DSA5204: Deep Learning and Applications
Final Project

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention [1]

Joanna Khek Cuina	Huang Youqin	Ju Qiaodan	Wang Zhenhao	Xu Shu	Yang Changyong
A0235908X	A0236138E	A0235984N	A0236054L	A0236043N	A0236000A

1 Background Information

If we were to teach young children a language, we would most likely show them pictures of various objects and their respective names. By associating the image and the text, this is exactly what image captioning is about. Image captioning is the idea of taking an image and then producing a sentence that describes the image. Fang et al. [2] first trained a detector model using object detection methods and applied a language model trained on captions to the detector. Xu et al. [1] improved upon the object detectors method by proposing a framework that does not explicitly use object detectors but instead learns latent alignments from scratch thus allowing the model to learn abstract concepts.

A stop sign is on a road with a mountain in the background.

In the example on the left, it is not sufficient to detect a stop sign exactly in this box. We would require not just understanding the object but also understanding the relationships between the objects in the image and to express it in a natural language.

The concept of attention allows for salient features to dynamically come to the forefront as needed [3]. This is advantageous as images are often cluttered and by focusing on the relevant portions of the image, the accuracy of the captions generated can be improved. In particular, Xu et al. employed two variants of attention - "soft" attention mechanism and "hard" attention mechanism.

Image captioning can help to solve many real world problems and one example is providing assistance to the blind, which is the focus of our project.

2 Technical Details

There are three main parts to the model. First, a convolutional neural network (CNN) that extracts features from the images. Second, an attention mechanism that weights the image features and lastly, a recurrent neural network (RNN) that generates the captions to describe the weighted image features.

2.1 Encoder: Convolutional Feature Map

The model takes a raw image of size 224 x 224 x 3 as its input. The image is then processed using CNN to retrieve features. In an encoder-decoder model[4], CNN acts as an encoder. An encoder reads the entire input sentence and captures some contextual information about the input sentence into what is known as a context vector. In other words, an encoder provides a good representation of the image.

The CNN encoder used in the paper is the VGGNet16[5] architecture which is pre-trained on the ImageNet[6] dataset. The encoder processes an image through sequential convolution layers using stacks of 3 x 3 filters with stride size 1 followed by max pooling performed over 2 x 2 pixel window with stride size 2. The final encoding used is the 14 x 14 x 512 feature map produced by the fourth convolutional layer before max pooling. Unlike previous research works, Xu et al. used features from the lower convolutional layers to retain the correspondence between features and the 2D image.

The output vector of our encoder is the flattened version of this feature map which is referred to as the annotation vectors and it has a dimension of 196 x 512 (L x D). In other words, there are 196 vectors each of which is a 512 dimensional representation corresponding to a part of the image. This output will be fed into our decoder model.

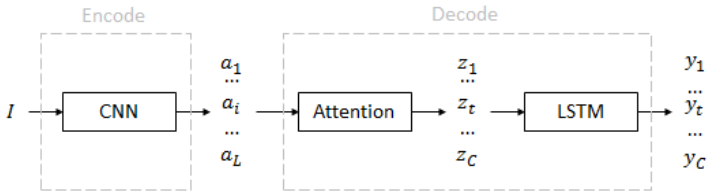


Figure 1: The attention decoder framework. Here, $\mathbf{a}_i$ refers to the annotation vector from our encoder. $\mathbf{z}_i$ refers to the context vector and $\mathbf{y}_i$ refers to our caption where C is the length of caption

2.2 Decoder with Attention: Recurrent Neural Network

The decoder takes the output of our encoder as input and generate a caption word by word through the sequence to sequence model, Long-Short Term Memory (LSTM) [7]. When the model is trying to generate the next word of the caption, this word is usually describing only a part of the image. Using the whole representation of the image to condition the generation of each word cannot efficiently produce different words for different parts of the image. The attention mechanism can help to solve this problem as it determines which sections of the image are more relevant by assigning weights to the different regions. The higher the weight for a pixel, the more relevant it is for the word to be output at the next timestep.

