Image Captioning Project Report

Alex Xie[1], **Jinqi Chen**[1], **Steven Lu**[2*]
School of Computer Science[1], Department of Statistics and Data Science[2]
Carnegie Mellon University
Pittsburgh, PA 15213
{alexx, jinqic, stevenl2}@andrew.cmu.edu

Abstract

Inspired by recent advances in image captioning as well as in language modeling and generation, we investigate a variety of decoder architectures for the COCO Captions 2014 dataset, including the Markov chain as a baseline model, recurrent neural networks, attention and transformer mechanisms. Our proposed transformer models improve upon the baseline models by +4 BLEU-4 score and create feasible captions that approach those created by humans. Our code is available on GitHub.[2]

1 Introduction

Image captioning has been an active field of research in deep learning where an algorithm takes in an image as input and generates a sentence that accurately describing the subject and actions within the image. While it is easy for humans, image captioning has remained a challenging problem for many years. Not only does the an image captioning algorithm have to recognize the subjects of the image, but it also has to recognize the background and the action that the subjects are taking. Thus, image captioning is a task that requires multiple complex steps in order to output a reasonable and well-rounded answer.

Given a deep learning model that works effectively, image captioning can help humans in many ways. For example, it is able to help blind users understand images: many applications of image captioning combine the algorithm with text-to-speech technology to describe the world in verbal languages. Other applications include social media, where it can be used to describe images uploaded or analyze current social media content engagement.

2 Related Work

Image captioning generally involves two parts: a visual module that can understand different regions and the background of images, and a language module that can generate human understandable texts using important information in the images. Since the rise of deep learning in recent years, many approaches have been proposed to train the two modules end-to-end. The proposed approaches can be roughly divided into the following [21]:

Visual Encoding Modules In order to understand the important regions of the images and their content, a carefully designed visual encoding module is very important. There are works that use CNN (Convolutional Neural Networks) that takes the whole image as input and output its feature embedding [12][23]. More recently, when attention mechanisms become more popular, there are work that attends to particular regions for different words instead of using the whole feature

[*] Alphabetical Order
[2] https://github.com/axie66/10701-project

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

embedding. The topdown attention method, for example, weigh each region for each word prediction [1]. Self-attention and transformers also demonstrates decent performance on image captioning tasks [8].

Language Modules In order to generate human readable, meaningful texts that accurately describe the key features of images, a well-designed natural language generation model is also important. LSTM (Long Short-Term Memory) models can capture long-term dependencies in sentences, thus is widely used as a language decoder for image captioning tasks [7][10]. In recent years, with the introduction of large pretrained transformer language models such as BERT (Bidirectional Encoder Representations from Transformers) [6] and GPT [18], many works have adopted transformer architectures for language decoding and achieved strong results [26][14][25][17].

3 Methods

3.1 Markov Chain

We consider the image captioning problem as a Markov chain problem. In the training processes, sentences were fed to the Markov Chain, which kept track of which words came after which in a typical sentence. For example, given the sentence: "Image Captioning refers to a task in machine learning." The word "Captioning" comes after "Image", "Task" comes after "a", and so on. This information is stored in a matrix, where each row represents is the current word and each column represents the next word. We use VGG16 to predict the main subjects of an input image. Then, these subjects were processed by the Markov chain, generating words one-by-one until the sentence ends.

The Markov chain model performed by far the worst out of all of the models, including the other baseline. The reason for this is the Markov assumption: formally, that $P(q_i|q_1, q_2, ...q_{i-1}) = P(q_i|q_1, q_2, ...q_{i-1})$. While this assumption is convenient for calculations, it means that when we are coming up with the next most likely word, we ignore all other words that previously appeared in the sentence except the previous one. An example is a picture of a woman holding an umbrella at a lake, labelled, "The Umbrella of Quitting after the First Mission." As the model performed so poorly, we decided to run a different baseline. This way, we can have a baseline to quantitatively compare the experiments to whilst maintaining the GRU/LSTM framework of the experiments.

