Pre-trained BERT Evaluation On Image Captioning

Group: XP_system
Member: ZeMing Gong, Zhuo Ning

Abstract

Image captioning is about generating a few sentences to describe the input image. As a task that crosses Computing Version (CV) and Natural Language Process (NLP), to solve it, the model must be able to both recognize the main objects in the image and generate semantically fluent sentences. In this project, we implemented a simple image captioning model with ResNet (He et al., 2015) and Gate Recurrent Unit (GRU) (Chung et al., 2014). Through simple twisting and 5 epochs training on Google Colab, it achieved 31 BLEU-1 (Papineni et al., 2002) on the COCO 2014 dataset (Lin et al., 2014). Beyond, we also explored the possibility of using Mask R-CNN (He et al., 2017), VisualBERT (Li et al., 2019), and Transformer (Vaswani et al., 2017) decoder to construct a stronger model as a new solution to solve this task. In the end, our Mask R-CNN + VisualBERT + Transformer decoder model achieved 77.2 BLEU-1 on the COCO 2014 dataset.

A vandalized stop sign and a red beetle on the road

Figure 1: An example of Image captioning's input and output

1 Introduction

Image captioning is a very interesting and popular research area that brings together both CV and NLP. By definition, image Captioning is "the process of generating a textual description for given images." In short, it means the model should take in an image as input and generate a few lines of sentences that are used to describe the image as output.

In practical application, image captioning tools can help to solve many issues. A simple example is image search. By generating a short description for images, users can quickly find the image they want by typing in keywords. Besides, for people with eye diseases, image captioning is a very meaningful tool. Combined with other devices, image captioning can help visually impaired people to perceive their surroundings. On an academic level, for many other CV/NLP tasks such as visual question and answering, image captioning is an important fundamental part.

Since both members in the group are also taking Computational Vision this semester, we choose image captioning as the topic of our final project. We hope to apply the knowledge we learned in these two courses to this project.

To solve the image captioning problem, the model must accurately identify the objects and other related information in the input picture, then transform the information into logical sentences. Traditionally, these two tasks are solved by Convolutional Neural Network (CNN) (Krizhevsky et al., 2012) and Recurrent Neural Networks (RNN) (Zaremba et al., 2014). In this project, we will implement two improved models with Mask R-CNN, Transformer and VisualBERT applied. Compared with the image captioning models formed with fundamental CNN and RNN models, our solution should have better performance. Compared with examining the improvement brought by the use of

Mask R-CNN and Transformer which are shown in other papers, we are more inclined to check the difference brought by using VisualBERT in the image captioning model.