2.2.1 Attention Mechanism

The process of assigning weights can be described by (1) and (2) below. Here, e_{ti} is the output score of a feedforward neural network described by the function f_{att}, which takes in the annotation vector $\mathbf{a}_i$ and the previous hidden state $\mathbf{h}_{t-1}$ as input. The attention weights αi of each annotation vector are calculated by normalizing the output score of e_{ti}

$$e_{ti} = f_{att}(\mathbf{a}_i, \mathbf{h}_{t-1}) \tag{1}$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{L} exp(e_{tk})} \tag{2}$$

The two attention mechanisms proposed by Xu et al. are the stochastic "hard" attention and the deterministic "soft" attention. In soft attention, the main relevant region consists of different parts of different sub regions (See Figure 2) Areas with higher attention are brighter in the picture. As soft attention is fully differentiable, gradients can be calculated easily through backpropagation. In hard attention, the main relevant region consists of only one subregion. As hard attention is non-differentiable, to perform gradient descent correctly in the backpropagation, we require more complicated techniques such as reinforcement learning to train. Once the weights are computed, the context vector z_t is computed by (3) where ϕ computes a soft attention weighted annotation vector.

$$\hat{z}_t = \phi(\{\mathbf{a}_i\}, \{\alpha_i\}) = \sum_{i=1}^{L} \alpha_{t,i}\mathbf{a}_i \tag{3}$$

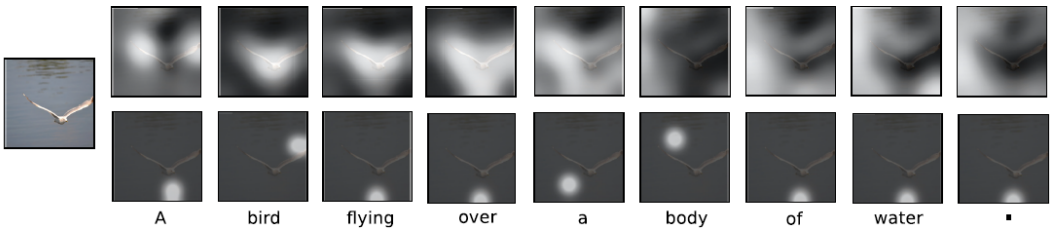


Figure 2: Soft attention (top row) and hard attention (bottom row)

2.2.2 Long Short Term Memory (LSTM)

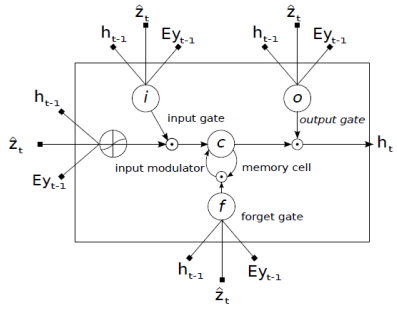


Figure 3: A LSTM cell

A LSTM network is used to produce the caption by generating one word at every time step conditioned on the context vector z_t, hidden state h_{t-1} and the previously generated word token y_{t-1}, which is multiplied by an embedding matrix E to give Ey_{t-1}. Inside the LSTM cell (See Figure 3), we have the update, forget and output gates. These gates modify the input and previous hidden states in some ways using the sigmoid activation to give the next hidden state.

2.2.3 Beam Search

The LSTM outputs a probability distribution over each word in the vocabulary for each word in the output sequence. A beam search decoder transform the probabilities into a final sentence of words. Beam search decoder takes in a user-specified parameter k which is known as the beam size. The top k words in a particular position is added to our candidate sequence and this process repeats until the algorithm picks end <END> token.

3 Reproduction and Extension

3.1 Dataset

We have chosen to evaluate our method on the VizWiz Caption dataset [8], which describes images taken by people who are blind. The dataset includes 23,431 training images, 117,155 training captions, 7,750 validation images and 38.750 validation captions. Each image is paired with five captions. Due to limited computing power, we randomly sampled 6000 training images from the train set, which amounts to around 26689 training captions for our model training process. An example of the dataset is shown in Figure 4.

some sort of USDA choice beef sirloin steaks meant to eat
A box of four 6 oz USDA sirloin steaks.
A photo of Schwan's USDA choice Beef Sirloin steaks.
A box of USDA choice beef sirloin steak is on a counter.
A package of USDA Choice Beef Sirloin Steaks, containing four six-ounce steaks.

A tarot card labeled nine on top and wands on the bottom with a picture of nine wands in the middle.
A playing card showing "nine wands" is on a wooden surface.
Nine wands tarot card with a picture of nine wands over fire.
A trading card is lying on top of a wooden table.
A card with objects symbolizing the number nine placed near the edge of a wooden table

Figure 4: VizWiz dataset examples

Compared to the COCO dataset used in the paper, VizWiz dataset is much more challenging as the quality of the images and captions are not as good.

