Find the best Convolutional Neural Networks and Recurrent Neural Networks for Image Captioning

Weizhong Lai

September 6, 2018

Abstract

The technology of image captioning combines two major fields of artificial intelligence which are image processing and natural language processing. It can help some people with image cognitive impairment to obtain image information. This paper compared the performance of recent CNN models as an encoder of image captioning. And it also compared the performance of three kinds of RNN architecture influenced by dimensions of word embedding, image processing method and pretrained word embedding.

Contents

Acknowledgements

I would like to thank to my supervisor, Dr. Adam Prugel-Bennett, a respectable and responsible professor for his supports in every week's meeting. And thanks to Dr. Sheng Chen for his kindness and help in the last stage of my project. Without their enlightening instruction I could not have completed my project.

Chapter 1

INTRODUCTION

1.1 Introduction

Automatic image captioning can make a picture generate a sentence that describes the content of the picture. The technology of image captioning have developed fast recent years, from an m-RNN model which proposed by Baidu Researcher in 2014 [1] to RNN with attribute prediction CNN models which proposed in 2016 by [2]. The image captioning have many applications. For instance, It can help some people with image cognitive impairment to obtain image information. Moreover, the technology of image captioning could help image search. Image search can use the sentences generated by Image Captioning to extract label to search similar images. Also, the technology of Image Captioning uses the technology of two main filed in artificial intelligence which is Image processing and natural language processing, which could help to improve the development of artificial intelligence. Therefore, it is beneficial for us to research the technology of Image Captioning. However, such a small task for human beings is a big challenge for machines. And it is more complicated than general image processing and Nature Language Processing(NLP) problems. The reason for that is that image captioning not only need to detect each object in the image and select a word to describe it but also need to find the relationship and connection between each object. Moreover, the sentences should be organised as rules of the stander English grammar. In order to implement technology of Image Captioning, I need to combine the image processing technology of Computer Vision and the language processing technology.

The essence of Image Captioning technology problem is the problem of Visual-to-Language [2]. The machine should be taught to describe an image using natural language as accurate as possible. The basic idea for building an Image Caption model comes from patterns of text translation which called encode-decode patterns. The encoder extracts a context vector from a source language, and the decoder decodes the context vector into target language [3]. The purpose of encode-decode patterns in translation is to transfer the variable length of sentences into fix length of sentences. Therefore, it can be easier for a decoder to generate a variable length of target sentences from a fixed length of sentences than a variable length of sentences. The patterns of Image Captioning can be a similar one with translation patterns. The encoder of Image Captioning patterns extracts images features by Convolutional Neural Network(CNN) models, which can be a kind of fix variable length of vectors and it contains the critical information of images. And the decoder of Image Captioning can decode the fix length of vectors into a variable length of sentences which can describe the images. I decided to use Recurrent Neural Network(RNN) as a decoder because RNNs can learn the information of context. Meanwhile, languages of human beings have many semantic associations between every single word. And RNN can learn these associations by every single word in sentences.

The first purpose of this paper is to compare the performance of five CNN models(Baseline CNN, VGG16, VGG19, ResNet50, InceptionV3) as an encoder and to find which is the best CNN models for encoding and find the possible factor that could influence the performance. And the second purpose of this paper is to compare three different kinds of RNN architecture as the roles of a decoder, have different treatment of image features, and to find whether the performance of RNN will be influenced by dimensions of word embedding, pretrained word embedding and image prepossessing method. And these three RNNs architectures I will introduce in the methodology section.

The motivation is that I read an article about Image Captioning, and I think the way combine image and caption is interesting. I want to find out some influencing factor which could help to improve the performance of the technology of Image captioning. The development and application of Image captioning technology can improve people's life. It can not only help people with image reading disabilities to know the content of pictures but also help the progress of image retrieval technology. I want to know whether using different CNN as the encoder will produce different results. And will the different treatment of image features in RNN affect the performance of image captioning.

1.2 Literature review

1.2.1 Background

Image Captioning technology is a new technology combining image processing technique and Natural Language Processing. The earliest research on image captioning technology started in 2014, and there was an increasing number of models and attempts to make more models optimisation on Image Captioning in 2015. Furthermore, the technology of Image Captioning became a hot topic on CVPR(IEEE Conference on Computer Vision and Pattern Recognition) in 2016.

1.2.2 Some RNN models

(1) m-RNN model In 2014, Baidu research institute proposed the multimodal Recurrent Neural Network(mRNN) model, which creatively combined the deep CNN and deep RNN to solve the problems of image captioning and the image retrieval [1]. The m-RNN model accepts an input of image and an input of a sentence, and the model will predict probability distribution of next word append the sentences. This model has six layers including two embedding layers, a recurrent layer, a multimodel layer and a softmax layer. And the multimodel layer will connect the output of the embedding layer and image features extracted by AlexNet CNN model. And the image features will feed to the model each time series.

(2) NIC model After the CNN-RNN architecture invented, there was an increasing number of paper proposed using these CNN-RNN architectures. One month later, the Neural Image Caption(NIC) model was proposed by Vinyals et al. of Google [4]. It replaces the regular RNN layer with the LSTM layer to solve the problem of gradient explosion because the sentences can be very long so that the time series can be very long. And the image feature will feed to the model one time in the start of time series. The NIC model shows better performance than the m-RNN model.

(3) RNN with attribute prediction CNN models In the first two models below, the image features extracted by CNN is directly converted into a sentence without any semantic processing. The RNN with attribute prediction CNN models which proposed by [2] improve the way of image feature extraction. They first built a

vocabulary list from the sentences of the training set, and then make the vocabulary list as a multi-label. After that, they trained a CNN model with multi-label classification, so that the image feature vectors go through by the CNN model will contain some semantic information.

1.2.3 Some CNN models

(1) VGGNet VGGNet is a deep convolutional network jointly developed by Oxford University computer vision group and DeepMind company, and it won the second place in the classification project and the first place in the positioning project in the ILSVRC competition in 2014[5]. VGGNet uses 16 layers(VGG16) or 19 layers(VGG19) to build a convolutional neural network, and the size of its convolution kernel is a continuous number of 3 by 3 kernel. A continuous number of small size of convolution kernel is superior to a large size of convolution kernel because multiple layers of nonlinearity can be added more deeply to learn more complex kind of features of images[5]. However, the small kernel size may cause the convolution layer in the middle to take up more memory space in the back propagation algorithm. Therefore, VGGNet used scale jittering data enhancement technology in training and used activation function ReLU after a layer of convolution. Batch gradient descent was used to train each convolutional layer.[5]

(2) Inception Although VGG can perform well on image classification, it is difficult to deploy it on a moderately sized GPU because it requires very high computational requirements on memory and time. In 2014, Google company built a new architecture called GoogLeNet. Ensuring the sparsity of the neural network can reduce the parameters that need to be trained. However, most hardware devices are optimised for a dense neural network. A sparse neural network can be clustered into a denser neural network to improve computational performance, just as the human brain can be seen as the repeated accumulation of neurons [6]. Therefore, the GoogLeNet team proposed the Inception network architecture to densify the sparse neural network, which can generate dense data using sparse neural networks and it can not only increase the performance of neural networks but also guarantee the efficiency of the use of computing resources [6]. Inception has been developed through multiple versions of V1, V2, V3, and V4. I used InceptionV3 in my project.

(3) ResNet As the network deepens, the accuracy of a training set decreases [7], This is not caused by Overfit, because the training set should be very accurate in overfitting situations. The ResNet convolution neural network model was proposed in 2015 and won the first place in the classification task of ImageNet competition. And it solved the problem through identity mapping and residual mapping. If the network has reached the optimum, and the network is further deepened, the residual mapping will be pushed to 0, and only identity mapping is left[7]. So the network is always in the best state, and the performance of the network will not decrease as the depth increases.

Chapter 2

METHODOLOGY

2.1 Evaluation indicators BLEU

2.1.1 Introduction

Bilingual Evaluation Understudy (Bleu) is an evaluation indicator used to evaluate the quality of a text being translated by a machine from one text to another[8]. BLEU is used to evaluate the degree to which machine-translated text is similar to human translated text and the higher the Bleu score of the machine-translated text, the closer this text is to human translation. Although this evaluation indicator is designed for machine translation, it can reflect the quality of the text generated by images. And this indicator is widely used in the evaluation of image annotation results. In the next section, I will introduce the way to calculate the score of BLEU[8].

2.1.2 Calculation steps

N-gram

The BLEU need two kinds of sentences, one is candidate sentences which are used to be evaluated and the other is reference sentences which are used to be as a standard sentence. The sentences used for evaluation can be divided into consecutive word pairs of length N which can compare how many N consecutive words equal to the reference sentence. However, in order to cope with the situation that most of the candidates words are all in reference, but only a few words of reference are in the candidate, the score of n-gram should be the minimum values of n-gram number in candidate and reference.

$$P_n = \frac{\sum_{C \in \{candidates\}} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in \{candidates\}} \sum_{n-gram' \in C'} Count_{clip}(n-gram')} \tag{2.1}$$

The denominator represents the number of n-gram occurrences in candidate sentences; the numerator represents the minimum number of n-gram occurrences in reference sentences and candidate sentences.

Brevity penalty

Brevity penalty(BP) is added to avoid the situation that the short candidate sentences could get a high score, but the quality of the sentences is bad.

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{2.2}$$

The 'r' denotes the number of words in reference sentences, and the 'c' denotes the number of words in candidate sentences.

Final form

Add some weights to each order of n-gram to compute the cumulative of the different n-grams.

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n) \tag{2.3}$$

w_n denotes the weights of each order of n-gram.

2.1.3 Different weight of BLEU score

Different BLEU indicator can be obtained by taking different weights for each order of n-gram. Since the overall BLEU score decreases exponentially with the increase of n-gram as the Figure 2.1 shows. So generally n-gram can be taken up to 4-gram.

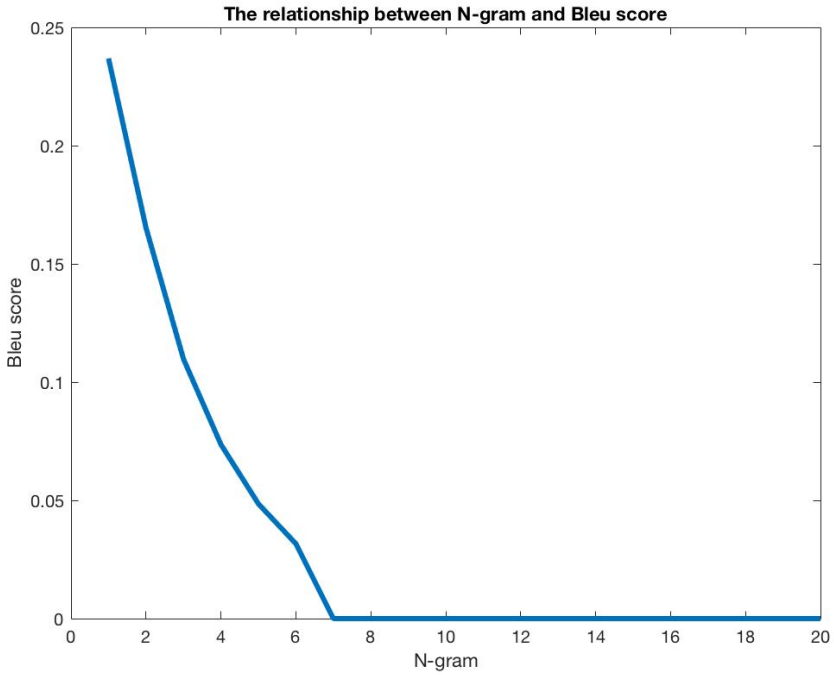


Fig. 2.1. The bleu score change when increasing the order of N-gram. The weight of each order of n-gram that is accumulated is subject to uniform distribution. That is in the process of n-gram cumulative increasing, the weight of each n-gram are the same. As n-gram increases, the overall blue score drops exponentially.

