Image Captioning

Xinze Liu
xl2822
`xl2822@columbia.edu`

Kun Liang
kl3056
`kl3056@columbia.edu`

Xiaoyun Qin
xq2189
`tc2945@columbia.edu`

Ting Cai
tc2945
`tc2945@columbia.edu`

Abstract

Image captioning, generating a description of an image, is one of the most important applications in artificial intelligence involves computer vision and natural language processing. The difficulty of image caption is that it requires the machine not only recognize the important objects and their attributes in the image but also the relationships between them. In this report, we present three different implementations on an encoder-decoder base model, which are the base model with multi LSTM layers, the base model that uses the pre-train GloVe word vectors, and the base model with the attention mechanism. We implement all the models on the flickr8k data set and compare their performances through the BLEU score.

1 Introduction

Nowadays, images have played an important role in daily life from various sources including the Internet, news articles, document diagrams, and advertisements. Though they are always displayed without specific descriptions, people can easily interpret what these images are trying to tell. However, the machine needs to interpret some form of image captions if humans need automatic image captions from it. The task of interpreting the image in a sentence is hard but meaningful since it helps the visually impaired to have a better understanding of images. Unlike relative straightforward tasks such as image classification and object detection, the challenging of image caption is it needs to detect the objects in the image as well as the relationship, expression, and activity between those objects. Moreover, it also demands to construct the semantics information in a sequence of words in human readable natural language. The task of detecting the salient visual information (relationship, expression and activity) involving objects seems not a problem for human beings as a result of instinct. However, it is the most difficult part for image caption.

The mainly two sub-tasks in this project are to visually determine the objects and their relationships in the image and then to describe it in a sentence that satisfies natural language semantics. In the model proposed in the paper we try to combine this into a single model, which consists of a Convolution Neural Network (CNN) encoder, which helps in creating image encoding. We use the Inception V3 architecture as pre-trained encoder and these encoded images are then passed to a long short-term memory (LSTM) network, which is a type of Recurrent Neural Network.

The rest of the report is organized as follows. We first discuss the related work in the image captioning area. Then we show the methods and experiments we used to improve the model in detail. In the end, we present the experimental results and analyze the reason behind the different performances.

Final Project of STAT GR5242 Advanced Machine Learning (Fall 2019), Section:002.

2 Related Work

Many methods proposed for object detection and recognition recently have significant impacts on motivating image caption. One of the most art-of-state approaches is encoder-decoder architecture-based image captioning. The general idea of the encoder-decoder model first introduced in 2014 by Google aim to map a fixed-length input with a fixed-length output while the length for input and output can be different. For the application in the image caption, usually, the image features are extracted from the hidden activation of CNN and then put them into LSTM model to generate a sequence of words.

Over the years, there are many different approaches have been developed. In the early stage, Farhadi at al.[2] described a system that computing a score to attach a descriptive sentence to a given image. Based on the achievements of scholars like Farhhadi, Vinyals et al.[9] in 2015 proposed Neural Image Caption (NIC), a generative model involves a convolution neural network as image encoder to represent high-level features of the input image and use the output of the last layer as input to generator recurrent neural network. Our approach is similar to NIC in integrating pre-trained networks for vision and language model while we directly use a built-in model as pre-trained networks, which is Inception V3 proposed by Google in ILSVRC 2015.[8]

Word embedding is frequently used for machine to generate sequences. For image captioning, there are only a few applications of word embedding. Jiang Wang et al.[4] used word embedding for multi-label image classification. They combined word embedding with neural network to capture the dependencies between objects in images. Zeynep Akata et al.[10] used Word2Vec to create vector representations of objects and help the machine learn the relationships between classes.

Attention is widely used in the computer version area to put attention on objects. There are two main methods to add attention mechanism in image captioning area. Kelvin Xu et al. [5]improved their image captioning model by estimating spatial attention for words using LSTM, while Quanzeng You et al. [7]refined the image captioning model by computing the semantic attention for words.

3 Method

3.1 Baseline Model

In the project, we use an encoder-decoder model as our base model to perform image captioning. The model takes a single raw image and generates a sequence of words encoded y representing caption.

$$y = \{y_1, y_2, \ldots, y_C\}, y_i \in \mathbb{R}^K \tag{1}$$

We use CNN as the encoder for images and LSTM as the decoder to generate captions. For the encoder, our task is capturing information from images encoded as an feature vector:

$$z = \{z_1, \ldots, z_L\}, z_i \in \mathbb{R} \tag{2}$$

Instead of training a CNN from scratch, we use an off-the-shelf pre-trained image encoder, Inception V3 [8]. The model is a version of a convolution neural network for object detection. Its output is typically a softmax-activated vector representing 1000 possible object types. Because we are interested in an encoded representation of the image, we just extract the second-to-last layer as the image encoding. Then, each image will be encoded as a vector of size $L = 2048$. This 2048-dimensional image encoding is connected to a 300-dimensional hidden layer and the output is concatenated with each embedded input words as the input of the decoder.

