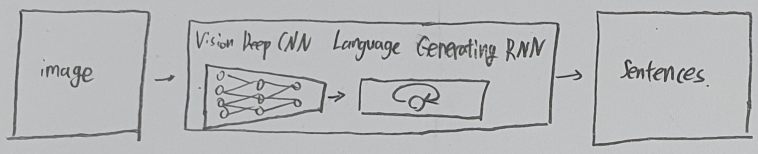
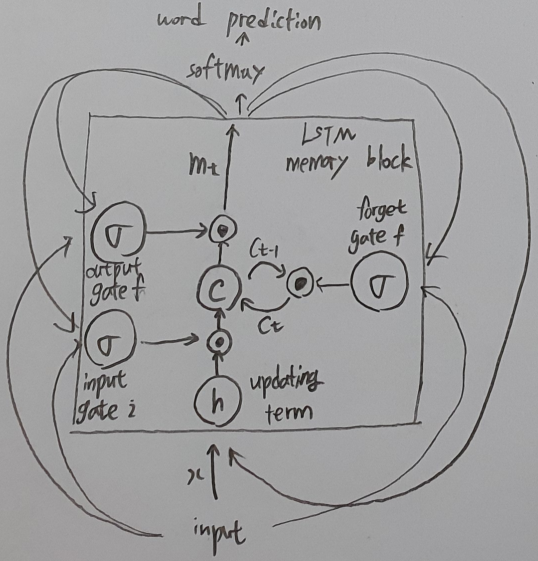
AIGS 538 Homework Assignment 4 20222421 심재윤 (JAEYOON SIM)

#1. Show and Tell: A Neural Image Caption Generator

Automatically describing the content of an image is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. They present a generative model based on a deep current archite cture that combines recent advances in computer vision and machine translation and that can be used to generate natural sentences describing an image. Their model is trained to maximize the likelihood of the target description sentence given the training image. Experiments on several datasets show the accuracy of the model and the fluency of the language it learns solely from image descriptions. They propose to follow elegant recipe, replacing the encoder RNN by a deep convolution neural network (CNN). It is natural to use a CNN as an image "encoder", by first pre-training it for an image classification task and using the last hidden layer as an input to the RNN decoder that generates sentences. They call this model the Neural Image Caption.

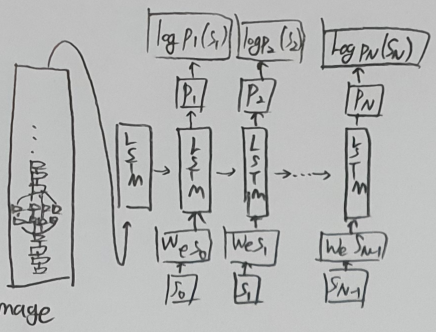


They propose a neural and probabilistic framework to generate descriptions from images. These models make use of the recurrent neural network which encodes the variable length input into a fixed dimensional vector, and uses this representation to "decode" it to the desired output sentence. They propose to directly maximize the probability of the correct description given the image by using the formulation $\theta^* = \arg\max_\theta \sum_{(I,S)} \log p(S|I;\theta)$ where θ are the parameters of their model, I is an image, and S its correct transcription. Since S represents any sentence, its length is unbounded. To make the RNN $h_{t+1} = f(h_t, x_t)$ more concrete two crucial design choices are to be made: what is exact form of f and how are the images and words fed as input x_t. For f they use a Long-Short Term Memory (LSTM) net and this model is outlined in the next section. For the representation of images, they use a Convolutional Neural Network (CNN).



The core of the LSTM model is memory cell c encoding knowledge at every time step of what inputs have been observed up to this step. The behavior of the cell is controlled by "gates" - layers which are applied multiplicatively and thus can either keep a value from the gated layer if the gate is 1 or zero this value if the gate is 0. In particular, three gates are being used which control whether to forget the current cell value, if it should read its input and whether to output the new cell value. Such multiplicative gates make it possible to train the LSTM robustly as these gates deal well with exploding and vanishing gradients.