3.2 Recurrent Neural Networks

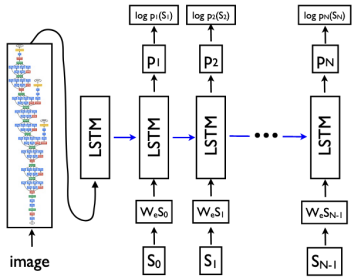


Figure 1: CNN-LSTM architecture for image captioning. Figure reproduced from [23]

To address the inability of the Markov model to use context prior to the last generated word, we take several recurrent neural architectures as additional baselines. The simplest of these is the Elman RNN, which maintains a hidden state h_t that is updated at each timestep based on the current input x_t and the previous hidden state h_{t-1}.

$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh}) \tag{1}$$

However, RNNs have been found to suffer from vanishing and exploding gradients [4], often making it difficult for them to learn long term dependencies. One solution to this problem is Long Short Term

Memory (LSTM) [9], which more selectively decides what information to keep and what information to forget. LSTMs are described by the following set of update rules:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{2}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{3}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{4}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{5}$$
$$h_t = o_t \odot \tanh(f_t \odot c_{t-1} + i_t \odot g_t) \tag{6}$$

where all W and b are learned weights and biases, respectively, x_t is the input at time t, and h_t is the hidden state at time t.

Additionally, Cho et al. introduce the Gated Recurrent Unit [5], a simpler and more computationally efficient LSTM variant that in many cases yields comparable or greater performance.

$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \tag{7}$$
$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \tag{8}$$
$$n_t = \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{t-1} + b_{hn})) \tag{9}$$
$$h_t = (1 - z_t) \odot n_t + z_t \odot h_{t-1} \tag{10}$$

For our overall image captioning model, we use an encoder-decoder structure as shown in Figure 1, with a pretrained CNN as our encoder and a recurrent model as our decoder [23]. Specifically, given an image, the CNN outputs a single vector representing its contents. This vector is passed into the recurrent model at the very first timestep (i.e. at time $t = 0$), after which the recurrent model iteratively generates the caption, ending with the special end of sentence token </s> (note that since we use the T5 tokenizer, there is no start of sentence token). During training, our model is described by the following:

$$x_0 = W_i \cdot \text{CNN}(I) \tag{11}$$
$$x_t = W_e S_t \qquad \forall t \in \{1, ..., N-1\} \tag{12}$$
$$h_t = \text{Decoder}(x_t) \qquad \forall t \in \{1, ..., N-1\} \tag{13}$$
$$p_{t+1} = \text{Softmax}(W_p h_t) \qquad \forall t \in \{1, ..., N-1\} \tag{14}$$

where I is the input image, $S = (S_1, ..., S_N)$ is a ground truth caption (where each S_i is a one-hot vector over the vocabulary), W_e is the word embedding matrix, W_p projects the hidden state h_t to the next token distribution p_{t+1}, and W_i projects the CNN image representation into the LSTM embedding space. We then minimize the negative log likelihood of each ground truth token $\mathcal{L} = -\sum_{t=2}^{N} \log p_t S_t$.

For our baseline models, we use a pretrained and frozen ResNet-50 as our encoder CNN and use Decoder $\in \{\text{RNN}, \text{LSTM}, \text{GRU}\}$. Additionally, we use weight tying between the embedding matrix and output layer weights (i.e. $W_e = W_p$) as a means of regularization.

Finally, as a means of bridging the gap between training, in which the output sequence is known, and test time, in which the output sequence must be generated from scratch, we use teacher forcing with a probability of 0.95. This means that 5% of the time, the input x_t is obtained not from the ground truth label from the previous step, but the model's own prediction from the previous step, i.e. $x_t = W_e S'_t$, where S'_t is a one-hot vector such that $S'_t[\arg\max p_t] = 1$.

3.3 Attention

Even with Long Short Term Memory and its variants, the recurrent decoders described in the previous section may eventually lose information about the input image during generation. Further, these models are only provided a single static representation of the image, which may not always be the

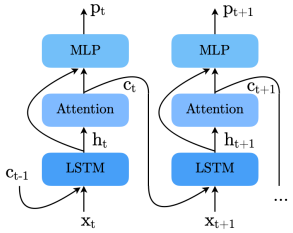


Figure 2: Our LSTM architecture with attention.

optimal representation at every step of generation. As such, inspired by Xu et al. [24], we experiment with adding **visual attention** to the decoder architectures described in the previous section.