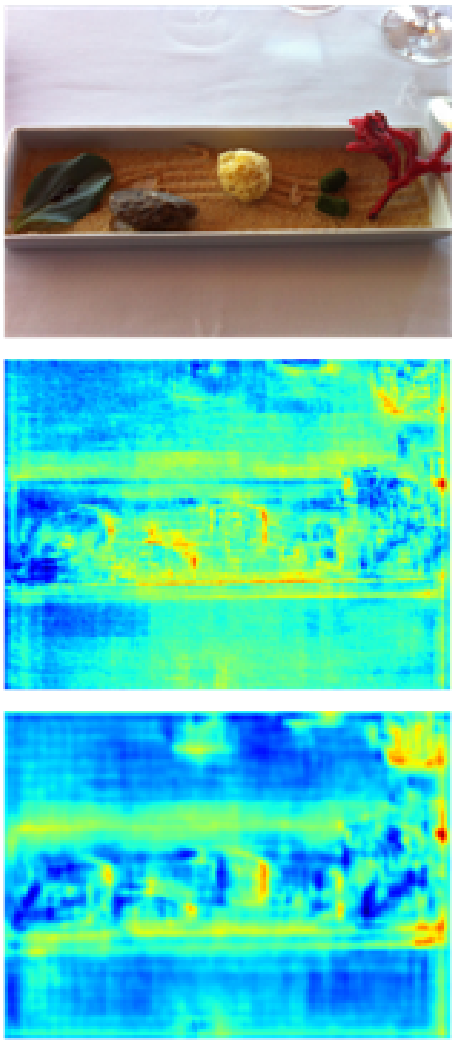


Figure 2: Example of features that extracted from the image

2 Related Work

Similar to other CV and NLP problems, many early image captioning solutions had a similar structure with two parts, encoder, and decoder.
The encoder can be seen as an image feature extractor, which is used to convert an image into encoded image features without classifying the image. Image feature extractor is usually implemented by CNN such as VGG16 and ResNet without the dense layers at the end. With the image features, we can use it as the visual embeddings of the original image. On the other hand, the decoder can be viewed as a caption generator. It is usually based on RNN such as Long Short Term Memory (Hochreiter and Schmidhuber, 1997) and Gate Recurrent Unit. They take the image features as input and generate logical and syntactically fluent sentences to describe pictures.

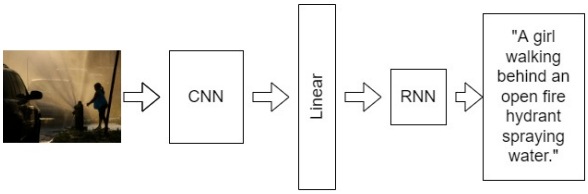


Figure 3: A simple diagram that shows the input, output and structure of the image captioning model.

The workflow of the image captioning model is shown in figure 2. The features of the image extracted by the CNN model. The shape of it can be viewed as a low-resolution picture with many color layers (For example, $14 \times 14 \times 2048$). Then, since the RNN models take sequence embedding as input, there is a Linear layer that is used to convert the image features to one dimension feature embedding. In the end, the RNN as decoder will recursively generate the description of the image as the final output based on the extracted feature.
As debuted in 2017, the Transformer model is more and more widely used in solving various NLP problems. Compared to RNN models, the Transformer abandoned the traditional sequential structure and only used attention. It performs much better than other RNN models such as the Long short term memory model with different kinds of attention. In the article "A Comprehensive Survey of Deep Learning for Image Captioning" (Hossain et al., 2018), the researchers introduced how they used Transformer to replace the RNN as the decoder of their image captioning model and achieved better results than the other solutions at that time. This inspired us to use Transformer as the caption generator in our project.
Besides the improvements on the decoder, researchers also attempt to use a better feature extractor to get a better result. Compared with models like VGG and ResNet that were originally used to identify the single main object in the image, models like Mask R-CNN that are designed to detect multiple objects in the image can extract more comprehensive information. The paper "CapSal: Leveraging Captioning to Boost Semantics for Salient Object Detection" (Zhang et al., 2019) showed how to use Mask R-CNN as a Local Perception Module

(LPM) to better solve image captioning tasks. In our project, we also attempted to use Mask R-CNN to extract More detailed feature information.

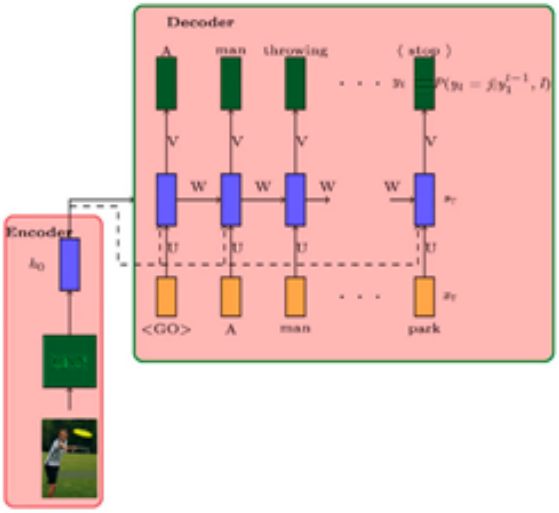


Figure 4: The architecture of traditional image captioning model with attention applied

3 Approach

For comparison, we constructed three different ensemble models to solve the image captioning task. Each ensemble model is formed by an image feature extractor and a caption generator.