In my project, I will use BLEU1, BLEU2, BLEU3 and BLEU4 to calculate the result of my experiment result. BLEU1 represent the cumulative 1 to 4 gram score with weight [1,0,0,0], BLEU2 represent the cumulative 1 to 4 gram score with weight [0.5,0.5,0,0], BLEU3 represent the cumulative 1 to 4 gram score with weight [0.33,0.33,0.33,0], and BLEU4 represent the cumulative 1 to 4 gram score with weight [0.25,0.25,0.25,0.25].

2.2 Softmax and loss function

2.2.1 Softmax

the activation function softmax $S_i = \frac{e^{V_i}}{\sum_j e^{V_j}}$ can make a vector into a probability distribution. V_i or V_j represent the i or j element of the vector [9].

2.2.2 Cross entropy

cross entropy can measure the distance from one probability distribution to another probability distribution [10].

$$loss = H(y, \hat{y}) = -\sum_{i} y_i log\hat{y}_i \tag{2.4}$$

y_i and y_j represent the two kind of probability distribution.

2.3 Encoder

2.3.1 What is Convolutional Neural Networks

Convolutional neural networks is a more powerful neural network than conventional artificial neural networks which contains convolution layer, pooling layer and fully connected layer[11]. Convolutional neural networks are generally used for image classification by training the weights of convolution layer and the fully connected layer. Usually, the convolution layer can learn the combination of various features of the images. Figure 2.2 is an example of a CNN architecture.

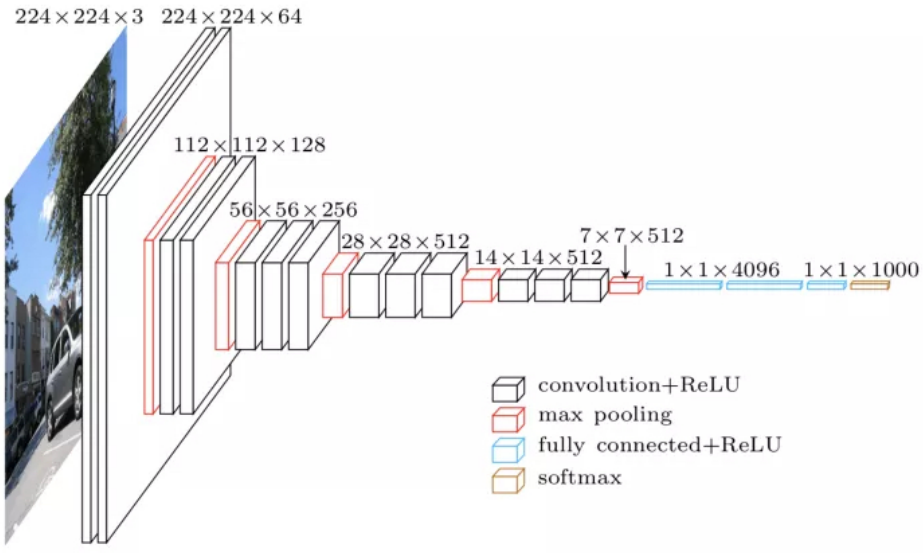


Fig. 2.2. An example of CNN architecture

Convolution kernels

One of the differences between a convolution neural network and a simple artificial neural network is that it can extract different features in images by convolutions [12]. convolution is an operation which can multiply the corresponding positions of different parts of the image matrix and convolution kernel matrix, and then add up each element in the matrix multiplied by these two matrices. The part of image matrix is the same size as the convolution kernel matrix. Formula 2.5 is the CNN convolution formula. The $*$ denotes a convolution operation. W denotes a convolution kernel [12], and X denotes an input image. The dimension of W varies according to the dimension of X. Usually, X is a three-dimensional tensor. When the convolution kernel completes one operation, and then the kernel will shifts one unit to complete the next operation until all the submatrices in the image are done

$$s(i,j) = (X * W)(i,j) = \sum_{m}\sum_{n} x(i+m, j+n)w(m,n) \tag{2.5}$$

Each convolution kernel in CNN can extract a feature of an image[13]. For example, if a feature of an object in the image is a straight line. The value of the position where the line is will be very big after operating by a line detection convolution kernels. Further, if I use a variety of convolution kernels, it can extract a variety of characteristics.
For the output after operating by a convolution kernel, the value is controlled by the ReLU activation function $f(x) = \max(0, x)$, and change the value of elements in the output tensor below zeros to zero.

Pooling

Pooling is the process of compressing each submatrix of a tensor and compress each submatrix into an element. This process can shrink the dimensions of a tensor. There are two common ways of pooling; one is max pooing which can take a maximal value among the element of submatrix, and the other is average pooling which can take an average value among the element.

Fully connected layer

After some convolutional layers and pooling layers, we get a lot of high-level feature combinations. And in order to do the final classification or other processing, we need to combine these features into a one-dimensional tensor. And then we can use a fully connected layer to turn this one-dimensional tensor into an N-dimensional vector, and usually, the N-dimensional vector corresponds to N categories.

2.3.2 CNN architectures in my project

Baseline CNN model

Layers	Dimensions	activation function
Input	224*224*3	
Conv2D-32	224*224*32	RELU
Conv2D-64	224*224*64	RELU
MaxPooling	112*112*64	
Conv2D-128	112*112*128	RELU
Conv2D-128	112*112*128	RELU
MaxPooling	56*56*128	
Flatten	1*401408	
Dense	1*1024	RELU
Dense(prediction)	1*100	SOFTMAX

Table 2.1: Baseline CNN model architecture

I build a baseline CNN model, the architecture just like Table 2.1 shows. It is constructed by four convolutional layers with activation function ReLU and two MaxPooling layers, and it ends with two fully connected layers.

VGG16 model

Layers	Dimensions	Layers	Dimensions
Input	224*224*3	Conv2D-512	28*28*512
Conv2D-64	224*224*64	Conv2D-512	28*28*512
Conv2D-64	224*224*64	Conv2D-512	28*28*512
MaxPooling	112*112*64	MaxPooling	14*14*512
Conv2D-128	112*112*128	Conv2D-512	14*14*512
Conv2D-128	112*112*128	Conv2D-512	14*14*512
MaxPooling	56*56*128	Conv2D-512	14*14*512
Conv2D-256	56*56*256	MaxPooling	7*7*512
Conv2D-256	56*56*256	Flatten	1*25088
Conv2D-256	56*56*256	Dense	1*4096
MaxPooling	28*28*256	Dense	1*4096
		Dense(prediction)	1*1000

Table 2.2: VGG16 model architecture

Table 2.2 shows the architecture of VGG16 model. The VGG16 model has 13 convolution layers and three fully connected layers. The first several layers of the architecture of VGG16 models are the overlapping of several convolutional layers and a MaxPooling layer, and the last three layers are fully connected layers connected in series. The first two convolutional layers both have 64 convolutional kernels which have a 3*3*3 size. After the two convolutional layers, it will go through a MaxPooling layer which can choose a maximal value among in a 2 by 2 square. After the MaxPooling layer, the tensor dimensions become the size of 112*112*128. Later, the tensor will go through several convolutional layers and MaxPooling layers just as the Table 2.2 shows. At last, the tensor will go through three fully connected layers with activation function ReLU.

VGG19 model

Layers	Dimensions	Layers	Dimensions
Input	224*224*3	Conv2D-512	28*28*512
Conv2D-64	224*224*64	Conv2D-512	28*28*512
Conv2D-64	224*224*64	Conv2D-512	28*28*512
MaxPooling	112*112*64	MaxPooling	14*14*512
Conv2D-128	112*112*128	Conv2D-512	14*14*512
Conv2D-128	112*112*128	Conv2D-512	14*14*512
MaxPooling	56*56*128	Conv2D-512	14*14*512
Conv2D-256	56*56*256	Conv2D-512	14*14*512
Conv2D-256	56*56*256	MaxPooling	7*7*512
Conv2D-256	56*56*256	Flatten	1*25088
Conv2D-256	56*56*256	Dense	1*4096
MaxPooling2D	28*28*256	Dense	1*4096
Conv2D-512	28*28*512	Dense(prediction)	1*1000

Table 2.3: VGG19 model architecture

The VGG19 model architecture is shown by the Table 2.3. It is mostly same with VGG16 model. The difference is that the VGG19 model is deeper than the VGG16, it have 16 convolutional layers and three fully connected layers. Like the VGG16 model, the first several layers are alternating between convolutional layers and MaxPooling layers. The performance should be better than the VGG16 model from experiment result [5].

ResNet

Layers	Dimensions	50-layer	101-layer
conv1	112*112	7*7, 64, stride 2	
conv2_x	56*56	3*3 max pool, stride 2	
		$\begin{bmatrix} 1\times1 & 64 \\ 3\times3 & 64 \\ 1\times1 & 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1 & 64 \\ 3\times3 & 64 \\ 1\times1 & 256 \end{bmatrix} \times 3$
conv3_x	28*28	$\begin{bmatrix} 1\times1 & 128 \\ 3\times3 & 128 \\ 1\times1 & 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1 & 128 \\ 3\times3 & 128 \\ 1\times1 & 512 \end{bmatrix} \times 4$
conv4_x	14*14	$\begin{bmatrix} 1\times1 & 256 \\ 3\times3 & 256 \\ 1\times1 & 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 & 256 \\ 3\times3 & 256 \\ 1\times1 & 1024 \end{bmatrix} \times 23$
conv5_x	7*7	$\begin{bmatrix} 1\times1 & 512 \\ 3\times3 & 512 \\ 1\times1 & 2058 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1 & 512 \\ 3\times3 & 512 \\ 1\times1 & 2048 \end{bmatrix} \times 3$
	1*1	average pool, 1000-d fc, softmax	

Table 2.4: ResNet model architectures

Table 2.4 shows the ResNet model architecture. And it is different from VGG net model. It not only have deeper layers than VGG net, but it also has a structure that VGG does not have which called 'shortcut'.

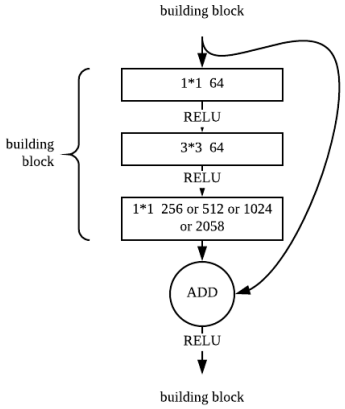


Fig. 2.3. Building block

Like Table 2.4 shows, the building block in the brackets has three convolution layers. Each building block has a shortcut to get to the next building block like Figure 2.3 shows.