For the decoder, we use a bidirectional LSTM to generate a caption by predicting one word at each step based on a context vector, the previous hidden state, and the previously generated words.

3.2 GloVe Vector based Model

Comparing to baseline model, here we use a pre-trained GloVe[3] word vectors instead of training the vector embedding by itself. GloVe is a widely used word embedding method. Comparing to Word2Vec, it not just relies on the context of the words but also contains the co-occurrences of word pairs in a large corpus. The intuition of using pre-trained GloVe vectors is that we believe the semantic relationships captured by GloVe should be very similar with the dependencies learned by using captions. Since our corpus are very small, using these word vectors should result in a better understanding of the similarities between words and improve the performance of the image captioning models.

For the words and symbols that do not exist in GloVe, we randomly initialize them. For randomly initialization, we use Xavier initialization.

$$Var(w_i) = \frac{1}{fan_in + fan_out} \tag{3}$$

3.3 Attention

We implemented attention mechanism similar to the work of Kelvin Xu et al[5]. There are three steps in the attention part:

1. Capture the visual information associated with a particular input caption time.
2. Update the encoded image with the extracted information.
3. Combine the information from image and caption by putting the adjusted image and the original encoded caption into a Bi-LSTM layer.

The whole architecture of the attention mechanism in our experiment is shown by the figure 1.

In the first step, we use a bidirectional long short-term memory network that extracts the association between image location and caption time. Our implementation of LSTM follows equation4. $T_{s,t} : \mathbb{R}^s \to \mathbb{R}^t$ denotes a simple affine transformation with parameters that are learned. i_t, f_t, c_t, o_t, h_t are the input, forget, memory, output and hidden state of the LSTM, respectively. The vector y_{t-1} is the encoded caption. The vector $z_i \in \mathbb{R}^D$ is the encoded image. $E \in \mathbb{R}^{m \times K}$ is an embedding matrix. The encoded image dimension and LSTM dimension are denoted by m and n respectively. σ and $\odot$ are the logistic sigmoid activation and element-wise multiplication respectively.

$$\begin{pmatrix} t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} Ey_{t-1} \\ z_i \\ h_{t-1} \end{pmatrix} \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \tag{5}$$

$$h_t = o_t \odot tanh(c_t) \tag{6}$$

In the second step, we define an attention function f_{att} that computes adjusted image $\hat{a_t}$ from the encoded vectors z_i, $i = 1, \ldots, L$ and the encoded caption. L is the number of features extracted at different image locations. The attention weight α_i for location i could be interpreted as the relative importance given to location i. In this report, we choose the tanh function as our attention function.

$$e_{ti} = f_{att}(z_i, h_{t-1}) \tag{7}$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\Sigma_{k=1}^{L} exp(e_{tk})} \tag{8}$$

Once the attention weights are obtained, the encoded image is adjusted to $\hat{a_t}$, which is computed by

$$\hat{a_t} = Multiply(\{z_i\}, \{\alpha_i\}) \tag{9}$$

In the third step, we concatenate the adjusted image $\hat{a_t}$ and the encoded caption and then put them into the Bidirectional LSTM layer to capture the combination information from image and caption. Since the implementation of Bidirectional LSTM is similar to the first step, we will not show the computation detail in this step.

3.4 Inference

After building the conditional language model which can predict the next word based on the image, we will generate a sentence given an image using decoding algorithm. In this report, we will use two different decoding algorithms, one is greedy decoding and the other one is beam search.