3.1 Baseline: Wide ResNet 101 + GRU

- Image feature extractor: Wide ResNet 101
- caption generator: GRU

To start and get familiar with the structure of the image captioning model, we referenced PyTorch's tutorial of image captioning and built a naive solution with ResNet and GRU model that are integrated and provided by PyTorch (PyTorch).
In CV, ResNet is an excellent model for image classification. By removing the last layers that are used for classification and adding a fully connected layer, the output will be a one dimension feature embedding. After using ResNet to extract the feature information of the input image, GRU will repeatedly generate hidden states. In the end, using the greedy search, the model can generate English statements that describe images with these hidden states. The reason we choose GRU is, compared with LSTM, GRU holds fewer parameters, which speeds up the training process. At the same time, it still has a good ability to learn.
The input data are images data $(Height \times Width \times Colour(3))$ with RGB channels. For Wide ResNet 101, we need to resize or crop the image into the same size and make batches. For each batch, our input is a $(Batch \times Colour(3) \times 144 \times 144)$ matrix, the 144×144 is the image size after cropping. And the output of ResNet's backbone is a $(Batch \times 14 \times 14 \times 2048)$ matrix, where the 14×14 is the size of the image feature and 2048 is the number of image features' layers. Then, after using a fully connected layer to convert image features to one dimension embedding with 512 features, the size output matrix is $(Batch \times 512)$.
We then feed all the feature vectors into the encoder block of the GRU model. The output of the encoder block is our hidden state. Feeding the hidden state and the tokenized start token to the decoder sequentially, we will get our final output. For GRU, we set the hidden layer size to 512 and the number of layers to 1, which are enough for a simple image captioning decoder.

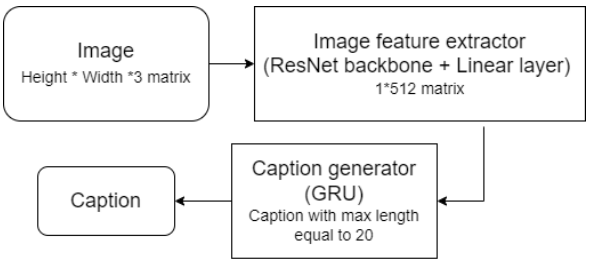


Figure 5: The architecture of ResNet + GRU model with output size of each component

3.2 Improve: Mask R-CNN + Transformer

- Image feature extractor: Mask R-CNN
- caption generator: Transformer

In this part, we want to examine how much improvement can be achieved by using Mask R-CNN and Transformer. The Mask R-CNN architecture is available in Detectron2 (Detectron2) and the Transformer is available in PyTorch. As mentioned before, compared with ResNet, Mask R-CNN is designed for detecting multiple objects in the image. Hence, we believe it will extract more sufficient feature information than ResNet. However, using Mask R-CNN to extract features is much harder than using ResNet. By referring to a tutorial by Chhablani (Chhablani, 2021), we managed to extract the main box features of the input images.
For Mask R-CNN with a feature pyramid network, we do not need to resize or crop the image into the

same size. Also, Mask R-CNN's outputs are box features over the main objects in the image. The form of these features is a $(Batch \times 100 \times 1024)$ matrix. Where 100 stands for box number and 1024 is the feature number of each box.

In this section, we chose the Transformer provided by PyTorch as a caption generator, since it has performed much better performance than the RNN models on many other CV and NLP tasks. The Transformer we use here has 6 encoder layers, 6 decoder layers, and the number of heads equal to 8.

