Simultaneous Visual and Linguistic Embeddings with CNNs and T-LSTMs

Nick Dufour, Jayant Thatte, Prasanth Veerina
Stanford University
450 Serra Mall, Stanford, CA 94305
{ndufour,jayantt,pveerina}@stanford.edu

Abstract

We explore embedding visual depictions of scenes and their linguistic descriptions in a common vector space by joint learning. This is achieved via a layered neural network system, leveraging a convolutional neural network (CNN) and a tree "long short term memory" network (T-LSTM) to transform the respective scene representations into a vector space such that distance between vector representation of associated image/description pairs is minimized and that between contrastive pairs is maximized. We find that T-LSTMs and our model architecture performs strongly, although it performs significantly inferior as compared to the current state-of-the-art. This is potentially due to an abbreviated training schedule.

1 Introduction

Given an image, humans can rapidly and flexibly generate a natural language description of what the image depicts. Perhaps more impressive, humans can readily envision and draw a scene given a natural language description. This fact implies the existence of an internal representation of a scene that is independent of the descriptive media. Here, we demonstrate that such nonlinear transforms over images and over natural language descriptions can be learned by machine learning methods, and explore the learned common space.

Much work has been done in this area, in both 'directions': mapping images to natural language descriptions either by selection from a pool ('extractive') or *de novo* generation ('abstractive') [13, 7, 3, 5] as well as using natural language descriptions to search for images or creating a system to permit both query directions [2, 10].

Our work is most similar to work by Socher *et al* [10] although we extend their work in several critical ways. First, a vastly expanded dataset is used. Second, following recent advances in natural language processing (NLP), we leverage the remarkable power of tree-structured LSTMs (T-LSTMs) [11]. Finally we utilize a two column neural network with a joint loss to transform the image and sentence vectors into a shared vector space (as opposed to using linear transformations from one space to another).

2 Related Work

Our work is inspired by a number of recent papers on the topic of unifying visual and linguistic data. Here we briefly describe a few of the most relevant works. Of course everything described here is built on a great deal of foundational work in vector space models and neural network models which we have omitted here due to space-constraints.

Grounded Compositional Semantics for Finding and Describing Images with Sentences [10]

In this 2013 paper, Socher *et al* aim to compute compositional sentence vectors (vectors derived from recursive composition of word and phrase vectors) and image vectors and then map them into a common vector space such that semantic relationships are maintained.

The authors develop a model called dependency tree recursive neural networks (DT-RNN) which computes sentence vectors in a way that they hypothesize captures more of the visual representation meaning of an image by weighting word and phrase compositions that are semantically important according to the sentences dependency tree. Image vectors are computed using a convolutional neural network (CNN) trained on ImageNet.

Once these image and sentence vectors are obtained the authors then train a linear transformation into a common vector space using a max-margin loss. Matched image and sentence vectors are encouraged to have a high inner product and mismatched pairs to have a low product.

Overall this model had limited training since at the time of publication only 1000 labeled image/sentence pairs were available, the authors note that a large dataset would provide a promising direction for this line of inquiry.

Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models [5]

In this 2014 work Kiros et al. explore image-text embedding using complex deep learning models and take on the task of image captioning. The novel contribution here is that they cast the captioning problem as translation, applying the encoder-decoder ideas of neural machine translation to it. The encoder learns a joint image-sentence embedding by first encoding sentences with a long short-term memory (LSTM) and then projecting image features from a deep convolutional network into the LSTM hidden state vector space. The networks are trained together using a pairwise ranking loss which encourages image vectors to be close to their most descriptive caption vectors. The authors propose that this metric, pairwise ranking, can be used as a proxy for training the encoder for the task of novel generated captions. The authors used the same dataset that we use in this paper (MSCOCO + Flickr30k).

Deep Visual-Semantic Alignments for Generating Image Descriptions [3]

In this work Karpathy et al 2015 establish the state of the art approach to generating sentence descriptions for images by training a model to learn alignments between sentence snippets and image regions. Their system uses CNNs and a bi-deictional recurrent neural network (B-RNN) to embed images and sentences in the same space. The key contribution in this work is introducing a subobjective of aligning regions in an image with words from a training sentence while still optimizing an overall max-margin objective for image-caption pairs.