Building blocks are designed to reduce the number of training parameters required. The first group of 64 convolution kernels which size is 1 by 1 in the convolution layer is to reduce the channel of the last tensor, and the next group of 64 kernels which size is 3 by 3 in the convolution layer is to extract image features and the last group of 1 by 1 convolution kernel in the building block is to recover the dimensions. For example, if the output tensor of last building block has 256 channels, the first convolution layer in the current building block which has 64 kernels of one by one size is to reduce the number of the channel to 64. The last convolution layer in the current building block which has 256 1 by 1 kernels is to recover the number of channels to 256, and the output of the current building block will add the other output of the shortcut of last building block. the number of parameters which will be trained are $1 \times 1 \times 256 \times 64 + 3 \times 3 \times 64 \times 64 + 1 \times 1 \times 64 \times 256 = 69632$. If there is no building block, the number of parameters which will be trained are $3 \times 3 \times 256 \times 256 + 3 \times 3 \times 256 \times 256 = 1179648$. The number of parameters of building block structure are much less than that of no building block structure.

Inception

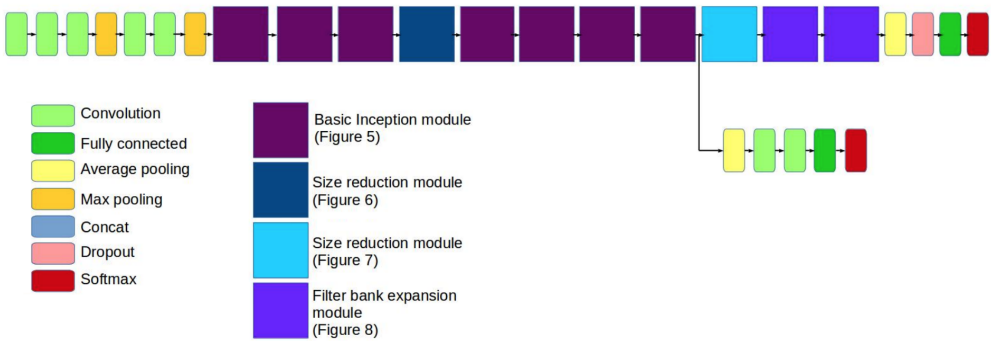


Fig. 2.4. The architecture of Inception model [14]

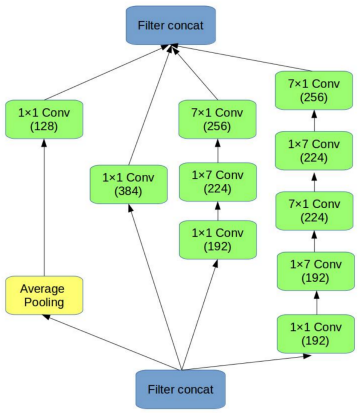


Fig. 2.5. Basic Inception module [14]

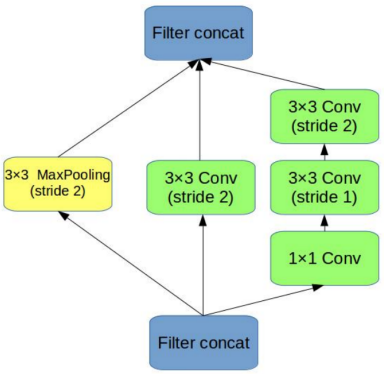


Fig. 2.6. Size reduction module [14]

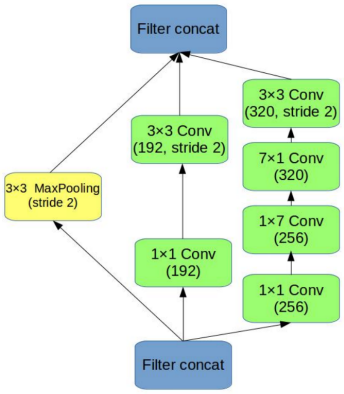


Fig. 2.7. Size reduction module [14]

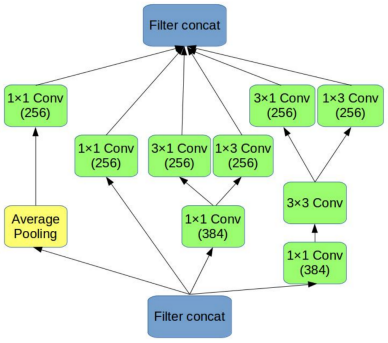


Fig. 2.8. Filter bank expansion module [14]

Figure 2.4 shows the architecture of Inception. The first seven and the last four layers of the Inception model are similar to the VGG net model. The convolution layer and MaxPooling layer are alternately in together. The Inception model can make the size and numbers of convolution kernels and even the need of MaxPooling as parameters to be trained. Inception model can concatenate different size of kernels to learn how to select kernels. And the Inception will try to decompose the big kernel into the small kernel. Figure 2.5 shows the grid of 7*7 is decomposed into a series of 1*7 and 7*1 kernels. Figure 2.6 shows the gird of 5*5 is decomposed into a two 3*3 kernels.

2.4 Decoder

2.4.1 What is Recurrent Neural networks

Simple Recurrent Neural networks

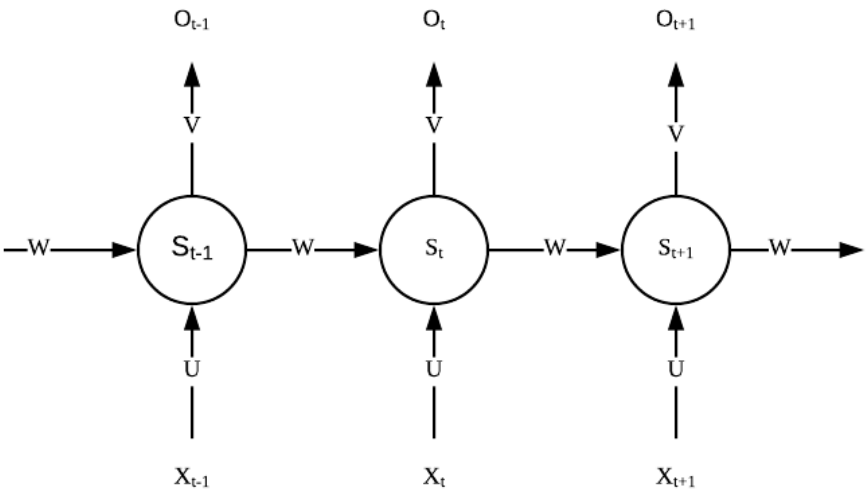


Fig. 2.9. The architecture of RNN. Legend: S - the state in each time series; W,U - weight; Q: the output of each time series

The recurrent neural network can process sequence information well. It can learn the connections between the inputs of each time series. In the task of Image captioning, if I use a simple artificial neural network, it just simply learn the connection between the current word, image information and the next word. The recurrent neural network can handle the task of word sequences very well. The Figure 2.9 shows the architecture of a recurrent neural network. The X represents every single word of a sentence. U is the weight matrix from the input layer to the hidden layer. O is a vector that represents the value of the output layer. And V is the weight matrix of the hidden layer to the output layer.

$$O_t = g(V \cdot S_t) \tag{2.6}$$

$$S_t = f(U \cdot X_t + W \cdot S_{t-1}) \tag{2.7}$$

As the formula 2.6,2.7 shows, the output (O) is depend on state(S), and the current state(S_t) is depend on last state(S_{t-1}).
In the process of training, if the word sequence is too long, it may cause the gradian explosion or disappear problem. Therefore, I used the structure of LSTM when constructing the model.

Recurrent Neural Networks with LSTM

Recurrent Neural Network with Long Short Time Memory(LSTM) can effectively avoid the problem of gradient explosion. Unlike the simple RNN, LSTM has two states, one for long-term memory and one for short-term memory. LSTM lets the forget gate and input gate to control the amount of long-term memory.

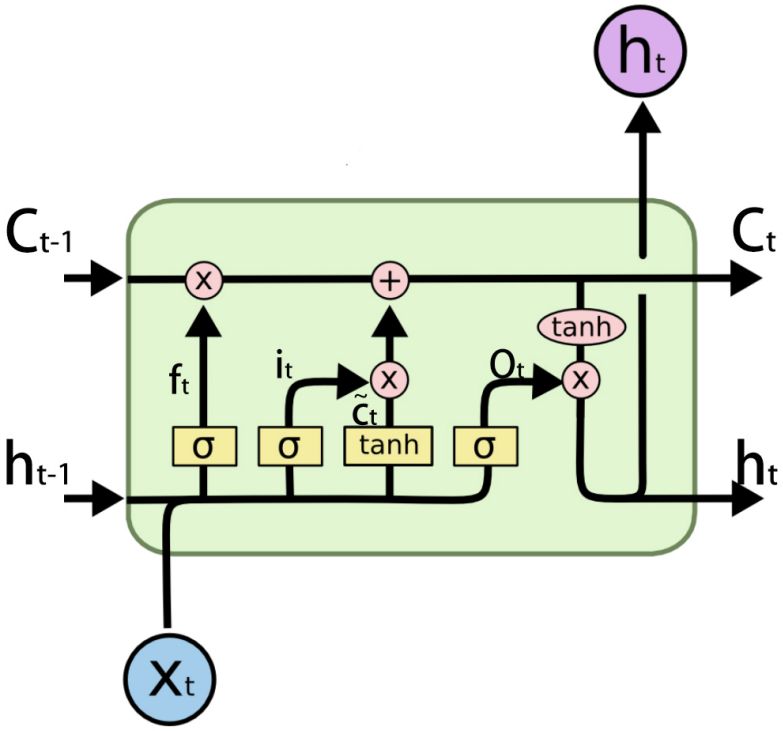


Fig. 2.10. archtecture of a single LSTM cell [15]

Figure 2.10 shows the architecture of a single LSTM cell. At time t, the LSTM have three kinds of inputs, the first is the input value in current time X_t, the second is the value of hidden state in previous time h_{t-1}, and the third is the cell state c_{t-1} in the previous time, cell state can preserve long-term memory. There is two kind of output of LSTM, the first is the hidden state in current time h_t and the second is cell state in the current time h_t. LSTM can be defined by following formula [16]:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.8}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.9}$$
$$\widetilde{C} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.10}$$
$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{2.11}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.12}$$
$$h_t = o_t * \tanh(C_t) \tag{2.13}$$

Forget gate (f) which made by formula 2.8 can make cell state in the previous time (C_{t-1}) forget something that is not important. Input gate i_t and $\widetilde{C}_t$ is to add some information to cell state C_{t-1}, the information for cell state is updated by the forgot gate and input gate like formula 2.11 shows. The symbol $*$ denotes elements multiply of two vectors.

2.4.2 Three different RNN architectures

This subsection I will talk about the decoder constructed with three kinds of RNN architectures. The first is **inject architecture**, which has an image feature located in start time steps. The second is **cross architecture**, which has many image features located in every two time steps. The third is **combine architecture**, which has no image features located in time series but it will combine the output of LSTM directly.

