Image-Based Question Answering with Visual Semantic Embeddings

Interim Report

Author: Mengye Ren (998905430)

Supervisor: Richard S. Zemel

February 14, 2015

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Information retrieval (IR) has been a hot topic since 1960s [3]. Momentous achievement has been established in the area of text retrieval and image retrieval. Traditional image retrieval are solely based on text information on web pages. In cases where tags and description are missing, retrieval with a text query is quite difficult [4]. Recently there has been an increasing focus on relating images and text together, producing a number of meaningful results on generating text description based on the image content [5, 6].

Question-Answering (QA) system is another type of IR. Instead of presenting the relevant information, the system produces an answer directly responding to users question. Such systems have been historically successful in text-based content [7]. However, limited research has been conducted to relate the image content with text-based questions.

Building an image-based question answering system has profound implication in the field of artificial intelligence. It can be regarded as an attempt to the visual Turing test [8]. Applications such as human-computer interaction in the context of augmented reality will also be feasible. In short, encoding computer vision knowledge with natural language will be a crucial step connecting human and computer intelligence. Tackling the problem is not an easy task, due to multiple layers of complexity. Not only does the model need to parse the scenes and objects in the image, it also needs to understand the question and generate the relevant answer. Compared to image annotation, QA requires answers to be targeted to the questions with higher precision.

Recently, [8] released a dataset with images and question answer pairs. They also presented their approach which uses latent logical forms to represent the question [9, 10], and latent image segmentation as their visual perception. As an important benchmark, we will further analyze [9]'s methodology and results in Chapter 2.

Our goal is to design an machine learning model that efficiently learns a representation combining both images and sentences. Unlike the approach mentioned above, we formulate the problem using embedding-based approach. First, we need to convert the images and questions into vector forms so

that they can be consumed by a neural network. We propose to use the state-of-the-art convolutional neural networks to obtain the visual features, and use recurrent neural networks to obtain sentence level embedding. We will cover the basics of neural networks and embedding model in Chapter 2.

Now, we can reduce the problem into a multi-class classification problem by assuming only single-word answers. We can learn a function that outputs either the answer class or the nearest neighbour to the correct answer in the semantic embedding space. Similar to pairing sentences and images together [11], here we are looking for a best match from a tuple of image and question to an answer in the visual-semantic embedding. Alternatively, we can use a recurrent neural network as a decoder to generate longer and more free-form answers. Lastly, collecting new data might also be future work for us, as currently available dataset might be insufficient for the task.

Image IR is still a fairly new field, specifically question answering based on an image input. Solving the problem would be a meaningful step in the field of machine learning and artificial intelligence. We believe that our proposed approach will produce encouraging results for building a QA system based on image data and will have theoretical contribution linking computer vision and natural language knowledge in a broader sense.

1.1 Overview of the report

The goal of this interim report is to summarize the research progress up until the mid-point of the thesis. Chapter 2 introduces the background of our proposed methodology, including the very basics of neural networks, and more recent researches on jointly learn image and text with neural network based embedding models. Chapter 3 describes the current progress along our line of research, by presenting several models and experiemental results. A brief discussion is also included. Lastly, Chapter 4 proposes several future items to experiment in the next month.

Chapter 2

Background

Deep architecture and recurrent topology have got significant development over the last decade that allow neural networks to automatically extract abstract and useful features. They are as of today the best tools to solve problems such as image detection [12, 13], speech recognition [14, 15], and language modelling [16, 17], outperforming many known machine learning models with hand-crafted feature-extractors. Since our model formulation relies mainly on deep neural networks, this chapter will begin with the basics of neural networks.

2.1 Neural networks

Artificial neural networks are computational models that consists of interconnected adaptive units [18]. A set of input is feeded to each unit, also called artificial neuron, with certain weights assigned.

Each neuron takes the weighted sum of the input, with an optional bias term, and applies an activation function. It is an intuition borrowed from neuroscience: a neuron is fired after the electricity passes some threshold.

$$z = \mathbf{w}^\top \mathbf{x} + b \tag{2.1}$$

$\mathbf{x}$ is a input vector with input dimension D, and $\mathbf{w}$ is a weight vector also of dimension D, and b is a scalar bias term.

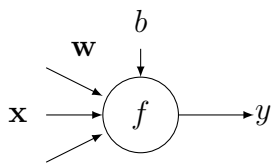


Figure 2.1: An artificial neuron

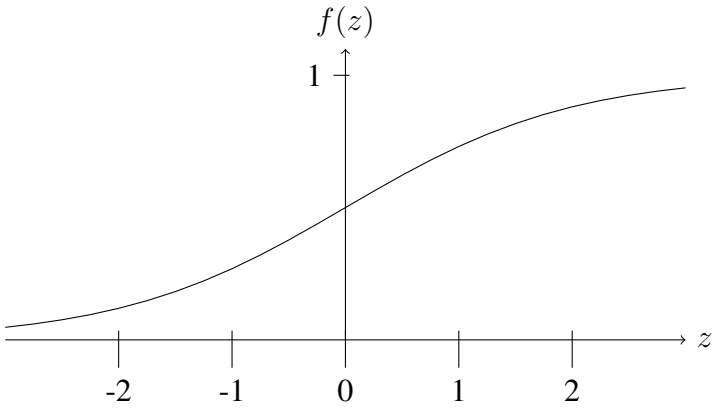


Figure 2.2: The sigmoid function

$$y = f(z) = f(\mathbf{w}^\top \mathbf{x} + b) \tag{2.2}$$

A common activation function are the sigmoid function: $f(z) = \dfrac{1}{1+e^{-z}}$, and the hyperbolic tangent function: $f(z) = \tanh(z)$. As shown in 2.2, the sigmoid function is in its "on" state with output to be one on the far right, and in its "off" state with output to be zero on the far left. There is a linear region around the origin where the output climbs from zero to one.

A feedforward neural network is composed of layers of parallel neurons that processes input from the last layer and transmits data to the next layer. In terms of network topology, a typical feedforward neural network can be decomposed into three types of layers: one input layer, possibly multiple hidden layers, and a output layer. The input layer contains simply the original data samples. The hidden layer and the output layer neurons contains activation functions. We present equations for a single hidden layer network below.

$$\begin{aligned} \mathbf{h} &= f^{(1)}(\mathbf{W}^{(1)}\mathbf{x} + b^{(1)}) \\ y &= f^{(2)}(\mathbf{W}^{(2)}\mathbf{h} + b^{(2)}) \end{aligned} \tag{2.3}$$

$f^{(1)}$ and $f^{(2)}$ are activation functions for the hidden layer and the output layer, which can be different functions. $\mathbf{W}^{(1)}$ is the weight matrix from the input layer to the hidden layer with dimension $H \times D$, where D is the input dimension and H is the hidden dimension. $\mathbf{x}$ is the input vector of the neural network and y is the output value. With a nonlinear activation function such as the sigmoid function, a single hidden layer neural network is proved to have capability on universal function approximation [19].

