Knowing Where to Reduce: Efficient Methods for Image Captioning

G034 (s1451292, s1451142, s1459898)

Abstract

Generating caption for images has shown promising results using the encoder-decoder architecture. We investigate the effect of having an adaptive over soft attention mechanisms in LSTMs. We implement the decoder with adaptive attention mechanism and train it on the MS COCO dataset. We then compare it with a soft attention LSTM on image captioning tasks. We also examine more efficient methods to train our model as a lot of computational resources and time is required to train. We replace the ResNet-101, with either a SE-ResNet-101 and Xception, to see whether we achieve speed up in training and get better performance. For our results, we found that our best adaptive attention model achieved a BLEU-4 score of 28.3, while our best soft attention model achieved a BLEU-4 of 21.9. We also achieve a slight speed up in training using SE-ResNet-101 over ResNet-101. When looking at the result qualitatively, the adaptive attention models didn't always produce better caption compared to soft attention. We also find that soft attention was better at attending to the salient parts of the image.

1. Introduction

Automatic caption generation of images is a challenging task in the field of computer vision and natural language processing. It involves producing meaningful texts for the images observed. These texts should flow naturally and be similar in content and grammar that humans would use to describe the images. Applications for image captioning include the automation of information extraction of images which would normally be done by human annotators. It can also be applied in the field of image retrieval, where the system is able to retrieve relevant images based on textual descriptions (Feng & Lapata, 2013).

There have been great advances in the field of automatic captioning. Systems are able to produce human-like descriptions in many cases. Most models use an encoder-decoder architecture to tackle the problem of image captioning. The encoder encodes the image into a vector representation, for instance with CNNs (convolutional neural network). The decoder, such as a RNN (Recurrent Neural Networks) takes the encoded vector of the images to produce a caption for them. More details will be put in **Section 3**.

Xu et al. (2015) produced state of the art result in 2015 with their architecture for image captioning. The attention mechanism was the main reason for their state of the art results. This attempts to describe images similarly to how humans would, such as by focusing their attention to the salient parts, rather than the whole image. Since then, many variations of the attention mechanisms have been proposed, and we experiment with some of the most promising ones, such as adaptive attention proposed by Lu et al. (2016).

One of the main bottlenecks is the training and computation time required for these models. Howard et al. (2017) with mobileNet and Chollet (2016) with the Xception, have made stark progress in the field of CNNs to reduce their size with negligible loss in accuracy and therefore increasing their parameter efficiency. We experiment with the Xception architecture in place of larger CNNs such as ResNet-101 (He et al., 2015) to see whether we can achieve similar performance with a speed up in training time. We also experiment with SE-ResNet-101 proposed by (Sutskever et al., 2014), which uses an attention-like mechanism for the CNN. While this may add more parameters to our model, It has been claimed to converge in fewer epochs; overall speeding up training. Lastly, another method to speed up training is decreasing the image resolution, we wish to investigate how much effect this has on performance.

Objective The aim of this project is to train and tune hyperparameters of neural networks. We then perform both qualitative and quantitative analysis, on the trained models. We use an encoder-decoder architecture for all our models. For the encoder, we want to utilize CNNs that use one of Xception , SE-ResNet-101 or ResNet-101 to extract features from the image. Then from the decoder, we use an LSTM (Long Short-Term Memory) with an attention mechanism (soft or adaptive) to predict the captions, based on the extracted features.

While implementation of stated attention mechanisms exists, to the best of our knowledge we haven't found anything in the literature that attempts to investigate performance using more efficient encoders and decreased image resolution. This paper addresses four **research questions**:

1. Does using adaptive attention have any improvement over soft attention proposed by Lu et al. (2016)?
2. How does using depthwise separable convolution, utilised by Xception (Chollet, 2016) instead of ResNet -101 (He et al., 2015) in the encoder stage affect performance (if at all) and whether training speeds up ? The aim is to see whether we can have a more

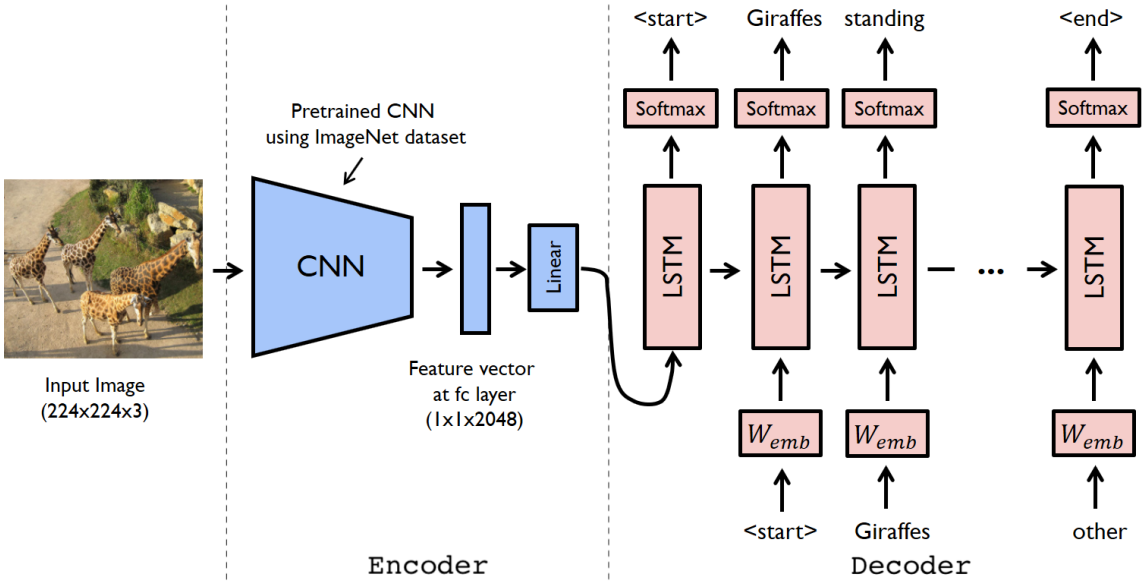

Figure 1. Example (Yun Jey, 2018) showing image captioning framework. We use the channels from the last convolution layer instead of the fully connected layers as our feature vectors.

efficient model, to increase performance.

3. Does adding "Squeeze-and-excitation" (SE) block (Sutskever et al., 2014) to the `ResNet` -101 (He et al., 2015),i.e (`SE-ResNet-101`), in the encoder stage improve performance (if at all)? Sutskever et al. (2014) claims to improve channel interdependencies at at minimal additional computational cost.

4. How much affect does decreasing the image resolution from 224×224 to 64×64 have on the performance of our models ?

In this report, we explore the data sets used for this task in **Section 2**. We will give a thorough explanation of the models, also introduce some concepts in **Section 3**. In **Section 4**, we will be describing the overall setup of our experiments and interpret the results. Next we discuss related work in the field of image captioning in **Section 5**. Lastly, we conclude the results of our findings in **Section 6**.

2. Data set and task

We use mainly `MS COCO`-2014(Lin et al., 2014) as our dataset to train our models. The dataset contains 81k training images and 41k for both the validation images and testing images. However, the annotations for the testing images are not publically available, we use a split created by Karpathy & Fei-Fei (2015). This has 111k training images with 5k images for both validation and testing. Each image is annotated with 5 different captions. This further increases the size of the training data set.

Since it is a large dataset to train on, we resized the images to a smaller resolution to reduce the time required to run each experiment. We pre-processed the images from `MS COCO` to 2 different sizes which are $[64 \times 64]$ and $[224 \times 224]$. We used the former to tune our hyperparameters and used the latter to train our model with the best hyperparameters.

`BLEU` (bilingual evaluation understudy) (Papineni et al., 2002) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. The score ranges from 0 to 1, with 1 being a perfect translation while 0 a total mismatch. We can use this translation metric since we can think of the caption generated by the machine as translating an image to a sentence. `BLUE-n` is the average number of n-grams we matched with the reference. The `BLEU` scores is the weighted sum of the n = 1 ,.., 4, `BLEU-n` scores. This is the most common automated metric for evaluating the quality of image captions.

