香港中文大學
The Chinese University of Hong Kong

Visual Question Answering with Deep Learning

Submitted by:
Tu Fengzhi (1155058610)

Supervisor:
Prof. Michael Lyu

Final Year Project report presented to
Faculty of Engineering of The Chinese University of Hong Kong
in partial fulfilment of the requirements of the degree of
Bachelor of Computer Science

2017

TABLE OF CONTENTS

Chapter 1: INTRODUCTION

1.1 OVERVIEW

In March 2016, a computer program named AlphaGo which is developed by Google DeepMind beat Lee Sedol in playing the board game Go and it was the first time a computer program beat human 9-dan professional in game Go because Go is one of most difficult games that have a large search space. AlphaGo uses algorithms in deep learning to “learn” from the previous game record both from top human players and computer-to-computer records. The success of AlphaGo makes deep learning become a popular topic in computer science and attracts more and more researcher dive into the deep learning area.

Recent years, more and more researchers in both university and industry devote them into the field of deep learning and the number of published articles about deep learning algorithms have exploded. Both the universities and companies are pushing forward to speed up the developments and research of this area since the deep learning can solve a lot of problems better than classic machine learning algorithm. Deep Neural Network are widely used nowadays in our daily life such as recommendation systems, automatic translation, auto-drive car, etc.

Among all the popular deep learning problems, we are interested in the problem that joining image and text based on the classic question-answering problem. This problem combines computer vision problems such as image caption and object detection and some text process problems such as natural language processing.

The objective of our project is to use the deep learning method to solve Visual Question Answering problem. Visual Question Answering is a new dataset which include both open-end questions and multiple choice questions about images, it allows us to rise a question to an image, and the deep learning model will answer the question based on the image. Visual Question Answering can be regarded as a promotion of classic text based Question Answering problem. From the previous works, we can find that the most common method to solve this Visual Question Answering problem is to use Convolutional Neural Network to extract visual features from images first. Then using the Long Short-Term Memory (a kind of Recurrent Neural Network) to process the embedded natural language question vectors.

In the last semester, we have built a simple model which can answer the questions based on a given image. But there were many drawbacks in that model, such as low accuracy, extract the image information from the whole image, etc. In this semester, we decide to improve the model to achieve a higher accuracy and make it a more powerful model.

1.2 STATEMENT OF PURPOSE

The task of our final year project in the second semester is to come up with a new model about Visual Question Answering to enhance the accuracy of previous models since there are many limitations in previous models.

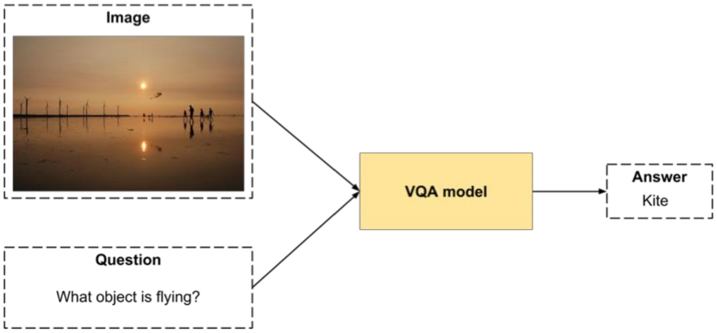


Figure 1: Visual Question Answering

Based on the model of the last semester, the objectives of the second semester are:

- Using new word embedding to substitute the current word embedding algorithm because of the limitation of current algorithm.

- Extract local information from an image instead of extract the whole image as an input.

- Extend the number of language support. For example, French, German.

- Extend the input from image to video. i.e. input is a video and a question, then the model will output the answer based on the input material.

Chapter 2: STUDY OF TECHNOLOGY

2.1 TENSORFLOW

2.1.1 Introduction to TensorFlow

Figure 2: Logo of TensorFlow

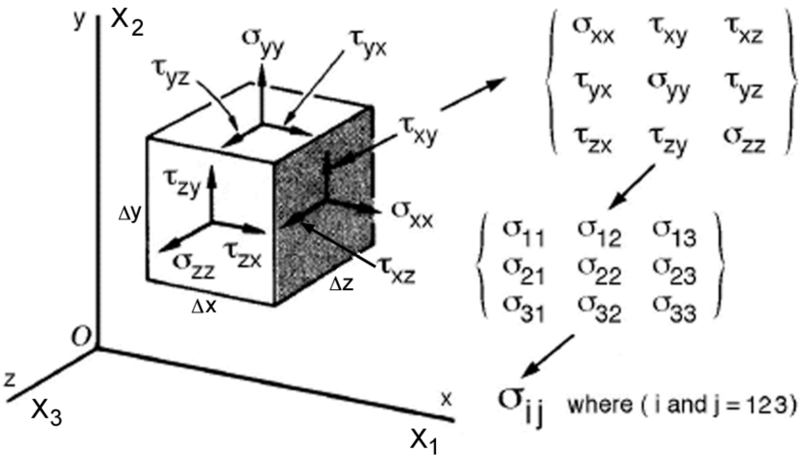
In our project, we use a famous deep learning platform TensorFlow, which support various of APIs of deep learning algorithms such as CNN (Convolutional Neural Network), RNN (Recurrent Neural Network) and LSTM (Long Short-Term Memory) which are the most popular deep learning algorithm models in the fields of image processing, speech process and natural language processing. TensorFlow is an open source library for machine learning which is developed by Google brain team and it is based on DistBelief and it is named TensorFlow because of its methodology.

There are two basic but important objects in TensorFlow, one is the Tensor, another is the data flow graph. The basic principle of TensorFlow is to do operation using Tensor. Tensor is a N-dimension array and Flow means the calculation is based on the data flow graph. And the process of “TensorFlow” is the flow of tensor from a node of date flow graph to another node of graph. TensorFlow is a system that transfers the complex data structures to the artificial neural network and analysis and process the input data.

2.1.2 Tensor

Formally, Tensors are multiple maps from vector spaces to the real number and its linear form function that can be used to represent the relationship of geometric vectors, scalars and other tensors. The relationship can be inner product, outer product and linear mapping and Cartesian product. In informal way, a tensor is a typed multi-dimensional array.

The following figure is an example of a tensor. As we can see, the coordinate of a tensor is in a N-Dimension space and contains n^r vectors.



y
X_2
σ_{yy}
τ_{yx}
τ_{yz}
τ_{xy}
τ_{zy}
Δy
σ_{xx}
σ_{zz}
τ_{zx}
τ_{xz}
Δz
Δx
O
z
X_3
x
X_1
σ_{xx} τ_{xy} τ_{xz}
τ_{yx} σ_{yy} τ_{yz}
τ_{zx} τ_{zy} σ_{zz}
σ_{11} σ_{12} σ_{13}
σ_{21} σ_{22} σ_{23}
σ_{31} σ_{32} σ_{33}
σ_{ij} where (i and j = 123)

Figure 3: Envisioning n-th order Tensors

2.1.3 Data Flow Graph

The data flow graph is the computation model using directed graph, nodes and edges. In this graph, nodes not only represent the mathematics operations, but also represent the operation that read/write the global variable, endpoints for feeding data and endpoints for pushing out result. Edges represent the communication between nodes and indicates the relationship between nodes by transiting tensor, multidimensional data arrays, between nodes. When running the TensorFlow program, the nodes will be assigned to computational devices like CPU and GPU, and each node can run in parallel asynchronously and execute instructions when data of all its incoming edges is really.

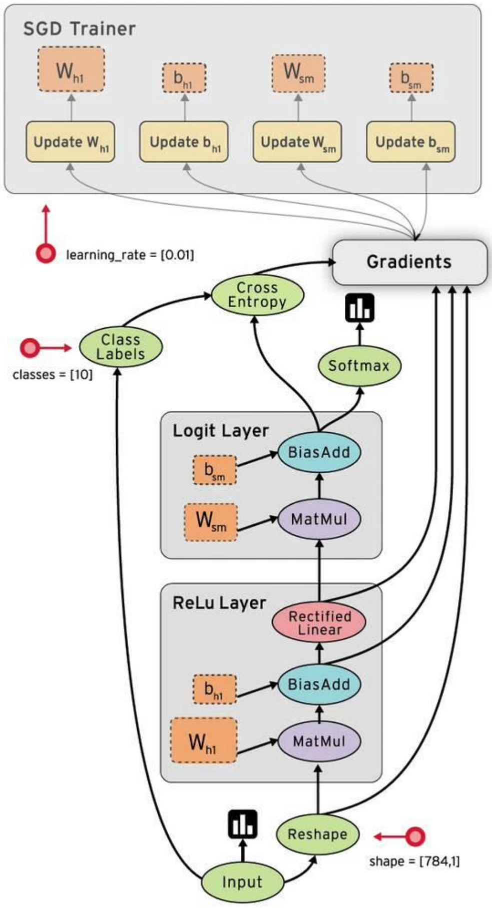


Figure 4: Data Flow Graph

2.1.4 Advantages of TensorFlow

- Flexibility

TensorFlow is not a strictly library of "neural network". You can use TensorFlow as if you can represent your calculation as a data flow graph. You can construct the graph and the internal circulation in your driver. TensorFlow provides useful tools to help assemble the sub-graph. Users can write the "upper library” based on the library of TensorFlow, users can also write the C++ program in the lower layer to optimize the operations in the TensorFlow.

- Portability

TensorFlow can run normally on both CPU and GPU. It can work on the laptops, servers and mobile phones.

- Connect research and product

In the past days, if you want to implement your ideas in machine learning, you need to program plenty of codes. Nowadays, using TensorFlow can simplify the implementation of ideas and increase the efficiency of the research.

- Auto-differentiate

Algorithms of machine learning which rely on gradient can be benefit of the auto-differentiate function of TensorFlow. Users only need to define the the structure of model and combine the structure with the objective function, then TensorFlow will automatically differentiate the related calculation with the given data.

- Multiple Programming Language Support

TensorFlow has a C++ and Python user interface to help user construct and execute the data flow graph. Users can write C++/Python programs directly or they can also use iPython interface to try some ideas.

- Performance Optimization

TensorFlow not only supports a single machine mode but also supports a distributed system. It also supports thread, queue, asynchronous and so on.

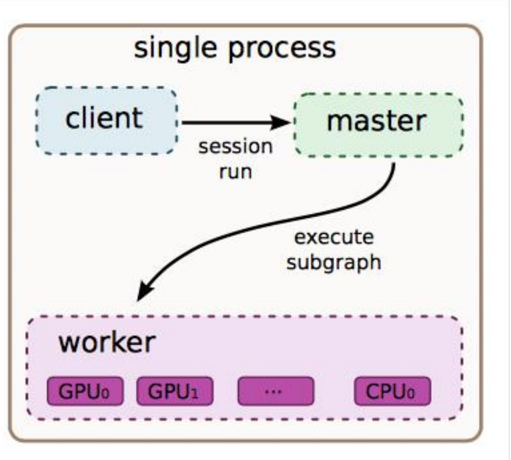


Figure 5: Single Machine Mode of TensorFlow

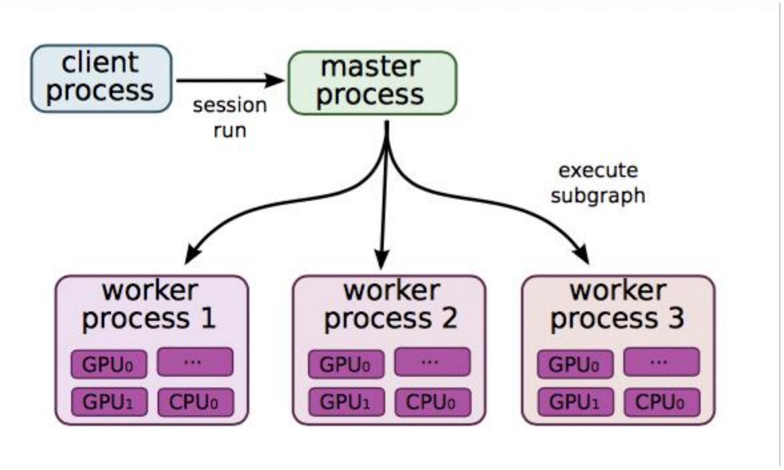


Figure 6: Distributed Model of TensorFlow

Thus, when the work load is too heavy for single machine, we can use a distributed system to reduce the runtime of the operations.

2.2 FEEDFORWARD NEURAL NETWORK

Before we understand the recurrent neural network, we need to introduce the feedforward neural network. The feedforward neural network is named after the way they channel information propagate at the nodes of the network. Each node in the network feeds information straight through and there is no connection between the nodes in the same layer of the network.

2.2.1 Multi-Layer Neural Network

In the feedforward neural network, each train sample are fed into the network, after a series of mathematical operations, finally the network has an output. In the supervised learning, the output of a training sample would be a label.

The following figure is the structure of a feedforward neural network:

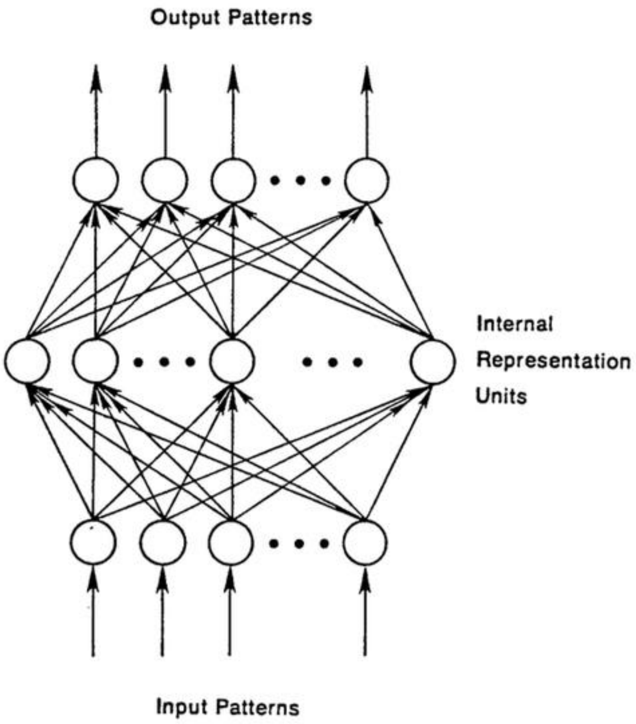


Figure 7: Overview of Feedforward Neural Network

2.2.2 Computational Unit - Neuron

The structure of nodes in the above network is as following:

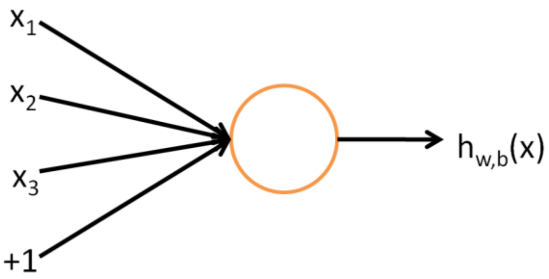


Figure 8: Diagram of Single Neuron

The neuron in the above case is the computational unit of neural network and it take x_1, x_2, x_3 (and +1 intercept item) as input and output $h_{w,b}(x)$.

$$h_{w,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$$

The f in the above equation is called activation equation. There are several kinds of activation equation, such as sigmoid function, tanh function, rectified linear function and so on.

- Sigmoid function: $f(z) = \frac{1}{1+\exp(-z)}$
- tanh function: $f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
- Rectified linear function: $f(z) = \max(0, z)$

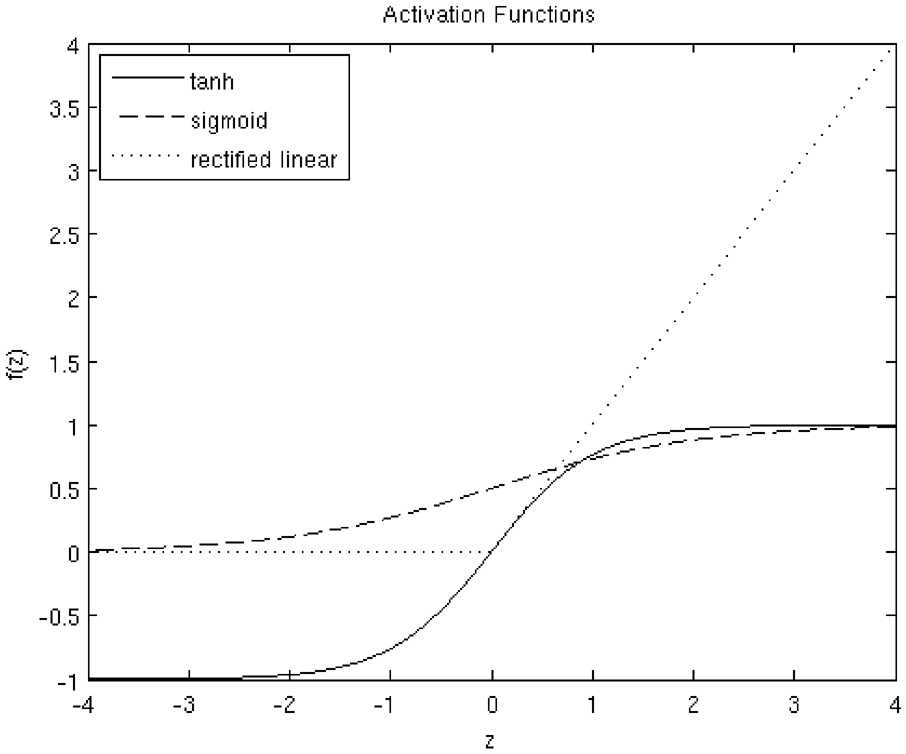


Figure 9: Plot of tanh, Sigmoid and Rectified Linear Functions

2.2.3 Neural Network Model

A neural network is aggregation of many of neurons, and the outputs of some neurons is the inputs of another, here is an example of small neural network:

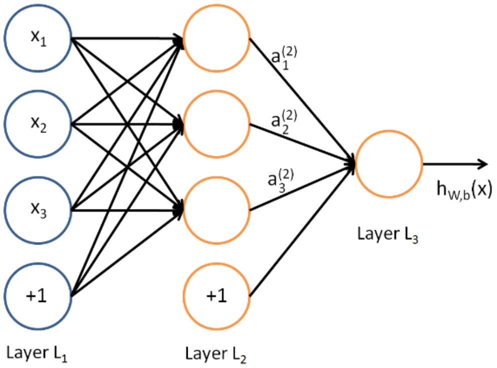


Figure 10: A Small Neural Network

In this figure, each circle is a neuron and the circles labeled “+1” are called “bias unit” which is the intercept term. The leftmost layer of the neural network is called input layer, x_1, x_2, x_3 are the neurons in the input layer in this figure. The rightmost layer is the output layer, and there is only one node in the output layer of the above neural network. The layers between the input layer and the output layer is called hidden layer, the values of hidden layer is not observed in the training set.

Let $W_{ij}^{(l)}$ be the parameter associated with the connection between unit j in layer i, and unit I in layer l + 1. Let $a_i^{(l)}$be the activation of unit i in layer l.

Then the computation of the above neural network can be represented by the following equations.

$$a_1^{(2)} = f\left(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}\right)$$

$$a_2^{(2)} = f\left(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}\right)$$

$$a_3^{(2)} = f\left(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}\right)$$

$$h_{W,b}(x) = a_1^{(3)} = f\left(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)}\right)$$

We can make the computation more quickly if we use vector to represent the values of nodes and the coefficients in each layers, i.e.,

$$f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$$

In this case, the above equations can be write as:

$$z^{(2)} = W^{(1)}x + b^{(1)}$$

$$a^{(2)} = f\left(z^{(2)}\right)$$

$$z^{(3)} = W^{(2)}A^{(2)} + b^{(2)}$$

$$h_{W,b}(x) = a^{(3)} = f\left(z^{(3)}\right)$$

This step is called forward propagation, to refine the neural network model, we need to compare the forward propagation result with the true label of the input and backpropagation the error.

2.2.4 Backpropagation Algorithm

The neural network can be trained using batch gradient descent if we have a training set $\{\left(x^{(1)}, y^{(1)}\right), \dots, (x^{(m)}, y^{(m)})\}$ of m training samples. For each training sample (x, y), we want to minimize the cost function:

$$J(W, b; x, y) = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2$$

For the whole train set, we can define the overall cost function as following according to the above equation:

$$J(w,b)=\left[\frac{1}{m}\sum_{i=1}^{m}j\left(W,b;x^{(i)},y^{(i)}\right)\right]+\frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$

$$=\left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}\left(x^{(i)}-y^{(i)}\right)\right\|^2\right)\right]+\frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2$$

The first term is the definition of error, and the second term is a regularization term to prevent overfitting. The λ controls the relative importance of the first term and second term.

Usually, we use gradient descent to optimize the parameters W and b simultaneously. The equations are as following:

$$W_{ij}^{(l)}=W_{ij}^{(l)}-\alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b)$$

$$b_{i}^{(l)}=b_{i}^{(l)}-\alpha\frac{\partial}{\partial b_{i}^{(l)}}J(W,b)$$

The α in the above equations is the learning rate and it should be choosing properly.

The derivative of the cost function of J(w, b) is as following:

$$\frac{\partial}{\partial W_{ij}^{(l)}}J(w,b)=\left[\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial W_{ij}^{(l)}}J\left(W,b;x^{(i)},y^{(i)}\right)\right]+\lambda W_{ij}^{(l)}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial b_i^{(l)}} J\left(W,b;x^{(i)},y^{(i)}\right)$$

In the last, the training process of a train sample can be written as following steps, and we donate the data in each layer as matrix.

1. Perform the feedforward pass to a train set and compute the activations for each layer. The activations for each layers can be donate as $L_2, L_3 \ldots L_{nl}$.
2. For the last layer (output layer), we have:

$$\delta^{(n_l)} = -\left(y - a^{(n_l)}\right) * f'\left(Z^{(n_l)}\right)$$

3. For each hidden layer:

$$\delta^{(l)} = \left((W^{(l)})^T \delta^{(l+1)}\right) * f'\left(Z^{(l)}\right)$$

4. Apply the gradient descent:

$$\nabla_{W^{(l)}} J(W,b;x,y) = \delta^{(l)}\left(a^{(l)}\right)^T$$

$$\nabla_{b^{(l)}} J(W,b;x,y) = \delta^{(l+1)}$$

2.3 RECURRENT NEURAL NETWORK

2.3.1 Typical Recurrent Neural Network

When human beings read articles, they comprehend the whole article by understanding each word based on the meaning of previous words since a single word doesn’t have any meaning.

In traditional Neural Network, there have no connection between the nodes in the same layer. That means traditional Neural Network cannot preserve the relation between the original input data and this is a major defect of traditional full connected neural network.

Recurrent Neural Network is designed to preserve the relationship between the node and its previous node. There are connections between nodes in the same layer of Recurrent Neural Network and this property makes Recurrent Neural Network becomes applicable to solve the problems like handwriting and natural language processing.

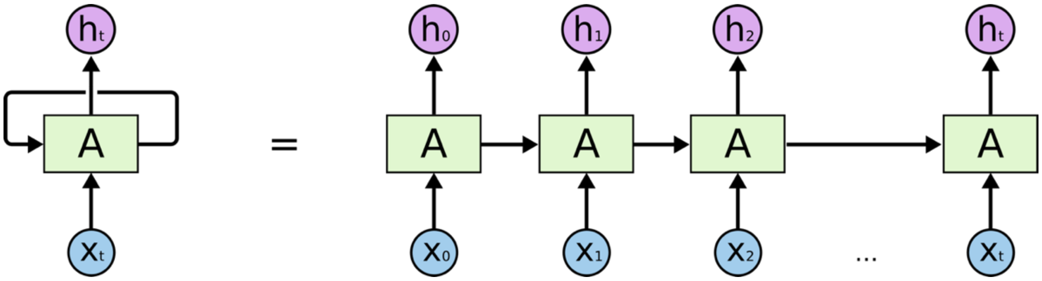


Figure 11: A Single Layer of Recurrent Neural Network

In the above figure, A is a single layer in a Recurrent Neural Network while it has input parameter X and output value h and the value of information can pass from the previous nodes to next nodes in the same layer, that is from X_t to X_{t-1}.

Recurrent Neural Network are widely used in recent years since it has been proved to have an excellent performance in solving a various of problems like speech recognition, language translation and so on.

2.3.2 Long Short-term Memory

Although Recurrent Neural Network have a good performance in some area, but it still has some shortcomings. Sometimes, we only need to loop back to just a few steps, for example, when we want to predict the next word the user may input, we may only look up the previous several words that has been inputted. In the above case, Recurrent Neural Network can have an excellent performance. But if we want to solve the Question Answering problem, looking up only several previous words doesn't work anymore. We need the content from further back. In practice, Recurrent Neural Network doesn't seem to solve this "Long-Term dependencies" problem well.

Long Short-Term Memory is explicitly designed to solve the "Long-Term dependencies" problem and it is kind of Recurrent Neural Network which proposed in 1997.

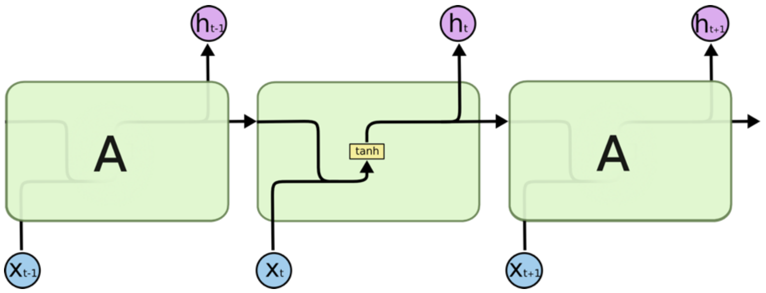


Figure 12: The Structure of a Single Node of Standard Recurrent Neural Network

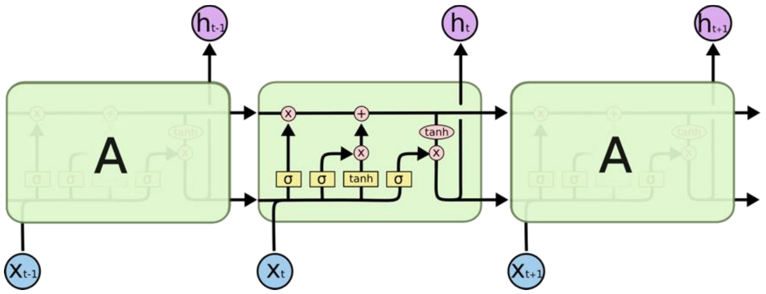


Figure 13: The Structure of a Single Layer of Long Short-Term Memory

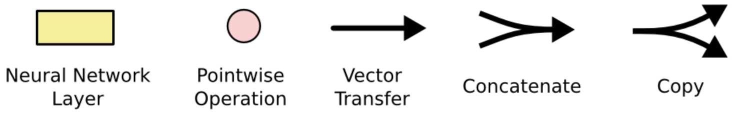


Figure 14: The Meaning of Symbols

Long Short-Term Memory have the same chain structure like Recurrent Neural Network, but they have a different repeat module in a single node. In General, a standard Recurrent Neural network only has single neural network layer while Long Short-Term Memory has four.

2.3.3 Bidirectional Recurrent Neural Network

Bidirectional Recurrent Neural Network is introduced by Schuster and Paliwal [2] in 1997. It is extension of Recurrent Neural Network and it eliminates some limitations of Multilayer Perceptron (MLPs), Time Delay Neural Network (TDNNs) and Standard Recurrent Neural Network (RNNs). The input data of these kinds of neural networks are fixed, and Bidirectional Recurrent Neural Network can be trained without this limitation and it does not require the input data to be fixed since Bidirectional Recurrent Neural Network is training simultaneously in both positive and negative time direction (See the following figure).

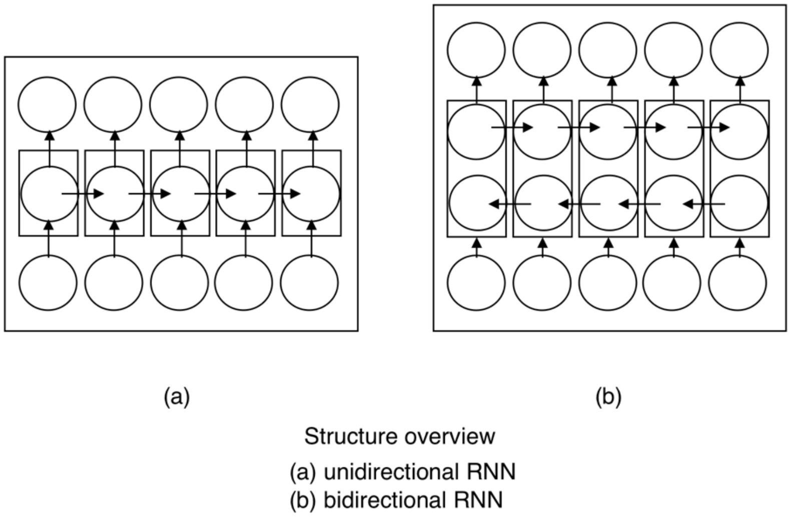


Figure 15: Unidirectional RNN and Bidirectional RNN

Currently, Bidirectional Recurrent Neural Network is widely used in speech recognition, translation, handwritten recognition and protein structure prediction.

2.4 Convolutional Neural Network

In machine learning, convolutional neural network, inspired by animal vision system, is a type of feed-forward neural network that is generally used for image processing because the architecture of convolutional neural network is very suitable for this kind of tasks. For example, the ImageNet (Krizhevsky, Sutskever and Hinton) model has an excellent performance on image classification.

A very important difference between standard artificial neural network and convolutional neural network is that the neurons in the layers of convolutional neural networks is arranged into three dimensions instead of two dimensions in artificial neural network. In addition, differing from the neurons in artificial neural network, the neurons in convolutional neural network only connects to a small region of neurons instead of all neurons in previous layer. (O'Shea and Nash)

2.4.1 Overall architecture

Convolutional neural networks are composed of three kinds of layers, which are fully-connected layers, pooling layers and convolutional layers.

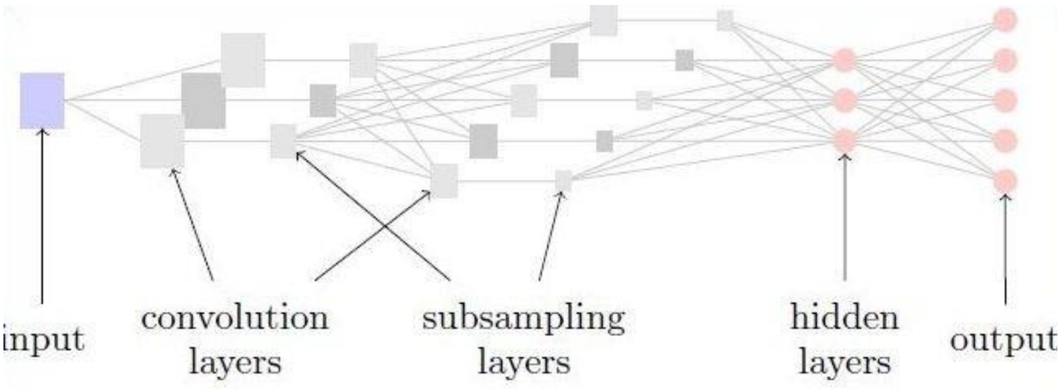


Figure 16: Simple Convolutional Neural Network with Five Layer

As an example, the architecture shown above can be decomposed into several parts.

- Same as traditional artificial neural network, the input layer of Convolutional Neural Network holds the input data.

- Every neuron in the convolutional layers generate its own output by calculating the scalar sum of the products of weights and inputs from the neuron which it connected.

- The pooling layers apply down-sampling to reduce the dimension of input from previous layer. Therefore, the number of neurons, such as, neurons in preceding convolutional layer, and the number of parameters, such as, weights hold by neurons in preceding convolution layer, will be reduced. Therefore, the total model complexity is reduced due to pooling layers.

- The full-layers have the same functionality as in artificial neural network which is calculating the class scores from the activations, which will be used for classification.

According to this small sample, convolutional neural networks can transform the input using several technologies, like convolution, down-sampling layer by layer to produce class score for classification or other purposes.

2.4.2 Convolution Layer

Convolution layer is the most important layer of convolutional neural network and the parameter in this kind of layer is the learnable kernels.

These kernels are commonly very small in terms of dimension comparing with input. In convolutional layer, when the input is ready, the input convolves with the small kernel to create a 2D activation map.

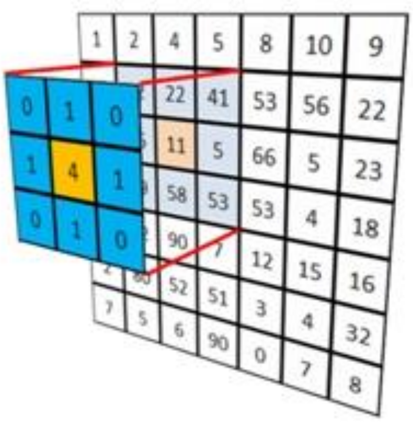


Figure 17: Using Gaussian Kernel to filter an image

As we scan the input, the convolution result will be calculated using the kernels for each portion of input. Kernels will learn whether a specific feature has been sawn at a certain position of the input from the network and these processes are usually called activations. Each kernel has its own activation map based on the input, and these activation maps will be stack together to form a 3D-array as output of convolutional layers.