2022242 심재윤 (JAEYOON SIM)

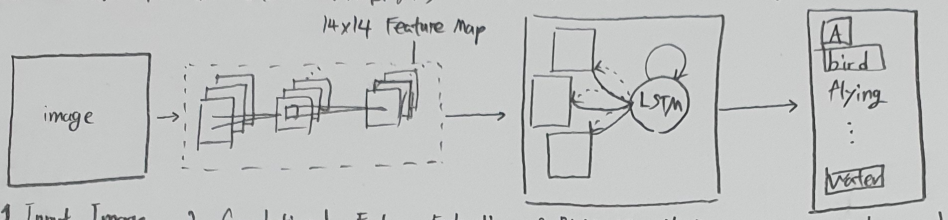


The LSTM model is trained to predict each word of the sentence after it hase seen the image as well as preceding words as defined by $p(S_t | I, S_0, \ldots, S_{t-1})$. For this purpose, it is instructive to think of the LSTM in unrolled form- a copy of the LSTM memory is created for the image and each senten word such that all LSTM share the same parameters and the output m_{t-1} of the LSTM at time t-1 is fed to the LSTM at time t. All recurrent connections are transformed to feed-forward connections in the unrolled version. In more details, if they denote by I the input image and by $S = (S_0, \ldots, S_N)$ a true sentence describing this image, the unrolling procedure reads $x_{-1} = CNN(I)$, $x_t = W_e S_t$, $P_{t+1} = LSTM(x_t)$ where $t = 0 \ldots N-1$ and the represent each word as a one-hot vector S_t of dimension equal to the size of the dictionary. In particular by emitting the stop word the LSTM signals that a complete sentence has been generated. Both the image and the words are mapped to the same space, the image by using a vision CNN, the words by using word embedding W_e. The image I is only input once, at $t=-1$, to inform the LSTM about the image contents. They empirically verified that feeding the image at each time step as an extra input yields inferior results, as the network can explicitly exploit noise in the image and overfits more easily. Their loss function is the sum of negative log likelihood of the correct word at each step as $L(I,S) = -\sum_{t=1}^{N} \log p_t(S_t)$. This loss function is minimized w.r.t. all the parameters of the LSTM, the top layer of the image embedder CNN and word embedding W_e. They explored several techniques to deal with overfitting. They initialize the weights of the CNN component of their system to a pretrained model. Another set of weights that could be sensibly initialized are W_e, the word embeddings. And they did some model level overfitting-avoiding techniques. They tried dropout and ensembling models, as well as exploring the size of the model by trading off number of hidden units versus depth. Dropout and ensembling gave a few BLEU points improvements. They trained all sets of weights using stochastic gradient descent with fixed learning rate and no momentum. All weights were randomly initialized except for the CNN weights, which we left unchanged because changing them had a negative impact. And they used 512 dimensions for the embeddings and the size of the LSTM memory. They used PASCAL, Flickr, SBU datasets to evaluate and Flickr datasets yield the bert performance. And human scores were computed by comparing one of the human captions against the other four. They do this for each of the five raters and average their BLEU scores. And they think it is more meaningful to report BLEU-4, which is the standard in machine translation moving forward. Their model had a 21.2 BLEU-4 and 59 for PASCAL, 66 for Flickr 30k, 63 for Flickr 8k, and 28 for SBU with BLEU-1 scores. Additionally, NIC is doing surprisingly well on both ranking tasks. For image annotation, NIC had 20 for R@1, 61 for R@10, 6 for Med r. For image search, NIC had 19 for R@1, 64 for R@10, and 5 for Med r. These are the recall of NIC with Flickr 8k. And they show good results for Flickr 30k.

20222421 심재윤 (JAEYOON SIM)

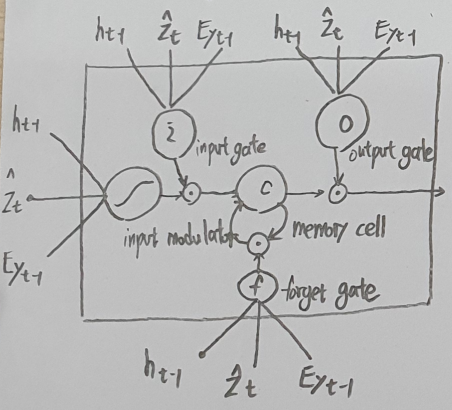
#2. Show, Attend and Tell : Neural Image Caption Generation with visual Attention.

Inspired by works in machine translation and object detection, they introduce an attention based model that automatically learns to describe the content of images. They describe how they can train this model in a deterministic manner using standard backpropagation techniques and stochastically by maximizing a variational lower bound. And they validate the use of attention with state-of-the-art performance on three benchmark datasets: Flickr8k, Flickr30k and MS COCO.



1. Input Image 2. Convolutional Feature Extraction 3. RNN with attention over the image 4. Word by word generation.

In this paper, they describe approaches to caption generation that attempt to incorporate a form of attention with two variants: a "hard" attention mechanism and a "soft" attention mechanism. They also show how one advantage of including attention is the ability to visualize what the model "sees."



