Image Caption with Full-Transformer

by

Jiahao Li

1155166092

A report

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

The Chinese University of Hong Kong

04/24/2022

Prof. Wong Kin Hong

Abstract

In this final thesis, we propose a full-attention sequence to sequence architecture to tackle the image captioning and vision questions answering tasks, called Captioning and Answering TransformeR (CATR). The CATR receives raw images and the start token as the input and automatically generates the descriptions of images. Compared to the traditional architectures (CNN encoder with RNN decoder) design paradigms, our model is convolutional-free. The current researches announce that the attention mechanism, more specifically the transformer, has quite an excellent performance in both Computer Vision (CV) and Natural Language Processing (NLP) regions. Meanwhile, a traditional RNN-based decoder cannot take the computations parallelly, making the training and evaluation cost much time. As for the new state-of-the-art (SOTA) architecture in Image Caption, called the Caption Transformer (CPTR), it is the first full-transformer model to deal with the Image Captioning task on the MSCOCO dataset. It declares that the full-attention-based architecture, so-called transformer, is good enough to connect the CV and NLP aspects. Whereas, there are still some drawbacks in the CPTR, though its works are fantastic. They are: 1) Directly transferring the traditional transformer from NLP to CV faces the huge time complexity when the resolution of the raw image is too fine. 2) Although the traditional transformer has an outstanding performance in extracting features of the global contexts, it is poor to extract the finer-level features in the image. However, both global and local features are significant in the CV tasks. Therefore, to overcome the above problems of the CPTR, we consider the following pipeline: 1) We employ the outlook-attention-based Vision Outlooker (VOLO) to replace the self-attention-based transformer in the encoder of the CPTR to extract the visual features, which can extract the fine and coarse features like the convolutional neural network (CNN). 2) We modify the standard decoder into two transformer-based decoders to simultaneously decode the visual and linguistic features to establish an information exchanging opportunity for both vision and linguistic context to complete the questions answering tasks.

1. Introduction

Image Captioning and Vision Questions Answering are pretty excited and challenging tasks. Image Caption requires the model to receive an image and automatically generate the description of it. Beyond this, the Vision Questions Answering considers understanding the context in the image and answering some questions by following its context. Therefore, to solve these two tasks must combine the efficient architectures and mechanisms in both Computer Vision and Natural Language Processing regions. These two tasks can help some visually impaired people to get knowledge about the scenes. The most helpful architecture to tackle these two tasks is the sequence to sequence (Seq2Seq) model [30]. Seq2Seq architecture consists of two independent neural networks, which are encoder and decoder. The encoder deploys to generate the features representation of the input data, so-called embedding. The decoder is responsible for generating the expected output from the embedding. It means that the Seq2Seq architecture is quite efficient in transforming a data representation into another, which is suitable for the two tasks mentioned before. We wonder to convert the data representation in an image into sentence.

There are three main stages with the development of the Seq2Seq architecture [30] in Image Captioning during the past five years. 1) Combination of the CNN encoder and RNN decoder [33], they employ the efficient CNN model [29] [11] [31], which is pre-trained by ImageNet classification tasks to be the encoder to produce the embedding of the image. Afterward, the decoder receives the visual representation to be the initial hidden state of the RNN [7], more specifically LSTM [8], to output the description word by word. 2) Due to the attention mechanism is efficient in machine translation, which is another data representation conversion issue. The researchers start deploying the attention mechanism into the Image Captioning and Vision Questions Answering tasks [34] [21] to find out what image sub-regions have to pay more attention to when generating sentence word by word. More detailed, there are two kinds of attention mechanisms. One is 'Hard' attention which directly assigns each word in the expected sentence to a fixed region in the image. Another one is 'Soft' attention which permits the neural network to learn the attention by itself. Besides, there are one varients which is called Visual Sentinel. Unlike the 'Hard' and 'Soft' attentions that focus on where to see in the image when producing the sentence, Visual Sentinel aims to figure out when the prediction of a word has to see the image. 3) With the development of the 'Hard' attention mechanism, researchers realize that the essence is to use the fixed regions in the image to predict the words. Hence, some of them started

to apply the object detection models [10] [9] [27] [24] [25] [26] [3] to generate the fixed regions. This pipeline consists of Bottom-Up and Top-Down architecture [1].

Inspired by the fantastic performance of the transformer [32] in machine translation and other NLP tasks, recently, some researchers have started transplanting this new sequence to sequence architecture into the Image Captioning and Vision Questions Answering. Meanwhile, the researchers are encouraged by successfully transplanting the transformer into some other vision tasks, like object detection and image classification, more specifically, Detection Transformer (DETR) [4] and Vision Transformer (ViT) [6]. The current state-of-the-art Neural Image Caption Generator (NIC) called Caption Transformer (CPTR) had been issued. Unlike the other transformer-based NIC, the CPTR is the first full-transformer-based NIC. It means that the CPTR replaces the traditional CNN-based encoder with a transformer. Whereas, like the DETR and the ViT, CPTR employs the traditional transformer architecture [32] either. It means that the CPTR has the same problems to which the DETR and the ViT face. The traditional transformer obtains a huge time-complexity when the image resolution is giant either and is poor to extract the local features from the images, so-called finer-level features. To remedy these drawbacks, researchers published the Swin Transformer [20] and Vision Outlooker (VOLO) [38]. In particular, the VOLO firstly obtained accuracy in ImageNet-1K image classification, which exceeds 87%. In this case, we provide our full-transformer-based architecture called Caption and Answering Transformer (CATR). It is consists of a VOLO-based encoder and a traditional transformer-based decoder. Meanwhile, we propose a two-decoders architecture to make the information in the visual and linguistic context can be exchanged.

To sum up, we are looking forward to modifying the CPTR architecture in the following way:

- Apply the Vision Outlloker to remedy the lack of finer-level feature extracting capability of the traditional transformer-based vision encoder.
- Design a new comparable full-attention-based Neural Image Captioning architecture.

As for the remaining parts, the section 2 will cover some previous Neural Image Captioning architectures. The section 3 will be theoretical introduction of our CPTR architectures. The section 4 and 5 will be the results, analysis and discussions of our experiments. In the end, the section 6 covers the conclusions of our final thesis.

2. Background

With the development of the Image Caption and Vision Questions Answering during the past five years, the Neural Image Caption Generator (NIC) [33] is the first CNN-RNN-based sequence to sequence architecture to tackle Image Captioning. Afterward, visual attention [34] has been proposed to combine with the NIC. Moreover, with a deeper understanding of the visual attention mechanism, the visual sentinel-based NIC [21] has been published. The Faster-RCNN-based [27] Bottom-Up and Top-Down Attention NIC [1] come to the public to modify the 'Hard' visual attention. Due to the transformer being quite popular in both Computer Vision and Natural Language Processing regions. More researchers have started paying attention to how to generate a good NIC with the transformer architecture [19].

2.1. Neural Image Caption Generator

The Neural Image Caption Generator (NIC) is the first Neural Network-based architecture to deal with Image Captioning. Before NIC, all the models are heavily human-designed. It means that all of them have poor generalization capabilities when the evaluating distribution is not suitable for the human-designed technique. Compared to all the human-designed architectures, the NIC established a new state-of-the-art (SOTA) performance at that time, which is a 27.7 BLEU-4 score on the COCO dataset.

To successfully deploy the backpropagation in training the NIC, [33] proposes an optimization target which is still applied right now. Literature speaking, the target is to obtain parameters of the NIC that can maximize the probability of the correct description of the input image. The formulation is shown in 1.

$$\theta^* = \underset{\theta}{argmax} \sum_{(I,S)} log\ p(S|I;\theta) \tag{1}$$

where θ^* is the optimum, θ is the parameters of the NIC, I is the image and S is the sentence. Since S represents any sentence and we can produce the word t only when we obtain the previous $t-1$ words. Therefore, the $log\ p(S|I)$ can be described like formula 2.

$$log\ p(S|I) = \sum_{t=0}^{N} logp(S_t|I, S_0, ..., S_{t-1}) \tag{2}$$

where S_i is the i^{th} word in the sentence S.

The architecture of the NIC is pretty simple, illustrated in figure 2, it consists of a CNN-based encoder and an LSTM-based decoder. The output of the encoder will be unfolded into a one-dimensional vector and treated as the initial hidden state in the first time step of the LSTM. As for the convolutional encoder, the NIC chose the InceptionNet [31], so-called GoogleNet, to extract the visual embeddings from the input image. Due to the InceptionNet has an outstanding performance to extract the visual features from different scales. After getting the visual embedding, the LSTM is applied to generate the description of the image word by word. NIC only deployed one-layer and one-directional LSTM to predict the descriptions. The LSTM unit is illustrated in figure 1.

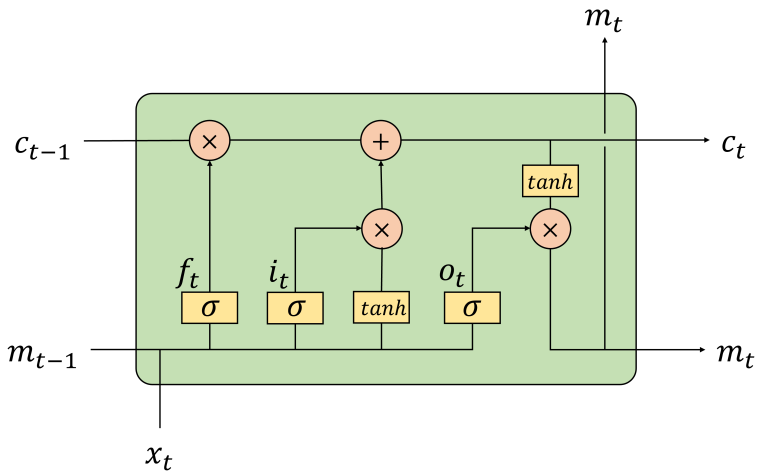

Figure 1: The architecture of the LSTM unit.

where i represents the i^{th} time step, c_i is the value of the cell gate, m_i is the value of the output, x_i is the input data, i_i is the input gate, f_i is the forget gate and o_t is the output gate. The LSTM provides two categories of activation functions: sigmoid activation function (σ) and tangent hyperbolic activation function ($tanh$). The definitions of each gate in the LSTM are demonstrated as following formulations.

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}) \tag{3}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1}) \tag{4}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1}) \tag{5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{cx}x_t + W_{cm}m_{t-1}) \tag{6}$$

$$m_t = tanh(c_t) \odot o_t \tag{7}$$

where all the $W_\bullet$ are the learnabel weights of the LSTM unit and $\odot$ is the element multiplication. Furthermore, a $softmax$ layer receives the output m_t and converts it into the probabilities of each word in the dictionary. To yield the output word, the $argmax$ method will choose the word corresponding to the maximum likelihood. Whereas, to generate a more explainable description, the NIC deploys the Beam Search algorithm with a beam of size 20. It means that, for each time step, the NIC will output 20 top-probability words as the results.