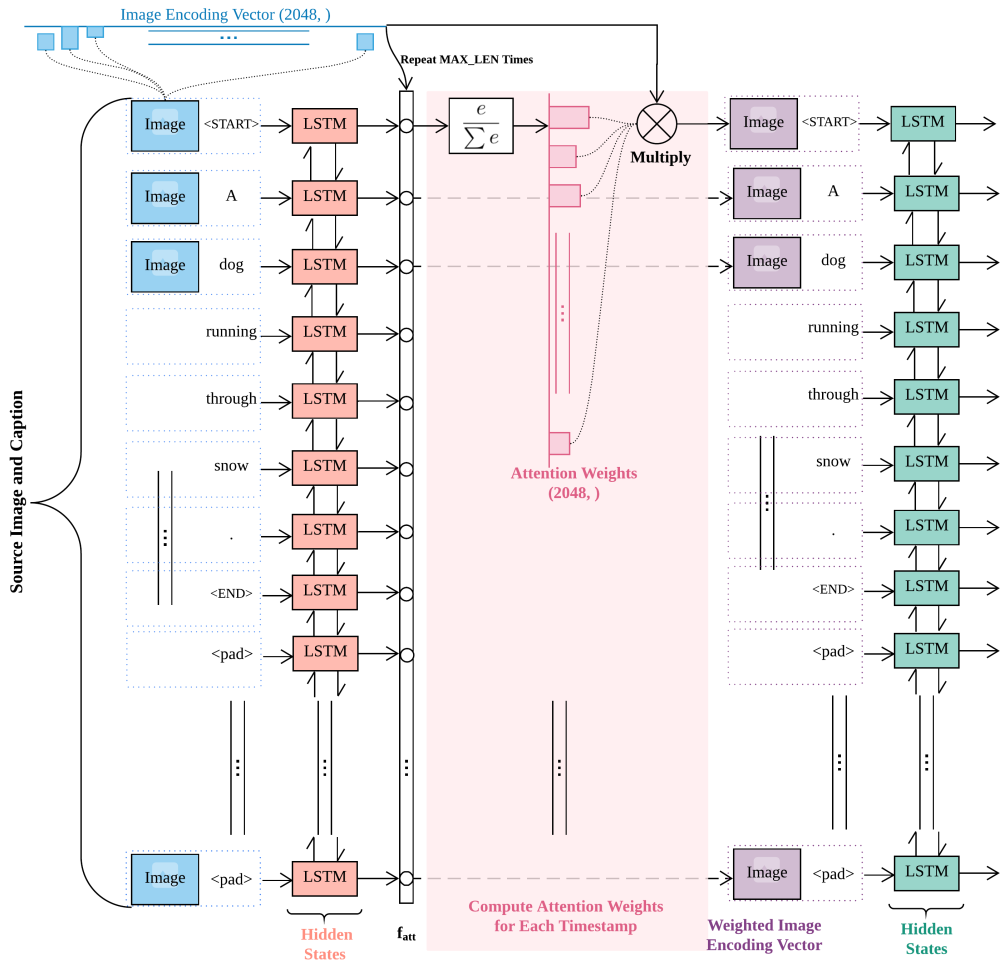


Figure 1: Architecture of the Attention Mechanism

3.4.1 Greedy Decoding

Greedy decoding is a simple algorithm. The decoder starts with the sequence starting word(["<START>"]), use the model to predict the most likely word at each step, append the predicted word to the sequence and then continue iterate until the stopping word("<END>") is predicted or the sequence reaches the maximum length of sentences.

3.4.2 Beam Search

In greedy decoding, we focus on a single word at each step. Using a beam search, instead of always selecting the most probable word, it aims at searching a high-probability sentence. On each step of decoder, it keeps track of the K(K is the beam size) most probable partial sequences and their total probability. Then, for each sequence, it computes the K most likely successor words, appends the word to produce K new sequences and computes their score.

$$score(y_1,..,y_t) = logP_{LM}(y_1,..,y_t|x) = \sum_{i=1}^{t} P_{LM}(y_i|y_1,..,y_{i-i},x) \tag{10}$$

Then after each step, it prunes the list to include only the K most probable sequences. Prune the list until the sequence has the stopping word("<END>") or reaches the maximum length of sentences.

4 Experiment

4.1 Description of Data

For the data part, we use flickr8k data set, which contains 8000 images with 5 captions per image extracted from Flicker to train the model. This data set mainly contains natural images of human and animals. We split the 8000 images into 6000 images as train set, 1000 images as validation set and the rest 1000 images as test set.

4.2 Evaluation

The models are scored by how well the sentence generated are similar to human written sentences. There are several automatic evaluation metrics to evaluate the natural language generation like CIDER, METEOR, BLEU scores. The most commonly used metric in the image description has been the BLEU score and we will use this score for our model evaluation and comparison.

BLEU, the bilingual evaluation understudy score, is a metric for evaluating the similarity of a machine generated sentence to one or several human-written reference sentences. It compute the similarity score based on n-gram(usually for 1, 2, 3 and 4-grams) precision and give a penalty for short translations as the formula below:

$$BLEU = exp(min(1 - \frac{r}{c}, 0) + \sum_{n=1}^{N} w_n log(p_n)) \tag{11}$$

Here, c is the total length of machine generated translation corpus, and r is the average length of all references. As the candidate length decreases, the ratio r/c increases, and the BLEU score decreases exponentially. We will prefer models with higher BLEU scores.

4.3 Implementation Details

All variants of our model were trained with RMSprop using adaptive learning rate.

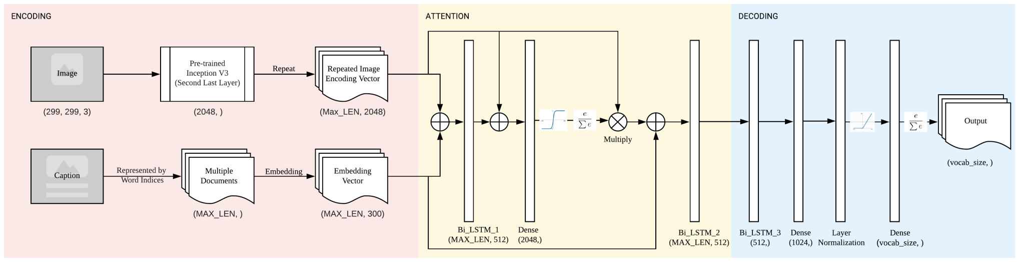


Figure 2: Overview Architecture of the Best Model

4.3.1 Data cleaning

We first do data cleaning before training the model as we usually concerned the word as a whole rather than character-level for text data. For data cleaning, we lowercase all the words (like regarding 'Cat' as 'cat'), remove special tokens (like '.','?', '.', etc.) and eliminate words that contain numbers. However, we find the performance of the model after doing data cleaning is worse than the one without doing data cleaning. The reason behind this phenomenon might be the punctuation such as comma and period will provide position information, which strengthen the relationship between those punctuation and their surrounding text so that it helps to predict the words after a target word. Thus, we finally decide not to do data cleaning. The detailed performance comparison of doing data cleaning and not doing it is listed in the table in result section.