They describe the two variants of their attention-based model by first describing their common framework. Their model takes a single raw image and generates a caption y encoded as a sequence of 1-of-K encoded words. $y = \{y_1, \dots y_C\}$ where $y_i \in R^K$. And they use a convolutional neural network in order to extract a set of feature vectors which they refer to as annotation vectors. The extractor produces L vectors, each of which is a D-dimensional representation corresponding to a part of the image. In order to obtain a correspondence between the feature vectors and portions of the 2-D image, they extract features from a lower convolutional layer. And they use a long short-term memory (LSTM) network that produces a caption by generating one word at every time step conditioned on a context vector, the previous hidden state and the previously generated words. Their implementation of LSTM closely follows the above figure. Once the weights are computed, the context vector $\hat{z}_t$ is computed by $\hat{z}_t = \phi(\{a_i\}, \{\alpha_i\})$ where ϕ is a function that returns a single vector given the set of annotation vectors and their corresponding weights. The initial memory state and hidden state of the LSTM are predicted by an average of the annotation vectors fed through two seperate MLPs (init c and init h). In this work, they use a deep output layer to compute the output word probability given the LSTM state, the context vector and the previous word. There are two alternative mechanisms for the attention model: stochastic attention and deterministic attention. Soft attention is a fully differentiable deterministic mechanism that can be plugged into an existing system, and the gradients are propagated through the attention mechanism at the same time they are propagated through the rest of the network. Hard attention is a stochastic process. Instead of using all the hidden states as an input for the decoding, the system samples a hidden state

with the probabilities. In order to propagate a gradient through this process, they estimate the gradient by Monte Carlo sampling. Stochastic hard attention algorithm for the parameters W of the models can be derived by directly optimizing $L_s = \sum_s p(s|a) \log p(y|s,a)$. Deterministic soft attention model is trained end-to-end by minimizing the penalized negative log-likelihood $L_d = -\log(p(y|x)) + \lambda \sum_i^L (1 - \sum_t^C \alpha_{ti})^2$.

Both variants of their attention model were trained with stochastic gradient descent using adaptive learning rate algorithms. For the Flickr8k dataset, they found that RMSprop worked best, while for Flickr30k/MS COCO dataset they used the recently proposed Adam algorithm. To create the annotations used by their decoder, they used Oxford VGGnet pre-trained on ImageNet without fine-tuning. With enough data, they could also train the encoder from scratch with the rest of the model. They use the 14x14x512 feature map of the fourth convolutional layer before max pooling. And they found training on a random group of captions to be computationally wasteful. To mitigate this problem, in processing they build a dictionary mapping the length of a sentence to the corresponding subset of captions. Then, during training they randomly sample a length and retrieve a mini-batch of size 64 of the length. On largest dataset MS COCO, their soft attention model took less than 3 days to train on an NVIDIA Titan Black GPU. In addition to dropout, the only other regularization strategy they used was early stopping on BLEU score. They observed a breakdown in correlation between the validation set log-likelihood and BLEU in the later stages of training during their experiments. Since BLEU is the most commly reported metric, they used BLEU on their validation set for model selection. In their experiments with soft attention, they also used Whetlab in our Flickr8k experiments. Some of the intuitions they gained from hyperparameters regions it explored were especially important in their Flickr30k and COCO experiments. They report results on the popular Flickr8k and Flickr30k dataset which has 8,000 and 30,000 images respectively as well as the more challenging Microsoft COCO dataset which has 82,783 images. The Flickr8k/Flickr30k dataset both come with 5 reference sentences per image, but for the MS COCO dataset, some of the images have references in excess of 5 which for consistency across our datasets they discard. They applied only basic tokenization to MS COCO so that it is concistent with the tokenization present in Flickr8k and Flickr30k. For all their experiments, they used a fixed vocabulary size of 10,000. Results for their attention-based architecture are reported as follow. The report results with the frequently used BLEU metric which is the standard in the caption generation literature. They report BLEU from 1 to 4 with their a brevity penalty.

Dataset	Model	BLEU				
		BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
Flickr8k	Google NIC	63	41	27	-	-
	Soft-Attention	67	44.8	29.9	19.5	18.93
	Hard-Attention	67	45.7	31.4	21.3	20.30
Flickr30k	Google NIC	66.3	42.3	27.7	18.3	-
	Soft-Attention	66.7	43.4	28.8	19.1	18.49
	Hard-Attention	66.9	43.9	29.6	19.9	18.46
COCO	Google NIC	66.6	46.1	32.9	24.6	-
	Soft-Attention	70.7	49.2	34.4	24.3	23.90
	Hard-Attention	71.8	50.4	35.7	25.0	23.04