Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 07, July-2023

Visual Question and Answering in Videos Using Deep Learning

Sanjay B
dept. of Computer Science Engineering
PES University
Bengaluru, India

Sasank K.S.M
dept. of Computer Science Engineering
PES University
Bengaluru, India

Sourabh N
dept. of Computer Science Engineering
PES University
Bengaluru, India

Sudhamsu B
dept. of Computer Science Engineering
PES University
Bengaluru, India

Dr. Preethi P
dept. of Computer Science Engineering
PES University
Bengaluru, India

***Abstract*—Images and videos pertaining to security and surveillance (like cc cameras in banks, dashcams on automobiles, etc.) necessitate the requirements of a lot of resources, especially with respect to the storage of the captured footage since they are required to record round-the-clock. From a financial point of view, the entities responsible for the storage of this content make the decision to only shoot and record, low-quality footage in a bid to lessen the storage consumed, instead, opting for the cameras to act as an imposing deterrent. Such practices tend to have a negative effect when an unexpected event occurs and the footage needs to be reviewed. This paper presents a model to assist in the review of security footage of any quality. The latest deep learning methods, with their increased efficiency, coupled with the Visual Question Answering (VQA) system possess the capability to understand and provide accurate interpretations for each unique scene. The model processes the scene, while simultaneously relating the user query to the specific frame and producing a trustworthy answer. This computation can answer queries involving counting, color detecting, object identification, and yes/no (object presence, etc) type questions. For the processing of the query, a technique known as the Hierarchical Coattention mechanism is employed, to increase the sensitivity and emphasize the most important words in the query. In this method, the multiple layers of recurrent neural networks (RNN) are stacked on top of each other, at the word level, following which the attention model extracts the essential words (words that give meaning to the sentence), amplifying the representation of such words and forming a sentence vector. This lets the model either pay less or more attention to each individual word when formulating the representation of the sentence. Further VGG16 and Faster RCNN have been used to extract visual image features. The dataset used is the COCO dataset, of which, 80% is the training dataset and the rest of the 20% is used as the validation set. The model achieved a global accuracy of 58% against the validation set. Yes/no type questions have the highest rate of accuracy at about 67% and question types involving traditional object counting have an accuracy of 62%.**

***Index Terms*—Visual Question Answering, Hierarchical Coattention mechanism, VGG16, Faster RCNN**

I. Introduction

One of the most important applications of deep learning is in the fields of visual recognition and natural language processing. One of the most prominent applications of these fields is high-level scene interpretation, such as 'Visual Question Answering'. The Visual Question Answering system typically seeks a very detailed understanding of the image and complex reasoning than a system producing generic image interpretations, since, visual questions selectively target different areas of an image, including background details and especially the underlying context. The system should not only have a very good understanding of the entire scene in the picture, but it also should have an understanding of what different activities mean in the context of the query image and relate them to the specific region in the image. The main difference in Visual Question Answering from that of other interpreters is that the search and the reasoning part must be performed in the context of the image. It should also be able to detect objects, classify scenes and use common-sense reasoning and knowledge reasoning wherever deemed necessary. Deep Learning approaches are preferred for VQA, as an increase in data does not result in overfitting. The classical method of deep learning for the Visual Question Answering system involves three stages. Namely, feature extraction from the question, feature extraction from the image, and combining the features to generate an answer to the query. There are many use cases of Visual Question Answering, the primary of which is to help blind and visually-impaired users. Another use case with potentially far-reaching implications is the integration of VQA with image retrieval systems. The primary aim here is to build a surveillance assistance product to overcome the challenges relating to the review of security footage, using 'Visual Question Answering'. In today's digital age, surveillance plays a central and integral role. Yet, many wonder why the technologies behind surveillance cameras, namely, cctv cameras and dashboard cameras used in automobiles, are lagging, especially in terms of resolution, in an age where most smartphones come equipped with very high-performing and powerful cameras. This discrepancy makes more sense when taking into consideration the fact that capturing 4k videos and storing them is a useless expense when 99 % of the captured footage is mundane and uninteresting. According to a British government watchdog, a significant number of homeowners and business owners who own cc cameras do not even switch on their cameras, instead opting to place these cameras in imposing and clearly visible places to act as a deterrent to potential miscreants. The cameras which do record videos, end up capturing grainy footage most of the time, to save up on storage costs. This practice poses a problem when an unforeseen circumstance presents itself, and the captured footage needs to be reviewed by either the owners of the private or commercial entities or by law enforcement officials. A huge number of crimes go unsolved every year, due to unusable surveillance footage. Hence, the intuition behind designing a surveillance assistance product that can overcome the above challenges using 'Visual Question Answering'. Visual Question Answering or VQA uses deep learning, computer vision, and natural language processing techniques. Natural Language Processing is used to process the user query. Computer Vision is used to convert the user-uploaded videos into frames. Deep Learning comes into the picture during query processing, question categorization, and image feature extraction. For the processing of the query, a technique known as the Hierarchical Coattention mechanism is employed, as previously stated.

