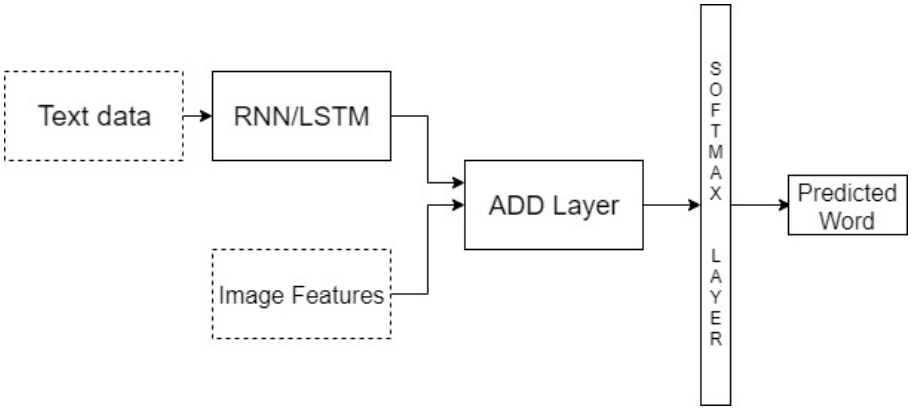


Figure 4: Merge Architecture

- **Convolutional Neural Network (CNN):** A Convolutional Neural Network or CNN is a special set of deep neural network architecture which is suitable for understanding features of an image dataset. CNN uses filters also called kernels which learn the shareable parameters by convolving around the image across the different axes or channels based on different hyperparameters like stride, padding, etc. In a series of dense convolutional layers, the model learns to generalize the image features from simple to complex. CNN uses the pooling layer in between the convolutional layers to reduce the number of parameters and information routing as shown in the Figure 6. The maxpooling operations

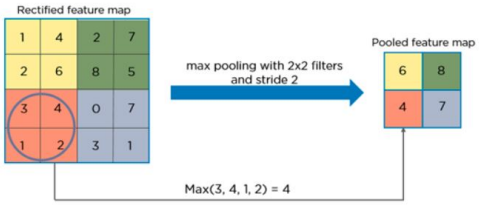


Figure 5: Maxpooling operation in CNN

check a matrix of pixels, often 2x2 and consider the maximum activation value in that region. This operation ensures that the presence of a feature is detected through a scalar output. The final output layer contains neurons equivalent to the number of classes in a dense layer. The implementation details of a CNN model to use it as a visual feature extractor is explained in the next section.

- **Capsule Network:** Capsule Network is another modified form of deep neural network which encapsulates the feature information in vector representation. Unlike CNN, in Capsule Network the feature information of the objects in an image is retained in vector forms termed as Capsules, which makes the capsule networks capable of detecting the presence of an object even after there are changes in the viewing angle. The probability of a feature is present in a capsule is determined by the output vector's length. Matrix multiplication with weight matrices is carried out to encode the feature relationship between lower-level capsules to higher-level capsules. The affine transformation on an input vector is calculated by multiplying it to a weight matrix, which ensures that information from a layer l passes to the next layers $(l+1)$ only if the features in $(l+1)$ has dependency on the feature of l. The learning algorithm of the weights is termed as 'Dynamic Routing' Sabour et al. (2017). The transformed vector undergoes a weighted sum calculation before non-linearity transformation 'Squash' is applied Sabour et al. (2017). The squash function squeezes the output vector length to be within 1. Where a value 1

being is a strong positive indicator of the presence of a feature and close to 0 being a weak indicator. Based on this vector, the same as CNN a dense layer outputs the probability of the classes.

- **Long Short Term Memory (LSTM):** From the literature, it was observed that most of the studies used the Recurrent Neural Network (RNN) as the decoder. RNN is well suited for sequence data, which means given a set of series of inputs it can make predictions until some end-condition is met. But RNN fails to generalize long term dependencies of the words generated in a sentence (Bengio et al. (1994) Pascanu et al. (2012)). LSTM is a modified version of LSTM to address this drawback with the usage of cell state Hochreiter and Schmidhuber (1997). The LSTM units contain three different gates namely input gate, forget gate and update gate. The input gate is responsible for figuring out which input needs modification considering the cell state, forget gate decides which part of the input block should be discarded and update gate decides the output based on the input and memory block. The first LSTM unit in a network takes feature input, which outputs a hidden active state and prediction which then again fed into the next LSTM cell for further predictions.

4 Implementation

This section elaborates on the end-to-end flow of this project implementation in different distinctive phases. The details about environment setup, data preparation, and transformation, modeling and training are discussed below:

4.1 Environment Setup

Training a model for image captioning is a hardware intensive task Vinyals et al. (2014). Deep neural network training involves complex mathematical calculations that require computing resources as well as libraries that facilitates carrying out the operations in parallel to yield faster experimentation. This project is developed using Keras 2.4.2 running on Tensorflow-GPU 1.13.0 backend with Python 3.6 in a standalone Conda environment. The reasons for choosing Keras are: it is an open-source framework with high-level APIs for Tensorflow operations, easy to implement for rapid prototyping and it provides classes, methods, and easy customizability. Windows 10 is used as the operating system on a Hardware platform of Nvidia GTX 1050Ti 4GB GPU, Intel Core-i7 processor and 16GB RAM with the required libraries installed for GPU support.