2.2 Error functions

We first define the error function as some measure of cost from the network output to the desired output. For example, the sum of square error is defined as following:

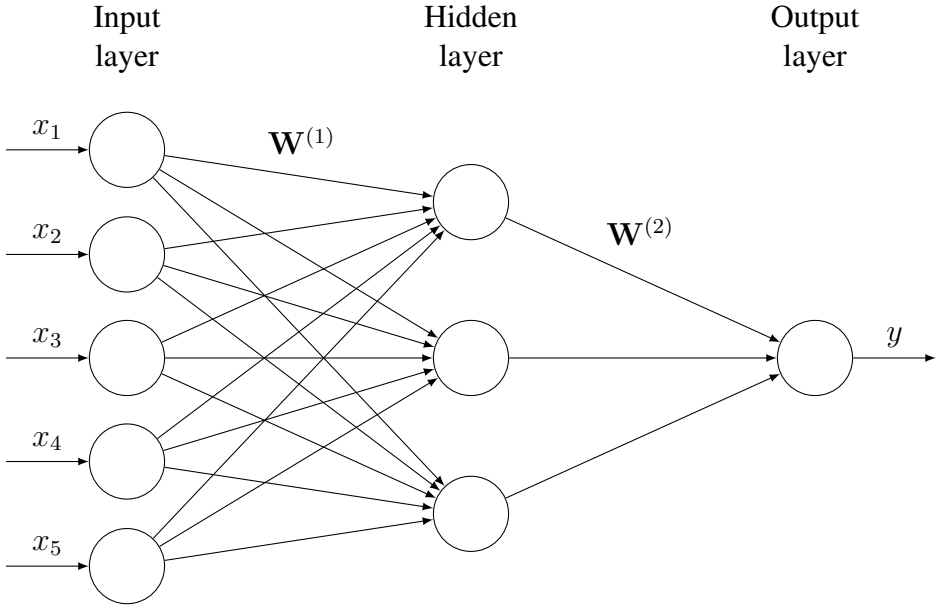


Figure 2.3: A feedforward net with one hidden layer (adapted from [1])

$$E(y) = \sum_i (t_i - y_i)^2 \tag{2.4}$$

y is the network output, and t is the desired output, or the ground truth in the training dataset. i denotes the index of the output element. For classification problems, usually the output is the probability of an example belonging to a class. It is more common to use the cross-entropy error function for this type of problems. For binary classification, the cross-entropy is defined as following:

$$E(y) = \sum_i -t_i \log(y_i) - (1 - t_i) \log(y_i) \tag{2.5}$$

t and y have the same definition as above. $t = 1$ means the example belongs to class 1 and 0 means class 0. If the output of the network is exactly the same with the ground truth, the value of the error function will be zero, and if the output is exactly the opposite, the value of the error function goes to infinity. For multiple class problems, the activation function is usually a softmax function, also called normalized exponential:

$$y = \frac{e^z}{\sum_i e^{z_i}} \tag{2.6}$$

where z is the activation value before output and i denotes the index of the output element. In this way, the class with the maximum probability is preserved and the function is differentiable. The multi-class cross-entropy error function is defined as following:

$$E(y) = \sum_i -t_i \log(y_i) \tag{2.7}$$

t is a vector with all zero except one entry with correct class of the example, and y is a vector of the network predicted class probability. This can be seen as an extension to the two-class case.

2.3 Learning of the weights

2.3.1 Backpropagation

Learning of the weights in neural networks is usually achieved through backpropagation. Equivalently speaking, learning can be regarded as a minimization problem of some error function, and backpropagation a first-order gradient descent method. The calculus chain rule is used to derive the error derivative with regard to weights in each layer. For each update, we take a small step of the gradient towards lower value of the error function.

$$\frac{\partial E}{\partial \mathbf{w}} = \sum_i \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial \mathbf{w}} \tag{2.8}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \gamma \left(\frac{\partial E}{\partial \mathbf{w}} \right)_t \tag{2.9}$$

γ is also called the learning rate or the step size. If we have want to further backpropagate to previous layers of neurons, we need to use the chain rule again to calculate the error derivative with regard to the input of the current layer.

$$\frac{\partial E}{\partial x_j} = \sum_i \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial x_j} \tag{2.10}$$

2.3.2 Momentum

Momentum is a common optimization technique applied on training neural nets. It takes a portion of the last weight update to propagate forward, which significantly speeds up the training by many times. 0.5 or 0.9 are common momentum values in practice.

$$\Delta \mathbf{w}_{t+1} = -\gamma \left(\frac{\partial E}{\partial \mathbf{w}} \right)_t + \alpha \Delta \mathbf{w}_t \tag{2.11}$$

In the equation above, α is the momentum term.

2.3.3 Weight regularization

Overfitting occurs when the model performs too well on the training set but generalizes poorly on the test set. Regularization is often applied to prevent overfitting. Common weight regularization methods include square weight loss, absolute weight loss, etc. In the case of square weight loss, we penalize the norm of the weight vector, to prevent each weight element from growing too large. In the backpropagation of the neural networks, the penalty can also be taken into the derivative.

$$\Delta \mathbf{w}_{t+1} = -\gamma \left(\frac{\partial E}{\partial \mathbf{w}} \right)_t + \alpha \Delta \mathbf{w}_t - \lambda \mathbf{w} \tag{2.12}$$

λ here is defined to be the weight regularization constant.

2.3.4 Dropout

Another way to prevent overfitting in deep neural networks with a large number weights is through dropout [20]. During training, dropout disables a random set of hidden units with a dropout probability, so the network cannot rely on co-adaptation of hidden activations to learn the desired output. During training time,

$$f(X) = D \odot X, D \sim \text{Bernoulli}(p) \tag{2.13}$$

D is a Bernoulli random vector of zeros and ones, with dropout probability $(1-p)$. $\odot$ denotes component-wise product. During testing, the output is rescaled by the dropout probability to match with training statistics.

$$f(X) = pX \tag{2.14}$$

2.4 Convolutional neural network (CNN)

One obvious advantage of neural network is fighting against the curse of dimensionality. For example, a small image patch of 28 pixel x 28 pixel with binary pixel values will generate $2^{28\times 28} = 2^{784}$ different possible images. In the classic MNIST handwritten digit dataset [21], there are only a few thousand training examples. Neural networks can learn the input-output relation with a continuous function that has the ability of generalize to unseen data.

However, one hidden layer neural networks still has its limitations. To achieve larger learning capacity, a large number of hidden neurons are required, but this also increases the total number of weights need to be learned. With only a few thousand examples, a neural network quickly overfits the training set, and perform poorly on the test set. Weight sharing is an important technique to preserve the generalization

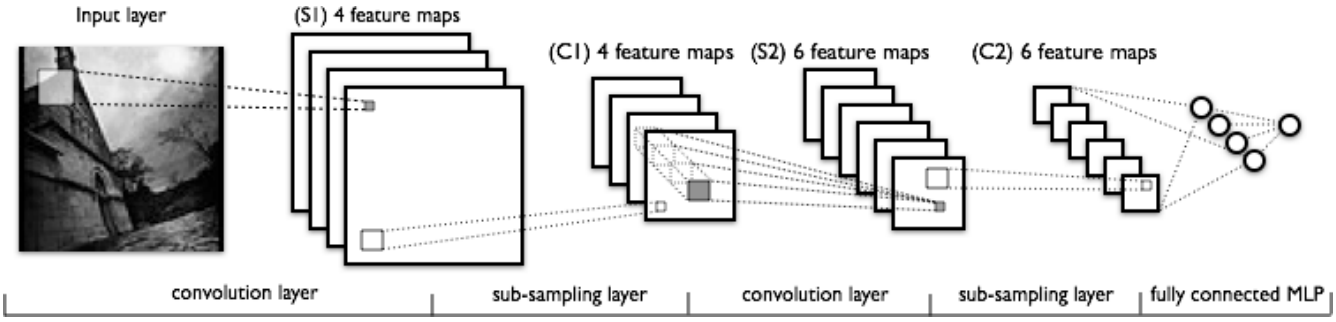


Figure 2.4: A convolutional neural net [2]

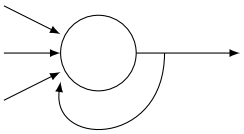


Figure 2.5: A recurrent neuron with a feedback loop

ability. The idea is to force some weights the same across the entire learning process, which is often done through summing all the error derivatives for those shared weights and update them with the sum. Convolutional neural networks [21], is exactly built on top of the the idea of weight sharing. LeNet [21], as one of the first CNNs, achieved the highest MNIST handwritten digit classification rate at the time.