3.2 Data Pre-processing

As the captions are not normalized, we had to first convert it to lowercase, remove punctuations as well as remove words that are one character or less in length. As neural networks cannot work directly on text-data, we have to convert text-words into integer-tokens. To do this, we first created a vocabulary list which is a dictionary of words tagged to a unique index. Zero padding was also done on the word vector to make sure the vectors have equal length. We added "start" and "end" tokens to our captions so our RNN model would know the starting and ending point. The images are resized to 224 x 224 x 3.

3.3 Evaluating Metric

To evaluate the model's performance, we will use the BiLingual Evaluation Understudy (BLEU) [9] evaluation metric. BLEU compare n-grams of the generated caption and count the number of matches it has with the reference captions. However, due to the fact that the generated caption can include many repeated n-grams, we clip the total count of each n-gram by its maximum reference count. In addition to the BLEU metric, we will also evaluate the LSTM and GRU results based on the training runtime to have a better sense of the potential trade-offs.

3.4 Model Training

For our decoder, we added dropout layer of keep rate = 0.5 in an attempt to reduce overfitting issue. We also added a batch normalization layer to ensure the stability of our learning rate and hence speeding up training. Our chosen optimizer is Adam as it works well with sparse data [10].

3.5 Results and Implementation

For our reproduction, we use the derterministic soft attention method as the backpropagation seems to be more effective. We tried a total of six different architectures with LSTM as our decoder and used Beam Search to produce the final caption. For our extension, we decided to try Gated Recurrent Unit (GRU)[11] as well as greedy search. Greedy search is similar to beam search except that it simply take the highest probability word at each position in the sequence [12]. To speed up the training process, we fixed our batch size to be 64 and only the top 5000 words were used. Table 1 summarizes our approach.

Table 1: Summary of our reproduction and extension plan

	Used in Paper	Reproduction	Extension
CNN Encoder	VGGNet16	VGGNet16, EfficientNet-B0, InceptionNetV3, MobileNetV2, DenseNet121, ResNet101	
Attention	Soft and Hard Attention	Soft Attention	
RNN Decoder	LSTM	LSTM	GRU
Evaluation Metric	BLEU (Beam Search)	BLEU (Beam Search)	BLEU (Greedy Search) Training Time

3.5.1 LSTM vs GRU

The results of using GRU and LSTM decoder is shown in Table 2 and the training time comparison can be visualized in Figure 5.

Table 2: Model performance of LSTM vs GRU on batch size = 64 and top words = 5000

Architecture	Decoder	Train Time (s)	Beam Search (k=3)			
			BLEU-1	BLEU-2	BLEU-3	BLEU-4
VGGNet16	GRU	1610	40.50	33.05	27.49	23.94
	LSTM	1751	42.41	34.15	27.75	23.90
EfficientNetB0	GRU	1020	45.14	37.29	30.27	26.25
	LSTM	2513	39.29	31.44	23.89	20.09
InceptionNetV3	GRU	2978	41.38	33.20	25.65	21.87
	LSTM	3186	43.30	38.50	33.20	27.89
MobileNetV2	GRU	615	46.15	37.41	29.21	24.52
	LSTM	753	46.13	36.78	28.33	23.23
DenseNet121	GRU	848	49.40	34.14	20.80	14.06
	LSTM	867	43.33	29.92	18.14	11.90
ResNet101	GRU	891	44.13	36.83	30.20	26.49
	LSTM	1163	51.03	41.29	33.15	28.74