3 Methods

3.1 Data

We utilize a large number of images/description pairs acquired by combining the recently released Microsoft Common Objects in Context (MSCOCO)[6] and Flickr30k[14] datasets. Overall the combined dataset contains 155 thousand images and 775 thousand captions (each image is associated with 5 captions)–far exceeding the dataset sizes of similar efforts.

Natural language data (i.e., descriptions of images) underwent several preprocessing steps. First, irregularities in capitalization were standardized by a maximum-vote method (i.e., 'human' being more common in the corpus than 'HUMAN,' caused 'HUMAN' to be changed to 'human'). Next, the data were arranged into dependency trees via the Stanford Natural Language Parser (using Universal Dependencies) [1]. The words themselves were tokenized using a GLoVe [8] lookup table, in which each word is mapped to a 300-element vector trained on some 640 billion tokens. These dependency trees form the input to the DT-LSTM as seen in figure 1. Any word or token not found in the GLoVe lookup table was assigned to the unknown token, which was randomly initialized from a normal distribution whose parameters were dictated by existing GLoVe vectors.

Images are resized to 224x224 and undergo mean pixel subtraction as described in [9]. These transformed images form the input to our pretrained CNN as seen in figure 1.

3.2 Model Architecture

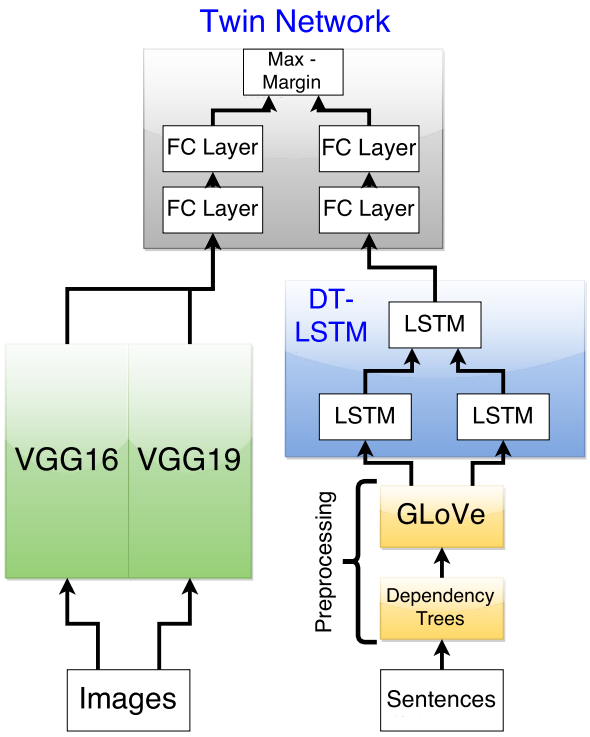

Figure 1: A schematic of the learning system.

Figure 1 shows an overview of the complete learning system. The system performs a forward pass on a batch of several pairs of images and captions at once.

Each sentence (arranged in a dependency tree) is transformed into an n-element vector by a T-LSTM and fed into the 'twin' network.

Simultaneously, the image corresponding to the description is transformed into a 4,096-element vector via either the 16- or the 19-layer pretrained neural network, termed 'VGG16' or 'VGG19,' depending. VGG-16 and VGG-19 are the record-breaking CNNs developed by the Visual Geometry Group [9]. These networks are already very well tuned for object detection on the ImageNet dataset; hence, they will remains static and the twin network will learn a transformation from VGG space into linguistic space. This was fed into the 'twin' network with no additional preprocessing.

The twin network consists of two layers in each column. The bottom layer in each column transforms the image or description vector into a shared dimension. Then the second layer applies an affine transform and so-called 'Leaky' rectified linear unit (ReLU) nonlinearity before feeding the two columns into the joint loss.

The role of the twin network was to learn a mapping from image vectors, x, and description vectors, y, into a shared vector space, which has the important property that the proximity of vectors is directly proportional to the similarity of the item they represent (either a description or an image). To ensure that this was the objective of the *entire* model, a max-margin cost function was used:

(a) A bunch of suitcases lined up along a wall.

(b) Two giraffes standing over a sun shielding shelter.

Figure 2: Example image-caption pairs from the dataset.

$$s_{1,i,j} = \mathbb{1}\{i \neq j\} \max(0, \Delta - x_i y_i + x_i y_j)$$
$$s_{2,i,j} = \mathbb{1}\{i \neq j\} \max(0, \Delta - x_i y_i + x_j y_i)$$

$$J(\theta)_i = \sum_j s_{1,i,j} + s_{2,i,j} \tag{1}$$

which attempts to maximally separate non-corresponding description/image pairs ('contrastive pairs') in vector space while bringing together corresponding description/image pairs ('true pairs').