The shared weights can be regarded as small square shape image filters, and the weighted sum of each hidden neurons is equivalent to 2-D convolution on the original image. Filtered images are then passed to a Rectified Linear Unit (ReLU) ($f(z) = \max\{0, z\}$) to obtain nonlinear activation to certain features. Next, to reduce data dimensionality and to make the model robust to translational invariance, a max pooling layer effectively downsamples the intermediate image with the maximum value in each subregion. Lastly, a fully connected softmax layer makes class predictions with each unit representing the probability of input data belonging to certain classes. Such deep architecture has achieved the state of the art image classification and detection results in recent object classification and detection competitions [12, 13].

2.5 Recurrent neural network (RNN)

Recurrent neural network is a type of network that sends the output back to the input neurons, creating feedbacks in the network topology. They are designed to learn time series input because time dependency features can be effectively captured by the feedback. Recently, recurrent neural networks have been applied on many natural language tasks, such as language modelling, machine translation etc. This section will focus on one popular type of recurrent neural network called long short-term memory (LSTM).

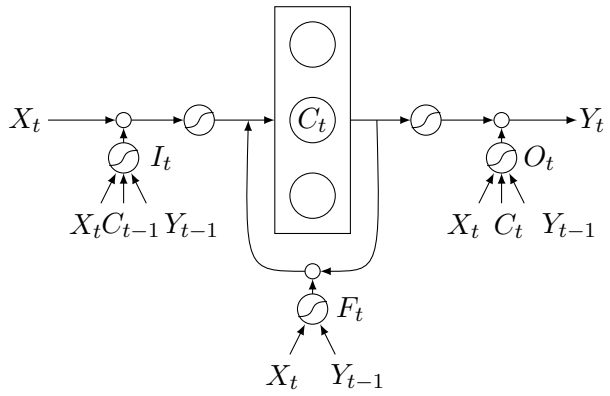


Figure 2.6: Long short-term memory

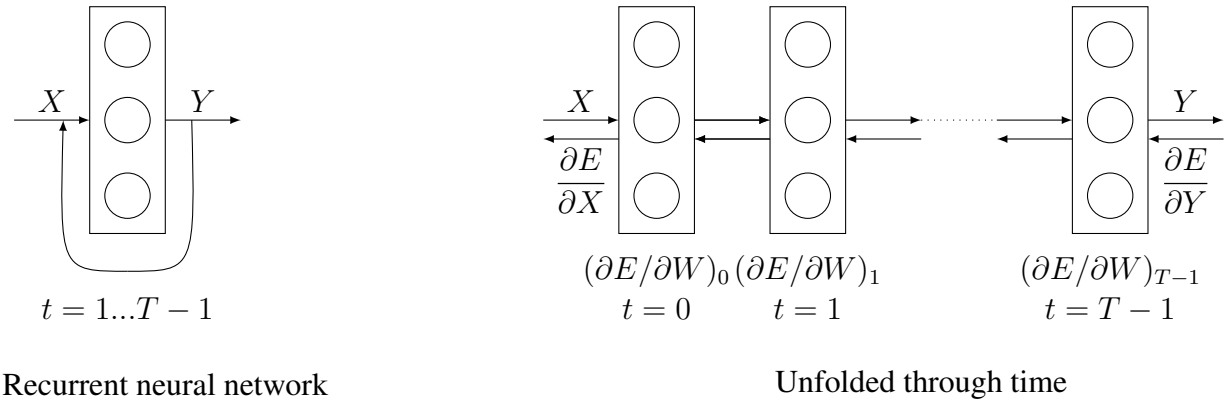
2.5.1 Long short-term memory (LSTM)

Long short-term memory (LSTM) [22] is a major variant of recurrent neural networks. It has a central memory unit (C) with three different types of multiplicative gating that controls the flow of information in the memory. The value of each gating at every time step is determined by a weighted sum of the current input (X_t), the current memory content (C_t), and the current output (Y_t). A sigmoid function maps the weighted sum to a gating value between zero and one. The memory of each time step is then updated by the forget gate (F_t) multiplying with the memory content from the previous time step, and the input gate (I_t) multiplying with the current input through a hyperbolic tangent ($\tanh$) activation function. Lastly, the output of the neural network can be obtained by the output gate (O_t) multiplying with the current memory through another hyperbolic tangent activation. The equations of LSTM can be summarized as below:

$$\begin{aligned} I_t &= \sigma(\mathbf{W}_{ix}X_t + \mathbf{W}_{iy}Y_{t-1} + \mathbf{W}_{ic}C_{t-1} + \mathbf{b}_i) \\ F_t &= \sigma(\mathbf{W}_{fx}X_t + \mathbf{W}_{fy}Y_{t-1} + \mathbf{W}_{fc}C_{t-1} + \mathbf{b}_f) \\ Z_t &= \tanh(\mathbf{W}_{cx}X_t + \mathbf{W}_{cy}Y_{t-1} + \mathbf{b}_c) \\ C_t &= F_t \odot C_{t-1} + I_t \odot Z_t \\ O_t &= \sigma(\mathbf{W}_{ox}X_t + \mathbf{W}_{oy}Y_{t-1} + \mathbf{W}_{oc}C_t + \mathbf{b}_o) \\ Y_t &= O_t \odot \tanh(C_t) \end{aligned} \tag{2.15}$$

Here, σ denotes the sigmoid function, and $\odot$ denotes component-wise product. X_t is an D dimensional input vector at time t. $\mathbf{W}_{ix}, \mathbf{W}_{iy}, \mathbf{W}_{ic}$ are $M \times D$ dimension weight matrices, which maps the input dimension to the memory dimension. Other weight matrices are of dimension $M \times M$. Y_t, C_t are output and memory content vector of dimension M at time t. And $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_c, \mathbf{b}_o$ are bias vectors of dimension M.

One of the main feature of LSTM is the constant error propagation in its memory unit. This is brought by the component-wise multiplication of the forget gate with the memory content from the last time step. This eliminates common RNN problems such as vanishing gradients which makes long sequential model



Recurrent neural network Unfolded through time

Figure 2.7: Back-propagation through time

training very slow.

2.5.2 Learning of LSTM

The learning of LSTM is achieved by backpropagation through time (BPTT) [23], a typical RNN training method. The idea is to imagine the recurrent network as a chain of feedforward network by unfolding the recurrent connections through time. The error derivatives can then be evaluated as in normal feedforward networks. In the end, the errors to weight connections are summed up through time.

$$\frac{\partial E}{\partial W} = \sum_t \left(\frac{\partial E}{\partial W} \right)_t \tag{2.16}$$

The full BPTT algorithm of LSTM can be found in the Appendix 6.1.

2.5.3 Gradient clipping

Exploding gradient is a common issue in the training of RNN. Instead of having a uniform learning rate for any gradient update, the gradient clipping method [24] imposes a hard constraint on the norm of the gradient update. We rescale the gradient if the the norm of the gradient is larger than the constraint.

$$\partial E/\partial W_{new} = \begin{cases} \partial E/\partial W, & \text{if } \|\partial E/\partial W\| \leq C, \\ \partial E \partial W / \|\partial E/\partial W\|, & \text{otherwise.} \end{cases} \tag{2.17}$$

2.5.4 Weight clipping

Weight clipping is equivalent to add a hard constraint to the weight vector. It rescales the weight vector whenever its norm surpasses the contraint. This effectively prevents the weights from exploding during the training.

$$\min E(w) \text{ subject to } g(w) = ||w||^2 - U \leq 0 \tag{2.18}$$

2.5.5 Dropout in RNN

Unlike in feedforward networks, dropout in RNN cannot be simply applied on any hidden units [25]. In fact, dropout is only shown to be effective if applied on non-recurrent connections, i.e. the input neurons of LSTM. In case of multiple stacked LSTMs, a dropout rate of 20% in the first LSTM input, and 50% in the following inter-stage inputs are shown to work the best [25].