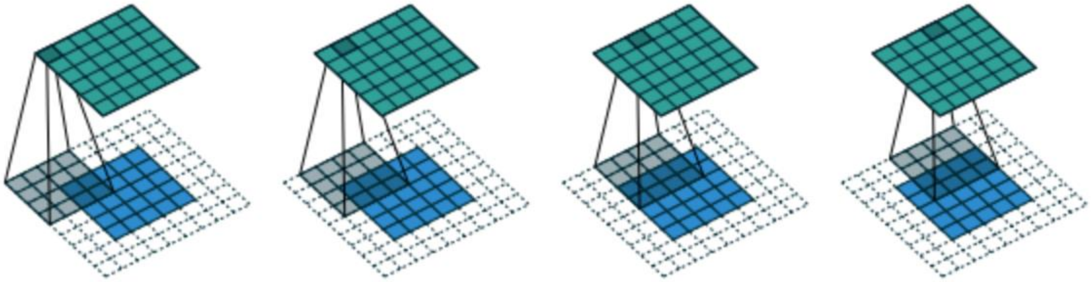


Figure 18: Convolution Process of An Image

As we all known, artificial neural networks which are been trained for image processing are always too large due to fully-connected manner of standard artificial neural network, therefore this kind of artificial neural networks cannot be efficient enough for training. To solve this problem, neurons in convolutional layers only connect to a small region of neurons in previous layer. The size of this region is called receptive field size.

For instance, if the input image is RBG image with size 100 * 100 * 3 and receptive field size is set to 5 * 5, each neuron in convolution layers should only maintain 75 weights comparing with 30000 weights for neurons in artificial neural networks.

Convolution layers also can optimize their output to reduce the model complexity significantly. There are three basic hyper-parameters for optimization which are the depth, the strike, and the size of zero-padding.

Depth is the third dimension of the output and it controls the number of neurons connect to the same region of input. Reducing the depth can significantly reduce the neurons in convolutional layers.

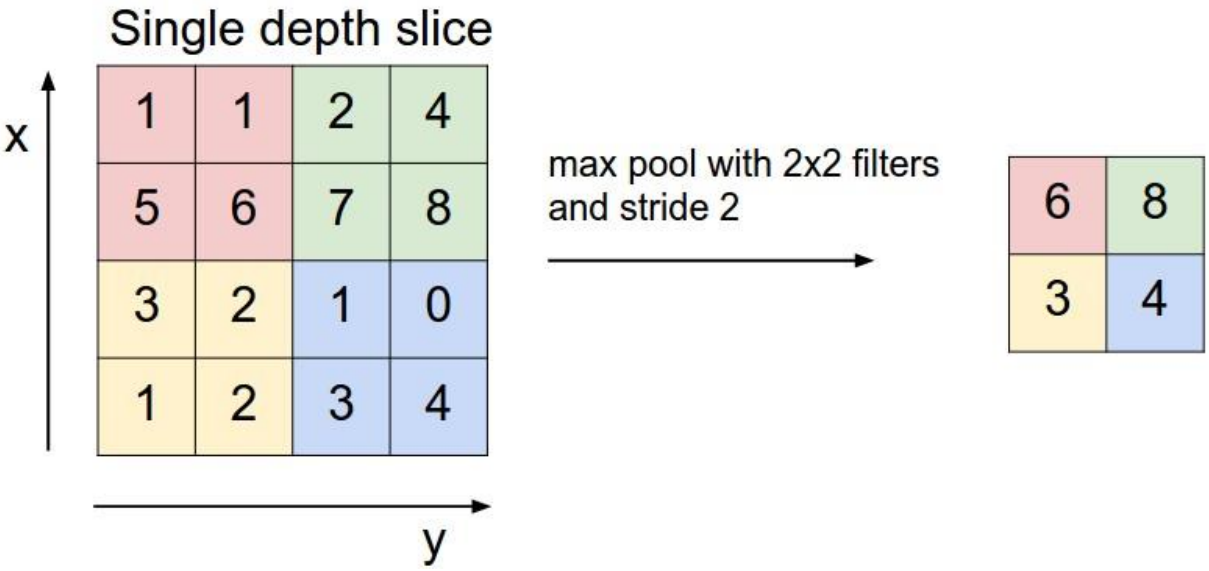
Stride controls the height and weight of output. If the stride is set to 1, the receptive fields will be heavily overlapped and the output will be very large. So increasing the stride can reduce the overlapping of receptive fields and the size of output.

Zero-padding is the process that padding on the border of the input. Therefore, the number of zeros that zero-padding applies can further control the size of output.

It is important to understand these three hyper-parameters and we can calculate the size of output based on the formula $(N - F + 2 * Z) / (S + 1)$ where N is the size of input, F is receptive field size, Z is the size of zeros padding on the border and S is the stride. If the result is not an integer, the stride is set incorrectly because the neurons cannot fit in this configuration on the input.

2.4.3 Pooling layers

Pool layers is used to reduce the size of data and therefore reduce the parameters (weights) of each neuron and model complexity. The pool layers take activation maps as input and reduce their size by using Max function. Since the pool layers use Max function, this kind of pool layers are usually referred to max-pooling layers. For instance, if a kernel with size 2 * 2 is applied with stride 2 to the input, the height and width will all be reduced to the half of their counterpart of input. Therefore, the size of output is reduced to only 25% of the size of input with the depth unchanged.



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Figure 19: Pooling Layer

As the pooling layers remove information from input, there are only two ways to implement max-pooling. The first one is already described in example above. The kernel is set to 2 * 2 and stride is set to 2. Another one is called overlapping-pooling, where the kernel is set to 3 * 3 and stride is set to 2. In general, the size of kernel cannot exceed 3, because it will remove too much information.

2.4.4 Fully-connected Layers

each neuron in fully-connected layer connects to all the neurons in previous layer and preceding layer. The functionality of fully-connected layers in convolutional neural networks is the same as theirs in artificial neural network.

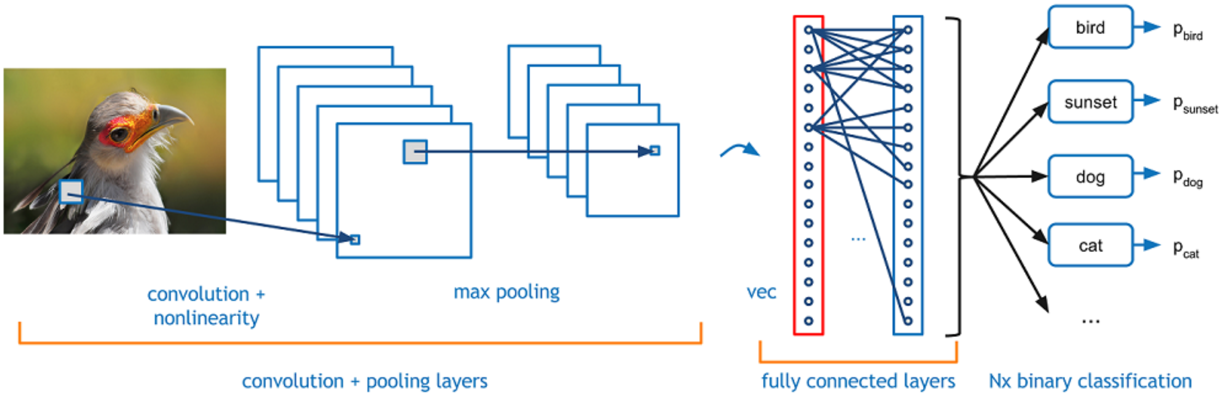


Figure 20: Fully Connected Layers In CNN

2.4.5 SoftMax Layer (Output Layer)

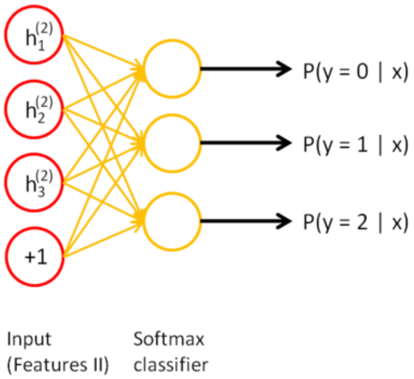


Figure 21: Softmax Layers

SoftMax function is an extension of logistic function. The logistic regression is to solve the binary problem which are labeled {0, 1}.

The hypothesis function of logistic regression is:

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)}$$

The cost function of logistic regression is:

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)} \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

The train process is to minimize the cost function $J(\theta)$. After we obtain the optimal parameter, we can use the logistic function to predict the result of input data. But the logistic function can only be used in binary problem, when the labels of data is larger than two, we need to use k logistic classifiers and it is fussy. In this case, we can use the SoftMax function to solve the problem.

The hypothesis function of SoftMax function is:

$$h_\theta\left(x^{(i)}\right) = \begin{bmatrix} p\left(y^{(i)} = 1 \middle| x^{(i)}; \theta\right) \\ p\left(y^{(i)} = 2 \middle| x^{(i)}; \theta\right) \\ \vdots \\ p\left(y^{(i)} = k \middle| x^{(i)}; \theta\right) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} \theta_1^T x^{(i)} \\ \theta_2^T x^{(i)} \\ \vdots \\ \theta_k^T x^{(i)} \end{bmatrix}$$

The cost function of logistic regression is:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y^{(i)} = j\} log \frac{\theta_j^T x^{(i)}}{\sum_{l=1}^{k} \theta_l^T x^{(i)}}\right]$$

The train process of the SoftMax function is like to the logistic function, and finally we can use the function to predict the result:

$$P(y = j|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}}$$

2.5 Neural Reasoner

2.5.1 Overview

Neural reasoner is proposed by Baoling Peng et al. [3] in 2015 and it is a framework that used in natural language sentence processing. The high level structure of neural reasoner is as following:

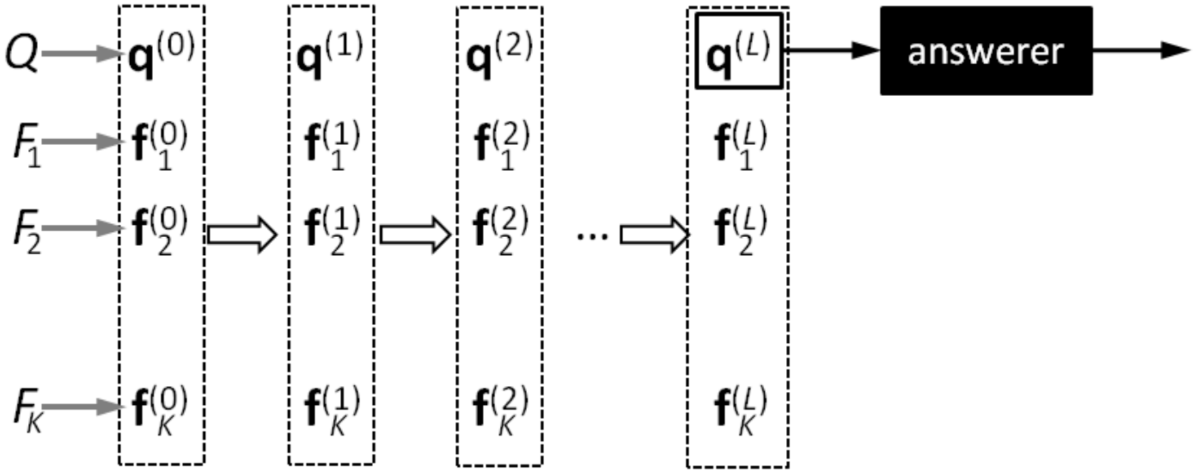


Figure 22: Diagram of Neural Reasoner [3]

As we can see, the neural reasoner consists of one decoding layer and several reasoning layers and the function of encoding layer is to convert the natural language sentences (including questions and facts) into matrices and the reasoning layer the recursively update the representations of the questions and facts.

$$Q \xrightarrow{encode} q^{(0)}, F_k \xrightarrow{encode} f_k^{(0)}, k = 1,2, \dots, K.$$

$$\left\{q^{(l)}, f_1^{(l)} \dots, f_K^{(l)}\right\} \xrightarrow{reason} \left\{q^{(l+1)}, f_1^{(l+1)} \dots, f_K^{(l+1)}\right\}$$

The process of encoding and reasoning is to simulate the process of human logic thinking. As there are several facts and one question about these facts, human can obtain the answer of the question by analyze the logics between the facts and the questions. In this case, the neural reasoning can handle a series of facts and obtain the final answer by iteratively reasoning between the facts questions.

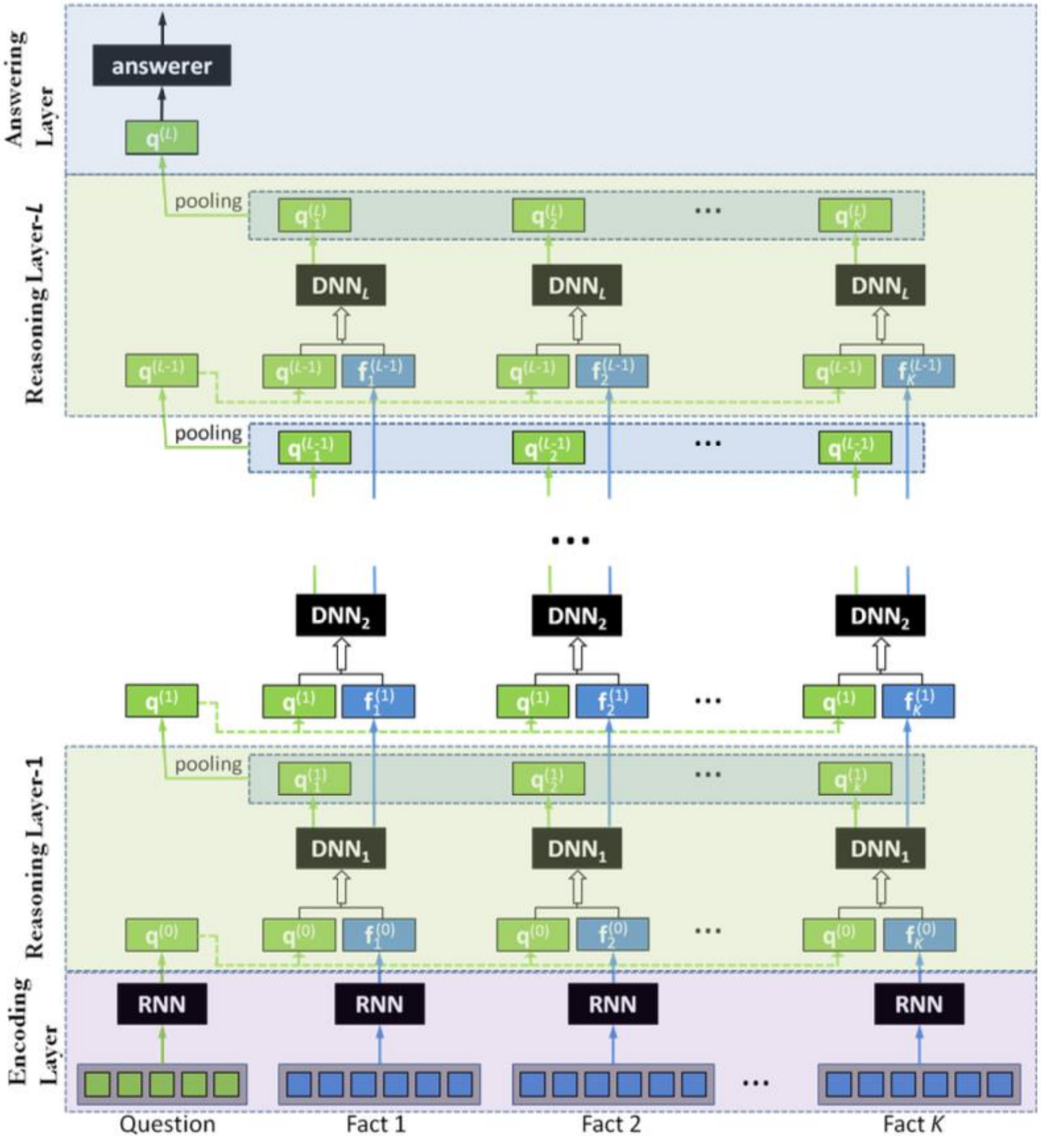


Figure 23: Implementation of Neural Reasoner [3]

2.5.2 Encoding Layer

The purpose of encoding layer is to find the representation of natural languages and there are different models to achieve it, e.g. Convolutional Neural

Network, Recurrent Neural Network. We choose LSTM (Long Short-Term Memory) to encode the questions in our project.

2.5.3 Reasoning Layer

The reasoning layer consists of some question-fact iteration, polling. We will introduce these in detail later.

i) Question-Fact Iteration

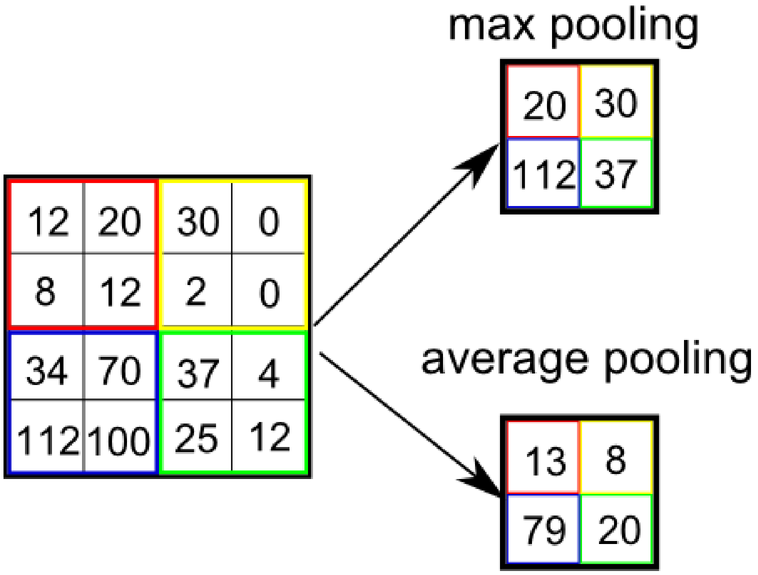
In the reasoning layer l:

$$\left[q_k^{(l)}, f_k^{(l)}\right] \stackrel{\text{def}}{=} gDNN_l\left(\left[\left(q^{(l-1)}\right)^T, {f_k^{(l-1)}}^T\right]^T ; \Theta_l\right)$$

ii) Pooling

We have three kinds of pooling:

- Average / Max Pooling:



12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

20	30
112	37

13	8
79	20

Figure 24: Max Pooling & Average Pooling

- Gating:

We can use a gating network to determine the weight of each input data based on the facts and questions in the current layer.

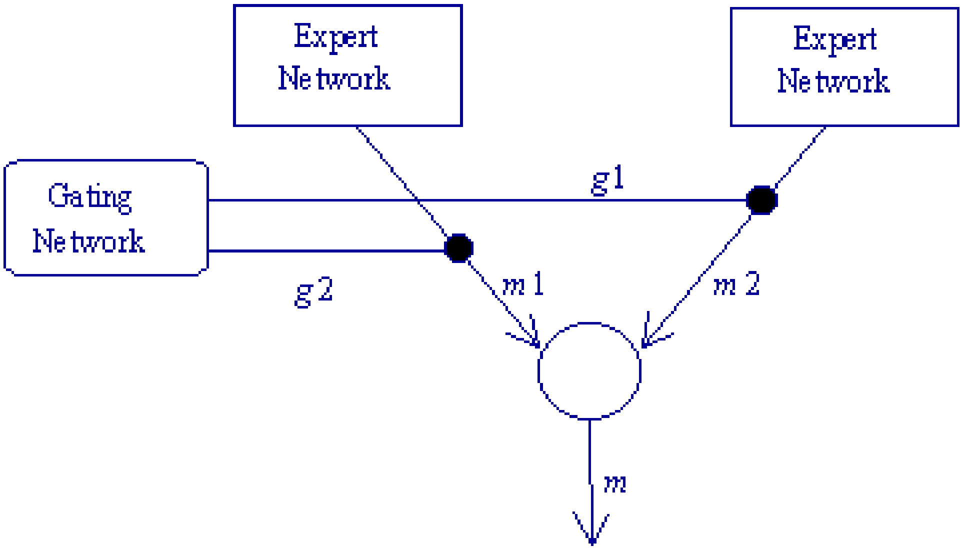


Figure 25: Gating Network

- Model-based:

We can use a Convolutional Neural Network or Recurrent Neural Network to handle the facts and questions.

Chapter 3: METHODOLOGY

3.1 NATURAL LANGUAGE PROCESSING

3.1.1 Traditional Method of Processing Natural Language

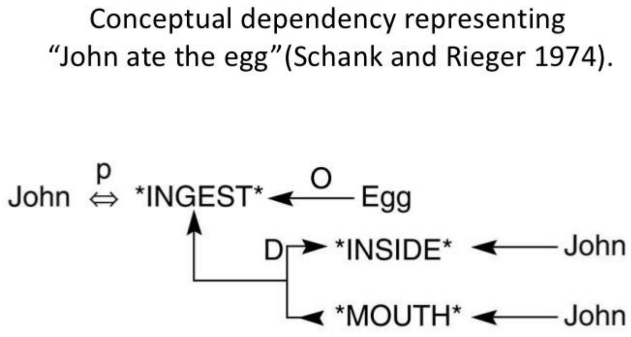
The ability for computers to communicating with human in natural language is what human are keeping on chasing. To let computers to be able to communicate with human in natural language, computers need to be able to understand the meaning of natural language and able to express the ideas in natural language. The first one is called Natural Language Understanding and the second one is called Natural Language Generating. Therefore, Natural Language Processing basically contains two parts which are Natural Language Understanding and Natural Language Generation. Historically, scientists did more research on Natural Language Understand and less on Natural Language Generation. However, situation is changed now.

Achieving the communication between human and computers is natural language is very difficult because of:

- Difficulty to determine the boundary of words. In speaking, there is not boundary between word and word. Therefore, computer need find out the best combination of words based on the context.
- Ambiguity of words. One word may have different meanings. The computer need to find out the most suitable meaning based on the context.
- Ambiguity of grammar. Sentences may have different parsing trees because of ambiguous grammar. Therefore, computer need to determine which parse tree is the best based on the context.

When neural network is not introduced in this field, conceptual dependency theory is widely used in Natural Language Processing.

Conceptual dependency theory is first introduced in 1969. In this model, two sentences with the same meaning will have the same representation even though the words in these two sentences are not the same. Four basic representational tokens are used in conceptual dependency theory model which are real word objects with some attributes, real world actions with some attributes, times, locations.



Conceptual dependency representing
“John ate the egg”(Schank and Rieger 1974).

Figure 26: Example of Conceptual Dependency Theory

After machine learning is introduced, A revolution occurred in this field. At the beginning, scientist uses machine learning to produce something like decision thing to represent the natural language which is like previous traditional approach. Later, scientists began to focus on develop representation algorithm, like word embedding, on statistical model.

3.1.2 Work Embedding

Word embedding is a set of techniques that maps words or phrases to vectors of real number. We will use word2vec model developed by Google to process the natural language text of train questions and answers.

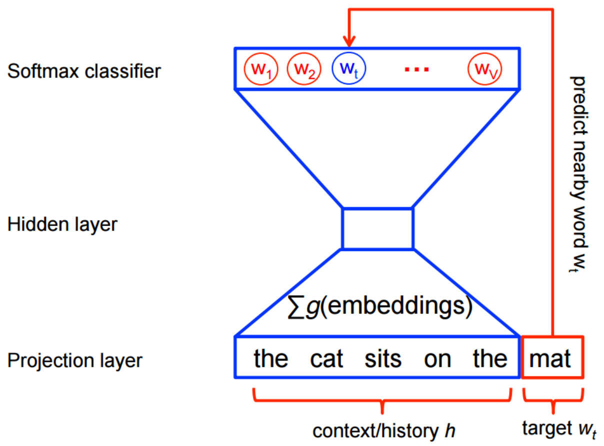


Figure 27: Model of Word Embedding

Word2vec learning how words are used in input text by using a shallow neural network. Word2vec output a matrix and each column of this matrix represents a word in input text and provides a numerical description on how the word is used in input text. If the input train text is large enough, the vector that represent two words with similar meaning will have small vector distance. For instance, the word “woman” and word “lady” may have a small vector distance. Most application of word2vec judge the similarity of two words using cosine similarity. Empirically, Word2vec has very nice performance when the input text is very large, consistent and without ambiguity.

To map the words or phrases to numerical vector, A technique called “skip-gram with negative sample” is used by Word2vec. This method can be roughly divided into 5 steps.

First, take a word in the input text as target and a few words that are close to this target as context.

Second, represent each word in the input text by a random numerical vector. After this step, we can get the numerical vector of the target and context.

Third, put the vector of target and context together and shorten the distance of these vectors.

Fourth, pick up the words randomly from the input text outside the context, and enlarge the distance between the vectors of these words and the vector of the target. By doing this, we can make our target be further away from the words which is rarely used in context.

As keep on applying this process on the targets and their contexts, the vectors of words which are always used together will be pulling closer and closer which the vectors of words which are rarely used together will be pushing further and further.

3.1.3 Bilingual Word Embedding

###

3.2 TEXT-BASED QUESTION ANSWERING

3.2.1 Word Embedding

First, we perform the word embedding to convert word strings into vectors. The method to perform the word embedding is call Skip-gram (Mikolov, Sutskever and Chen). First, we will traverse all the words in the training set and get a unique word dictionary vector. Second, we can replace words in all the sentence in the train set with the index of the word in the dictionary vector. After converting sentence in the train set into vectors, we can use Long Short-Term Memory (Understanding LSTM Networks) (LSTM) to capture more information about the sentence.

3.2.2 End-to-End Memory Network

After converting the words to vectors, end-to-end memory network is used to solve question answering problem, this network is a kind of recurrent neural network where the recurrence reads from a large data before it produces the result.

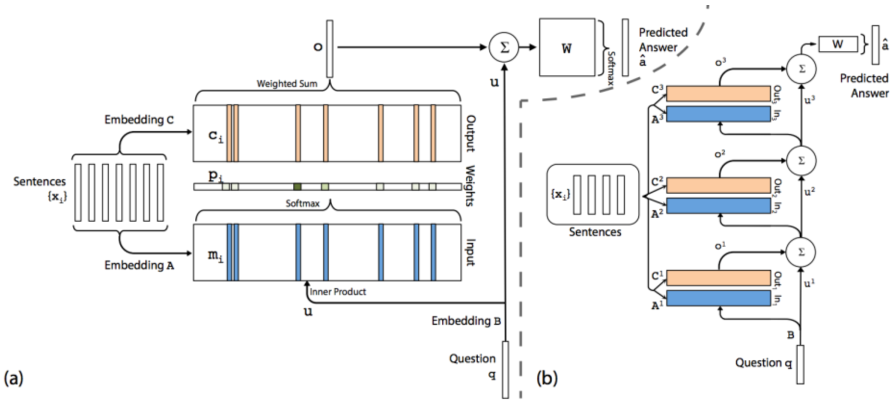


Figure 28: (a) Single Layer Version (b) Three Layer Version

End-To-End Memory Network can divide into the following parts:

- Input representation:

In this stage, the model maintains several embedding matrices with the same dimension. We use a certain embedding matrix to convert the input vectors to memory vectors and convert the query to an internal state. After that, we compute the match, which is a possibility, of the memory vectors and the internal state by using SoftMax function on the dot product of these vectors.

- Output representation:

In this stage, we use the output vectors, which are calculated for each input vector by using another embedding matrix, to compute to the response vector of the memory. The response vector is simply the sum of the products of each output vector and its corresponding possibility which is computed in last stage.

- Find out the prediction:

In the single layer case, the prediction of the network is to sum all response vector and the internal state and then pass this sum through the weight matrix. Finally, we can use Softmax function to get the result.

3.3 IMAGE PROCESSING

In image science, image processing represents a various of technologies that used to analysis and process the images such that the images meet the visual or mentality request or some other technical requirements. It is an application of signal processing in the domain of images. Nowadays, most of images are stored in digital format, thus image processing most refers to the digital image processing. Besides, some methods based on the optical theories still holds an importance position.

Image processing is sub-class of signal processing and has a deep connection with computer science and artificial intelligence. The traditional ways in process the one-dimension signals can be used in image processing such as quantization and noise elimination. But the images are still two-dimension signals which have their own special methods.

3.3.1 Traditional Method

When the neural network is not widely used, scientists use feature detection to gather information from images, deciding whether every single pixel belongs to an image feature. Therefore, the result of feature detection is usually a collection of sets of pixels and the sets contains isolated points, continuous lines or continuous regions.

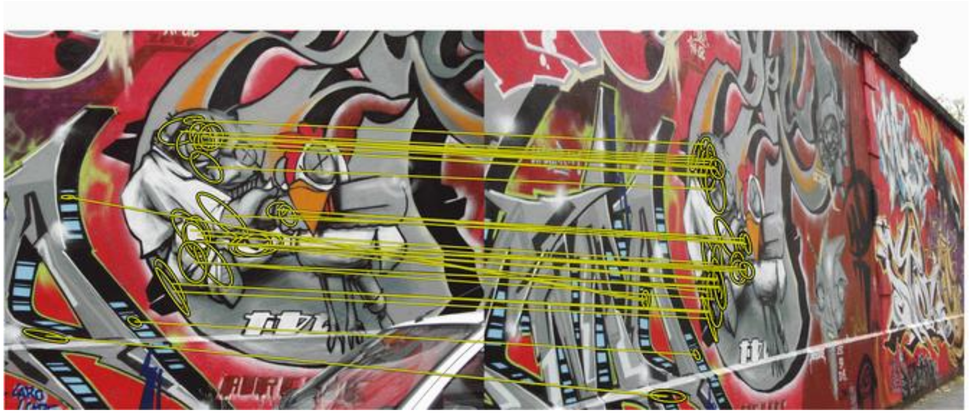


Figure 29: Feature Detection

Nowadays, there is not universal and accurate definition of features. Features are usually determined by the problems or the applications. Features can be the interest parts in a digital image and they are the starting point of many computational images analysis algorithm. Whether an algorithm will be successful or not depends on which features this algorithm uses. Therefore, features must be repeatable which means the features extracted on different images in same scenario should always be the same.

Feature detection is a very fundamental algorithm in image processing, which means it is always the first processing on the images. Feature detection checks every single pixel and determine whether this pixel represents a feature. If feature detection is a part of a larger algorithm, then feature detection usually only check the feature regions of images. As the preprocessing of feature detection, the input images are usually smoothed by using Gaussian kernel.

Many different image processing algorithms use feature detection as its starting point. Therefore, many different detection algorithms are developed and the types of features, the computational complexity and repeatability are varied from each other.

i) Edge Detection

Edge detection is a fundamental problem in computer vision and image processing and aims to find the pixel at which image brightness changes a lot in the images. The sharp change in pixels usually refers to important change of feature. These changes include discontinuities in depth, discontinuities in surface orientation, changes in material properties and variations in scene illumination.

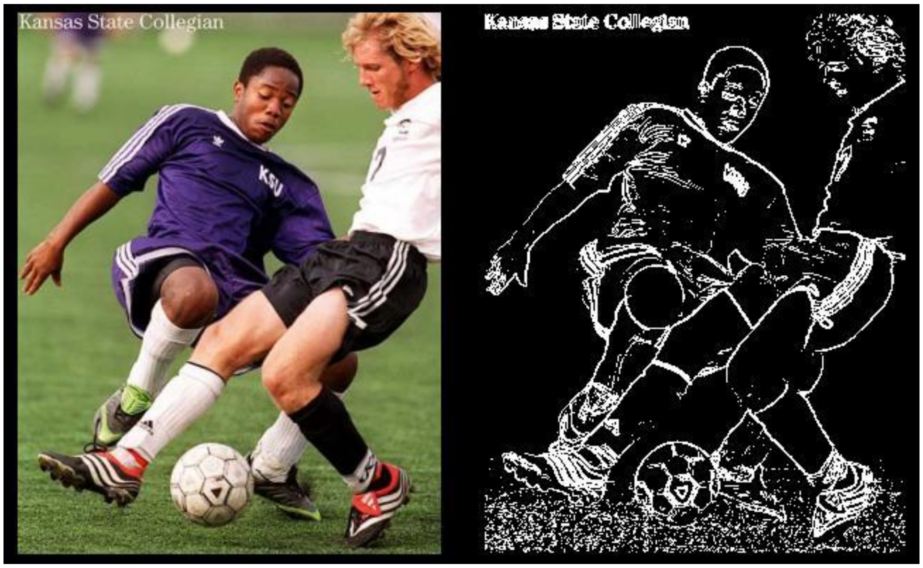


Figure 30: Sample of Edge Detection

After applying feature detection, the size of data is significantly reduced, some information which is considered as not important is removed and the important structural features of image are remained.

There are many different methods to perform edge detection, most of them can be categorize into two part, search-based and zero-based. The search-based method detects edges by computing the gradient of images and find out the local directional maximum of gradient. The zero-crossing based methods search for the zero-crossing of the Laplacian of images.

ii) The Shortcomings of Traditional Method

Although some traditional method in image processing can achieve a better performance, there are still some shortcomings in these ways. As the rapidly development of the computation speed of computers, deep learning algorithms are wildly used in image processing, such as CNN (Convolutional Neural Network). The results of experiments show that deep learning can perform better than the traditional methods in image processing.