Inject architecture

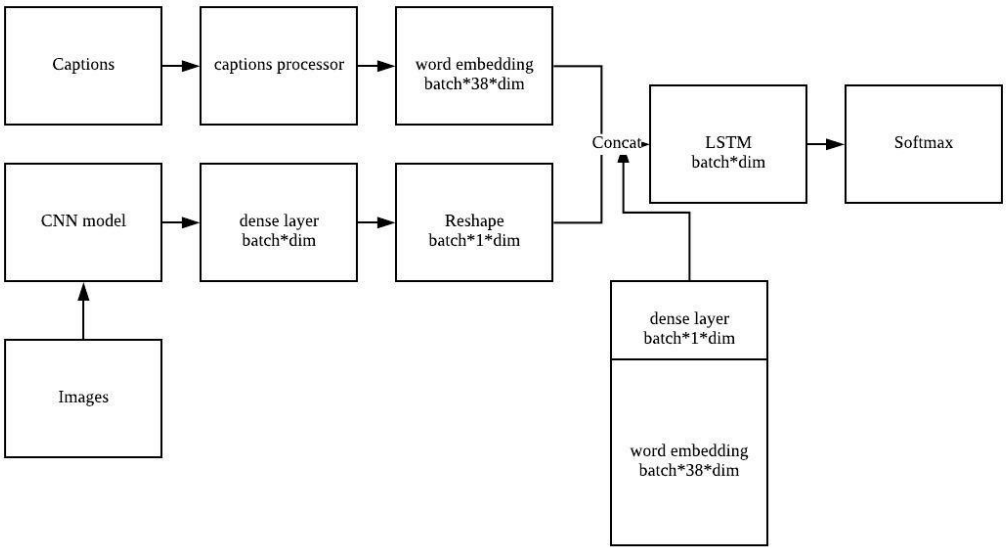


Fig. 2.11. Inject architecture. The batchsize is set to 128. The dim is 4 different dimensions. captions processor is a text preprocessing method that I will explain at experiment. dense layer is a fully connected layer.

The Fig.2.11 shows the architecture of inject. The inject architecture was proposed by [17] and I build it using Tensorflow. The inject architecture need two input which is captions and images. The caption is some sentences coming from Flickr8k database. After going through the captions processor that can make the form of captions into the form of input needed by word embedding layer which I will talk later in next subsection. The word embedding layer converts the word code that is the output of captions processor layer into a tensor of word vectors that have dimensions batch*38*dim, the batch is the training parameters which I will set to 128 and the dim is the dimensions of word vectors which I will make a comparison between 50,100,200 and 300 in my experiment result. CNN models will extract some features from images and send the features into dense layers. Then the dense layer will compress the dimensions of image features to match the dimensions with the word vector. Because the dimensions of the output of word embedding layer is a three-dimensional tensor, I add a reshape layer to reshape the 2-dimensional tensor which comes from a dense layer into a 3-dimensional tensor. After that, I will concatenate the output tensor of the dense layer which comes from image features and the output tensor of word embedding layer. The LSTM layer will make the second dimensions as a time series. The LSTM layers will process 39 time steps, and the output of LSTM is the output of the last cell. The output tensor of LSTM will go through several dense layers to be the same dimensional as the one-hot vector of a word. I will introduce the one-hot word in the next subsection. In the last layer, I will use softmax function to turn the vector of the output to a probability distribution.

Cross architecture

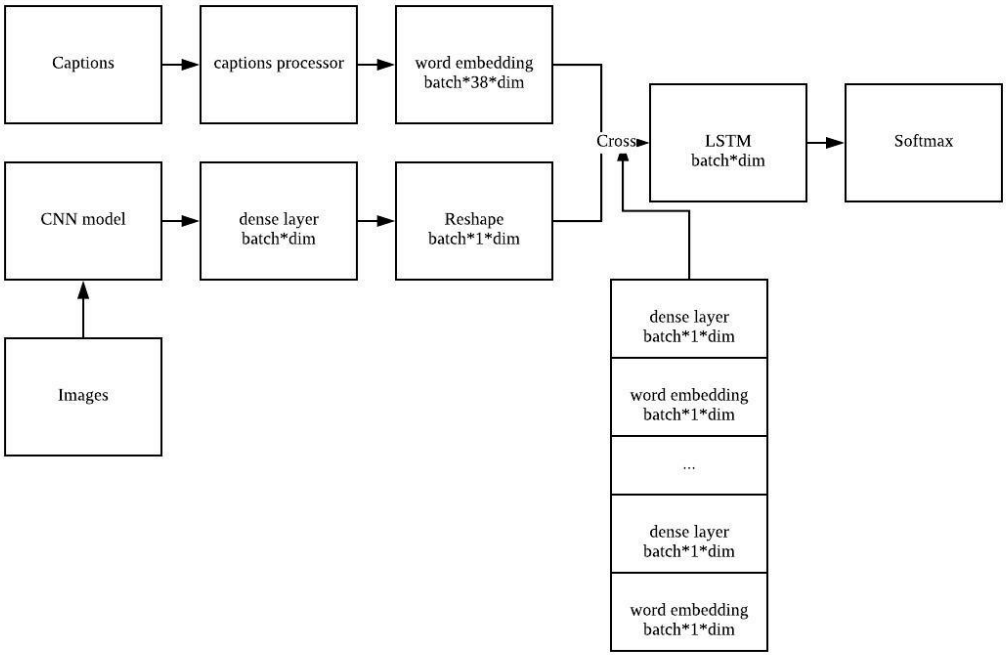


Fig. 2.12. Cross architecture. The batchsize is set to 128. The dim is 4 different dimensions. captions processor is a text preprocessing method that I will explain at experiment. dense layer is a fully connected layer.

The Figure 2.12 shows the cross architecture. I made this architecture and implemented it using Tensorflow. The different from inject architecture is that I make a different way to combine image features and word embedding which is a kind of cross method. In order to find out whether image features play an important role in the LSTM layer, I make a lot of the same image features that made by dense layer cross the tensor that made by the embedding layer like Figure 2.12 shows. This architecture will make the LSTM process image features every two time steps.

Combine architecture

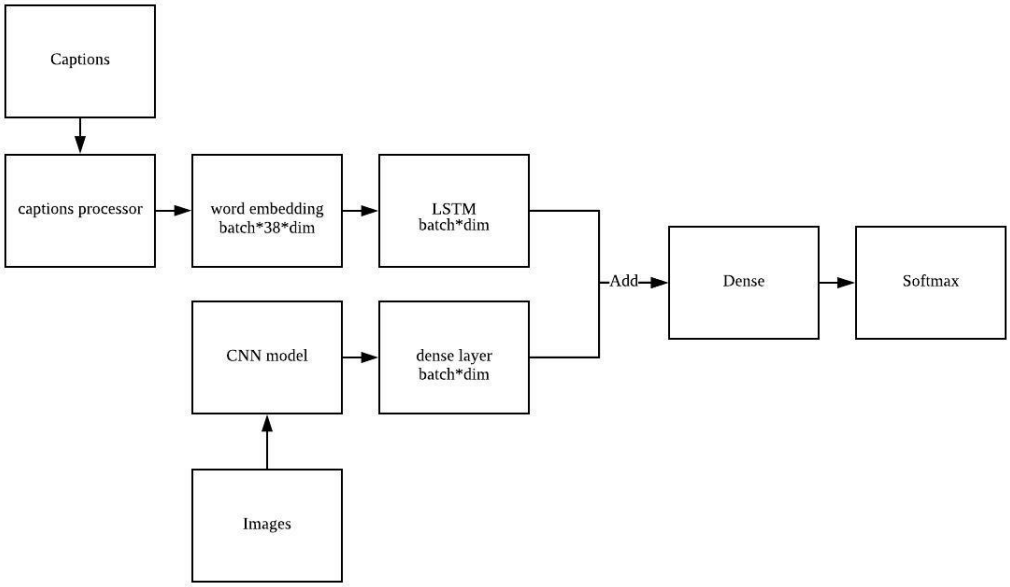


Fig. 2.13. Combine architecture. Add: add each elements of two tensors. The batchsize is set to 128. The dim is 4 different dimensions. captions processor is a text preprocessing method that I will explain at experiment. dense layer is a fully connected layer.

Figure 2.13 shows the combine architecture. This architecture was proposed by [17], and I build it using Tensorflow. In this architecture, the image features are not processed by the LSTM layers. The inject architecture will add the last word of LSTM and image features, and every element will be added together, like Figure 2.13 shows. The LSTM layers will only process the tensor of word embedding layer.

2.5 Word embedding

The coputer cannot recognise the words that humans can recognize, so I need to convert each word into a computer-recognizable code. One method of encoding words is one hot encoding[18] the other method of encoding is word vector. A word that encoded in one hot representation is one dimension, it is independent of each other. The one hot vector is usually used for classification. However, the fact that each word has nothing to do with each other is not consistent with our reality. We know a lot of words that are related like girl and woman. They are different words but they are semantically similar, so the distance between the two words in word vector should be close. So I use word vector to encoding my captions. In this section I will introduce how to train the word vectors [19].

2.5.1 Glove vectors

Train glove vectors

(1) Co-occurrence Probabilities Matrix First we use sliding window to slide through each sentences in the training set. And use X_{ij} to represent the number of word j appears in the context of word i. And $X_i = \sum_k X_{ik}$ represent the total number of words in the context of word i. $P_{ij} = \frac{X_{ij}}{X_i}$ is the probability that word j appears in word i context. The ratio of P_{ik} and P_{jk} can reflect the correlation between words.

So if we found the word vector w_i, w_j, w_k, they should be the same relation as $\frac{P_{ik}}{P_{jk}}$.

$$F(w_i, w_j, w_k) = \frac{P_{ik}}{P_{jk}} \tag{2.14}$$

(2) Final form of F We can use the difference of two vectors to represent a relationship between two vectors.

$$F((w_i - w_j), w_k) = \frac{P_{ik}}{P_{jk}} \tag{2.15}$$

And because the form of $\frac{P_{ik}}{P_{jk}}$ is a scalar, so the $F((w_i - w_j), w_k)$ is a scalar.

$$F((w_i - w_j)^{\mathrm{T}} w_k) = F(w_i^{\mathrm{T}} w_k - w_j^{\mathrm{T}} w_k) = \frac{P_{ik}}{P_{jk}} \tag{2.16}$$

And this form of F can become a form of division by adding a exp.

$$\exp(w_i^{\mathrm{T}} w_k - w_j^{\mathrm{T}} w_k) = \frac{\exp(w_i^{\mathrm{T}} w_k)}{\exp(w_j^{\mathrm{T}} w_k)} = \frac{P_{ik}}{P_{jk}} \tag{2.17}$$

Then we make the two numerators equal, and take logarithm

$$w_i^{\mathrm{T}} w_k = \log(P_{ik}) = \log(\frac{X_{ik}}{X_i}) = \log X_{ik} - \log(X_i) \tag{2.18}$$

(3) Loss function We make the left and right sides as close as possible, and construct a loss function

$$J = \sum_{ik} (w_i^{\mathrm{T}} w_k + b_i + b_k - \log(X_{ik}))^2 \tag{2.19}$$

And then we weighted each of terms.

$$J = \sum_{ik} f(X_{ik})(w_i^{\mathrm{T}} w_k + b_i + b_k - \log(X_{ik}))^2 \tag{2.20}$$

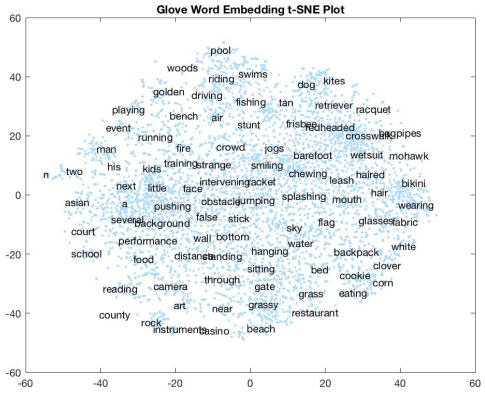
And the function of f in the term above is

$$f(x) = \begin{cases} (\frac{x}{x_{max}})^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{2.21}$$

And the best choice is $x_{max} = 100$, $\alpha = \frac{3}{4}$

Implementation

I download the pretrained glove word vectors from [20]. The filename is 'glove.6B.zip', which is pretrained by 6 billion token corpus, contains 400k vocabulary. And I extract the word vector from the text files, which have four kinds of dimensions 50, 100, 200, 300. I will make a comparison of this dimensions which I will write in my experiment section.