2.6 Word embedding

In previous sections, we have covered some neural network basics, popular configurations and effective training tricks. In the following few sections, we will introduce some recent applications in the field of natural language processing and question answering with deep and recurrent neural networks.

A short definition for word embedding is a dictionary which assigns each vocabulary a high dimensional vector that captures semantic similarity. Due to the curse of dimensionality [26] and Zipf's law [27], natural language processing tasks are often limited by the long term dependency and rare words. One would like to share knowledge between similar words without redundant training examples for each word. For example in language modelling tasks, if we have seen the sentence "the cat is walking in the bedroom", then the other sentence "the dog was running in a room" should also be assigned with higher probability [26]. While traditional non-parametric n-gram models fail to capture the associativity of similar words, neural networks have the ability to approximate the probability distribution over a continuous hyperspace. But to enable this task, we need to convert every word into its vector representation. We can first randomly initialize the word embedding matrix as the first layer of the neural network, and then train the network to predict the next word given a sequence of previous words. In the end, the first layer of trained weights can be then used as a fine-tuned word embedding. The trained embedding vectors are distributed representation of words that preserves semantic similarity. Earlier word embeddings are mostly trained from a neural language models, until recently researchers found that a language model task is not necessary for obtaining a good word embedding.

2.6.1 Skip gram embedding model

The skip gram model [28] is a very successful word embedding model. In their paper, they used two methods to train word vectors: the continuous bag of words (CBOW) and the skip gram model. The former aims to predict the missing word given a bag of context words, and the latter aims to predict the context surrounding the given single word. The model is streamlined to one single linear mapping which

significantly shortens the training time down to a few minutes on a standard computer, comparing to a few weeks for previous language models. The methods introduced does not directly aim to train a language model; however, the word embeddings trained seem to be more valuable. It is also astonishing to see algebraic operators can be directly applied on word vectors, and preserve the intended meaning. For example, "king" + "woman" - "man" $\sim$ "queen". The software package released with the paper is called "word2vec", enabling many subsequent researches that directly uses the software to train embeddings as their initial or frozen weights in the first layer of the neural network.

2.7 Jointly learn image and word

Human beings are capable of associating objects of their different forms. For example, when people see a text "pink pig", they are likely to imagine a picture of a pink pig in their head. Likely, when people see blue sky in their eyes, they are likely to have some vocabulary-level mental activity. It is an intriguing question to ask if we can engineer a "visual-semantic joint embedding," an abstract vector space in which the word "sky" and a picture of sky has very small distance. It is very tempting to build such an embedding on top of the existing convolutional neural networks and word embeddings. In DeViSE [29], researchers used model proposed earlier [30] but replaced the image feature extractors with the last hidden layer convolutional net [12] (just before the softmax layer). The image representation is then mapped to the word embedding space [28] with a linear transformation. The model is trained to correctly classify images, and the joint embedding outperforms the standard convolutional neural net by a large margin on a test set with a large number of unseen classes. The reason is that even though the exact images of the classes are unseen, the class names have already had their word similarity captured in the word embedding space, and thus beat the simple fully connected softmax layer in the standard convolutional neural networks. Not only does joint embedding improves the result for pure image tasks, it also encourages higher cooperation of visual and semantic knowledge. The work by [31] proposed models that generate descriptions based on an image input using visual semantic embeddings, and generated sentences have much better plausibility and accuracy compared to earlier results [6, 32]. In their work, they used LSTM introduced earlier as part of the encoder of training descriptions as well as the decoder for generated descriptions. These recent researches have shown us very strong feasibility to jointly learn image and text in a multi-modal space.

2.8 Image-based question answering

The following sections will focus on the background of question-answering and the recent research specifically related to image-based question answering. Traditional QA builds entity relationships within a large text collection, or a corpus, using natural language processing (NLP) techniques. Large-scale question answering has a long history, mostly initiated via the Text Retrieval Conference (TREC). De-

spite some eminent successes in building such systems, such as [33], building a knowledge graph requires a great amount of engineering and is heavily dependent on the knowledge base or the search engine, and is also hard to generalize to unseen entities. Recently, there has been an ongoing effort to use embedding-based model and recurrent neural networks towards question-answering tasks. [34] showed that their memory-based recurrent neural network can perfectly answer questions that involve long time dependency in a paragraph.

Image-based question answering has its own differences compare to text-based question answering: First, the question is centered on the image. Therefore, the model needs to acquire both visual and semantic knowledge. Second, unlike traditional question answering where the training data contains almost the exact answer with certain predicates revealing its relation with the question, here the training data provides no direct information and is solely contained the unseen test image. However, the training data provides very strong clue of what catagories of answers are expected. For example, the model should be able to conclude from the training data that chair is a more probable answer than window for a question like "What is around the table?" These differences actually make the problem unique from traditional QA tasks.

2.8.1 DAQUAR dataset

In 2014, [8] released a dataset with images and question-answer pairs. It is called DAtaset for QUestion Answering on Real-world images (DAQUAR). All images are from the NYU depth v2 dataset [35], and are taken with indoor scenes. Human segmentations, image depth values, and object labellings are available in the dataset. The QA pairs were created by human crowdsourcing. The original train-test split ratio is about 3800:3300. The QA data has two sets of configuration: the 37-class and the 894-class dataset, differed by the number of object classes appeared in the questions. There are mainly three types of questions in this dataset:

1. Object type
2. Object color
3. Number of object

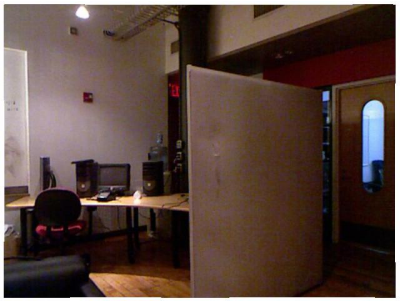
Some of the questions provide modifiers to restrict the subject, such as location and color, while others do not. Qualitatively speaking, a large proportion of the questions are very hard to answer. Figure 2.8 summarizes the sources of difficulties. Since DAQUAR is the only publicly available image-based QA dataset, it is currently our primary benchmark to evaluate our models.

2.8.2 Previous attempt

Together with the release of the DAQUAR dataset, [9] presented an approach which combines semantic parsing and image segmentation. In the natural language part of their work, they used a semantic parser

Q7: what color is the ornamental plant in fornt of the fan coil but not close to sofa ? -red

(a) Hard to locate the object



Q20: how many black objects are on the desk ? -three

(b) Counting ambiguity, could be three to five

Q1129: what is behind the tap ? -blinds

(c) Object ambiguity, could be a window or blinds

Q5: what is under the white cabinet on the right side of the oven and and on the left side of the fridge ? -sink

(d) Lengthy description of spatial relations

Q1423: what is the colour of the furniture ? -brown

(e) Color ambiguity due to dim lighting

Q1467: what is on the night stand ? -paper

(f) Object is too small

Figure 2.8: Various types of difficult questions in the DAQUAR dataset

[10] to parse the sentence into latent logical forms. The parser is specially designed for QA tasks, and can be trained well where the true dependency tree is missing from the training data. In the image part, they evaluated different latent logical forms of the question with multiple segmentation. They obtained the multiple segmentations of the image by sampling the uncertainty of the segmentation algorithm. Their model is based on a Bayesian formulation that every logical form and image segmentation has certain probability. To make the inference step of their algorithm scalable, they chose to sample from the nearest neighbours in the training set according to the similarity of the semantic parsing.

