Implementing and Improving Show, Attend, and Tell

Isabella Mercado [1]

Abstract

This report presents an implementation and architectural improvements of the "Show, Attend and Tell" model for neural image caption generation with visual attention. I begin with a brief overview of the original architecture, which pioneered the use of attention mechanisms in image captioning. I then detail my implementation of this model, highlighting both similarities and differences with the original paper's approach. Furthermore, I describe several architectural enhancements made to the original implementation, including improvements to the encoder-decoder framework, attention mechanism refinements, and training methodology optimizations. My modifications aim to enhance the model's performance in generating accurate image descriptions while maintaining computational efficiency.

1. Introduction

The task of automatically generating natural language descriptions for images is a complex, but important challenge in computer vision and natural language processing. The seminal "Show, Attend and Tell" paper by Xu et al. (2015) introduced a novel approach to image captioning by integrating attention mechanisms with latent representations of images. The authors demonstrated both qualitatively, through visualizations of attention, and quantitatively, through BLEU Score measurements, the utility in guiding a model's focus to specific, relevant regions of an image when generating each word of a caption.

The original architecture combined a convolutional neural network (CNN) for image feature extraction with a long-short term memory network (LSTM) for textual sequence generation. As opposed to utilizing the feature vectors outputted by the final fully-connected layer of a CNN similar to previous approaches, the authors extract features from a lower convolutional network to retain spatial information that can leveraged by the attention mechanism. In this paper, I trained and evaluated a replication of the core architecture described in the original paper in order to establish a baseline reference point. From there, I extend beyond the methodology implemented in the original paper, improving the encoder, decoder, and training architectures to ultimately drive higher performance than the original paper.

[*]Equal contribution [1]University of Chicago, Chicago, USA. Correspondence to: Isabella Mercado <imercado@uchicago.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

2. Original Implementation Analysis

The base implementation follows the core architecture described in the original paper, which I trained and validated on the 2014 Microsoft COCO (Common Objects in Context) dataset with 82,783 images in my training set and 40,504 images in my validation set.

2.1. Encoder Architecture

The original paper employed a modified VGG-16 network as the encoder, removing the final fully connected layer to maintain spatial information in the feature maps. The baseline implementation replicates their work with a similarly modified VGG-19 network with a marginal difference of 3 convolutional layers differentiating the two.

The features extracted from the lower convolutional layer are representations of the inputted image that correspond mathematically with:

$$\mathbf{a} = \{a_1, \ldots, a_L\}, a_i \in \mathbb{R}^D \tag{1}$$

where $\mathbf{a}$ represents the set of L annotation vectors, each encoding a portion of the image. For VGG-19, $L = 196$ (corresponding to a 14×14 spatial grid) and $D = 512$.

2.2. Attention Mechanism

The attention mechanism in the implementation closely follows the soft attention approach described in the paper. For each hidden state h_t of the decoder, attention weights α_{ti} are computed for each spatial location i in the image feature map. These attention weights then serve to calculate the context vector $\hat{z}_t$, which is a weighted sum of annotation vectors:

$$e_{ti} = f_{att}(a_i, h_{t-1}) \tag{2}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{L} \exp(e_{tj})} \tag{3}$$

$$\hat{z}_t = \phi(\{a_i\}, \{\alpha_i\}) = \sum_{i=1}^{L} \alpha_{ti} a_i \tag{4}$$

Although the details were not explicitly identified in the original paper, the attention block inputs the previous hidden state and current annotation vector into respective linear layers. It concatenates the results and feeds to a tanh function to generate e_{ti}.

$$f_{att} = \tanh(W_{ht} h_{t-1} + b_{ht} + W_a a_i + b_{ai}) \tag{5}$$

Additionally, the implementation includes a gating scalar β_t that modulates the context vector before it is fed into the LSTM, allowing the model to adaptively weight the importance of context vector at each time step.

$$\beta_t = \sigma(W_\beta h_{t-1} + b_\beta) \tag{6}$$

Although this adaptive attention technique was not mentioned in the original paper, but is recognized to improve performance.