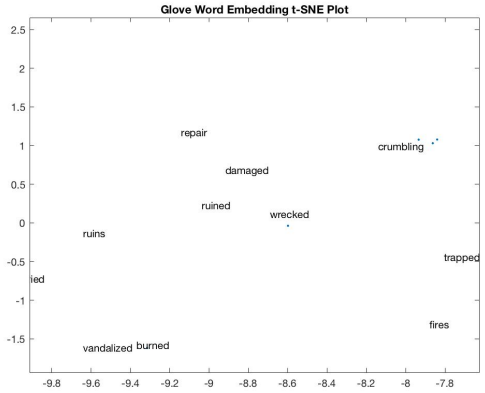


Fig. 2.14. The 50 dimensions of Glove word embedding t-SNE plot.

Fig. 2.15. Zoom in the glove word embedding plot.

The Figure 2.14 and 2.15 shows the distance relationship if each word vector. I transfer every word in training set of Flickr8k into 50 dimensions vector using glove pretrained vector[20]. Then I use the function tsne from Matlab to reduction the word dimensions to two dimensions and plot it on a two dimensions figure. Every blue point in the figure represents a word vector. The closer the two word vectors are, the greater their semantic similarity. As we can see in Figure 2.15, the three words "damaged", "ruined" and "wrecked" is close to each other, and they also have very similar semantics. I will compare different results between none pretrained word embedding and glove pretrained word embedding.

2.5.2 One-hot vectors

One-hot vector has only one element which is 1, and the rest of the other element is 0. The location of the element with a value of 1 can represent a class in a classification, the number of elements in the one-hot vector can represent the total number of categories. Each element of the one-hot vector can represent a category of classification.

Chapter 3

EXPERIMENTS AND RESULTS

3.1 Preparation work

My project will be implemented by Tensorflow and Keras. Tensorflow is a second generation artificial intelligence learning system developed by Google, and it can be used in many kinds of machine learning and deep learning areas such as voice recognition or image recognition, runs on everything from a smartphone to thousands of data central servers [21]. Keras is a high-level neural network API and it is written by Python and it is based on Tensorflow, Theano or CNTK backend [22]. Keras supports fast experimentation and it allowed me to quickly transfer my idea into results. I download some CNN weights from Keras and I use the PIL, scipy and matplotlib libraries to process and display images. And I use string, keras etc. to process texts and I use tensorflow, keras etc to build train, test dataset and evaluate the performance of CNN and RNN models with library keras.
The dataset I used in my project is Flickr8k which is collected by Yahoo and each image is annotated with five sentences from Amazon Mechanical Turk which is a networked crowdsourcing marketplace that allows individuals and businesses to coordinate using human intelligence to accomplish tasks that computers currently cannot[23]. Flickr8k contains about 8,000 images include 6,000 training set images and 1,000 validation set images, and the images in the dataset are selected by the user querying for specific objects and operations [24]. Each image contains five sentences to describe the content of the image, and the reason why I used flickr8k is that it is a very small dataset and I can build some models on my own CPU computer.

3.2 Image preprocessing

The images in dataset Flickr8k can be of various sizes, and CNN models need a 224 by 224 input of images. So I need to convert every image to size 224 by 224. There are two ways to crop images. The first method is to crop the image by proportion; the second is to crop the image in the central. I will write the results of the different two kinds of cropping methods in the experiment section.

3.2.1 Proportion crop

The proportion crop means the images are cropped by proportion, and it contains all the information that the original image has. And I use the function imresize from scipy library to crop the image directly. The Figure 3.1 and Figure 3.2 shows the an example of a image cropped by proportion.

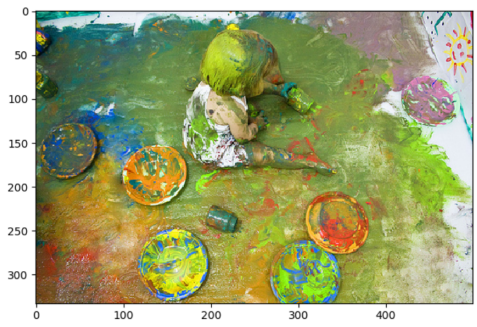


Fig. 3.1. The image before cropping

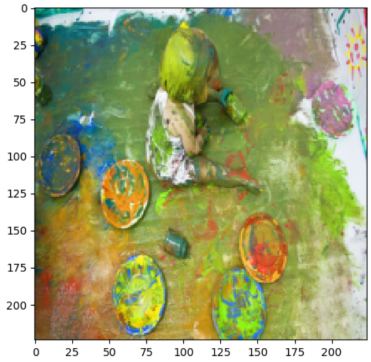


Fig. 3.2. The image after proportion crop

The Figure 3.1 is a image that size is 333 by 500 by 3. After proportion crop, the image become the size of 224 by 224 by 3 like the Figure 3.2 shows.

3.2.2 Central crop

The central crop means the images are cropped in a central area. That means the image will not keep all the information, and it will only keep the central area information. The Figure 3.3 and Figure 3.4 shows an example that a image cropped by a central area.

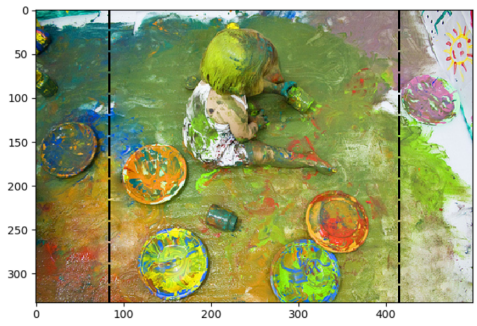


Fig. 3.3. The image before cropping

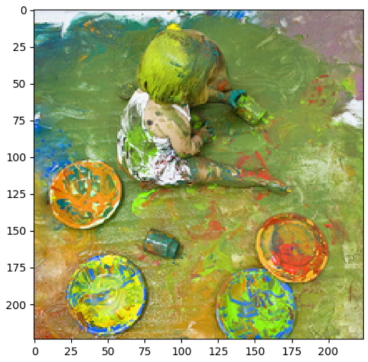


Fig. 3.4. The image after proportion crop

The Figure 3.3 shows the image before cropping, and the area between the two dotted black line is the cropped area. The Figure 3.4 shows the image after central crop, and we can see it contains the main information after cropping. And the central crop implemented by the following pseudocode.

Algorithm 1 Central Crop

Input: The shape of a image, width(**W**),height(**H**).

Output: The remove area from left side of origin image to left side of cropped image, which is denoted by **L**;
The remove area from right side of origin image to right side of cropped image, which is denoted by **R**;
The remove area from top side of origin image ro top side of cropped image, which is denoted by **T**;
The remove area from bottom side of origin image to bottom side of cropped image, which is denoted by **B**;

```
if W > H then
    L ← (W−H)/2
    R ← L
    T ← 0
    B ← 0
else
    T ← (H − W)/2
    B ← T
    L ← 0
    R ← 0
end if
```

3.3 Text preprocessing

3.3.1 Clean up text and construct Tokenizer

The flickr8k dataset has 8091 images and each image contains five description statements to describe the content of the image. First, I will clean up each description of images, that means I remove some punctuation of every sentence such as comma, period, etc. Figure 3.5 shows the relative percentage of these punctuations presented in the total number of words in all the sentences of the training set.

I change each capital word to a lowercase word. Next, I added **startsen** and **endsen** at the beginning and end of each sentence in training set to define the beginning and end of a sentence, so that my model could automatically recognise the beginning of a sentence and the end of a sentence after the training. Finally, I construct a Tokenizer for the 30,000 sentences of the training set. Now, this Tokenizer has a dictionary which contains 7,633 key-value pairs, and the key is all non-repeating words which come from 30,000 sentences.

key	a	child	plays	with	paint
value	1	43	111	10	620

Table 3.1: the example of some key-value pairs dictionary in tokenizer

3.3.2 Construct the training set and validation set

The input required by the decoder are image features and a single word. I need to feed the decoder image features that produced by encoders and word codes produced by the tokenizer. Then I will pad the word code into a length of 38 (the maximal length of all the sentences) and use 0 to fill in the digits just like Table 3.2 shows. I change every sentence of training set into these input form. Then I get roughly 350000 rows data. And then, each data will go through an embedding layer, and it will be changed to a word vector. The number of time steps of LSTM will be 38 because the maximal number of words in all sentences is 38. And then every column will be fed into 38 LSTM cells. The form of Y in Table 3.2 will change into one-hot encodings.