- **Dataset:** `MS COCO`-2014 (Lin et al., 2014), Split: (Training-111k, Validation-5k, Testing-5k) (Karpathy & Fei-Fei, 2015)
- **Evaluation metrics:** `BLEU` (Papineni et al., 2002)

Concisely put, our models take as input a normalised raw image and output a caption $\mathbf{y} = \{y_1, \cdots y_T\}$, where each y_i is a word in our dictionary, and T is the length of the caption. More details are presented in the following section.

3. Methodology

As stated in the introduction we use an encoder-decoder architecture in all of our models. In this section, we explain the necessary background.

We briefly define some of the techniques we used in our project

- **Transfer Learning** Allows us to take a pre-trained `CNN`, rather than us having to train the network from scratch which can take many days when training on `ImageNet` (Huh et al., 2016).
- **SE block** Sutskever et al. (2014) introduces a new architectural unit, "Squeeze-and-Excitation" (SE) block, which is adding weights attentively to each and every feature map in the layer. Instead of fusing the spatial and channel information of an image which might cause loss of information on the channel-wise feature dependencies, it uses global information to selectively emphasize informative features and suppress less useful one in a computationally efficient manner.
- **Beam Search** Beam search, which has been introduced and described by (Graves, 2012) and (Boulanger-Lewandowski et al., 2013) for sequence-to-sequence models, is an algorithm that is better suited to generate the most probable sentence. It does it by keeping the k most probable sentences generated at every step and discarding the rest, rather than greedily picking the best at each time step, which could result in a sentence that might not be the best globally generated caption.
- **`Teacher forcing`** is a technique where we are supplying the ground-truth as the input at each decode-step during training, regardless of the word last generated. This is commonly used during training to speed-up convergence.
- **`GloVE-300`** is a commonly used word embedding weights in natural language processing tasks to project words into a 300-dimensional vector space. The appeal behind `GloVE`-300 is that similar words are closer to each other in the vector space.

3.1. Decoder: `LSTM`

While in theory vanilla `RNNs` should be able to capture long term dependencies, in practice the gradients which are back-propagated through time can still "vanish".
A `LSTM` is a special kind of `RNN`, capable of learning long-term dependencies proposed by Hochreiter & Schmidhuber (1997). `LSTM` are well suited for tasks based on time series data, such as predicting future stock prices given previous data and predicting the next word in a sentence given the previous words. This is because the network can sometimes pass on information from previous time steps unchanged so that it can learn from distant inputs.

Forget gate:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f)$$

Input gate:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i)$$

Output gate:

$$\begin{aligned} \mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\ \mathbf{c}_t &= \mathbf{f}_t \cdot \mathbf{c}_{t-1} + \mathbf{i}_t \cdot \tanh\left(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c\right) \\ \mathbf{h}_t &= \mathbf{o}_t \cdot \tanh\left(\mathbf{c}_t\right), \end{aligned} \tag{1}$$

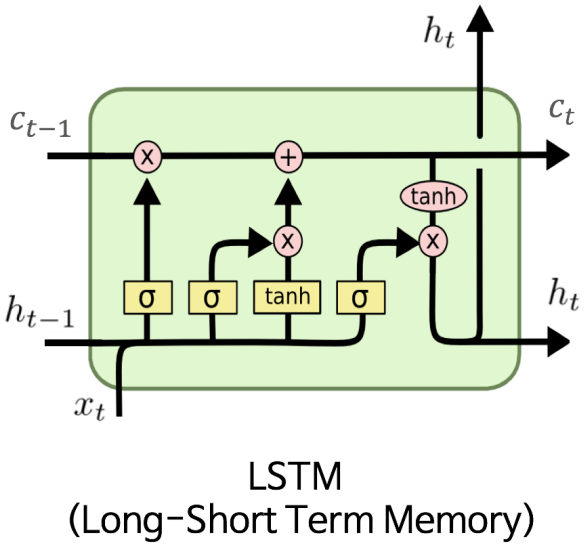

Figure 2. High level model for `LSTM` (Olah, 2015).

where t is the time step, $\mathbf{x}_t$ is the input vector, $\mathbf{c}_t$ is the cell state vector, $\mathbf{h}_t$ is the hidden state vector, $\mathbf{f}_t$ is the forget gate's activation vector, $\mathbf{i}_t$ is the input gate's activation vector, $\mathbf{o}_T$ is the output gate's activation vector, and $\mathbf{W}, \mathbf{U}, \mathbf{b}$ are the weight matrices and bias vector parameters to be learned during training.

3.2. Overall Framework for Image Captioning

Encoder-decoder framework: The encoder's task is to extract a set of feature vectors which we refer to as annotation vectors. We can take the last convolution layer's channels as the extracted image features. The image feature, $\mathbf{V} = \{\mathbf{v}_1, ..., \mathbf{v}_k\}$, $v_i \in \mathbb{R}^d$, are a d dimensional representation corresponding to a part of the image. These annotation vectors are then fed as input to the decoder. The LSTM decodes the vector into a sequence of words. The global image feature, $\mathbf{v}^g = \frac{1}{k}\sum_{i=1}^{k} \mathbf{v}_i$, is used to initialise $\mathbf{h}_0$ the initial hidden state of our decoder.

For each pair of images, $\mathbf{I}$ and a captions, $\mathbf{y} = \{y_1, \cdots, y_T\}$ consisting of T words, the encoder-decoder model is trained to minimise the following negative log-likelihood (cross entropy loss) to obtain the optimum parameters of the model:

$$\begin{aligned} \mathcal{L} &= -\log p(\mathbf{y}|\mathbf{I}) \\ &= \sum_{l=1}^{T} -\log p(y_t|y_1, \cdots, y_{t-1}, \mathbf{I}), \end{aligned} \tag{2}$$

where $p(y_t|y_1, \cdots, y_{t-1}, \mathbf{I})$ is the conditional probability of generating word y_t given the image I and the previous words $\{y_1, \cdots, y_{t-1}\}$. Xu et al. (2015) also recommend including $\sum_{p=1}^{k}(1 - \sum_{t=1}^{T} \alpha_{p,t})^2$ to the $\mathcal{L}$ as "doubly stochastic regularization", to encourage the weights at a single pixel p to sum to 1 across all time steps T.[1]

[1] We use this only for the soft attention models

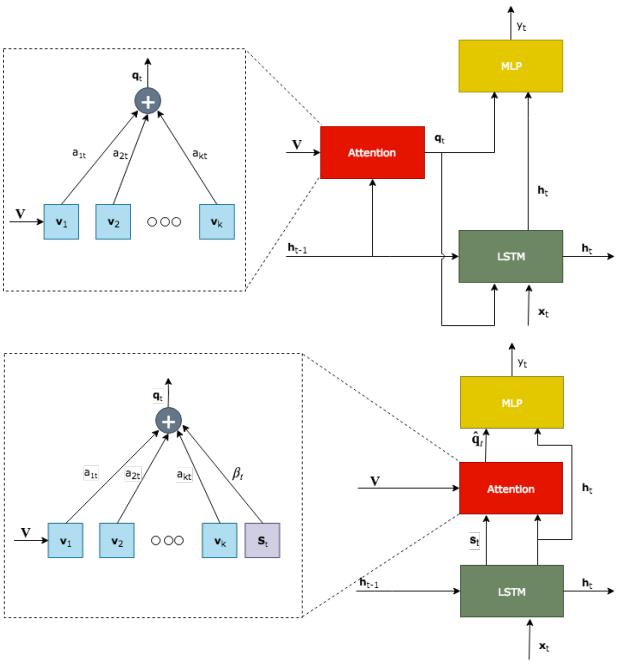

Figure 3. High level architecture for LSTM with soft and adaptive attention mechanism (Lu et al., 2016).

The LSTM unit computes the current hidden state, $\mathbf{h}_t$ based on the input vector, $\mathbf{x}_t$, the previous hidden state, $\mathbf{h}_{t-1}$ and the memory cell vector at time $t-1$, $\mathbf{m}_{t-1}$:

$$\mathbf{h}_t = \text{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{m}_{t-1})\,. \tag{3}$$

With LSTM, the t-th word, y_t is generated by the conditional distribution over possible output words at each time step, $\mathbf{p}_t$ is given by:

$$\begin{aligned} y_t \sim \mathbf{p}_t :&= p(y_t|y_1, \cdots, y_{t-1}, \mathbf{I}) \\ &= softmax(\mathbf{h}_t, \mathbf{q}_t) \end{aligned} \tag{4}$$

where $\mathbf{q}_t$ is the attention context vector. Note that we only have the context vector $\mathbf{q}_t$ in models with attention. The details of the context vector $\mathbf{q}_t$ are explained below.