2.3. Decoder Architecture

The decoder utilizes an LSTM network to produce captions by sequentially generating words given the context vector, previous hidden state, and previously generated words. At each time step t, the model computes:

$$i_t = \sigma(W_{ix} x_t + W_{ih} h_{t-1} + W_{ic} c_{t-1} + b_i) \tag{7}$$

$$f_t = \sigma(W_{fx} x_t + W_{fh} h_{t-1} + W_{fc} c_{t-1} + b_f) \tag{8}$$

$$o_t = \sigma(W_{ox} x_t + W_{oh} h_{t-1} + W_{oc} c_{t-1} + b_o) \tag{9}$$

$$g_t = \tanh(W_{gx} x_t + W_{gh} h_{t-1} + b_g) \tag{10}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{11}$$

$$h_t = o_t \odot \tanh(c_t) \tag{12}$$

where x_t is the concatenation of the word embedding and the gated context vector $\beta_t \cdot \hat{z}_t$. Just as in the original paper, the output word probability is calculated using a deep output layer.

3. Architectural Improvements

My enhanced implementation includes several architectural improvements which serve to increase the efficiency and accuracy of my model.

3.1. Modern CNN

My advanced implementation leverages the final convolutional layer of the ResNet-101 network (obtained by removing the final layer and penultimate average pooling layer). Although both were trained on the substantial ImageNet dataset, the VGG model utilizes a simple sequential structure with 16 convolutional layers and 3 fully connnected layers, while the ResNet has 101 layers with residual blocks that allow information to bypass layers. As a result the ResNet is computationally more efficient and accurate, driving superior gradient flow and feature extraction. Mathematically, for an input image $I \in \mathbb{R}^{H \times W \times 3}$, my ResNet101 encoder produces feature vectors $F \in \mathbb{R}^{h \times w \times d}$ where $h = 7, w = 7, d = 2048$. These features are then reshaped to $F' \in \mathbb{R}^{49 \times 2048}$ before being passed to the LSTM.

3.2. Length-aware batch processing

The decoder sorts the ground-truth captions by decreasing length and continues generating captions only for samples whose current timestep is less than the length of their ground-truth caption. This technique ensures that each sample in a batch is processed only for the necessary number of timesteps, improving the efficiency of batch training for variable-length sequences. By focusing computation on active sequences, the model avoids unnecessary processing of completed captions, optimizing both computational resources and training time.

3.3. CNN fine-tuning

The original implementation as well as the models described in this paper use a fixed pre-trained CNN, however, my repository additionally integrates the option to fine-tune the final convolutional blocks of a specified CNN model. This could enable the CNN model to adapt the visual features to the specific task of image captioning and improve the accuracy of my entire pipeline.

4. Training Methodology Improvements

The original paper named multiple training techniques such as dropout, early stopping based on BLEU score, and evaluating models based on BLEU scores calculated on the validation set. The base model leverages all of these techniques, however, my enhanced version uses a variety of more advanced techniques to improve model performance.

4.1. Label Smoothing Loss

Label smoothing is a regularization technique used to improve the generalization of neural networks. Instead of assigning a probability of 1 to the correct class and 0 to all others, label smoothing assigns a slightly lower probability

to the correct class and distributes the remaining probability mass across the other classes. Unlike standard cross-entropy loss, label smoothing discourages the model from assigning extremely high probabilities to training examples, making it less susceptible to overfitting the training data and better at generalizations.

In my implementation, I use a label smoothing factor of 0.1. The loss is calculated as a combination of the negative log-likelihood of the correct class and the average log-likelihood of all classes, weighted by the smoothing factor. This is mathematically represented as:

$$\mathcal{L} = (1 - \epsilon) \cdot \text{NLL} + \epsilon \cdot \text{Smooth\_Loss} \tag{13}$$

where ϵ is the smoothing factor, NLL is the negative log-likelihood of the correct class, and Smooth_Loss is the average log-likelihood of all classes.