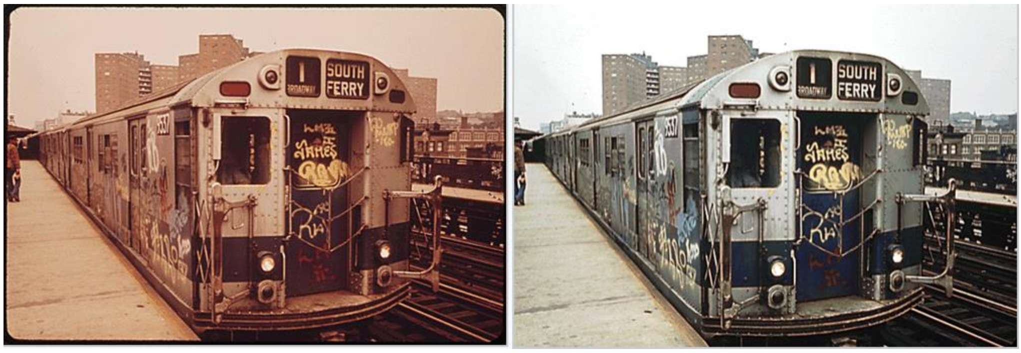


Figure 31: Enhance of Image Using Optical Theory (htt5)

In our project, we should extract the features of an image and feed the features into the neural network. According to the previous experiments, CNN (Convolutional Neural Network) has an excellent performance in extracting the features of images. And we will introduce the Convolutional Neural Network in the next chapter.

3.4 Extend Text QA to Image QA

3.4.1 Extract Image Features.

In our application, we used the VGG model to extract the feature from training set.

VGG model contains a stack of convolutional layers with kernel size of 3 * 3. The convolutional stride is 1 and the zero-padding size is also 1. For pooling layers, VGG model contains five max-pooling layers with 2 * 2 kernel and stride equal to 2. Following the convolutional layers, there are three Fully-connected layers. The first and second fully-connected layer contains 4096 neurons each while the third layer has only 1000 neurons because this layer is used to perform ILSVRC classification. Finally, we add a soft-max layer as final layer.

Instead of training our own VGG model, we use the VGG provided Keras. After deploy the model in our application, we extract the feature from the images in MSCOCO dataset. In our program for extracting feature, we set parameter batch size to 10, which means the program will process 10 images in one turn. In each turn, we extract feature of these ten images, which are vectors, using the VGG model. After the features are produced by the VGG model, we store the features into a file as the input of LSTM.

3.4.2 Feed the Image Features to LSTM

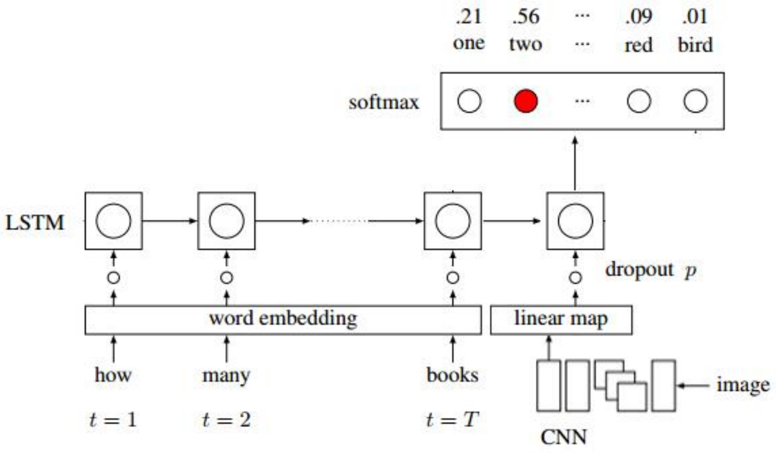


Figure 32: Whole Model of Visual Question Answering

After using Convolutional Neural Network to extract the features from the image train set, we feed these features with the text vectors to the Long Short-Term Memory to train the model.

3.5 NEURAL REASONER

The neural reasoner is widely used in question answering problems, and we can extend it to visual question answering. The image can be regards as the facts in the question answering problems.

In this case, we use object localization algorithms to detect the significant object in an image and find the box proposals of these objects. Then update the matrices of question and images iteratively using reasoning.

3.6 FRENCH LANGUAGE SUPPORT

In this term, since we want to achieve the VQA in French, we introduce a method called Bilingual word embedding. This method is developed by Stanford.

First, we introduce monolingual models and bilingual model. monolingual models are the models like word2vec model that we used in last semester and can convert a single word of certain language into a vector. Bilingual models are the models that contains two monolingual models that interprets two different languages and these two monolingual models will embed two words with same meaning in two different languages into two vectors which are closer than other word-embedding vectors. For example, the distance between the word-embedding vector of cat and word-embedding vector of chat is smaller than distances between word-embedding vector of cat and word-embedding vectors of French words other than chat.

When building up bilingual model, in terms of monolingual components, any famous word-embedding model can be a proper candidate. Therefore, we choose the word2vec model used in the last semester. To coordinate two monolingual models that interpret two languages and interpret words into vectors that bilingual model needs, we adapt the skipgram model with negative sampling which described in previous section to the bilingual context. Such a consistent choice of architectures results in a natural and effective way of building bilingual models from existing monolingual models. The basic idea of this method is that if we have sentence S1 w-A1 wA2 wA3 in language L1, sentence S2 wB1 wB2 wB3 in language L2 and we know that the meaning of S1 and S2 are the same with the meaning of wA2 and wB2 are the same, we can conclude that there is high probability that the meaning of wA1 and wB1 are the same. Therefore, according to the conclusion above, the skipgram model will shorten the distance between vector representations of wA1 and wB1. By doing this iteratively using a very large dataset, we can build up a bilingual model we need.

3.7 VIDEO SUPPORT

3.7.1 Surveys

To make our model support the video input, we surveyed plenty of algorithms and find the common approach to achieve this goal, that is, we use some algorithms to extract the video information and then combine this information with question and train the model.

There are several approaches [7] to extract the information of a video or represent a video, e.g., GRU (Gated Recurrent Unit, a kind of Recurrent Neural Network), additional Bidirectional Long Short-Term Memory, Extended Soft-

Attention model and Extended Sequence-to-sequence (E-SS) model. The structure of these model is as following:

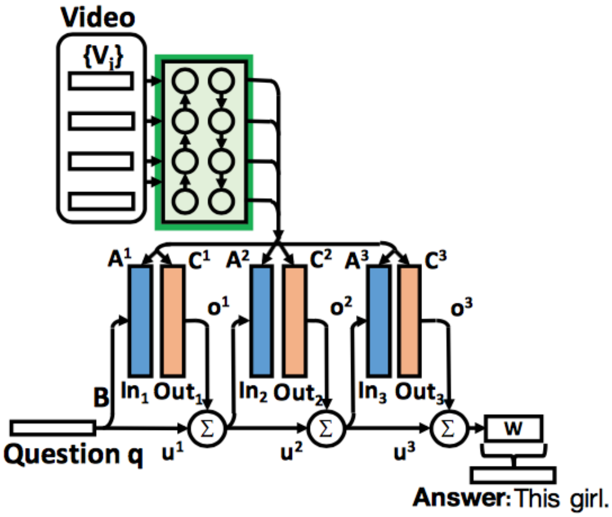


Figure 33: Extended End-to-End Memory Network

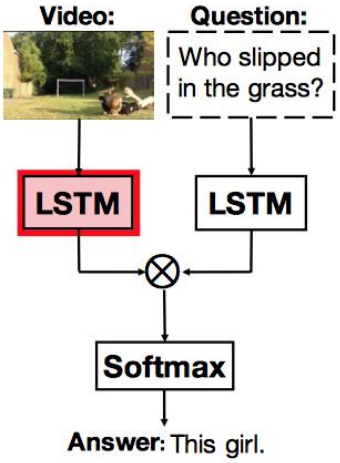


Figure 34: Extended Visual Question Answering Model

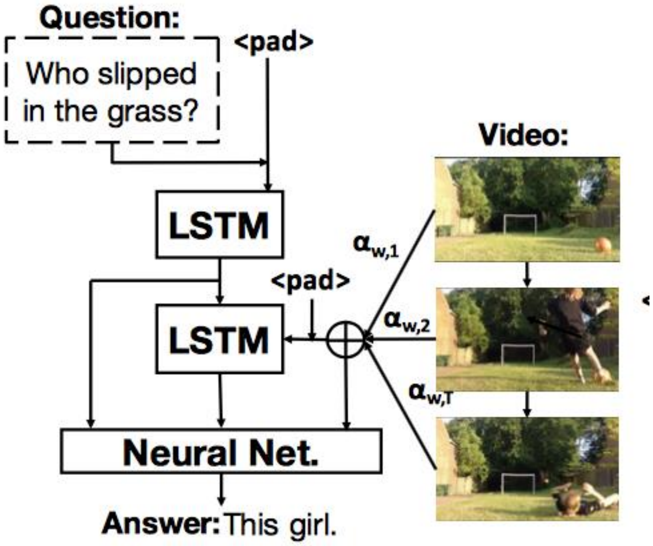


Figure 35: Extended Soft-Attention Model

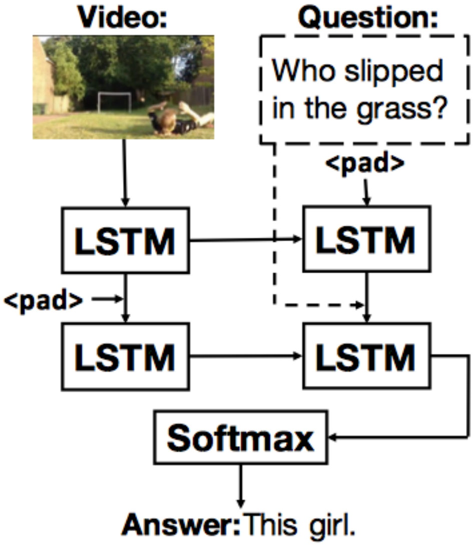


Figure 36: Extended Sequence-to-Sequence Model

3.7.2 Method

We have analyzed these models above and we find some limitation in our project. First, if we want to train a new model based on the above approach, we need the video and question-answer training set to train the model, but we didn't have enough time to collect the train set. Second, if we want to use the current model to process the videos, we cannot input the video or the representation of the video directly, since the meaning of the representation of video is different with the meaning of representation of image.

In this situation, we come up with the following approach to achieve the video question answering. The structure is as following:

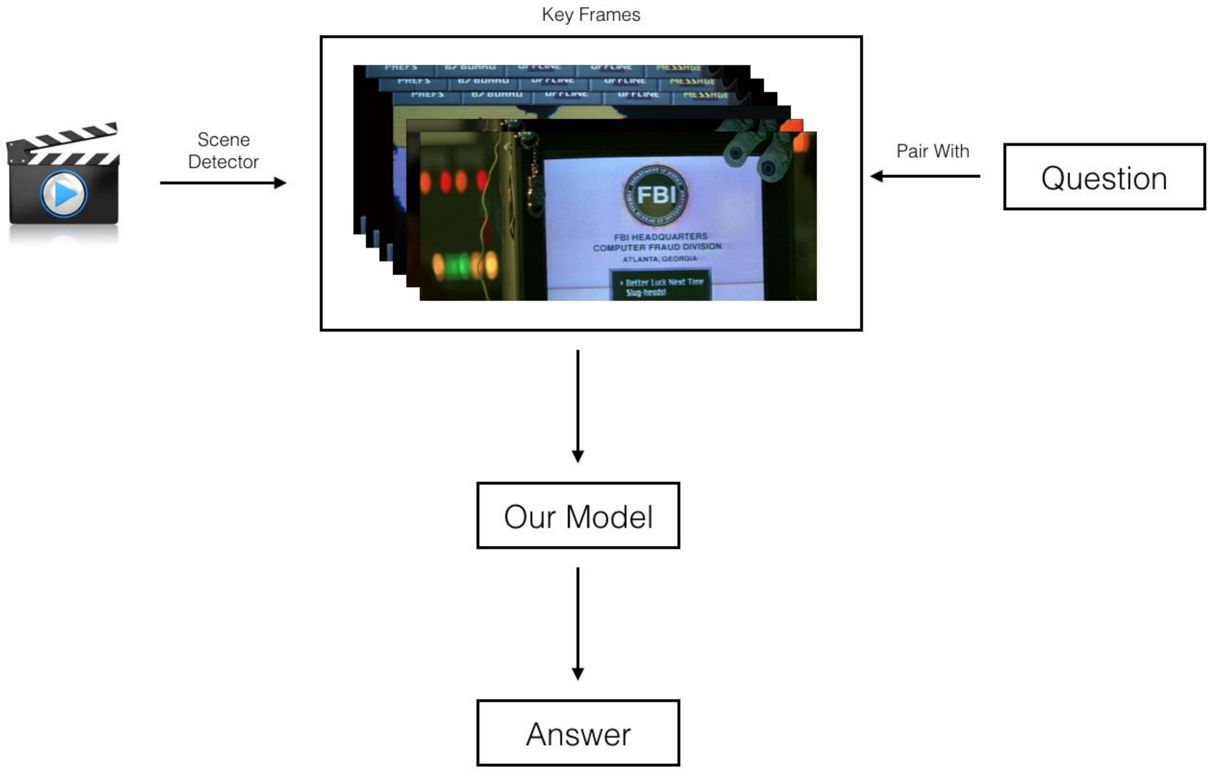


Figure 37: Structure of Our Video Question Answering

We first use the scene detector to cut out the key scenes in a video and using the model to predict an answer for each frame that pair with the question. In the last, we will output the answer that has the highest probability.

Chapter 4: WORK OF FIRST SEMESTER

4.1 MODELS

In general, an image can be described as a row * column * 3-dimension matrix which in a high dimension and text can be represented as a vector which in a low dimension. What we need to is to reduce the dimension of images or increase the dimension of text so that these two items can be combined and feed to neural network.

In the first, we consult to the structure of Question-Answering model based on text, then extent the text-base to image-base structure.

4.1.1 Text-Based QA Model

i) Word Embedding

First, we perform the word embedding to convert word strings into vectors. The method to perform the word embedding is call Skip-gram. First, we will traverse all the words in the training set and get a unique word dictionary vector. Second, we can replace words in all the sentence in the train set with the index of the word in the dictionary vector. After converting sentence in the train set into vectors, we can use Long Short-Term Memory (LSTM) to capture more information about the sentence.

ii) End-to-End Memory Network

After converting the words to vectors, end-to-end memory network is used to solve question answering problem, this network is a kind of recurrent neural network where the recurrence reads from a large data before it produces the result.

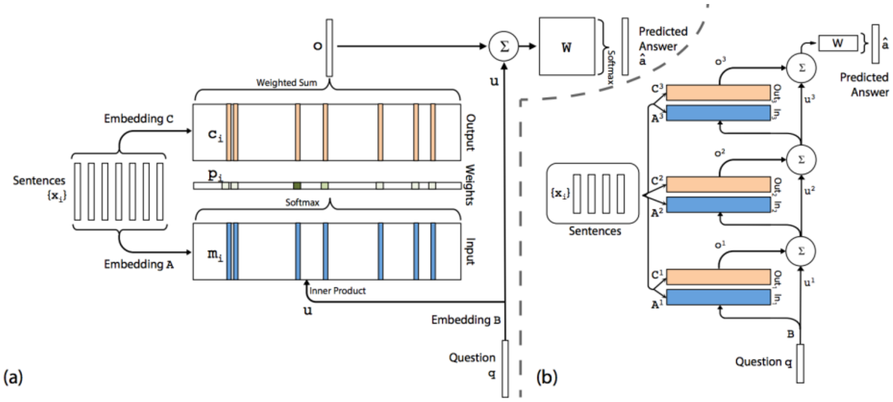


Figure 38: (a) Single Layer Version (b) Three Layer Version

End-To-End Memory Network can divide into the following parts:

- Input representation:

In this stage, the model maintains several embedding matrices with the same dimension. We use a certain embedding matrix to convert the input vectors to memory vectors and convert the query to an internal state. After that, we compute the match, which is a possibility, of the memory vectors and the internal state by using SoftMax function on the dot product of these vectors.

- Output representation:

In this stage, we use the output vectors, which are calculated for each input vector by using another embedding matrix, to compute to the response vector of the memory. The response vector is simply the sum of the products of each output vector and its corresponding possibility which is computed in last stage.

- Find out the prediction:

In the single layer case, the prediction of the network is to sum all response vector and the internal state and then pass this sum through the weight matrix. Finally, we can use Softmax function to get the result.

4.1.2 From Text to Image

i) Extract Image Features.

In our application, we used the VGG model to extract the feature from training set.

VGG model contains a stack of convolutional layers with kernel size of 3 * 3. The convolutional stride is 1 and the zero-padding size is also 1. For pooling layers, VGG model contains five max-pooling layers with 2 * 2 kernel and stride equal to 2. Following the convolutional layers, there are three Fully-connected layers. The first and second fully-connected layer contains 4096 neurons each while the third layer has only 1000 neurons because this layer is used to perform ILSVRC classification. Finally, we add a soft-max layer as final layer.

Instead of training our own VGG model, we use the VGG provided Keras (VGG16 model for Keras). After deploy the model in our application, we extract the feature from the images in MSCOCO dataset. In our program for extracting feature, we set parameter batch size to 10, which means the program will process 10 images in one turn. In each turn, we extract feature of these ten images, which are vectors, using the VGG model. After the features are produced by the VGG model, we store the features into a file as the input of LSTM.

ii) Feed the Image Features to LSTM

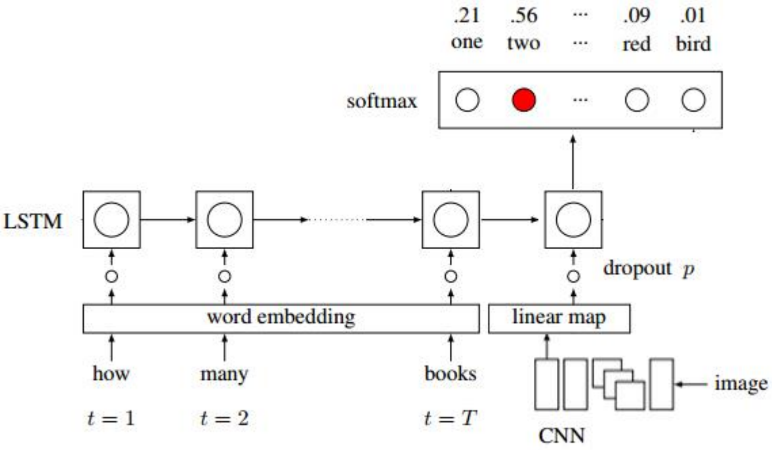


Figure 39: Whole Model of Visual Question Answering

After using Convolutional Neural Network to extract the features from the image train set, we feed these features with the text vectors to the Long Short-Term Memory to train the model.

4.2 HYPOTHETICAL MODEL

To improve the accuracy of number-relative question, we decide to use some ways to process the image such that we can count all objects in an image and feed the result into the neural network as image features.

4.2.1 Object Counting Model

To improve the performance of our model for visual question answering, we build a sub-model for objection detection which may have benefit to our model.

This object detection is referred to the paper, which describe a kind of neural network called Fast Region-based Convolutional Network. Comparing with other neural network, this neural network has faster training speed and more accurate detection.

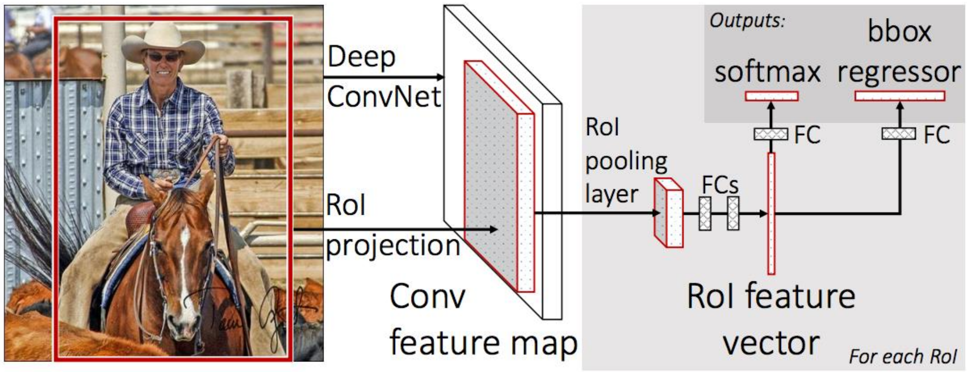


Figure 40: Fast R-CNN Architecture

Fast R-CNN uses the whole image and its object proposals as input. First, Fast R-CNN passes the input through several convolutional layers and max-pooling layer to generate the convolutional feature map of this image. After that, the convolutional feature map is passed to a region of interest (RoI) layer to generate a fixed-length feature vector for each proposal. Finally, these feature vectors are passed through two separate fully-connected layer. For the first fully-connect layer which perform SoftMax function, this layer produces a numerical number for each proposal which indicate the possibility that this object is in this image. For the second fully-connected layer, this layer output four numbers for each proposal which indicate the location of the object in this image.

In our model, as the accuracy of answering question like predicting number of objects is very low, I think Fast R-CNN is very suitable to our model. As mentioned earlies, the output of Fast R-CNN can be considered as a set of regions of interest (rectangles) indicating objects and the corresponding label for each object in these regions of interest. Therefore, by reviewing the output of Fast R-CNN, we can find out the numbers of different objects in input image.

For example, the input image is shown below.

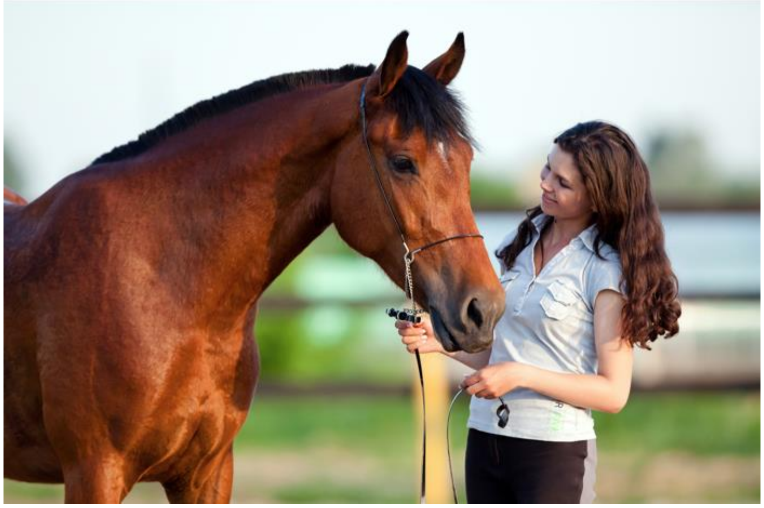


Figure 41: Input Image for Fast R-CNN

In this image, there are a female and a horse. After the processing of Fast R-CNN, we can toughly consider the output as image below.

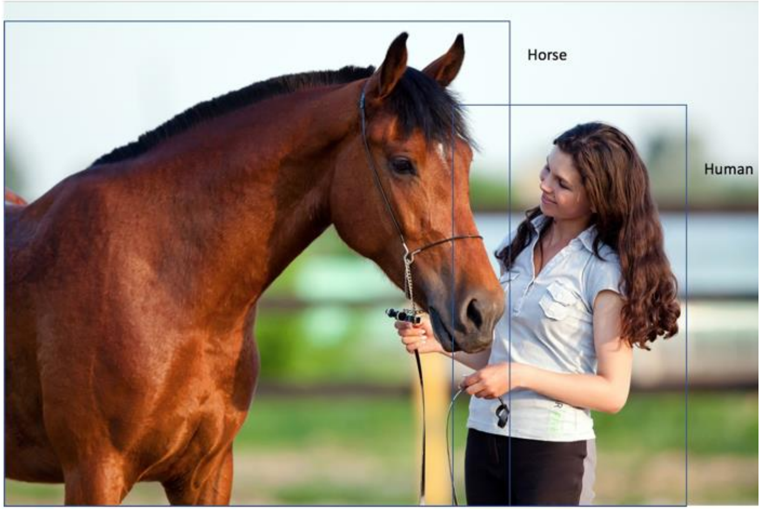


Figure 42: Output of Fast R-CNN

In the output of Fast R-CNN, there are two regions of interest (rectangles) and a region of interest contains a female while the other region of interest contains a horse.

Therefore, by pushing these kinds of information to the LSTM for question answering. It is possible that the model can answer the question about counting, like “How many people are there in this image” or “Is there any horse in this image”, more accurate. Therefore, this model can help us improve the accuracy of our visual question answering model.

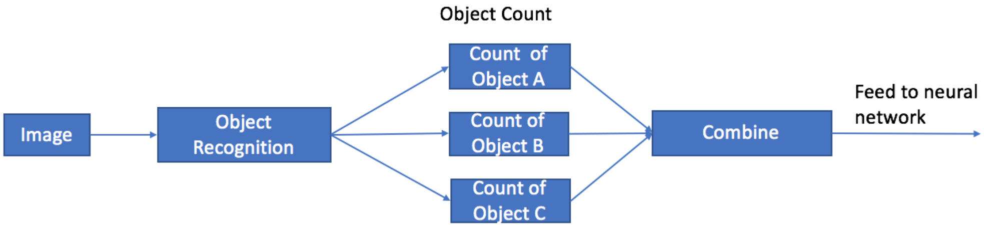


Figure 43: Count Model

In this model, we want to use object recognition to count each object in an image and together with the image features to enhance the accuracy of model.

4.2.2 Image Caption Model

i) NIC Model

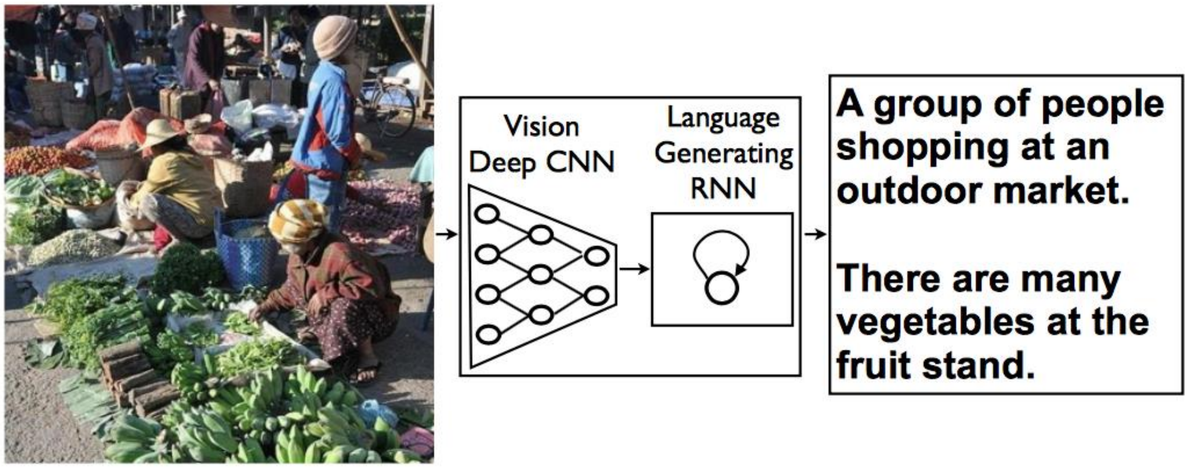
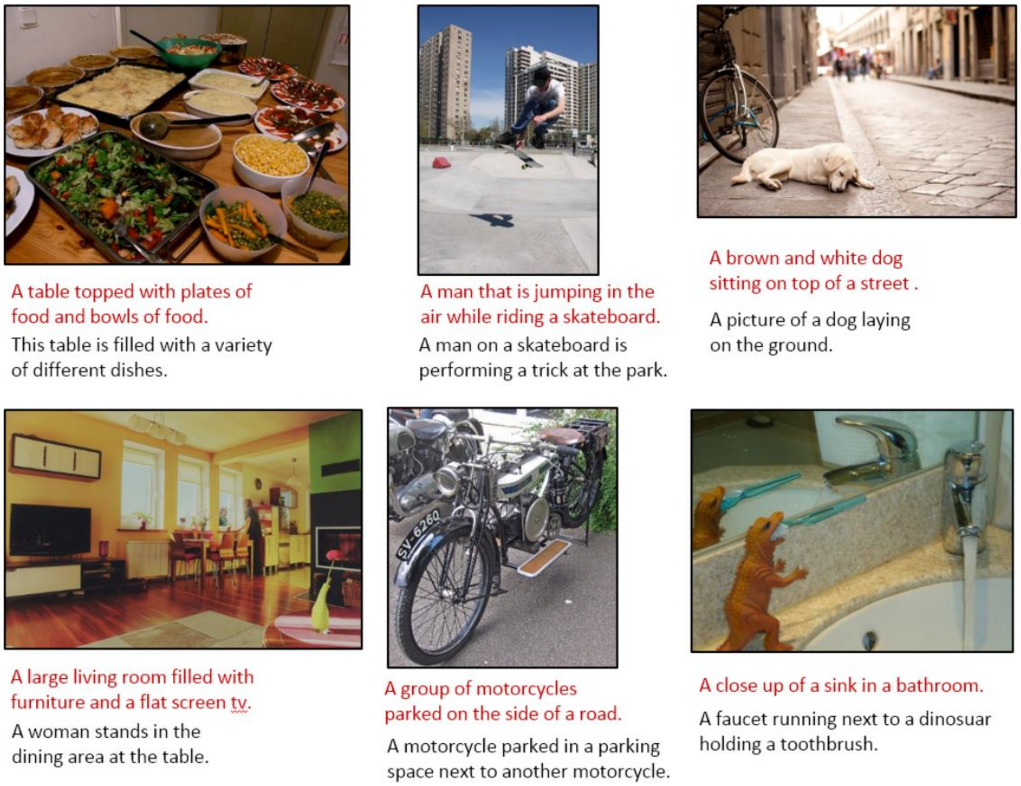

Vision
Deep CNN
Language
Generating
RNN
A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.

Figure 44: Sample Output

The score of this model is as following:

Metric	BLEU-4	MeTEOR	cider
NIC	27.7	23.7	85.5
Random	4.6	9.0	5.1
Neighbor	9.9	15.7	36.5
Human	21.7	25.2	85.4

ii) Recurrent Visual Representation


A table topped with plates of food and bowls of food.

This table is filled with a variety of different dishes.

A man that is jumping in the air while riding a skateboard.

A man on a skateboard is performing a trick at the park.

A brown and white dog sitting on top of a street .

A picture of a dog laying on the ground.

A large living room filled with furniture and a flat screen tv.

A woman stands in the dining area at the table.

A group of motorcycles parked on the side of a road.

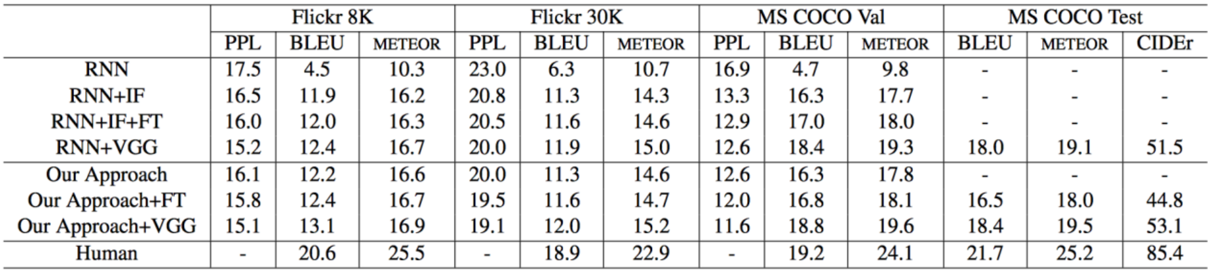
A motorcycle parked in a parking space next to another motorcycle.

A close up of a sink in a bathroom.

A faucet running next to a dinosuar holding a toothbrush.

Figure 45: Sample Result

The overall score of this model:



	Flickr 8K			Flickr 30K			MS COCO Val			MS COCO Test		
	PPL	BLEU	METEOR	PPL	BLEU	METEOR	PPL	BLEU	METEOR	BLEU	METEOR	CIDEr
RNN	17.5	4.5	10.3	23.0	6.3	10.7	16.9	4.7	9.8	-	-	-
RNN+IF	16.5	11.9	16.2	20.8	11.3	14.3	13.3	16.3	17.7	-	-	-
RNN+IF+FT	16.0	12.0	16.3	20.5	11.6	14.6	12.9	17.0	18.0	-	-	-
RNN+VGG	15.2	12.4	16.7	20.0	11.9	15.0	12.6	18.4	19.3	18.0	19.1	51.5
Our Approach	16.1	12.2	16.6	20.0	11.3	14.6	12.6	16.3	17.8	-	-	-
Our Approach+FT	15.8	12.4	16.7	19.5	11.6	14.7	12.0	16.8	18.1	16.5	18.0	44.8
Our Approach+VGG	15.1	13.1	16.9	19.1	12.0	15.2	11.6	18.8	19.6	18.4	19.5	53.1
Human	-	20.6	25.5	-	18.9	22.9	-	19.2	24.1	21.7	25.2	85.4

iii) Overall Structure

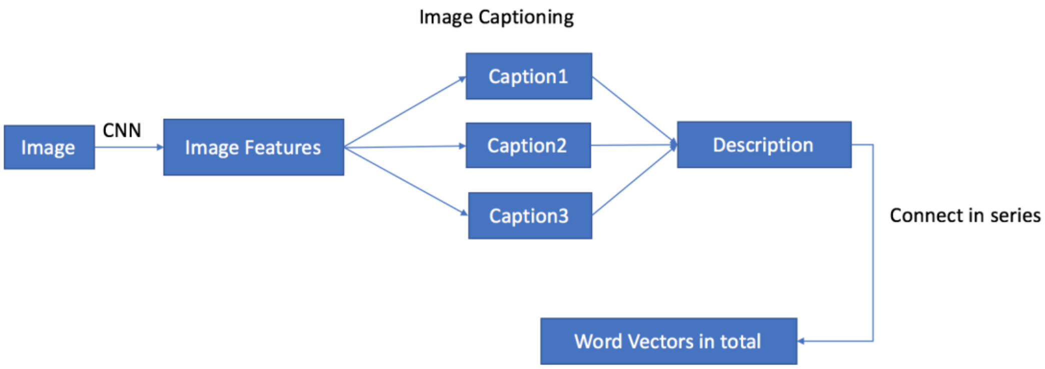


Figure 46: Caption Model

In our first model, we decide to using several different ways to do generate image captions and combine them into a sentence which represent the image. And in this case, we convert an image into sentence so that the Visual Question Answering problem is converted to a Text-Based Question Answering problem.