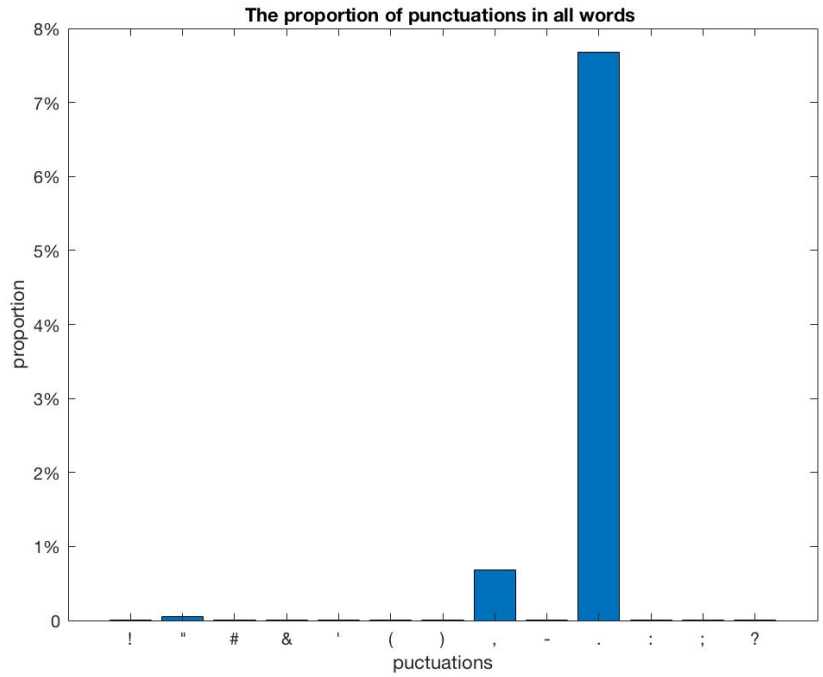


Fig. 3.5. Period accounts for about 7.8 percent of all words and comma accounts for about 9 percent of all words. And quotation marks account for about 2 percent of all words. These punctuation marks account for about 19 percent of all words. Therefore, clearing out these punctuation marks is an important step

The way to construct a validation set is the same with the way to construct a training set. The total number of images of validation set is 1000, and the total number of sentences of validation set is 5000.

sentence	startsen	a	child	plays	with	paint	endsen	Y
word code	2	1	43	111	10	620	3	
X	0 ... 0 0	0	0	0	0	0	2	1
	0 ... 0 0	0	0	0	0	2	1	43
	0 ... 0 0	0	0	0	2	1	43	111
	0 ... 0 0	0	0	2	1	43	111	10
	0 ... 0 0	0	2	1	43	111	10	620
	0 ... 0 0	2	1	43	111	10	620	3

Table 3.2: the example of turning a sentences into training set X and Y, the total columns of X is 38

3.4 Build encoders

3.4.1 Construct CNN models

I construct CNN models using keras and download the weights from keras. And then I remove the last layer of CNN models to keep the combinations of high-level image features. The CNN models are not used for classification, so the last layer is not needed in my project. And then I use these CNN models to extract images features of Flick8k dataset. I use combine architecture of RNN models as my decoder and use BLEU evaluation indicator to evaluate the performance of CNN models to find out what kind of CNN model is the best for image extraction in my project. And I repeated the training of my model 6 times and averaged the results to increase the robustness of my experimental results.

3.4.2 Result of five CNN models

Layers	BLEU1	BLEU2	BLEU3	BLEU4
50	0.348	0.190	0.095	0.051
100	0.312	0.172	0.088	0.047
200	0.387	0.214	0.115	0.066
300	0.397	0.217	0.114	0.064

Table 3.3: The result of Baseline CNN model

Layers	BLEU1	BLEU2	BLEU3	BLEU4
50	0.401	0.243	0.139	0.080
100	0.429	0.261	0.148	0.086
200	0.458	0.278	0.161	0.095
300	0.456	0.281	0.167	0.101

Table 3.4: The result of VGG16 model

Layers	BLEU1	BLEU2	BLEU3	BLEU4
50	0.436	0.265	0.151	0.086
100	0.412	0.250	0.142	0.082
200	0.455	0.279	0.164	0.097
300	0.482	0.298	0.177	0.105

Table 3.5: The result of VGG19 model

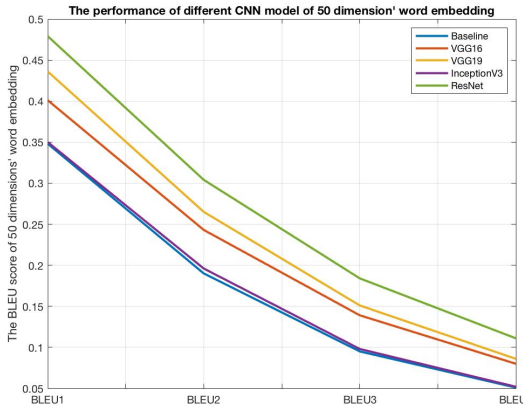
Layers	BLEU1	BLEU2	BLEU3	BLEU4
50	0.350	0.196	0.098	0.052
100	0.378	0.212	0.108	0.059
200	0.337	0.177	0.088	0.046
300	0.296	0.162	0.085	0.047

Table 3.6: The result of Inception model

Layers	BLEU1	BLEU2	BLEU3	BLEU4
50	0.479	0.304	0.184	0.111
100	0.421	0.265	0.162	0.099
200	0.508	0.322	0.197	0.119
300	0.589	0.394	0.257	0.165

Table 3.7: The result of ResNet model

I use keras construct 5 different CNN models to extract image features. Then I will feed these image features to a **combine decoder architecture**. Table 3.3 to Table 3.7 shows the experiment result of each CNN models' performance with the same decoder model. I test the different dimensions of word embedding to see it whether the BLEU score will increase with the same CNN model. As we can see from these tables, as word embedding's dimensions increased, each cumulative weight of BLEU score(BLUE1,BLUE2,BLUE3,BLUE4) is overall increased. The reason is that the more dimensions that word embedding have, the more information will be fed to the decoder model. Ignore some experimental error, on the basis of the decoder model, the BLEU score increase as the dimensions of word embedding increase.



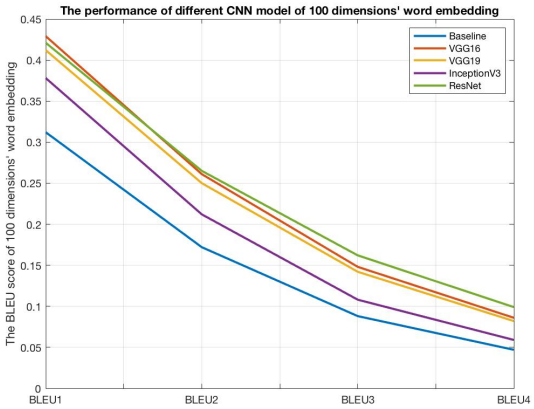
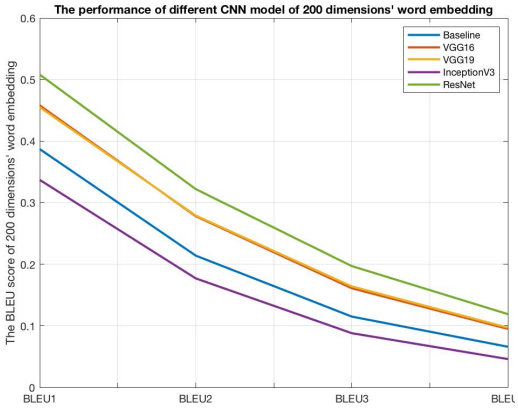


Fig. 3.6. 50 dimensions' word embedding **Fig. 3.7.** 100 dimensions' word embedding



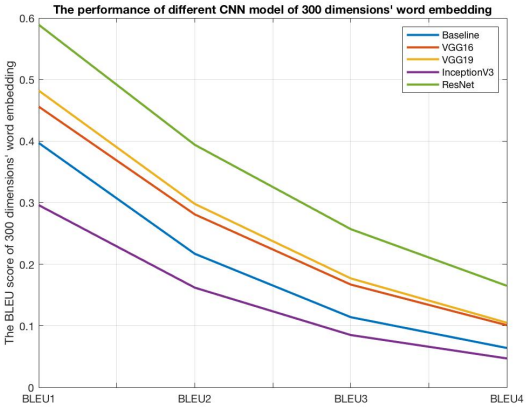


Fig. 3.8. 200 dimensions' word embedding **Fig. 3.9.** 300 dimensions' word embedding

Figure 3.6 to Figure 3.9 compare the performance of different CNN models. X axis represents the different weight of BLEU score, and the purpose is to see a different kind of results of n-gram. Y axis represents the specific BLEU score. Red line represents the VGG16 model, yellow line represent the VGG19 model, purple line represent the InceptionV3 model, and Green line represents the ResNet model. Each figure represents a different kind of dimensions that word embedding has. As we can see from these figures, ignore some experimental error, the ResNet get the highest score in each BLEU of each dimension. And the InceptionV3 and my baseline CNN model got the lowest BLEU score respectively in different dimensions. In 50 and 100 dimensions, the InceptionV3 model gets the lowest BLEU score. And in 200 and 300 dimensions, my baseline CNN model get the lowest BLEU score. In these CNN model, the depth increases as the CNN change **Baseline-VGG16-VGG19-ResNet50-InceptionV3**. In general, if the depth of CNN increased, the performance of CNN as an encoder will increase. However the deepest layer InceptionV3 decomposed some big kernel convolution, it may break the integrity of the features extracted by kernel convolution. As a conclusion, the ResNet50 is the best CNN model as an encoder to extract image features in the roles of Image Captioning.

3.5 Build decoder

3.5.1 The status of RNN model

RNN is a deep learning model that can learn the associations between words through context. And if I extract words from a large number of sentences as the training set fed to RNN model, the RNN model can learn the grammar rules, the Figure 3.10 shows the change of BLEU scores when the number of epochs of training RNN model

increase and the Table 3.8 shows the changes in sentences when the number of epochs increases.

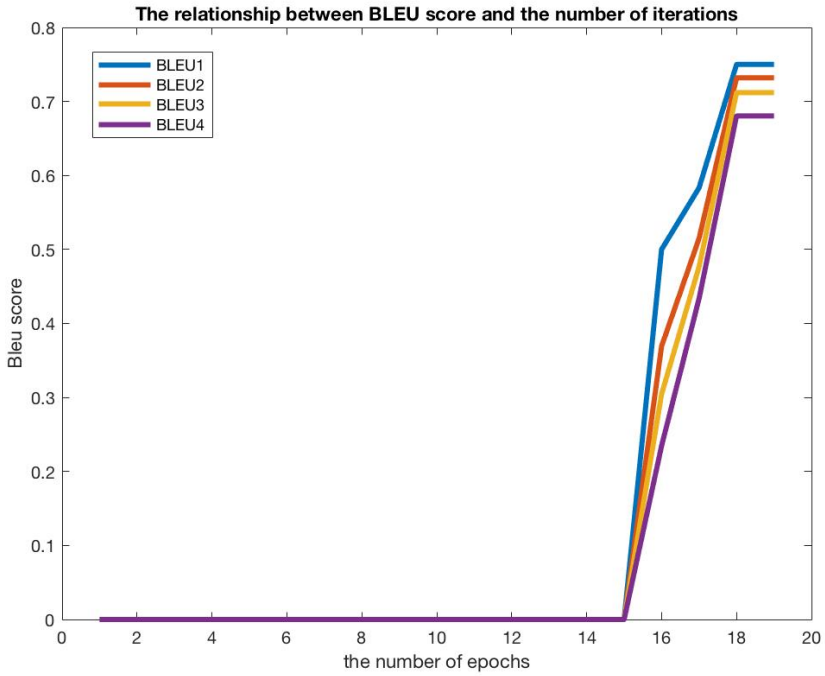


Fig. 3.10. The total number of epochs is roughly 20000, and this figure records the change of BLEU score per 100 epochs, and the BLEU score increase started at roughly 1500 epochs and ended with roughly 1800 epochs.

image	epochs	sentences made by RNN models
	1	startsen ghostbuster ... ghostbusterr
	100	startsen a a a a a a ... a
	200	startsen dog a dog a dog a ... a
	300	startsen a dog dog in a in endsen
	400	startsen a man in a in a in a ... in a
	500	startsen a dog dog in a dog endsen
	...	
	1400	startsen a man is on a hill endsen
	1500	startsen a person is riding a bike is endsen
	1600	startsen a person is riding a bike endsen
	1700	startsen a person is riding a bike endsen
	1800	startsen a person is riding a bike endsen

Table 3.8: The table shows the changes of sentences as the increase in numbers of epochs, before the 1400 epochs the sentences are disorganised and have no grammatical rules at all. When the epochs reach 1400, RNN model learn the grammar rules, and as the increase of epochs, the sentences produced by RNN models are close to the sentences which can describe the photos.