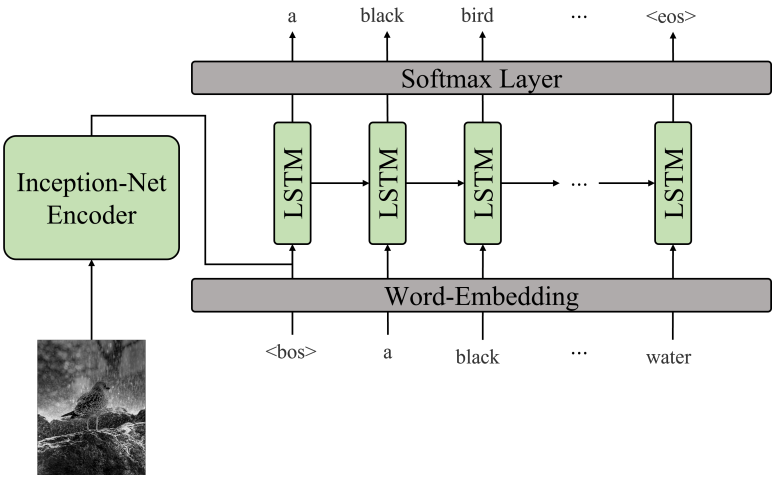

Figure 2: The architecture of the Neural Image Caption Generator.

In accordance with figure 2, the one-hot coding word representations will be fed into a word embedding layer to generate a more reasonable word vector description. In short, the main pipeline of the NIC can be rewritten as the following formulations.

$$x_v = Inception(I) \tag{8}$$

$$x_t = W_e S_t, \quad t \in 0...N-1 \tag{9}$$

$$c_t = LSTM(x_t), \quad t \in 0...N-1 \tag{10}$$

where x_v is the visual embedding of the image I, x_t is the input word embedding of the t^{th} time step, W_e is the word embedding weight, and c_t is the output of the t^{th} time step.

2.2. Neural Image Caption Generator with Visual Attention

Although the traditional NIC establishes a new state-of-the-art at that time, it still suffers from data redundancy. The original NIC pays the same attention to all regions in the image when generating each

word in the caption. With the development of the attention mechanism [2] [22] in NLP aspects, the visual attention-based NIC [34] has been issued establishing a new state-of-the-art on the Flickr8k, Flickr30k, and COCO datasets.

The main pipeline of the visual attention-based NIC is similar to the traditional one. The first difference is that visual attention-based NIC implements an attention mechanism between the CNN-based encoder and RNN-based decoder. The other difference is that visual attention-based NIC does not employ the output vector of the fully-connected layer in the CNN. In contrast, it applies the output feature maps from the deep convolutional layer to be the visual embedding. It only unfolds the visual embedding on the channel-wise to generate L visual embedding vectors $a = \{a_1, a_2...a_L\}$, where $a_i \in \mathbb{R}^D$ and $D = width * height$. After obtaining the visual embedding vector from the CNN-based encoder, the visual attention-based NIC will feed it into an attention block to compute the attention weight. This attention weight will multiply with the visual embedding vector later to yield visual context. Afterward, the visual context will be concatenated with the current input word embedding and the previous time step's hidden state to be the input of the LSTM. The mathematical definitions of the computations in the attention block are shown in the following formulations.

$$e_{ti} = f_{att}(a_i : h_{t-1}) \tag{11}$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{L} exp(e_{tk})} \tag{12}$$

$$\overset{\frown}{z_t} = \phi(a_{ti}, \alpha_{ti}) \tag{13}$$

where e_{ti} is the partial attention weight of the i^{th} visual embedding vector in the t^{th} time step, f_{att} is a linear layer to compute the partial attention weight, " : " means concatenation, a_i is the i^{th} visual embedding vector, h_{t-1} is the hidden state of the LSTM in the previous time step, α_{ti} is the attention weight of the i^{th} visual embedding vector in the t^{th} time step, $\overset{\frown}{z_t}$ is the visual context of the t^{th} time step, and ϕ is the attention function to compute the visual context.

The function $\phi(\bullet, \bullet)$ defines whether the applied attention mechanism is the "Hard" attention or the "Soft" attention. In the "Hard" attention, after obtaining the attention weights by deploying formulas 11 and 12, each visual embedding vector a_i will be assigned a one-hot vector which consists of the dictionary. If the word S_t has to pay attention to the corresponding region a_i, the related elements in the

one-hot vector of the a_i will be set to be one. In this case, each word will be forced to pay attention to fixed regions in the image. Afterward, the one-hot vector will be combined with the attention weight α to compute the visual context. Because the one-hot vector is manually designed, the "Hard" attention cannot be trained by the traditional backpropagation. Due to the "Hard" attention is not related to our Captioning and Answering Transformer, more details are in [34]. As for the "Soft" attention, it aims to let the neural model consider where to focus on in the image by itself. The $\phi(\bullet, \bullet)$ function employs the multiplication between the visual embedding vectors and attention weights. Therefore, the "Soft" attention is easy to train by applying the backpropagation. The formula of the "Soft" attention-based function ϕ is shown in 14.

$$\phi(a_{ti}, \alpha_{ti}) = \sum_{i=1}^{L} \alpha_{ti} a_{ti} \tag{14}$$

The architecture between traditional NIC [33] and visual attention-based NIC [34] is quite similar. The visual attention-based NIC consists of a CNN-based encoder and an LSTM-based decoder either. The architecture of the LSTM is the same as in figure 1. However, the x_t is made up of the visual context $\hat{z}$, the word embedding e, and the previous hidden state of the LSTM. The main architecture of the visual attention-based NIC is demonstrated in figure 3.

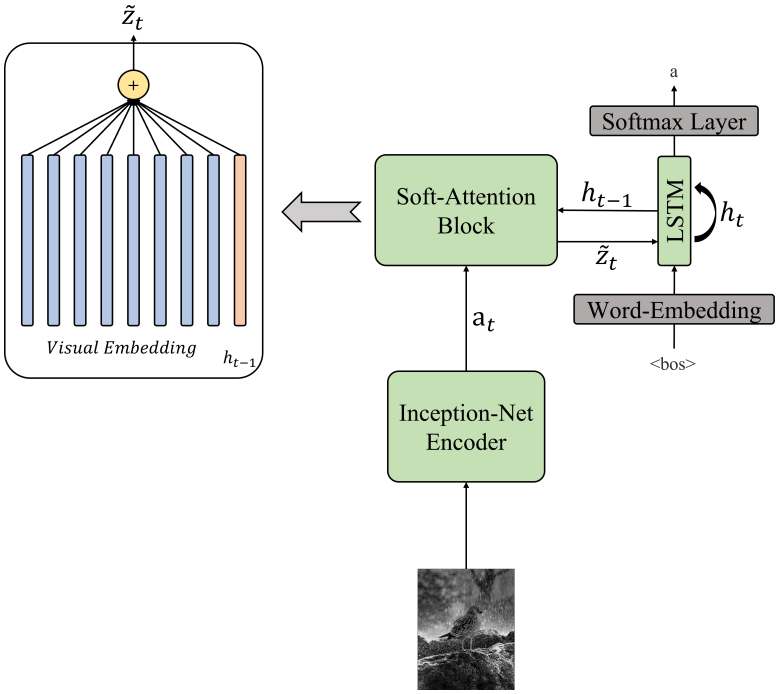

Figure 3: The architecture of the Visual Attention-based Neural Image Caption Generator.

2.3. Neural Image Caption Generator with Visual Sentinel

Whatever the ”Hard” or ”Soft” visual attention, all the visual attention-based NICs consider the visual attention for every generated word in the description. However, the prediction of some words in the description, such as ”and”, ”of”, ”the”, and so on, is not requiring visual information. In this case, the Visual Sentinel has been issued. It determines what kinds of words do not have to see the image, so-called visual embedding. This new visual attention-based NIC is called Neural Image Caption with Visual Sentinel [21]. This new NIC establishes a new state-of-the-art comparing with the Visual Attention-based NIC. In Visual Sentinel-based NIC, the encoder is the advanced CNN, and the decoder is made up of one LSTM and one Multi-Layer Perceptron (MLP). Unlike the Visual Attention-based NIC, the Visual Sentinel Attention block implements between these two decoder layers. Meanwhile, the visual sentinel is employed to figure out whether the MLP has to predict the current word according to the visual context or not. The main pipeline of this visual sentinel-based model is a sequence to sequence model either. The illustration of the Visual Sentinel-based NIC is shown in figure 4.

The Visual Sentinel Attention block consists of traditional Visual Attention in figure 3 and the Visual Sentinel. The visual attention part is ”Soft” attention which is approximately the same as formulas 11, 12, and 13. The only difference is the method to obtain partial attention e_{ti}. The modified formula is shown in equation 15.

$$e_{ti} = W_h^T tanh(Waa_{ti} + W_g h_t) \tag{15}$$

where $W_\bullet$ are the learnable weights. Due to the Vision Sentinel Attention block being deployed between one LSTM layer and one MLP layer. The applied hidden state in visual attention is the current hidden state from the LSTM layer. During generating the hidden state involved in the computation of the visual attention, the LSTM layer will yield out the Visual Sentinel either. A new gate called the sentinel gate receives the word embedding and the previous hidden state to generate the sentinel in the LSTM layer. The mathematical definitions are shown in the following formulations.

$$g_t = \sigma(W_x x_t + W_h h_{t-1}) \tag{16}$$

$$s_t = g_t \odot tanh(m_t) \tag{17}$$

where g_t is the sentinel gate in the t^{th} time step, $W_\bullet$ is the learnable weights, x_t is the input word embedding in t^{th} time step, h_{t-1} is the hidden state in previous time step, m_t is the output in the t^{th} time step, s_t is the visual sentinel in the t^{th} time step, and $\odot$ is the element multiplication. To employ the visual sentinel s_t, it will participate in the computation of the attention weights α. The Visual Sentinel Attention block will concatenate the visual embedding vectors a, the hidden state of the LSTM h_t, and the visual sentinel s_t to compute the attention weight. Afterward, the whole attention weight will be separated into α_i where $i \in 1...L$ and β_t. All the α_i will be used to multiply with the visual embedding vectors to produce the visual context $\overset{\frown}{z_t}$. Meanwhile, the β_t is employed to indicate whether feed the visual context into the MLP layer, according to the formula 18.

$$z_t = \beta_t s_t + (1 - \beta_t)\, \overset{\frown}{z_t} \tag{18}$$

where z_t is the final context which is fed into the MLP layer.