But this model still has many defects. First, current models in image caption field cannot describe the content in image well and too many information will lose in this process.

4.3 Implementation and Training

The overall architecture of our model is like:

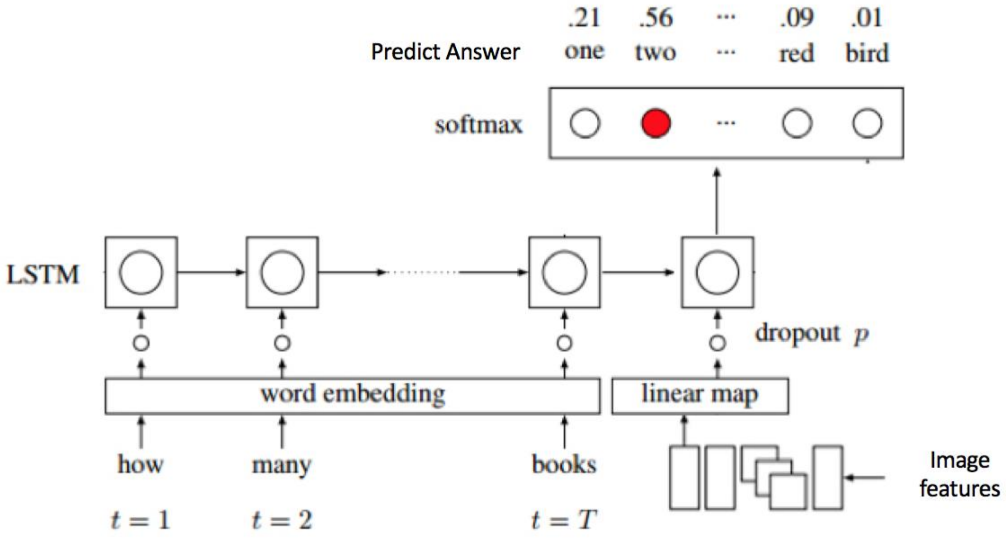


Figure 47: Overall Architecture

In the first, we used a preprocess program to extract the train images' features and using HDF5 binary data format to store it. Then we used the word embedding to convert the words to vectors and combining the question vectors and the image features, and last we feed the combined data to the LSTM (Long Short-Term Memory) to train the model.

4.3.1 Data Loader

Since the training images and the questions are separated in different files and in the format of JSON.

The image file name is like COCO_train2014_[image id].jpg. For example, there is an image whose name is COCO_train2014_000000348957.jpg, then the image is 348957.

In the train question file, the format is as following and it contains an image id and a question id.

```
{
    "question": "How many windows can you see?",
    "image_id": 434410,
    "question_id": 4344102
}
```

The format of answer is as following, and it contains a question id and several choices while each choice has its own answer id.

```
{
    "answer": "kettles",
    "answer_confidence": "yes",
    "answer_id": 6
}
```

Above all, the image id is not and question id may not continuously. In this case, we have a function to load the image, question and answer and binding them together.

4.3.2 Extract Image Features

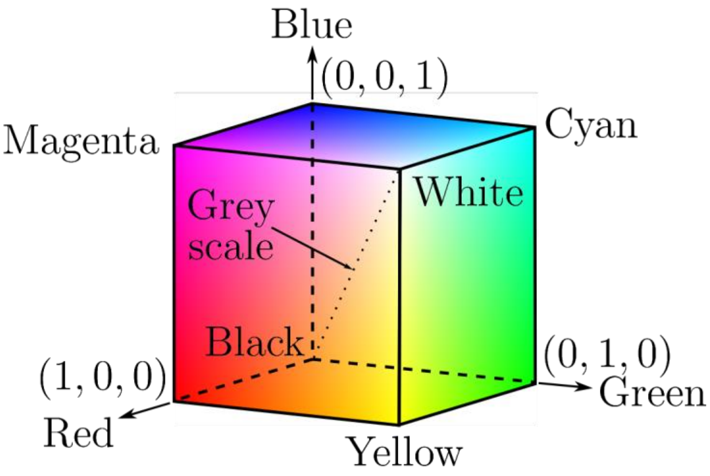


Figure 48: RGB Channels

After we can load the train data, we need to using Convolutional Neural Network to extract the feature from the images. The VGG-16 model accepts R, G, B channels that between 0 and 1. Then for each image we split it to R channel, G channel and B channel respectively. And then resize it to 224 * 224 dimension.

For example, in the following image, we first separate it into its three-color channel, and then resize its dimension to (224, 224).

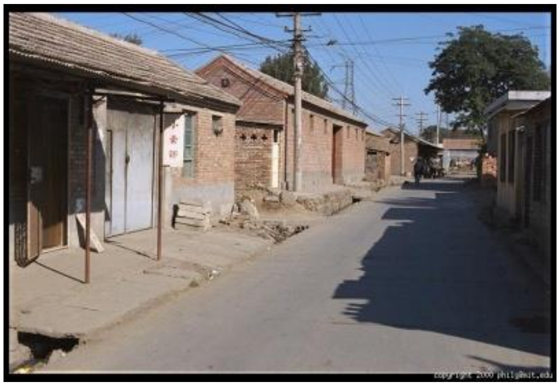


Figure 49: Train Image

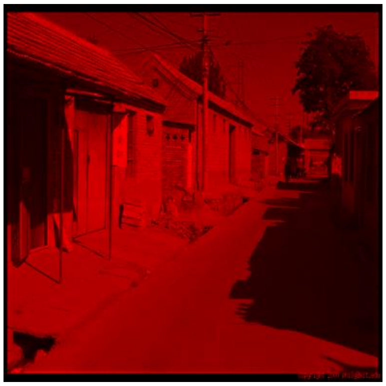


Figure 50: R Channel

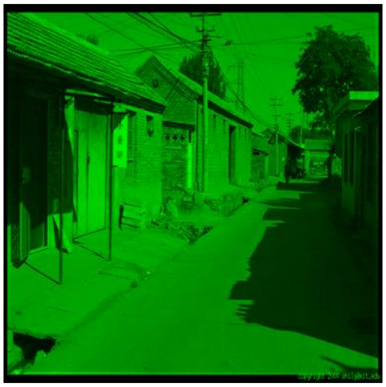


Figure 51: G Channel

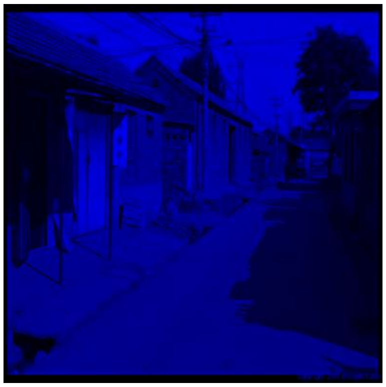


Figure 52: B Channel

```
def load_image_array(image_file):
    img = misc.imread(image_file)
    # GRAYSCALE
    if len(img.shape) == 2:
        img_new = np.ndarray( (img.shape[0], img.shape[1], 3), dtype =
'float32')
        img_new[:, :, 0] = img
        img_new[:, :, 1] = img
        img_new[:, :, 2] = img
        img = img_new

    img_resized = misc.imresize(img, (224, 224))
    return (img_resized/255.0).astype('float32')
```

After preprocessing the train images files, we feed them into the VGG-16 Model to extract their features.

i) VGG-16 Model

The preprocess program use the VGG-16 model as the basic model, the structure is like following.

```
layer {
  name: "conv1_1"
  type: "Convolution"
  bottom: "data"
  top: "conv1_1a"
  convolution_param {
    num_output: 64
    pad: 1
    kernel_size: 3
  }
}
layer {
  name: "relu1_1"
  type: "ReLU"
  bottom: "conv1_1a"
  top: "conv1_1"
}
.
.
.
layer {
  name: "drop7"
  type: "Dropout"
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc8"
  type: "InnerProduct"
  bottom: "fc7"
  top: "fc8"
  inner_product_param {
    num_output: 1000
  }
}
layer {
  name: "prob"
  type: "Softmax"
  bottom: "fc8"
  top: "prob"
}
```

The rough structure can describe as following:

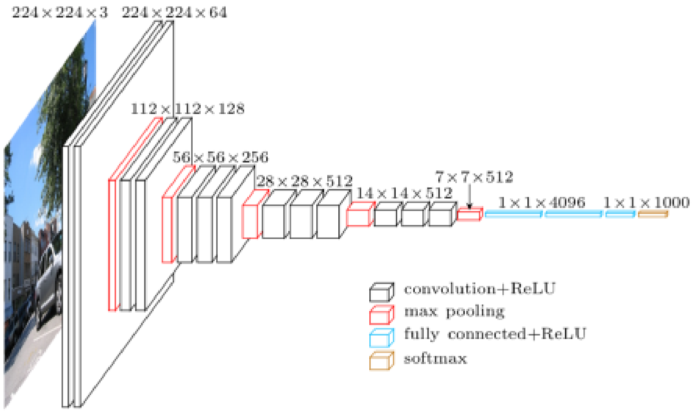


Figure 53: The Structure of VGG-16 Model

ii) Features Format

The feature of an image is a vector whose shape if 4096 * 1.

For each train images, we have a vector whose shape is 4096 * 1, after we finishing extracting the image features, we can obtain a matrix whose shaper is number of images * 4096, and this large matrix will store in HDF5 binary data format file. Since HDF5 has a smaller size when we store the data.

4.3.3 Training

We build up a LSTM Model. This LSTM model contains 2 hidden layers. and each hidden layer has 512 neurons. Therefore, for each hidden layer, it will output a 512-length vector.

For Input layer, it maintains three parameters which are Wimg, Wemb, bimg. Wimg is a matrix [4096, 512] and all elements in this matrix are random values in range [-1, 1] initially. Wemb is a matrix [q_vob_len, 512] where q_vob_len is the number of words in all questions and all the elements in this matrix are 0 initially. bimg is a 512-length vector which represents the bias. The usages of three parameters will be described below. When the data batch feed in, since the dimension of image features and words is not the same, we need to regularize these dimensions. For words, we do word embedding to generate a 512-length vector to represents words using Wemb. For image, we multiply the Wimg and 4096-length image feature vector and add the bimg to generate another 512-length vector. Therefore, the dimensions of image feature vector and word vector are regularized.

4.3.4 Predicting

We first load the pre-trained QA model and VGG model from local. Then we use VGG to produce the image feature vector of input image. After that, we use pre-known knowledge to embedding the words in question and reduce the dimension of image feature vector. We concatenate these vectors and feed resultant matrix into pre-trained QA and generate the word vector representation of answer. Finally, we look up this vector in our pre-known knowledge and find out the answer.

4.4 Training Process

The train log can help us to improve our models, and the following figures are showing the Training Accuracy and Losses during each epoch.

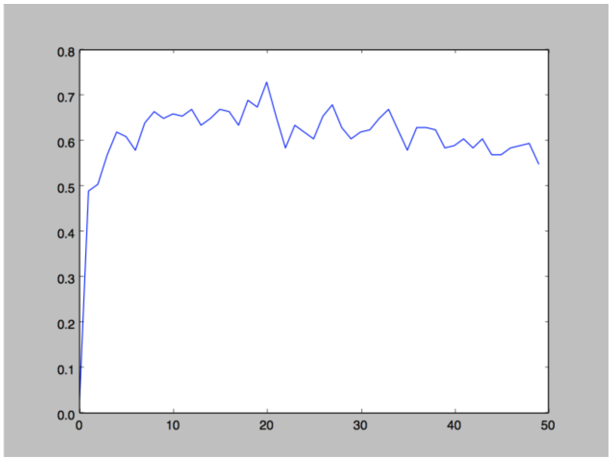


Figure 54: Training Accuracy

As we can from the above figure, the training accuracy go up quickly in the beginning of train process, then in maintain an accuracy between 50% to 60% and trend to converge in the middle of training process. And after finishing the whole training, the final accuracy is about 55%.

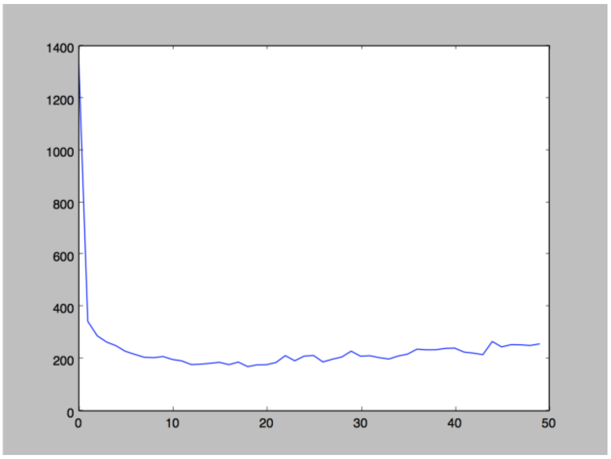


Figure 55: Training Losses

From the above figure, we can know that the train process was about to converge quickly. The training accuracy and the training losses didn't change a lot after epoch 10. And the total training loss is about 300 eventually.

4.5 EVALUATION

4.5.1 Evaluation Metric

We use the evaluation code which offer by the Visual Question Answering challenge organizers. In this evaluation metric, the accuracy of machine will be averaged over all 10 choose 9 sets of human annotators to enhance the robustness of accuracy.

The accuracy formula is as follow:

$$Acc(ans) = \min\{\frac{\#\ humans\ that\ said\ ans}{3}, 1\}$$

4.5.2 Evaluation Dataset

The dataset we used for evaluation is mentioned in Chapter Methodology.

4.6 Accuracy

4.6.1 Overall Test Result

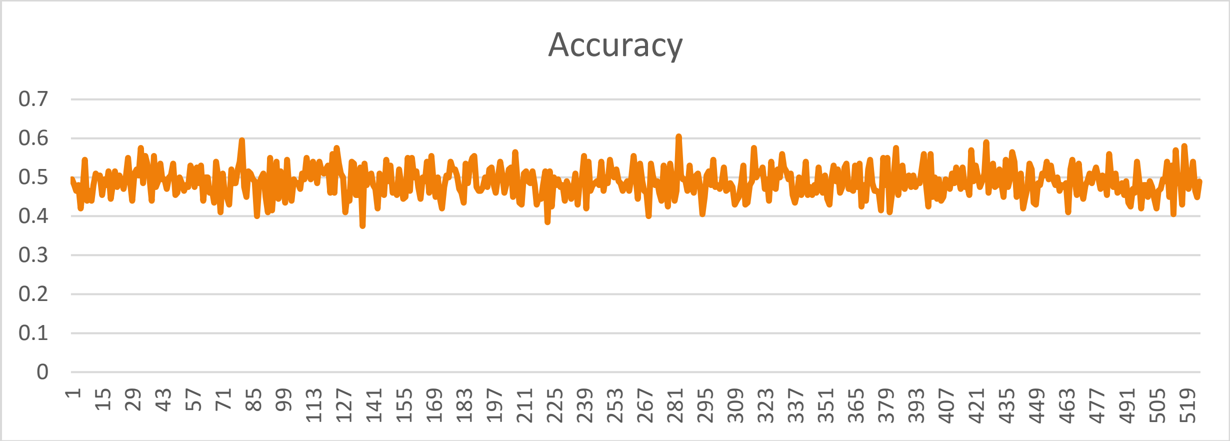


Figure 56: Overall Accuracy of Current Model

4.6.2 Detail Accuracy

Figure 57: Accuracy of Our Model

	Test Set			
Model	Yes / No	Number	Other	Overall
Baseline All Yes	70.97	0.36	1.21	29.88
Baseline Prior Per Question Type	71.40	34.90	8.96	37.47
Baseline Nearest Neighbor	71.89	24.23	22.10	42.85
UC Berkeley & Sony	83.62	39.47	58.00	49.12
Our Model	74.62	31.76	31.32	45.87

4.7 SAMPLES

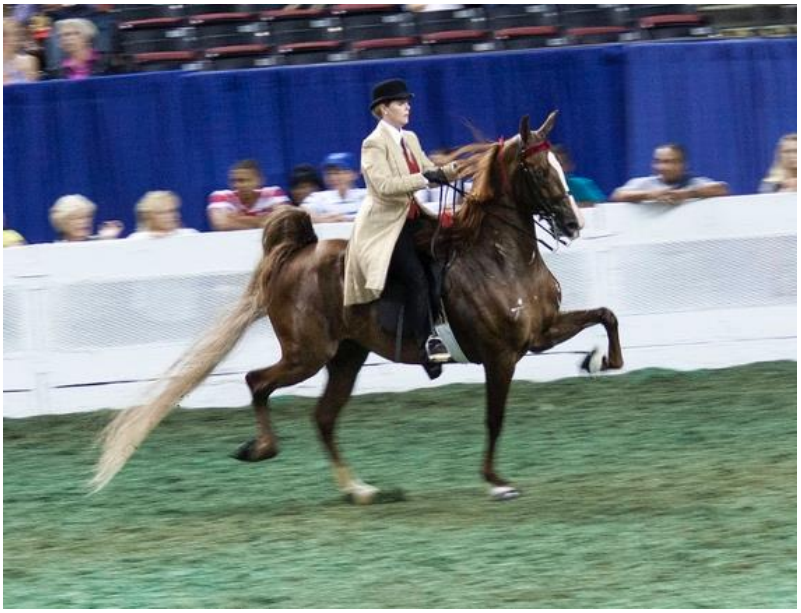


Figure 58: Sample 1

Question: What animal is this?

Top 5 Answers: horse, cow, elephant, dog, zebra

Question: What color is this animal?

Top 5 Answers: brown, white and brown, white, black and white, black

Question: What are they doing now?

Top 5 Answers: walking, talking, eating, standing, taking picture

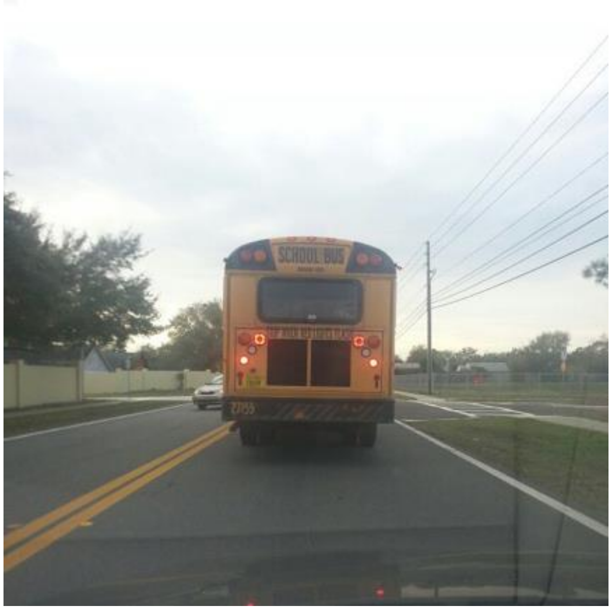


Figure 59: Sample 2

Question: What is this that in the front?

Top 5 Answers: train, building, truck, bikes, teddy bears

Question: How about the weather now?

Top 5 Answers: cloudy, cold, rainy, wet, evening

Question: How many cars are in there?

Top 5 Answers: 2, 3, 1, 5, 4

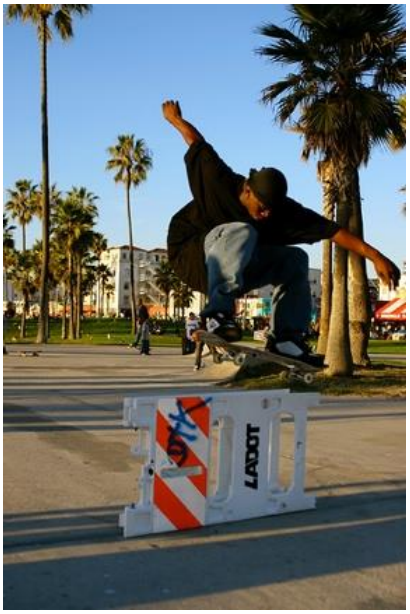


Figure 60: Sample 3

Question: What is this man doing?

Top 5 Answer: skateboarding, jumping, snowboarding, sitting, taking picture

Question: How about the weather?

Top 5 Answer: rainy, sunny, cloudy, clear, overcast

Question: What is the gender of the person?

Top 5 Answer: male, female, child, boy, women

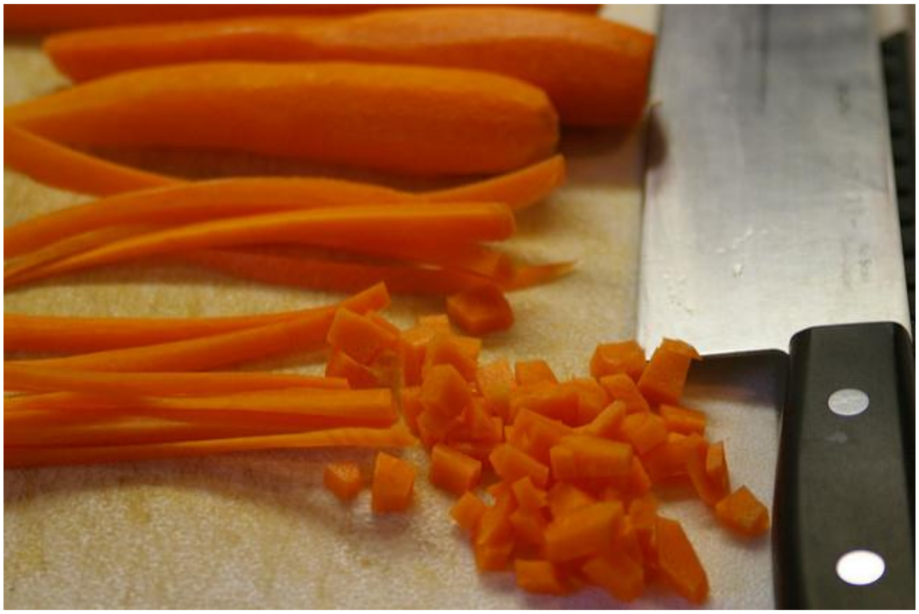


Figure 61: Sample 4

Question: what are sitting on the counter in different stages of cutting with a knife?

Top 5 Answers: napkin, chocolate, glass, bread, fries

Question: What is the color of this vegetables?

Top 5 Answer: orange, yellow, green, blue, brown

Question: What is the name of this vegetable?

Top 5 Answer: carrot, parsley, apple, bananas, carrots

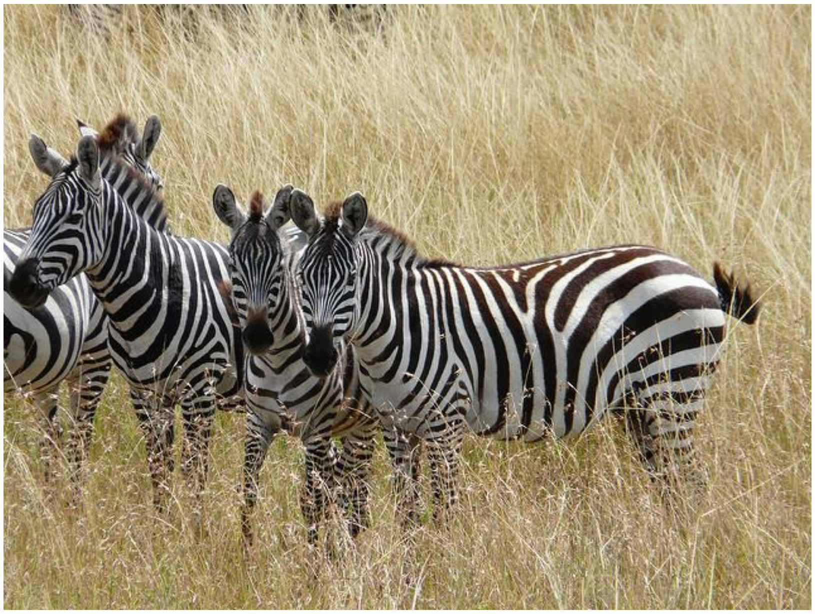


Figure 62: Sample 5

Question: What is the color of this animal?

Top 5 Answer: black and white, white, brown, black, gray

Question: What is this animal?

Top 5 Answer: zebra, giraffe, horse, cow, zebras

Question: How many animals are there?

Top 5 Answer: 2, 3, 4, 1, 5

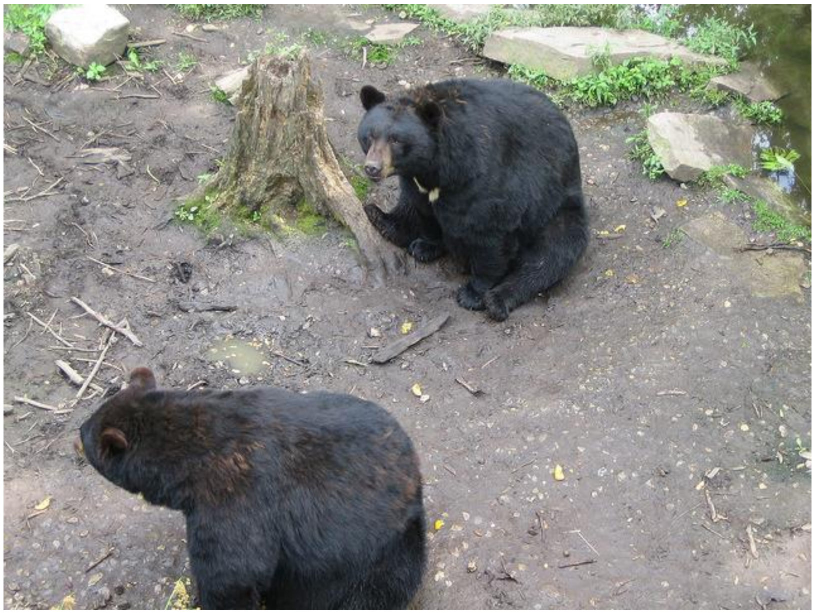


Figure 63: Sample 6

Question: How many animals are there?

Top 5 Answer: 2, 1, 3, 4, 5

Question: What is the name of this animal?

Top 5 Answer: bird, bear, elephant, cat, dog

Question: what are sitting down on the ground?

Top 5 Answer: teddy bear, bear, snow, stick, bird

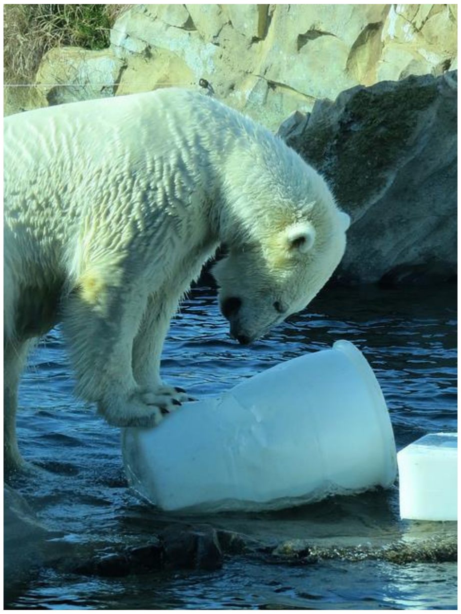


Figure 64: Sample 7

Question: what is playing with the large UNK of ice?

Top 5 Answer: surfer, surfing, surfboard, water, boat

Question: What is the color of this animal?

Top 5 Answer: white, brown, black, white and brown, gray

Question: How many animals are there in the image?

Top 5 Answer: 2, 1, 4, 3, 6

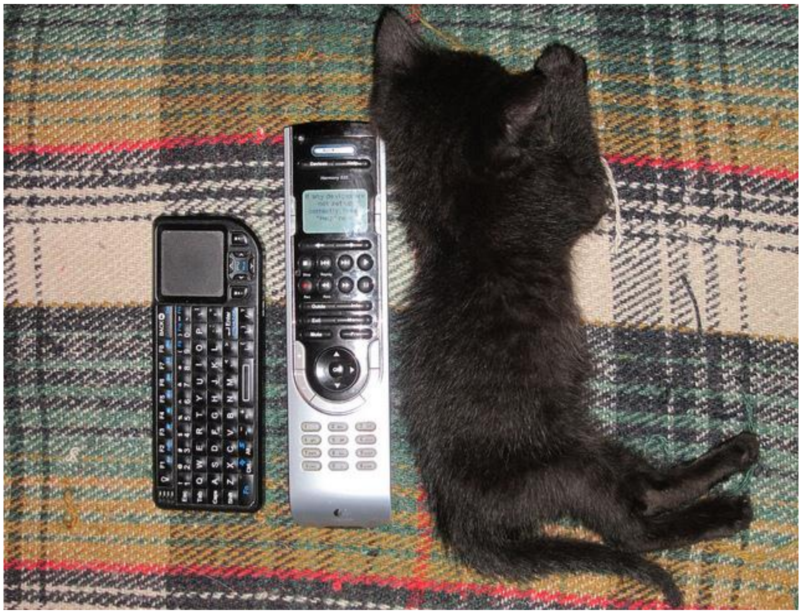


Figure 65: Sample 8

Question: How many animals are there in the image?

Top 5 Answer: 2, 3, 1, 4, 5

Question: What is the name of object that in the middle?

Top 5 Answer: remote, laptop, toy, cat, books

Question: What is the name of object that in the left?

Top 5 Answer: laptop, keyboard, remote, phone, bowl

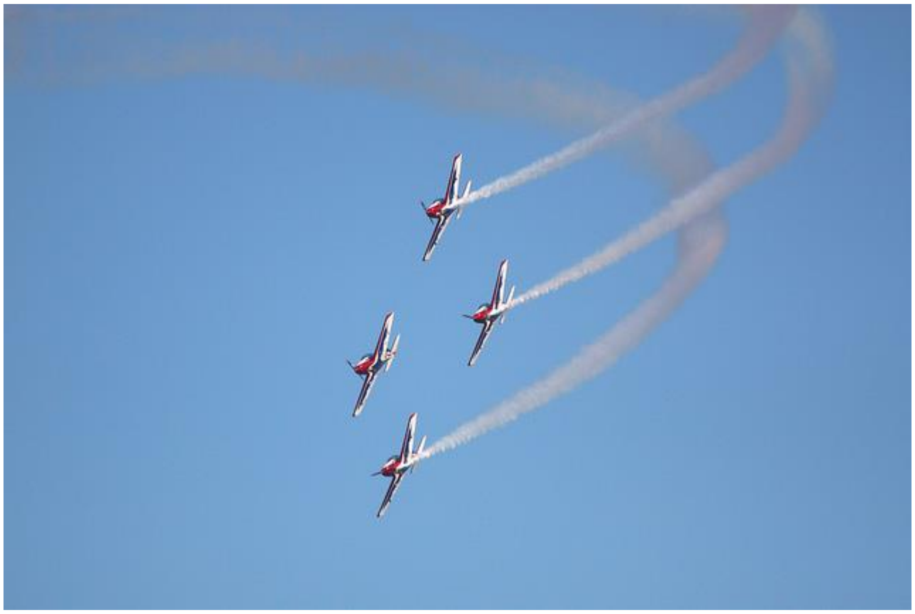


Figure 66: Sample 9

Question: what are flying through the sky?

Top 5 Answer: kites, plane, kite, clouds, airplane

Question: What is the color of background?

Top 5 Answer: blue, red, green, orange, yellow

Question: How many objects in the sky?

Top 5 Answer: 13, 10, 4, 5, 1

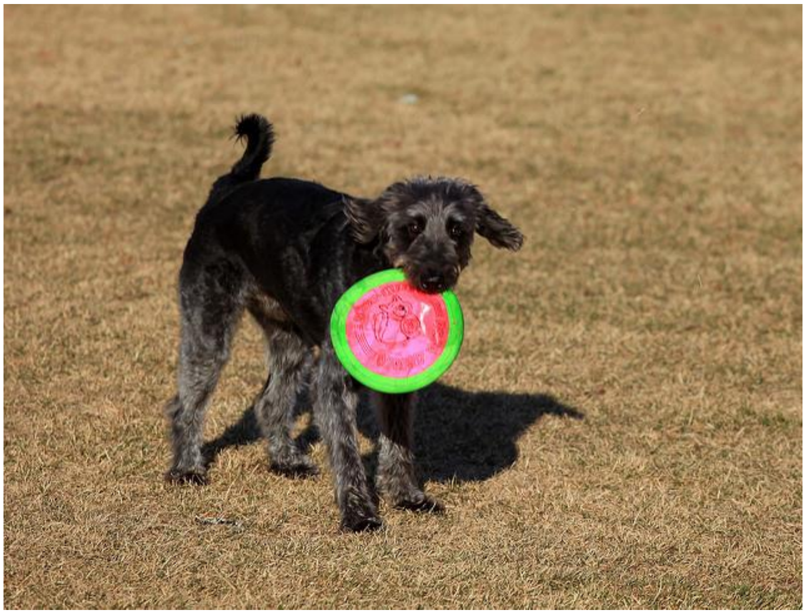


Figure 67: Sample 10

Question: what is the black dog holding?