The loss in equation (1) is computed over all contrastive pairs in the dataset however due to the size of the dataset and it was simply not feasible to compute this cost during each forward pass. In our implementation we draw on ideas from negative sampling and simply apply the loss function over each batch. For example given a batch of 10 image caption pairs (no duplicates) we have 10 true pairs and 90 contrastive pairs. This modified loss allows our model to actually train in practice.

3.2.1 DT-LSTM

Our DT-LSTM model is based on the Tai *et al*'s recent T-LSTM model [12]. We used it in place of a simple RNN due to the improved semantic representations it produces.

The DT-LSTM accepts, as input, captions in the form of parse dependency trees, where each node in the tree has a word vector associated with it. Semantic embedding of captions is given by the root node output of the DT-LSTM. Max-margin loss is computed inside the twin network using the embedded sentence and image vectors and the loss is then back propagated through the DT-LSTM. The DT-LSTM model uses unique sets of parameters for the first $p-1$ left children and the first $p-1$ right children (where p is a user-specified parameter) and the left nodes thereafter share a set of parameters and right nodes thereafter share another set of parameters. That is, in effect, the states and the outputs of all the left children index beyond and including p are summed up into a single effective state and output and the same is done for right nodes indexed beyond and including p. The DT-LSTM is described in greater detail in appendix A.

See appendix C for a link to the code for this model and the rest of the learning system.

4 Experiments

Five experiments were conducted, which are distinguished primarily by differences in the model hyperparameters. In all but one case, the models ran continuously from the point where the coding was complete to the submission of this report. See Tables 1 and 2 for details. Because the running time of the models proved to be quite long, none of the models were able to complete cover a single epoch. Differences in running time were partially a function of the number of parameters and partially a function of the speed of the computer used.

Learning was performed by stochastic gradient descent with 'momentum' updates, with a learning rate decay parameter of 0.95 every 2,500 iterations. All models used momentum constant $\mu = 0.5$ asymptotically grew to $\mu = 0.95$. The 'leakiness' of the Twin network ReLU units was 0.01.

Name	T-LSTM Hidden Dim	T-LSTM N Children	Twin Net Hidden Dim
Very Small (VS)	300	2	300
Small (S)	350	3	350
Medium 1 (M1)	450	2	450
Medium 2 (M2)	400	2	512
Large (L)	600	2	600

Table 1: Description of the five experimental models architectures. T-LSTM N Children indicates the maximum number of right or left children to consider before summing across the remaining children, as described in the appendix.

Name	Image Type	Batch Size	α	N Training Examples	Run Time (Hrs:Mins)
VS	VGG16	10	5×10^{-4}	495,000	23:46
S	VGG16	50	5×10^{-4}	11,550	102:07
M1	VGG16	10	5×10^{-4}	371,000	113:55
M2	VGG19	10	1×10^{-4}	328,000	99:00
L	VGG16	20	1×10^{-3}	168,000	103:51

Table 2: Experimental model running parameters, where α is the learning rate.

5 Results

The effectiveness of the models was assessed using the standard 'Recall@K' metric, which relates the percentage of times the model assigned each test image the correct caption in the first K instances and *vice versa* for captions, in the same manner as [5] and [4].

Our best performing model was VS, which was also trained on the largest amount of data. See Table 3 for details. When tested using 1,000 image / caption pairs, the model ranked the correct caption at number 1 a total of 33 times, and the correct image 24 times (the next best model was 'L', with R@1 values of 27 and 23, respectively.

6 Conclusion

The results demonstrate the effectiveness of the model, and specifically T-LSTMs, at mapping natural language and visual content into a shared space. Keeping in mind that the training was abbreviated due to time constraints, it is not unreasonable to hypothesize that these models would perform substantially better given a longer training period (and that they remain training).