As for the Visual Sentinel Attention-based NIC, the input word embedding x_t has two parts. One is the standard word embedding in the t^{th} time step. Another one is the global visual embedding a^g which is the sum of all the visual embedding vectors. Therefore, the sentinel gate can extract the relationships between the visual and linguistic information to check whether the visual context is necessary for predicting the current word. Afterward, the sentinel weight β_t will further aggregate all this information to control the proportion of the visual context in the final context.

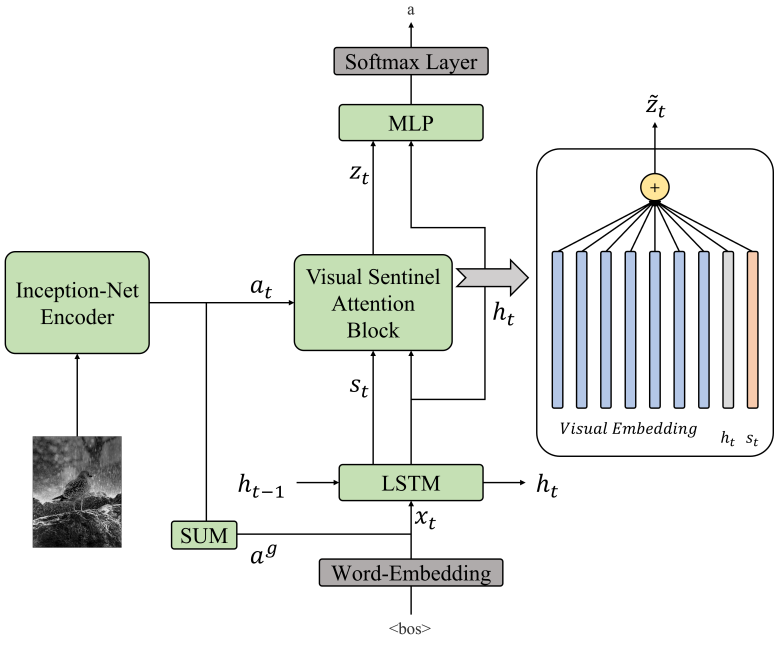

Figure 4: The architecture of the Visual Sentinel Attention-based Neural Image Caption Generator.

2.4. Neural Image Caption Generator with Bottom-Up and Top-Down Attention

If the visual sentinel extends the ”Soft” visual attention, then the Bottom-Up Visual Attention will extend the ”Hard” visual attention. These architectures are called Bottom-Up and Top-Down Attention-based Neural Image Caption Generator [1]. The essence of the ”Hard” visual attention is to assign the description's words to the image's fixed regions. However, [34] clarifies that the traditional ”Hard” visual attention is hard to train with the backpropagation. To solve this drawback, the Bottom-Up attention publishes a novel view to implementing the ”Hard” attntion, directly obtaining the fixed regions that have to be paid attention to in the image by using the object detection algorithm. In this case, both the ”Hard” and ”Soft” visual attentions can be implemented by neural network architectures. Therefore, the whole model can be trained by backpropagation. In [1], the Faster-RCNN [27] has been applied. Whereas all the object detection neural networks can be transplanted to achieve the Bottom-Up visual attention, such as You Only Look Once Series (YOLO) [24] [25] [26] [3].

2.4.1. Faster-RCNN-based Bottom-Up Visual Attention

The significant theory of the Bottom-Up Visual Attention is to obtain the fixed regions that need to be paid attention to in the image. The Faster RCNN is the most used module in the Bottom-Up Visual Attention pipeline. Faster Region Convolutional Neural Network (Faster RCNN) [27] is a typical two-stages object detection architecture. The first stage aims to find out all the Regoins-of-Interests (RoI) that may contain the objects. The second stage fine-tunes all the RoIs to yield all the accurate locations of the objects.

Unlike the traditional Region Convolutional Neural Network (RCNN) [10], the Faster RCNN is a full-convolutional neural network. It employs the Region Proposal Network (RPN) in Fast RCNN [9] to generate the RoIs in the first stage. RPN receives the feature maps from the CNN-based backbone network and applies them into two separated data flows to generate the RoIs, according to 9 manually desigend Anchor Boxes. One is called label regression that generates the confidence score and classification accuracy for each RoIs. Another one is named bounding box regression that fine-tunes all the RoIs. To indicate the positive and negative RoIs instances, the RPN employs the Intersection over Union (IoU) between the predicted RoIs and Ground Truth to be the criterion. The target of the RPN is to yield the RoIs rather than the accurate locations of the objects. Therefore, as long as the IoU is larger than the threshold and the confidence score is above 50, the RoI will be treated as a pos-

itive sample, vice versa. The non-maximum suppression (NMS) mechanism is used to fine-tune the RoIs either. Eventually, the RPN will offer 300 high-quality RoIs into the second stage of the Faster RCNN.

In the second stage, the 300 high-quality RoIs will crop the corresponding regions from the feature maps. Meanwhile, Faster RCNN applies RoI Align rather than the RoI pooling in the Fast RCNN [9] to convert different scales and sizes RoIs into the standard ones. Due to the feature maps being extracted from the raw image, which means there is a scales transformation between them, and the coordinates of the RoIs are not the integers sometimes. Whereas the RoI pooling ceils or floors all the float numbers into the most approximate integers. It means that some errors occur after applying the RoI pooling in the Fast RCNN [9] to convert the RoIs into standard ones.

Meanwhile, these errors will be enlarged when mapping the RoIs from the feature maps to raw images. However, the RoI Align remedy this drawback. It maintains all the float numbers and uses bilinear interpolation to get the pixel value at the position of the corresponding float number. After crop all the RoIs, the RoI Align will split the RoIs into some standard windows and apply the max pooling for each window to produce standard-sized RoIs. Atferward, the standard-sized RoIs will be fine-tuned to generate the accurate locations of the objects in the raw images.

All of these locations are the targets of Bottom-Up visual attention. In [1], the Bottom-Up visual attention deploys the Faster-RCNN with the Residual Network-based backbone [11] to generate all the attention regions. All these regions contain both background and front-ground in the raw images. All of them will be unfolded as the visual embedding vectors a_i where $i \in 1...L$, and fed into the LSTM-based decoder.

2.4.2. LSTM-based Top-Down Visual Attention

The essence of the Top-Down Visual Attention is the ”Soft” Visual Attention. In [1], the decoder consists of one-directional and two-layers LSTM and applies the ”Soft” attention in [34] between two LSTM layers. The first LSTM layer is called Top-Down Attention LSTM, which receives the previous hidden state of the second LSTM layer, the visual embedding vectors, and word embedding as the input to generate the input of the visual attention blocks. Another LSTM layer is named Language LSTM, which yields the prediction of the word in the description. The mathematical definitions of visual attention are shown in formulations 11, 12, and 13. The entire architecture of the Bottom-Up and

Top-Down Visual Attention-based NIC is illustrated in figure 5.

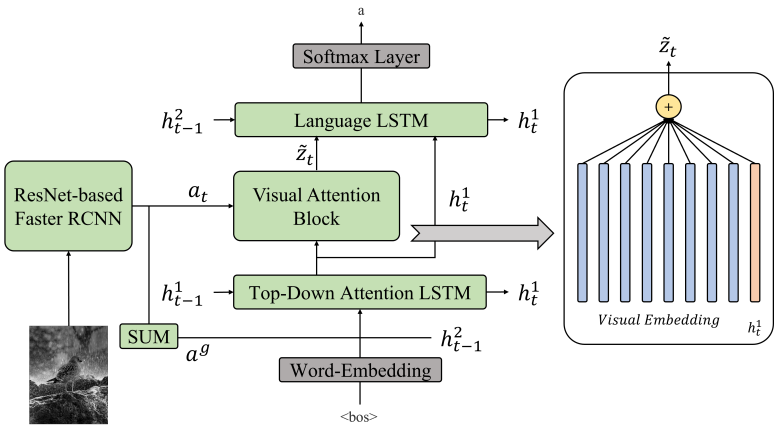

Figure 5: The architecture of the Bottom-Up and Top-Down Visual Attention-based Neural Image Caption Generator.

2.5. Neural Image Caption Generator with Transformer

With the development of the attention mechanism, Google has published a fantastic self-attention-based novel sequence to sequence architecture named Transformer [32]. The essence of almost all the attention mechanisms is that apply a query to look through the key of the values and compute the attention weights for values between the query and key. The self-attention highlights this process by directly employing three linear layers to compute the query, key, and value from the input data and compute the attention between all the time steps parallelly, according to formulation 19.

$$Self - Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{19}$$

where Q, K, and V are the query, key, and value that calculate from the raw sequential data. Their dimensions are $l \times d$, where l is the number of the time step and d is the dimension of each data in each time step. By applying the self-attention mechanism to replace the traditional LSTM, each time step of the sequential data can be computed parallelly, because self-attention is only some dot-production.

The transformer employs multi-head self-attention to enrich the attention representations which combines plenty of different self-attention heads. Each head contains one independent self-attention based on formula 19. The transformer absorbs the excellent residual connection idea in the ResNet [11]. After each multi-head self-attention block, there is a residual connection and a layer normalization. At the end of each Transformer block, the output of the multi-head self-attention will be fed into

a feed-forward network (FFN), and there is a residual connection and layer normalization after the FFN either. The architecture of the Transformer is demonstrated in figure 6.

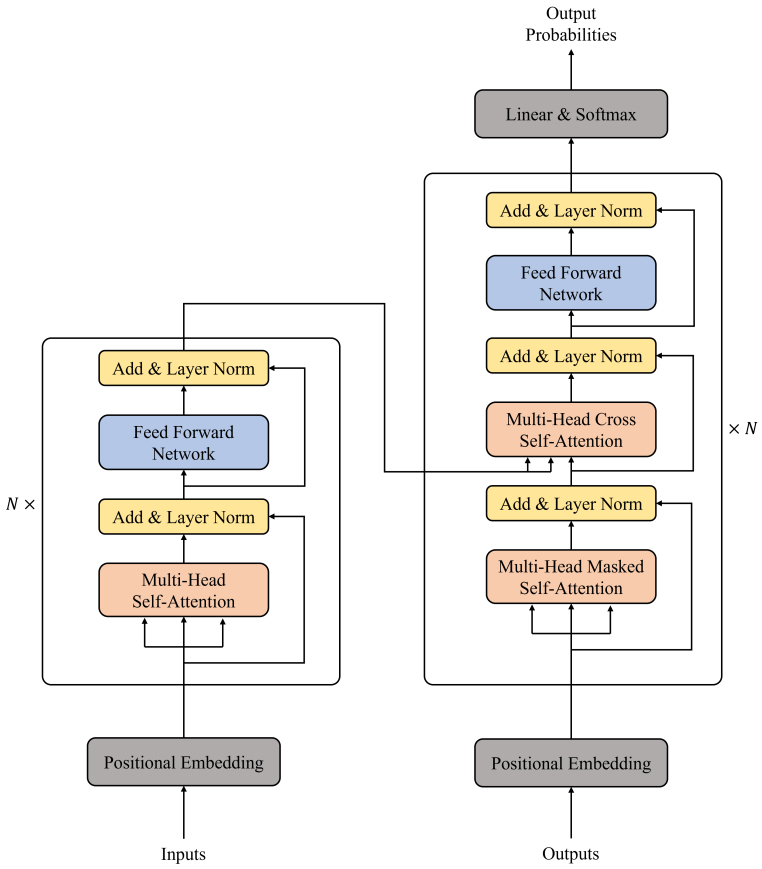

Figure 6: The architecture of the Transformer, , where N is the number of the transformer blocks.