Top 5 Answer: frisbee, kite, bat, carrot, dog

Question: What is the color of this dog?

Top 5 Answer: black, brown, black and white, white and brown, white

Question: what is the color of the object that the black dog holding?

Top 5 Answer: yellow, blue, white, pink, red

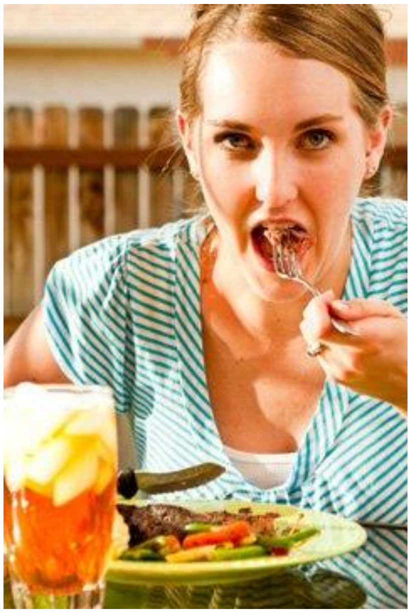


Figure 68: Sample 11

Question: What is this woman doing?

Top 5 Answer: cooking, eating, taking picture, smiling cutting

Question: What is this woman holding?

Top 5 Answer: banana, fork, plate, knife, pizza

Question: what color is this woman's eyes?

Top 5 Answer: blue, brown, black, green, red

4.8 Analysis of Result

The model we used now get an accuracy of 49.12% over the test set and we present the accuracy of baseline model and the model from UC Berkeley & Sony. The accuracy of model from UC Berkeley & Sony represent the highest accuracy of current Visual Question Model, we can see that the Visual Question Answering is still a hard problem over the world since the accuracy of number-relative question is much low than the accuracy of Yes / No questions.

In our opinion, there are several reasons that the accuracy is not so high. First, the Convolutional Neural Network we use is mainly used to classify the objects, which means that the image features that we extracted contained the class of each object in the image but didn't contain the count of each objects. In this case, the model is to use "common sense" to answer the number-relative questions. Thus, the model has much high accuracy on Yes / No question than the number-relative question.

In some extent, the current model has a high performance in recognizing the type of questions. If the question is asking the number of an object, the top 5 answers will be a number, and if the question is asking about the weather, the top answers will also show some kinds of weathers. In this case, we can say that our current model has high accuracy of answering the question-type but may have low accuracy in answering the question.

We will focus on all the above bottlenecks in the next term and to enhance the whole accuracy of the model.

Chapter 5: Implementation of Neural Reasoner

5.1 Overview

5.1.1 Overall Structure

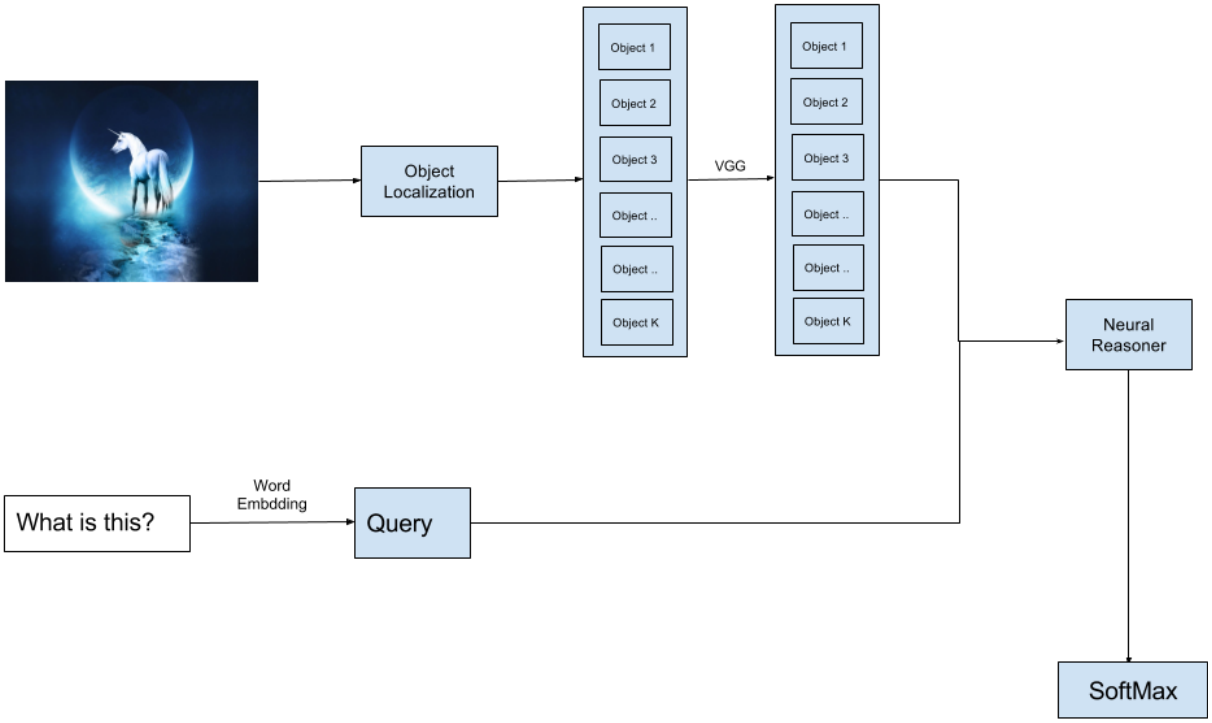


Figure 69: Overall Structure of Neural Reasoner Version

5.1.2 Introduction

A train sample can be split into two part, image and question.

For the image, we can first apply the object localization to find the significant objects in an image. Then we use VGG-Model to process these extracted objects. These objects and their location can be regarded as the facts in the neural reasoner.

For the question, we use word embedding to convert the natural language sentence into matrix.

Until now, we obtain the facts and question, the feed them into the neural reasoner iteratively to obtain the result.

5.2 LOCALIZATION

We have surveyed several kinds of object localization method, e.g., Faster R-CNN [4], Microsoft’s Edge Boxes, and YOLO (You Only Look Once). We choose to use YOLO in our project since it is the easiest to modify.

5.2.1 Faster R-CNN

i) Structure

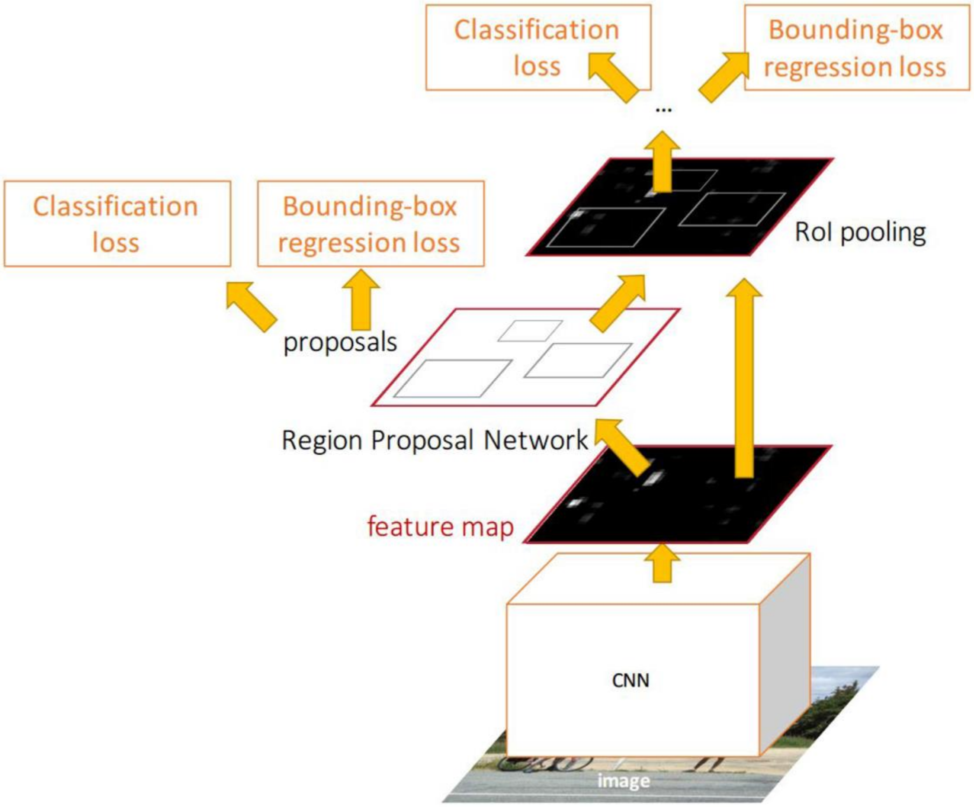


Figure 70: Structure of Faster R-CNN

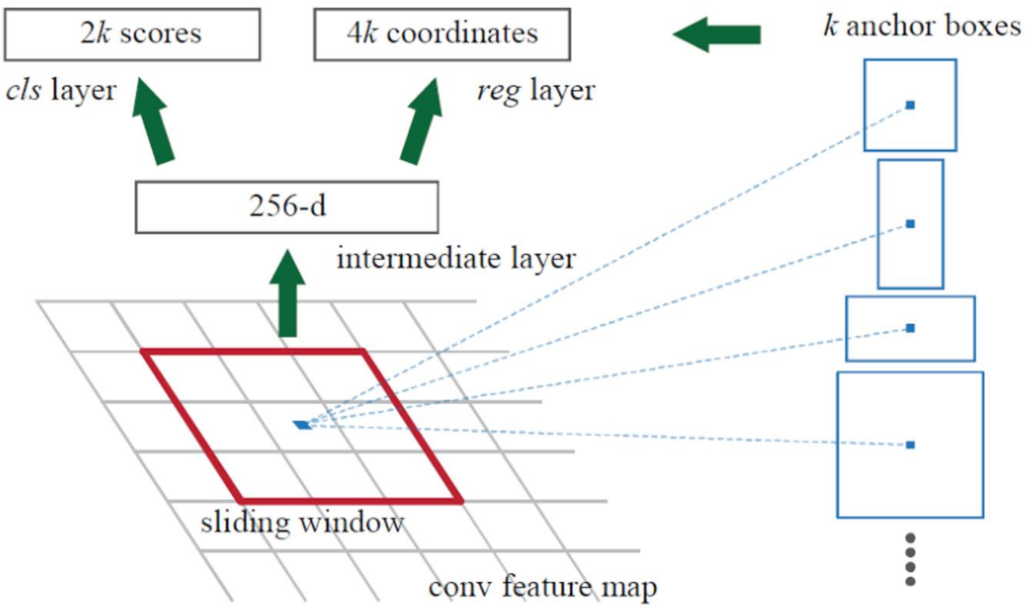


Figure 71: Structure of Region Proposal Network

ii) Detection Result

	Fast R-CNN
Test time per image	**0.32 seconds**
(Speedup)	**146x**
Test time per image with Selective Search	**2 seconds**
(Speedup)	**25x**

iii) Disadvantages:

Faster R-CNN is hard for us to modify.

5.2.2 Edge Boxes

Edge Boxes [5] uses edges in the image to generate the object box proposal. It does not depend on the deep learning method.

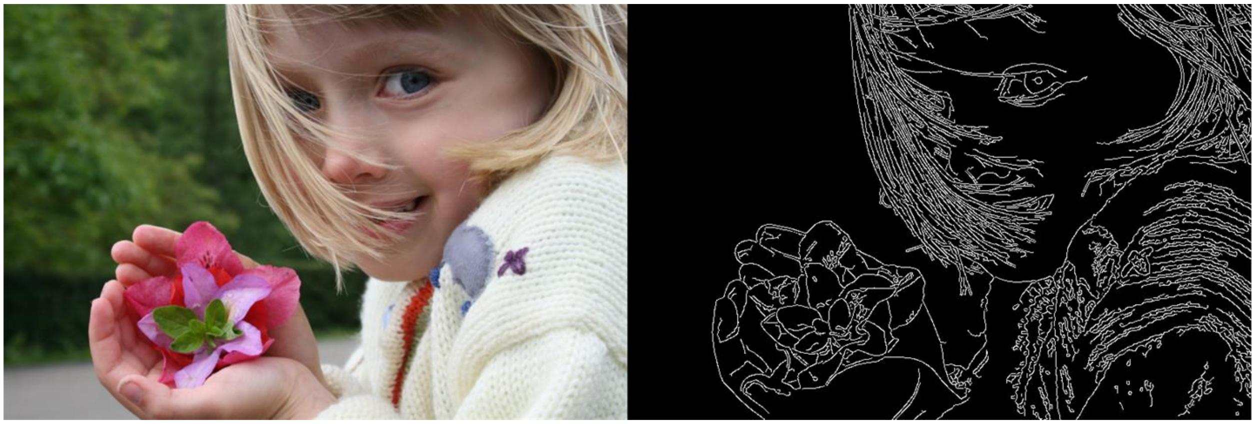


Figure 72: Detect Edges in an Image

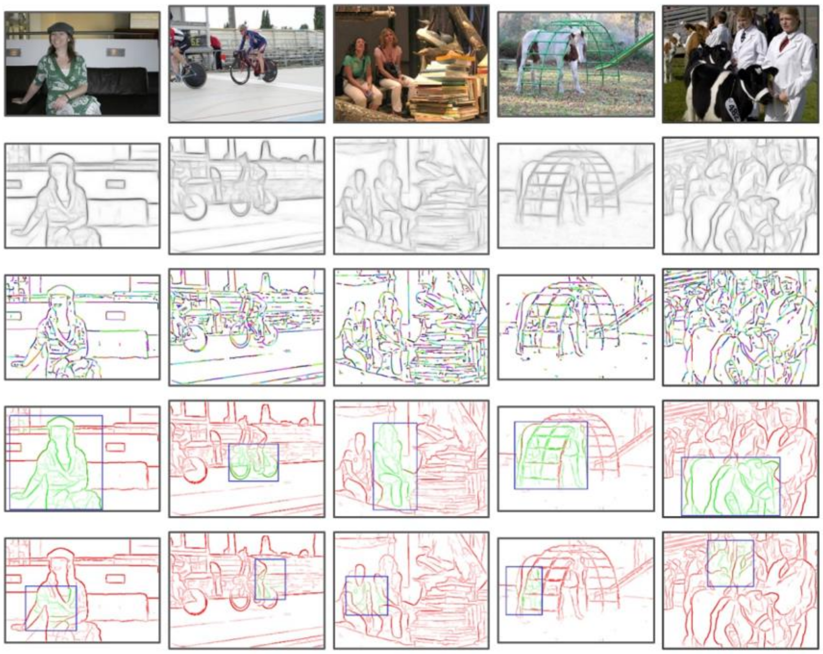


Figure 73: Process of Edge Boxes

5.2.3 YOLO

YOLO (You Only Look Once) is a real-time object detection system and we use YOLO in our project. We add a function in the source code of YOLO help us extract the object in the images.

```
void extract_detector(char *datacfg, char *cfgfile, char *weightfile, char
*filename, float thresh, float hier_thresh)
{
    printf("%s¥n", "Start Extracting");
    list *options = read_data_cfg(datacfg);
    char *name_list = option_find_str(options, "names", "data/names.list");
    char **names = get_labels(name_list);

    image **alphabet = load_alphabet();
    network net = parse_network_cfg(cfgfile);
    if(weightfile){
        load_weights(&net, weightfile);
    }
    set_batch_network(&net, 1);
    srand(2222222);
    clock_t time;
    char buff[256];
    char *input = buff;
    int j;
    float nms=.4;

    DIR *d;
    struct dirent *dir;
    d = opendir(filename);
    if (d)
    {
        while ((dir = readdir(d)) != NULL)
        {
            if (dir->d_name[0] != '.')
            {
                strncpy(input, filename, 256);
                strcat(input, "/" );
                strcat(input, dir->d_name);

                printf("%s ", dir->d_name);

                image im = load_image_color(input,0,0);
                image sized = letterbox_image(im, net.w, net.h);

                layer l = net.layers[net.n-1];

                box *boxes = calloc(l.w*l.h*l.n, sizeof(box));
                float **probs = calloc(l.w*l.h*l.n, sizeof(float *));
                for(j = 0; j < l.w*l.h*l.n; ++j) probs[j] = calloc(l.classes +
1, sizeof(float *));

                float *X = sized.data;
                time=clock();
                network_predict(net, X);
                get_region_boxes(l, 1, 1, thresh, probs, boxes, 0, 0,
hier_thresh);
                if (l.softmax_tree && nms)
                    do_nms_obj(boxes, probs, l.w*l.h*l.n, l.classes, nms);
                else if (nms)
                    do_nms_sort(boxes, probs, l.w*l.h*l.n, l.classes, nms);

                draw_detections(sized, l.w*l.h*l.n, thresh, boxes, probs,
names, alphabet, l.classes);
                save_image(sized, dir->d_name);

                free_image(im);
                free_image(sized);
                free(boxes);
                free_ptrs((void **)probs, l.w*l.h*l.n);
            }
        }
        closedir(d);
    }
}
```

Firstly, we will use the YOLO to preprocess the train image. For each image in the train set, we apply the YOLO to find the significant objects and print out their positions. The output is as following:

$$Image\ Name\ Left_1\ Right_1\ Top_1\ Bottom_1 \ldots Left_n\ Right_n\ Top_n\ Bottom_n$$

The sample output is as following:

```
COCO_train2014_000000000009.jpg 207 265 49 99 234 294 54 96 253 304 90 126 307
366 91 138 0 155 134 278 321 412 173 307 212 402 45 182 146 388 197 367 0 291
73 285 0 407 59 353 10 381 195 350
```

```
COCO_train2014_000000000049.jpg 171 197 219 258 282 289 276 289 141 153 282
305 124 188 196 280 156 196 207 271 218 269 201 274 213 275 355 396 111 203
207 337 195 292 198 337
COCO_train2014_000000000089.jpg 300 310 126 223 271 300 134 224 327 343 127
215 360 374 120 221 379 397 104 241 324 406 282 333 0 91 305 362 328 411 304
347 5 83 48 185 94 301 183 368
COCO_train2014_000000000109.jpg 5 26 130 142 385 391 247 257 338 350 256 276
COCO_train2014_000000000149.jpg 283 289 122 129 287 294 193 203 249 259 246
251 356 365 277 293 30 69 281 299 0 415 262 290
COCO_train2014_000000000309.jpg 297 356 49 131 156 273 152 276 46 162 157 344
119 215 155 316
COCO_train2014_000000000349.jpg 0 377 115 315
```

Once we finishing the preprocess of the train image, we extract the image and objects information using VGG model and store this information in HDF5 format.

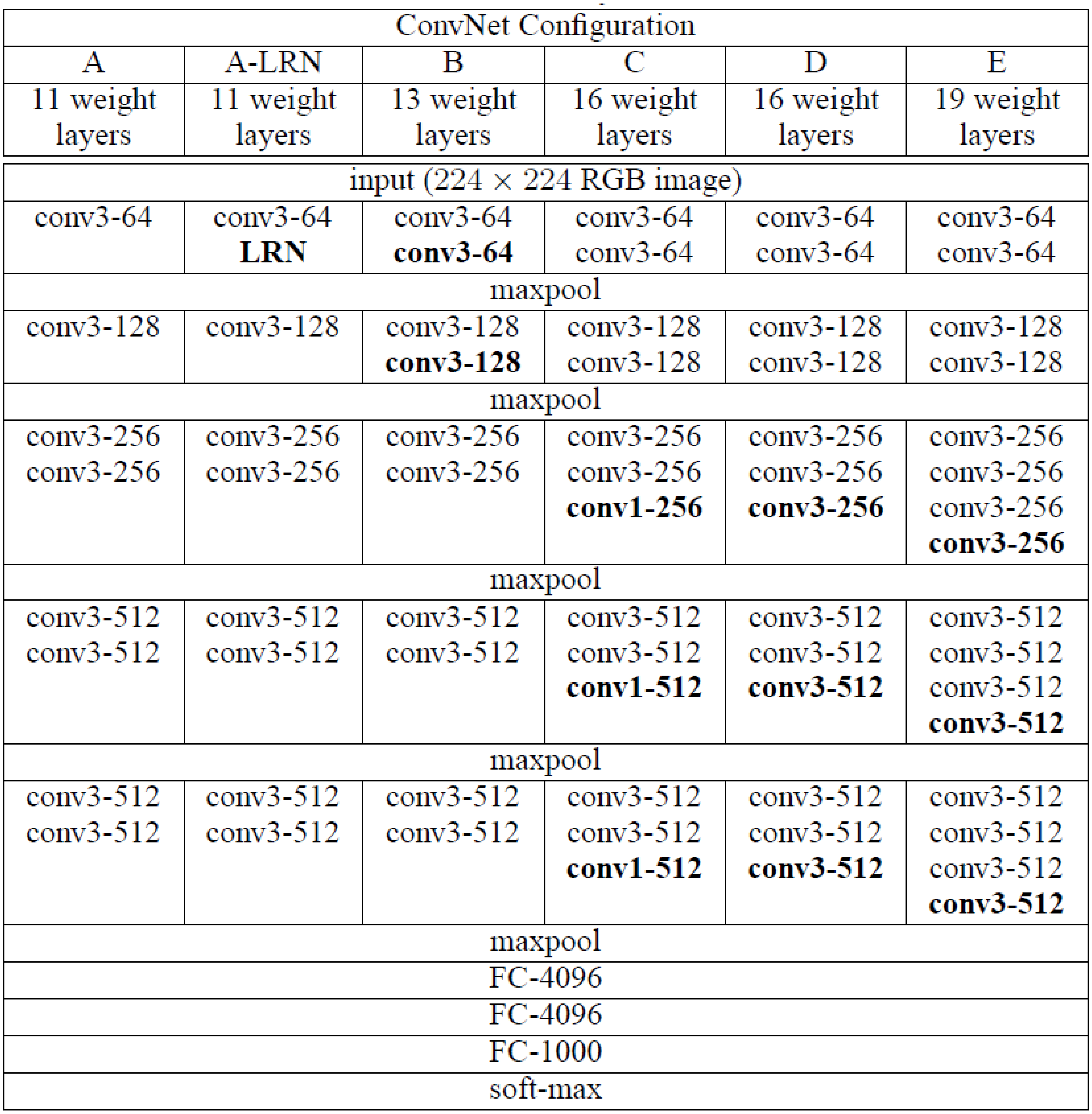
```
def load_image_localization(filename):
      file = open(filename, ’r’)
      dic = {}
      for line in file.readlines():
            info = line.split()
            image_id = re.findall(″[0-9]+″, info[0])[1]
            objects = []

            for i in range((len(info) - 1) / 4):
                  objects.append([int(info[4 * i + 1]), int(info[4 * i + 2]),
int(info[4 * i + 3]), int(info[4 * i + 4])])
            dic[image_id] = objects

      return dic
```

5.3 VGG MODEL

The configuration of VGG model is as following:



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 74: Configurations of VGG Model

Configuration D is the best model in these configurations. For the configuration D, it has about 138M parameters and occupy 93M memory.

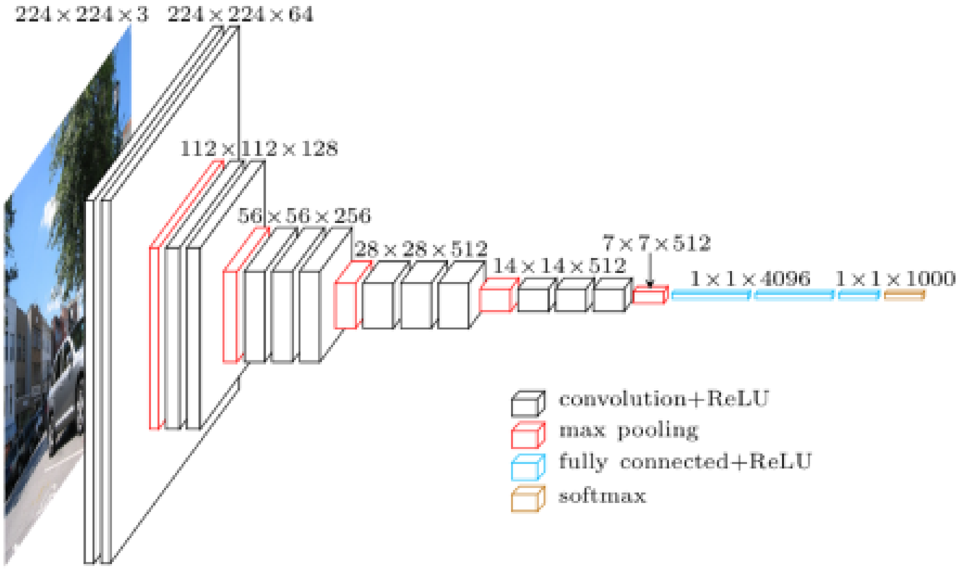


Figure 75: Structure of VGG Model

We do not need the final output of VGG model, what we want is the matrix that represent the input image. In this case, we do not need the output of SoftMax and we use the data in the second ReLU layer as the representation of this image. Since there are 4096 nodes in the second ReLU layer, the representation vector of the image is a vector whose size is 4096 * 1.

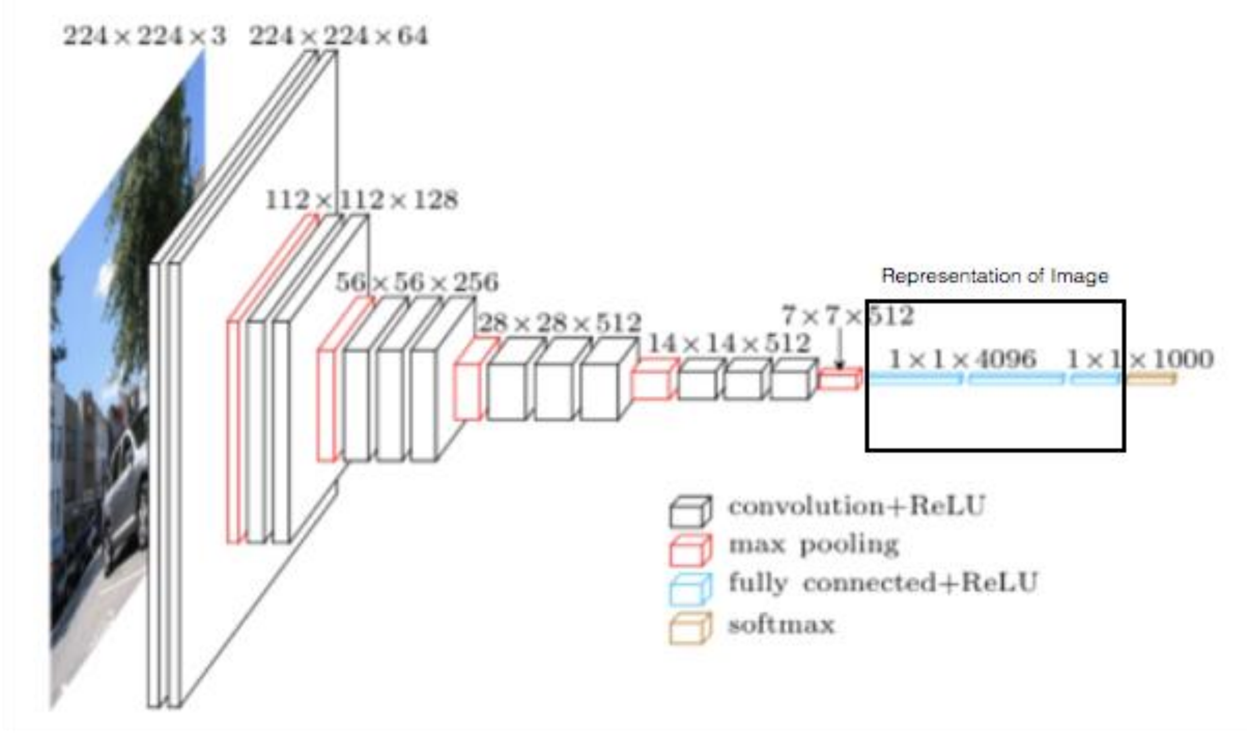


Figure 76: Approach to Extract the Image Information

5.4 REASONER

As we described in Chapter 3, multilayer perceptron (MLP) is able to determine the relationship between two input sentences according to supervision [11]. Therefore, we also can use MLP to determine the relationship between fact vectors and vector representation of question.

In our model, we implement three layers of reasoning. Inside each reasoning layer, there are two parts called question-image interaction and weighted pooling. For question-image interaction, we feed the question reasoning result from last layer and each fact vector into MLP, which can be represent as

$$q_i^l = \varphi(W_l \times (q^{l-1} * v_i) + b_l)$$

With q_i^l is the reasoning result of layer output of l – 1 layer and ith fact.

```
for j in range(self.options['num_facts']):
        feed = fact_update * query_update
        layer_1 = tf.add(tf.malmul(feed, reason_weights['layer1']),
reason_biases['layer1'])
        layer_1 = tf.nn.relu(layer_1)
        layer_2 = tf.add(tf.malmul(layer_1, reason_weights['layer2']),
reason_biases['layer2'])
        layer_2 = tf.nn.relu(layer_2)
        layer_o = tf.add(tf.matmul(layer_2, reason_weights['out']),
reason_biases['out'])
        query_out.append(layer_o)
        c.append(tf.add(tf.matmul(layer_o, pool_weights['A']),
        tf.add(tf.matmul(query_update, pool_weights['B']), pool_biases['B'])))
```

After all the question-layer interaction of this layer is produced, we need to select one as the most suitable answer and pass it into next layer. Therefore, we use weight pooling to find out the best answer. The process is represented as

$$C_i = tanh\left(W_A \times q_l^i + (W_B \times q^{l-1} + b_B)\right),$$
$$P = softmax(W_P \times C + b_P),$$
$$q^l = \sum_i P_i q_i^l$$

```
c = tf.nn.tanh(c)
p = tf.nn.softmax(c, name='p')
for j in range(len(p)):
        if j == 0:
                query_update = query_out[j] * p[j]
```

```
else:
    query_update += query_out[j] * p[j]
```

5.5 PREDICT

When predicting, we feed the image into YOLO model to generate objects and its coordinates. The object images are extracted from original image and fed into VGG model to get their feature vectors. Then these object image feature vector, the original image vector and word representation of question are fed into our VQA model to generate final answer.

Chapter 6: IMPLEMENTATION OF FRENCH QUESTION ANSWERING

6.1 OVERVIEW

6.1.1 Overall structure

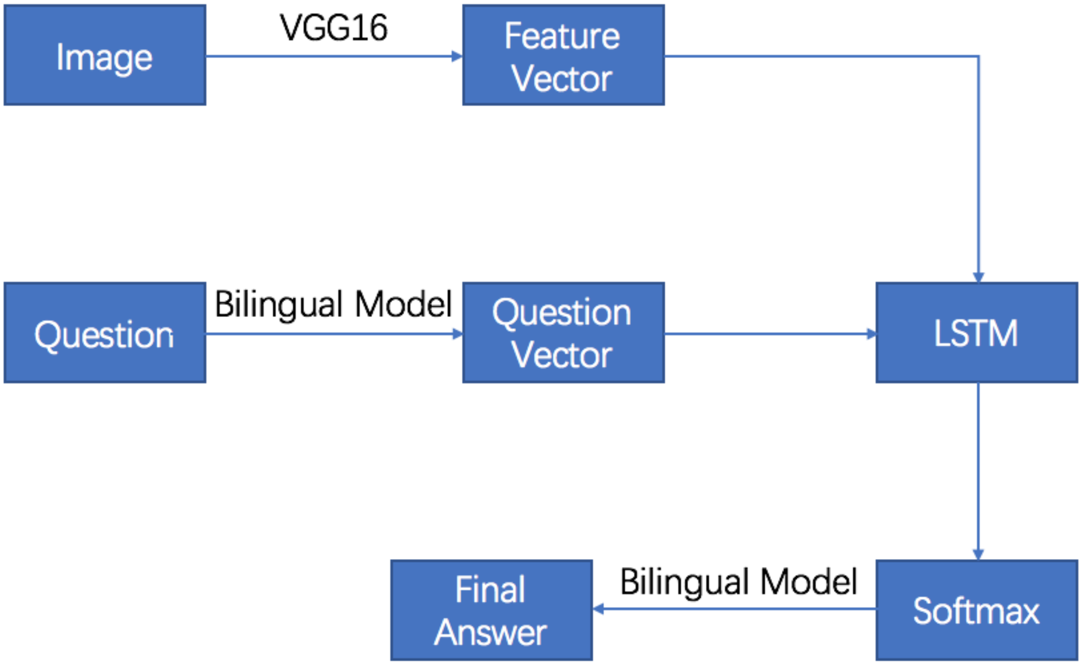


Figure 77: Structure of France Question Answering

6.1.2 Introduction

In this model, we implement a VQA which supports French Question and French Answer. Comparing with the previous model, we change original word-embedding model into a bilingual model which supports English and French.

When English question is fed in, we use monolingual model, which support English, in bilingual model to embed the question in vectors and feed these vectors into model together with image feature vector.

When French question is fed in, we use monolingual model, which support French, in bilingual model to embed the question in vectors and feed these vectors into model together with image feature vector.

The reason we use this structure is that we suppose the questions with same meaning in English and French should have similar vector representations.

6.2 BILINGUAL WORD EMBEDDING

6.2.1 Bilingual Model:

Bilingual Model contains two monolingual models that can do word embedding in two languages and a parameter alpha [9].