4.2. Teacher forcing

Teacher forcing is a training technique used in sequence-to-sequence models, such as those used for language generation tasks. During training, at each step of sequence generation, the model receives the correct token from the ground-truth sequence as input for the next step, instead of using the token it predicted in the previous step. By providing the correct context at each step, teacher forcing helps the model learn the correct sequence patterns more quickly, leading to faster convergence, greater stability, and improved learning.

4.3. Gradient clipping

To prevent exploding gradients, I applied gradient clipping:

$$\nabla_\theta \mathcal{L} = \min\left(\max\left(\nabla_\theta \mathcal{L}, -c\right), c\right) \tag{14}$$

where c is the clipping threshold.

4.4. Learning rate scheduling

I implemented step decay for the learning rate:

$$\eta_{epoch} = \eta_0 \cdot \gamma^{\lfloor \frac{epoch}{step\_size} \rfloor} \tag{15}$$

where γ is the decay factor and $step\_size$ is the number of epochs after which to decay the learning rate.

4.5. Regularization

I introduced dropout in the decoder to prevent overfitting. During training, dropout randomly sets a fraction of the neurons in a layer to zero at each forward pass. This fraction is determined by a hyperparameter called the dropout rate, typically denoted as p:

$$p_t = \text{dropout}(h_t, p) \tag{16}$$

$$y_t = \text{softmax}(W_o p_t + b_o) \tag{17}$$

Dropout helps prevent overfitting to the training data and improve the generalizability to new data.

5. Results and Discussion

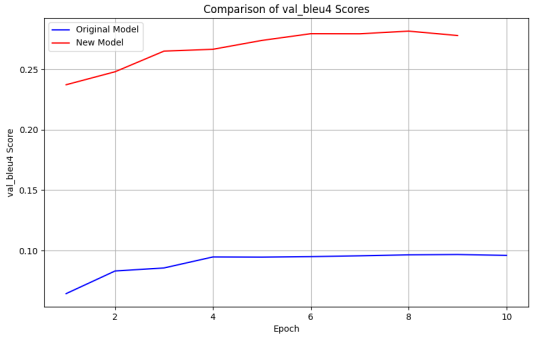

Figure 1. BLEU-4 Score Comparison.

As visible in Figure 1, the previously described architectural improvements drove a remarkable improvement between the baseline model created based on the original paper's architecture and my enhanced model. The baseline model reached a peak BLEU-4 score of 0.08, while the enhanced model outperformed the original model with a BLEU-4 score of 28.77, an approximately 3.6x improvement. Similar advances are seen in BLEU-1, BLEU-2, and BLEU-3 (See Figure 2 in Appendix). The most remarkable feature of the improved architecture were the efficiency gains in training time.

I trained the baseline model for 10 epochs and the enhanced model for 9 epochs each on an H100 GPU. I stopped training the enhanced model prior to 10 epochs because, as reported by my early stopping indicator, the BLEU-4 stopped improving by epoch 8. The enhanced model took about 4 hours to train, significantly less than the 3 day period the original paper trained their model for, while still outperforming the paper's BLEU-4 score of 24.3. The baseline model took two hours longer with significantly worse performance. While observing marginal performance gains from including the RESNET-101 network and many of the other advanced techniques, training efficiency and quantitative model performance significantly improved from the incorporation of teacher forcing. However, I did not measure

the qualitative performance of an enhanced model without teacher forcing.

The increase in BLEU scores correspond to significant qualitative improvements in generated image captions. The baseline model outputted incoherent caption with frequent word repetitions, while the enhanced model produced highly descriptive captions for the same validation set images (See Figures 7 and 8 in Appendix). The same qualitative improvements are observed in the representations I generated of what the model "attended" to whilst generating a particular word in a caption (See Figures 3-6 in the Appendix). With the white areas indicating where the model attended to, the original model's attention appears uncorrelated with the word generated while the improved model's focus is on the areas associated with the word generated at that time step. These results indicate the remarkable improvements that can be made over the original paper's model with the incorporation of enhanced model architecture and training methodologies.

