IMAGE CAPTIONING USING ENCODER-DECODER ARCHITECTURE
CSE 253 DEEP LEARNING PA 4

Shujie Chen
A53303042
shchen@ucsd.edu

Felix Gabler
A53329004
fgabler@ucsd.edu

Yongchuang Huang
A53322973
yoh001@ucsd.edu

Jingpei Lu
A91033661
jil360@ucsd.edu

Yiran Xu
A53270264
y5xu@ucsd.edu

March 2, 2020

ABSTRACT

In this project, we worked on the image captioning problem with encoder-decoder neural networks. Different decoders and word embeddings are explored. The dataset for evaluation is a subset of COCO dataset [1]. The baseline model achieves 90.59 and 47.80 of BLEU-1 and BLEU-4, respectively. For Vanilla RNN, it achieves 89.65 and 40.75 of BLEU-1 and BLEU-4. By using GloVe embedding, we achieves 90.31 and 48.09 of BLEU-1 and BLEU-4, respectively.

1 Introduction

Image captioning is a task that requires the model to generate a caption that describes the contents of the image. This enables fast image-based keyword search. Recently, as the deep learning has achieved incredible performance on computer vision and natural language processing, we try to leverage the deep learning techniques in this project. A common architecture for this task is encoder-decoder architecture. An image first is encoded by a Convolutional Neural Network (CNN) to extract image features. Then, the features are fed into the decoder, which is usually a Recurrent Neural Network (RNN), to generate temporal sequence features for text generation.

Currently, a common backbone for image feature extraction is ResNet [2]. For sequences, we usually use Long-term Short Memory (LSTM) [3] or Vanilla RNN [4]. Furthermore, to encode words as features, word embedding is considered. Popular word embeddings include index embedding and semantic embedding such as Word2Vec [5] and GloVe [6].

In this report, we used ResNet-50 as our encoder, and we experimented with LSTM and Vanilla RNN as decoder on COCO Captioning dataset [1]. We also explored the difference between normal index word embedding and pre-trained GloVe embedding.

2 Related Work

COCO image captioning challenge. COCO dataset [1] is a famous dataset which contains images covering plentiful daily objects and scenes. In this project, we used a subset of it, which contains () training images and () validation images.

RNNs. RNN is first developed to handle temporal sequences, such as texts and natural languages. However, Vanilla RNN will result in gradient vanishing/exploding problem, so it is not proper to deal with long-term dependency. Later,

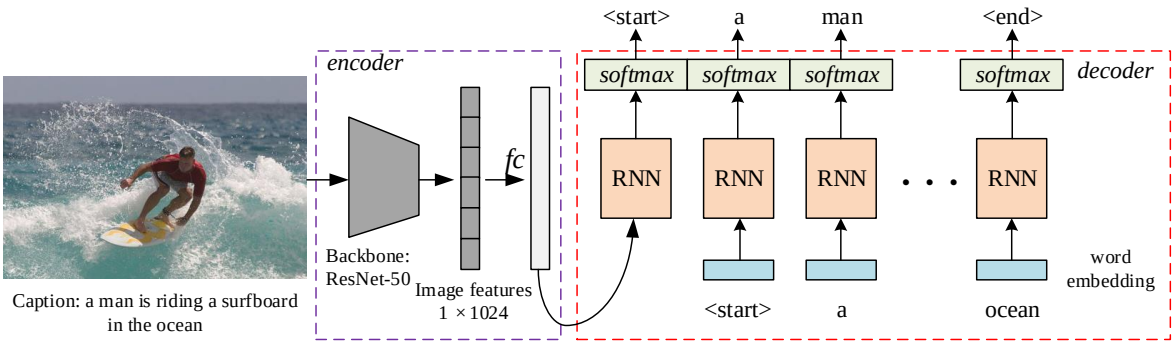


Figure 1: The architecture of our image captioning network.

Long-term Short Memory (LSTM) unit is proposed and it successfully learned long-term dependency by introducing memory cell and forget gates.

Word embedding. GloVe [6] uses window-based co-occurrence matrix to count the frequency of each word with its paired word. As a result, the distance in the mapped vector space between the words with similar meaning is close.

3 Methods

3.1 General

General encoder-decoder architecture. We show the overview of our networks in Figure 1. For CNN backbone, we are using ResNet-50 [2] to extract features from the images. Then, after passing image features through a fully-connected layer, we use RNN to deal with the temporal sequence. We explored different RNN structures including LSTM [3] and Vanilla RNN. The details of these two RNNs will be discussed in the following sections.