3.3. Soft Attention

As mentioned in equation (4), $\mathbf{q}_t$ is the context vector, which emphasises the salient parts of the image to focus on when generating the word y_i, at time step t. This enables the decoder to focus on the relevant pixels when generating the next word.

For each time step, t, a normalized attention score α_{it} is computed for each encoded input, $\mathbf{h}_t$ to obtain a context vector $\mathbf{q}_t$:

$$\begin{aligned} e_{it} &= f_a(\mathbf{h}_{t-1}, \mathbf{v}_i) \\ \alpha_{it} &= softmax(e_{it}), \end{aligned} \tag{5}$$

where f_a is the attention function, $\mathbf{h}_{t-1}$ is the LSTM hidden state vector and $\mathbf{v}_i$ is the annotation vector. f_a is a multi-layer perceptron, with a ReLU as the activation function. The optimal weights of this neural network are learned during training.

Once the weights are obtained, the context vector, $\mathbf{q}_t$ is computed as:

$$\mathbf{q}_t = \sum_{i=1}^{k} \alpha_{it}\mathbf{v}_i\,. \tag{6}$$

3.4. Adaptive Attention

The adaptive attention encoder-decoder framework can automatically decide when to rely on spatial image information and when to just rely on the visual sentinel information. For example, we don't need to attend to the image when predicting function words such as "of" and "that". For computing context vector, $\mathbf{q}_t$: Instead of using the previous hidden state $\mathbf{h}_{t-1}$ in Equation 5 and 6, Lu et al. (2016) use the current hidden state $\mathbf{h}_t$ to generate $\mathbf{q}_t$.

We calculate context attention vector as follows:

$$\begin{aligned} e_{it} &= \mathbf{w}_h^T \tanh\left(\mathbf{W}_e\mathbf{V} + \mathbf{U}_e\mathbf{h}_t\mathbb{1}^T\right) \\ \alpha_{it} &= softmax(e_{it}). \\ \mathbf{q}_t &= \sum_{i=1}^{k} \alpha_{it}\mathbf{v}_i \end{aligned} \tag{7}$$

where $\mathbb{1} \in \mathbb{R}^k$ a vector with all elements set to 1. $\mathbf{W}_e, \mathbf{U}_e \in \mathbb{R}^{k\times d}$ and $\mathbf{w}_e \in \mathbb{R}^k$ are parameters to be learnt.

The visual sentinel vector, $\mathbf{s}_t$ is a latent representation of what the decoder already knows. The gate, $\mathbf{g}_t$ is element-wise multiplied with the memory cell to obtain the visual sentinel vector. The model can exploit this, choosing to fallback when not attending the image. The senital gate, β_t decides whether to attend to the image or to the visual sentinel.

$$\begin{aligned} \mathbf{g}_t &= \sigma(\mathbf{W}_g x_t + \mathbf{U}_g\mathbf{h}_{t-1} + \mathbf{b}_g) \\ \mathbf{s}_t &= \mathbf{g}_t \odot \tanh(\mathbf{m}_t), \end{aligned} \tag{8}$$

where $\mathbf{x}_t$ is the input vector, $\mathbf{m}_t$ is the decoder's memory cell, and $\mathbf{W}, \mathbf{U}, \mathbf{b}$ are the weight matrices and bias vector parameters to be learned during training.

The adaptive context vector, $\hat{\mathbf{q}}_t$ is a combination of the context vector, $\mathbf{q}_t$ and the visual sentinel vector, $\mathbf{s}_t$.

$$\hat{\mathbf{q}}_t = \beta_t\mathbf{s}_t + (1-\beta_t)\mathbf{q}_t\,, \tag{9}$$

where $\beta_t \in [0, 1]$ is the new sentinel gate at time t. β_t, for time step t is obtained as:

$$\hat{\alpha}_t = softmax([e_{it}; \mathbf{w}_h^T \tanh\left(\mathbf{W}_s\mathbf{s}_t + \mathbf{U}_e\mathbf{h}_t\right)]) \tag{10}$$

where $[.;.]$ indicates concatenation and $\hat{\alpha}_t \in \mathbb{R}^{k+1}$, is an extended version of the α seen in equation (7). The last element of this vector is interpreted as $\beta_t = \hat{\alpha}_t[k+1]$

4. Experiments

In this section we present the experiment that we ran, their results and answer the research questions we set out in section 1.

4.1. Setup

In all our models, we used the image features encoded using a pretrained ResNet-101, Xception or the SE-ResNet-101 network. The last two layers of CNNs - pooling and linear layers are removed since we are not doing image classification. Since we are using pretrained models, we preprocess our dataset by normalising the image using the mean, $[0.485, 0.456, 0.406]$ and standard deviation, $[0.229, 0.224, 0.225]$ of the RGB channels of the ImageNet's images. Pixel values must be in between 0 and 1. Note that we applied the global average pooling to all of our experiments in the encoder. To meaningfully compare the attention mechanisms and the encoder CNNs, we needed the same output dimensions from the encoder to prevent the extracted image features size from being the causing factor and influencing the performance of any experiments. The final spatial feature outputs produced by ResNet-101 and Xception have a dimension of 2048×7×7 (*channels* × *height* × *width*).

We fix the following hyperparameters for all our experiment:

- We used Adam learning algorithm, with a learning rate of 4e-4, which was found to work optimally in (Xu et al., 2015; Lu et al., 2016). We use the default values for the hyperparameters of Adam, $\rho_1 = 0.9$, $\rho_2 = 0.999$, $\eta = 0.9$, and $\epsilon = 1e-8$.
- We use a gradient clipping of 5 for the soft attention and 0.1 for the adaptive attention, same as in (Xu et al., 2015; Lu et al., 2016).
- Lu et al. (2016) used a batch size of 80 for the adaptive attention model, while Xu et al. (2015) used a batch size of 64 in their soft attention model. We decided to use a batch size of 80 for all of our experiments to fairly compare between soft and adaptive attention.

We could not tune the above parameters as (1) we didn't have the computational resources to tune over a large set of hyperparameters, (2) we found that the above hyperparameters to be similar to that was used in most of the literature.

We tune the following hyperparameters to find the best settings for our models. We choose these as they seemed to vary a lot across in the literature. We found that in our initial experiments (where we trained for a few epochs) that these had the most impact on the model performance:

- Decoder dimension = {300, 400, 500, 600, 700}
- Attention dimension = {300, 400, 500, 600, 700}
- Drop out rate = {0.25, 0.5}

Note that we didn't tune the attention dimension for the adaptive attention model, due to the architecture constrain, where the attention dimension has to be the size of the output of the encoder ($7 \times 7 = 49$).

Additionally we the GloVE-300 feature representation as our word embedding. We used Xavier initialisation for the weights and biases. Cross Entropy loss is used as our loss function. To avoid model over-fitting, early stopping is applied when the BLEU-4 score has stopped increasing after 8 epochs. We trained up to 30 epochs in total. The model will stop generating word when a special <END> token is predicted or the maximum sentence length (50) is reached. We also filtered out words that appeared less than 5 times when generating the word map. We use teacher forcing during training to speed up the convergence of our models.

4.2. Model Comparison

In total we train 6 different models. Our models consist of the two different attention mechanism (soft and adaptive) and the three different CNN architectures: ResNet-101, Xception and SE-ResNet-101.

1. **Baseline:** ResNet-101 + LSTM with Soft attention
2. Xception + LSTM with Soft attention
3. SE-ResNet-101 + LSTM with Soft attention
4. ResNet-101 + LSTM with Adaptive attention
5. Xception + LSTM with Adaptive attention
6. SE-ResNet-101 + LSTM with Adaptive attention

Due to the fact that training the model can take up to 2 days. We also experimented whether we can reduce the image resolution from 224×224 to 64×64. The reason for training the models on the reduced image size is two-folded, (1) speed up training due to computational limitation, and (2) to investigate whether we can achieve similar performance using smaller image resolutions. It must be noted the optimal hyperparameters that we get for these set of experiments may not necessarily correspond to the hyperparameters, had we tuned on the 224×224 images.