4.2 Data Preparation and Transformation

Data preparation is an essential stage for effective modelling. The raw data available is not always in the required form to fit into the model directly. For example image data can be of different sizes, text data can contain mixed cases, stopping words, etc. which create noises in the dataset. These noises cause difficulty for the model to generalize the data and often result in underfitting. Hence, pre-processing is done before modelling and training. After the pre-processing of the data, it is also required to transform the data in the form required by the model.

The Flickr8k dataset contains 8000 images containing different activities described in 5 different sentences for each of the images resulting in a total of 40000 sentences. The Figure 6 shows a sample image from the dataset with the available captions for that image. As the 1st step of Design Specification, the downloaded dataset is split into three subsets as training, validation, and testing at random in a ratio of 80%, 10%, and 10% respectively on image

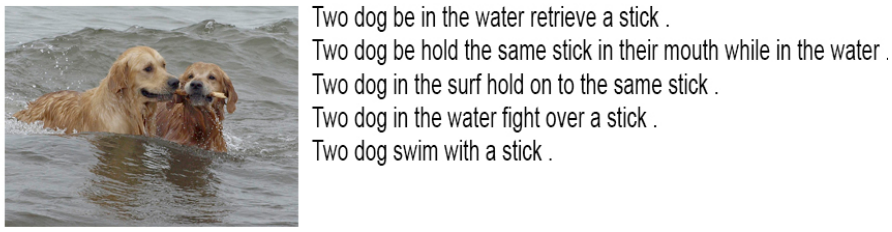


Figure 6: Sample Image and Description from the dataset

id. Before training the final language model, the data need to process and transformed into consistent vectors. As the dataset contains image and text data, both are processed separately. The steps followed during the process are discussed below:

- **Image Data:**

  As discussed in the Design Specification (2), to use images in the training of the language model, it is required to extract the features and transform them into vectors first. Training an image recognition model from scratch with the limited data available is complex as well as time-consuming Vinyals et al. (2016). In a survey by Hossain et al. (2018) on image captioning using deep learning, the authors found that most of the state-of-the-art solutions used pretrained CNN architectures for image feature extraction. In this study, we have followed a similar approach to using pretrained models for image feature extraction. In such approaches, the last layer of the network is removed which is used for classification purposes otherwise and the activation output of the second last layer is used as a feature vector. The flow of generating the feature vector from an image is illustrated in the Figure 7.

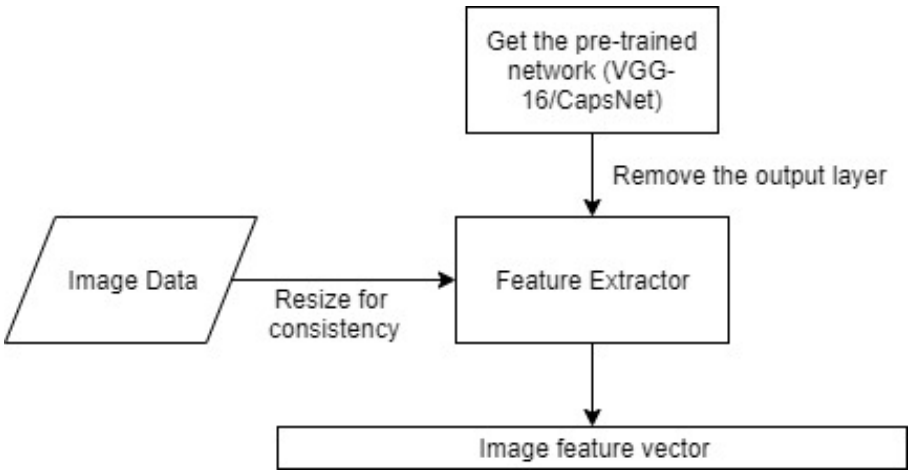


Figure 7: Process of converting images to feature vector

CNN as feature extractor: Hossain et al. (2018) in their study mentioned that pre-trained classifiers like VGG-16, Alexnet, ResNet50, GoogleNet, etc. were the most widely used CNN architectures for this task. The models are pretrained on ImageNet dataset Deng et al. (2009), which has over 13 million images and 1000 classes. This paper uses a pretrained VGG-16 architecture as the feature extractor for the baseline model based on the observations made by Bai and An (2018). The authors have mentioned that image captioning models trained using extracted features from VGG-16 have shown remarkable

performance. VGG-16 has total of 16 layers and 138 million learned parameters in total Simonyan and Zisserman (2015) as shown in Figure 8. To use the network as a feature extractor, the last layer has been removed.