We show the architectures of LSTM and RNN in Figure 2. An LSTM cell can be expressed as:

$$\begin{aligned} i_t &= \sigma(W_{xh}x_t + W_{hi}h_{t-1}) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1}) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1}) \\ z_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot z_t \\ h_t &= o_t \odot \tanh(c_t)\,, \end{aligned} \tag{1}$$

where W's are weight matrices, $\sigma(z)$ is the sigmoid function, x_t is the input at time step t, h_t is the hidden state at t, o_t is the output at t, c_t is the cell state.

A vanilla RNN cell, however, is simply expressed by:

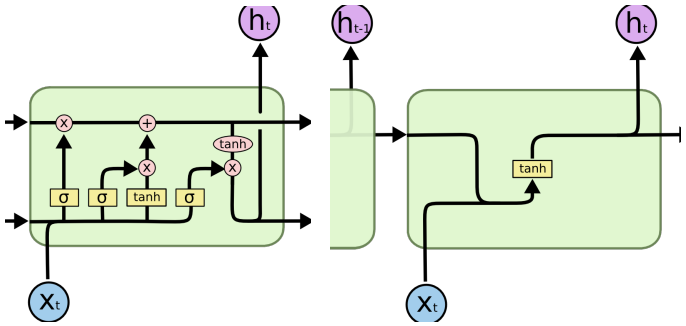
$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1})\,. \tag{2}$$

Word embedding. For Baseline and Vanilla RNN, we used the index to embed the words. In Section 3.4, however, we used GloVe [5] embedding.

Cross-Entropy loss and perplexity. In this project, we use Cross-Entropy to compute the loss between target caption embeddings and prediction logits. Cross-Entropy loss is

$$E = -\sum_{n=1}^{N}\sum_{c=1}^{C} t_c^n log(y_c^n)\,, \tag{3}$$

where N is the batch size, C is the number of classes, t_c^n is the target of n-th sample belonging to c-th class, y is the prediction from softmax layer. For language model, we usually also consider to use perplexity to evaluate its



(a) LSTM cell (b) Vanilla RNN cell

Figure 2: LSTM (left) and Simple RNN (right) cells.

performance, the perplexity can be computed simply by taking exponential term of Cross-Entropy:

$$Perplexity = \exp(E) = \exp(-\sum_{n=1}^{N}\sum_{c=1}^{C} t_c^n log(y_c^n)). \tag{4}$$

Training and caption generation. For image captioning, we train our model by using "Teacher Forcing", *i.e.* using the supervised signal from the training set at the current time step $target(t)$ as the input $x(t+1)$ to feed RNN, instead of using the output $y(t)$ generated by RNN. However, when we evaluate our model and try to generate captions, we use the output of current time step $y(t)$ instead to feed into RNN at the next step $t+1$. To obtain the word, we usually have two different ways: Deterministic and Stochastic.

For Deterministic way, we take the maximum output at each step as the input to the next step. By doing this we will have the same caption given the fixed image. The other approach is Stochastic generation, *i.e.* we sample from the probability distribution to get the word at each time step. We use a weighted softmax to get the distribution:

$$y^j = \frac{\exp(o^j/\tau)}{\sum_i^{|V|}\exp(o^i/\tau)}, \tag{5}$$

where o^j is the output from RNN, $|V|$ is the size of vocabulary, τ is the weight and usually called as "temperature". Larger τ we give, "softer" distribution we can get.

Evaluation metrics: BLEU-1 and BLEU-4. BLEU [7] evaluates the quality of text which is based on the matching similarity between the references and the output from the model. First, a modified n-gram precision metric is computed by

$$p_n = \frac{\sum_{c\in\{Candidate\}}\sum_{n-gram\in c} Count_{clip}(n-gram)}{\sum_{c'\in\{Candidate\}}\sum_{n-gram\in c'} Count(n-gram)}, \tag{6}$$

where $Candidate$ is generated words, $n-gram$ is the window size, $Count()$ computes the frequency of words while $Count_{clip}()$ only calculates the unique words between the candidate and the reference.

In addition to precision, BLEU also uses BP (Brevity Penalty) to deal with short sentences:

$$BP = \begin{cases} 1, & \text{if } c > r \\ \exp(1-r/c), & \text{if } c \le r \end{cases}, \tag{7}$$

where r is the length of reference and c is the length of candidate. Then BLEU is

$$\text{BLEU} = BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n), \tag{8}$$

where w_n is the weights for different window size. In this project, we use N to compute different n-gram. For example, if $N=4$ then we compute BLEU-4.