For visual attention, rather than using a single vector representation for the image, we take the flattened and transposed feature maps outputted by the final convolutional layer of our CNN encoder. More specifically, if the final layer feature maps M have shape $C \times H \times W$, where C is the number of channels and (H, W) are the dimensions of the image, we first flatten the last two layers to obtain a matrix of shape $C \times (H \cdot W)$, then transpose the matrix to obtain sequence F with shape $(H \cdot W) \times C$. Hence, each vector within the sequence F can be thought of as encoding a certain region of the input image. Our overall model is describes by the following:

$$F = \text{CNN}(I) \tag{15}$$
$$x_t = W_e S_t \tag{16}$$
$$h_t = \text{Decoder}([x_t, c_{t-1}]) \tag{17}$$
$$c_t = \text{Attention}(W^Q h_t, W^K F, W^V F) \tag{18}$$
$$p_{t+1} = \text{MLP}([h_t, c_t]) \tag{19}$$

Note that W^Q, W^K, W^V are learned projection matrices, MLP refers to a 2-layer feed-forward network whose final layer weights are tied with the embedding matrix, and $[\cdot, \cdot]$ denotes the concatenation of two vectors. As before, we run experiments with Decoder $\in$ {LSTM, GRU} and use teacher forcing at a rate of 0.95. In addition, unlike the original paper which uses Bahdanau attention [3], we use scaled dot product attention, which is more computationally efficient [16]:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{20}$$

where $Q \in \mathbb{R}^{d_k}$ is the query vector, $K \in \mathbb{R}^{|F| \times d_k}$ is the key matrix, $V \in \mathbb{R}^{|F| \times d_v}$ is the value matrix, and d_k and d_v are the dimension of the keys and values, respectively. Intuitively, this can be thought of as blending together the features F into a single context vector, where each feature is weighted by its current relevance. Thus, the attention distribution (i.e. the output of the softmax) can be interpreted as what the model is "looking at" at each step of generation.

3.4 Transformers

In recent years, attention has shifted from recurrent models to purely attentional models, as transformer-based models have set new state-of-the-arts on a number of sequence modeling and generation tasks [22][6][18]. As such, we experiment with various transformer architectures for our caption model decoder.

Transformers follow an encoder-decoder architecture with separate encoder and decoder networks (not to be confused with the overall caption model's encoder, a CNN, or the overall model's decoder, the transformer itself). The encoder network is composed of a stack of N identical self-attentive (i.e.

attending to the previous layer outputs) encoder layers:

$$h_{\text{sub}}^{(l)} = \text{LayerNorm}(h^{(l-1)} + \text{MultiHeadAttention}(h^{(l-1)}, h^{(l-1)}, h^{(l-1)})) \tag{21}$$

$$h^{(l)} = \text{LayerNorm}(h_{\text{sub}}^{(l)} + \text{FFN}(h_{\text{sub}}^{(l)})) \tag{22}$$

where FFN (feed-forward network) denotes a two-layer MLP with dropout applied to each position of the input sequence, layer normalization is a learned transform that normalizes neuron inputs [2], and multi-head attention (hereafter abbreviated as MHA) is described as follows:

$$\text{MHA}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O \tag{23}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \ \ \forall i \in \{1, ..., h\} \tag{24}$$

Note that Q, K, V are the attention queries, keys, and values, W_i^Q, W_i^K, W_i^V, and W^O are learned projection matrices (each head has its own W_i^Q, W_i^K, W_i^V), and h is the number of heads. Note also that we use dot product attention as described in the previous section.

Meanwhile, the decoder network consists of another stack of N identical decoder layers, which are similar to encoder layers but have an added cross-attention component (denoted by $g_{\text{cross}}^{(l)}$ in the equations below) to attend to the encoder network output.

$$g_{\text{sub}}^{(l)} = \text{LayerNorm}(g^{(l-1)} + \text{MaskedMHA}(g^{(l-1)}, g^{(l-1)}, g^{(l-1)})) \tag{25}$$

$$g_{\text{cross}}^{(l)} = \text{LayerNorm}(g_{\text{sub}}^{(l)} + \text{MHA}(g_{\text{sub}}^{(l)}, h^{(N)}, h^{(N)})) \tag{26}$$

$$g^{(l)} = \text{LayerNorm}(g_{\text{cross}}^{(l)} + \text{FFN}(g_{\text{cross}}^{(l)})) \tag{27}$$

Note that since the decoder network autoregressively generates the output one token at a time, only tokens it has previously generated can be attended to; hence, masked multi-head attention (MaskedMHA), in which tokens after the current timestep are masked and cannot be attended to, is performed.