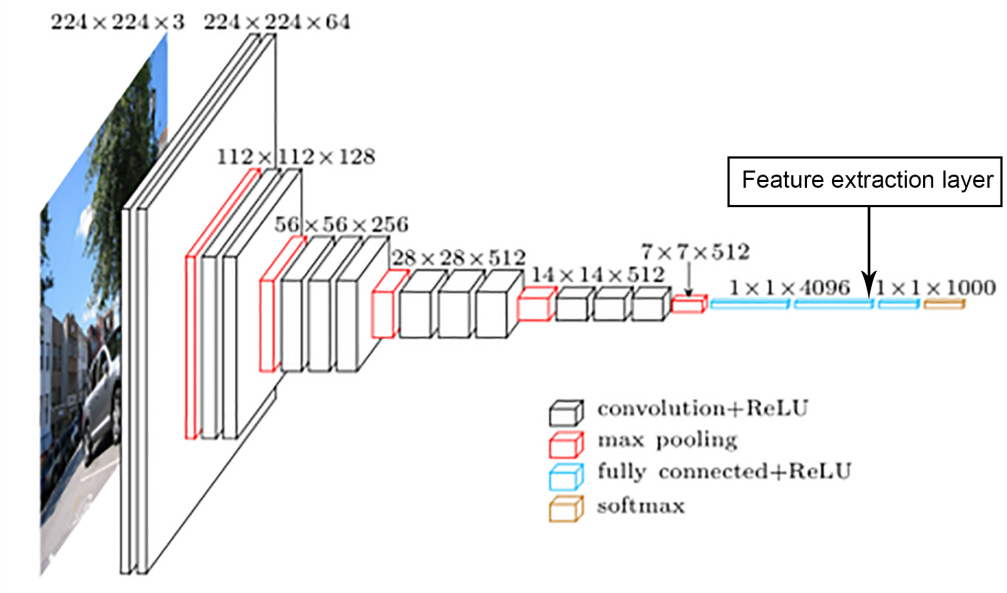


Figure 8: VGG-16 as feature extractor Simonyan and Zisserman (2015)

The resultant network has 134 million pretrained parameters. The second last layer of the VGG-16 network has 4096 neurons, which acts as the feature extractor layer for our model resulting in a vector of length 4096 for each image. The VGG-16 implementation requires the image inserted to be in the size of 224x224, hence all the images are resized to 224x224 before feeding into the model for feature extraction. Keras 'applications' module has been used to further preprocess the images for operations like normalization etc.

Capsule Network as feature extractor: To use Capsule Network as a feature extractor a similar approach of using a pretrained network has been taken. Unlike CNN architectures, there is no official API or support for Capsule Network available in Keras as well as in other libraries till the time of implementation of this project. Hence, a capsule network model had to be pretrained and then use for feature extraction. Capsule Network being relatively new in the research area, most of the studies are done using smaller datasets. The architecture of the capsule network used in this study is inspired by the study done by Rajasegaran et al. (2019). The architecture has 16 Convolutional Capsule Layers with four skip connections as shown in the Figure 9

The resultant architecture has over 13 million learn-able parameters. Training the state-of-the-art CNN models on ImageNet takes a lot of time, hence considering the time and available resource, a smaller dataset CIFAR10[1] for training was chosen which contains total 60000 images of 10 different classes. Marginal loss has been used as the loss function and Adam has been used as the optimizer. The output layer of the network has been removed. The second last layer contained 10 capsules of dimension 32, which is flattened to create the feature extraction layer. The resulting feature vector for each of the images extracted is of size 320. The required dimension of input images for the Capsule Network is 64x64x3. Hence, each image is resized to 64x64 for consistency.

For both the cases, the features are first extracted and stored in a Python dictionary where the image ids are the keys and the respective feature vectors are the values. For