While their model seems to handle a number of spatial relations, their predefined set of possible predicates constrains the question form and the environment. Moreover, the quality of answers also depends on the accuracy of the automatic image segmentation algorithm. Lastly, inferring all possible predicates between every pair of objects is not scalable to larger dataset. The embedding space models introduced earlier have significant differences with their approach, and we hope that our embedding space model will outperform their model in terms of answer accuracy and similarity.

Chapter 3

Current progress

3.1 Problem restatement

We consider the dataset of image-based question answering as a triplet of an image vector, a question of a sequence of word indices, and an answer of a sequence of word indices. We start off by assuming that the answers are all one-word answers, i.e. the length of the answer sequence is always one. The assumption is established based on more than 96% of the data entries in DAQUAR have only one-word answers. Our goal is to learn a function that takes the input of an image vector and a sequence of word indices, and emits the output of the answer word.

3.2 LSTM sentence model

Due to the sequential nature of natural languages, researchers have been looking for ways to model sentences using recurrent neural networks. There has been increasing interests in using LSTM to model the embedding vector for a whole sentence. Recently, [36] uses LSTM as both encoders and decoders in machine translation, and achieved BLEU score [37] better than the traditional phrase-based models [38]. The architecture [39] is composed of the following elements:

1. Look up words in a word embedding lookup table. This table can be either pre-trained or randomly initialized.
2. At each time step input one word vector into the LSTM. The LSTM latent memory dimension can be different than the input dimension.
3. Retrieve the output of the LSTM at the last time step as the sentence vector.

To test the LSTM sentence model, we train it on the sentence polarity 1.0 dataset [40], and used 10% held-out data as our validation. In this experiment we used the skip gram word embedding model trained on Google News [28] as a frozen embedding layer in our task (i.e. the error does not get back propagated

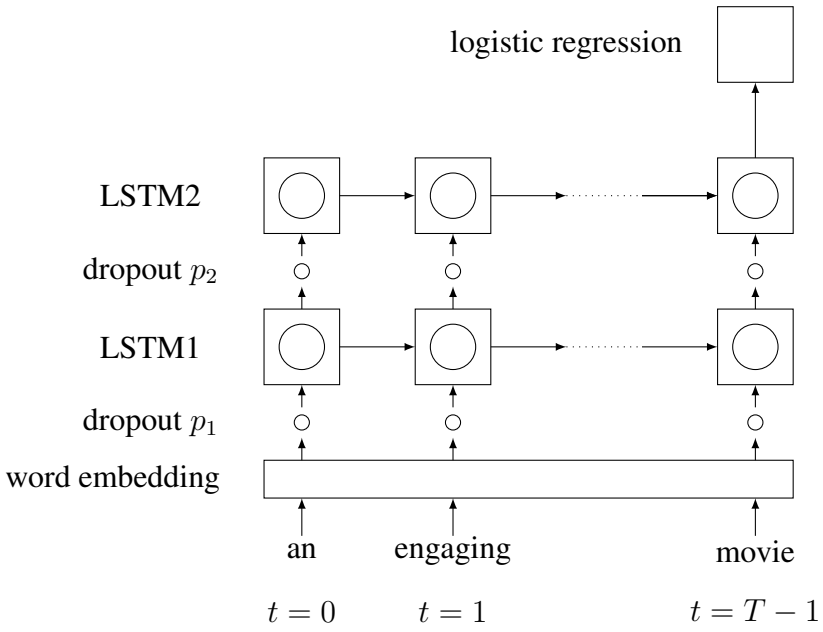


Figure 3.1: Multi-layer LSTM sentence model

into the word embedding matrix.) We used the output of the LSTM at the last time step as the latent representation of the sentence, and apply logistic regression to output positive or negative prediction. We alternatively stack two LSTM together: the output of the first LSTM is send to the input of the second LSTM, with a dropout layer in the middle. This idea of stacking LSTM is from [36]. Figure 3.1 shows the architecture of the of the model.

We compare our results with some previous approaches:

Table 3.1: Performance of LSTM sentence model on sentence polarity dataset

	LSTM-2	LSTM-1	CNN [41]	Sent Parser [42]	RAE [43]
Rate	**0.8178**	0.7886	0.815	0.795	0.777

In Table 3.1, LSTM1 is one-layer LSTM model, and LSTM2 is two-layer LSTM model, with dropout in between. The full training detail is included in Appendix 6.2.1. LSTM sentence model matches the state of the art performance in this particular dataset. Therefore we believe that LSTM will be a reasonable model for sentence embedding.

3.3 Image-word model and blind model

We start off the experiment for image-based question answering by directly building on top of the sentence model.

In this experiment, we designed a model called the "image-word model" because it treats the image as if being one word of the question. We borrowed the idea of treating image as the first word of the image

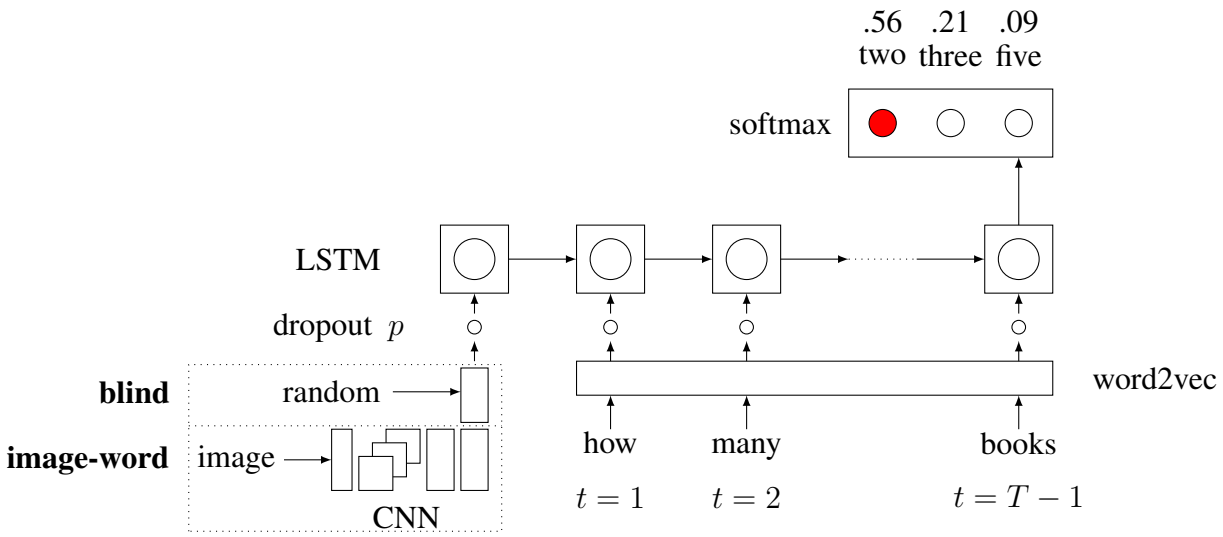


Figure 3.2: Image-word model and blind model

from [44] in their caption generation paper. The difference with the caption generation is that here we are only output the answer at the last time step.

1. We use the last hidden layer of the Oxford convolutional neural net (4096 dimension) [45] as our visual embedding model.
2. We use the word2vec embedding model (skip gram) trained from Google News [28] as our frozen semantic embedding (300 dimension).
3. We then treat the image as if it is the first word of the sentence. In order to map the image embedding to our latent semantic embedding space, we used a linear transformation layer of dimension 4096×300 (similar to [29]).
4. The last time step of the LSTM output is passed to a softmax layer. The final output is a 68-dimension vector representing the probability of the answering being each possible answer classes.

To simplify our model, we used the 37 class dataset, so the question and answers will only focus on 37 object classes. We also filtered dataset down to only one-word answers. This simplification trims the possible set of answers to 68 words. To evaluate the model, we used the plain answer accuracy as well as the Wu-Palmer similarity (WUPS) measure [46, 9]. The WUPS calculuates the similarity between two words based on their longest common subsequence in the taxonomy tree. To compare the results, we also designed a baseline model to avoid over-interpretation. In this model, the images are replaced by randomly generated vectors (noises) instead of meaningful graphics. We call this baseline the "blind model." The training detail of the models above can be found in Appendix 6.2.2. Table 3.2 summarizes the performance of image-word model compared to our baseline and the model in DAQUAR.