4.3.2 Comparison of Different Batch Sizes

During the implementation, we notice an interesting phenomenon. We train our models using different batch sizes to compare their impact on neural network. We set a relative large batch size to 1024 and a relative small batch size to 128 and then separately use them to train the model. In our experiments, the larger batch size get better performance than the small batch size. Larger batch sizes always converge fast faster and sometimes give better performance because it improves the effectiveness during optimization and reduces the communication overhead. Therefore, we decide to use the larger batch size (1024) to train the variants version of models.

4.3.3 Comparison of Beam Size

We also compare how the choice of beam size may influence the sentences generation. In general, a large beam size has better performance than a small beam size in beam search. Small beam size produces similar results to greedy decoding. When the beam size equals to one, it equals to greedy decoding. Larger beam size could produce more grammatical and natural language. However, larger beam size is more computationally expensive. In our final inference, we set beam size to 3 to reach a balance.

4.3.4 Training Baseline Model

The baseline model has two inputs, image input and caption input. The image input is encoded by the pre-trained Inception V3 model. The caption input is encoded by word-to-id dictionary and then put into the Embedding layer of Keras. We concatenate the two inputs and put them into the bidirectional LSTM layer with 256 hidden size. After that, we use the dense layer with softmax activation to predict the next word.

The training data set contains 6000 images and each picture has 5 descriptive sentences. The validation data set contains 1000 images and 5000 captions correspondingly. Due to the large training data set, we build generator to produce training data in batch.

We would train the model using the input/output pairs shown in Table 1:

Table 1: Input Output format

Image Input	Caption Input	Output
Image-1	[<'START'>]	['a']
Image-1	[<'START'>,'a']	['small']
Image-1	[<'START'>,'a','small']	['dog']
Image-1	[<'START'>,'a','small','dog']	['.']
Image-1	[<'START'>,'a','small','dog','.']	['<END>']

We choose RMSprop optimizer with initial learning rate 0.001. We use callbacks function in Keras to monitor the validation accuracy and adjust learning rate if the validation accuracy is stuck. The callbacks function also allows the model to save the best model with highest validation accuracy during training process automatically.

4.3.5 Adding Multiple LSTM layers to the baseline model

To earn more accuracy, we try to stack LSTM hidden layers to make the model deeper. We add another bidirectional LSTM layer above the bidirectional LSTM layer in the baseline model. The two bidirectional LSTM layers have same number of hidden units, 256. The first bidirectional LSTM layer provides a sequence output rather than a single value output to the second bidirectional LSTM layer.

4.3.6 Adding GloVe to the baseline model

We use pre-trained GloVe word vectors with size of 300 in order to compare randomly initializing the caption embedding layer with using pre-trained word vectors as the weight of caption embedding layer.

We will not retrain the caption embedding layer using pre-traned word vectors in our experiment. For the words and symbols that do not exist in GloVe, we initialize them using Xavier initialization.

4.3.7 Adding attention mechanism to the baseline model

Instead of putting the concatenated inputs into the bidirectional LSTM layer directly, we put the inputs into the attention mechanism introduced by section 3.3. The output of the attention mechanism is connected to the bidirectional LSTM layer. Then we add layer normalization before the softmax classification layer. The key feature of layer normalization is that it normalizes the inputs across the features, which makes it more suitable for RNN model. The optimizer is the same as the baseline model.

5 Results

Through the experiment we explained above, the performance of all models is shown in Table 2. Here, we only show the details of result produced by the best model: Base + Attention. The rest can be found in our GitHub[1].

As we could see from the table, data cleaning dose not improve the performance of the model. Both accuracy in development data set and BLEU scores of the base model with data cleaning step are lower than the base model without data cleaning. This may because the punctuation will provide position information, which strengthen the relationship between those punctuation and their surrounding text so that it helps to predict the words after a target word.

Adding GloVe does not significantly improve the performance of base model. After using the pre-train GloVe word vectors in context embedding layer, all the scores are very similar to the base model. The BLEU-3 and BLEU-4 score are higher than the base model while the BLEU-1 and BLEU-2 socre are lower than the base model. The difference may because we add the caption embedding layer using GloVe word vectors without retraining it in the model. The dependencies between words in image captioning area may have slightly differences with the relationships between words in broader context. This may be the reason why adding GloVe slightly decrease the model's performance in BLEU-1 and BLEU-2. However, since GloVe word vectors are training using large corpus, it has great performance in BLEU-3 and BLEU-4. Besides, using GloVe word vectors, the model converge more quickly than without it. It is computational efficient to add pre-trained word embedding models in image captioning area.