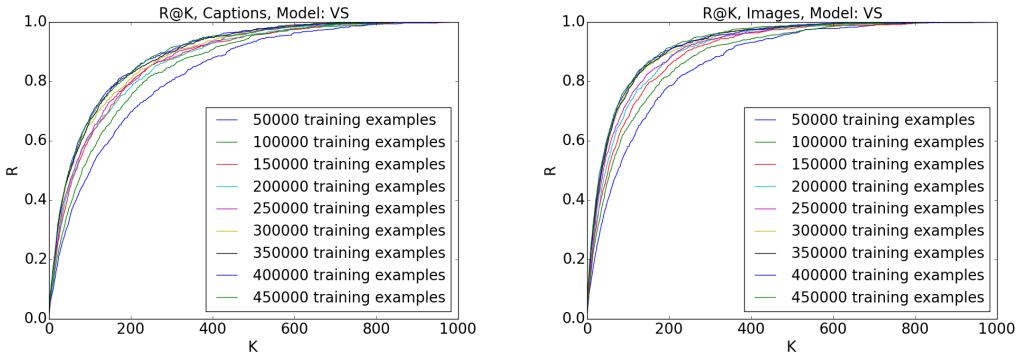

Figure 3: R@K on Very Small Model

Model	Image to Caption					Caption to Image				
	R@1	R@5	R@10	R@20	R@50	R@1	R@5	R@10	R@20	R@50
VS	3.3	14.0	24.2	37.4	61.0	2.4	9.5	16.2	28.5	49.7
S	1.1	5.2	8.4	14.0	28.2	1.8	4.8	7.3	13.7	25.1
M1	2.1	10.0	18.3	30.9	53.1	1.4	6.8	13.7	21.7	43.5
M2	3.4	10.6	19.9	32.2	56.8	1.8	7.8	14.2	24.8	46.9
L	2.7	12.9	20.8	32.8	58.1	2.3	10.2	17.9	27.9	47.5
BRNN [3]	38.4	69.9	80.5	-	-	27.4	60.2	74.8	-	-

Table 3: R@K Values (higher is better) for the various models compared to the current state-of-the-art (BRNN [3]). All of these were obtained from a test set of 1,000 images / caption pairs.

Because the training was time-restricted, it is difficult to make any remarks with respect to the effectiveness of the hyperparameter choices, although earlier runs with ultra small networks (both hidden dims at 200) show that smaller networks tend to saturate before completing a run over the dataset (i.e., their expressive capacity and hence learning ability is maximized). Nonetheless our best-performing model was also our smallest (although it also trained on the most data), and in general the largest predictor of performance was the amount of training completed (unsurprisingly). Early exploratory work indicated that differences between VGG16 and VGG19 were minimal.

As found in previous studies, the model is better able to map images to captions than the other way around. This is potentially because the degree of transformation was reduced for images, as the CNN networks are fixed whereas training error was propagated back through the T-LSTM.

Note: The relevant codes and dataset can be found on our Github repository at `https://github.com/pveerina/imgcap`

References

[1] D. Chen and C. D. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, 2014.

[2] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, pages 853–899, 2013.

[3] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*, 2014.

[4] A. Karpathy, A. Joulin, and L. Fei-Fei. Deep fragment embeddings for bidirectional image sentence mapping. *CoRR*, abs/1406.5679, 2014.

[5] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014.

[6] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[7] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Explain images with multimodal recurrent neural networks. *arXiv preprint arXiv:1410.1090*, 2014.

[8] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12, 2014.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[10] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014.

[11] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015.

[12] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[13] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

[14] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

A T-LSTM Model

The T-LSTM model uses unique sets of parameters for the first $p-1$ left children and the first $p-1$ right children (where p is a user-specified parameter) and the left nodes thereafter share a set of parameters and right nodes thereafter share another set of parameters. That is, in effect, the states and the outputs of all the left children index beyond and including p are summed up into a single effective state and output and the same is done for right nodes indexed beyond and including p. If a node has $m > p$ left children and $n > p$ right children, then

$$c_{lp} \leftarrow \sum_{k=p}^{m} c_{lk} \qquad h_{lp} \leftarrow \sum_{k=p}^{m} h_{lk} \qquad c_{lp} \leftarrow \sum_{k=p}^{n} c_{lk} \qquad h_{lp} \leftarrow \sum_{k=p}^{n} h_{lk} \tag{2}$$

With these assumptions, the forward propagation equations are listed below.