First, our model wins by a large margin compared to the results from the publisher of the DAQUAR dataset. Second, we argue that most gain in accuracy results from a good sentence embedding, because the image-word model has almost the same accuracy compared to the blind model. Third, the higher

Table 3.2: Performance of image-word model

	Image-word	Blind	Multi-world [9]	Human [9]
Rate	**0.3121**	0.3042	0.1273	0.6027
WUPS @ 0.9	**0.4789**	0.3066	0.1810	0.6104
WUPS @ 0.0	**0.7874**	0.7214	0.5147	0.7896

similarity score obtained by the image-word model suggests that the visual embedding still has some positive influence in making correct decisions.

In Figure 3.3, we further discovered the some weak visual ability by a direct comparison of test examples. We observed that image-word model seems to perform better on questions centered on the dominant objects or the dominant colors, but does not seem to learn how to count.

3.4 Data augmentation

From previous experiments, we observed that our model is powerful at remembering sentences which overfits the training set badly. We also believed that the difficulty level is too high as for computers to answer. Therefore in the following experiments we aim to create more training data and decrease the level of difficulty by generating some synthetic QA-pairs based on the ground truth labeling in the original image dataset. The following two types of questions have been considered:

1. How many object ?
2. What is the color of the object ?

Due to an uneven distribution of the colors and numbers in the training set, we take a maximum of 400 questions of each color and number class. This will give a more evenly distributed training set. In total we generated 2000 color-related, 1200 counting-related questions in the training set, 1600 color-related and 1000 counting-related questions in the test set.

We ran two kinds of experiments on the synthetic data.

1. We used only synthetic questions to make up trainining and test set.
2. We augmented the original training set with synthetic questions but evaluate on the original test set.

Up until this document is due, we have not observed significant improvement on answer accuracy with our generated questions. In experiment 1, no matter how different the images are, the model always predicts similar confidence level for all colors and numbers, possibly directly learned from the answer distribution. In experiment 2, the model does predict different confidence level for different images on the synthetic questions, but there is no improvement in terms of answer accuracy in the original test set. We decided to further investigate data augmentation methods in our future works.

Q193: what is the largest object ?
Correct answer: table
Image-word: table (0.576)
Blind: bed (0.440)

(a) The question gives no extra clue of the class of the object. The visual model recognizes the correct object through the shape.

Q1129: what is behind the tap ?
Correct answer: blinds
Image-word: window (0.470)
Blind: mirror (0.347)

(b) The visual model recognizes the object class through the luminence.

Q2370: what is liquid colour in third bottle ?
Correct answer: yellow
Image-word: white (0.504)
Blind: white (0.064)

(c) The visual model recognizes the dominant color with large probability, but failed to attend to the point of interest.

Q212: what is the object left of the room divider ?
Correct answer: door
Image-word: toilet (0.3973), sink (0.1367), towel (0.1323)
Blind: refridgerator (0.5318)

(d) The visual model recognizes the correct scene.

Q146: how many pink cubes are on the top of the night stand on the left side of the bed ?
Correct answer: one
Image-word: two (0.4283), one (0.4186)
Blind: one (0.9610)

(e) The visual ability also creates ambiguity. Blindly guessing sometimes wins.

Q1615: how many pictures are there on the wall ?
Correct answer: seven
Image-word: three (0.4518)
Blind: three (0.6047)

(f) No evidence shows that image-word model learns how to count.

Figure 3.3: Direct comparison between image-word model and blind model.

Chapter 4

Future works

4.1 Data augmentation and new data

As mentioned above, the current model has very large learning ability and we believe that data augmentation will help the learning model to perform better. There is further need to extend our dataset, potentially more synthetic questions and more images. First, we can consider generating more question types based on the ground truth labelling in DAQUAR. Second, we can convert image descriptions to QA-pairs. The currently available image description datasets include Microsoft COCO [47], and Flickr image dataset [48], which contain far more images, object types, and object relations.

4.2 More models

Our image-word model is the first attempt in this project to combine the visual and semantic embedding space together. From the experimental results, it is obvious that this model has more bias on the sentence content than the image content. The similar answer accuracy of the blind model suggests that the model heavily relies on memorizing the questions. And our direct comparison also shows that the visual capacity is only limited to dominant objects. It is possible that other models may perform better. We here introduce more models that will be implemented in the future.

4.2.1 Ranking model

This type of model outputs a vector that is the nearest neighbor of the answer word in the semantic embedding. The error function uses the pair-wise ranking loss [30], defined below:

$$\sum_{\mathbf{y}} \sum_{i \neq j} \max\{0, \alpha - s(\mathbf{y}, \mathbf{a}_j) + s(\mathbf{y}, \mathbf{a}_i)\} \tag{4.1}$$

$\mathbf{y}$ and $\mathbf{a}$ are both vectors in the answer embedding space. $\mathbf{y}$ is the output of the model, $\mathbf{a}_j$ is the correct answer for the question, and $\mathbf{a}_i$ is one of any possible answers. $s(\cdot, \cdot)$ denotes the similarity measure of two vectors. It is usually implemented as the cosine similarity in high dimensional spaces:

$$s(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|} \tag{4.2}$$

This error function will penalize the model if the right answer is not winning over other wrong answers by a certain margin α. It is a widely used in ranking image descriptions [31].

4.2.2 Visual attention model

As people are answering a question on some picture, they often skim through the entire picture and focus on the point of interest that is closely related to the question. For example, a question like "what is on the table?" expects them to first locate the object "table" and look above the object. Knowing where to look at is actually extremely important in this task. And as shown in the DAQUAR dataset, there are many questions that do not focus on the main object in the image. Figure 3.3 further suggests that there is an acute need for the model to learn to capture visual details. Therefore we believe that building alignment between words and the attention of the image will help the model get to the correct answer.

Fortunately we can borrow the recent breakthroughs in attention modelling [49] to help us on this task. The input of the attention model is the convolutional layer features in the CNN [45], rather than the fully connected layer. In this way, it will preserve the local information to a larger degree. The attention function gives a probability distribution (visual attention) over the entire image at every time step. The visual attention, the image and the question word are input to the recurrent neural network, and the output of the network forms a feedback loop to the attention function through a multilayer feedforward network. The attention model has the state-of-the-art scores on image description generation, and we believe that incorporation of this model will boost our answer accuracy.

4.3 Longer answers and free-form answers

Up until now we have assumed that there are only one-word answers, but ideally we would like to consider longer answers as well. Inspired by [36], we could possibly extend the length of our answers by using an RNN decoder such as LSTM. We can borrow the idea in [31], to either have a template to generate a word in certain word category at a time, or generate a list of free-form answer candidates and select the best candidate. It should be noted that free-form answers also need a better quantitative metric to measure answer quality, in term of its accuracy and language soundness. Lastly, this would need a larger varieties of question-answer types for the model to learn variations in language generation.

Chapter 5

Summary

This interim report covered the background of our proposed approach on the topic of image question answering, introduced a few models that we have experimented, and listed some future works to be done by the end of the project.

Chpater 2 introduced the background of neural networks and recent highlights on using neural networks in the fields of computer vision and natural language processing. In particular, I used the word embedding and convolutional neural net hidden layer as building blocks of our subsequent work on image question answering.