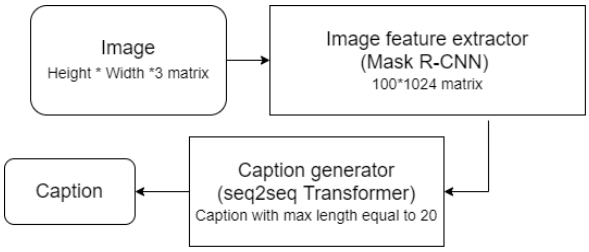


Figure 6: The architecture of Mask R-CNN + Transformer model with output size of each component

3.3 Final: Mask R-CNN + VisualBERT + Transformer

- Image feature extractor: Mask R-CNN
- caption generator: VisualBERT + Transformer decoder

In 2018, BERT: Pre-trained Bidirectional Transformers (Devlin et al., 2018), an extension of Transformer encoder, will leap the performance to a new stage. Through stacking Transformer encoders and pre-training with general tasks using large amounts of data, BERT can generalize to most of the existing NLP problems and achieve better results in most of them. Moreover, BERT is also able to adapt visual tasks. Hence, in this final model, we want to see if we can task advantage on a BERT-like model to get better image captioning results.

After searching, we found VisualBERT, a pre-trained model that is designed to help solve many version-and-language problems such as visual question answering and visual reasoning. We can easily use it since it is integrated by huggingface (Hugging Face). In our final solution, we used it to transform the image features to more language-related embedding before we use another Transformer decoder to decode it.

Compared to using a typical Transformer encoder as the caption generator encoder, we believe the pre-trained VisualBERT model that is designed for cross CV and NLP tasks performs better. Note that VisualBERT is trained with features extracted from Mask R-CNN. Thus, the use of VisualBERT also gives another reason to use Mask R-CNN.

In this final model, we used the VisualBERT and a Transformer decoder as our caption generator. The Transformer decoder has 6 layers and 8 heads.

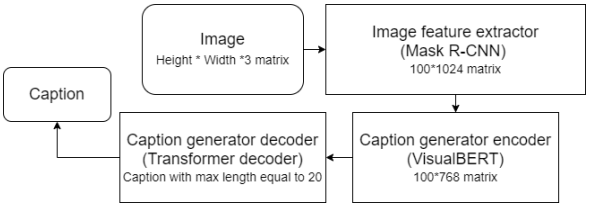


Figure 7: The architecture of Mask R-CNN + VisualBERT + Transformer model with output size of each component

3.4 Loss function

In short, we used the cross entropy loss which is available in PyTorch library to complete the training of all three caption generators. This calculates the token-wise cross entropy between the caption generator output and the grand truth caption label. However, the models are trained in a more complex way, the detail of training progress will be explained in the next section.

4 Experimental Setup

Figure 8: Introduction of COCO dataset

4.1 Data

In our plan, we wanted to use Mask R-CNN as the image feature extractor of our second and third models. To train Mask R-CNN, it requires the dataset must have a rich number of object categories labeled with a well-labeled segmentation

mask. Besides, since we are working on image captioning, the dataset must also have a proper caption for every image. After searching, we find that the COCO dataset exactly matches our requirement. In brief, COCO contains about 330,000 image data with different sizes. It contains 1.5 million object instances in 80 object categories, each with a perfectly labeled segmentation mask. Also, they provide 5 or more captions for each image in the dataset. After consideration, we decided to use the COCO dataset 2014 to train and validate our models since both of the team members had experience with it.

4.2 Overview of training progress

Since we are training on Google Colab, to avoid the training process being terminated by Google due to the long-term use of GPU, we decided to separate the training of image feature extractor and caption generator.

For the baseline model, we skipped the training progress of the ResNet backbone to save more time. Since the ResNet model provided by PyTorch was pre-trained with the COCO dataset, it can extract meaningful features in this scenario. During the training progress of the baseline model, we used the cross entropy loss to train the last linear layer and GRU model together.