[1]https://www.cs.toronto.edu/ kriz/cifar.html

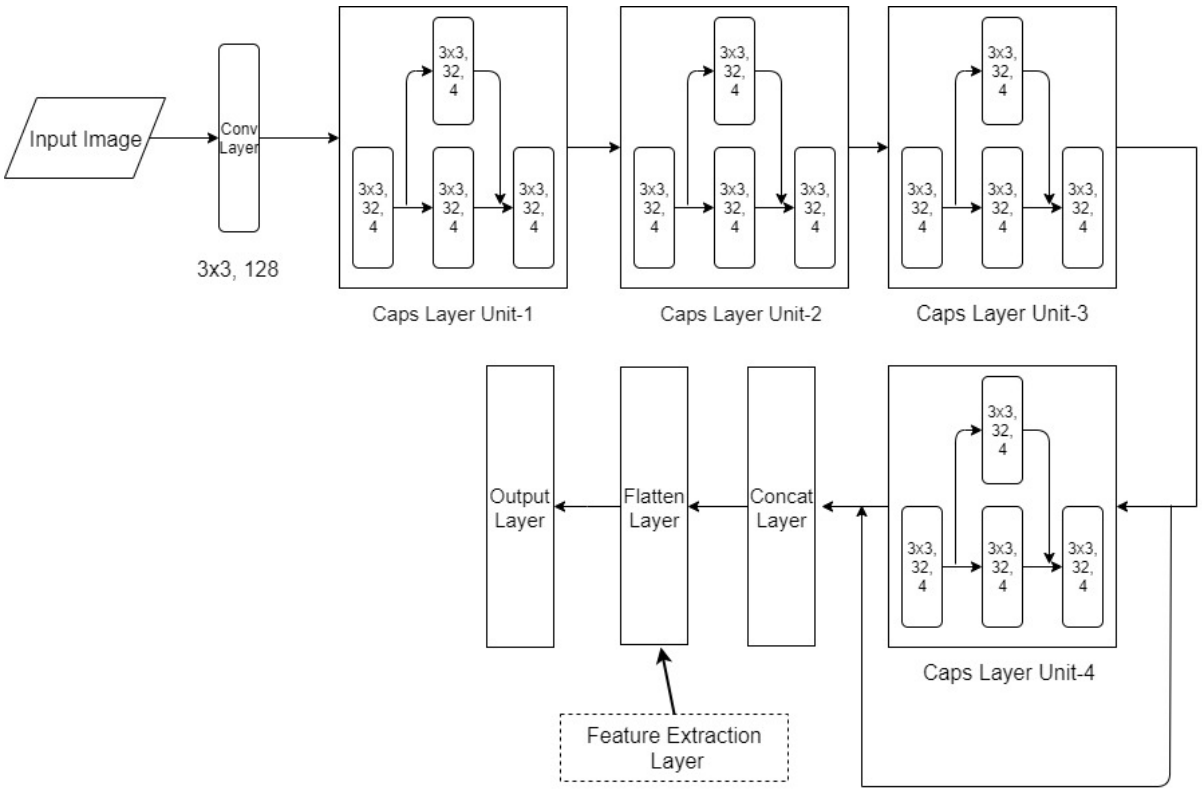


Figure 9: Capsule Network architecture as feature extractor

faster loading and processing for later stages, the dictionary is stored in a byte format in pickle files.

- **Text Data:** This section refers to the Design Specification (3). The text data available contains 40,000 sentences. The sentences need to be preprocessed as required by the model and which helps in enhancing the effectiveness of the model. The corpus contains 9630 unique words used in different contexts and frequencies. As the language model predicts the probabilities of all the words in the vocabulary a smaller vocabulary size is always preferred for faster computation. Reducing the vocabulary size while keeping the maximum information is a challenging task as it results in compromising with fluency. There are multiple ways of cleaning text data. However, considering the time limitations for this project the following steps are followed to clean and transform the data.

  1. **Converting into lowercase:** The sentences contain words with varying cases of characters, which means the words 'two' and 'Two' will be considered as two different words by the model. The main objective of this study is to build a model that is capable of generating a sentence which describes the image content. Hence, all the words in the sentences are converted into lowercase letters to decrease the complexity of the problem.
  2. **Removal of punctuation:** Same as the case of a character, the presence of punctuation in a word adds noise to the dataset. The word $'cat's$ and $'cats'$ are considered as distinct words in the vocabulary. Though removing the punctuation compromises with the fluency of a sentence, for the effectiveness of modelling punctuation are removed from the words.
  3. **Removal of single characters:** After removing the punctuation, single-character words get generated e.g. $'cat's'$ gets transformed to $'cat'$ and $'s'$ which again increases

the size of the vocabulary. Such single character words are removed to lower the vocabulary size for smoother training.

4. **Removal of words containing numbers:** Any word which contains number is removed. Words with a number can be due to human error or a quantifier. In such scenarios, the word $'2'$ and $'two'$ will be two separate words in the vocabulary. Hence, removing those words helps in reducing the vocabulary size.
5. **Wrapping with start and end tokens:** It is necessary to let the model know about the start and end of a sentence. This is done by adding starting and ending identifiers at the two endpoints of a sentence. After passing through the above steps each sentence is wrapped around with '$startseq$' and '$endseq$'.
6. **Tokenization:** After cleaning the vocabulary the resulting count is 7865, which means that the output layer will have 7865 neurons each predicting the probability of a word being generated. For this task, the vocabulary needs to be mapped with unique indexes. Assigning a number with the words is done by the Tokenizer class available in Keras.

The next step after the cleaning of the text data is to transform the data for consistency among all the sentences. The maximum length of a tokenized sentence in the training dataset is found to be 34. All the other sentences having length lesser than the maximum length are padded with 0 to match the required size. The output vector is converted to binary transformation using to_categorical function in Keras. This operation assigns 1 to the correct index and 0 for all the others.

4.3 Modelling

The final model follows the encoder-decoder framework as explained by Tanti et al. (2018). This project implements Inject (Par-inject and Init-inject) and Merge architectures for experimenting and finding the best suitable architecture for our approach. There is a total of three different models trained using each type of image feature extractor resulting in six experiment models. The models are different in terms of when the inputs are fed into the caption generation network or the decoder. This section illustrates the decoder model architecture with layer definitions and functionalities:

1. **Image Input layer:** This layer is responsible for getting the Image feature vector. The shape of the vector depends on the feature extractor used, i.e for the images extracted from CNN the shape is 4096 and for the Capsule Network, it is 320. A dense layer of 256 neurons follows the image input layer to reduce the number of parameters as well as to match the shape of the embedding layer for further processing.

2. **Text Input layer:** The text input layer has the shape of the maximum length of tokenized sentences i.e 34. Any other sentences that have length lesser than maximum length are padded in the data preparation stage.

3. **Embedding layer:** An Embedding layer is added after the text input layer which is used to project words into a vector space. For all the models a vector space of 256 dimensions is used. This layer embeds the tokenized words into floating numbers to represent the semantics of each word in a sentence. The mask_zero argument is set to 0 to let the layer ignore the indexes with 0 paddings.

4. **LSTM layer:** All the models in the experiments have an LSTM layer with 256 memory units also called a sequence processor unit. Based on the architecture of the encoder-decoder framework used the LSTM layer is added at different positions of the network.

For the Par-inject and Init-inject approaches, the LSTM layer considers both image and text as input features. In Init-inject the image feature is inserted into the LSTM cell as the first hidden state. While in Par-inject both the text and image features are concatenated and then used as input. On the other hand, the LSTM layer of the Merge model is used only to encode text inputs.