6.2.2 Monolingual Model:

Monolingual Model contains a Huffman tree that stores its own vocabulary and two matrix whose sizes are (size of vocabulary) times word embedding dimension size and represents input weights and output weights. The word embedding of a word, which is in vocabulary, can be calculated by using input weights and output weights. The meanings and usages of input weights and output weights can be found at [10]

6.2.3 Overall Training implementation:

When creating a bilingual model with some configurations, this model automatically creates two monolingual models with the same configurations. After these two monolingual models have been created, these two models initialize their skipgram models based on the user input and input dataset, which are two articles with same meaning in two different languages. The initialization process will be described below. After all parameter is initialized, the bilingual model starts training. The training process is a loop that takes two individual sentences synchronously from the dataset and training the monolingual models with these two sentence.

6.2.4 Initialization Process:

Bilingual model first initializes the parameter alpha based on the user input. Then the internal monolingual models initialize the input weight and output weight using random variable and the range of each entry in input weight and output weight is from -0.5 to 0.5.

After that, the monolingual models read the words from dataset and create a Huffman trees that contains its vocabulary set. The structure of this Huffman tree also form a structure of neural network. All nodes except leaf nodes serves as a neuron and contains a weight.

6.2.5 Sentence Training:

In this sentence training process, the input two sentences are split into words. Then we perform uniform alignment on these words. The reason why we perform uniform alignment is that the numbers of words in these two sentences may be different and i^{th} word in sentence 1 may not be mapped into i^{th} word in sentence 2 semantically. Therefore, we need to find out the mapping between the words in these two sentences based on the lengths of these two sentences. After the uniform alignment is done, we train model using word pairs based on the word mapping generated in uniform alignment.

6.2.6 Word training:

Word training part contains four training parts. Part 1 and Part 2 are training only involve individual monolingual model. Part 3 and Part 3 are training involving both two monolingual models.

Part 1: source monolingual model trains its own word-embedding model.

Part 2: target monolingual model trains its own word-embedding model.

Part 3: set the source monolingual model as source model and target monolingual model as target model, and then do training.

Part 4: set the target monolingual model as source model and source monolingual model as target model, and then do training.

6.3 PREDICT

As we support two languages in our VQA model, we introduce a variable called “language”, which indicate what language we use in prediction.

When variable “language” is “English”, we feed the question into the target language model in bilingual word-embedding model, which interpret English. Then we concatenate the vector representation of this question with the image vector, which is produced by VGG16 model, and feed this combination into our VQA model. We will found the word whose vector representation is the nearest one to the vector output by VQA model through target language model in bilingual word-embedding model and consider this word as the final answer.

When variable “language” is “French”, the prediction procedure is similar to the procedure describe above except that we replace the target language model by source language model.

Chapter 7: Implementation of Video Question Answering

7.1 Overview

As we mentioned in previous, we will first apply the scene detector to the target video and extract the key scene based on the content change of the frame. Next, using these key scenes to predict the answer of the question.

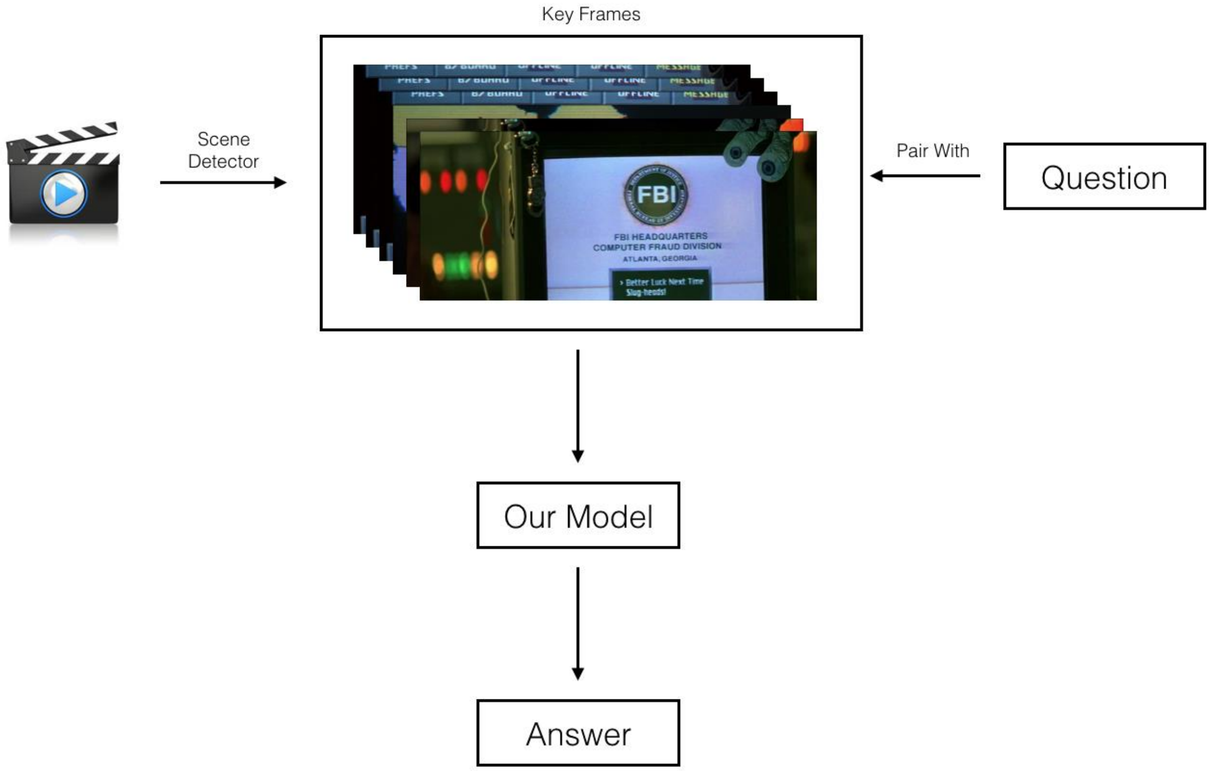


Figure 78: Diagram of Video Question Answering

7.2 Scene Detector

7.2.1 Introduction

The scene detect algorithms can detect the scene changes in a video and cut the video into separate frames. Since if there are just little difference between several frames, the information inside these frames are also almost similar. We can extract the frames which change suddenly and use them to represent the input video.

We use an open source scene detector whose name is PySceneDetect. It is a free command line tools and there are several detection algorithms available, e.g., threshold based fade in-out detection algorithms, advanced content aware fast-out detection algorithm [8].

i) Content-Aware Fast-Out Detection Algorithm

The content-aware detector detects the frames where there is huge difference between the two neighbor frames (exceed the threshold value). This algorithm can find the abrupt changes (jump cut in file) and the scene changes in the video.

ii) Threshold Based Fade In-Out Detection Algorithm

The threshold based fade in-out detector is the most traditional method to find the key scene in the video. It works by comparing the intensity or brightness of current frame with the threshold, if the average value of the R, G, B values of all pixels exceeds the threshold, this frame is regarded as a key frame.

7.2.2 Example

Let use a short video to present the function of PySceneDetect. The website of these video is https://www.youtube.com/watch?v=OMgIPnCnlbQ (movie of James Bond, GoldenEye (Copyright © 1995 MGM)).

The command is as following:

- Content-Aware Detection:

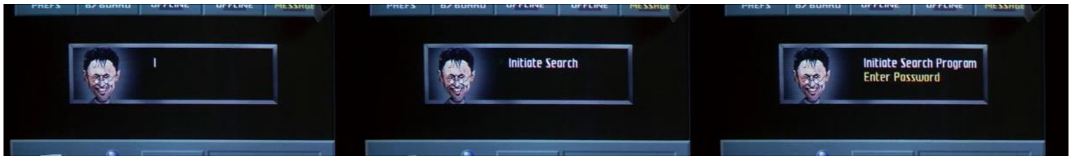
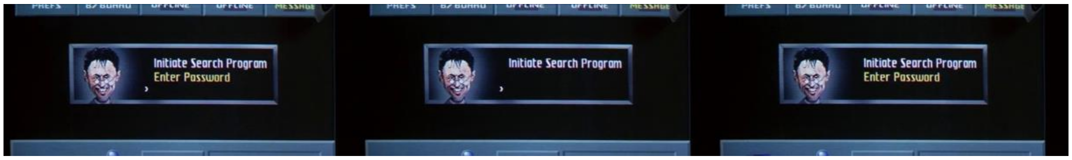
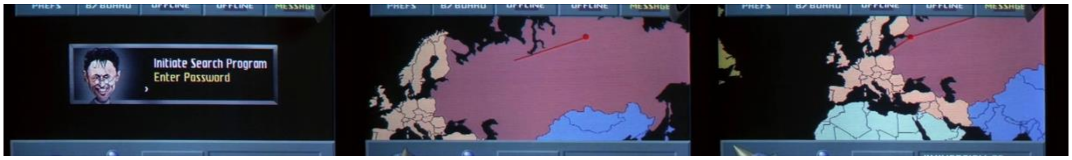
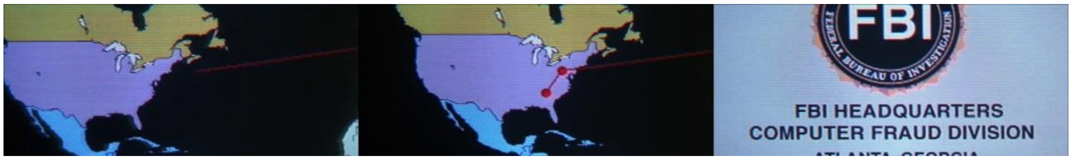
```
scenedetect -i goldeneye.mp4 -d content -t 20
```

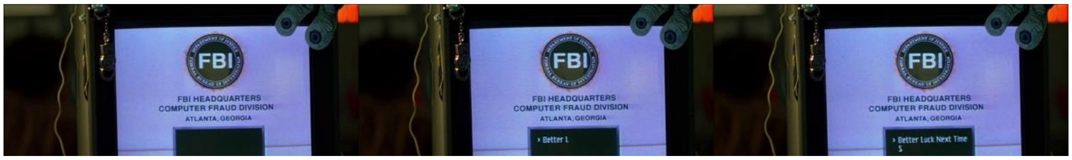
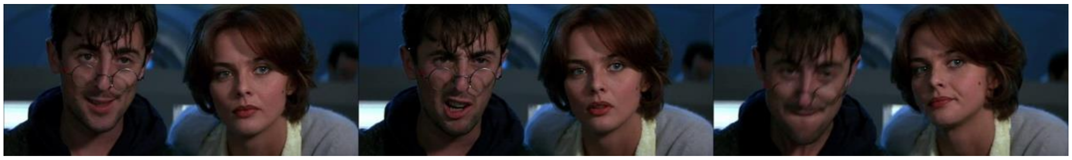
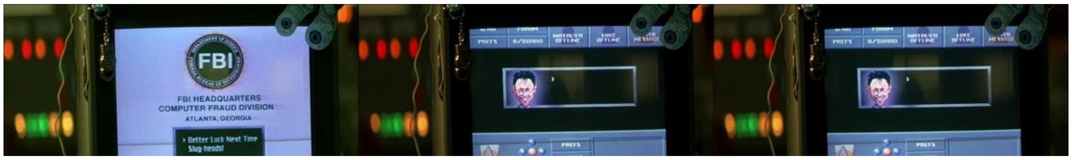
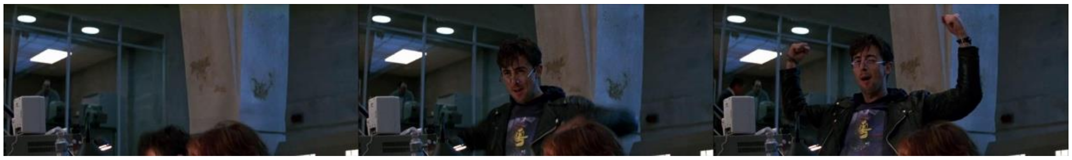
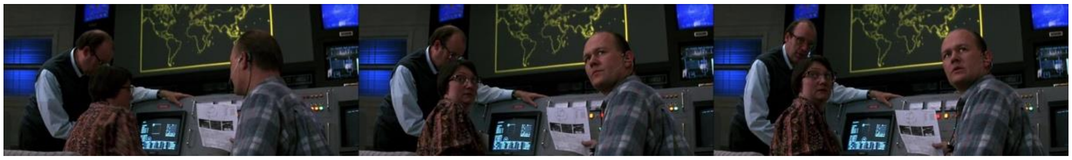
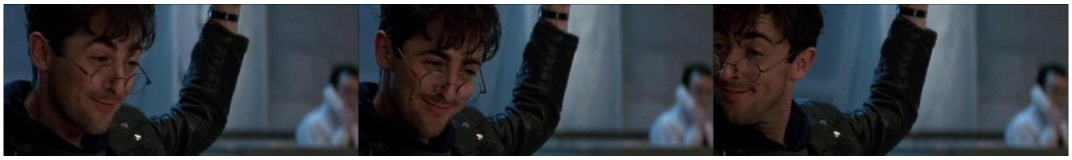
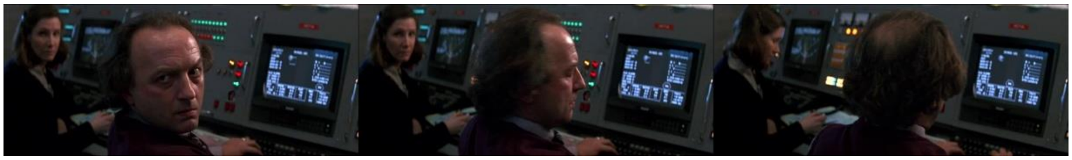
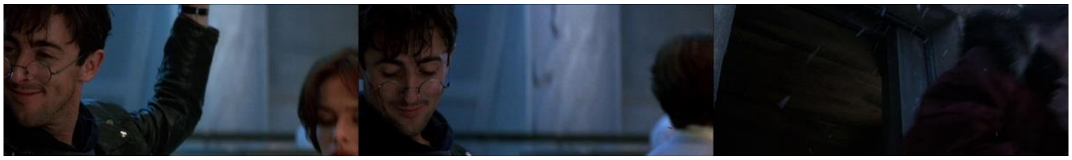
- Threshold-Based Detection Mode:

```
scenedetect -i goldeneye.mp4 -d threshold -t 20
```

The detect result is as following:

Scene Number	Scene Preview
1	

2	 Initiate Search Initiate Search Program Enter Password
3	
4	 Initiate Search Program Enter Password Initiate Search Program Initiate Search Program Enter Password
5	
6	 Initiate Search Program Enter Password
7	
8	 FBI FEDERAL BUREAU OF INVESTIGATION FBI HEADQUARTERS COMPUTER FRAUD DIVISION ATLANTA GEORGIA
9	

10	 FBI FBI HEADQUARTERS COMPUTER FRAUD DIVISION ATLANTA, GEORGIA › Better L › Better Luck Next Time
11	
12	 FBI FBI HEADQUARTERS COMPUTER FRAUD DIVISION ATLANTA, GEORGIA › Better Luck Next Time Slug-heads! PREFS
13	
14	
15	
16	
17	

18	
19	
20	

This 83 seconds video is divide into about 20 frames according to the content changes between the neighbor frames. Then we feed the 20 frames with the question into our model and find the optimal answer for this question.

7.3 PREDICT

For each key scene, we can obtain a series of answer with probabilities. Then we combine all the answers and use the answer that has highest probability as the final answer.

Chapter 8: TRAINING PROCESS

8.1 DATASET

8.1.1 Visual Question Answering Dataset

Since we need a large amount of data to train a model, and the data set should contain image, question and answer pairs.

The training and testing dataset we used in our project is provided by the organizers of Visual Question Answering challenge which is sponsored by MS COCO website. This dataset contains three subsets which are train set, validation set and test set. Inside the dataset, there are two types of questions and first one is Open-End questions and another one is Multiple-Choice questions. The detail is as following (16ht):

Real Images:

- 82783 MS COCO training images
- 40504 MS COCO validation images
- 81434 MS COCO testing images
- 248349 questions for training
- 121,512 questions for validation

- 244,302 questions for testing (3 per image)
- 2483490 answers for training
- 1,215,120 answers for validation (10 per question)

Abstract Images:

- 20000 training images
- 10,000 validation images
- 20,000 MS COCO testing images
- 60000 questions for training
- 30000 questions for validation
- 60000 questions for testing (3 per image)
- 600000 answers for training
- 300000 answers for validation (10 per question)

8.1.2 Bilingual Word Embedding

Since we aim to train a bilingual word-embedding model, we need to dataset that is articles with same meaning in English and French.

The dataset we use is called European Parliament Proceedings Parallel Corpus. The Europarl parallel corpus is extracted from the proceedings of the European Parliament. It includes versions in 21 European languages. Since we are only interest in English and French, we only use English version and French version.

- English version dataset contains 2,218,201 sentences and 53,974,751 words.
- French version dataset contains 2,190,579 sentences and 54,202,850 words.

8.2 Training Process of Neural Reasoning

In our model, we choose Adam optimizer as our optimizer and train 3 models using 3 different learning rate.

i) Learning Rate: 0.002

Accuracy:

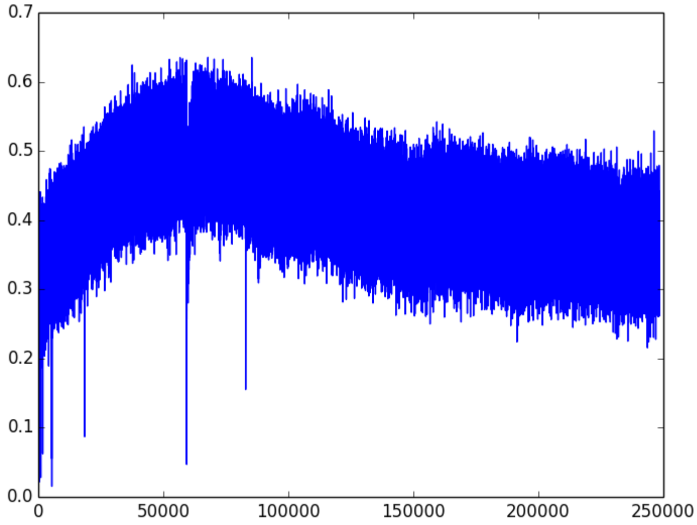


Figure 79: Accuracy of training when learning rate is 0.002

Loss:

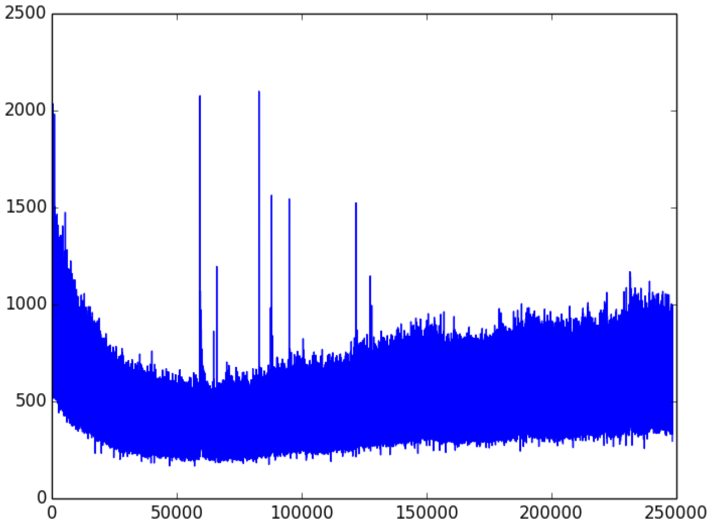


Figure 80: Loss of training when learning rate is 0.002

When learning rate is 0.002, the final accuracy is around 50% and the final loss is around 350.

ii) Learning rate: 0.001

Accuracy:

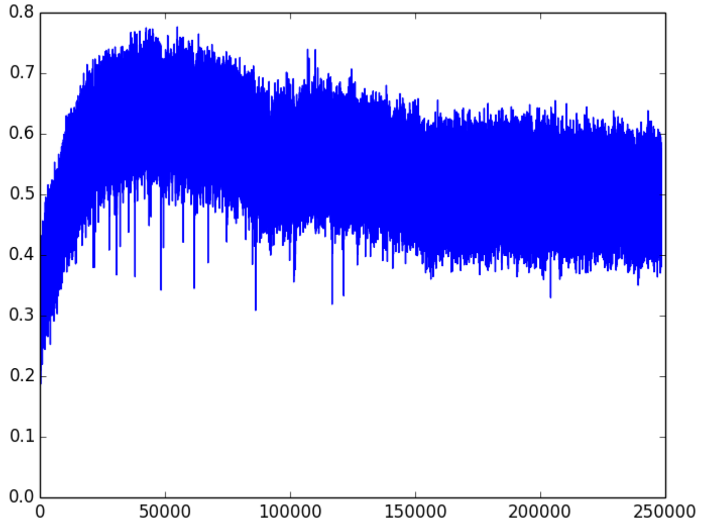


Figure 81: Accuracy of Training when learning rate is 0.001

Loss:

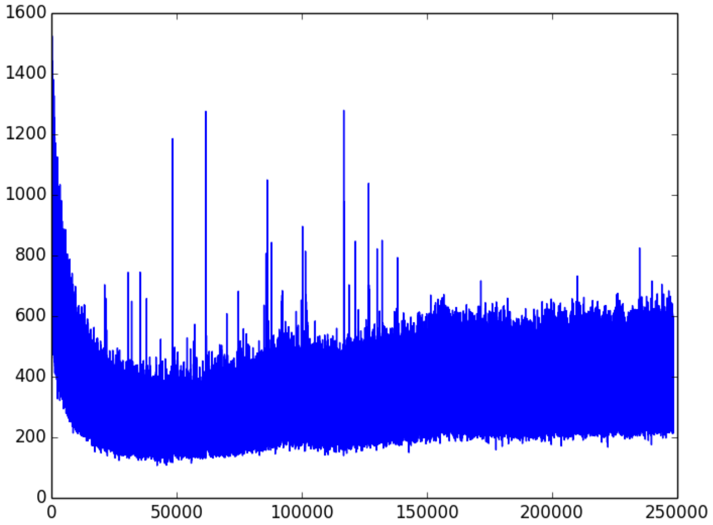


Figure 82: Loss of training when learning rate is 0.001

When learning rate is 0.001, the final accuracy of training is around 60% and loss of training is around 200.

iii) Learning rate: 0.0005

Accuracy:

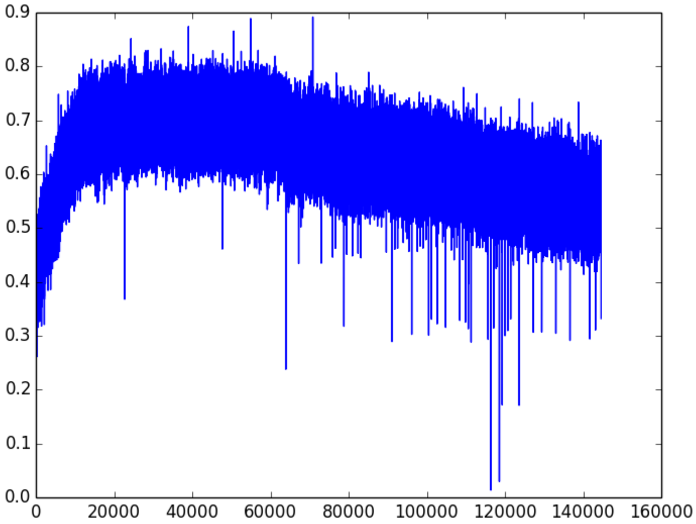


Figure 83: Accuracy of Training when learning rate is 0.0005

Loss:

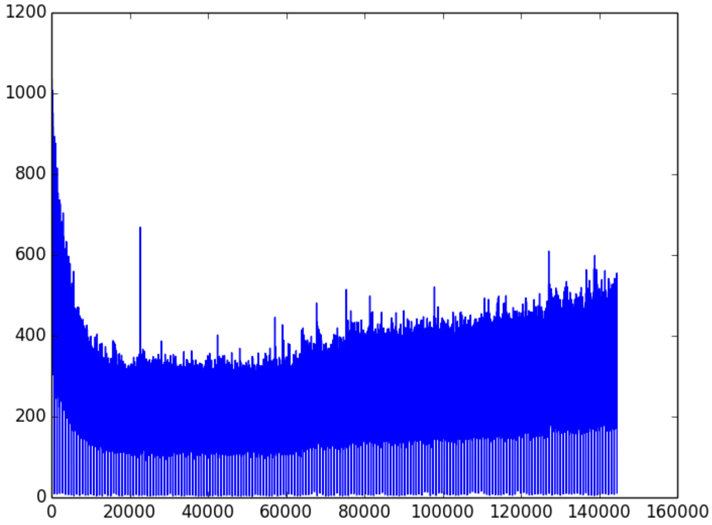


Figure 84: Accuracy of Training when learning rate is 0.0005

When learning rate is 0.0005, the final accuracy of training is around 65% and loss of training is around 250.

By comparing the result of these three models, we can find that the model with learning rate equals to 0.001 and model with learning rate equals to 0.0005 is better than the model with learning rate equals to 0.002 based on the statistics result. Practically, we find that the model with learning rate equals to 0.0005 is the best one.

8.3 Training Process of French Question Answering

In our model, we choose Adam optimizer as our optimizer and train 3 models using 3 different learning rate.

i) Learning Rate: 0.002

Accuracy:

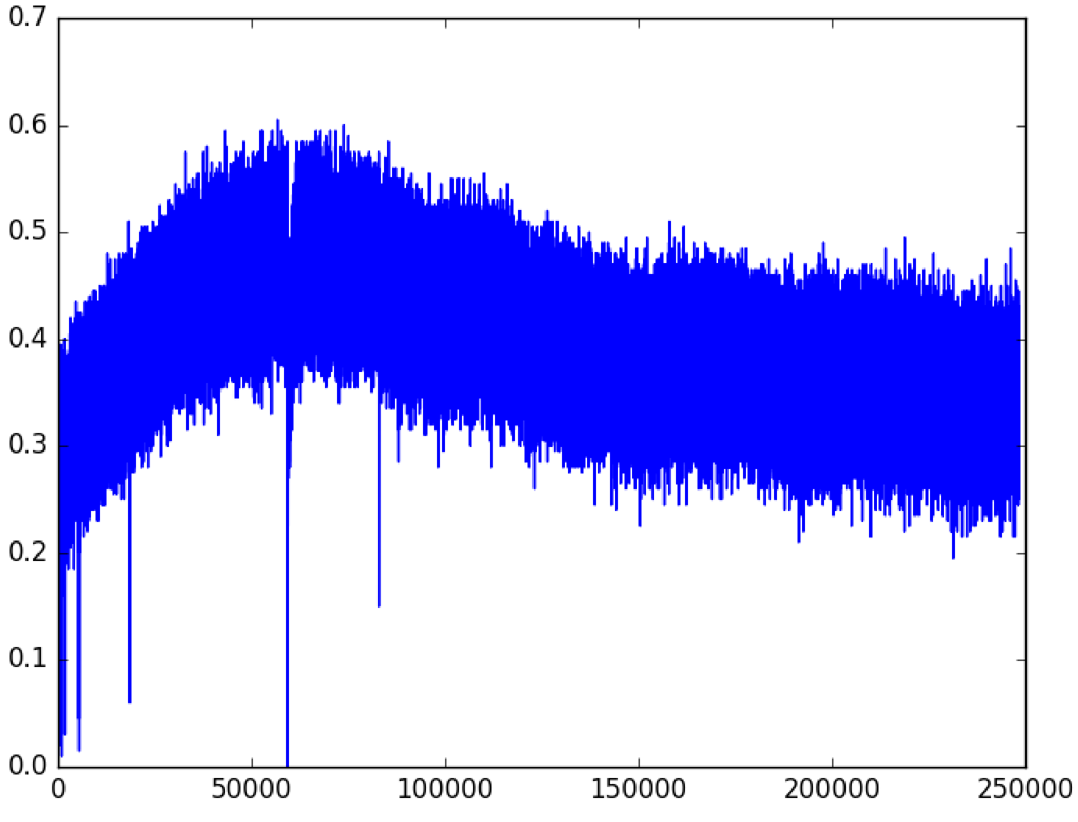


Figure 85: Accuracy of training when learning rate is 0.002

Loss:

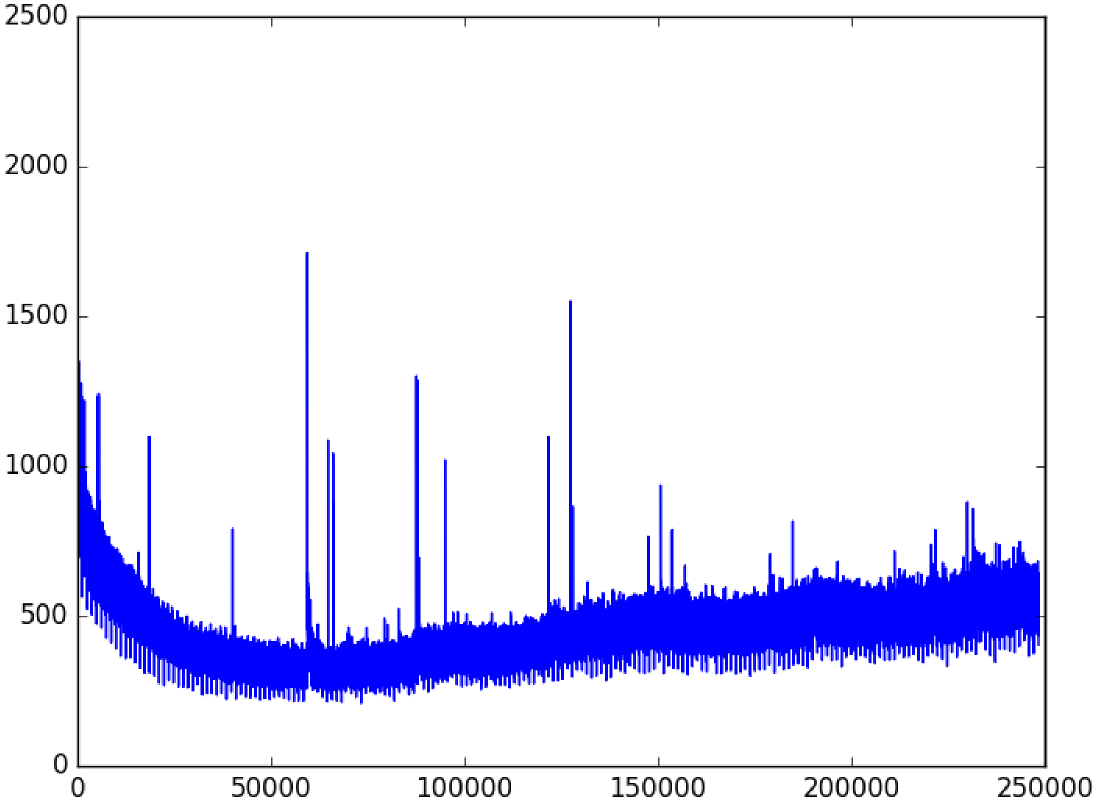


Figure 86: Loss of training when learning rate is 0.002

When learning rate is 0.002, the final accuracy is around 40% and the final loss is around 500.

ii) Learning rate: 0.001

Accuracy:

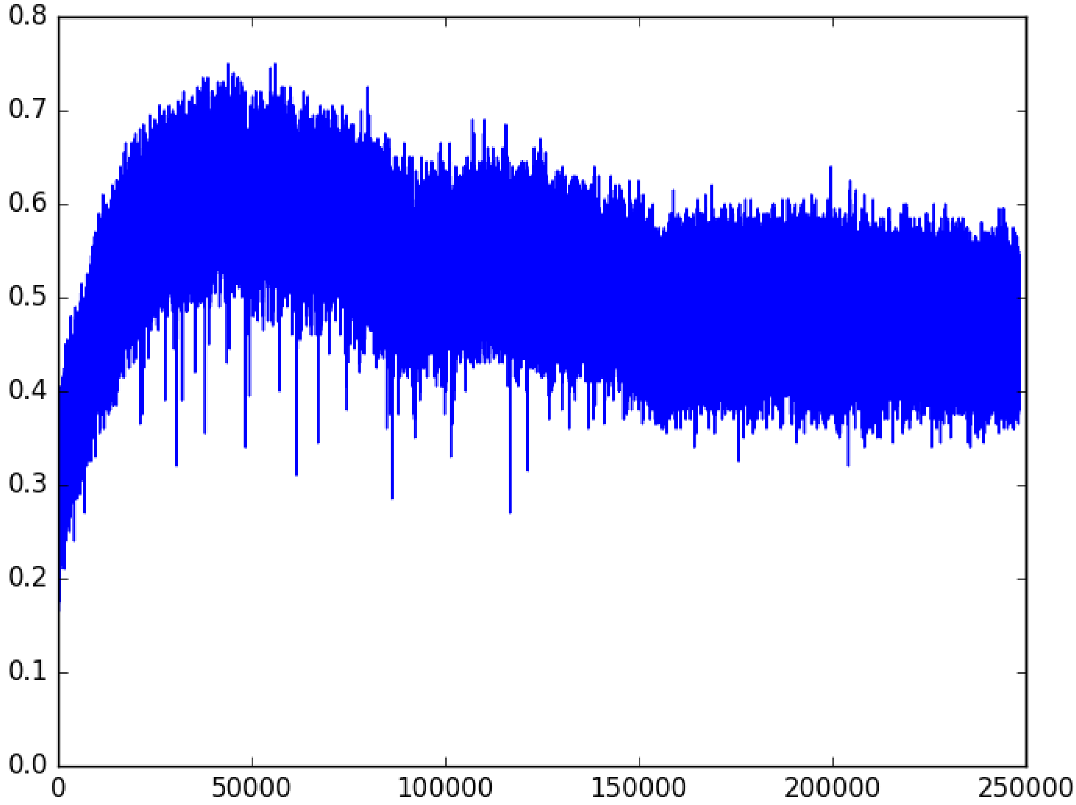


Figure 87: Accuracy of Training when learning rate is 0.001

Loss:

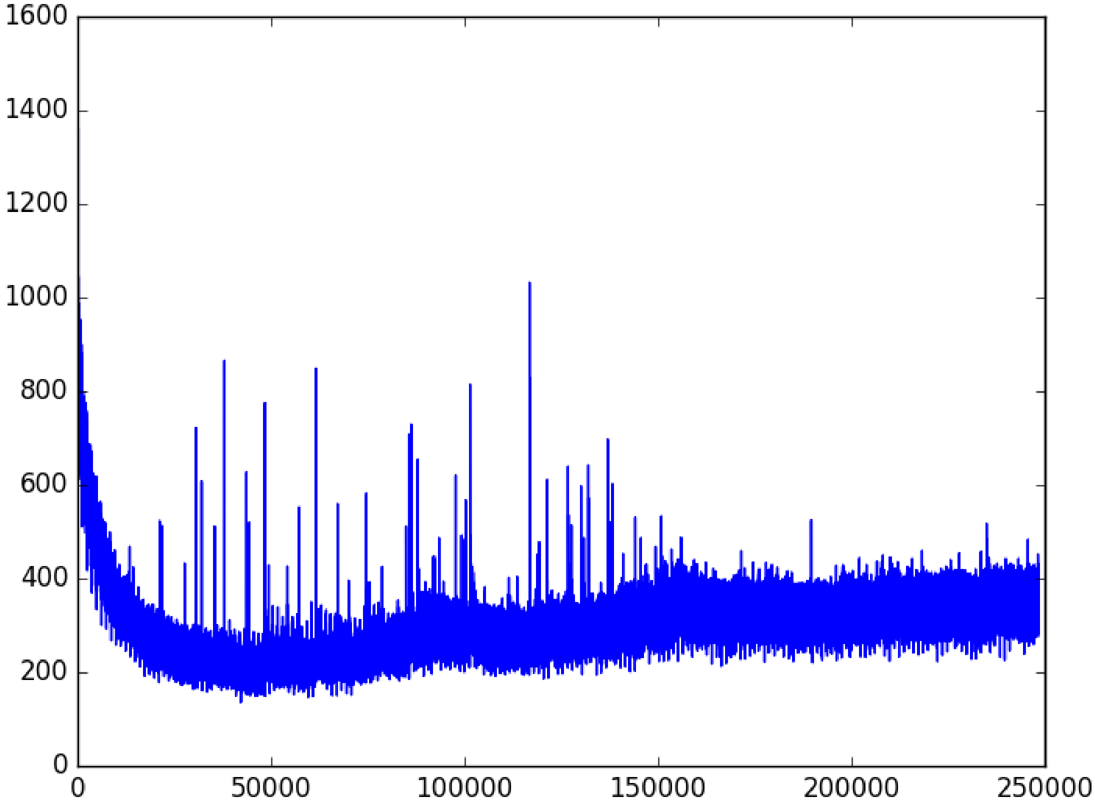


Figure 88: Loss of training when learning rate is 0.001

When learning rate is 0.001, the final accuracy of training is around 50% and loss of training is around 300.

iii) Learning rate: 0.0005

Accuracy:

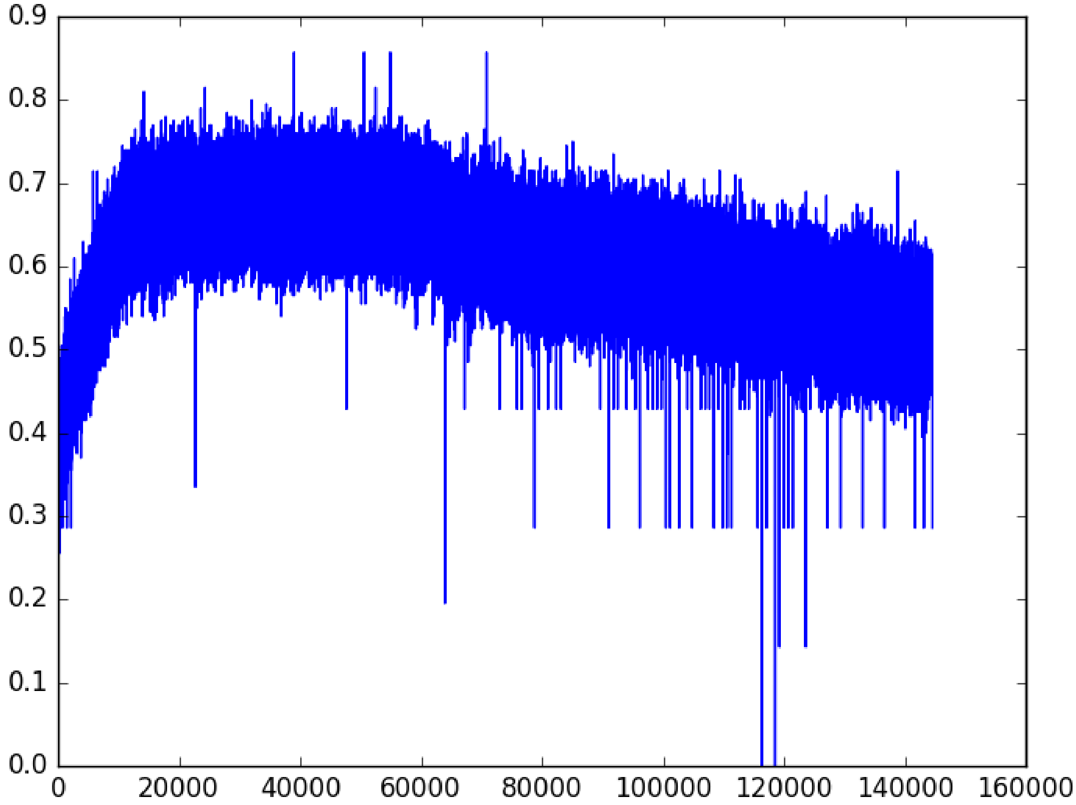


Figure 89: Accuracy of Training when learning rate is 0.0005

Loss:

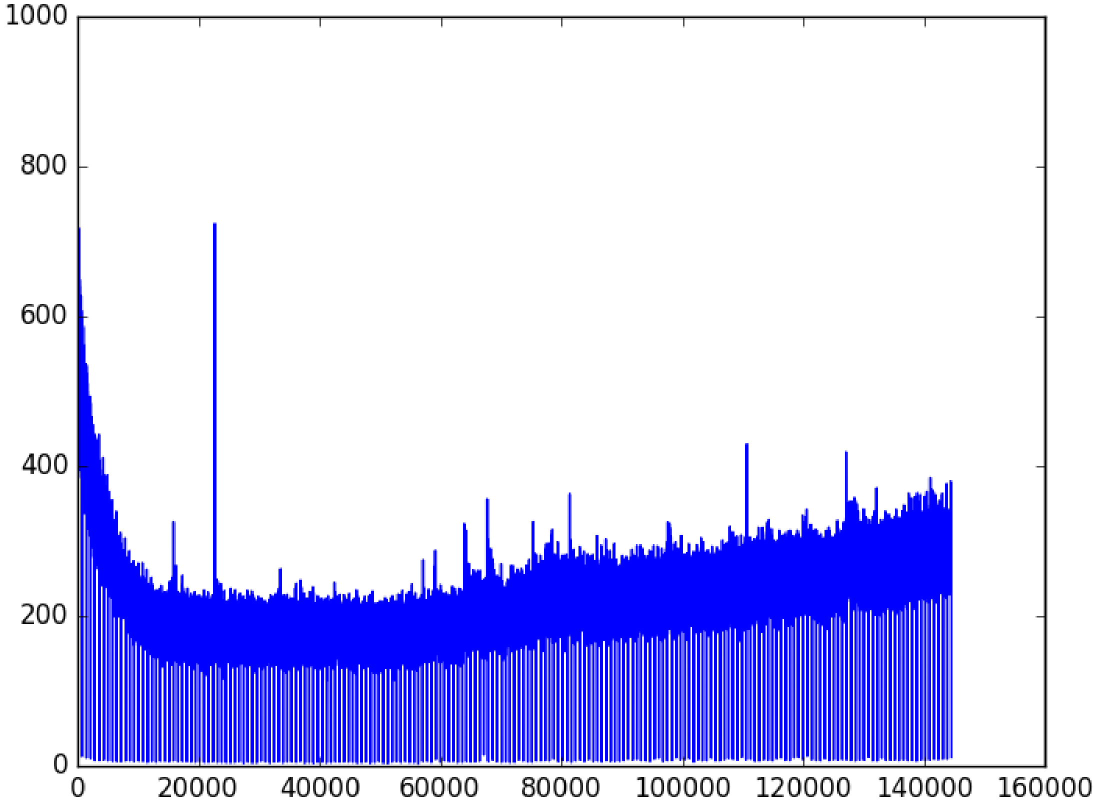


Figure 90: Accuracy of Training when learning rate is 0.0005

When learning rate is 0.0005, the final accuracy of training is around 50% and loss of training is around 300.

By comparing the result of these three models, we can find that the model with learning rate equals to 0.001 and model with learning rate equals to 0.0005 is better than the model with learning rate equals to 0.002 based on the statistics result. Practically, we find that the model with learning rate equals to 0.0005 is the best one.

Chapter 9: RESULT

9.1 VISUAL QUESTION ANSWERING (REASONER VERSION)

9.1.1 Evaluation

i) Evaluation Metric

The evaluation metric we used is same as what we used in the last semester.

We use the evaluation code which offer by the Visual Question Answering challenge organizers. In this evaluation metric, the accuracy of machine will be averaged over all 10 choose 9 sets of human annotators to enhance the robustness of accuracy.

The accuracy formula is as follow:

$$Acc(ans) = \min\{\frac{\#\ humans\ that\ said\ ans}{3}, 1\}$$

ii) Evaluation Dataset

The dataset we used for evaluation is mentioned in Chapter Methodology.

9.1.2 Accuracy

i) Detail Accuracy

	Test Set			
Model	Yes / No	Number	Other	Overall
Baseline All Yes	70.97	0.36	1.21	29.88
Baseline Prior Per Question Type	71.40	34.90	8.96	37.47
Baseline Nearest Neighbor	71.89	24.23	22.10	42.85
Model in Last Semester	74.62	31.76	31.32	45.87
Current Model	80.62	35.78	40.02	48.53

9.1.3 Sample Result

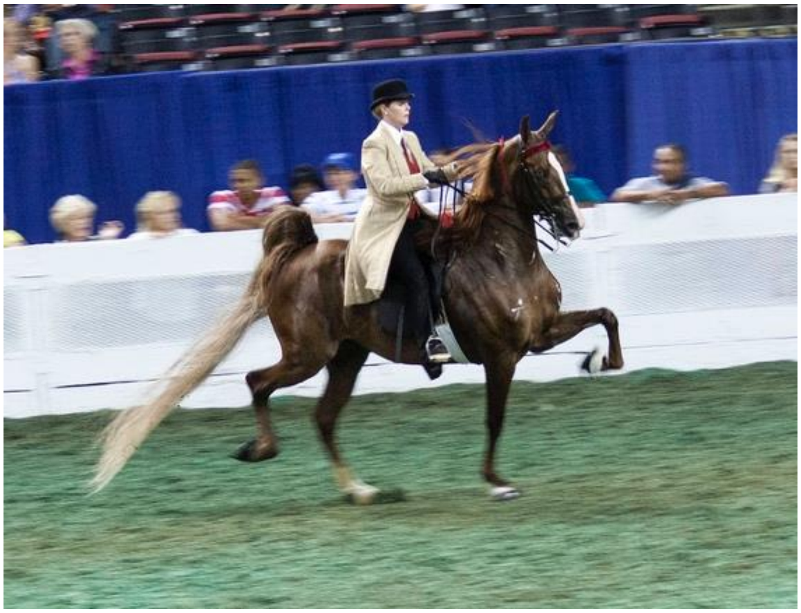


Figure 91: Sample 1

Question: What animal is this?

Top 5 Answers: horse, zebra, elephant, dog, cow

Question: What color is this animal?

Top 5 Answers: brown, white, black, blue, green

Question: What are they doing now?

Top 5 Answers: riding, talking, eating, standing, skiing

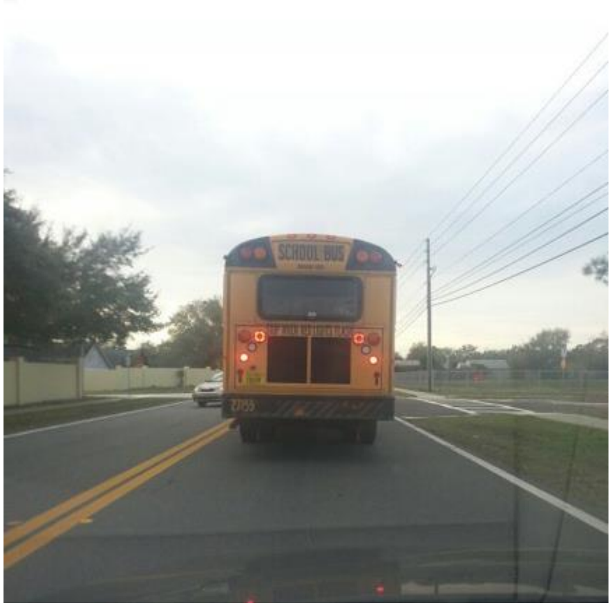


Figure 92: Sample 2

Question: What is this that in the front?

Top 5 Answers: truck, train, building, bikes, teddy bears

Question: How about the weather now?

Top 5 Answers: cloudy, cold, rainy, wet, evening

Question: How many cars are in there?

Top 5 Answers: 2, 3, 1, 5, 4

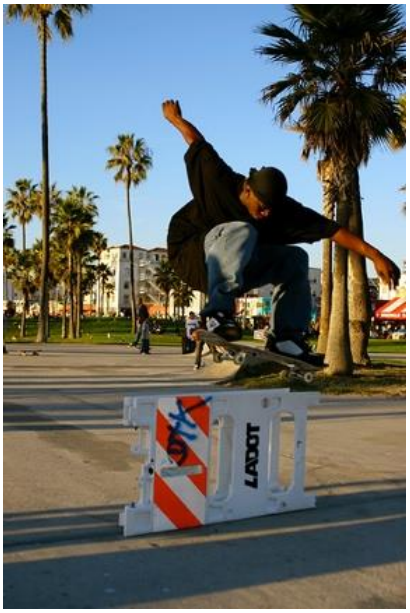


Figure 93: Sample 3

Question: What is this man doing?

Top 5 Answer: skateboarding, jumping, snowboarding, sitting, taking picture

Question: How about the weather?

Top 5 Answer: cloudy, sunny, rainy, overcast, clear

Question: What is the gender of the person?

Top 5 Answer: male, boy, female, child, women

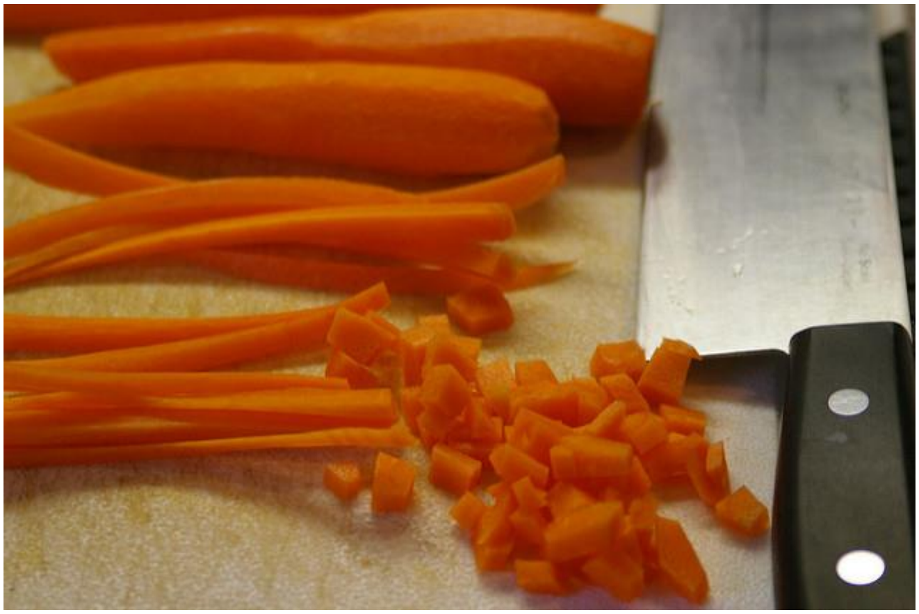


Figure 94: Sample 4

Question: What are sitting on the counter in different stages of cutting with a knife?

Top 5 Answers: carrot, chocolate, glass, bread, fries

Question: What is the color of this vegetables?

Top 5 Answer: orange, yellow, green, blue, brown

Question: What is the name of this vegetable?

Top 5 Answer: carrot, parsley, apple, bananas, carrots

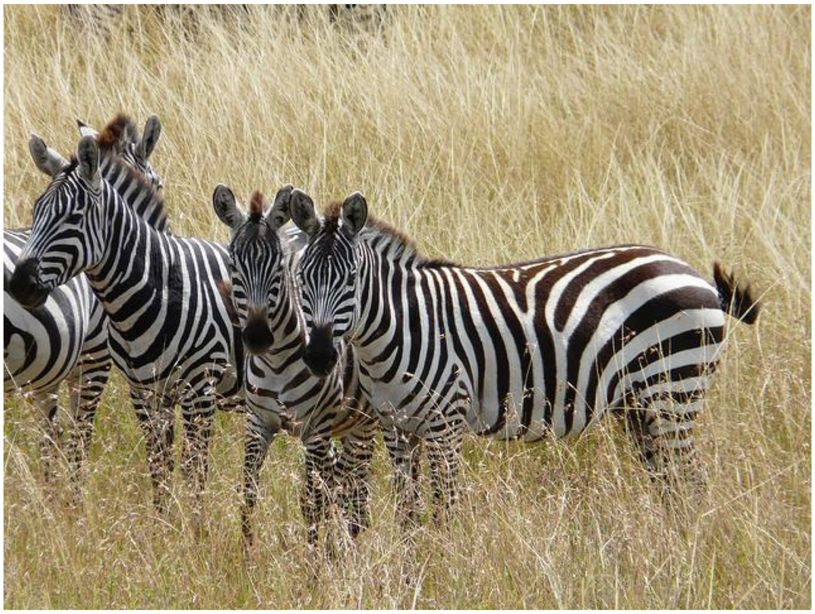


Figure 95: Sample 5

Question: What is the color of this animal?

Top 5 Answer: black, white, brown, black, gray

Question: What is this animal?

Top 5 Answer: zebra, giraffe, horse, cow, zebras

Question: How many animals are there?

Top 5 Answer: 3, 4, 2, 1, 5

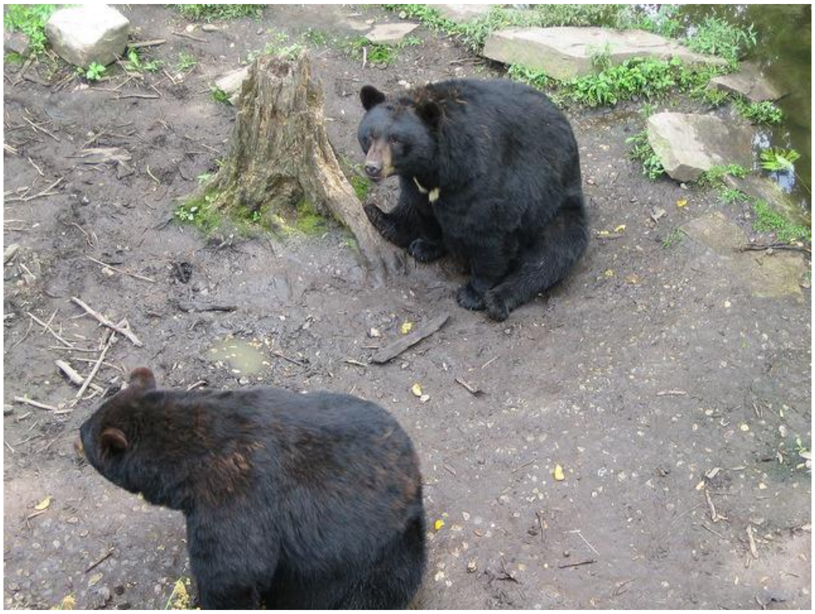


Figure 96: Sample 6

Question: How many animals are there?

Top 5 Answer: 2, 3, 1, 5, 4

Question: What is the name of this animal?

Top 5 Answer: bear, bird, elephant, cat, dog

Question: what are sitting down on the ground?

Top 5 Answer: bear, snow, stick, bird, cow

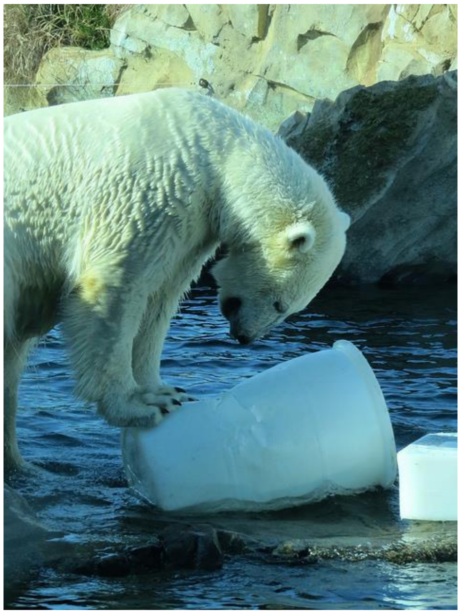


Figure 97: Sample 7

Question: what is playing with the large piece of ice?

Top 5 Answer: cow, bird, surfboard, water, boat

Question: What is the color of this animal?

Top 5 Answer: white, brown, black, blue, gray

Question: How many animals are there in the image?

Top 5 Answer: 2, 8, 1, 3, 6

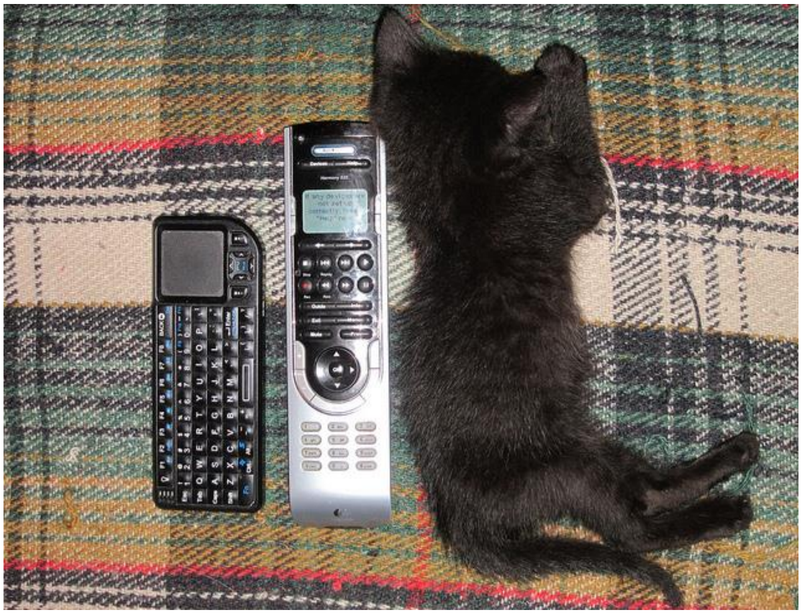


Figure 98: Sample 8

Question: How many animals are there in the image?

Top 5 Answer: 3, 1, 2, 4, 5

Question: What is the name of object that in the middle?

Top 5 Answer: remote, toy, laptop, cat, books

Question: What is the name of object that in the left?

Top 5 Answer: remote, keyboard, books, phone, bowl

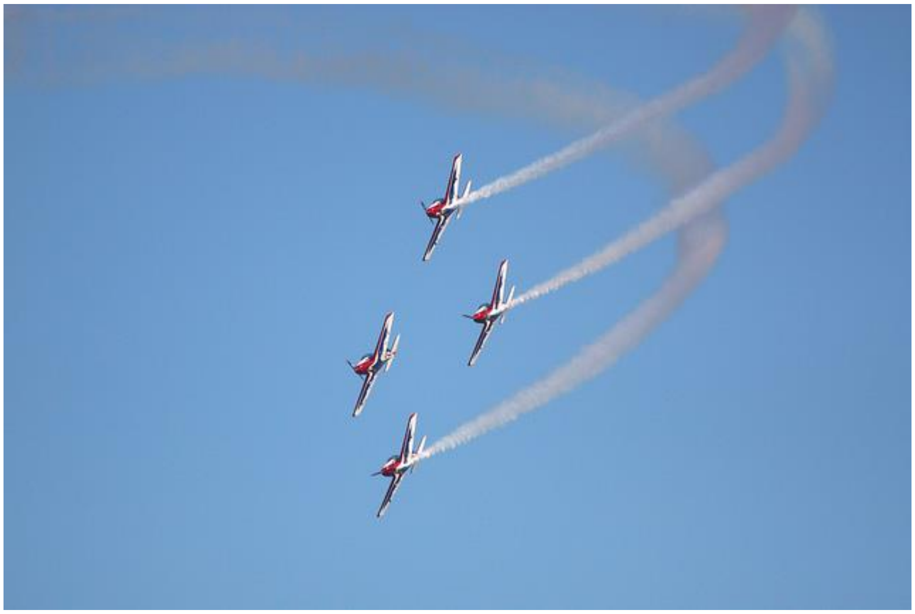


Figure 99: Sample 9

Question: what are flying through the sky?

Top 5 Answer: plane, kite, kites, airplane, birds

Question: What is the color of background?

Top 5 Answer: blue, red, yellow, orange, green

Question: How many objects in the sky?

Top 5 Answer: 8, 4, 13, 5, 1

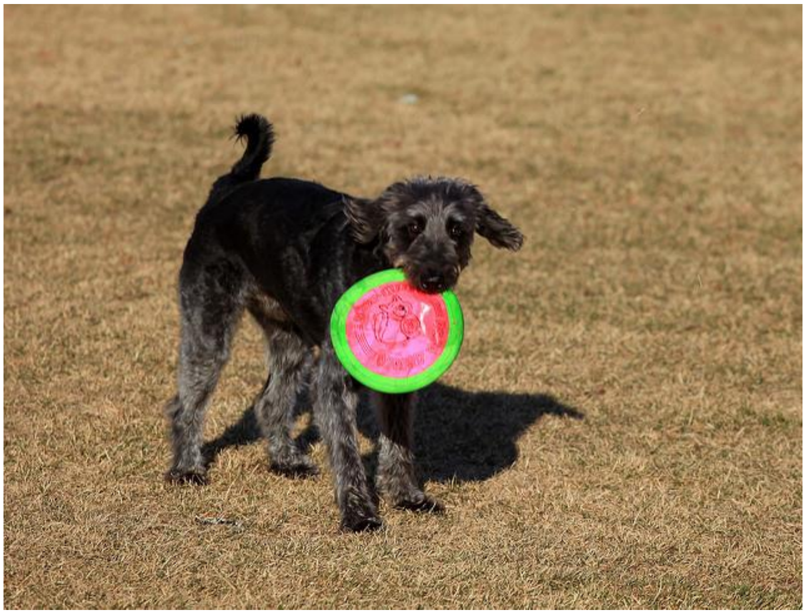


Figure 100: Sample 10

Question: what is the black dog holding?

Top 5 Answer: frisbee, carrot, bat, bowl, dog

Question: What is the color of this dog?

Top 5 Answer: black, brown, white, blue, yellow

Question: what is the color of the object that the black dog holding?

Top 5 Answer: blue, yellow, white, pink, red

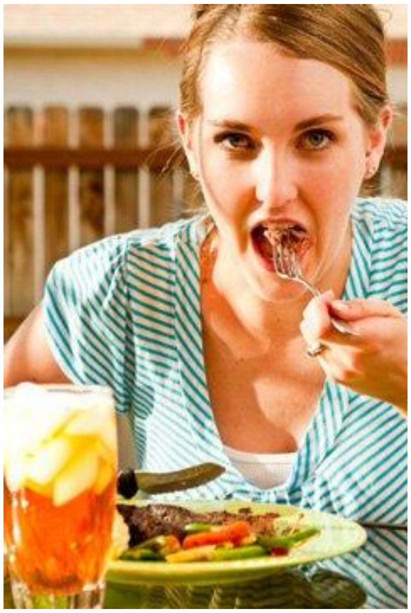


Figure 101: Sample 11

Question: What is this woman doing?

Top 5 Answer: eating, cooking, smiling, dog

Question: What is this woman holding?

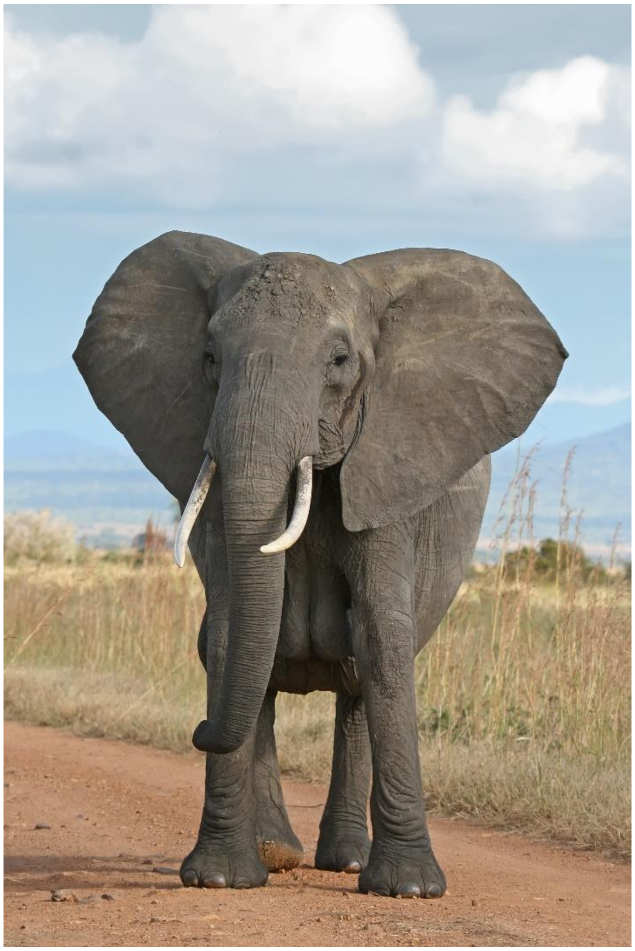
Top 5 Answer: plate, fork, apple, knife, pizza

Question: what color is this woman's eyes?

Top 5 Answer: brown, yellow, black, green, red

9.2 FRENCH QUESTION ANSWERING

i) Sample 1



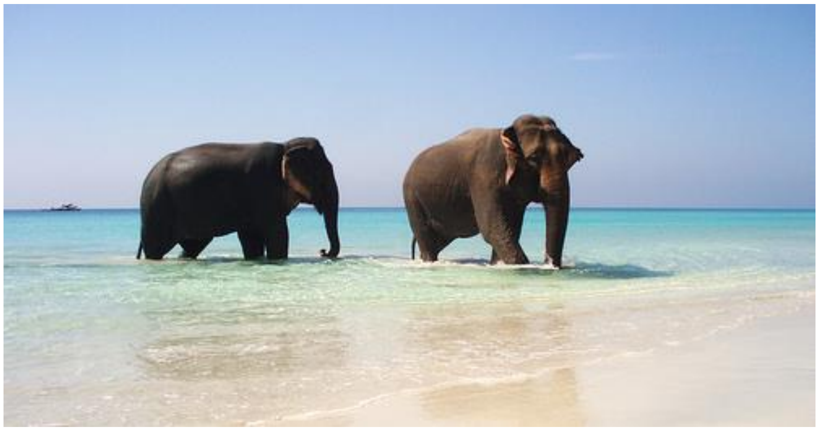
English Version Question: What animal is this?

Top 5 Answer: elephants, bear, winter, elephant, horse.

French Version Question: Quel animal est-ce?

Top 5 Answer: l'éléphant, chien, Gardant, chat, droit

ii) Sample 2



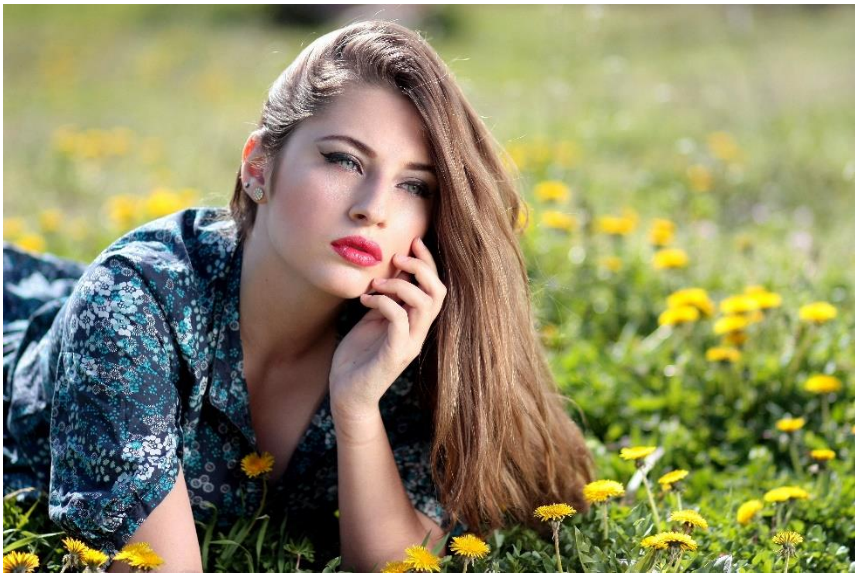
English Version Question: How many elephants are there in this picture?