6. Conclusion

I have presented an implementation and enhancement of the "Show, Attend and Tell" model for image captioning. My modifications to the encoder-decoder architecture, attention mechanism, and training methodology address several limitations of the original implementation while maintaining the core insights of the original paper.

The flexibility to use modern CNN architectures, the improved decoder model, and the enhanced training procedures collectively represent a significant advancement over the base implementation. These improvements not only enhance model performance but also improve training efficiency and stability.

Future work could explore the integration of more advanced attention mechanisms or testing the encoder fine-tuning paradigm included, but not tested, in this paper's associated repository.

References

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 International Conference on Machine Learning, pp. 2048–2057. PMLR, 2015.

A. C. Wong. Show-Attend-and-Tell. GitHub repository, 2017. URL https://github.com/AaronCCWong/Show-Attend-and-Tell.

A. Appendix.

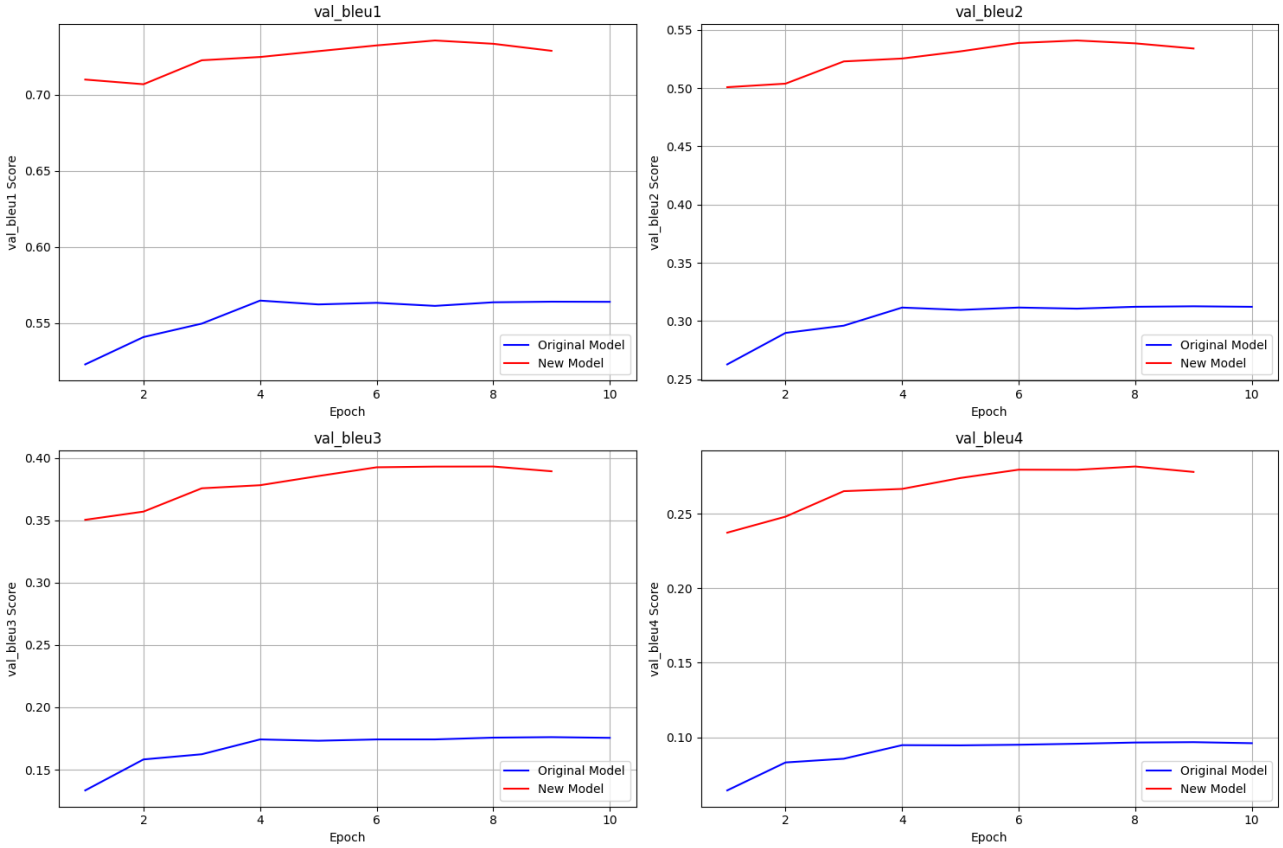

Figure 2. Comparing BLEU-1, BLEU-2, BLEU-3, & BLEU-4 scores across epochs for baseline and enhanced model.

Figure 3. Training Set Image for Baseline Model.

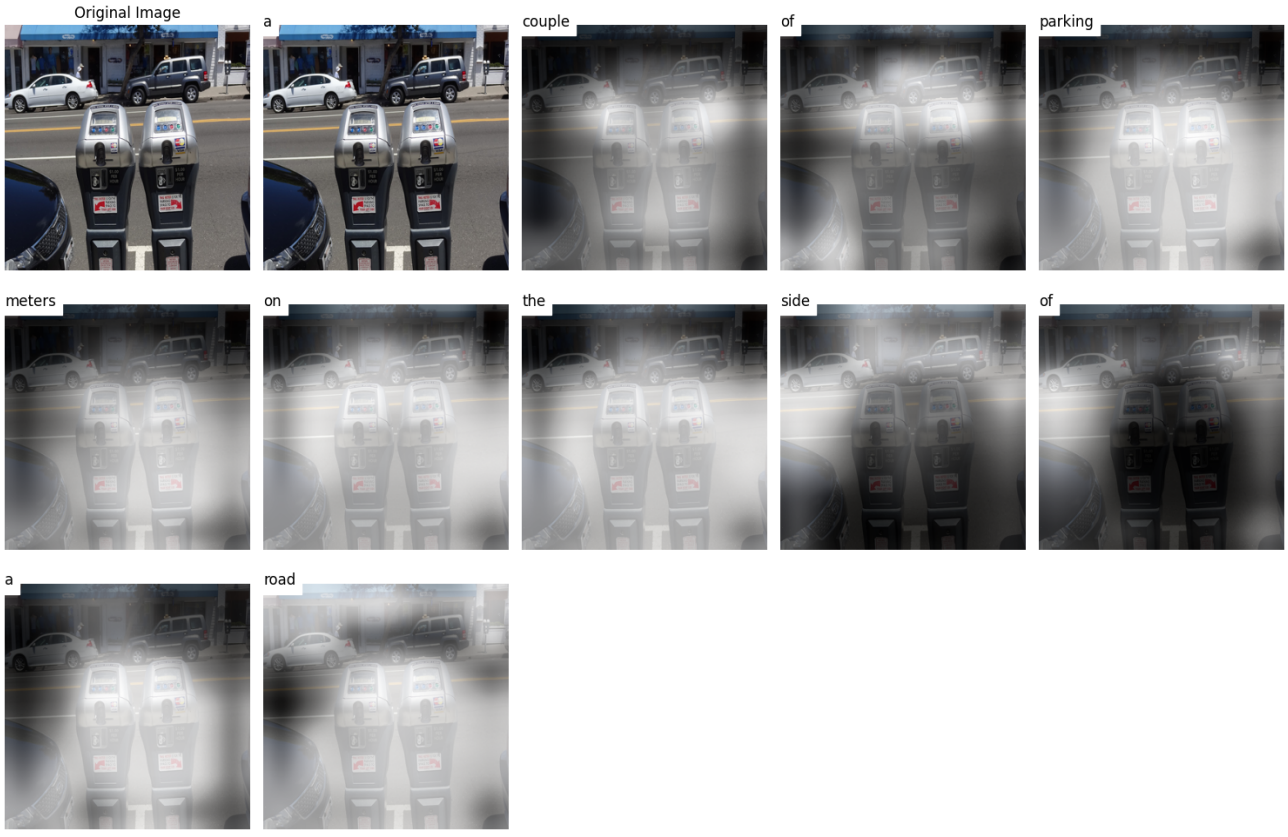

Figure 4. Training Set Image for Enhanced Model.