Since the process of extracting features by Mask R-CNN and training is very time-consuming, in the second and third models, we individually pre-trained the Mask R-CNN with detertron2. Moreover, It is very tedious to extract image feature information using Mask R-CNN and this process does not support batch processing. If we choose to perform this progress for every image during the training progress, the training time will be unacceptably long. Therefore, we chose to extract the feature and put it in the H5 file before the training progressed. However, the time it took to train the Transformer models was still very long. In the end, during the training of the second and final models' caption generator, we chose to train the randomly selected one-eighth of the training set for every epoch to save more time.

4.3 Hyper-parameters

4.3.1 Baseline: Wide ResNet 101 + GRU

- We used Following hyper-parameters to train the linear layers and GRU model:
- Number of epochs: 5
- Batch size: 50
- Learning rate: 0.00001
- Optimizer: SGD (Ruder, 2016) with momentum equal to 0.9
- Scheduler: One Cycle Policy (Smith and Topin, 2017) with max learning rate 0.1

4.3.2 Mask R-CNN

- We used Following hyper-parameters to train the Transformer model:
- Max iteration: 100
- Base learning rate: 0.00025
- Images per batch: 4
- Batch size per image: 256
- Weight decay: 0.001
- Momentum: 0.9

4.3.3 Improve: Mask R-CNN + Transformer

- For Mask R-CNN, we trained it with the default trainer of detectron2 with following hyper-parameters:
- Number of epochs: 50
- Learning rate: 0.00001
- Batch size: 64
- Optimizer: Adam (Kingma and Ba, 2014) with betas equal to (0.9, 0.98) and eps equal to 1e-9
- Scheduler: One Cycle Policy with max learning rate 0.0001

4.3.4 Final: Mask R-CNN + VisualBERT + Transformer

- We used Following hyper-parameters to train the VisualBERT encoder and Transformer decoder:
- Number of epochs: 20
- Learning rate: 0.0001
- Batch size: 64
- Optimizer: Adam with betas equal to (0.9, 0.98) and eps equal to 1e-9
- Scheduler: One Cycle Policy with max learning rate 0.0001

4.4 Evaluation Matrix

We decided to use BLEU as our estimation standard. BLEU can simply detect how many identical words and terms appeared between the prediction and the reference. It is undeniable that BLEU is an imperfect scoring standard. Although in many cases the caption generated by the model has a similar meaning to the reference caption, the word-level difference will cause the BLEU score to become lower. However, BLEU is still a relatively intuitive, easy-to-understand, and widely used standard in evaluating NLP models.

4.5 Results

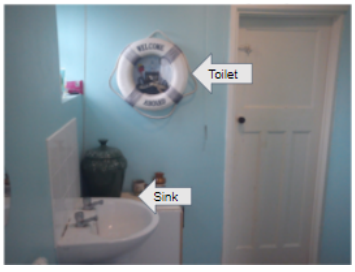
After a few epochs of training and some twisting, here are the BLEU scores of our models compared with other image captioning models on the COCO captions benchmark. The models we compared with are, Meshed-Memory Transformer (Cornia et al., 2019), Transformer NSC (Luo, 2020), RefineCap (Chai et al., 2021), RDN (Ke et al., 2019), and the result is as follow.

Architecture	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Mashed-Memory Transformer	**80.8**	-	-	39.1
Transformer_NSC	80.7	**65.6**	**51.3**	**39.4**
RefineCap	80.2	64.5	49.9	37.8
RDN	80.2	-	-	37.3
Baseline (Ours)	31.8	0	0	0
Improved (Ours)	57.8	24.8	12.6	8.1
Final (Ours)	**77.2**	49.6	31.9	21.2

Table 1: BLEU Score Comparison

There is still a gap in the BLEU score of our final model compared to other models on the benchmark, but if there is more time for us to further twist and train our model, this gap is expected to be further narrowed.