There are two remaining parts of the transformer which are positional embedding and multi-head masked self-attention. Due to the self-attention is only some dot-productions, it means that even the order of each time step in the input data is not correct, the self-attention can process the data either. Therefore, the cosine and sine-based positional embedding has been applied to indicate the order in the raw data. The formulas 20 and 21 show the definitions of positional embedding.

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{20}$$

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{21}$$

where PE is the positional embedding, pos is the position index of each time step in the data, i is the dimension index of each time step's data, and d_{model} is the dimension of each time step's data. The illustration of the positional embeeding is shown in figure 7.

As for the multi-head masked self-attention, it only is employed in the decoder of the transformer.

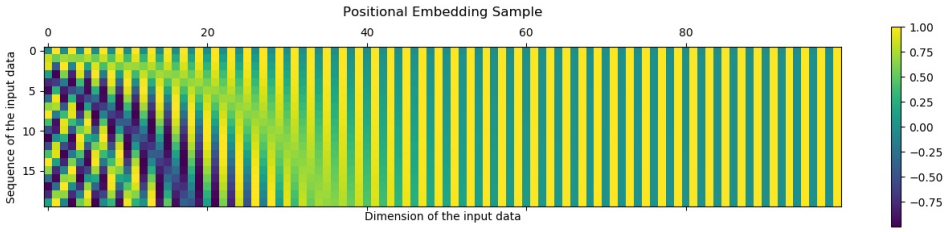

Figure 7: The illustration of the positional embedding sampel.

Because it is unreasonable to use the $i + 1^{th}$ time steps information to predict the i^{th} time steps information. Therefore, it is necessary to avoid the attention of the future time step's data when predicting the previous time step's data. In [32], they designed an attention mask, like figure 8, where the values in the purple region are zeros and the ones in all yellow areas.

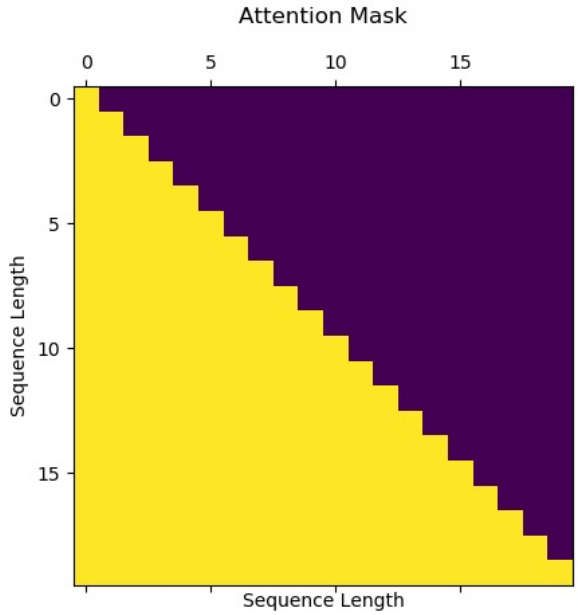

Figure 8: The illustration of the attention mask.

When the transformer becomes popular in the Natural Language Processing region, researchers are interested in transplanting it into the Computer Vision aspect. There are plenty of efficient transformer-based vision architectures, such as the Detection Transformer (DETR) [4] for object detection tasks, Vision Transformer (ViT) for image classification tasks. In terms of the Image Caption task, the Caption Transformer (CPTR) firstly employs the full-transformer-based sequence to sequence architecture. The main pipeline of the CPTR is similar to the traditional transformer, which is illustrated in figure 6. However, the CPTR does not apply the sine and cosine-based positional embedding. It deploys the learnable positional embedding like the DETR. Except for positional embedding, almost

all the visual transformers like DETR, ViT, and CPTR apply the patch embedding to preprocess the input raw image. This progress will separate the raw image into plenty of fixed-sized windows. Each window will be unfolded into a one-dimensional vector to simulate each time step's data.

In this case, all the vision transformers with the traditional transformer's architecture (Figure 6) will face two main problems. One is the considerable time complexity of the dot-production in the self-attention. Because the image resolution determines the dimension of each unfolded patch vector, if the resolution is too huge, then the dimension of the unfolded patch is enormous either. Afterward, the dot-production between each patch requires giant time complexity. Another one is that the traditional transformer is poor to extracting the local features from the image because the traditional transformer only can compute the attention between each patch. As for the features inside the patch, the traditional transformer is hard to extract.

Inspired by convolution, the Swin Transformer [20] tries only to compute self-attention inside fixed-sized windows. Meanwhile, it employs the shifted-window self-attention to exchange the locations of each window by rolling up the windows in the bottom to the top and rolling right the windows in the left to the right to extract the features between two adjacent windows. Via these two novel techniques, the Swin Transformer reduces the time complexity of the transformer-based architecture in the Computer Vision region. It remedies the drawback of poor local features extraction capability at the same time. However, currently, a new visual attention technique has been issued called outlook-attention [38]. It is pretty fantastic in handling the ImageNet-1k classification tasks. There are more details about it in our proposed theory.

3. Theory

Our proposed thought of the Captioning and Answering Transformer (CATR) can be seen as a full-transformer sequence to sequence architecture. The encoder consists of the outlook-attention-based Vision Outlooker [38] that generates the visual embedding vectors. The decoder consists of two stages. One stage is the word2vec to yield the word embedding. Another stage is the traditional transformer-based decoder to predict the description of the image. The entire pipeline is demonstrated in the figure 9. The input of the CATR is a raw image and the output is the description of the input image.

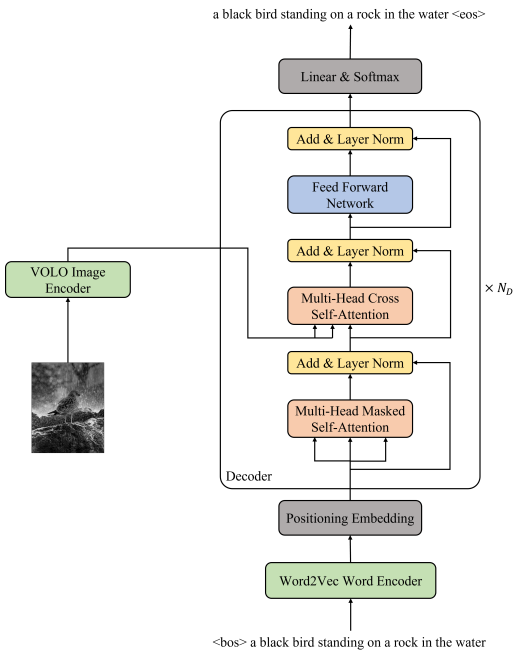

Figure 9: The architecture of the Captioning and Answering Transformer, where N_D is the number of the decoder blocks.

3.1. Vision Outlooker based Encoder

Vision Outlooker is an excellent attention-based architecture that establishes 87.1% top-1 accuracy on the ImageNet-1k classification tasks for the first time. It consists of two stages. One is the outlook-attention layer followed by a Feed Forward-Network (FFN). Another one is the self-attention-based traditional transformer. Inspired by convolution, the outlook-attention realizes that the center pixel in the 3×3 receptive field has enough information to compute the attention weights for all the adjacent pixels.

Meanwhile, to prevent the huge time complexity of the dot-production in the self-attention, outlook-attention employs two linear layers to yield the attention and value directly. Afterward, the value will unfold like the convolutional layers to obtain the 3×3 receptive regions for each pixel. The mathematical definition is shown in the following formulations.

$$\alpha = Softmax(AttentionLayer(P)) \tag{22}$$

$$value = Unfold(ValueLayer(P)) \tag{23}$$

$$output = Fold(\alpha * value) \tag{24}$$

where α is the outlook-attention weights, $value$ is the outlook-attention value, $output$ is the outlook-

attention output, and P is the patch embedding of the input image. Assume the dimension of the patch embedding is (W, H, C) where W is the width, H is height, and C is the number of the channel, after $ValueLayer$ and $Unfold$, the dimension will become $(W * H, K * K, C)$ where K is the window size which equals 3 in our model. After $AttentionLayer$, the dimension of the α will be $(W * H, K * K, K * K)$. In this case, the dimensions between the output of $AttentionLayer$ and $ValueLayer$ are matched to complete the dot-production. After folding, the dimension of the $output$ will be the same as the input.

Similar to the traditional transformer, the Vision Outlooker (VOLO) consists of plenty of outlook-attention heads. Each head contains an independent outlook-attention. The mathematical definition of the outlooker block is shown in formulas 25 and 26.

$$\hat{x} = Outlooker(LN(x)) + x \tag{25}$$

$$Z = MLP(LN(\hat{x})) + \hat{x} \tag{26}$$

where z is the output of the outlooker block, $\hat{x}$ is the output of the multi-head outlook-attention, x is the input of the outlooker block, LN is the leyar normalization, and MLP is the Feed Forward Layer. The architecture of the Vision Outlooker is demonstrated in the figure 10. The same as the DETR, the positional embedding of the vision outlooker is trainable.

The target of the outlook-attention is to remedy the drawback of the self-attention in extracting local features. In order to obtain both the finer and coarser features, both multi-head outlook attention and multi-head self-attention have been combined to form the entire architecture of the Vision Outlooker.