3.5.2 Performance of different decoder

However, knowing the syntax is not enough, I also need to pass image features into the RNN model as another input parameter. In this way, the RNN model can learn the relationship between grammatical sentences and images. There are three different

kinds of RNN models and have the input parameters of image features appear in different locations.

Inject architecture

In inject architecture, the image features appear in the top of embedding layers, the first cell of LSTM will process the image features. Therefore, when the LSTM process the word sequence, it will contain the image information. I build this decoder architecture by Tensorflow. And I test the different dimensions of word embedding and the image cropped by the way of proportion or central.

Dimensions	BLEU1	BLEU2	BLEU3	BLEU4
Inject	None Glove	None Glove	None Glove	None Glove
50 procrop	**0.500** 0.476	**0.225** 0.120	**0.105** 0.032	**0.055** 0.014
100 procrop	**0.497** 0.419	**0.223** 0.087	**0.099** 0.011	**0.055** 0.001
200 procrop	**0.495** 0.474	**0.230** 0.111	**0.113** 0.019	**0.063** 0.001
300 procrop	**0.495** 0.432	**0.230** 0.190	**0.113** 0.083	**0.063** 0.042
50 cencrop	**0.494** 0.480	**0.117** 0.107	0.014 **0.015**	0.003 0.003
100 cencrop	**0.471** 0.461	**0.209** 0.082	**0.095** 0.010	**0.052** 0.002
200 cencrop	**0.490** 0.444	**0.229** 0.041	**0.109** 0.001	**0.058** 0.001
300 cencrop	**0.472** 0.432	**0.216** 0.038	**0.101** 0.001	**0.053** 0.001

Table 3.9: The table shows results of inject architecture. And the table shows results of comparison on different dimensions of word embedding. There are four dimensions in the table from 50,100,200,300. Moreover, it shows results of comparison between Glove word embedding and none pretrained word embedding, the label **None** denotes none pretrained word embedding and the label **Glove** denotes Glove word embedding. At last, it shows the results of different crop method which are proportion crop and central crop. The term **procrop** denotes the proportion crop method, and term **cencrop** denotes the central crop method.

Combine architecture

In combine architecture of decoder, the image features do not appear in LSTM layer. The image features will be processed after the LSTM finish the process of word sequence by adding the output of LSTM and image features. Therefore, the LSTM will not contain the information of images.

Dimensions	BLEU1	BLEU2	BLEU3	BLEU4
Combine	None Glove	None Glove	None Glove	None Glove
50 procrop	**0.451** 0.448	**0.288** 0.283	**0.177** 0.170	**0.107** 0.101
100 procrop	**0.486** 0.454	**0.311** 0.285	**0.192** 0.171	**0.117** 0.103
200 procrop	**0.508** 0.497	**0.325** 0.312	**0.201** 0.190	**0.123** 0.116
300 procrop	**0.524** 0.500	**0.336** 0.317	**0.210** 0.195	**0.130** 0.121
50 cencrop	**0.478** 0.463	**0.306** 0.293	**0.188** 0.177	**0.114** 0.106
100 cencrop	**0.495** 0.480	**0.318** 0.305	**0.194** 0.185	**0.117** 0.111
200 cencrop	**0.505** 0.490	**0.322** 0.301	**0.200** 0.191	**0.122** 0.118
300 cencrop	**0.527** 0.502	**0.339** 0.317	**0.212** 0.194	**0.132** 0.119

Table 3.10: The table shows results of combine architecture. The table shows results of comparison on different dimensions of word embedding. There are four dimensions in the table from 50,100,200,300. Moreover, it shows results of comparison between Glove word embedding and none pretrained word embedding, the label **None** denotes none pretrained word embedding and the label **Glove** denotes Glove word embedding. At last, it shows the results of different crop method which are proportion crop and central crop. The term **procrop** denotes the proportion crop method, and term **cencrop** denotes the central crop method.

Cross architecture

In cross architecture, I make the image features distribute inside the tensor of word embedding. The LSTM contains a lot of image information.

Dimensions	BLEU1		BLEU2		BLEU3		BLEU4	
cross	None	Glove	None	Glove	None	Glove	None	Glove
50 procrop	0.458	**0.508**	0.280	**0.315**	0.161	**0.182**	0.092	**0.104**
100 procrop	0.484	**0.522**	0.302	**0.333**	0.178	**0.198**	0.104	**0.117**
200 procrop	0.470	**0.504**	0.300	**0.320**	0.176	**0.192**	0.104	**0.115**
300 procrop	0.490	**0.496**	0.305	**0.314**	0.179	**0.189**	0.104	**0.114**
50 cencrop	0.477	**0.487**	0.287	**0.300**	0.159	**0.169**	0.088	**0.095**
100 cencrop	0.493	0.493	0.311	**0.312**	0.184	**0.186**	0.108	**0.110**
200 cencrop	0.499	**0.515**	0.313	**0.327**	0.186	**0.198**	0.110	**0.119**
300 cencrop	0.496	**0.517**	0.307	**0.331**	0.179	**0.202**	0.105	**0.122**

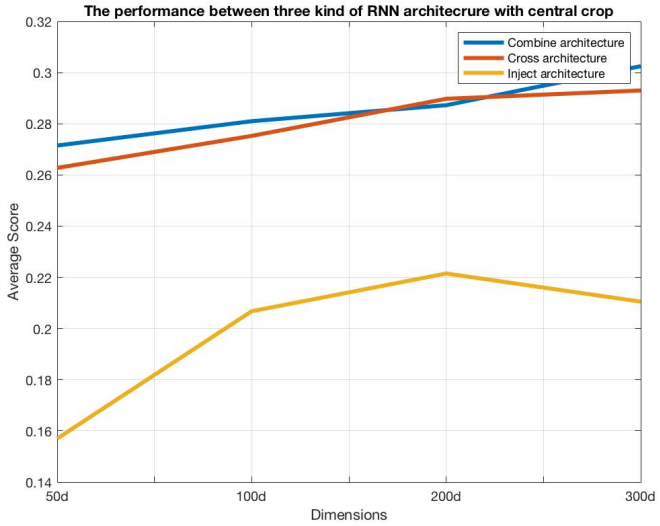
Table 3.11: The table shows results of cross architecture. The table shows results of comparison on different dimensions of word embedding. There are four dimensions in the table from 50,100,200,300. Moreover, it shows results of comparison between Glove word embedding and none pretrained word embedding, the label **None** denotes none pretrained word embedding and the label **Glove** denotes Glove word embedding. At last, it shows the results of different crop method which are proportion crop and central crop. The term **procrop** denotes the proportion crop method, and term **cencrop** denotes the central crop method.

Analyse the results

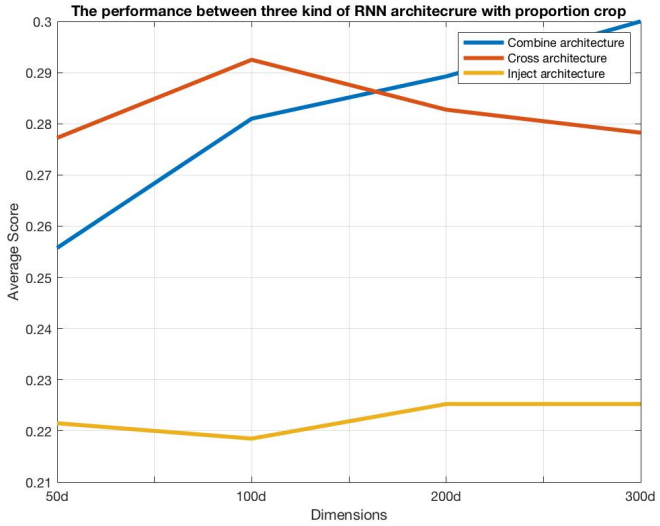
As we can see from Tab. 3.9 to Tab. 3.11, the performance of cross architecture with Glove word embedding is better than that with none pretrained word embedding. And the performance of combine architecture with Gove word embedding is also better than that with none pretrained word. But the performance of architecture with Glove word embedding is worse than that with none pretrained word embedding. The reason I think is that the cross architecture contains much more image feature vectors, so the semantic similarity of word vectors cannot be guaranteed after words and image features are alternately processed by LSTM. But the combine architecture and inject architecture contains less image feature vectors so that the word embedding will be trained with semantic similarity.
As the dimensions of word embedding increase, the performance of combine architecture becomes better than before, and the performance of inject architecture and cross architecture is not related to the dimension of word vectors. The reason I think is that LSTM of combine architecture would not process the image feature vectors, the LSTM only process word vectors. So the image features will not be influenced by the changes of dimensions of vectors. The larger the dimensions of the word vector, the more information the word vector contains, so the performance with larger dimensions is good. The LSTM of inject architecture and cross architecture will process the image feature vectors so that the LSTM will process the image feature vectors, and the image features may not lose some image information.
When I use central crop methods, the combine architecture and cross architecture both shows better performance than proportion crop method. However, the inject architecture shows worse performance than proportion crop method. The reason I think is that the central crop method preserves the scale of objects in the image before cropping, and most of the information in images concentrated at the central area, so it also contains a lot of information. The output of LSTM in combine architecture will not contain image features. The image features will add to the output of LSTM directly, so it keeps the information that image features have. Therefore, it makes the advantage of central method work. And the output of LSTM in inject architecture will forget a lot of information of images because the image features only

appear in the start of the time series. Cross architecture will contain a lot of image features in the time series of LSTM, so the output of LSTM will contains a lot of image information.

Compare with different architecture



(1) The performance between three kind of RNN architecture with central crop



(2) The performance between three kind of RNN architecture with propotion crop

Fig. 3.11. Comparison between different RNN architecture. Legend:Average Score - the average value of Bleu1 to Bleu4; Dimension - the dimensions of word embedding; Blue line - combine architecture; Red line - cross architecture; Yellow line - inject architecture;

As the Figure 3.11 shows below, I compare the None pretrained word embedding of inject architecture, the None pretrained word embedding of combine architecture and Glove word embedding of cross architecture. The best architecture in central crop method is to combine architecture. The best architecture in proportion crop method depends on dimensions of word embedding. Cross architecture get a good performance at 50 dimensions and 100 dimensions. Inject architecture get a good performance at 200 dimensions and 300 dimensions.

3.5.3 Results between different models

As the Figure 3.12 shows, the random algorithm will not process any image information and caption information which result is very bad. RNN model will not process any image information but will process caption information which result is more good than the random algorithm. CNN+RNN model will process not only image information but also process caption information which result is the best.