Attention Mechanism	**Network**	MS COCO	
		64	224
		BLEU-4	
Soft-Attention	ResNet-101	17.6	22.6
	Xception	15.8	19.6
	Se-ResNet-101	16.6	22.4
Show Attend and Tell (Xu et al., 2015)		**24.3**	
Adaptive-Attention	ResNet-101	19.2	30.0
	Xception	11.3	23.9
	Se-ResNet-101	18.0	29.5
Know When to Look (Lu et al., 2016)		**33.2**	

Table 1. BLEU-4 metric compared to other methods on validation images MS-COCO with crop sizes 64×64 and 224×224.

Network	Total Computation Time (hours)
ResNet-101	20.66
Xception	47.95
Se-ResNet-101	19.48

Table 2. Comparison on runtime using different network architectures.

4.3. Result

We present the best validation BLEU-4 scores we achieved for each of the six models in table 1. As mentioned earlier, we tuned the hyperparameters on the 64×64 images. We then retrained on the 224×224 images, with the optimal hyperparameters we found from the smaller images.

For all our models we found that a dropout rate of 0.25 worked best. We found that a dimension of 700 for both attention and the decoder hidden state produced the best results. We, however, picked attention and hidden state dimension of 500, when retaining on the larger dataset. This was due to the fact that we got very small improvements (less than 1% increase) with more than 500 dimensions. We further justify our choice as both Xu et al. (2015) and Lu et al. (2016) trained their models using a hidden state of dimension 512. This could be because they also noticed diminishing returns when training on larger dimensions for their hidden state.

In table 2 we present the total time taken for each model to converge to their best BLEU score on the validation set. Both ResNet-101 and SE-ResNet-101 converged in about 20 hours, while the Xception model took nearly 50 hours.

In Fig. 5 we present the results of training the soft attention models. We show the changes in our BLEU-4 score across each epoch. Similarly, in Fig. 4, we show the results for the adaptive attention models.

Our best model was ResNet-101 with adaptive attention with hidden state dimension 500, a dropout rate of 0.25. This achieved a BLEU-4 score of 30.0.

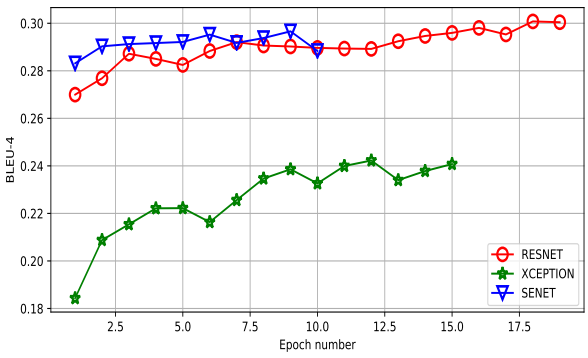

Figure 4. BLEU-4 score against epoch number for Adaptive Attention. Training will stop if the score has not improved over last 8 epochs.

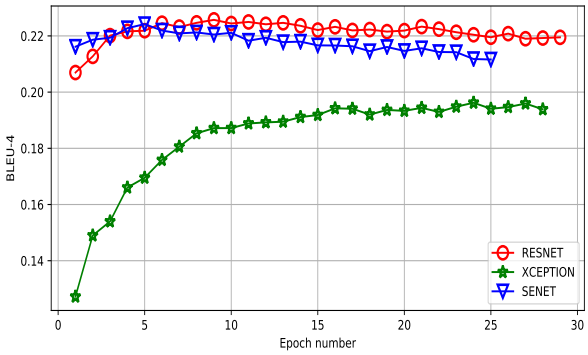

Figure 5. BLEU-4 score against epoch number for Soft Attention. Training will stop if the score if not improved over last 8 epochs.

4.4. Quantitative analysis

Adaptive attention models all had a better BLEU score compared to soft attention. This agrees with the hypothesis that our model would do better when it could fall back to just the visual sentinal. Since we used the current hidden state at time step t, instead of the previous hidden state, $\mathbf{h}_{t-1}$ as in soft attention, this could have had an influence on the result as well. We could have implemented the soft attention, in the same manner, to investigate whether that played any role in improved performance for future work.

For both soft and adaptive attention, we found that the ResNet without the squeeze and excitation block to be slightly better in terms of BLEU-4 score. However, the SE-ResNet-101 converged a lot faster in terms of epoch number. As seen in table 2, we see that SE-ResNet-101 converged faster in terms of hours taken. However, this was minimal as the average batch time taken to complete was about 70% higher for SE-ResNet-101. This partly agrees with the argument set forth by Sutskever et al. (2014), who say we get faster converge with minimal increase with computation. However, we saw that theoretical analysis can be misleading in deep learning when comparing the efficiency of algorithms. Layers which be only slightly slower in practice, may not be due to lack of optimisation, or the fact that more expensive operation may have been heavily optimised. This was evident for the squeeze and excitation operation. This should have added minimal computation time, but ended having a big impact on training time.

The Xception model clearly performed worse in all cases. We would expect that depth-wise separable convolution operation make Xception perform better than ResNet-101, as (Chollet, 2016) claimed Xception outperform ResNet-101 and 152. We investigated the cause of this and found 3 possible answers. One reason could be that the implementation that we found was off. We downloaded the Xception model from two different authors [2] [3] for PyTorch. Both of them produced very different results. It could be as simple as a bug causing the low BLEU-4 scores. Another

[2] https://github.com/tstandley/Xception-PyTorch
[3] https://github.com/Cadene/pretrained-models.pytorch

Test Image	Model	Caption
	ResNet-101 with Soft-Attention	A dog laying on the sidewalk next to a bike.
	ResNet-101 with Adaptive Attention	A dog sitting on the sidewalk next to a bike.
	SE-ResNet-101 with Soft Attention	A dog that is laying down on a sidewalk.
	SE-ResNet-101 with Adaptive Attention	A dog sitting on the ground next to a bicycle.

Table 3. Examples of visualisation results using different models.

reason is that we assumed the hyperparameters that worked for `ResNet` could be also applied to `Xception`. We could not find the optimal hyperparameters for the `Xception` model due to the time constraint. Lastly, the implementation recommended image resolution of 299×299 and some sources (Chollet, 2016) recommended values of 0.5 for both the mean and standard deviation in normalising for the red, green and blue channels of the image. This could be investigated in future work.

Lastly, we achieved slightly lower `BLEU-4` scores compared the literature as shown in table 1. This likely stems from the fact that the authors fine-tuned their encoder and learned the word embeddings from scratch, instead of using pre-trained embeddings. The image resolution of size 64×64 performed a lot worse compared to the 224×224. This is likely due to the fact that too much information was lost when our `CNNs` extracted features from the image. This results in the attention not being able to probably attend to the correct parts due to the image features being too coarse. This problem was also noticed by Lu et al. (2016).

4.5. Qualitative analysis

We provided some qualitative examples in tables 3-5 and Fig. 6 for a better understanding of each model.

We can see from table 3, that `ResNet-101` produces slightly more detailed captions compared to `SE-ResNet`. It gets all the important object and attributes in the image correct, "sidewalk", "sitting/laying" and "bike". While the `SE-ResNet` model only incorporates two of salient objects and attributes from the image. Also we can argue the soft attention with `ResNet-101` produced a better caption compared to adaptive attention `ResNet-101`. It gets the subtle distinction correct that the dog is "laying" instead of "sitting".

Test Image	Beam Size	Caption
	1	A baseball player is standing on the field.
	5	A baseball player is pitching a ball on the field.

Table 4. Top: Using beam size 1. Bottom: Using beam size 5.

In table 4, we can see the importance of beam search in generating captions. We get more descriptive captions with a beam width of 5.

Interestingly, we found that our models had problems distinguishing between genders as shown in table 5. We can also see some failure cases, such as with the skateboard caption. The posture of the person in the picture confused both our models to believe that he was on a skateboard.

We have provided more examples of visualisation of each model we used in the Appendix.

5. Related work

In this section, we discuss the work that has gone into improving image captioning and the techniques used prior to attention mechanism.