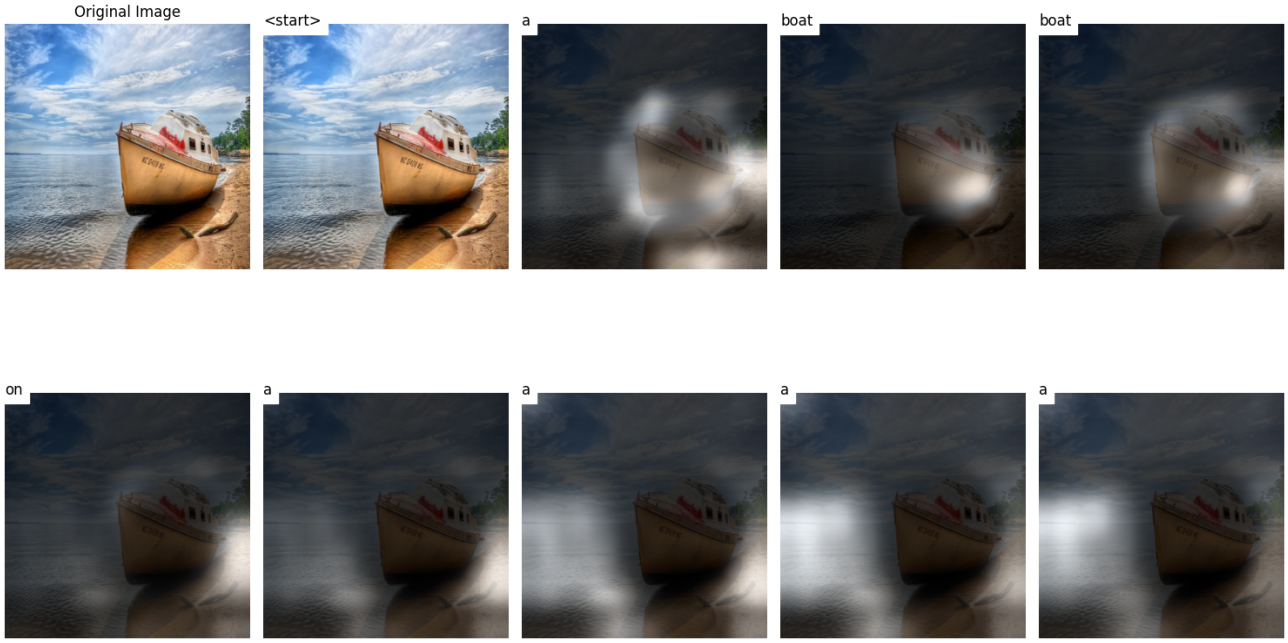

Figure 5. Validation Set Image for Baseline Model.