3.2. Transformer based Decoder

In accordance with figure 9, our decoder has two main stages. One is the typical decoder of the transformer that contains one multi-head self-attention and one multi-head cross self-attention. Because the decoder aims to predict the sentence of the Image Caption, similar to the machine translation, it is significant that the future linguistic information can not be applied to predict the previous linguistic information. The attention mask, like figure 8, employs in our transformer-based decoder either. Meanwhile, similar to machine translation, our task is to analyze the input image and yield the cap-

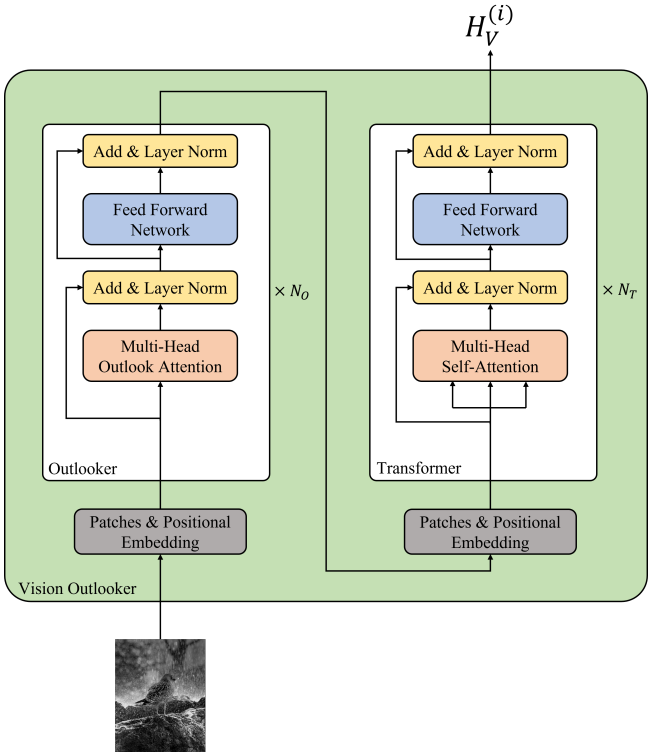

Figure 10: The architecture of the Vision Outlooker, where N_O and N_T are the number of the Outlooker and Transformer blocks.

tioning or answering, a translation mission either. It is necessary to translate visual information into linguistic information. Therefore, the architecture to permit the information exchange for both visual context and linguistic context is needed. As for the past five years of research, visual attention [34] and the visual sentinel [21] have been published to implement this critical part. In our CATR, like CPTR, we applied multi-head cross self-attention to tackle this mission. It will receive the visual embedding vectors as the key and value, and linguistic embedding as the query to compute the self-attention, to automatically learn the relationships between the visual and linguistic context.

The visual encoder in figure 10 employs the Vision Outlooker [38] to extract the finer-level features from the image, so-called local information. It deploys a transformer to obtain the coarser-level information from the image, so-called global context. Similar to the image, the sentence contains local and global information either. The global information is the linguistic context that can be analyzed by transformer very well. The local features of the sentence can be seen as the relationships between each word. For example, 'apple', 'banana', 'watermelon' are fruits, 'cat' and 'dog' are pets. The progress in extracting the relationships between word and word is called word embedding. In our CATR, the traditional word to vector (word2vec) embedding training pipeline has been applied.

4. Implementation and Experimental Result

As for our project, in the first stage, we aimed to figure out the Road Surface Marking Recognition task, to check whether the transformer-based architecture is useful in processing images. It can be seen as an object detection mission. During the experiments, we realized that the transformer has an excellent capability to complete the Computer Vision tasks. Therefore, to dig into the performance of the transformer-based architecture in both Computer Vision and Natural Language Processing regions, we proposed the second stage, which are Image Captioning and Vision Questions Answering. Besides, before we go through the next stage, we completed some object detection experiments by using the You Only Look Once Serises (YOLO) [24] [25] [26] [3], Region Convolutional Neural Network Series (RCNN) [10] [9] [27], Detection Transformer (DETR) [4], and Single Shot MultiBox Detector (SSD) [18] on our own Road Surface dataset. The evaluation results are shown in figures 11, 12, 13, and 14.

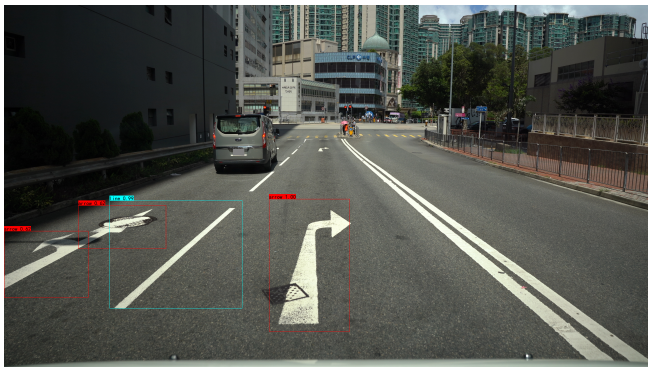

Figure 11: The experimental results of the Faster RCNN.

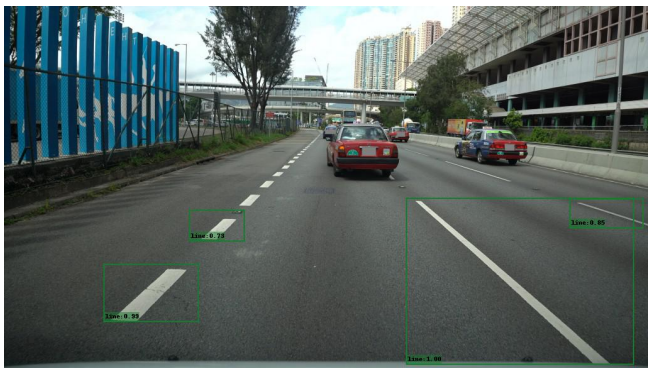

Figure 12: The experimental results of the SSD.

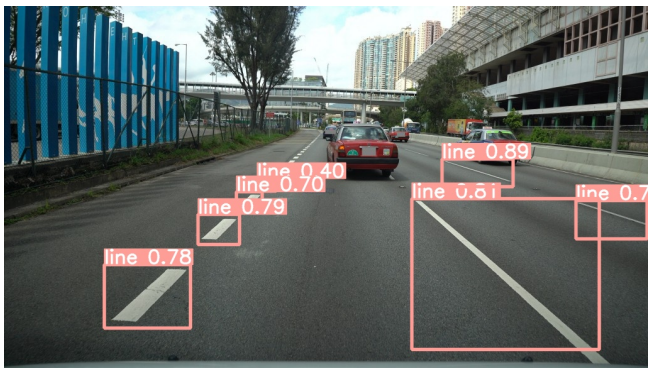

Figure 13: The experimental results of the YOLO version five.

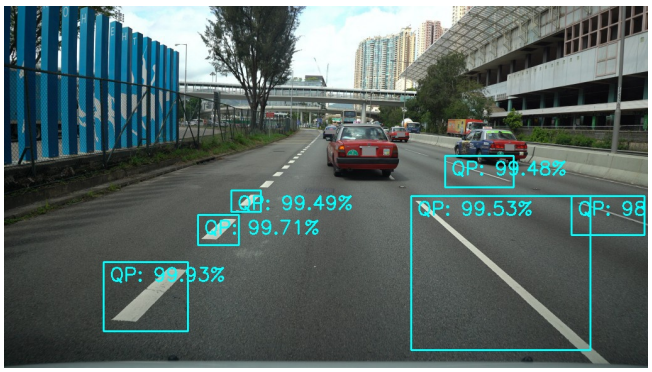

Figure 14: The experimental results of the DETR.

The Detection Transformer obtained an excellent performance with a high confidence score and high-quality bounding boxes in the experiments of the first stage. This is why we thought the transformer is more powerful to handle some complicated tasks and move to the second stage. More experimental results of stage one are shown in Appendix A.

4.1. Training Pipeline

In terms of the stage two, we implement five types of Captioning and Answering with TransformeR (CATR) architecture with different sizes, according to the different Vision-Outlookers. Table 1 demonstrates all the parameters' settings of all these five types of CPTR.

We evaluate our proposed architecture on Microsoft COCO 2014 dataset (MSCOCO) [17] which is the most popular benchmark for Image Captioning. In the MSCOCO 2014 Dataset, there are 82783, 40504 and 40775 images for training, validation and test datasets. As for each image, there are 5 or more label captions. However, we only pick up the first 5 captions of each image to train the architecture. The validation dataset will be used during the training to check the overfitting situation and offline evaluations.

Modules	Specification	Type-1	Type-2	Type-3	Type-4	Type-5
Encoder	Patch Embedding	8×8	8×8	8×8	8×8	8×8
	Outlook-Attention (28×28)	head: 6, stride: 2 kernel: 3×3 mlp: 3, dim: 192 $\times 4$	head: 8, stride: 2 kernel: 3×3 mlp: 3, dim: 256 $\times 6$	head: 8, stride: 2 kernel: 3×3 mlp: 3, dim: 256 $\times 8$	head: 12, stride: 2 kernel: 3×3 mlp: 3, dim: 384 $\times 8$	head: 12, stride: 2 kernel: 3×3 mlp: 4, dim: 384 $\times 12$
	Patch Embedding	2×2	2×2	2×2	2×2	2×2
	Self-Attention (14×14)	#heads: 12 mlp: 3, dim: 384 $\times 14$	#heads: 16 mlp: 3, dim: 512 $\times 18$	#heads: 16 mlp: 3, dim: 384 $\times 28$	#heads: 16 mlp: 3, dim: 768 $\times 28$	#heads: 16 mlp: 4, dim: 768 $\times 36$
Decoder	Word Embedding (word2vec)					
	Self-Attention	#heads: 8 mlp: 3, dim: 384 $\times 8$	#heads: 8 mlp: 3, dim: 512 $\times 8$	#heads: 12 mlp: 3, dim: 512 $\times 12$	#heads: 12 mlp: 3, dim: 768 $\times 12$	#heads: 16 mlp: 4, dim: 768 $\times 16$
Total Layers		26	32	48	48	64

Table 1: The architecture information of all five types of Captioning and Answering with TransfromeR (CATR).

We train the CATR in an end-to-end fashion, whereas, only the parameters of the transformer-based decoder will be updated, due to the device limitation. The outlooker-based encoder will be intialized by the corresponding VOLO [38] architecture which pre-trained from the ImageNet-1k dataset, without the classes-prediction layer. The input images of the CATR are resized to 224×224 which is the standard input resolution of the VOLO [38].

Due to the transformer is a quite large architecture, it is necessary to train it with a small learning rate at the very begining, to avoid the gradient fall into the local optimum too fast. Hence, we applied the Noam learning rate scheduler in [32]. The mathematical definition of the Noam learning rate scheduler is shown in equation 27.

$$lr = dim^{-0.5} * min(iteration^{-0.5}, iteration * warmUp^{-1.5}) \tag{27}$$

where lr is the learning rate for the next iteration, $iteration$ is the current iteration's number, dim is the hidden dimension of the decoder, and $warmUp$ is the learning rate increasing period of the Noam scheduler. After employed the Noam learning scheduler, the learning rate will be increased from zero utill the iteration reaches the warm up setting, according to the equation 27. As long as the iterations beyond the warm up setting, the learning rate will be decreased respectively.