The main idea of a template based model is to project images and captions to a vector space. We first generate caption templates, which we fill in based on the results of the object and attributes that are detected in the image. For instance, Farhadi et al. (2010) had proposed using the following template $< object, action, scene >$ to generate caption based on an image. The advantage of this approach was that this generated natural sentences and syntax, however, it missed some of the more salient objects in the images (Ordonez et al., 2011). The second approach to generating from a template was first to retrieve similarly captioned images from some large database then modify these captions to fit the image which is being captioned (Kuznetsova et al., 2012). Both of these approaches involve a generalisation step before the concrete text is generated for an image. These methods have fallen out of favour to the neural network methods.

Image captioning has been inspired by the success in machine translation, reducing the problem of image captioning as to “translating" an image to text. Thus all state of the art method from 2014 onward use a sequence to sequence encoder-decoder architecture to caption images. Kiros et al. (2014) proposed a feed-forward neural network,`FFNN` with a multi-modal log-bilinear model to predict the next word given the image and previous word. Later the `FFNN` was replaced by a `RNN`, and eventually Sutskever et al. (2014) used a `LSTM`, instead of a vanilla `RNN` as the decoder. Karpathy & Fei-Fei (2015) have used a `R-CNN` to detect objects

Test Image	Model	Caption
	ResNet-101 with Soft-Attention	A woman sitting at a table with a plate of food.
	ResNet-101 with Adaptive Attention	A man sitting at a table with a plate of food.
	ResNet-101 with Soft-Attention	A man riding a skateboard down a sidewalk.
	ResNet-101 with Adaptive Attention	A man on a skate board does a trick.

Table 5. Examples of visualisation results failure cases.

in an image and use the output from a bidirectional RNN to jointly learn the embedding space for caption generation.

More recently, attention has made great strides in improving the state of the art. Xu et al. (2015) proposed using attention to generate better captions for an image by only focusing on certain parts of the image, when generating words at a particular time step. This was again inspired by machine translation. In machine translation involving attention, only the relevant source words are fed to the decoder when translating the target word. This made huge improvements in translating long sentences as the decoder didn't have to capture the context of the entire source sentence (which may be longer than 40 words) when generating the translation.

A disadvantage of the soft and hard attention is that the model attends to the image for every time step. Lu et al. (2016) argue that is not necessary when generating functional words such as "of" and "that". Also, consider the case when we have a stop sign in the image and we have already predicted "a red stop", we do not need to attend to the image to predict the next word as "sign". Hence Lu et al. (2016) proposed an attention model which decides whether to trust the visual signals or just the language model.

Some researchers have adopted the use of Generative Adversarial Net, GAN (Goodfellow et al., 2014) as a decoder to generate captions. A GAN consist of two components: a generator and a discriminator network. The idea behind GANs is that the generator produces a caption given an image along with the reference caption. The discriminator attempts to assign low scores to the fake captions while high scores to the real captions. The aim of the generator is to produce realistic captions to fool the discriminator. During training, we optimize the generator and discriminator at different intervals. The end result of training is that we end in a Nash equilibrium with both the discriminator and generator at their optimal states. Shetty et al. (2017) applied GANs to image captioning task and achieved similar performance to the state of the art (Shetty et al., 2017).

There are however problems with GANs, most commonly counting and global structures as pointed out by Salimans et al. (2016). During the generation, it can get the counts of objects wrong, which can often be caused by the lack of variety of similar objects in the training data. Another problem is that it learns only limited caption variety (one realistic caption for an image is enough to fool the discriminator). Again due to time constraints we were not able to explore GANs for this project, but would be an interesting task for future work.

6. Conclusion

In conclusion, our aim was to explore the adaptive attention mechanism and faster methods to train image captioning models. We did find that the Se-ResNet-101 converged faster than ResNet-101, the difference when looking at training time in hours was modest. This could partly be attributed to the implementation of Se-ResNet-101. We did find the adaptive attention was much better compared to soft attention when evaluating using the automatic metric BLEU-4. However, from our qualitative analysis, we found that in quite a few cases the soft attention model produced better captions compared to the adaptive attention model.

While we already looked at quite a few methods to decrease training time, we could have implemented more techniques to decrease training time. One method would have been to use GRUs proposed by Cho et al. (2014), which have fewer gates and parameters compared to LSTMs and perform just as well as LSTMs on some tasks (Chung et al., 2014). Ba et al. (2016) have proposed a layer normalization for LSTMs, similar to batch normalization for CNNs, which they claim reduces training time substantially. Both of these techniques could have been investigated for decreases in training time.

Model	BLEU-4
ResNet-101 with Adaptive-Attention	28.4
ResNet-101 with Soft-Attention	21.9

Table 6. BLEU-4 metric on testing images MS-COCO.

References

Ba, Jimmy Lei, Kiros, Jamie Ryan, and Hinton, Geoffrey E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, and Vincent, Pascal. Audio chord recognition with recurrent neural networks. In *ISMIR*, 2013.

Cho, KyungHyun, van Merrienboer, Bart, Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014. URL http://arxiv.org/abs/1409.1259.

Chollet, François. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL http://arxiv.org/abs/1610.02357.

Chung, Junyoung, Gülçehre, Çaglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

Farhadi, Ali, Hejrati, Mohsen, Sadeghi, Mohammad Amin, Young, Peter, Rashtchian, Cyrus, Hockenmaier, Julia, and Forsyth, David. Every picture tells a story: Generating sentences from images. In *European conference on computer vision*, pp. 15–29. Springer, 2010.

Feng, Yansong and Lapata, Mirella. Automatic caption generation for news images. 2013.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Graves, Alex. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012. URL http://arxiv.org/abs/1211.3711.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Howard, Andrew G., Zhu, Menglong, Chen, Bo, Kalenichenko, Dmitry, Wang, Weijun, Weyand, Tobias, Andreetto, Marco, and Adam, Hartwig. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL http://arxiv.org/abs/1704.04861.

Huh, Mi-Young, Agrawal, Pulkit, and Efros, Alexei A. What makes imagenet good for transfer learning? *CoRR*, abs/1608.08614, 2016. URL http://arxiv.org/abs/1608.08614.

Karpathy, Andrej and Fei-Fei, Li. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3128–3137, 2015.

Kiros, Ryan, Salakhutdinov, Ruslan, and Zemel, Rich. Multimodal neural language models. In *International Conference on Machine Learning*, pp. 595–603, 2014.

Kuznetsova, Polina, Ordonez, Vicente, Berg, Alexander C, Berg, Tamara L, and Choi, Yejin. Collective generation of natural image descriptions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pp. 359–368. Association for Computational Linguistics, 2012.

Lin, Tsung-Yi, Maire, Michael, Belongie, Serge J., Bourdev, Lubomir D., Girshick, Ross B., Hays, James, Perona, Pietro, Ramanan, Deva, Dollár, Piotr, and Zitnick, C. Lawrence. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

Lu, Jiasen, Xiong, Caiming, Parikh, Devi, and Socher, Richard. Knowing when to look: Adaptive attention via A visual sentinel for image captioning. *CoRR*, abs/1612.01887, 2016. URL http://arxiv.org/abs/1612.01887.

Olah, Christopher. Understanding lstm networks, Aug 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Ordonez, Vicente, Kulkarni, Girish, and Berg, Tamara L. Im2text: Describing images using 1 million captioned photographs. In *Advances in neural information processing systems*, pp. 1143–1151, 2011.

Papineni, Kishore, Roukos, Salim, Ward, Todd, and Zhu, Wei-Jing. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002. URL http://aclweb.org/anthology/P02-1040.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Shetty, Rakshith, Rohrbach, Marcus, Anne Hendricks, Lisa, Fritz, Mario, and Schiele, Bernt. Speaking the same language: Matching machine to human captions by adversarial training. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4135–4144, 2017.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron C., Salakhutdinov, Ruslan, Zemel, Richard S., and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015. URL http://arxiv.org/abs/1502.03044.

Yun Jey, Choy. Pytorch tutorial for deep learning researchers. https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning, 2018.