4.6 Analysis



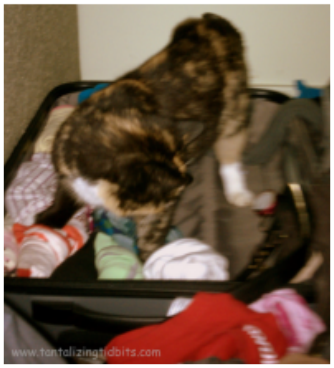
Hypothesis: a bathroom with a toilet and sink References:A room with blue walls and a white sink and door

Figure 9: A caption generated by the ResNet + GRU model.

4.6.1 Baseline: Wide ResNet 101 + GRU

Here is a representative prediction made by the baseline model that can be used for analysis. For this image, our model mistakenly thinks there is a toilet. However, there is not. We conjecture that the model detected a sink and therefore considered this room to be a bathroom. The swim ring on the wall, as a white circular object, is identified by the model as a toilet. Because it's reasonable to have a toilet in a bathroom since the model has seen many bathrooms with sinks and toilets. This kind of mistake has happened more than once. It shows that our decoder is more dependent on self-deriving information instead of analyzing the extracted features. This kind of situation is probably caused by the deficiency of the ResNet. Beyond this, there are also some images that the model prediction is mostly wrong. This may also prove that we made a mistake in the design of the experiment, that is, we should carry out some classification training on ResNet to improve its recognition ability on this dataset.

Hypothesis: person is holding tennis racquet , wearing uniform , and tennis ball tennis racquet . References: the woman is holding her tennis racket .



Hypothesis: cat sitting on top of laptop computer . References:a cat that is resting in a full bag of luggage .

Figure 10: Two predictions made by the Mask R-CNN + Transformer model.

4.6.2 Improve: Mask R-CNN + Transformer

By implementing Mask R-CNN and Transformer, We can see a significant increase in the BLEU score. Also, by observing many captions generated by this model, we found that, although the Mask R-CNN + Transformer model can not always generate the correct caption, it does not generate a completely wrong caption as the baseline model did. We believe that this is related to the extra training of Mask R-CNN and its ability to extract box features that are more relevant to each object.

4.6.3 Final: Mask R-CNN + VisualBERT + Transformer

Hypothesis: close up of plate of doughnuts on table
References: a close up of a plate of doughnuts on a table

Figure 11: A prediction made by the Mask R-CNN + VisualBERT + Transformer decoder model.

After applying VisualBERT and performing 13 only epochs training, we found a significant improvement in BLEU scores. Meanwhile, for about 20 randomly selected images, this final model generated captions that are reasonably similar to reference captions. We believe the BLEU score and the quality of captioning can be further improved by performing more training, but unfortunately, Google Colab terminated our training process after 13 epochs. However, with the result we have so far, we still can see the significant improvement brought by VisualBERT compared to the previous model. In the early stages of our project, one of the reasons that prompted us to use a BERT model was, we were worried that the transformer would take a long time to train to achieve certain results. In contrast, the pre-trained BERT model should require only a small amount of fine turns to bring better results. And here, the improvement of our final model compared to the second model agrees with our initial hypothesis, which is, the improvement brought by VisualBERT is immediate compared to using Transformer Encoder.

5 Limitations

Our models and experiments are not perfect due to the shortcomings of pre-planning and time constraints.

On the one hand, the span between the baseline model and the final model we designed and built is too large, and the training uses different parameters and times. As a result, the comparability between models is not high enough to make a clear and fair comparison. For example, we cannot be sure whether the performance gap between the baseline model and the second model is more caused by replacing ResNet by Mask R-CNN or replacing GRU by Transformer.

We discovered these flaws in planning in the later stage of the project, but due to limited time, we finally decided to spend time on implementing and improving the last two models.