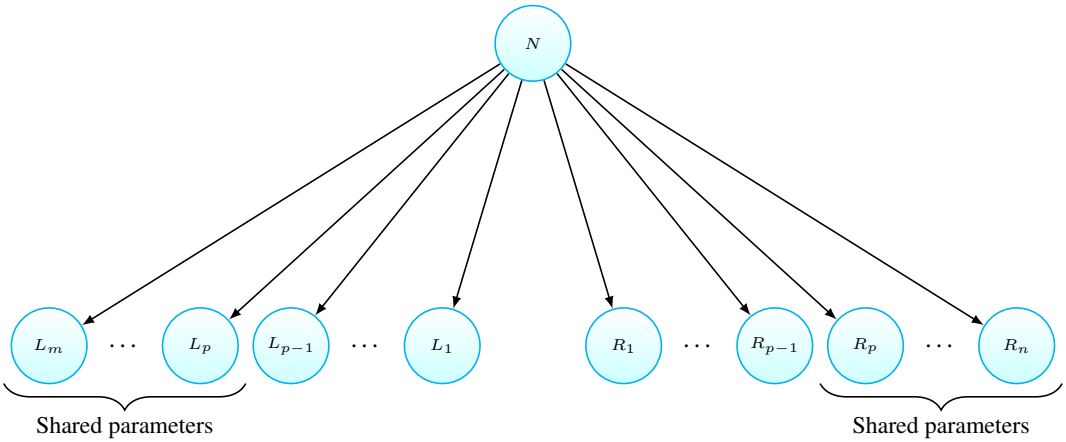

Figure 4: The figure shows the typical structure of the tree for LSTM network. One particular node along with its children is shown as an illustration.

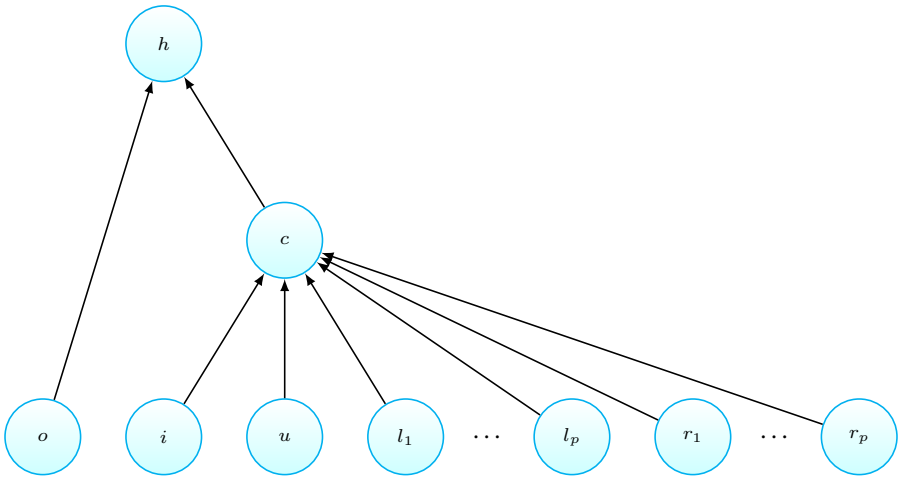

Figure 5: A Typical T-LSTM node. h and c of a node act as inputs to the parent node (in forward prop) and i, o, u, $\mathbf{l}$, $\mathbf{r}$ are computed using inputs from children nodes.

$$h = o \odot \tanh(c) \tag{3}$$

where h of the root node is the embedding for the corresponding caption and is further fed into the "twin" network as input.

$$c = i \odot u + \sum_{k=1}^{m} l_k \odot c_{lk} + \sum_{k=1}^{n} r_k \odot c_{rk} \tag{4}$$

where c_{lk} is the state of the k^{th} left child and c_{rk} is the state of the k^{th} right child.

$$l_j = a_l(W^{(f)}x + \sum_{k=1}^{m} U_{jk}^{(l)} h_{lk} + \sum_{k=1}^{n} V_{jk}^{(l)} h_{rk} + b^{(f)}) \tag{5}$$

$$r_j = a_r(W^{(f)}x + \sum_{k=1}^{m} U_{jk}^{(r)} h_{lk} + \sum_{k=1}^{n} V_{jk}^{(r)} h_{rk} + b^{(f)}) \tag{6}$$

$$u = a_u(W^{(u)}x + \sum_{k=1}^{m} U_{k}^{(u)} h_{lk} + \sum_{k=1}^{n} V_{k}^{(u)} h_{rk} + b^{(u)}) \tag{7}$$

$$o = a_o(W^{(o)}x + \sum_{k=1}^{m} U_{k}^{(o)} h_{lk} + \sum_{k=1}^{n} V_{k}^{(o)} h_{rk} + b^{(o)}) \tag{8}$$

$$i = a_i(W^{(i)}x + \sum_{k=1}^{m} U_{k}^{(i)} h_{lk} + \sum_{k=1}^{n} V_{k}^{(i)} h_{rk} + b^{(i)}) \tag{9}$$

where a_j's are the activation functions, i is the input from children, u is the update gate, $\mathbf{l}$ and $\mathbf{r}$ are the forget gates for left and right children respectively, o is the output gate, c is the node state and h is the node output.