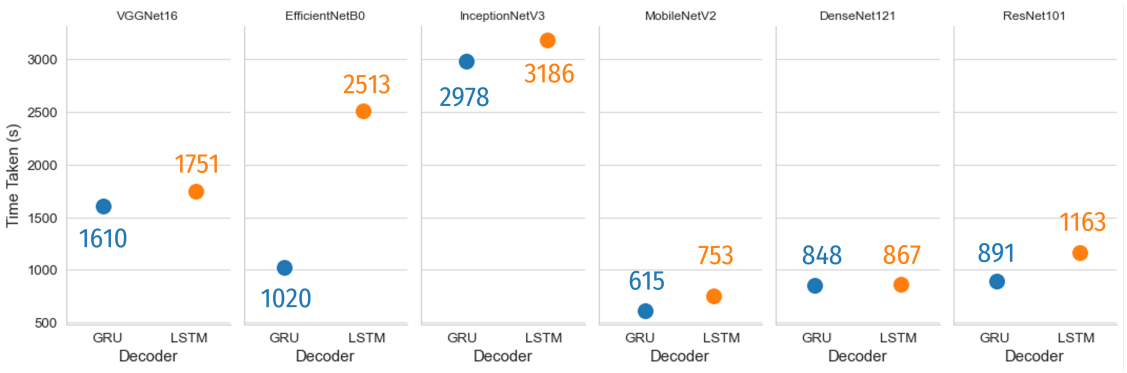


Figure 5: Comparison of Training Time between LSTM and GRU

We see that in general, GRU performs slighly better than LSTM and is computationally more efficient as seen in Figure 5. This could be attributed to the fact that there are only two gates in GRU compared to three gates in LSTM and GRU does not require internal memory hence it takes much less time to train.

3.5.2 Hyperparameter Tuning

We performed hyperparameter tuning on the GRU decoder using both beam search and greedy search to assess how the different batch sizes and top vocab words chosen would affect the model performance.

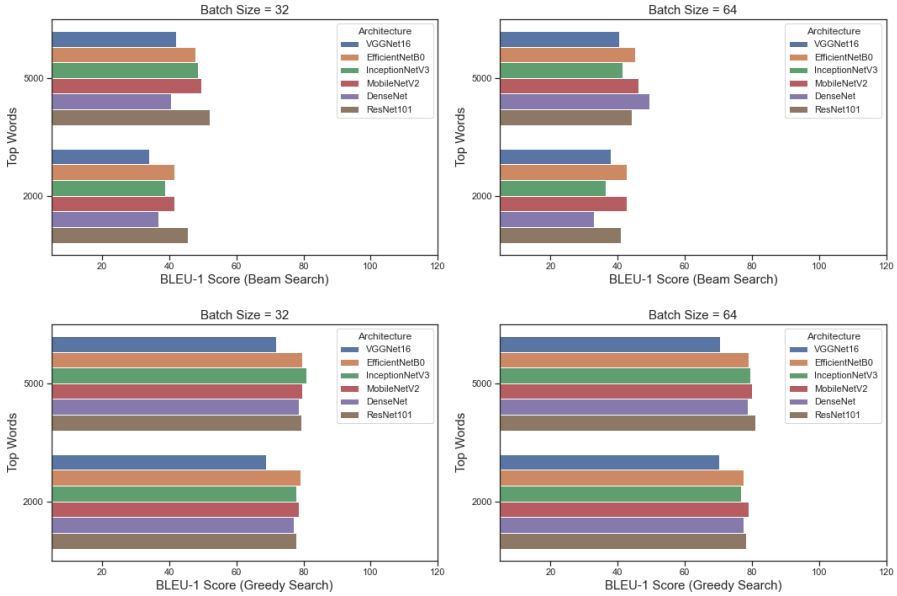


Figure 6: Comparison of BLEU-1 score using beam search (top) and greedy search (bottom)

Comparing BLEU-1 and BLEU-4 scores from both Figure 6 and Figure 7, we see that smaller batch size and larger pool of words led to a slightly better performance. Our model was able to capture the correct words individually but unable to form a good string of words as seen by the poor BLEU-4 score.

Figure 8 is an example of our greedy search generated caption using VGGNet16 on batch size = 64 and top words = 5000