During our training pipeline, we employ the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$, $momentum = 0.9$, and $weight\ decay = 0.00005$, and the cross-entropy loss with label smoothing set to be 0.1. Meanwhile, we set the $batch\ size = 32$ and train the whole architecture in just one RTX 3090 GPU. Due to the device limitation, we only are able to complete the training of the Type-1 CATR. As for each epoch, the training takes approximate 30 minutes, and the overall number of epochs are

100. Hence, the total training for one demo will cost about 2 days. Because the $batch\ size = 32$, each epochs will contain 2587 iterations. Thus, the warm up of the Noam scheduler sets to be 3000. The visible learning rate variation is shown in figure 15.

To overcome the overfitting problems, we not only apply the label smoothing and weight decay methods, but also employ two kinds of dropout blocks which are the attention dropout and projection dropout. The attention dropout is applied after the computation of attention score. The projection dropout is employed after each Feed Forward Layer. Both of the dropout rate are set to be 0.1.

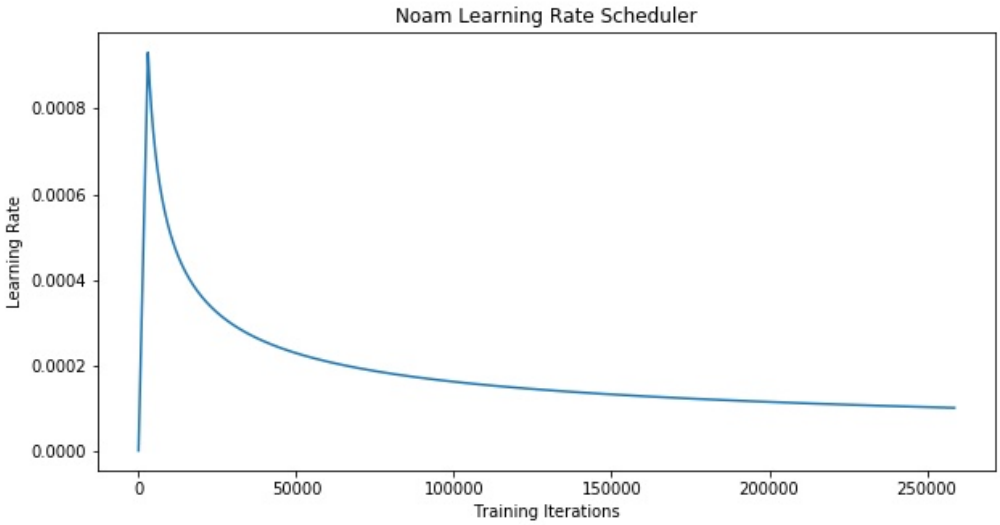

Figure 15: The visible learning rate variation.

In terms of the evaluation, the trained CATR will be split into Outlooker-Encoder and Transformer-Decoder. The encoder will only receive the input image just once to extract the vision features. The same vision features will be applied into Transformer-Decoder every time, till the decoder predicting the end token. To generate the captions, we apply the beam search with $beam\ size = 3$. Due to the limitation of the device, we only can evaluate the images in the validation set and test set one by one. Thus the evaluation pipeline will cost about 1.5 days for each dataset. The overall time consumption of the evaluation is about 3 days.

4.2. Experiments Analysis

Due to we only are able to train the architecture with just one RTX 3090 GPU, we can only complete the training of the Type-1 CATR. We train totally 10 Type-1 CATRs, and pick up the best one to finish the evaluation. After training, we apply the best architecture to check the overfitting situation

by the validation dataset. Both of the training and validation results are shown in table 2. During the training and validation, the loss, accuracy, and 4 Bi-Lingual Evaluation Understudy scores (BLEUs), which are uni-gram, bi-gram, tri-gram and fourth-gram, are applied to be the metrics to check the performance of the models.

Model	Metrics	Train	Validation
Type-1 CATR	Loss	2.5176	2.6181
	Accuracy	76.90	75.08
	BLEU-1	81.63	80.29
	BLEU-2	71.20	69.10
	BLEU-3	62.00	59.40
	BLEU-4	54.68	51.74

Table 2: The training results of the Type-1 CATR.

According to the training results, almost all the differences between train and validation results in each metrics are smaller than 3 percent. This difference criterion is acceptable. It means that, the demo of trained Type-1 CATR dose not occur the overfitting problems. However, during the training, the decoder will receive the whole captions to be the input rather than the start token. Whereas, when creating the demo of generating the caption of one image, the whole sentence is unknowable. We can only input the image and the start token, to let the start token randomly sampled the next word according to the input image, till predicting the end token, the caption will be generated. Therefore, the training and validation results are inefficient to check the performance of the trained architecture. It is necessary to retest the model.

During the test, we set the trained arhcitecture into evaluation mode and apply the validation set to complete the offline performance checking, due to the MSCOCO dataset only provides the annotations of the training and validation dataset. The test dataset can only be used for online performance checking. The evaluation metrics of the test are the 4 BLEU scores, METEOR, ROUGE and CIDEr scores [5]. The test results of the CATR and the previous Neural Image Captioning generator are demonstrated in table 3. We compared the trained Type-1 CATR with the CPTR [19], LSTM [33], SCST [28], LSTM-A [37], RFNet [15], Up-Down [1], GCN-LSTM [36], LBPF [23], SGAE [35], ORT [12], and ETA [16].

It is obvious that our Type-1 CATR can beat some previous CNN-and-RNN-based Neural Image Caption generator, such as LSTM, SCST and LSTM-A. It means our Type-1 CATR has learnt some knowledges of the image captioning generation. However, comparing with all the other Neural Image

Caption generator, especially the CPTR, our Type-1 CATR still exist some performance gaps. Comparing with the CPTR, the BLEU-4 performance gaps of our Type-1 CATR is almost 4 percent. The gap for CIDEr even reaches almost 9 percent. Hence, in this stage, the current SOTA full-transfomer-based architecture still outperforms our Type-1 CATR. Figure 16 demonstrate some demos of our Type-1 CATR. More Image Captioning demos are shown in Appendix B.

Architecture	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE	CIDEr
CNN+RNN							
LSTM [33]	-	-	-	31.9	25.5	54.3	106.3
SCST [28]	-	-	-	34.2	26.7	55.7	114.0
LSTM-A [37]	78.6	-	-	35.5	27.3	56.8	118.3
RFNet [15]	79.1	63.1	48.4	36.5	27.7	57.3	121.9
Up-Down [1]	79.8	-	-	36.3	27.7	56.9	120.1
GCN-LSTM [36]	80.5	-	-	38.2	28.5	58.3	127.6
LBPF [23]	80.5	-	-	38.3	28.5	58.4	127.6
SGAE [35]	80.0	-	-	38.4	28.4	58.6	127.8
CNN+Transformer							
ORT [12]	80.5	-	-	38.6	28.7	58.4	128.3
ETA [16]	81.5	-	-	39.3	28.8	58.9	126.6
Full Transformer							
CPTR [19]	**81.7**	**66.6**	**52.2**	**40.0**	**29.1**	**59.4**	**129.4**
Our Type-1 CATR	78.9	59.3	47.6	35.9	27.5	56.2	120.3

Table 3: Performance comparisons on MSCOCO of Type-1 CATR and on COCO Karpathy test split [14] of other models.

5. Discussions

During our reasearching, it is obvious that the transformer has a great potential in both Computer Vision and Natural Language Processing aspects. Even though the smallest architecture in our CATR architecture can still get some performance. The current SOTA CPTR displays a better evidence for this either. According to the training results, although our proposed architecture still obtains some performance gaps, compared with current SOTA results, we still think it is possibility to figure out this gap in the future.

5.1. Problems

Through the results in table 3, our Type-1 CATR architecture is not performing very well, compared with the other Neural Image Captioning generators. There are four reasons may caused this situation: 1) In this stage, our trained architecture is the Type-1 CATR. It means the Vision-Encoder

employs the Type-1 Vision-Outlooker to be the vision features' extractor. The top-1 accuracy for the ImageNet classification task of the Type-1 Vision-Outlooker is only 84.2 percent, which is even lower than the accuracy of LV-ViT-S [13] (84.4 percent). However, the current SOTA CPTR applies the ViT to initialize the Vision-Encoder. Therefore, comparing with the vision features' extraction capability between our Type-1 CATR and CPTR, our model is poorer. 2) The current SOTA CPTR commits the upsampling of the input image before training, which means they will crop the raw image in the MSCOCO dataset into a higher resolution (384×384) than the standard resolution (224×224). The higher resolution means that the Vision-Encoder of the CPTR can see finer features than ours, such as some tiny objects. Whereas, due to the device limitation, we are not able to train the architecture with the higher resolution. 3) Even though both of our Type-1 CATR and current SOTA CPTR apply the MSCOCO 2014 dataset to do the training, the CPTR employs a more efficient split of MSCOCO dataset for Image Captioning task called 'Karpathy Splits' [14]. This spilt contains a larger training dataset than the standard training dataset. However, as for some large architecture like transformer, the large training dataset can fine-tune a more stable architecture. 4) After initializing the Vision-Encoder by ViT, the CPTR will still fine-tune the parameters of the Vision-Encoder. Whereas, due to the limitation of the device, if we fine-tune the Outlooker-based Encoder either, the time consumption is intolerable.

5.2. Advantages

Even though there are still some performance gaps of our Type-1 CATR, we still think our thought of the CATR is still correct. Because all of above problems can be solved by training the architecture with a more powerful GPU-Matrix. Due to the Type-1 CATR has already obtained some performace, we still have expectation for the Type-5 CATR. Because of the powerful capabilities of the attention-based architecture, so-called transformer, and the fantastic experimental results of the DETR on the Road Surface Marking Detection project, we are confident that our proposed Captioning and Answering Transformer (CATR) will perform greate either. The reason is shown below: 1) Our proposed CATR deploys the Vision Outlooker (VOLO) to replace the traditional transformer-based Vision Encoder in the Caption Transformer (CPTR) [19] to remedy the drawback of the typical transformer in extracting the local visual features. Meanwhile, VOLO has been proved that be excellent for the ImageNet-1k classification tasks. 2) Our proposed CATR modifies the current state-of-the-art archi-

tecture of Caption Transformer(CPTR) by applying advanced transformer-based models.

6. Conclusions

In this final thesis, we proposed a full-transformer-based architecture to tackle the Image Captioning and Vision Questions Answering tasks, called Captioning and Answering Transformer (CATR). Based on the current state-of-the-art full-transformer-based architecture called Caption Transformer (CPTR). We consider employing some advanced transformer-based architectures in Computer Vision regions, which is Vision-Outlooker (VOLO) to modify the CPTR and remedy its drawbacks. Due to the current device constraints, we have not outperformed the current SOTA results. However, via the analysis of the CATR and CPTR, we think our researching still reveals that the CATR has a great potential in the Image Caption task.