The model with two stacked bidirectional LSTM layers outperforms the baseline model using both greedy decoding and beam decoding. The highest validation accuracy of this model is higher than the base model's, which meets our expectation. Adding one more bidirectional LSTM layer increases the model's capacity. The additional information in the hidden states of the first bidirectional LSTM layer helps to improve the model's performance.

The attention-based model is standout among all models experimented. Using both greedy decoding and beam decoding, it turns out the highest BLEU scores. The attention mechanism helps in capturing

Table 2: Performance of All Models

Models	**Val-acc**	**Inference**	**BLEU-1**	**BLEU-2**	**BLEU-3**	**BLEU-4**
Base	0.419	Greedy	0.560	0.356	0.258	0.138
		Beam	0.573	0.370	0.273	0.153
Cleaned Base	0.385	Greedy	0.547	0.367	0.274	0.154
		Beam	0.561	0.379	0.288	0.164
Base + Multi LSTM	0.431	Greedy	0.578	0.373	0.276	0.154
		Beam	0.585	0.377	0.284	0.164
Base + GloVe	0.425	Greedy	0.556	0.351	0.260	0.143
		Beam	0.569	0.363	0.273	0.155
Base + Attention	0.437	Greedy	0.584	0.378	0.279	0.157
		Beam	0.593	0.389	0.293	0.168

the major part of the features provided by the image decoder, corresponding to each word in the caption. However, it costs much longer time to tune parameters and learn until convergence. In terms of the improvement brought by the attention, this trade-off is worthy and acceptable.

5.1 Tensor board

We use Tensor Board to monitor the training process, including the changes of accuracy on train data, loss on train data, accuracy on validation data, loss on validation, and the learning rate. The model architecture can also be observed. Details are shown in the Figure 3-6 at the end of the document.

5.2 Examples

To illustrate what the final generated captions look like, we take some images as examples, shown in the Appendix. Since the base model with attention mechanism reaches the highest BLEU score, we use this model to generate captions. The performance of the model varies among the different kind of image. Thus, we discuss the model's performance in case of different kinds of images.

5.2.1 Simple Images

We first choose three simple images with one subject and an uncluttered background. The captions generated by the best model are fluent and match the main information of the images. They are quiet approximate to the captions provided in the data set. This might because the subject and the background could be distinguished easily.

5.2.2 Complicated Images

Then, three relatively complicated images with multiple subjects or a mixed background are chosen. In this case, the performance of the generated captions is not satisfactory enough. Our model could recognize the part of the color or part of the subject in the image. But it will mismatch between the color and the subjects.

6 Conclusion and Future Work

A relatively effective image caption generator is produced in our project. During the experiment, data cleaning, multi-LSTM, pre-trained GloVe, and attention mechanism are attempted and compared. The attention based model can generate reasonable captions for simple images, and relatively imperfect captions for Complicated Images.

There are some strategies can be explored in the future:

- Enlarge the data set, like using MS COCO[6].
- Enrich features captured from complicated images to help in recognizing multiple subjects.
- Based on the decoder output, retrain some layers of image encoder from the back forward.
- Tune the size of the vector of hidden states in LSTM.

References

[1] Github. https://github.com/XiaoyunQIN/5242Project.

[2] Ali Farhadi, Mohsen Hejrati, Amin Sadeghi, and Peter Young. *Computer Vision - ECCV 2010*. 2010.

[3] Christopher D. Manning Jeffrey Pennington, Richard Socher. Glove: Global vectors for word representation, 2014.

[4] Junhua Mao Zhiheng Huang Chang Huang Wei Xu Jiang Wang, Yi Yang. Cnn-rnn: A unified framework for multi-label image classification., 2016. CVPR.

[5] Ryan Kiros Kyunghyun Cho Aaron Courville Ruslan Salakhutdinov Richard Zemel Yoshua Bengio Kelvin Xu, Jimmy Ba. Show, attend and tell: Neural image caption generation with visual attention, 2016. https://arxiv. org/abs/1502.03044.

[6] Maire Michael Belongie Serge Hays James Perona Pietro Ramanan Deva Dolla r Piotr Lin, Tsung-Yi and C Lawrence Zitnick. Microsoft coco: Common objects in context. *In ECCV, pp*, pages 740–755, 2014.

[7] Zhaowen Wang Chen Fang Jiebo Luo Quanzeng You, Hailin Jin. Image captioning with semantic attention, 2015. https://arxiv.org/abs/1503.08677.

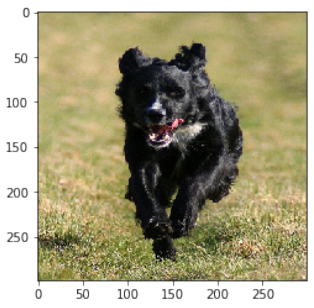
[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jonathon Shlens. Rethinking the inception architecture for computer vision. 2015.