Appendix

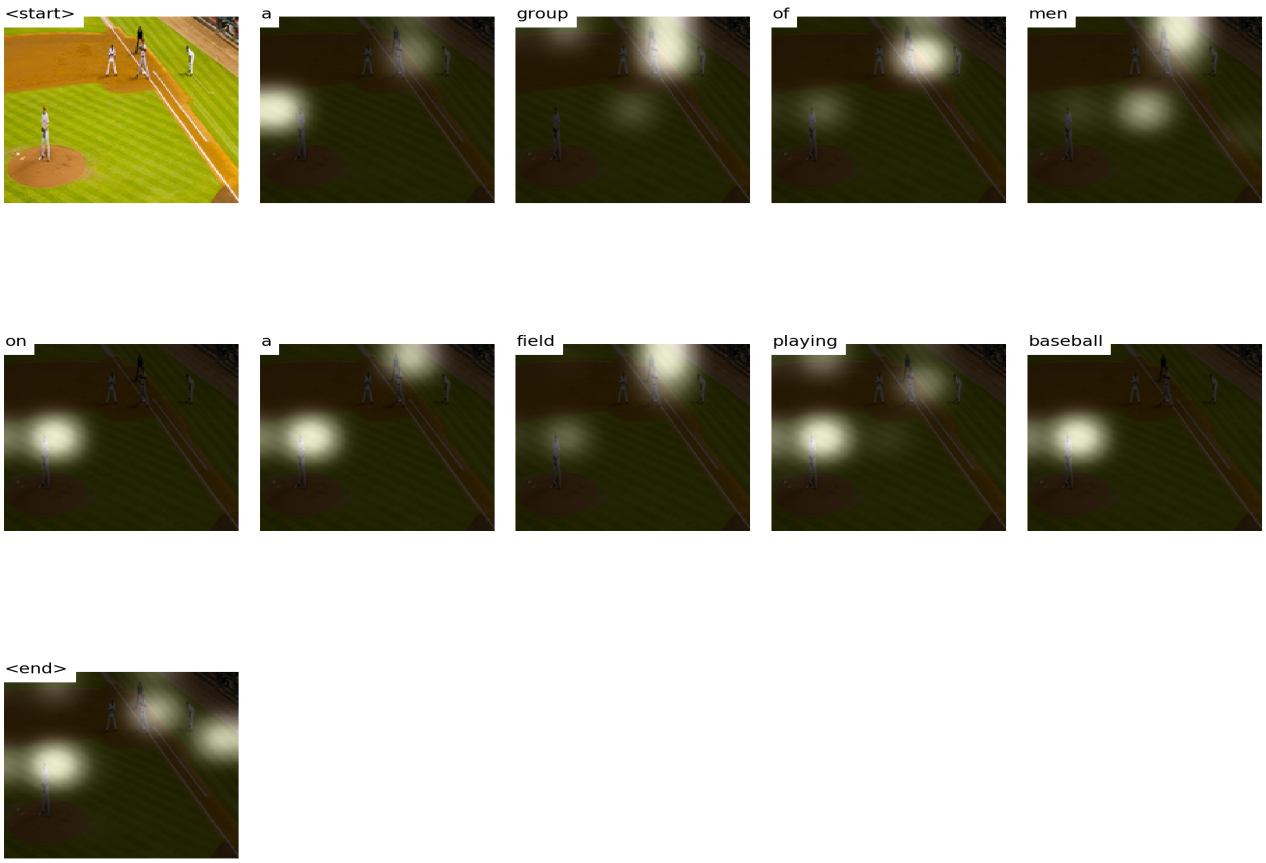

Figure 6. **ResNet-101 with Adaptive-Attention**: A group of men on a field playing baseball.

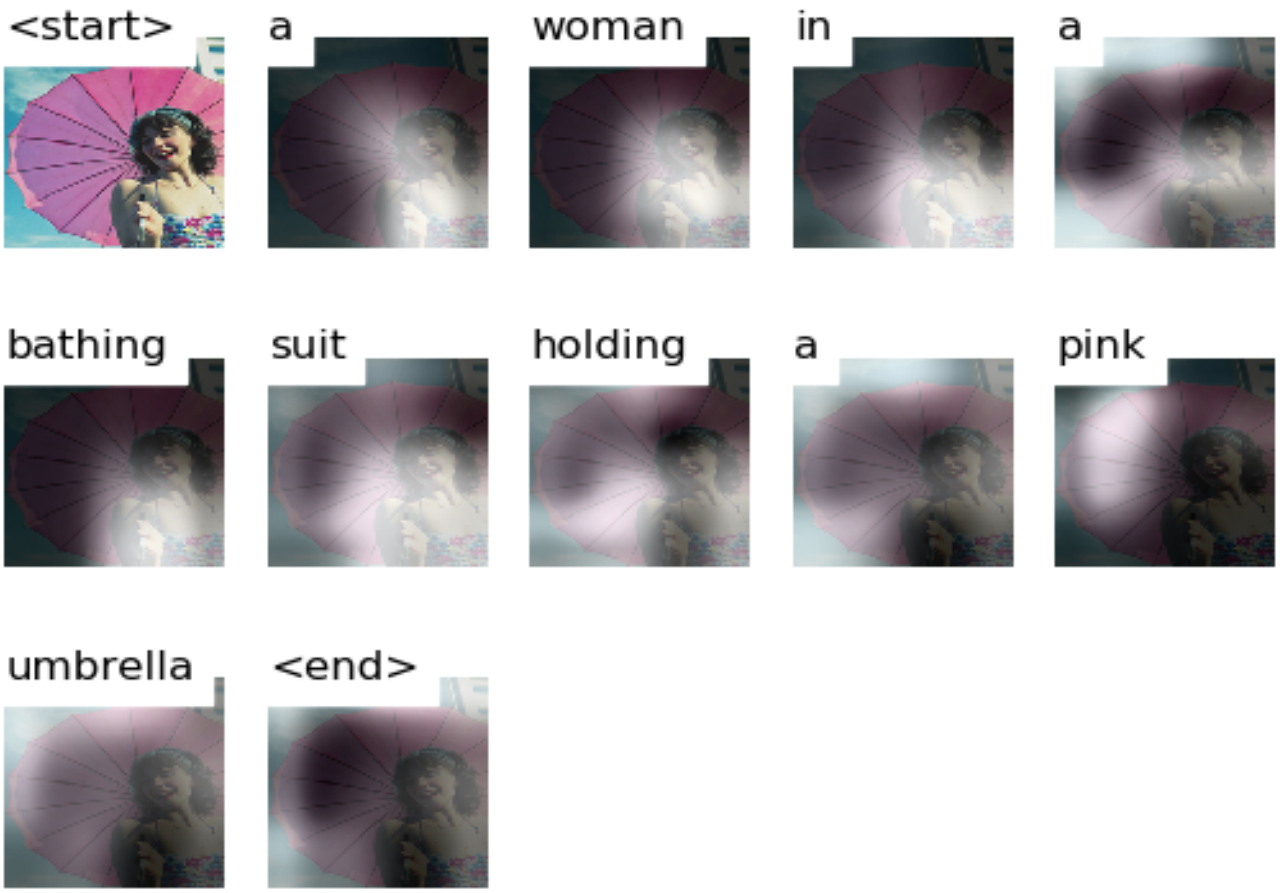

Figure 7. **`ResNet-101` + `LSTM` + with Soft attention**: A woman in a bathing suit holding a pink umbrella.