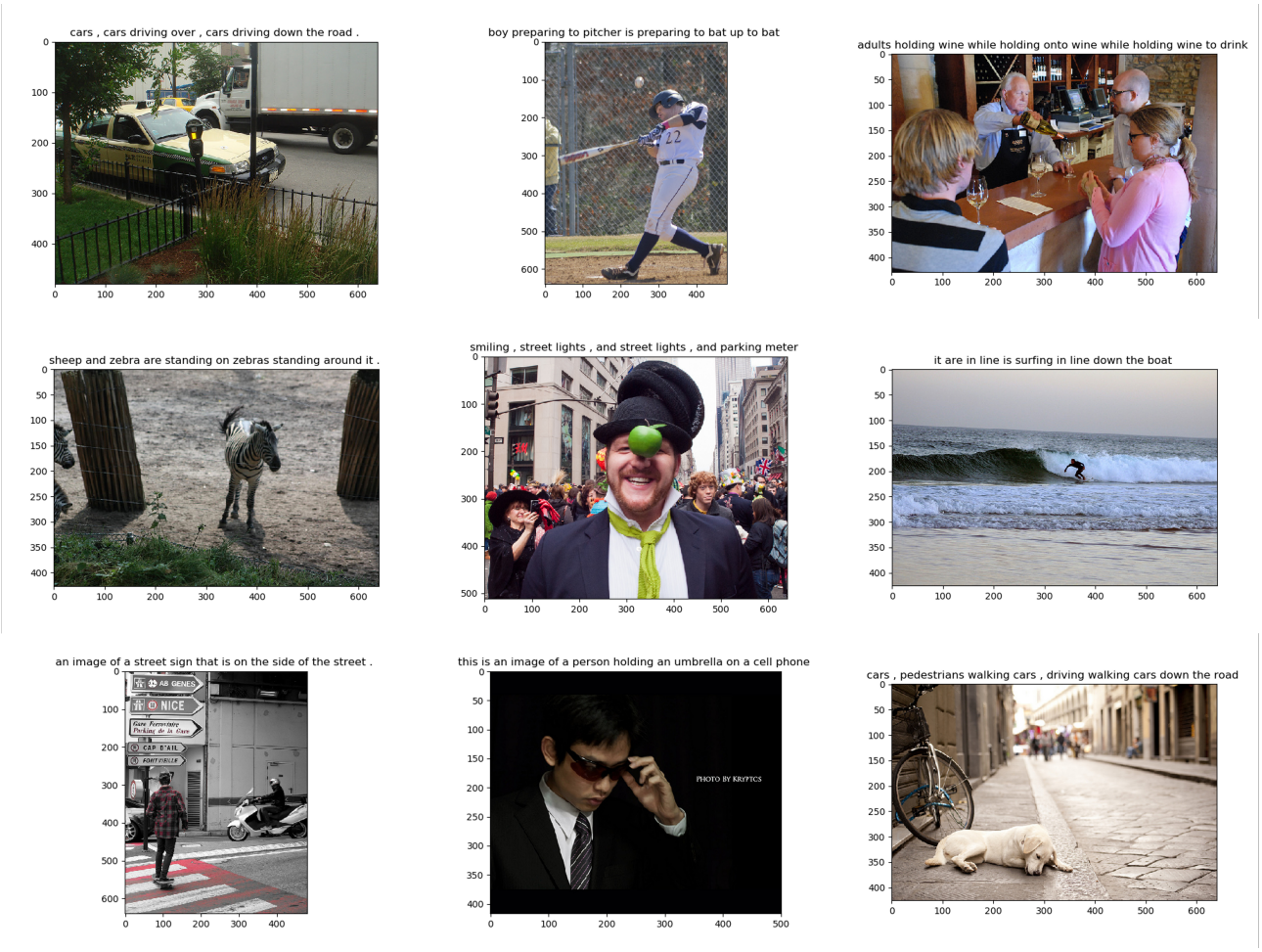

Figure 16: Some demos of our Type-1 CATR.

References

[1] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[5] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[6] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[7] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[8] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[9] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] S. Herdade, A. Kappeler, K. Boakye, and J. Soares. Image captioning: Transforming objects into words. *Advances in Neural Information Processing Systems*, 32, 2019.

[13] Z.-H. Jiang, Q. Hou, L. Yuan, D. Zhou, Y. Shi, X. Jin, A. Wang, and J. Feng. All tokens matter: Token labeling for training better vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[14] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[15] L. Ke, W. Pei, R. Li, X. Shen, and Y.-W. Tai. Reflective decoding network for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8888–8897, 2019.

[16] G. Li, L. Zhu, P. Liu, and Y. Yang. Entangled transformer for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8928–8937, 2019.

[17] 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. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[19] W. Liu, S. Chen, L. Guo, X. Zhu, and J. Liu. Cptr: Full transformer network for image captioning. *arXiv preprint arXiv:2101.10804*, 2021.

[20] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[21] J. Lu, C. Xiong, D. Parikh, and R. Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 375–383, 2017.

[22] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[23] Y. Qin, J. Du, Y. Zhang, and H. Lu. Look back and predict forward in image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8367–8375, 2019.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[25] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[26] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[27] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[28] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7008–7024, 2017.

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

[30] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[33] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[34] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[35] X. Yang, K. Tang, H. Zhang, and J. Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019.

[36] T. Yao, Y. Pan, Y. Li, and T. Mei. Exploring visual relationship for image captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 684–699, 2018.

[37] T. Yao, Y. Pan, Y. Li, Z. Qiu, and T. Mei. Boosting image captioning with attributes. In *Proceedings of the IEEE international conference on computer vision*, pages 4894–4902, 2017.

[38] L. Yuan, Q. Hou, Z. Jiang, J. Feng, and S. Yan. Volo: Vision outlooker for visual recognition. *arXiv preprint arXiv:2106.13112*, 2021.

Appendix

1. More Experimental Results

The extra results of the Signal Shot MultiBox Detector (SSD):

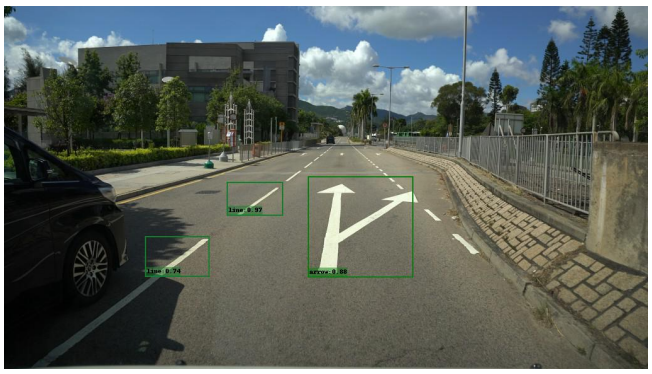

Figure 17: The extra experimental results of the SSD.

The extra results of the You Only Look Once version five:

Figure 18: The extra experimental results of the YOLOv5.

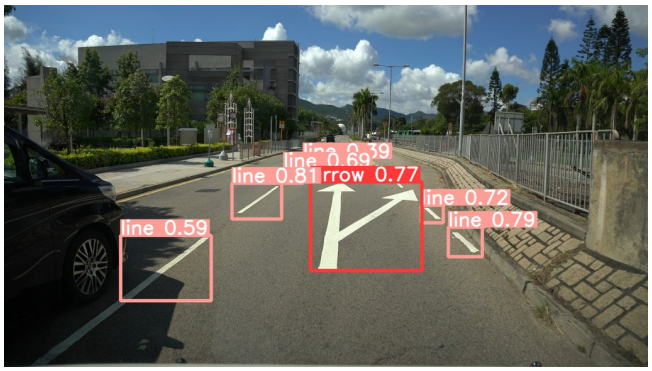

Figure 19: The extra experimental results of the YOLOv5.

The extra results of the Detection Transformer (DETR):

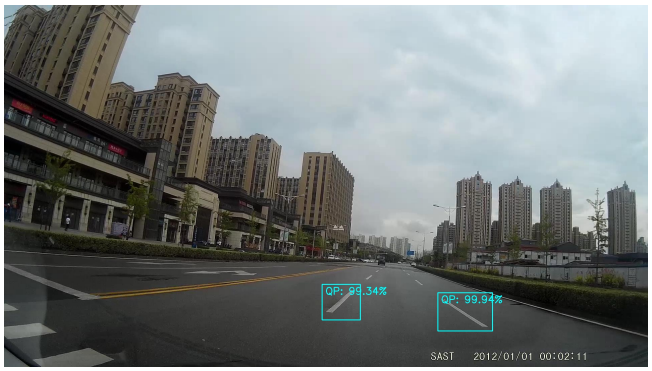

Figure 20: The extra experimental results of the DETR.

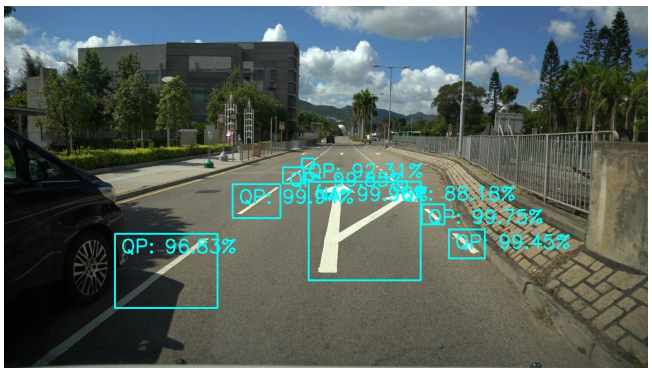

Figure 21: The extra experimental results of the DETR.

2. Type-1 CATR Image Captioning Demos

The extra results of the Type-1 CATR Image Captioning Demos:

Figure 22: The extra results of the Type-1 CATR Image Captioning Demos.

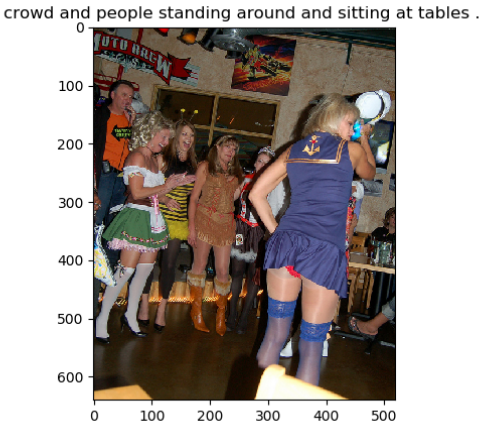

Figure 23: The extra results of the Type-1 CATR Image Captioning Demos.