5. **Add layer:** The 'add' layer is used to add features of images and texts. In the Merge architecture, the output generated from LSTM is added with the input features of the images. On the other hand in Par-inject, both the features are first added and then used as input to the LSTM unit. On Init-inject no 'add' layer is used.

6. **Dense layer:** A dense layer with 256 neurons has been used in all the models for final processing of the inputs coming from the sequencer and the merge layers. This layer uses ReLU activation.

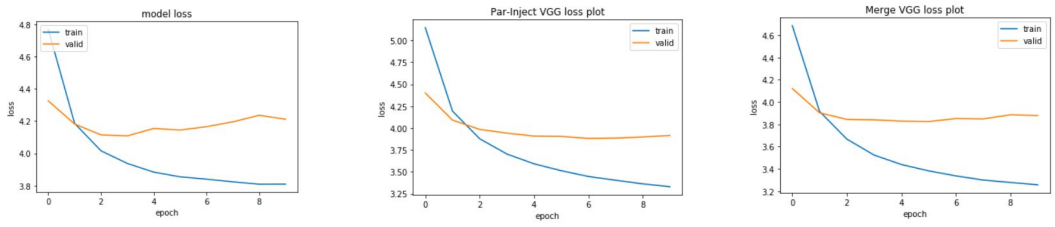
7. **Dropout layer:** Dropout layer is used to randomly turn off neurons based on while training to overcome sparse learning which results in overfitting. The implementation of the model uses .5 dropout ratio after dense layers.

8. **Softmax/Output layer:** The final dense layer of the decoder model has the number of neurons equal to the vocabulary size. This layer uses Softmax activation that outputs probability distribution across the words.

The layers are stacked following the structure as described in Figure 3 and Figure 4. The architecture definition of each of the models is provided in section 6.2 Modelling in the configuration manual.

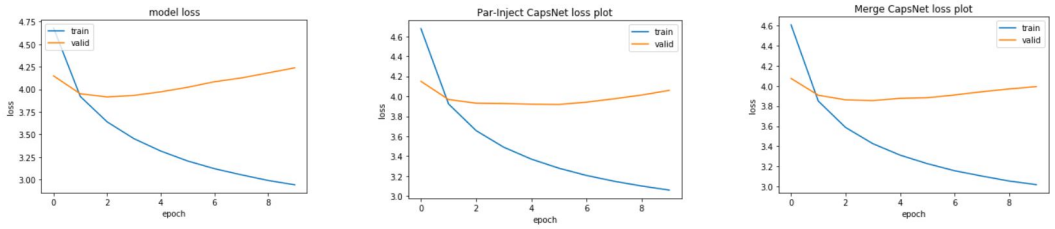
4.4 Training

The defined models are trained separately on 6400 images and evaluated on 800 images. The extracted image features and tokenized texts are used in the training process. A data generator is used for optimum utilization of available resources. As the output of the final model is expected to be just one among many possible outcomes 'categorical_cross_entropy' is used as the loss function. For back-propagation, there are different optimizers available, e.g. stochastic gradient descent, RMSProp, Adam, etc. It was observed from the literature that Adam optimizer has been widely used in the training of many approaches Tanti et al. (2018). Based on that, this paper also uses 'adam' as the optimizer with a learning rate of 0.001. In the early stages of experimentation, the models were trained for a higher number of epochs, but it was observed that the model starts overfitting after the initial few epochs. Hence, each of the models has been trained for 10 epochs. The training curve for each of the models is plotted as shown in the Figure 11 and Figure 12. The blue graphs denote the training loss while the orange graphs denote validation loss.



(a) Init-Inject architecture (b) Par-Inject architecture (c) Merge architecture

Figure 10: Training plots of VGG-16 feature based models



(a) Init-Inject architecture (b) Par-Inject architecture (c) Merge architecture

Figure 11: Training plots of Capsule Network feature based models

Training and validation for each epoch take over 10 minutes. From the plots, it was observed that the model overfitted in the early epochs. Hence to store the best-fitting model, which performed the best on unseen data i.e. validation data, 'ModelCheckpoint' callback has been used. This ensures that the model weights to be saved when there is a decrease in the validation loss. From the saved weights, the weight with lowest validation loss has been considered for evaluation on the BLEU score which is discussed in the next section.

5 Evaluation and Result

This section explains the metric used for the evaluation of the models. In the literature, we observed that for validating the model's efficacy which involves natural language processing, BLEU-(n) and METEOR are the commonly used metrics. Apart from these two, few researchers also evaluated the fluency of sentences from humans. But considering the time constraint human evaluation has not been done in this study. This study uses BLEU- 1-4 scores for the evaluation of the trained models for both greedy search and beam search of beam length 2. Natural Language Toolkit (NLTK) library has been used to implement the BLEU score process.

- **Bilingual Evaluation Under Study (BLEU-(n)):** BLEU-(n) was first proposed by Papineni et al. (2002), it is a metric that evaluates the closeness of the predicted sentence to the original one. This metric was first used for neural machine translation task. Here, 'n' depicts the number for n-gram consideration. When n is 1 it calculates how many times the predicted words are present in the reference sentence, when n is 2 it refers to how many times two words together have been found in the original sentence. Larger the n value, more the fluency check is evaluated. The BLEU score can also be considered as a precision metric for this kind of task which measures how many times predicted words overlap with the original word.