The shortage of time mainly comes from two aspects. On the one hand, the complexity Mask R-CNN as an image feature extractor exceeded our expectations. In fact, in the early stages of the project, besides doing more searches to complete the plan, most of the time was spent on researching the code used to obtain image features by Mask R-CNN. It delayed the completion of the code and ended up squeezing the time we planed for training the models.

Another reason for the time constraints is that we chose to work on Google Colab virtual machines. The reason why we chose Google Colab is that it does provide a very ideal development environment. Compared with using our computer and GPU for training, Google Colab provides a more complete development environment and higher GPU memory. Moreover, since we chose to use Mask R-CNN that is provided by Detectron2 and Detectron2 is not officially supported on Windows, choosing Google Colab can help us bypass complex and uncertain environmental configuration problems. However, Google Colab has a hidden GPU using restrictions. This represents that during long-term model training, the training process may be terminated without warning. The training of our final

model was forced to be interrupted at the thirteenth epoch. We were aware of this risk and saved the model's parameters after each epoch. Although we did not complete the 20 epochs training as expected, we still achieved some results.

To further improve the performance of the last two models, firstly, we should further twist the hyper parameters of the models and conduct further training. Judging from the current rate of loss reduction, our models should still have some room for improvement. On the other hand, our current data loader will select the first caption of each image for training. But, as mentioned above, every image in the COCO dataset has at least five labeled captions. We should use all captions to train the models in a more complete way. This can effectively reduce the skewness of the models.

6 Conclusion

Although there are certain flaws in our experimental design, and some accidents occurred during the implementation and training phase of models, according to the result, using Mask R-CNN, Transformer and VisualBERT can build a better image captioning model. After seeing the improvement brought by using VisualBERT as the encoder caption generator, we think that one of our early hypothesis has also been verified, that is, compared to using transformer directly, when conditions permit, using a pre-trained BERT model can bring better results faster.

Beside these, after actually looking at the difference between the captions generated by the model and the reference captions, we think that BLEU was an intuitive but narrow way to score. However, inventing an objective but human-like mechanism to evaluate an image caption may be as difficult as perfectly performing image captioning. So far, we may only be able to rely on BLEU and other word-based evaluation metrics.

Contributions

7 Contributions

7.1 Zhuo Ning

Completed the last two Models, edited and completed the final report, and attended to the presentation.

7.2 ZeMing Gong

Completed the baseline Model, wrote the draft of the final project report, and made the demonstration video.

References

Yekun Chai, Shuo Jin, and Junliang Xing. 2021. Refinecap: Concept-aware refinement for image captioning. *CoRR*.

GunJan Chhablani. 2021. Generating visual embeddings using detectron2 for visualbert.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. Cite arxiv:1412.3555Comment: Presented in NIPS 2014 Deep Learning and Representation Learning Workshop.

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. 2019. M^2: Meshed-memory transformer for image captioning. *CoRR*, abs/1912.08226.

Detectron2. 2021. Detectron2. Https://github.com/facebookresearch/detectron2.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. Mask R-CNN. *CoRR*, abs/1703.06870.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Md. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. 2018. A comprehensive survey of deep learning for image captioning. *CoRR*, abs/1810.04020.

Hugging Face. 2021. Hugging face. Https://huggingface.co/.

Lei Ke, Wenjie Pei, Ruiyu Li, Xiaoyong Shen, and Yu-Wing Tai. 2019. Reflective decoding network for image captioning. *CoRR*, abs/1908.11824.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. *CoRR*, abs/1908.03557.

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

Ruotian Luo. 2020. A better variant of self-critical sequence training. *CoRR*, abs/2003.09971.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

PyTorch. 2021. Pytorch. Https://pytorch.org/.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Leslie N. Smith and Nicholay Topin. 2017. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. Cite arxiv:1409.2329.

Lu Zhang, Jianming Zhang, Zhe Lin, Huchuan Lu, and You He. 2019. Capsal: Leveraging captioning to boost semantics for salient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.