Chapter 3 explained the current project progress. First I presented a sentence model based on LSTM. It consumes a sequence of word vectors and output the hidden vector representation of a sentence. We tested it on the sentence polarity dataset and achieved the state-of-the-art classification rate. Next, we briefly explored possible ways to combine the visual and semantic embedding space. One naive way is to treat the image as the first word of the sentence. We assumed that all answers are single-word, and reduced the problem into a multi-class classification problem. We compared this model with the previous approaches on the same dataset, and observed a large winning margin in terms of classification rate and the similarity measure. Further, we noticed that the blind model can achieve almost the same classification rate. We then systematically analyzed the limitation of the visual capability of the image-word model. We initiated the work of data augmentation, but this work has not generated improvement yet.

In Chapter 4, some future works are listed, including new data, more advanced embedding models, and possibly template-based or free-form answers. I hope these options can be explored in the next month and their results can improve upon our current progresses.

Chapter 6

Appendix

6.1 LSTM BPTT Algorithm

$(\partial E/\partial Y)_{t,i}$ is the error derivative with regard to the output from some external error function, for each timestep and each output dimension. $(\partial E/\partial W)_{i,j}$ is the error derivative with regard to the weight matrix, which is a matrix concatenation of all the weight matrices along the last axis. $(\partial E/\partial X)_{t,j}$ is the error derivative with regard to the input, for each timestep and each input dimension.

input: $(\partial E/\partial Y)_{t,i}$

output: $(\partial E/\partial W)_{i,j}, (\partial E/\partial X)_{t,j}$

procedure LSTMBPTT($(\partial E/\partial Y)_{(t,i)}$)
    $S^{(1)}_{t,i} \leftarrow (X_{t,i}, Y_{t-1,i}, C_{t-1,i}, \mathbf{1}_t)$
    $S^{(2)}_{t,i} \leftarrow (X_{t,i}, Y_{t-1,i}, \mathbf{1}_t)$
    $S^{(3)}_{t,i} \leftarrow (X_{t,i}, Y_{t-1,i}, C_{t,i}, \mathbf{1}_t)$
    $U_{t,i} \leftarrow \tanh(C_{t,i})$
    $(\partial U)_{t,i} \leftarrow 1 - U_{t,i} \odot U_{t,i}$
    $(\partial Z)_{t,i} \leftarrow 1 - Z_{t,i} \odot Z_{t,i}$
    $(\partial I)_{t,i} \leftarrow I_{t,i}(1 - I_{t,i})$
    $(\partial F)_{t,i} \leftarrow F_{t,i}(1 - F_{t,i})$
    $(\partial O)_{t,i} \leftarrow O_{t,i}(1 - O_{t,i})$
    $(\partial C/\partial I)_{t,i} \leftarrow Z_{t,i}(\partial I)_{t,i}$
    $(\partial C/\partial F)_{t,i} \leftarrow C_{t-1,i}(\partial F)_{t,i}$
    $(\partial C/\partial Z)_{t,i} \leftarrow I_{t,i}(\partial Z)_{t,i}$
    $(\partial Y/\partial O)_{t,i} \leftarrow U_{t,i}(\partial O)_{t,i}$
    for $t \in \{T-1 ... 0\}$ **do**
        $(\partial Y/\partial C)_{i,j} \leftarrow \mathbf{diag}(O_{t,i}(\partial U)_{t,i}) + (\partial Y/\partial O)_{t,i} W^{oc}_{i,j}$
        if $t = T - 1$ **then**

$(\partial E/\partial Y)^c_j \leftarrow (\partial E/\partial Y)_{t,j}$
$(\partial E/\partial C)^c_j \leftarrow \sum_i (\partial E/\partial Y)_i (\partial Y/\partial C)_{i,j}$
else
$(\partial E/\partial Y)^c_j \leftarrow \sum_i (\partial E/\partial Y)^c_i (\partial Y/\partial Y)_{i,j} + \sum_i (\partial E/\partial C)^c_i (\partial C/\partial Y)_{i,j} + (\partial E/\partial Y)_{t,j}$
$(\partial E/\partial C)^c_j \leftarrow \sum_i (\partial E/\partial C)^c_i (\partial C/\partial C)_{i,j} + \sum_i (\partial E/\partial Y)^c_i (\partial Y/\partial C)_{i,j}$
end if
$(\partial E/\partial I)_{t,i} \leftarrow (\partial E/\partial C)^c_i (\partial C/\partial I)_{t,i}$
$(\partial E/\partial F)_{t,i} \leftarrow (\partial E/\partial C)^c_i (\partial C/\partial F)_{t,i}$
$(\partial E/\partial Z)_{t,i} \leftarrow (\partial E/\partial C)^c_i (\partial C/\partial Z)_{t,i}$
$(\partial E/\partial O)_{t,i} \leftarrow (\partial E/\partial Y)^c_i (\partial Y/\partial O)_{t,i}$
$(\partial C/\partial C)_{i,j} \leftarrow \mathbf{diag}(F_{t,i}) + (\partial C/\partial F)_{t,i} W^{fc}_{i,j} + (\partial C/\partial I)_{t,i} W^{ic}_{i,j}$
$(\partial C/\partial Y)_{i,j} \leftarrow (\partial C/\partial F)_{t,i} W^{fy}_{i,j} + (\partial C/\partial Z)_{t,i} W^{cy}_{i,j} + (\partial C/\partial I)_{t,i} W^{iy}_{i,j}$
$(\partial Y/\partial Y)_{i,j} \leftarrow (\partial Y/\partial O)_{t,i} W^{oy}_{i,j}$
end for
$(\partial E/\partial W_i)_{i,j} \leftarrow \sum_t (\partial E/\partial I)_{t,i} S^{(1)}_{t,j}$
$(\partial E/\partial W_f)_{i,j} \leftarrow \sum_t (\partial E/\partial F)_{t,i} S^{(1)}_{t,j}$
$(\partial E/\partial W_c)_{i,j} \leftarrow \sum_t (\partial E/\partial Z)_{t,i} S^{(2)}_{t,j}$
$(\partial E/\partial W_o)_{i,j} \leftarrow \sum_t (\partial E/\partial O)_{t,i} S^{(3)}_{t,j}$
$(\partial E/\partial W)_{i,j} \leftarrow ((\partial E/\partial W_i), (\partial E/\partial W_f), (\partial E/\partial W_c), (\partial E/\partial W_o))_{i,j}$
$(\partial E/\partial X)_{t,j} \leftarrow \sum_i (\partial E/\partial I)_{t,i} W^{ix}_{ij} + (\partial E/\partial F)_{t,i} W^{fx}_{ij} + (\partial E/\partial Z)_{t,i} W^{cx}_{ij} + (\partial E/\partial O)_{t,i} W^{ox}_{ij}$
return $(\partial E/\partial W)_{i,j}, (\partial E/\partial X)_{t,j}$
end procedure

Note: In the algorithm above, $\mathbf{diag}(\cdot)$ produces a diagonal matrix given a vector. $(\partial E/\partial Y)^c$ and $(\partial E/\partial C)^c$ denote the cumulative error to the output and to the memory at the time t in the for-loop. These values are cascaded while looping back from the last time step to the beginning, whereas $(\partial E/\partial Y)$ is the initial error at each time step due to some error function from the later stages of the model. $(\partial E/\partial Y)^c$ takes into account of both the error from later output due to the current output, and the error directly from the current output, i.e. $(\partial E/\partial Y)_t$.

6.2 Training details

6.2.1 LSTM sentence model

We list here the hyperparameters of the model by layers.