- **Results and Inference:** Below are the BLEU scores achieved for all the models using the trained weights having best validation accuracy. An image was taken from the test set and predictions are made using all the six models as provided in Figure 12.

6 Discussion

In this research, six different models were implemented to build image captioning solutions using features extracted from Capsule Network and VGG-16. Each of the trained models was evaluated with BLEU-(1-4) score for beam length 1 and 2 as tabulated in Table 1 and Table 2 and visualised in Figure 13 and Figure 14,

From the results it can be concluded that the models trained with capsule network features perform almost equivalent to CNN based models if not better. The Merge architecture turned

Architecture	BLEU-1	BLEU-2	BLEU-3	BLEU-4
VGG-Init-inject	0.532	0.277	0.178	0.078
VGG-Par-inject	0.513	0.281	0.192	0.088
VGG-Merge	0.534	0.286	0.196	0.091
CapsNet-Init-inject	0.470	0.220	0.152	0.068
CapsNet-Par-inject	0.493	0.234	0.155	0.073
CapsNet-Merge	0.536	0.267	0.178	0.088

Table 1: BLEU-(n) score for with greedy sentence generation

Architecture	BLEU-1	BLEU-2	BLEU-3	BLEU-4
VGG-Init-inject	0.288	0.102	0.045	0.006
VGG-Par-inject	0.260	0.102	0.052	0.011
VGG-Merge	0.351	0.127	0.056	0.014
CapsNet-Init-inject	0.375	0.124	0.057	0.011
CapsNet-Par-inject	0.286	0.091	0.031	0.004
CapsNet-Merge	0.335	0.106	0.040	0.006

Table 2: BLEU-(n) score for with Beam length 2

VGG_merge: startseq dog is running through the grass endseq
VGG_Par_Inject: startseq dog is running through the grass endseq
VGG_init_Inject: startseq brown dog is running through the grass endseq

CapsNet_merge: startseq two dogs are playing in the grass endseq
CapsNet_Par_Inject: startseq two dogs are playing in the grass endseq
CapsNet_Init_Inject: startseq two dogs are running through the grass endseq