Dimensions	BLEU1	BLEU2	BLEU3	BLEU4
random algorithm	0.693e-3	0.825e-3	0.994e-3	0.976e-3
RNN model	0.247	0.136	0.068	0.036
CNN+RNN model	0.527	0.339	0.212	0.132

Table 3.12: Results of different image captioning algorithm. legend:random algorithm - randomly generate sentences; RNN model - the image features are random; CNN+RNN model:ResNet50+Combine architecture

3.6 Training details

If I use a CPU to train my model, it is very slow. It takes me nearly 3 hours to train a model. So I decided to use GPU in the University of Southampton. The GPU I used in university is a four-core GPU which is GeForce GTX 1080 made by Nvidia. And I search for some tutorials to configure Cuda and Cudnn. After using the GPU to train my model, it takes only 30 minutes to train a model. During the training process for the three kinds of RNN model, the result produced by my model is different for each training. It because the initialisation of variables is random, and because I want to get a more stable result, each model I will train six times and take the average result of the six times. Every time of training, I will use a random batch data to feed the model. For the first time, I used 32 batch data, and I get better performance than 128 batch data, but it takes more time for training. Because my project is not to reach the state-of-the-art performance and in order to save time I decided to use 128 batch data to train models. The number of epochs I will set to 3 times. And every epoch I will use a validation dataset to get the validation loss value, if the validation loss value is bigger than the last time, I will stop training. I did this because I want to avoid overfitting.

3.7 Generate new captions

I will feed a start word **startsen** and an image feature for my model. My model will predict the next possible word for this input word as the Figure 3.12 shows, the next most likely word for **a** is **child**, and the next most likely word for **child** is **plays**, and the next most likely word for **plays** is **with**, and the next most likely word for **with** is **paint**.

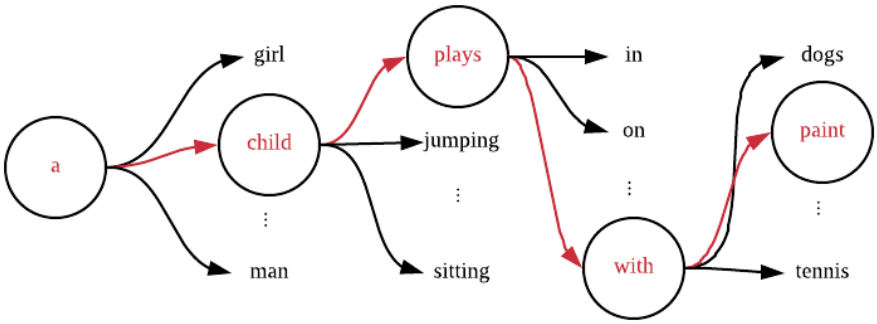


Fig. 3.12. the example of predicting the next word

After I get the prediction of the second word by my model, I will feed the first two words and image features to my model. And the model will predict the third most possibly word. The process of prediction will end with finding the word **endsen** or the length of sentence reach 38(the maximal length of sentences in the Flickr8k dataset)

a dog is running through the grass

a dog is running through the water

a man is riding a bike in the air

a group of people are sitting on a bench

Fig. 3.13. Some example of caption generated by ResNet50+Inject architecture

As we can see from the Figure 3.13, machines generate good captions in the first two images in the first row. But machine generates relatively bad captions in two images of the second row, the caption "a man is riding a bike in the air" generated by machines should be changed into a caption that "a man is playing a skateboard in the air". The machine regards a skateboard as a bike. The machine might have thought the man was in a similar position to riding a bicycle, and the yellow skateboard is extremely similar to a bike in the training set. The caption "a group of people are sitting on a bench" generated by machines only recognise "a group of people", machine may regards the half body picture of these people as they are sitting.

Chapter 4

PROJECT MANAGEMENT

4.1 Time management

The ability of time management is important for a project, and good time management can effectively push the project forward. In order to manage the image of my project well, I use a Gantt Chart to manage my time. And a Gantt Chart is shown in Figure. 4.1.

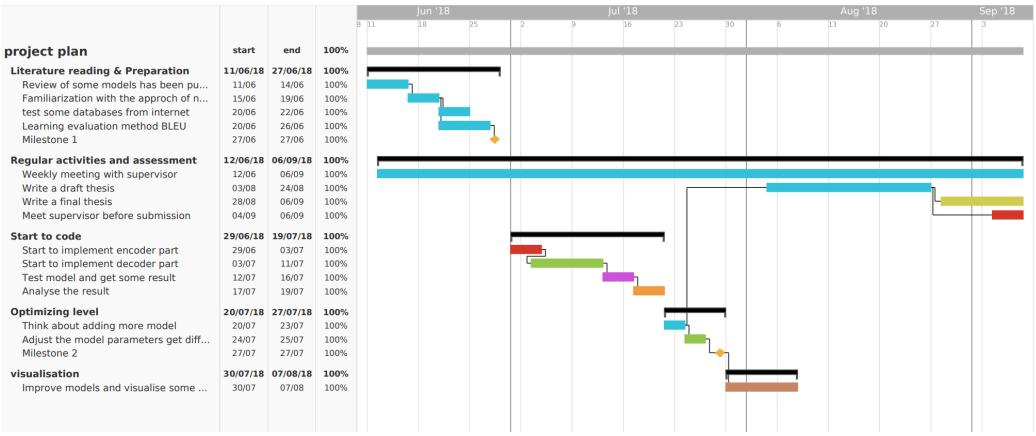


Fig. 4.1

I started my project on the 11th of June. At the first week, I start to read some paper related to my project. And I reviewed some Neural Network Model that I have been learned from, especially on CNN and RNN models, because this two model is the main method to implement Image Captioning. Moreover, I started to meet my project supervisor every week on Thursday, and my project supervisor gave me lots of useful guidance such as the time to write my dissertation and some evaluation indicator. Furthermore, I started to code in the 29th of June, and it took a longer time than I expected. So I separate a day into two part, one part is to code at night, the other is to write my dissertation in the daytime.

Chapter 5

CONCLUSION

5.1 Conclusion

In conclusion, this paper analyses what the best CNN and RNN for image captioning is. And the paper use five different CNN model and three different RNN architecture.

As the experiment result shows, the ResNet CNN model got the highest score than the other four CNN model. It is mainly because the deepest of depth. The InceptionV3 CNN model has a deeper depth than ResNet, but the kernel has been decomposed to combinations of a small size of kernels, and it may destroy the features that image captioning need. Therefore, the depth of the CNN model with no modification of kernel determines the performance of CNN as a role of an encoder. As CNN's depth increases, CNN's performance as an encoder gradually increases.

The performance of three different RNN model is determined by an image processing method which is a central crop method or proportion crop method. And their performance is also determined by the dimensions that word embedding have. The combine architecture got the highest score than other two RNN architecture in central crop method. It is mainly because the central crop method enhances the performance of combine architecture. The output of LSTM in combine architecture add image features directly, the central crop method improve combine architecture's performance. The other evidence for this argument is that the combine performance looks bad when it use proportion crop method.

As the proportion crop method, the performance of cross architecture is better than that of combine architecture with 50 dimensions of word vectors and 100 dimensions of word vectors. And the performance of combine architecture is better than that of cross architecture with 200 dimensions of word vectors and 300 dimensions of word vectors.

5.2 Future work

This paper is to find out what is the best CNN model for encoding and what is the best RNN model for decoding. And the model I did not try to make it reach the state-of-the-art performance mainly because the three months time is limited, so in the future, I will do the following work:

(1) Compare more dataset The flickr8k dataset is enough for my project, but the images of the Flickr8k dataset only described the activation of human or related to human. It does not involve much more another object that has nothing to do with humans. Moreover, the number of Flick8k is too small. In order to improve the performance of my model, I should increase the number of the dataset. So I will add

Microsoft COCO dataset in my project to compare with the Flickr8k dataset, which has nearly 330k images.

(2) Beam search The way that model always use the output word with maximum probability as the second words are not the best way to generate new captions. The search methods I use is a greedy search, and the greedy search can only find the local optimal solution in generating new captions, it can not find the global optimal solution. So I will use a beam search to find generate new captions, and it can be as close as possible to the global optimal solution.

(3) Semantic analysis The model that CNN+RNN is a method to directly apply image features to text, instead of analysing image features with advanced semantics first. So [2] find a way to transfer the images into a high-level semantic feature vector. The main idea is the CNN not just to get the last two layers' output as the image features; its last layer is a softmax activation function's multi-label classification layer. But it takes much time to train a model from scratch, so I can use the method of transferring learning to transfer a single-label classification CNN parameter to a multi-label classification CNN model and fine tune the CNN model. If I use CNN to get a vector that contains the vocabulary, then feeds the vector to RNN, perhaps I will get a better result.

Bibliography

[1] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille, "Explain images with multimodal recurrent neural networks," *arXiv preprint arXiv:1410.1090*, 2014.

[2] Q. Wu, C. Shen, L. Liu, A. Dick, and A. van den Hengel, "What value do explicit high level concepts have in vision to language problems?," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 203–212, 2016.

[3] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[4] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156–3164, 2015.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[7] Z. Wu, C. Shen, and A. v. d. Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *arXiv preprint arXiv:1611.10080*, 2016.

[8] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318, Association for Computational Linguistics, 2002.

[9] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks.," in *ICML*, pp. 507–516, 2016.

[10] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[12] B. Zeng, Q. Huang, A. El Saddik, H. Li, S. Jiang, and X. Fan, *Advances in Multimedia Information Processing–PCM 2017: 18th Pacific-Rim Conference on Multimedia, Harbin, China, September 28-29, 2017, Revised Selected Papers*, vol. 10736. Springer, 2018.

[13] A. Deshpande, "A beginner's guide to understanding convolutional neural networks @Online." https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/, Sept. 2016. [Accessed: 29- Aug- 2018].

[14] H. Hassannejad, G. Matrella, P. Ciampolini, I. De Munari, M. Mordonini, and S. Cagnoni, "Food image recognition using very deep convolutional networks," in *Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management*, pp. 41–49, ACM, 2016.

[15] "Understanding lstm networks @Online." http://colah.github.io/posts/2015-08-Understanding-LSTMs/, Aug. 2015. [Accessed: 31- Aug- 2018].

[16] R. Fu, Z. Zhang, and L. Li, "Using lstm and gru neural network methods for traffic flow prediction," in *Chinese Association of Automation (YAC), Youth Academic Annual Conference of*, pp. 324–328, IEEE, 2016.

[17] M. Tanti, A. Gatt, and K. P. Camilleri, "What is the role of recurrent neural networks (rnns) in an image caption generator?," *arXiv preprint arXiv:1708.02043*, 2017.

[18] R. Vasudev, "What is one hot encoding? why and when do you have to use it? @Online." https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f, Aug. 2017. [Accessed: 23- Aug- 2018].

[19] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[20] C. D. M. Jeffrey Pennington, Richard Socher, "Glove: Global vectors for word representation@Online." https://nlp.stanford.edu/projects/glove/, 2014. [Accessed: 31- Aug- 2018].

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: large-scale machine learning on heterogeneous distributed systems. arxiv preprint (2016)," *arXiv preprint arXiv:1603.04467*.

[22] F. Chollet *et al.*, "Keras: Deep learning library for theano and tensorflow.(2015)," *There is no corresponding record for this reference*, 2015.

[23] "@Online." `https://www.mturk.com/worker/help/`. [Accessed: 22- Aug- 2018].

[24] W. Havard, L. Besacier, and O. Rosec, "Speech-coco: 600k visually grounded spoken captions aligned to mscoco data set," *arXiv preprint arXiv:1707.08435*, 2017.