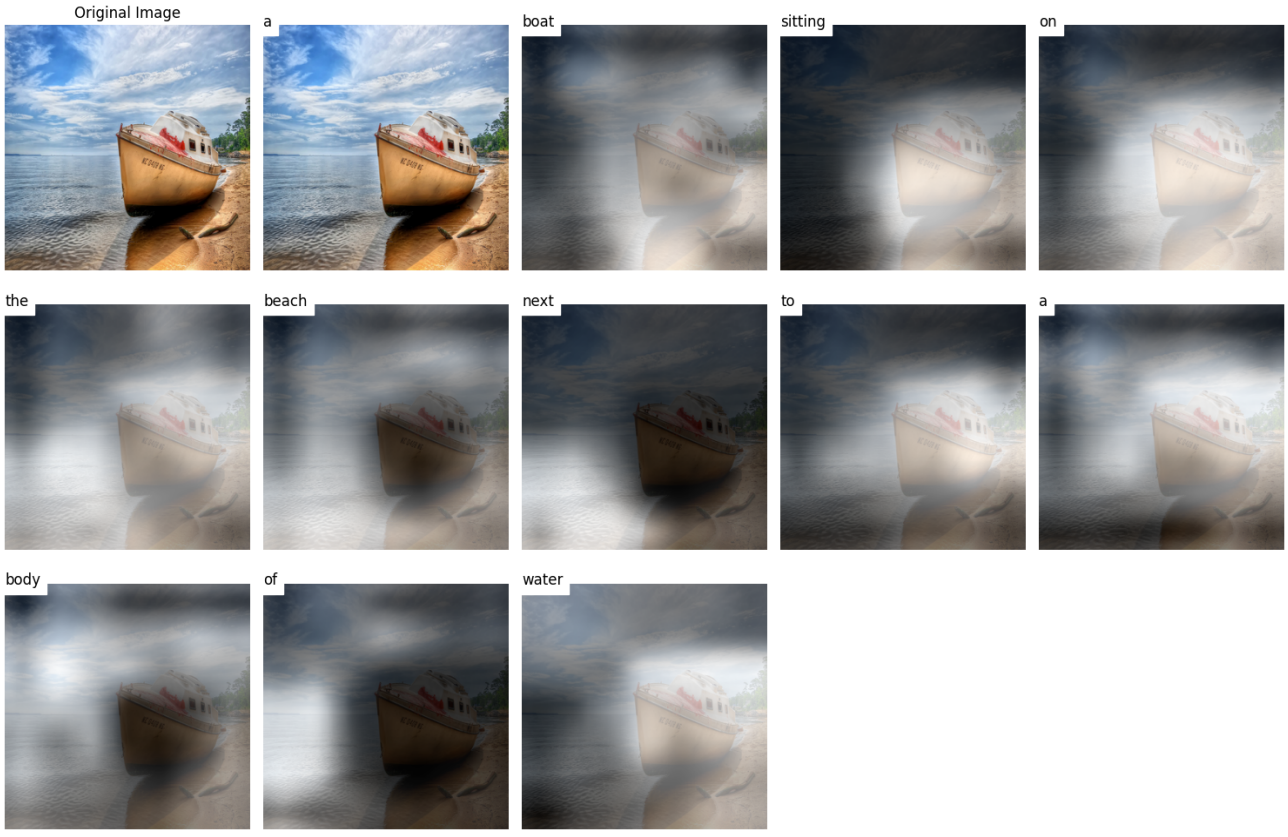

Figure 6. Validation Set Image for Enhanced Model.

```
Model: model_vgg19_10, Split: val
================================================

Image: COCO_val2014_000000137803.jpg
Caption: <start> a group of people a a a a
------------------------------------------------

Image: COCO_val2014_000000358658.jpg
Caption: <start> a living room with a a a a a
------------------------------------------------

Image: COCO_val2014_000000293452.jpg
Caption: <start> two group of on standing a a the
------------------------------------------------

Image: COCO_val2014_000000266250.jpg
Caption: <start> a cat is a a a a a a
------------------------------------------------

Image: COCO_val2014_000000397628.jpg
Caption: <start> a stop sign a a a a a a
------------------------------------------------
```

Figure 7. Validation Set Captions for Baseline Model.

```
Model: model_resnet101_9, Split: val
==================================================

Image: COCO_val2014_000000137803.jpg
Caption: a group of people standing on top of a sandy beach
--------------------------------------------------

Image: COCO_val2014_000000358658.jpg
Caption: a living room filled with furniture and a flat screen tv
--------------------------------------------------

Image: COCO_val2014_000000293452.jpg
Caption: a flock of birds standing on top of a sandy beach
--------------------------------------------------

Image: COCO_val2014_000000266250.jpg
Caption: a black cat sitting in front of a flat screen tv
--------------------------------------------------

Image: COCO_val2014_000000397628.jpg
Caption: a red stop sign sitting on the side of a road next to a street
--------------------------------------------------
```

Figure 8. Validation Set Captions for Enhanced Model.