[9] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *Proceedings of the IEEE Image Captioning conference on computer vision and pattern recognition*, 2015.

[10] Zaid Harchaoui Cordelia Schmid Zeynep Akata, Florent Perronnin. Label-embedding for image classification, 2015. https://arxiv.org/abs/1503.08677.

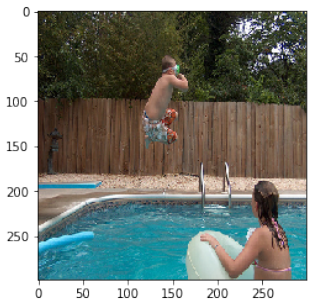
Appendix

Three examples of simple images.



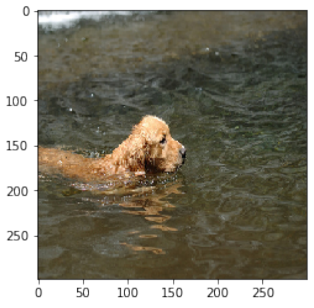
Greedy Decoder:
a black dog runs through the grass.

Beam Search:
a black dog is running through the grass .
a black and white dog is running through the grass .
a black and white dog is running through the grass .



Greedy Decoder:
a boy in a blue shirt is jumping into a pool .

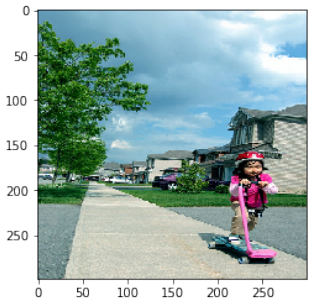
Beam Search:
a boy in a blue shirt is jumping into a swimming pool .
a boy in a blue shirt is jumping into a swimming pool .
a boy in a blue shirt is jumping into a swimming pool .



Greedy Decoder:
a dog is swimming in the water .

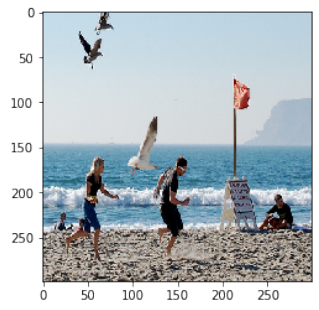
Beam Search:
a dog swims in the water .
a dog swims in the water .
a dog swims in the water .

Three examples of complex images.



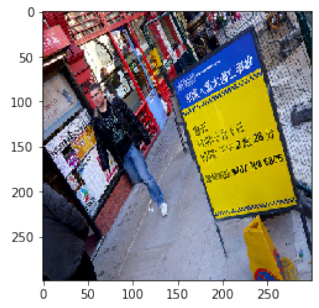
Greedy Decoder:
a boy in a red shirt is jumping on a skateboard .

Beam Search:
a boy in a red shirt is jumping on a skateboard .
a boy in a red shirt is jumping on a skateboard .
a boy in a red shirt is jumping on a skateboard .



Greedy Decoder:
a man in a blue shirt is jumping into a lake .

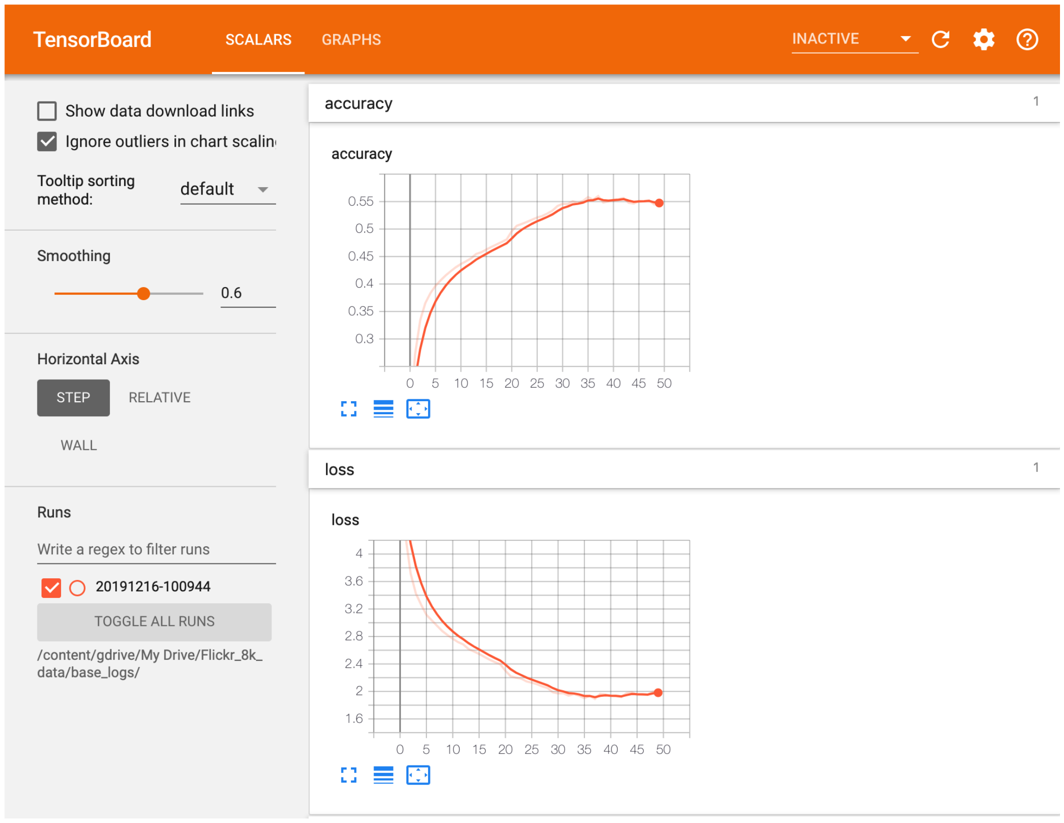
Beam Search:
a man in a blue shirt is jumping into the air into the water .
a man in a blue shirt is jumping into the air into the water .
a man in a blue shirt is jumping into the air into the water .