3.2 Baseline (LSTM) Model

The architecture of baseline model is shown in Figure 1, where RNN is replaced by LSTM. The baseline model contains two parts: for encoder, we used ResNet-50, the details are listed in Table 1. Note that the last fully-connected layer is replaced by a new one with output channels of 1,024; for decoder, a single layer LSTM is used. The dimensions of hidden state and input are 256 and 1,024, respectively. The output will be a vector living in $\mathbb{R}^{|V|}$ by passing through a fully-connected layer.

layer name	layer dimensions
conv1	kernel-size 7×7, 64 out-channels, stride 2
conv2_x	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
avgpool	output-size 1×1

Table 1: Detailed architecture of ResNet-50.

3.3 Vanilla RNN

In RNN models, the output of previous states is passed as the input onto the current state of the network, which makes RNN good for training sequential data. The encoder of our vanilla RNN is the same of that in the baseline model shown in Table 1. In the decoder part, we used a single-layer vanilla RNN.

3.4 Pre-trained Word Embeddings

What is different from the architecture of baseline model is just the decoder part. In the experiment of using Pre-trained Word Embeddings, we use Glove [6] pre-trained vectors as input for the network, instead of using words as input by converting them to a fixed-size feature embedding. Specifically, the weights of the embedding layer are initialized with a matrix containing Glove pre-trained vectors for all words in our vocabulary and frozen, i.e., unchangeable, while training.

4 Results

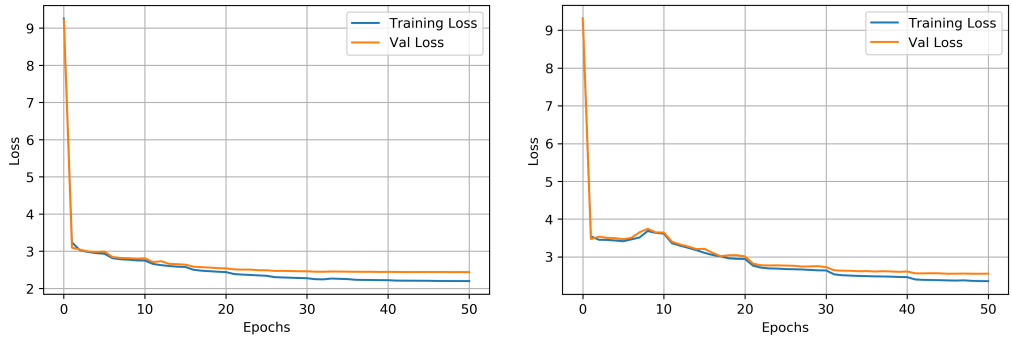
4.1 Data setup

We used a subset of COCO captioning dataset [1] to evaluate our model. We split the training set into training data (90%) and validation data (10%). We used the original validation set as test set for evaluation.

4.2 Baseline (LSTM) Model

To train our baseline model, we initialized the learning rate $\alpha = 0.01$ and it will be divided by 2 in every 5 epochs. We trained our baseline model for 50 epochs, but the lowest validation loss is achieved at epoch 48. Since the image sizes in the training set are different, we center-cropped the images with a size of 448×448 to keep the original as much as we can do (we noticed that the a common size in the training set is $\sim 640 \times 400$). The batch size is 128. Caption padding is used for different captions. The training loss and validation loss are shown in Figure 3(a). Note that we displayed the losses of untrained model at Epoch 0. **The test loss is 2.4249 and perplexity is 11.3011.**

We report BLEU-1 and BLEU-4 scores with respect to different generation methods in Table 2. There is a range $\tau \in [0.1, 0.2]$ where Stochastic generation achieves slightly better results than Deterministic way. The reason for this is



(a) Baseline loss curves (b) Vanilla RNN loss curves

Figure 3: The training loss and validation loss during training procedure. Epoch 0 indicates the case that the model has not started to be trained.

Table 2: BLEU-1 and BLEU-4 scores for baseline model, where τ represents the temperature.

	Deterministic	τ=0.1	τ=0.2	τ=0.3	τ=0.7	τ=1.0	τ=1.5	τ=2
BLEU-1	90.43	**90.59**	90.48	90.31	89.88	87.44	70.24	47.37
BLEU-4	47.71	**47.80**	47.15	46.66	39.35	27.18	8.04	4.32

that with a small τ, we sharpened the possible words with higher probability. This strategy in some cases could be better because we give some opportunity to other words that also have fair possibility compared to the highest one to be picked out. Some possible errors are avoided. We show a qualitative result of the first image in the test set to illustrate this problem. In Figure 4, the caption generated by Deterministic way is "a man riding a surfboard on a river with a surfboard". However, there is no river in the image. Compared to this, the Stochastic caption "a man riding a surfboard on a snowy day" makes more sense.