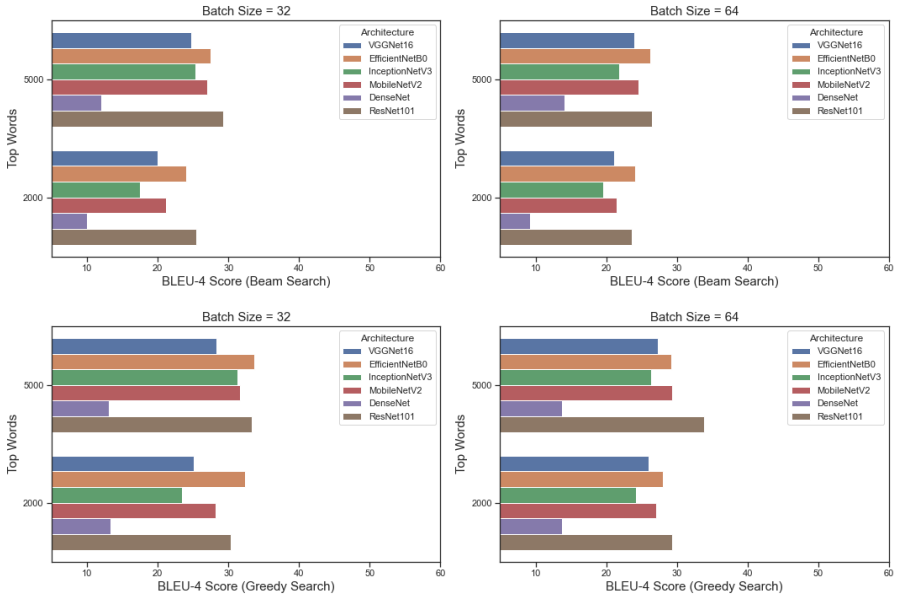
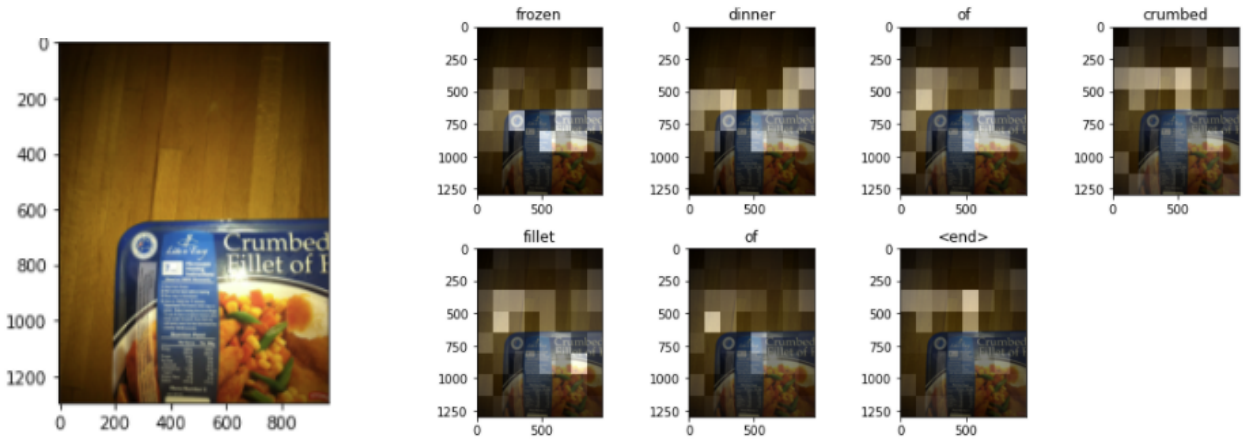


Figure 7: Comparison of BLEU-4 score using beam search (top) and greedy search (bottom)



(a) Image (b) Attention over image with caption

Figure 8: Example of our caption result

4 Conclusion and Future Work

In this project, we attempted to generate captions from images using the encoder-decoder model. We tried six different CNN architectures and ResNet101 seems to be the better architecture. In order to be more efficient, the attention mechanism was introduced as part of the decoder so that the model is able to determine relevant sections of the images. We see that in general, GRU is a much better decoder option as compared to LSTM due to its lower computation time and slightly better performance. Through hyperparameter tuning, we see that smaller batch size and larger pool of words led to better model performance. Our extension of greedy search have also shown much better performance on BLEU-1 score compared to beam search. However, results on BLEU-4 score are approximately the same.

Due to the limitation of time and computing power, we were not able to make use the full training set and some parameters such as learning rate and beam size were not tuned. For future work, we could perhaps use the entire training set and further tune our parameters to achieve better results. In addition, we could try other word embedding techniques such as Word2Vec or GloVe which could lead to better results due to the ability to capture semantic representations.

5 Contributions

Name	Student ID	Contributions
Joanna Khek Cuina	A0235908X	Data pre-processing, caption and image pre-processing, model training for VGGNet16
Huang Youqin	A0236138E	Model training for EfficientNetB0, code for greedy search algorithm
Ju Qiaodan	A0235984N	Model training for InceptionNetV3, code for beam search algorithm
Wang Zhenhao	A0236054L	Model training for MobileNetV2
Xu Shu	A0236043N	Model training for DenseNet121
Yang Changyong	A0236000A	Model training for ResNet101

References

[1] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[2] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1473–1482, 2015.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

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

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Danna Gurari, Yinan Zhao, Meng Zhang, and Nilavra Bhattacharya. Captioning images taken by people who are blind. In *European Conference on Computer Vision*, pages 417–434. Springer, 2020.

[9] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[12] Chenze Shao, Yang Feng, and Xilin Chen. Greedy search with probabilistic n-gram matching for neural machine translation. *arXiv preprint arXiv:1809.03132*, 2018.