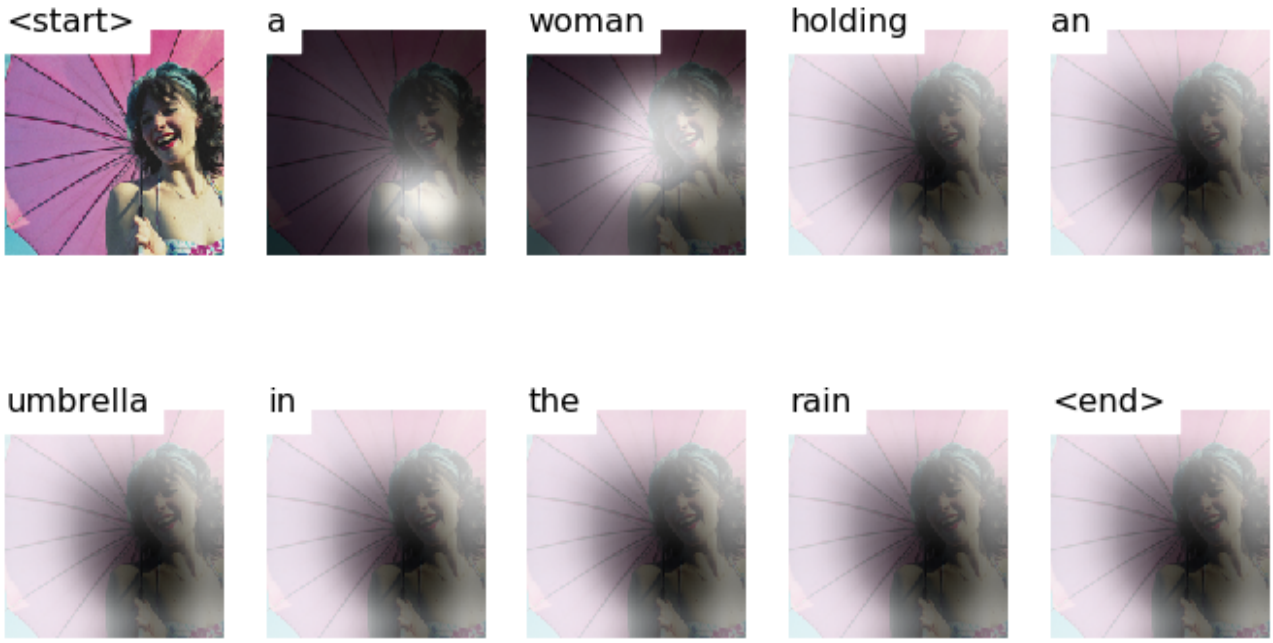

Figure 8. **`Xception` + `LSTM` with Soft attention**: A woman holding an umbrella in the rain.

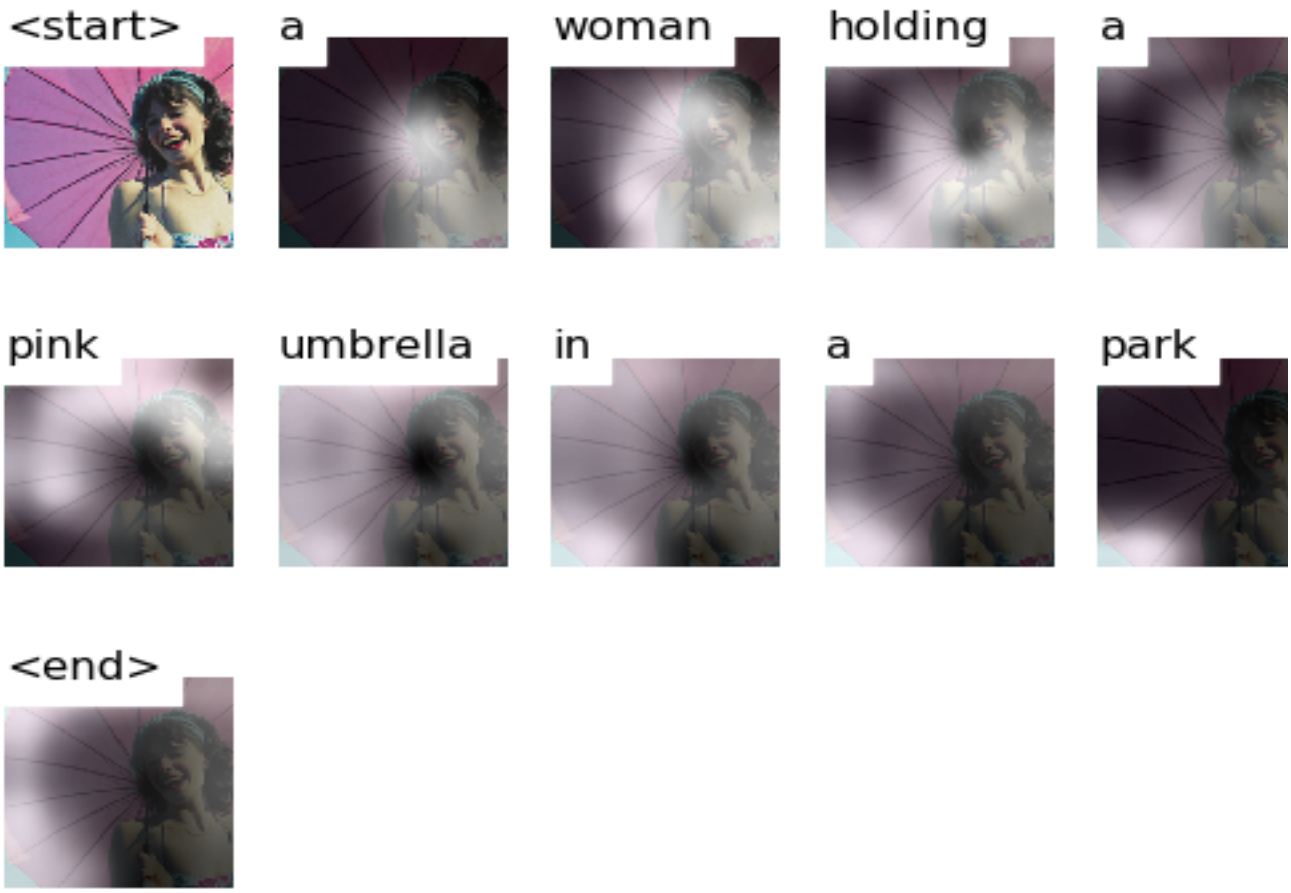

Figure 9. **SE-ResNet-101 + LSTM with Soft attention**: A woman holding a pink umbrella in a park.