4.3 Vanilla RNN

In the implementation of vanilla RNN model, we used the same set of hidden units, optimizer, learning rate and other hyper parameters. The output of vanilla RNN is shown in Table 3. The loss curves are shown in Figure 3(b). **The test loss is 2.5237 and perplexity is 12.4747.**

Compared to the baseline model, vanilla RNN has both a higher test loss and a higher perplexity score, which indicates a worse performance than the baseline model. Also, from Figure 3, both training and validation loss of Vanilla RNN are higher than the baseline model. Therefore, it is true that Vanilla RNN needs more epochs to achieve a similar result

References:
A skier is high up in the air over a snowy hill

Generated Captions:
Deterministic:
a man riding a surfboard on a river with a surfboard

Stochastic $\tau = 0.1$:
a man riding a surfboard on a snowy day

Figure 4: Baseline: captions generated by Deterministic way and Stochastic way, respectively.

Table 3: BLEU-1 and BLEU-4 scores for vanilla RNN model, where τ represents the temperature.

	Deterministic	τ=0.1	τ=0.2	τ=0.3	τ=0.7	τ=1.0	τ=1.5	τ=2
BLEU-1	89.65	**89.67**	89.79	89.50	88.91	87.44	68.05	45.15
BLEU-4	**40.75**	40.61	40.31	39.53	33.16	27.18	7.60	4.24

from LSTM. This met our expectation since vanilla RNNs typically have vanishing or exploding gradient problems, whereas the additive interactions in baseline LSTM model improve the gradient flow. Another point is that, Vanilla RNNs cannot capture long-term information as well as LSTM.

Table 4: BLEU-1 and BLEU-4 scores for word embeddings, where τ represents the temperature.

	Deterministic	τ=0.1	τ=0.2	τ=0.3	τ=0.7	τ=1.0	τ=1.5	τ=2
BLEU-1	90.34	90.31	**90.45**	90.42	89.84	87.46	69.14	44.45
BLEU-4	47.87	**48.09**	47.39	46.92	39.72	27.63	7.72	4.07

4.4 Pre-trained Word Embeddings

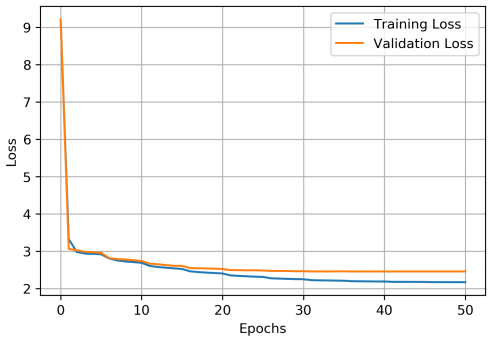


Figure 5: The training loss and validation loss during training procedure using pre-trained word embeddings

To train the model using pre-trained word embeddings, we use the same set of hyper-parameters in order to compare the results with that of baseline model. We used **GloVe**, the pre-trained word vectors, for word representation, and its data **Common Crawl** (42B tokens, 1.9M vocab, uncased, 300d vectors) to represent all the words in our caption. The reasons for using **GloVe** are that it takes the advantages of the NNLM and word2vec and that it is able to fit the differences between co-occur of different word-pairs. We trained the model for 50 epochs, but the lowest validation loss is achieved at about epoch 22 as shown in Fig.4.4, the figure of the loss curves. The BLEU-1 and BLEU-4 scores under different generation methods are shown in Tabel 4. Using Deterministic generation method, **the test loss is 2.4357 and perplexity is 11.42381**.

Compared to the baseline model, the model with pre-trained word embeddings has a slightly higher test loss and, of course, a higher perplexity score. However, for BLEU scores, the model with pre-trained word embedding performs slightly better than the baseline. This is because cross-entropy loss and BLEU score are not strictly equivalent. The reason for that pre-trained word embedding is better than word index embedding is, GloVe considers the contextual information when it was trained by computing co-occurrence matrix, while word index simply encodes the words by their indices, which ignores the the context. As a result, the model with GloVe leverages the semantic meanings of each word and global representation of a sentence.

5 Conclusion

In this project, we compared the performance among baseline (LSTM decoder), Vanilla RNN and the model with pre-trained word embeddings. For LSTM vs Vanilla RNN, LSTM performs better because it captures long-term information. For baseline vs the model with pre-trained word embeddings, the latter is better since GloVe uses global

contextual information. We also compared the generation ways between Deterministic and Stochastic. In a specific range, Stochastic way is better.

6 Contribution

Shujie Chen Implemented vanilla RNN model.

Felix Gabler Implemented code for training / vocabulary / data loading.

Yongchuang Huang Implemented vanilla RNN model and pre-trained word embeddings.

Jingpei Lu Implemented pre-trained word embeddings.

Yiran Xu Implemented the baseline model and worked on code for training / vocabulary.

References

[1] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[2] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[5] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[6] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[7] Kishore Papineni et al. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318.