Figure 12: Inference on a test image

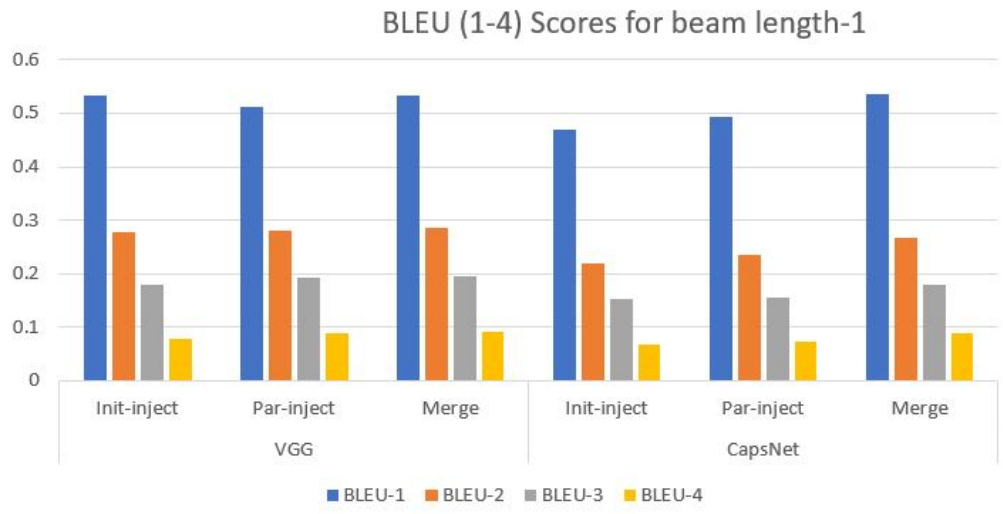


Figure 13: BLEU score for greedy search

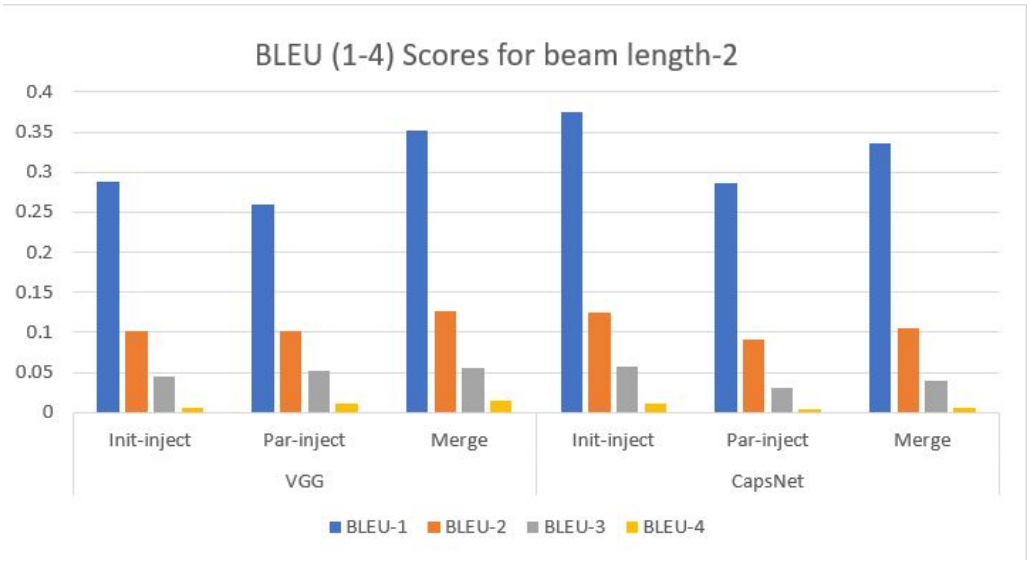


Figure 14: BLEU score for beam search

out to be the best suitable approach for both CNN and CapsNet based models. However, a noticeable difference in the score of Init-inject architecture is observed between the two. In the Init-inject the image feature state was added to the LSTM cell as its first hidden cell as suggested by Tanti et al. (2018). The CNN based model had a score of 0.532 while CapsNet based architecture had 0.470. Among all the three different architectures used in the experiments, the Merge architecture performed better in the case of CapsNet than CNN with beam length 1. Using a greedy search, the BLEU-1 score achieved for the capsule network is 0.536 compared to 0.534 for CNN. The evaluation of this study is limited to the BLEU score only. In terms of inference, as shown in Figure 13, a few images with a high BLEU-(1) score was tested. It can be seen that the generated sentences follow a similar pattern. For instance, in the given example, 'dog', 'grass', 'walking', etc. are common in all six models and the rest of the context like there two dogs, colors of the dogs, etc. are not present. This kind of issue is quite common in general encoder-decoder frameworks. Here the search algorithm allocates more probability to commonly occurring words than least frequent words (Jia et al. (2015)). This kind of problem can be handled using attention-based implementation as seen in the literature.

As mentioned in the Methodology all the other stages except the feature network were kept the same for unbiased experimentation. There were a few architectural key differences between the CapsNet and the VGG-16, which may have an impact on the final outcome. In the researches on the capsule network, it was mentioned that Capsule network trains better and performs well on smaller datasets than CNN (Edgar et al. (2018b), Rajasegaran et al. (2019)). In our study, the Capsule Network used for feature extraction was over 10 times lighter than the VGG-16 extractor in terms of size and number of parameters. The CapsNet was trained on CIFAR-10 dataset which has only 60000 images and 10 classes compared to ImageNet with 14 million images and 1000 classes in the case of VGG-16. The number of learned parameters in the CapsNet is over 13 million while on the other side VGG-16 has over 134 million learned parameters. In the case of CapsNet the size of the output feature vector was just 320 in comparison to 4096 for the VGG-16 based feature extractor. Even after having a noticeable difference between both the architectures the results from the experiments convince that the CapsNet makes promising results and proves that it can be used as a feature extractor in the task of image captioning.

7 Conclusion and Future Work

In this research, we tried to find out whether an encoder-decoder framework with image features extracted using Capsule Network can outperform CNN based solutions. To answer the question based on inferred literature multiple image captioning models were developed using a deep capsule network as the image feature extractor. Three different architectures were followed in terms of when to insert the image features into the language model. The implemented models were trained on the Flickr8k dataset and evaluated based on BLEU-(1-4) scores. A baseline model using VGG-16 as the image feature extractor was also developed to compare the novel approach of this study with a CNN based solution. The study explained how pretrained models can be used as image feature extractors, steps to preprocess the descriptions and finally, encode the both to train the language model or the decoder. The models trained with Capsule Network features showed satisfactory results when compared to the CNN based models. It was observed that the Merge encoder-decoder architecture performed to be the best for both CNN and Capsule Network and the BLEU-(1) score of Capsule Network was higher than the score of CNN using greedy search for sentence generation.

In terms of limitation, the Capsule Network used in this study is smaller in comparison to the depth of state-of-the-art CNN models. It will be interesting to see how a deeper capsule network trained on a larger dataset can impact on the performance of the caption generation as the learned parameters will be more feature-rich and exposed to more complex classes. Moreover, in this research, only a simple encoder-decoder framework was implemented. To enhance the performance, we believe an attention-based encoder-decoder framework can be a strong alternative. Finally, considering the implementation constraints the experiment results found to be persuasive and we believe the findings from this research will contribute to future research in image captioning.

Acknowledgement

I would like to take this opportunity to express my gratitude towards everybody who helped me in finishing this research project on time and as the way planned. At first, I would like to convey my sincere gratitude to my supervisor Dr. Vladimir Milosavljevic for his regular feedback and supervision. His constant support helped me to stay motivated and carry out this research smoothly. I would like to thank the other members of our group under his supervision, I believe constant discussion and suggestion with each other have helped each one of us to gain more knowledge about diverse domains in these few weeks. I would also like to thank NCI library help center, they have been a great support by providing enormous resources to do the literature and guidance in report writing. Lastly, many thanks and regards to my family and friends for their support and motivation throughout the year which helped me to try harder for better.

References

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S. and Zhang, L. (2018). Bottom-up and top-down attention for image captioning and visual question answering, *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Bai, S. and An, S. (2018). A survey on automatic image caption generation, *Neurocomputing* **311**: 291 – 304.
URL: *http://www.sciencedirect.com/science/article/pii/S0925231218306659*

Bengio, Y., Simard, P. and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks* **5**(2): 157–166.

Deng, J., Dong, W., Socher, R., Li, L., Kai Li and Li Fei-Fei (2009). Imagenet: A large-scale hierarchical image database, *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Darrell, T. and Saenko, K. (2015). Long-term recurrent convolutional networks for visual recognition and description, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2625–2634.