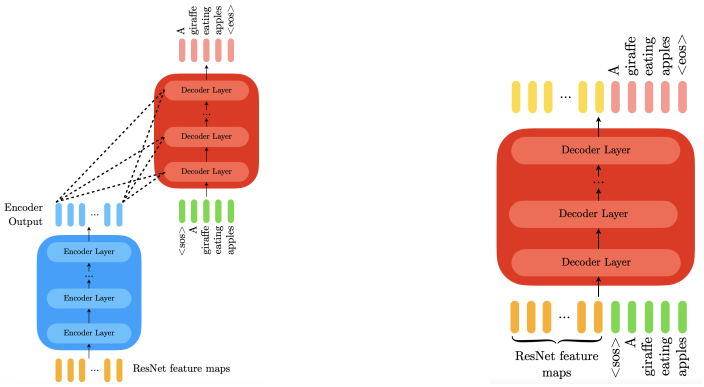
We experiment with two varieties of transformer models: full encoder-decoder transformers and decoder-only transformers (i.e. using only the decoder stack). Both versions take in the flattened and transposed feature maps (described in the previous section) as input. However, in the encoder-decoder model, the feature maps are passed into the transformer encoder, which outputs a further encoded version of the feature maps. The transformer decoder then autoregressively generates the caption, performing cross-attention at each step on the encoder output. Meanwhile, in the decoder-only model, the transformer decoder is fed at each step the feature maps concatenated with the previously generated tokens (note that there is no cross attention here since there is no encoder). We note that our decoder-only model strongly resembles the ClipCap architecture introduced by Mokady et al., with the differences being the choice of encoder and the fine-tuning of the entire decoder rather than only the prefix input [17].

Following the pretrain-finetune paradigm within NLP, we initialize our transformer models using pretrained weights, which are then fine-tuned on the image caption generation task. Specifically, for the full encoder-decoder transformer model, we use the weights and architecture of T5, a state-of-the-art transformer encoder-decoder model pretrained on a combination of denoising and language modeling objectives [20]. For our decoder-only model, we use the pretrained weights and architecture of GPT-2, a large transformer decoder-only model pretrained with a language modeling objective [19].

4 Experimental Evaluation

4.1 Dataset and Preprocessing

We use the COCO 2014 dataset (Common Objects in Context dataset), a commonly used dataset for various vision-related tasks. The objects in COCO dataset are **in context**, which means that the



(a) Encoder-decoder architecture (T5)

(b) Decoder-only architecture (GPT-2)

Figure 3: Transformer architectures for caption model decoder

pictures are within a natural setting, with other objects and actions being performed. We use the standard Karpathy split for training and evaluation. The training set contains 113,287 images with 5 captions each, and the validation set contains 5K images (we do not use the test set).

We preprocess the data by tokenizing the captions using the T5 tokenizer (based on the SentencePiece tokenizer) and normalize all images using ImageNet normalization.

4.2 Evaluation Metrics

In evaluating each model, we used the standard metrics for image captioning. Namely, we used BLEU-1, BLEU-2, BLEU-3, BLEU-4, METEOR, ROUGE, and CIDEr. These metrics each vary slightly, but the overall goal is to measure how close the generated caption is to the original, with higher scores indicating greater relevance. We pay special attention to BLEU, or the Bilingual Evaluation Understudy score, which measures this relevance to the ground truth (the given caption) in terms of n-gram overlap, on a scale of 0 to 1.

4.3 Experiments

For comparability, all models use the same encoder CNN, a pretrained ResNet50 model.[3] Further, we freeze the encoder CNN and only learn the decoder model. We optimize all recurrent models using the Adam optimizer [13] and optimize all transformer models using the AdamW optimizer, which differs from Adam in its weight decay fix that decouples the learning rate from the (L2) weight decay factor [15]. For recurrent models, we use a learning rate of 1e-3 and a weight decay factor of 1e-4, while for transformers, we use a learning rate of 2e-5 with a weight decay factor of 1e-5.

In addition, for our recurrent models, we use a batch size of 128, hidden dimension of 512, and two recurrent layers. Meanwhile, for our transformers, we use a batch size of 32 (due to hardware constraints). For the encoder-decoder transformer, we use the T5-small architecture (6 encoder layers, 6 decoder layers), while for the decoder-only transformer, we use the GPT-2 architecture (12 decoder layers). Note that we use the Huggingface `transformers`[4] implementations of T5 and GPT-2.