Greedy Decoder:
a man in a black shirt and a white shirt is standing on a sidewalk .

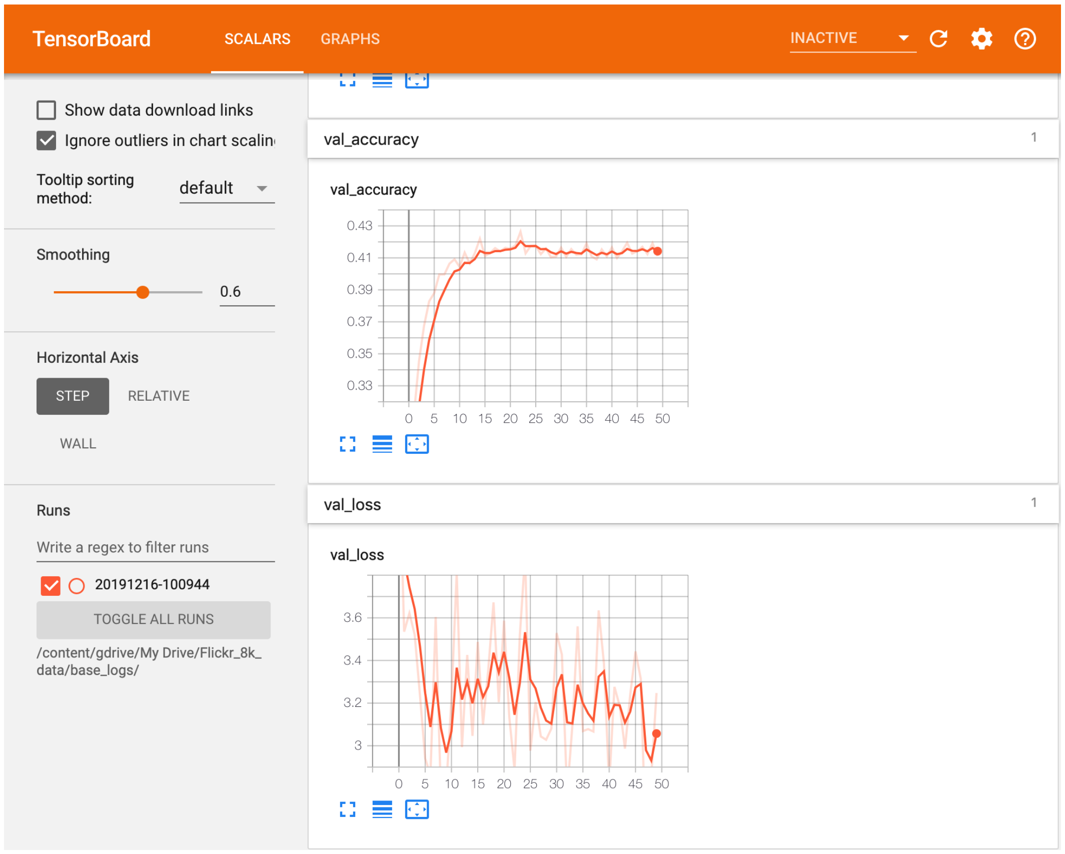
Beam Search:
a man in a black shirt is standing in front of a white building .
a man in a black shirt is standing in front of a white building .
a man in a black shirt is standing in front of a white building .

Tensor board examples.