Edgar, X., Selina, B. and Jin, Y. (2018a). Capsule network performance on complex data, *Computing Research Repository (CoRR)* **/abs/1712.03480**.

Edgar, X., Selina, B. and Jin, Y. (2018b). Capsule network performance on complex data, *CoRR* **/abs/1712.03480**.

Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J. and Forsyth, D. (2010). Every picture tells a story: Generating sentences from images, *in* K. Daniilidis, P. Maragos and N. Paragios (eds), *Computer Vision – ECCV 2010*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 15–29.

Fei-Fei, L., Iyer, A., Koch, C. and Perona, P. (2007). What do we perceive in a glance of a real-world scene?, *Journal of Vision* **7**(1): 10–10.

Freitag, M. and Al-Onaizan, Y. (2017). Beam search strategies for neural machine translation, *CoRR* **abs/1702.01806**.
URL: *http://arxiv.org/abs/1702.01806*

Hinton, G. E., Krizhevsky, A. and Wang, S. D. (2011). Transforming auto-encoders, *in* T. Honkela, W. Duch, M. Girolami and S. Kaski (eds), *Artificial Neural Networks and Machine Learning – ICANN 2011*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 44–51.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory, *Neural Computation* **9**(8): 1735–1780.

Hodosh, M., Young, P. and Hockenmaier, J. (2013). Framing image description as a ranking task: Data, models and evaluation metrics, *J. Artif. Int. Res.* **47**(1): 853–899.
URL: *http://dl.acm.org/citation.cfm?id=2566972.2566993*

Hossain, M. Z., Sohel, F., Shiratuddin, M. F. and Laga, H. (2018). A comprehensive survey of deep learning for image captioning.

Jia, X., Gavves, E., Fernando, B. and Tuytelaars, T. (2015). Guiding the long-short term memory model for image caption generation, *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2407–2415.

Karpathy, A. and Fei-Fei, L. (2017). Deep visual-semantic alignments for generating image descriptions, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39**(4): 664–676.

Karpathy, A., Joulin, A. and Li, F. (2014). Deep fragment embeddings for bidirectional image sentence mapping, *CoRR* **abs/1406.5679**.

Kiros, R., Salakhutdinov, R. and Zemel, R. S. (2014). Unifying visual-semantic embeddings with multimodal neural language models, *CoRR* **abs/1411.2539**.

Ma, L., Lu, Z., Shang, L. and Li, H. (2015). Multimodal convolutional neural networks for matching image and sentence, *CoRR* **abs/1504.06063**.

Mao, J., Xu, W., Yang, Y., Wang, J., Huang, Z. and Yuille, A. (2014). Deep captioning with multimodal recurrent neural networks (m-rnn).

Ordonez, V., Kulkarni, G. and Berg, T. L. (2011). Im2text: Describing images using 1 million captioned photographs, *in* J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira and K. Q. Weinberger (eds), *Advances in Neural Information Processing Systems 24*, Curran Associates, Inc., pp. 1143–1151.

Palvanov, A. and Im Cho, Y. (2018). Comparisons of deep learning algorithms for mnist in real-time environment, *International Journal of Fuzzy Logic and Intelligent Systems* **18**: 126–134.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation, *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 311–318.
URL: *https://doi.org/10.3115/1073083.1073135*

Pascanu, R., Mikolov, T. and Bengio, Y. (2012). Understanding the exploding gradient problem, *CoRR* **abs/1211.5063**.

Rajasegaran, J., Jayasundara, V., Jayasekara, S., Jayasekara, H., Seneviratne, S. and Rodrigo, R. (2019). Deepcaps: Going deeper with capsule networks, *CoRR* **abs/1904.09546**.
URL: *http://arxiv.org/abs/1904.09546*

Sabour, S., Frosst, N. and Hinton, G. E. (2017). Dynamic routing between capsules, *Computing Research Repository (CoRR)* **abs/1710.09829**.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Socher, R., Karpathy, A., Le, Q. V., Manning, C. D. and Ng, A. Y. (2014). Grounded compositional semantics for finding and describing images with sentences, *Transactions of the Association for Computational Linguistics* **2**: 207–218.

Tanti, M., Gatt, A. and Camilleri, K. P. (2018). Where to put the image in an image caption generator, *Computing Research Repository (CoRR)* **abs/1703.09137**.
URL: *http://arxiv.org/abs/1703.09137*

Vinyals, O., Toshev, A., Bengio, S. and Erhan, D. (2014). Show and tell: A neural image caption generator, *Computing Research Repository (CoRR)* **abs/1411.4555**.

Vinyals, O., Toshev, A., Bengio, S. and Erhan, D. (2016). Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge, *CoRR* **abs/1609.06647**.
URL: *http://arxiv.org/abs/1609.06647*

Wang, W., Li, H., Pan, L., Yang, G. and Du, Q. (2018). Hyperspectral image classification based on capsule network, *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 3571–3574.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S. and Bengio, Y. (2016). Show, attend and tell: Neural image caption generation with visual attention, *CoRR* **abs/1502.03044**.
URL: *http://arxiv.org/abs/1502.03044*

You, Q., Jin, H., Wang, Z., Fang, C. and Luo, J. (2016). Image captioning with semantic attention, *CoRR* **abs/1603.03925**.