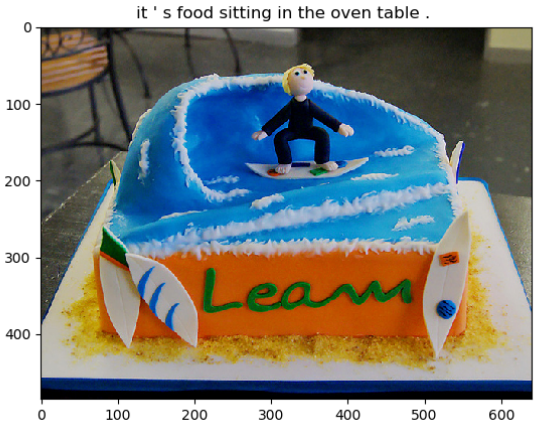

Figure 24: The extra results of the Type-1 CATR Image Captioning Demos.

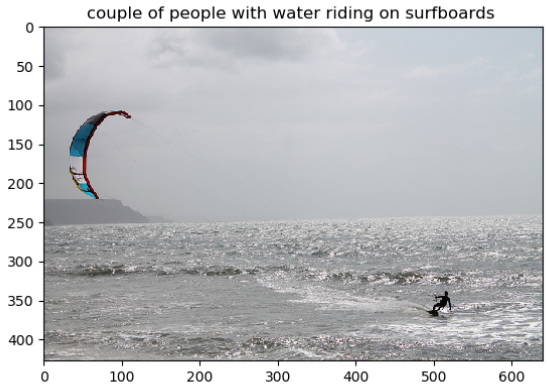

Figure 25: The extra results of the Type-1 CATR Image Captioning Demos.

Figure 26: The extra results of the Type-1 CATR Image Captioning Demos.

Figure 27: The extra results of the Type-1 CATR Image Captioning Demos.

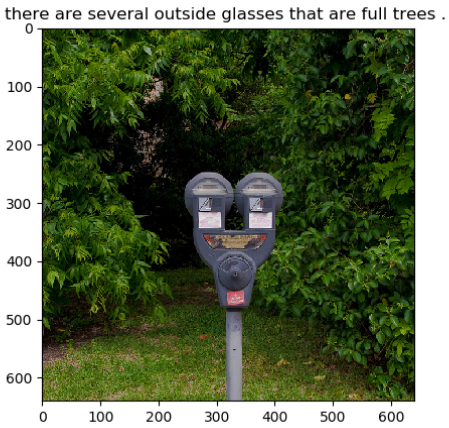

Figure 28: The extra results of the Type-1 CATR Image Captioning Demos.

Figure 29: The extra results of the Type-1 CATR Image Captioning Demos.

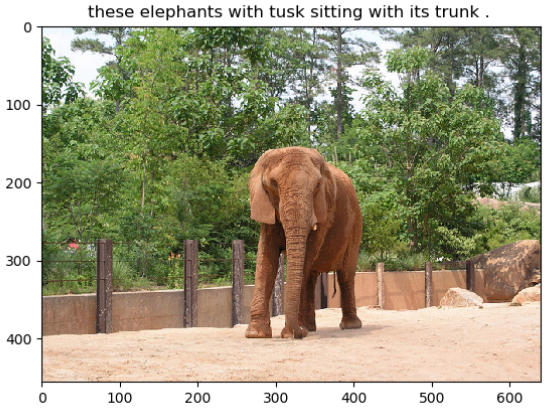

Figure 30: The extra results of the Type-1 CATR Image Captioning Demos.

Figure 31: The extra results of the Type-1 CATR Image Captioning Demos.

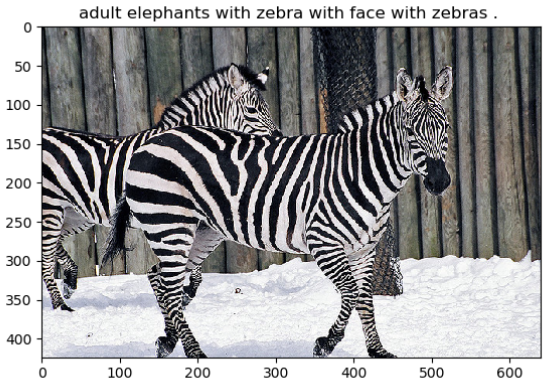

Figure 32: The extra results of the Type-1 CATR Image Captioning Demos.

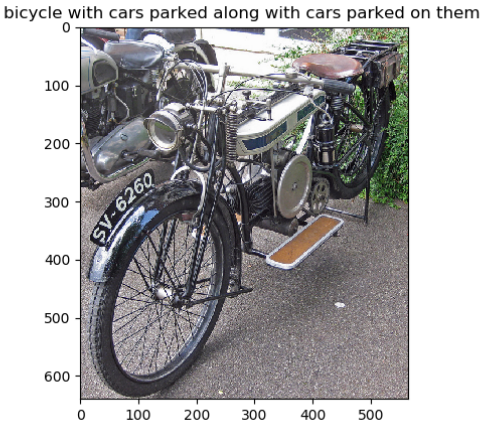

Figure 33: The extra results of the Type-1 CATR Image Captioning Demos.

Figure 34: The extra results of the Type-1 CATR Image Captioning Demos.

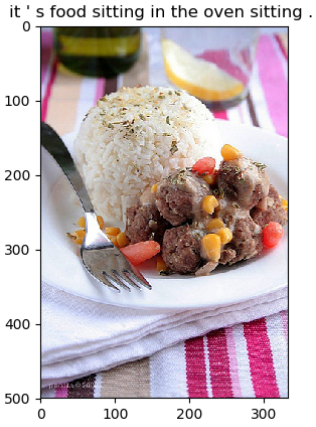

Figure 35: The extra results of the Type-1 CATR Image Captioning Demos.

Figure 36: The extra results of the Type-1 CATR Image Captioning Demos.

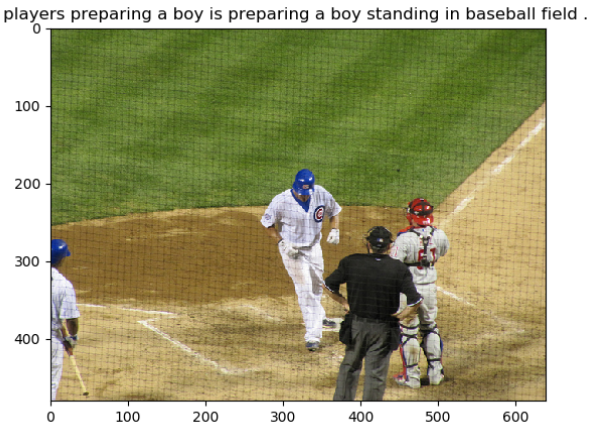

Figure 37: The extra results of the Type-1 CATR Image Captioning Demos.

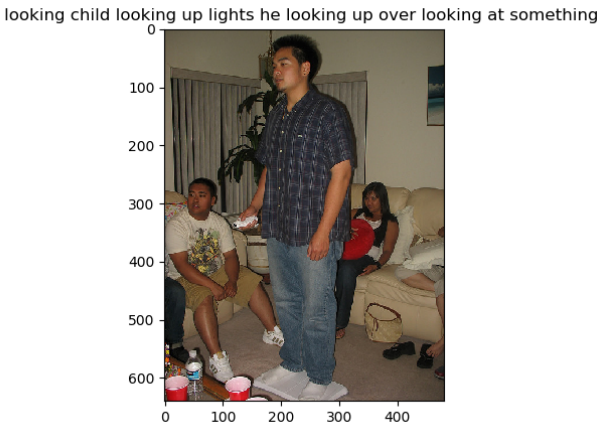

Figure 38: The extra results of the Type-1 CATR Image Captioning Demos.

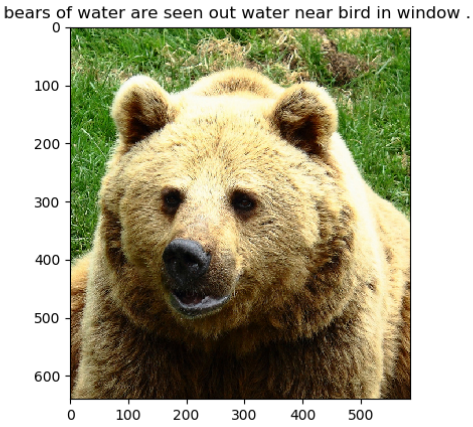

Figure 39: The extra results of the Type-1 CATR Image Captioning Demos.

Figure 40: The extra results of the Type-1 CATR Image Captioning Demos.

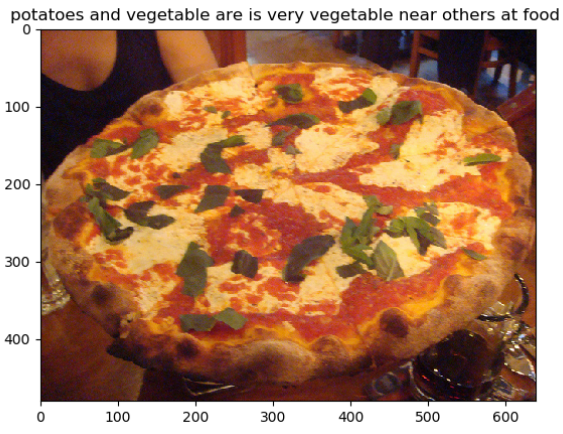

Figure 41: The extra results of the Type-1 CATR Image Captioning Demos.

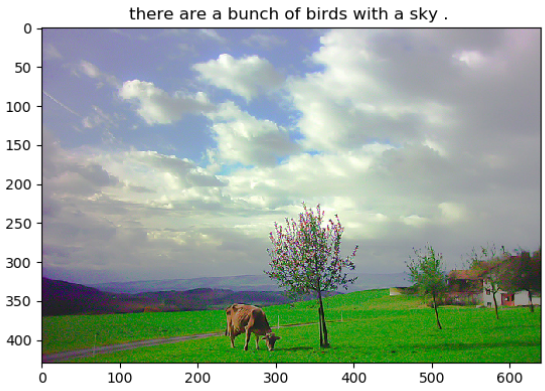

Figure 42: The extra results of the Type-1 CATR Image Captioning Demos.

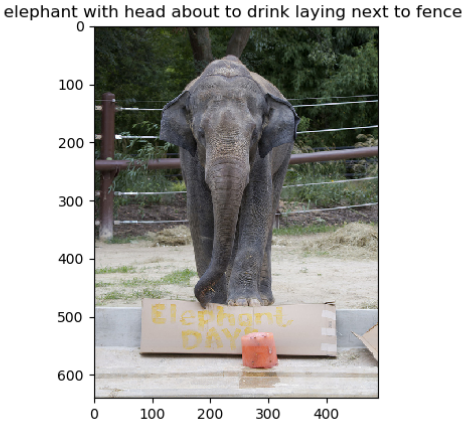

Figure 43: The extra results of the Type-1 CATR Image Captioning Demos.