1. Word embedding layer
   Using word2vec skip gram embedding vectors trained from Google News[28]
   Learning rate: 0.0

2. Input dropout layer
   Dropout rate: 20%
3. 1st LSTM layer
   Input dimension: 300
   Memory dimension: 50
   Initialization of $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o$: fixed at 1.0
   Initialization of $\mathbf{b}_c$: fixed at 0.0
   Initialization of other weights: uniform [-0.05, 0.05]
   Learning rate: 0.8
   Momentum: 0.9
   Gradient clipping: 0.1
   Weight clipping: 20.0
   Weight regularization: square loss, λ = 5e-5
4. Inter-LSTM dropout layer (Only in LSTM2 model)
   Dropout rate: 50%
5. 2nd LSTM layer (Only in LSTM2 model)
   Input dimension: 50
   Other configuration same as 1st LSTM layer.
6. Sigmoid layer
   Input dimension: 50
   Output dimension: 1
   Initialization: uniform [-0.05, 0.05]
   Learning rate: 0.01
   Momentum: 0.9
   Gradient clipping: 0.1
   Weight clipping: 5.0
   Weight regularization: square loss, λ = 5e-5

The model is trained for 120 epochs. Every epoch we shuffled the training data order. We took a mini-batch of size 25.

6.2.2 Image-word model

We list here the hyperparameters of the model by layers.

1. Image embedding layer
   Using Oxford net [45] last hidden layer 4096 dimension features
   Learning rate: 0.0
2. Word embedding layer

Using word2vec skip gram embedding vectors trained from Google News [28]
Learning rate: 0.0

3. Image linear transformation layer
   Input dimension: 4096
   Output dimension: 300
   Initialization: uniform [-0.05, 0.05]
   Learning rate: 0.8
   Momentum: 0.9
   Gradient clipping: 0.1
   Weight clipping: 1000.0
4. Input dropout layer
   Dropout rate: 20%
5. LSTM layer
   Input dimension: 300
   Memory dimension: 150
   Initialization of $\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o$: 1.0
   Initialization of $\mathbf{b}_c$: 0.0
   Initialization of other weights: uniform [-0.05, 0.05]
   Learning rate: 0.8
   Momentum: 0.9
   Gradient clipping: 0.1
   Weight clipping: 100.0
   Weight regularization: square loss, λ = 5e-5
6. Softmax layer
   Input dimension: 150
   Output dimension: 68
   Initialization: uniform [-0.05, 0.05]
   Learning rate: 0.01
   Momentum: 0.9
   Gradient clipping: 0.1
   Weight clipping: 10.0
   Weight regularization: square loss, λ = 5e-5

The model is trained for 60 epochs. Every epoch we shuffled the training data order. We took a mini-batch of size 25.

6.2.3 Blind model

All hyperparameters are kept the same with the image-word model, except that we replaced the 4096 dimension frozen image feature input with 4096 dimension frozen random vectors, uniformly initialized from range [-0.05, 0.05].

Bibliography

[1] K. M. Fauske, "Example: Neural network," 2006. [Online]. Available: http://www.texample.net/tikz/examples/neural-network/

[2] DeepLearning.net, "Lenet." [Online]. Available: http://deeplearning.net/tutorial/lenet.html

[3] C. D. Manning, *Introduction to Information Retrieval*, 1st ed. Cambridge University Press, 2008.

[4] V. Gudivada and V. Raghavan, "Content based image retrieval systems," *Computer*, vol. 28, no. 9, pp. 18–22, Sep 1995.

[5] R. Kiros, R. Salakhutdinov, and R. Zemel, "Multimodal neural language models," in *International Conference on Machine Learning (ICML'14)*, 2014.

[6] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg, "Baby talk: Understanding and generating simple image descriptions," in *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, 2011, pp. 1601–1608.

[7] T. Strzalkowski and S. Harabagiu, *Advances in Open Domain Question Answering*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[8] M. Malinowski and M. Fritz, "Towards a visual turing challenge," *CoRR*, vol. abs/1410.8027, 2014. [Online]. Available: http://arxiv.org/abs/1410.8027

[9] ——, "A multi-world approach to question answering about real-world scenes based on uncertain input," in *Neural Information Processing Systems (NIPS'14)*, 2014. [Online]. Available: http://arxiv.org/abs/1410.0210

[10] P. Liang, M. I. Jordan, and D. Klein, "Learning dependency-based compositional semantics," *Computational Linguistics*, vol. 39, no. 2, pp. 389–446, 2013.

[11] A. Karpathy, A. Joulin, and L. Fei-Fei, "Deep fragment embeddings for bidirectional image sentence mapping," *CoRR*, vol. abs/1406.5679, 2014. [Online]. Available: http://arxiv.org/abs/1406.5679

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012.*, 2012, pp. 1106–1114.

[13] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, 2014, pp. 580–587.

[14] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[15] L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. L. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, 2013, pp. 8604–8608.

[16] A. Mnih and G. E. Hinton, "Three new graphical models for statistical language modelling," in *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, 2007, pp. 641–648.

[17] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, 2010, pp. 1045–1048.

[18] M. Hassoun, *Fundamentals of Artificial Neural Networks*. A Bradford Book, 2003.

[19] K.-L. Du and M. N. S. Swamy, *Neural Networks and Statistical Learning*. Springer, 2014.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," *Complex Systems*, 1995.

[24] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, 2013, pp. 1310–1318.

[25] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409.2329, 2014. [Online]. Available: http://arxiv.org/abs/1409.2329

[26] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003. [Online]. Available: http://www.jmlr.org/papers/v3/bengio03a.html

[27] G. K. Zipf, *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

[28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[29] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov, "Devise: A deep visual-semantic embedding model," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013.*, 2013, pp. 2121–2129.

[30] J. Weston, S. Bengio, and N. Usunier, "Large scale image annotation: learning to rank with joint word-image embeddings," *Machine Learning*, vol. 81, no. 1, pp. 21–35, 2010.

[31] R. Kiros, R. Salakhutdinov, and R. S. Zemel, "Unifying visual-semantic embeddings with multimodal neural language models," *CoRR*, vol. abs/1411.2539, 2014. [Online]. Available: http://arxiv.org/abs/1411.2539

[32] M. Mitchell, J. Dodge, A. Goyal, K. Yamaguchi, K. Stratos, X. Han, A. Mensch, A. C. Berg, T. L. Berg, and H. D. III, "Midge: Generating image descriptions from computer vision detections," in *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, 2012, pp. 747–756.

[33] B. L. Lewis, "In the game: The interface between watson and jeopardy!" *IBM Journal of Research and Development*, vol. 56, no. 3, p. 17, 2012. [Online]. Available: http://dx.doi.org/10.1147/JRD.2012.2188932

[34] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *CoRR*, vol. abs/1410.3916, 2014. [Online]. Available: http://arxiv.org/abs/1410.3916

[35] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[36] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[37] K. Papineni, S. Roukos, T. Ward, and W. jing Zhu, "Bleu: a Method for Automatic Evaluation of Machine Translation," in *Meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[38] T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the rare word problem in neural machine translation," *CoRR*, vol. abs/1410.8206, 2014. [Online]. Available: http://arxiv.org/abs/1410.8206

[39] R. Kiros, personal communication.

[40] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[41] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014. [Online]. Available: http://arxiv.org/abs/1408.5882

[42] L. Dong, F. Wei, S. Liu, M. Zhou, and K. Xu, "A statistical parsing framework for sentiment classification," *CoRR*, vol. abs/1401.6330, 2014. [Online]. Available: http://arxiv.org/abs/1401.6330

[43] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, 2011, pp. 151–161.

[44] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *CoRR*, vol. abs/1411.4555, 2014. [Online]. Available: http://arxiv.org/abs/1411.4555

[45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[46] Z. W. Department and Z. Wu, "Verb semantics and lexical selection," in *In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 1994, pp. 133–138.

[47] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, 2014, pp. 740–755.

[48] M. Hodosh, P. Young, and J. Hockenmaier, "Framing image description as a ranking task: Data, models and evaluation metrics," *J. Artif. Intell. Res. (JAIR)*, vol. 47, pp. 853–899, 2013.

[49] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," *ArXiv e-prints*, Feb. 2015.