B T-LSTM Backprop

B.1 Error Flows

There are a total of $4+2p$ error outlets from each parent to each of its children. $h_{\text{child}} \rightarrow o$, $h_{\text{child}} \rightarrow i$, $h_{\text{child}} \rightarrow u$, $h_{\text{child}} \rightarrow \{l_1, \ldots, l_p\}$, $h_{\text{child}} \rightarrow \{r_1, \ldots, r_p\}$, $c_{\text{child}} \rightarrow c$.

Total Error at h : let the total error at h be denoted by e_h.

$$e_h = \delta_o U_k^{(o)} + \delta_i U_k^{(i)} + \delta_u U_k^{(u)} + \sum_{j=1}^{p} \delta_{lj} U_{jk}^{(l)} + \sum_{j=1}^{p} \delta_{rj} U_{jk}^{(r)} \tag{10}$$

where δ_j's are the input errors from parent node.
Note: In the above equation, it is assumed that the node under consideration is the k^{th} left child of its parent. If the node is a right child, replace all U-parameters in the equation by the corresponding V-parameters.

Total Error at c : let the total error at c be denoted by e_c.

$$e_c = \delta_c \text{diag}(l_k) + e_h \frac{\partial h}{\partial c} \tag{11}$$

Note: In the above equation, it is assumed that the node under consideration is the k^{th} left child of its parent. If the node is a right child, replace f_l by f_r.

Output Errors

let the output errors (going from node to its children be denoted by Δ_j's.

$$\Delta_o = e_h \text{ diag}(\tanh(c)) \, \Sigma^{(o)} \tag{12}$$

$$\Delta_i = e_c \text{ diag}(u) \, \Sigma^{(i)} \tag{13}$$

$$\Delta_u = e_c \,\text{diag}(i)\; \Sigma^{(u)} \tag{14}$$

$$\Delta_{lk} = e_c \,\text{diag}(c_{lk})\; \Sigma^{(l_k)} \tag{15}$$

$$\Delta_{rk} = e_c \,\text{diag}(c_{rk})\; \Sigma^{(r_k)} \tag{16}$$

$$\Delta_c = e_c \tag{17}$$

where $\Sigma^{(j)} = \text{diag}(a'_j)$ and a'_j denotes the elementwise derivative of the activation function for a_j.

B.2 Parameter Derivatives

= defined only for non-leaf nodes

Bias Terms

$$\frac{\partial J}{\partial b^{(j)}} = \Delta_j; \;\; \frac{\partial J^{\#}}{\partial b^{(f)}} = \sum_{k=1}^{p} \Delta_{lk} + \sum_{k=1}^{p} \Delta_{rk} \tag{18}$$

where $j \in \{o, i, u\}$

U, V Parameters (#)

$$\frac{\partial J}{\partial U^{(j)}} = h_l \Delta_j; \;\; \frac{\partial J}{\partial V^{(j)}} = h_r \Delta_j \tag{19}$$

where $j \in \{o, i, u\}$

$$\frac{\partial J}{\partial U^{(l)}_{jk}} = h_{lk} \Delta_{lj}; \;\; \frac{\partial J}{\partial U^{(r)}_{jk}} = h_{lk} \Delta_{rj}; \;\; \frac{\partial J}{\partial V^{(l)}_{jk}} = h_{rk} \Delta_{lj}; \;\; \frac{\partial J}{\partial V^{(r)}_{jk}} = h_{rk} \Delta_{rj} \tag{20}$$

W Parameters

$$\frac{\partial J}{\partial W^{(j)}} = \sum_j x \Delta_j \tag{21}$$

where $j \in \{o, i, u, l_k, r_k\}$ and $k = 1 \ldots p$

C Implementation

Our comple implementation is available online at: https://github.com/pveerina/imgcap