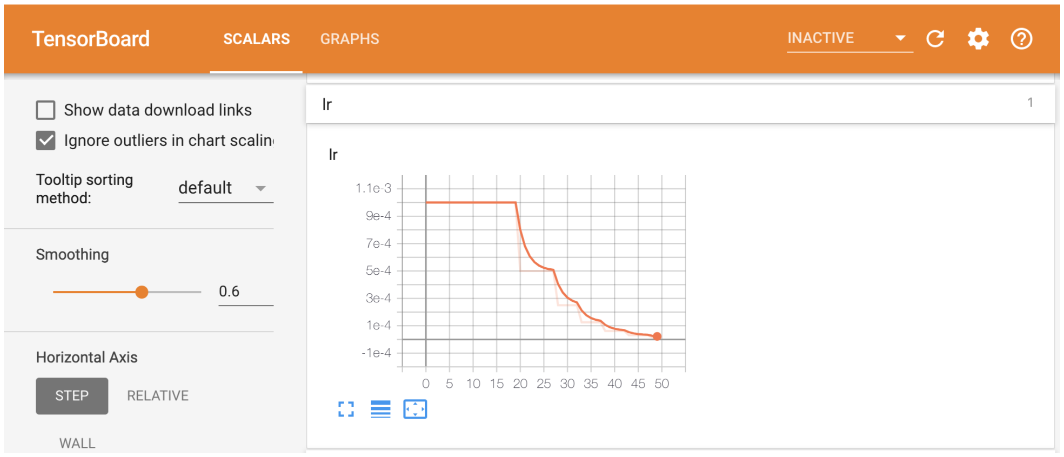
Show data download links
✓ Ignore outliers in chart scalin
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis
STEP RELATIVE
WALL
Runs
Write a regex to filter runs
✓ 20191216-100944
/content/gdrive/My Drive/Flickr_8k_data/base_logs/

Figure 3: Tensorboard Example on Training data of Training Process



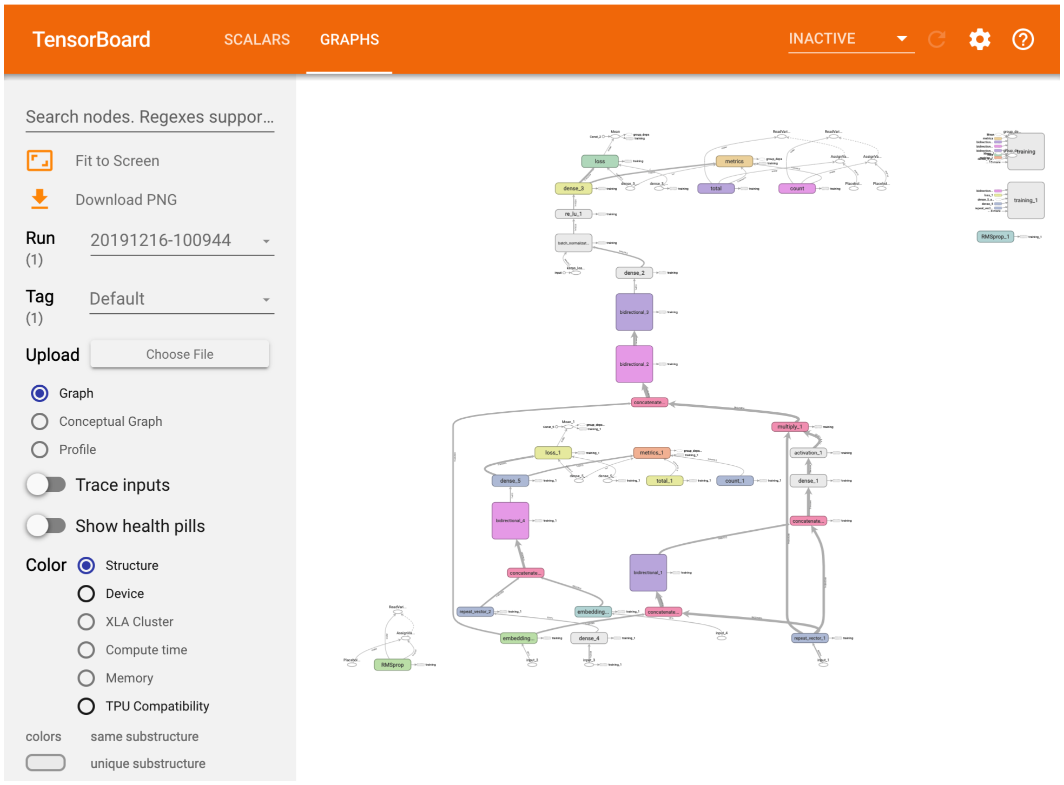
☐ Show data download links
☑ Ignore outliers in chart scalin
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis [✓ STEP] RELATIVE WALL
☑ 20191216-100944
/content/gdrive/My Drive/Flickr_8k_data/base_logs/

Figure 4: Tensorboard Example on Validation data of Training Process



☐ Show data download links
☑ Ignore outliers in chart scalin
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis [✓ STEP] RELATIVE WALL

Figure 5: Tensorboard Example on Learning Rate of Training Process


TensorBoard SCALARS GRAPHS INACTIVE

Search nodes. Regexes suppor...

Fit to Screen

Download PNG

Run (1) 20191216-100944

Tag (1) Default

Upload Choose File

[✓ Graph]
Conceptual Graph
Profile

Trace inputs

Show health pills

Color [✓ Structure]
Device
XLA Cluster
Compute time
Memory
TPU Compatibility

colors same substructure
unique substructure

Figure 6: Tensorboard Example on Graphs of Training Process