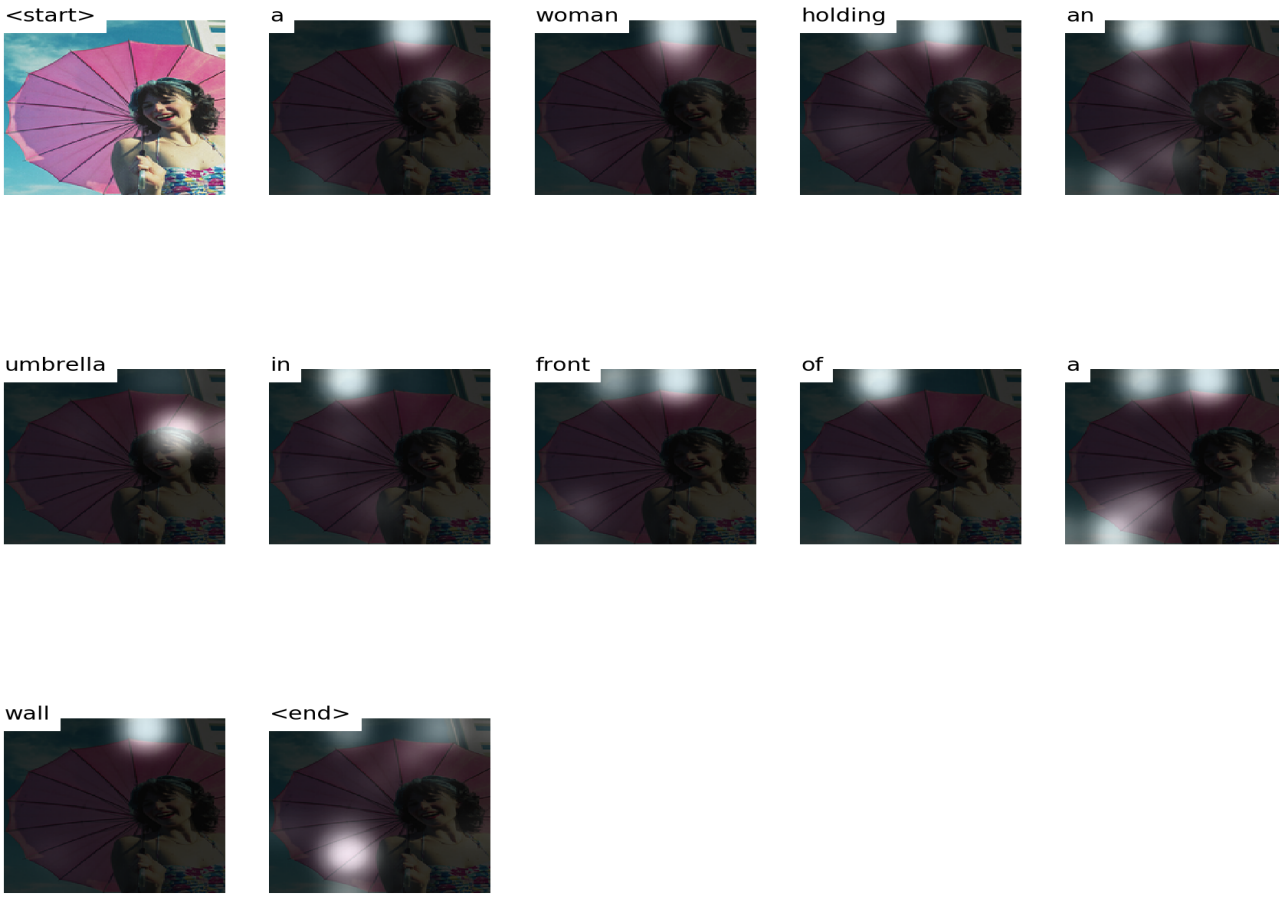

Figure 10. **ResNet-101 + LSTM with Adaptive attention**: A woman holding an umbrella in front of a wall.

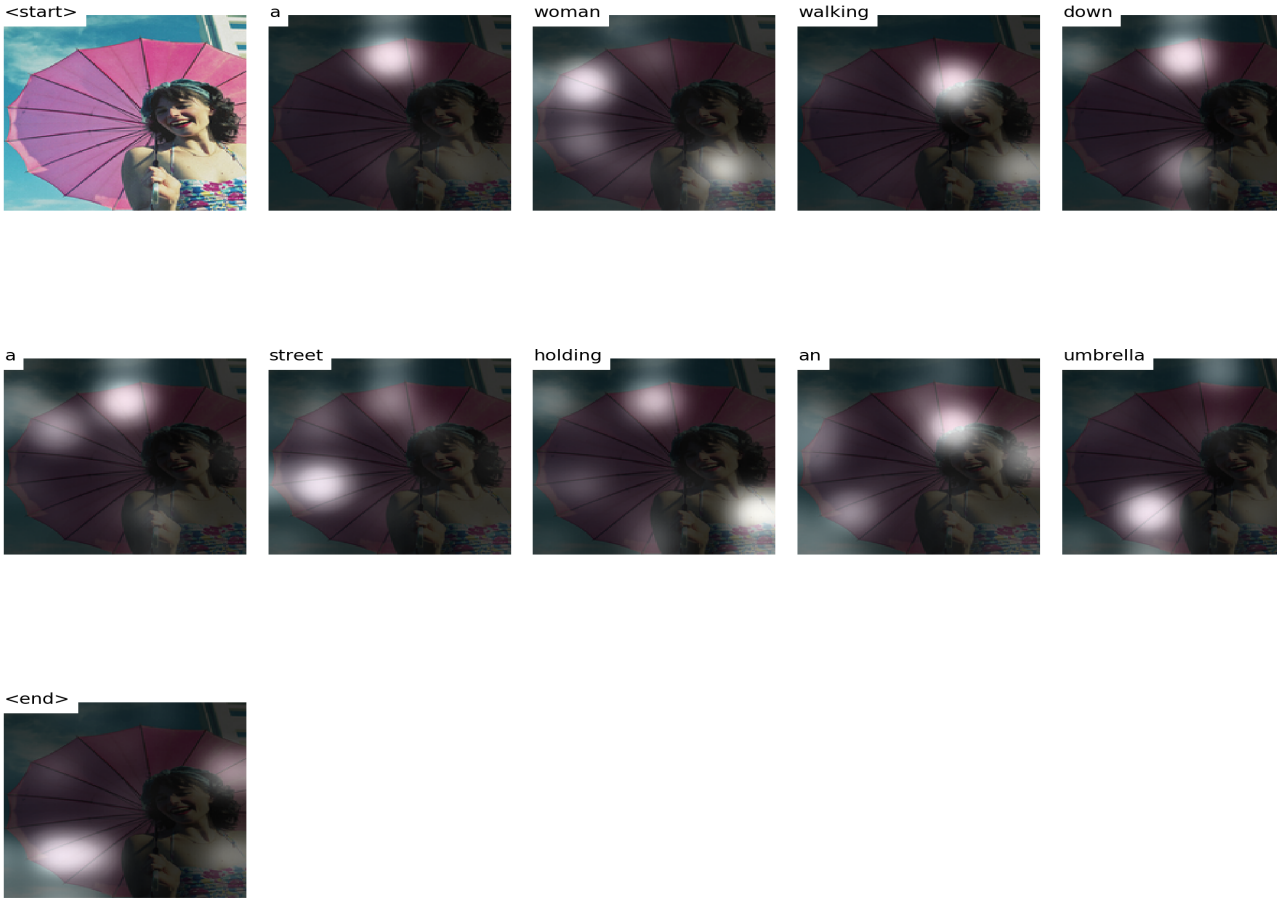

Figure 11. **`Xception` + `LSTM` with Adaptive attention**: A woman walking down a street holding an umbrella.

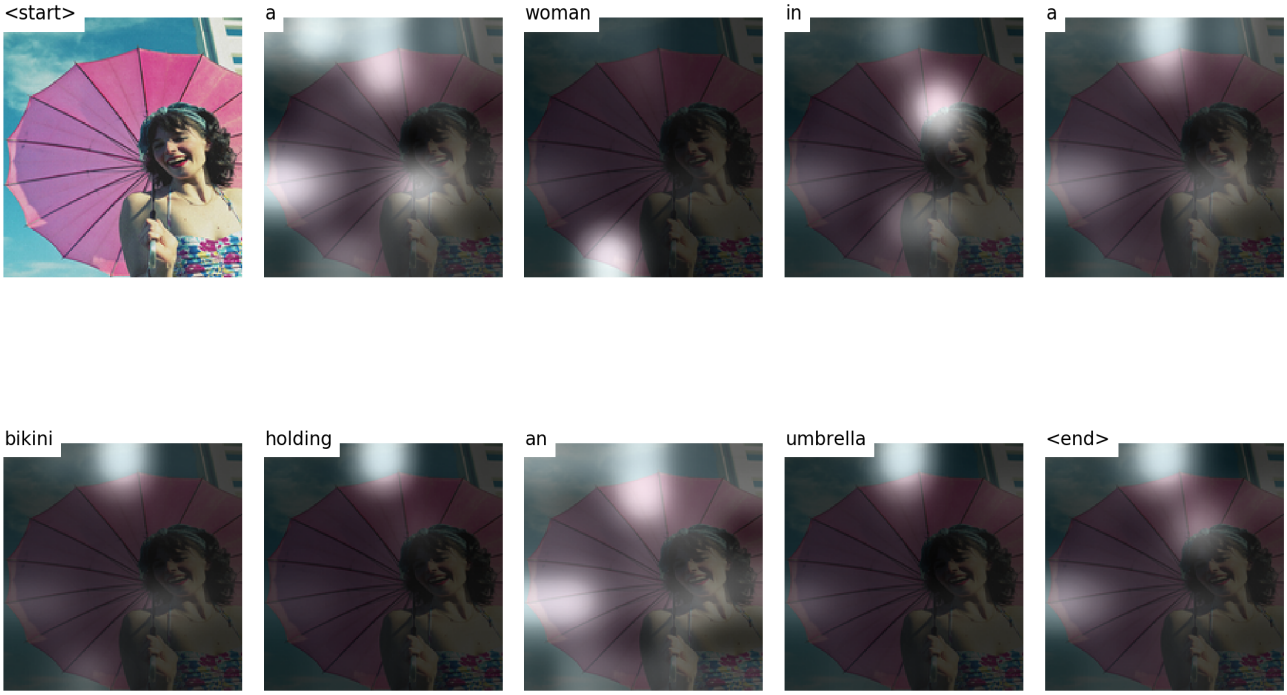

Figure 12. **`SE-ResNet-101 + LSTM` with Adaptive attention**: A woman in a bikini holding an umbrella.