Due to computational constraints, we train our baselines for ten epochs, finding that our baseline recurrent architectures largely plateau after the tenth epoch. Meanwhile, we train our attentional recurrent models for twenty epochs as we find that the attention mechanism takes longer to learn. We train our transformer models for ten epochs, finding that fine-tuning for any longer does not improve results.

[3] `https://pytorch.org/hub/pytorch_vision_resnet/`

[4] `https://github.com/huggingface/transformers`

Decoder	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE	CIDEr
RNN	64.6	45.4	30.9	20.8	19.9	46.8	60.3
GRU	65.4	46.8	32.6	22.7	21.5	48.0	69.1
LSTM	65.0	46.4	32.2	22.4	21.4	47.9	67.2
GRU + attn.	68.1	50.1	35.7	25.4	23.2	50.1	79.7
LSTM + attn.	68.2	50.3	35.9	25.5	23.2	50.2	79.5
T5	**69.3**	**51.7**	**37.2**	**26.7**	**23.5**	**50.8**	**82.9**
GPT-2	68.3	50.3	35.7	25.1	23.3	50.2	79.5

Table 1: Quantitative results for our models. Note that for comparability, all models use the same frozen pretrained ResNet-50 encoder; we vary only the decoder model.

5 Results and Discussion

We report quantitative results (BLEU, METEOR, ROUGE, CIDEr) for all our models in Table 1. Additionally, we report per-epoch results for certain models in Figure 4. We also report qualitative captioning results in Table 5 and attention plots for a sample generated caption in Figure 2 to better understand the inner workings of visual attention.

5.1 Quantitative Results

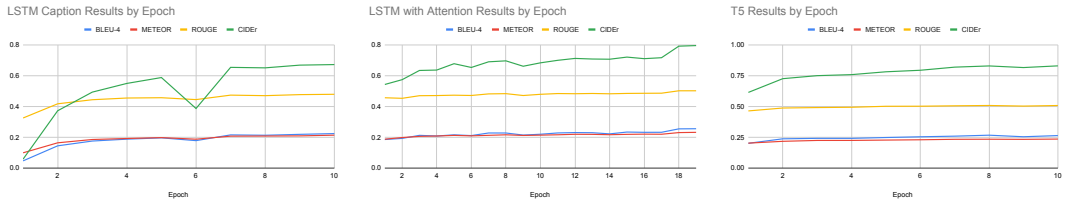
Of our baseline models, the RNN performs worst as expected; the simple RNN architecture struggles with long term dependencies and may lose information from the image vector inputted at the first timestep. The LSTM and GRU both perform more strongly, although they too often generate supbar captions, likely for the same reasons as the RNN model. This demonstrates the shortcomings of only using a static image representation for caption generation.

Our experiments with adding attention to our baseline models yield further improvements over our neural recurrent baselines, with the attention-augmented LSTM yielding a +3.1 increase in BLEU-4 score over the LSTM baseline. This indicates the effectiveness of attention for caption generation, as it allows for more expressive representations within the decoder that yield more relevant captions.

Finally, our strongest results are attained by our encoder-decoder (T5) transformer model. However, considering the significant complexity increase of our transformer models (our LSTM + attn contains 22M parameters, whereas our T5 model contains 63M) the increase the gains in performance are somewhat lower than expected. Further, our GPT-2 model underperforms our attentional models. This may be due to a disconnect between the pretraining objectives of these language models and the downstream image captioning they are fine-tuned on. Language models generally expect discrete inputs (i.e. words) whose embeddings have been learned to fit the model. However, for image captioning fine-tuning, we provide continuous inputs in the form of feature maps. While we do learn a MLP to map image inputs into the model's embedding space, it is possible that this MLP lacks sufficient capacity to capture all relevant information in the image or maps to an incompatible space. Thus, a more expressive mapping network (i.e. a transformer-based one) might yield stronger results.