Top 5 answers: 2, 3, 1, 8, 6.

French Version Question: Combien y a-t-il d'éléphants dans cette image?

Top 5 answers: 2, oui, aucane, 5, coupées

iii) Sample 3



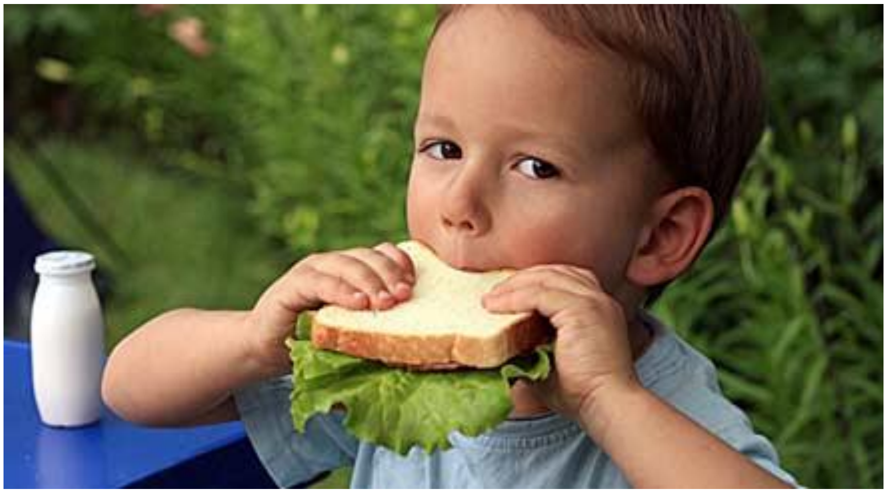
English Version Question: What is the gender of this people?

Top 5 Answers: female, male, yes, food, light.

French Version Question:

Top 5 Answers: femelle, mâle, oui, aucune, salubre.

iv) Sample 4



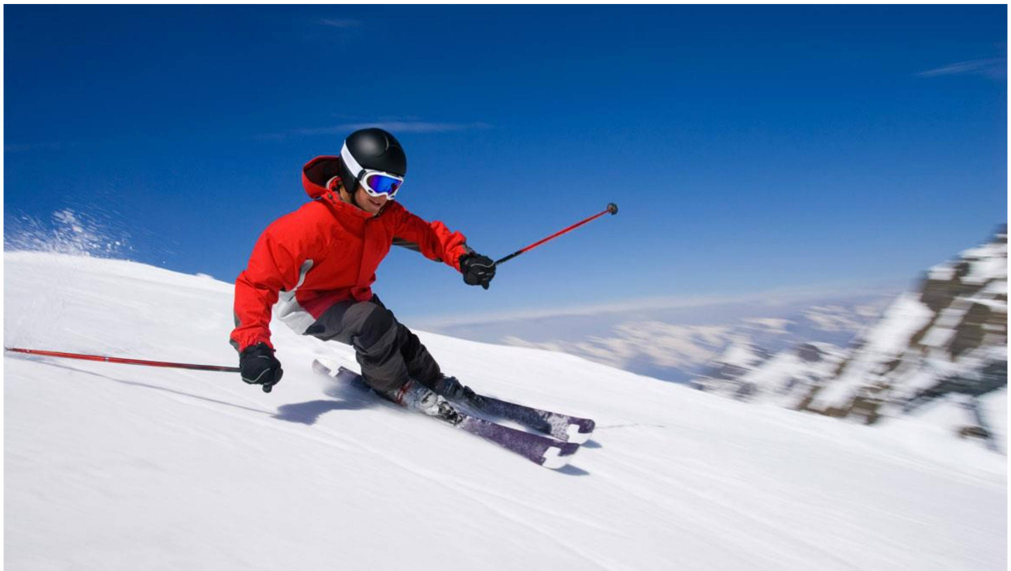
English Version Question: What is this boy doing?

Top 5 Answers: eating, standing, 2011, 309, skiing.

French Version Question: Qu’est-ce qu’il fait?

Top 5 Answers: manger, voir, oui, 2011, debout

v) Sample 5



English Version Question: What is he doing?

Top 5 Answers: skiing, surfing, sitting, standing, walking

French Version Question: Qu'est-ce qu'il fait?

Top 5 Answers: skier, surfer, debout, marcher, assis.

9.3 VIDEO QUESTION ANSWERING

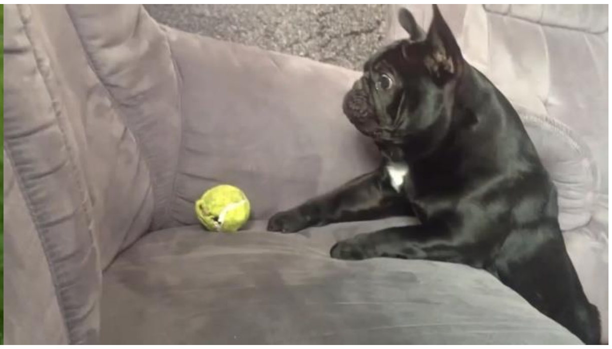
We have tried several videos and the results is as following:

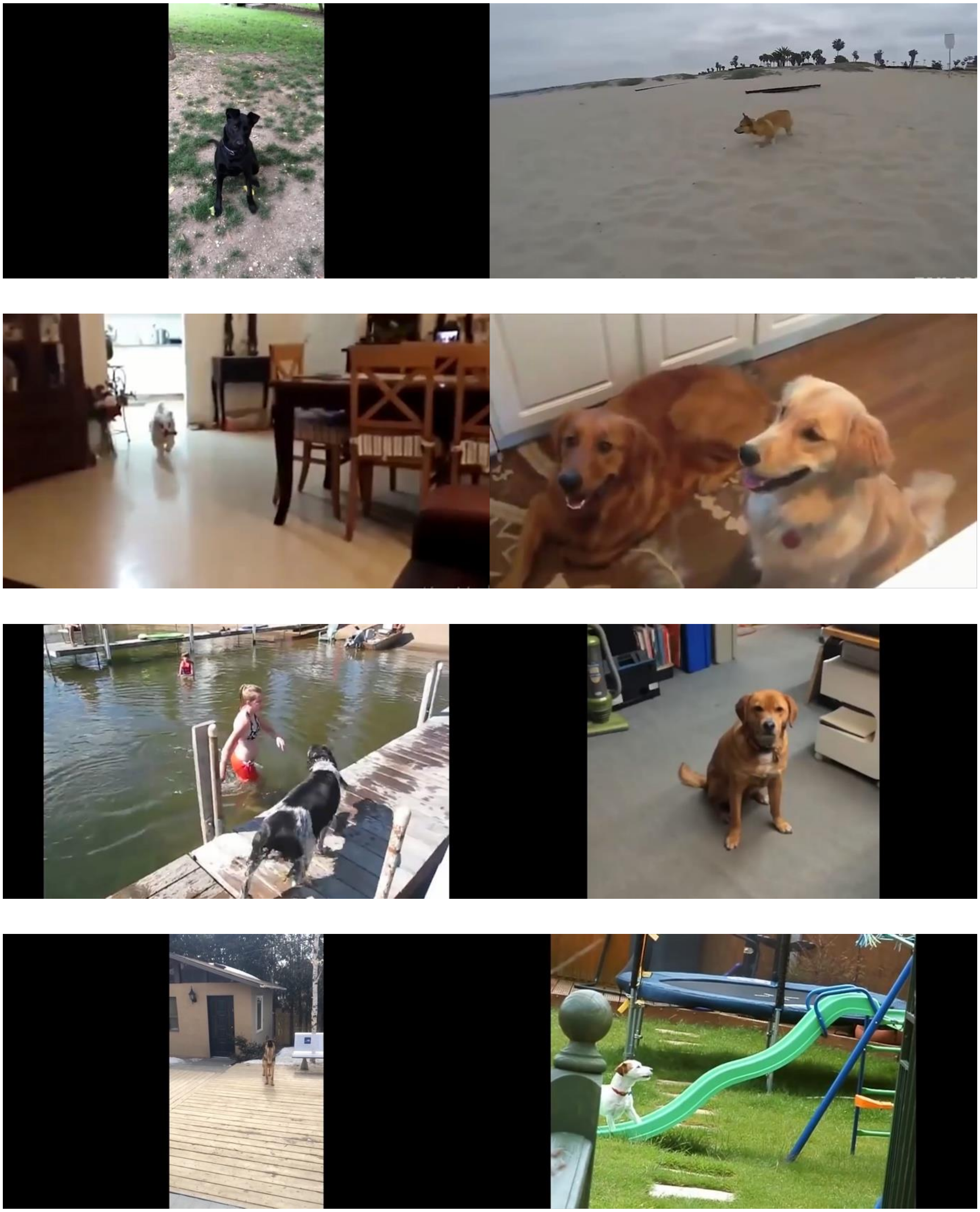
9.3.1 Video Sample 1

The URL of video sample 1 is https://www.youtube.com/watch?v=kMhw5MFYU0s. The title of this video is "Dogs Who Fail At Being Dogs".

Firstly, we apply the scene detector to the video and the extracted key scenes of these video. This video is divide into 17 key scenes and they showed as following:

i) Key Scenes





ii) Question & Answer

- Which animal is this?
  - Dog
- Which animal is it?
  - Cow
- What are they doing?
  - Posing

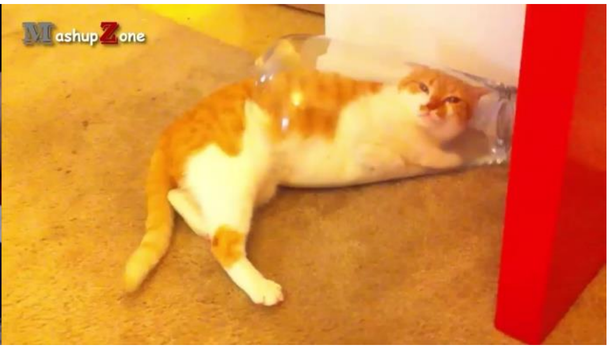
9.3.2 Video Sample 2

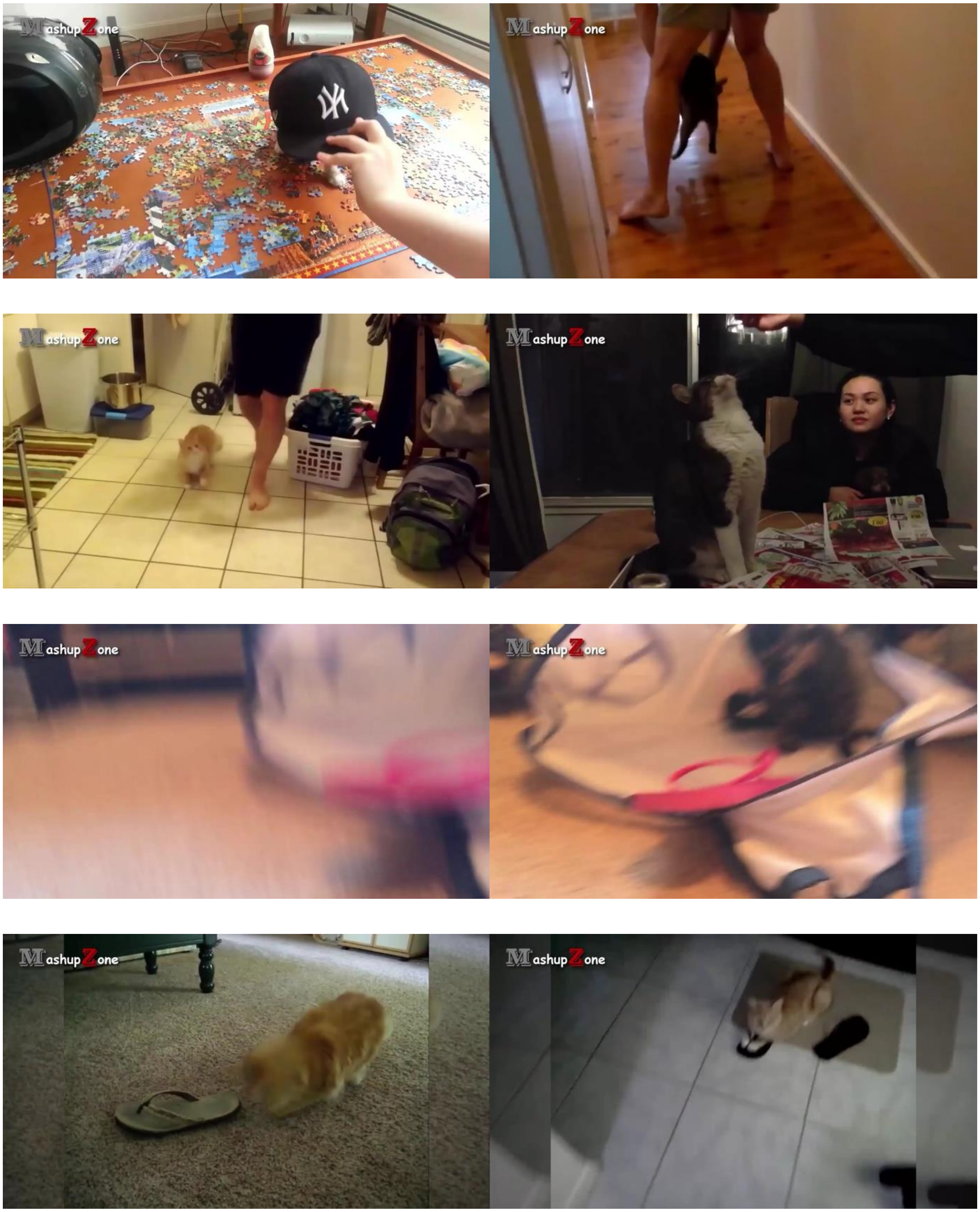
The URL of video sample 2 is https://www.youtube.com/watch?v=G8KpPw303PY. The title of this video is "Funny Cats - A Funny Cat Videos Compilation 2016 || NEW HD".

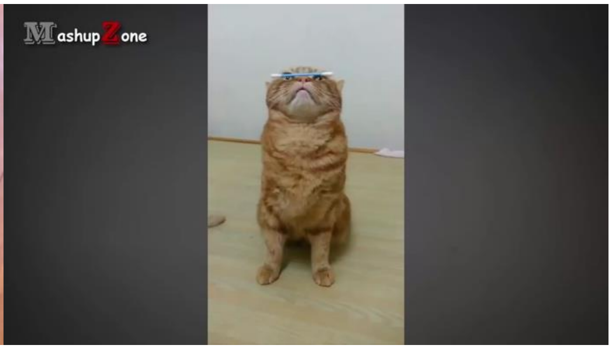
Firstly, we apply the scene detector to the video and the extracted key scenes of these video. This video is divide into 15 key scenes and they showed as following:

i) Key Scenes

MashupZone







ii) Question & Answer

- What animal is this?
  - Cat
- What are they doing?
  - Playing

Chapter 10: DISCUSSION

10.1 VISUAL QUESTION ANSWERING (REASONER VERSION)

10.1.1 Accuracy Comparison

For the same test batch, the accuracy of the model in the second semester and first semester is as following:

Model in Last Semester	74.62	31.76	31.32	45.87
Current Model	80.62	35.78	40.02	48.53

From the above data, we can see that the reasoner can help increase the accuracy of visual question answering.

Instead of converting the whole image into vector in the last semester, we not only convert the original image, but also the significant objects in the image. In this way, the information inside the image was represented richer than that in the last semester.

10.1.2 Cases Analysis

i) Case 1

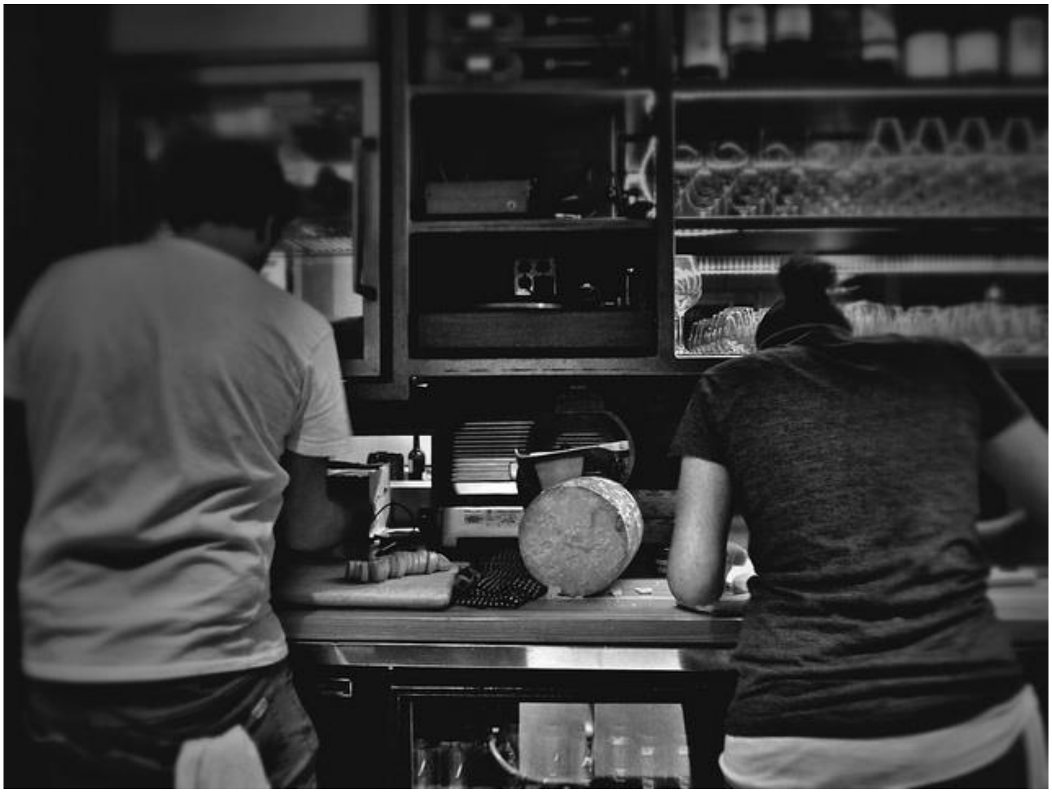


Figure 102: Sample Image 1

In this image, two people are working in the desktop. When using the model in the last semester, the whole image will feed into the neural network. Using the approach in this semester, we will apply the object localization first and the result is as following:

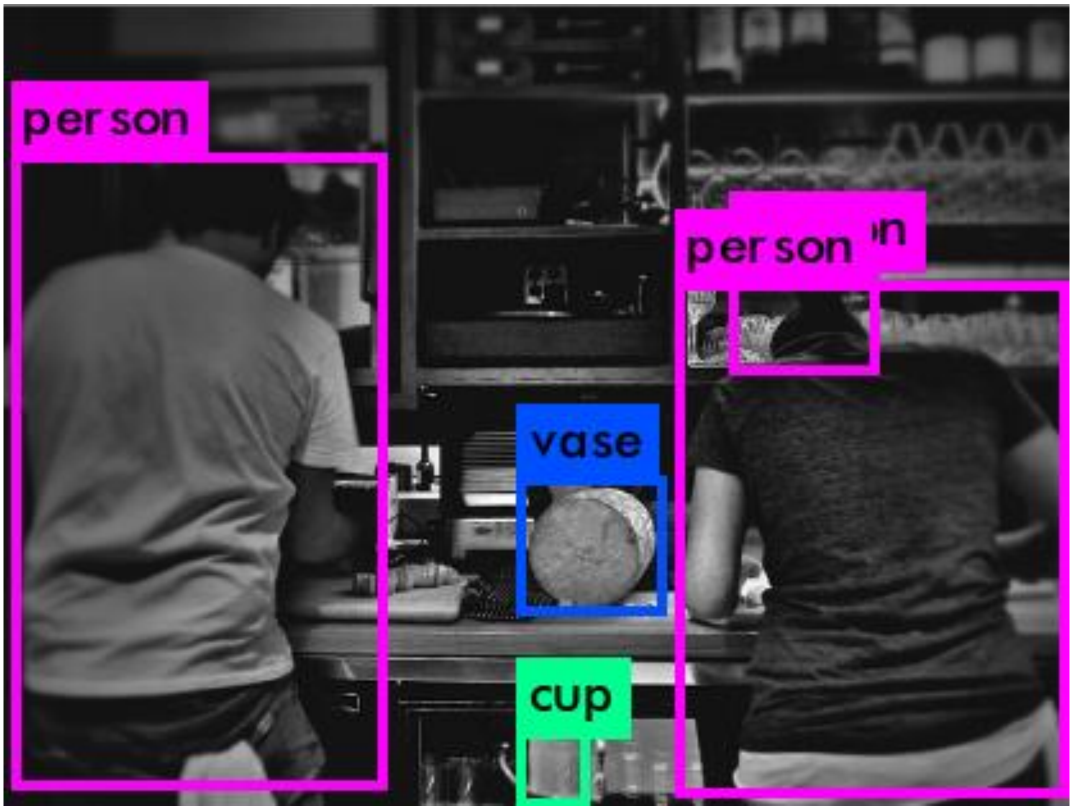


Figure 103: Image After Object Localization

The object detector can find there are two persons in the image as well as a vase and a cup. However, the vase and the cup are false positive since these are no vase and cup in the image.

In this case, if we ask question about the number of person in the image or whether there are human in the image, it may response a correct answer. Vice versa, if question is about the vase and cup, it may response a false answer.

ii) Case 2

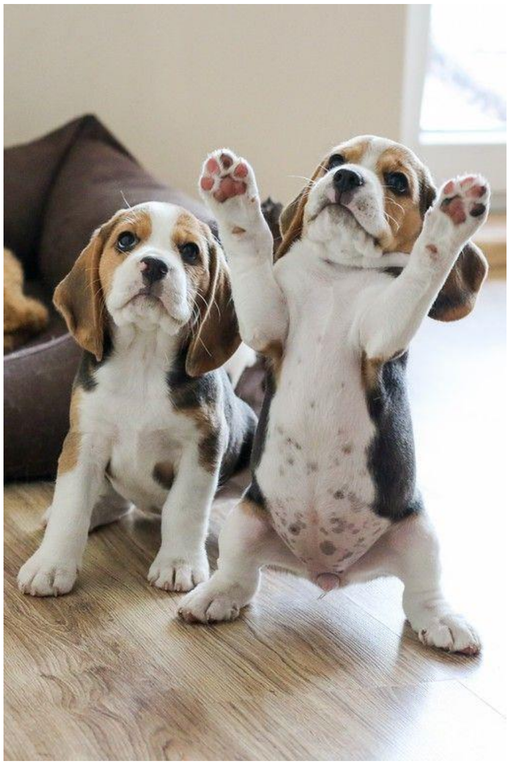


Figure 104: *Sample Image 2*

As we can see, there are two dogs in the above image. The object localization also response the correct box proposals. In this case, the model may have a good performance when answer the question about the animals and numbers.

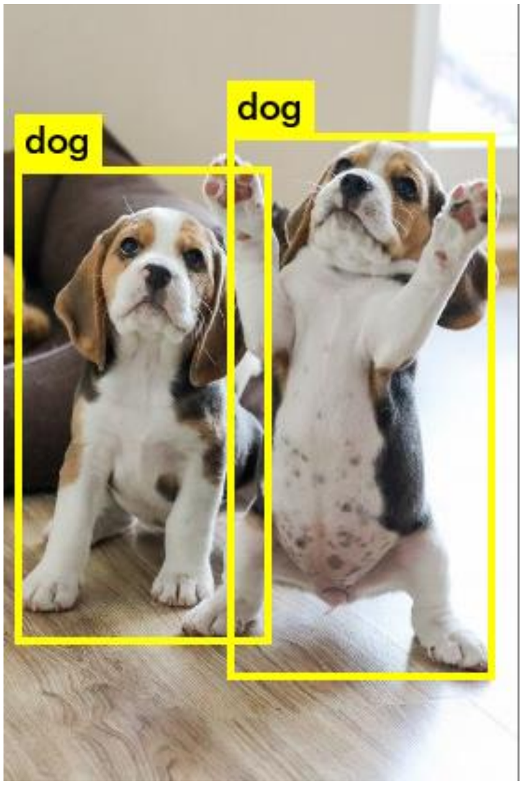


Figure 105: Image after Object Localization

iii) Case 3

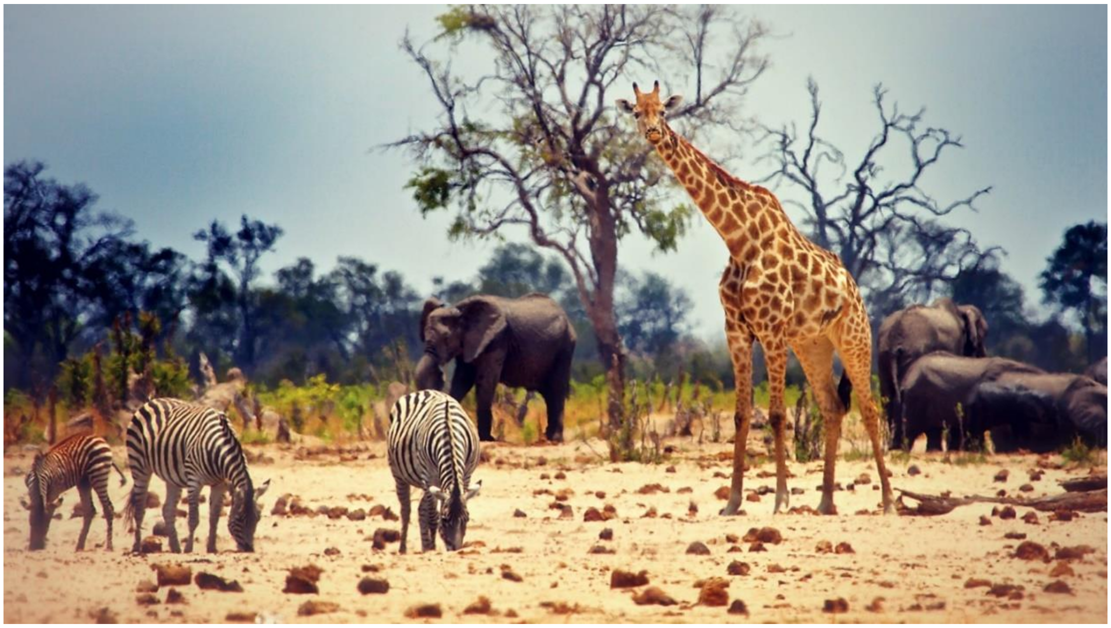


Figure 106: Sample Image 3

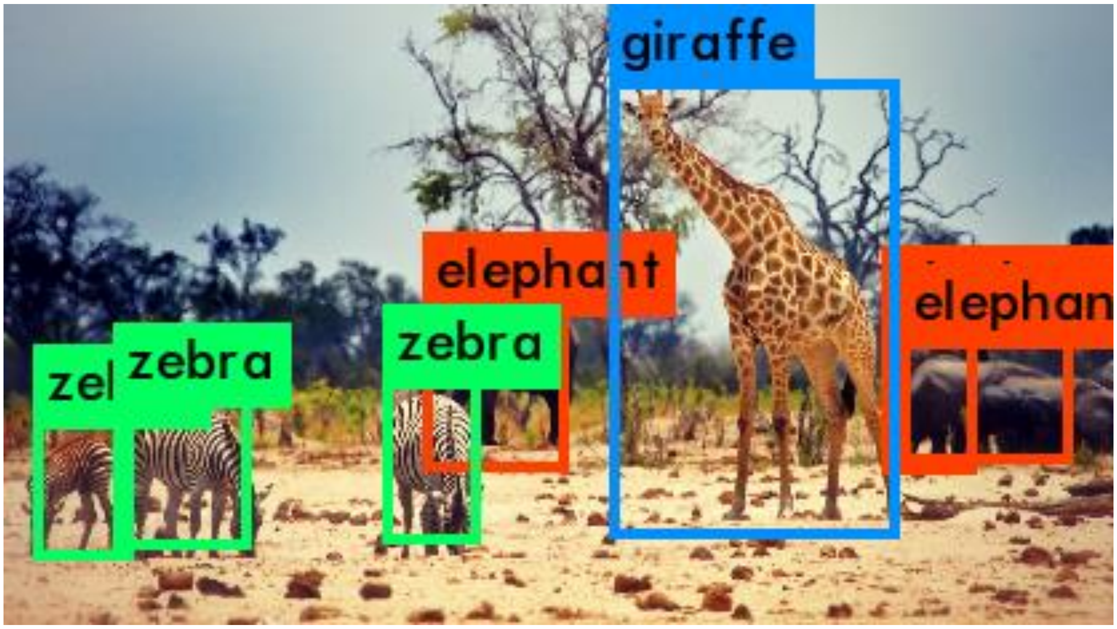


Figure 107: Image after Object Localization

In the above example, the object localization recognize the hippopotamus as the elephant by mistakes, which may cause the wrong answer.

10.2 FRENCH QUESTION ANSWERING

In this semester, we implement a model that support French. Although we can find some samples that can show our model answers properly, our model gives wrong answers most of time.

After we analyze the French VQA procedure, we find that the vector representations of questions in French and English is not very similar. There are several reasons that may explain why this happens.

First, the semantic structures of English and French are not exactly the same, which means the order of words in sentences of English and French are not the same. For example, “What does he do” and “Qu'est-ce qu'il fait?” are the same in meaning. However, if we translate the French sentence into English words by words, the result is “What that he do”, which is different from “What does he do”. Since this two sentence has different semantic structures, the LSTM may consider the French question as different question and give wrong answers.

Second, our bilingual model is not accuracy enough. Since the training dataset is only 200MB, this model does not know many words. In addition, although we map each English word to a French word properly by finding the closest word, the vector representation of English word and French word is not the same. Therefore, when embedding the French question, the whole vector representation of this question is different from vector representation of English version, even though the orders of words are the same in two languages. This difference can lead to the misunderstanding and wrong response of LSTM.

Therefore, according to these two reasons, we need a larger dataset and a bilingual model that is not influenced by order of words.

Chapter 11: CONCLUSION

When we start our final year project, we know little about neural networks or even about machine learning. Therefore, we hope that we can learn something about machine learning, and then have better understanding on machine learning.

At the end of the last semester, we had basic knowledge about Computer Vision, Natural language processing and developed a very basic model of Visual question Answering. We knew that this model was not enough and we needed more knowledge to develop a more powerful model.

At the beginning of this semester, we searched for paper for our new model structure and wanted to support French to make our model more impressive. Finally, we found the paper [9] and this paper led us to develop our model using neural reasoner. In addition, we adopted the bilingual model developed by Stanford to enable French support.

At the end, based on these knowledge and technologies, we are able to implement the final version of Visual Question Answering.

Chapter 12: CONTRIBUTION

Items	Zheng Zhixuan	Tu Fengzhi
Theoretical Discussion	✓	✓
Neural Reasoner Model Design	✓	✓
Bilingual Model Design	✓	✓
French Support Function Design	✓	
Video Question Answering Design		✓
Neural Reasoner Model Implementation	✓	✓
Bilingual Model Implementation	✓	✓
French Support Function Implementation	✓	
Video Question Answering Implementation		✓
Report Writing	✓	✓

Chapter 13: ACKNOWLEDGEMENT

We would like to express our gratitude to our supervisor Prof. Michael Lyu for providing a lot suggestions and opinions. Secondly, we would also like to thank Mr. Edward Yau for his technical support. Last but not least, we would like to express our special thanks to Li Jian, Su Yuxin, Zeng Jichuan who gave us a lot of ideas to do this wonderful project. Without their help, we would encounter more difficulties in this project.

REFERENCE

[1] "Unsupervised Feature Learning and Deep Learning Tutorial." *Unsupervised Feature Learning and Deep Learning Tutorial*. N.p., n.d. Web. 07 Apr. 2017.

[2] Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." *IEEE Transactions on Signal Processing* 45.11 (1997): 2673-2681.

[3] Peng, Baolin, et al. "Towards neural network-based reasoning." *arXiv preprint arXiv:1508.05508* (2015).

[4] Ren, RGJS Shaoqing, Kaiming He, and R. C. N. N. Faster. "Towards real-time object detection with region proposal networks, arXiv preprint." *arXiv preprint arXiv:*1506.01497.

[5] Zitnick, C. Lawrence, and Piotr Dollár. "Edge boxes: Locating object proposals from edges." *European Conference on Computer Vision*. Springer International Publishing, 2014.

[6] Zhang, Ziming, et al. "Bing++: A fast high quality object proposal generator at 100fps." *arXiv preprint arXiv:*1511.04511 (2015).

[7] Zeng, Kuo-Hao, et al. "Leveraging Video Descriptions to Learn Video Question Answering." *arXiv preprint arXiv:1611.04021* (2016).

[8] Castellano, Brandon. "PySceneDetect ." *PySceneDetect*. N.p., n.d. Web. 16 Apr. 2017.

[9] Luong, Thang, Hieu Pham, and Christopher D. Manning. "Bilingual word representations with monolingual quality in mind." *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. 2015.

[10] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013.

[11] Peng, Baolin, et al. "Towards neural network-based reasoning." *arXiv preprint arXiv:1508.05508 (2015)*.