5.2 Qualitative Results

We present generated captions for a sample image from the COCO validation set in Table 5. The caption results shown generally align with our quantitative results. The baseline models generate captions that are grammatically correct but semantically incorrect; while they correctly recognize that an individual is sitting, they ignore the bike and instead generate irrelevant words (i.e. “bench", “water"). Meanwhile, the recurrent models with attention generate captions that capture more relevant details (i.e. “motorcycle", “dirt bike") but are still somewhat factually inaccurate. T5 yields the most accurate and descriptive caption, mentioning both the motorcycle and the dirt road. However, even T5 misses smaller details such as the color of the man's helmet or the size of the moped, demonstrating the persisting gap between neural captioning models and human capabilities.



(a) CNN + LSTM (b) CNN + LSTM with attention (c) Transformer encoder-decoder

Figure 4: Results for selected models by epoch. We observe that our LSTM and transformer models plateau by around epoch 10, while our LSTM + attention model keeps improving up to around epoch 20 as it gradually learns the attention mechanism.

Decoder	Generated Caption
RNN	A man is sitting on a bench in the water.
GRU	A man and a woman sitting on a bench in the woods.
LSTM	A man sitting on a bench next to a bench.
GRU + attn.	A man standing next to a parked motorcycle.
LSTM + attn.	A man standing next to a man on a dirt bike.
T5	A man riding a motorcycle down a dirt road.
GPT-2	A man and a woman are standing next to a horse.
Ground truth	A man with a red helmet on a small moped on a dirt road.

Figure 5: Generated captions for a sample image from the COCO validation set.

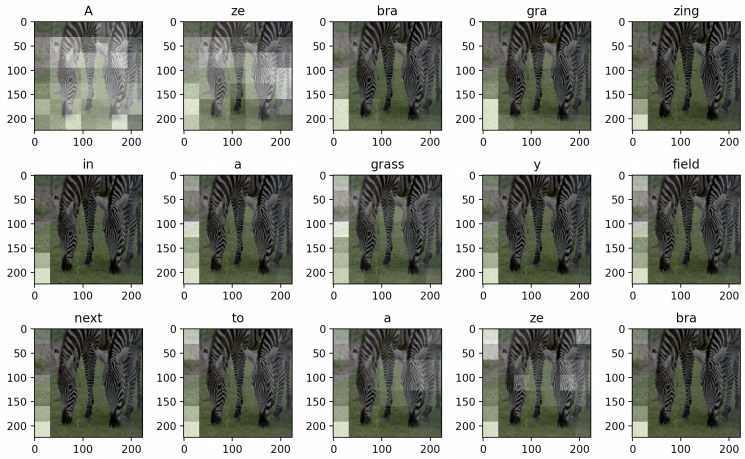


Figure 6: Attention plots at each timestep for a sample generated caption. The attention mechanism guides the decoder toward recognizing that there are two zebras in the image, allowing it to generate an accurate caption.

6 Future Work

In the future, we aim to further improve results in this paper by exploring dense captioning as a future experiment. Introduced by Johnson et al. [11], dense captioning is the task of identifying multiple salient areas within an image separately, then combining them. Since the image is broken down into more parts to be identified and thus works well with objects in context, this approach would potentially produce better results. Another clear avenue for future work is the use of stronger attention mechanisms, such as attention-on-attention, grid-based attention, or region-based self-attention.

References

[1] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. Jan. 2015. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

[4] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66, 02 1994.

[5] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[8] S. Herdade, A. Kappeler, K. Boakye, and J. Soares. Image captioning: Transforming objects into words. *arXiv preprint arXiv:1906.05963*, 2019.

[9] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[10] X. Jia, E. Gavves, B. Fernando, and T. Tuytelaars. Guiding the long-short term memory model for image caption generation. In *Proceedings of the IEEE international conference on computer vision*, pages 2407–2415, 2015.

[11] J. Johnson, A. Karpathy, and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[12] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[14] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020.

[15] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[16] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics.

[17] R. Mokady, A. Hertz, and A. H. Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.

[18] A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. 2018.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[21] M. Stefanini, M. Cornia, L. Baraldi, S. Cascianelli, G. Fiameni, and R. Cucchiara. From show to tell: A survey on image captioning. *arXiv preprint arXiv:2107.06912*, 2021.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

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

[24] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul 2015. PMLR.

[25] P. Zhang, X. Li, X. Hu, J. Yang, L. Zhang, L. Wang, Y. Choi, and J. Gao. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5579–5588, 2021.

[26] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao. Unified vision-language pre-training for image captioning and vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13041–13049, 2020.