II. Related work

The most crucial aspect of the project is to localize the region of interest based on the inquiry. Every model has advantages and disadvantages. According to the project requirements, the Dual attention model is the most accurate.

Akanksha Mishra et.al has proposed the Dual attention approach to focus on the cross-domain representation between image and text. It essentially involved a pre-trained model, Faster RCNN, which focuses on the important regions of the image and passes it to the attention of the image mechanism where the queries LSTM output enters with numerical values which help in cross-representation between the image and text. The question and image are passed on to a fully connected neural network which helps in the categorization of the question into one of the 12 classes available the max of the resultant SoftMax function is selected and the image and text are passed to the appropriate model in the subsystem. Even though the model looks rigid, there are improvements that can be made to lessen the computation by considering a metric of choice between the output of AoI and AoQ to eliminate the "absurd" category earlier in the architecture.

Kevin J Shih et.al presented a method to answer visual questions by selecting appropriate image regions according to the query posed. The system aims to find correspondence and to learn to non-linearly map natural language and visual region features to determine relevance. The input is a question and image features from a set of selected candidate regions. The question is split into bins using Stanford Parser, and the answers are encoded using word2vec and a three-layer network. The visual features for each region in the image are encoded using the top two layers of a CNN trained on ImageNet. The language and visual features are then embedded and compared with a dot product, which is soft-maxed to produce a relevant weighting for each region. These weights are then used to calculate a weighted average of concatenated vision and language features. This acts as the input to a two-layered network that outputs confidence for the answer candidate. The model's disadvantage is that it only performs effectively on multiple-choice questions and cannot perform specialized tasks like counting, object and attribute detection, and activity recognition.

Yiwei Ma et.al has used Dual Attention Mechanism (DAM), which completely captures both spatial and channel-wise dependency. The DAM model has a total of three basic blocks i.e., Spatial Attention Block (SAB), Channel Wise attention Block (CAB), and Spatial channel Fusion Block(S-CFB). They have integrated DAM into the standard transformer to form a semantically enhanced dual attention transformer (SDTAR). Which helps in further exploring visual information and performing more effective multi-modal reasoning for image captioning. SAB helps in applying spatial attention to the features of images and sentences. The disadvantage of this approach is that it detects and categorizes items more correctly and can generate sentences. However, if the question is about the position of an object, this model cannot provide a response.

Ashwani Kumar et.al increased the classification accuracy of detecting objects by improving the SSD algorithm by keeping the speed constant. They have improved it in convolutional layers, by using depth-wise separable convolution along with spatial separable convolution generally called multilayer convolutional neural networks. CNN is used to develop a system model which classifies the given object into any of the defined classes. The schemes use multiple images and detect multiple images from these objects and label them to their respective classes.

CNN, Faster CNN, SSD is only suitable for highly powerful computing machines which require a lot of time to train so the authors of this research paper have tried to overcome the limitations of the SSD algorithm by introducing an improved SSD algorithm that has higher detection precision with real-time speed. It is hard for SSD to detect tiny objects as it overlooks the context from outside of the bounding boxes. To address this issue, they used depth-wise separable convolution and spatial separable convolution.

This algorithm comprises two phases, firstly it reduces the feature maps extraction of spatial dimensions by using resolution multipliers. Secondly, it has the application of small convolutional filters for detecting objects by using the best aspect ratio values. It has an SSD with some improvements that include a multi-scale feature map and default boxes for detecting smaller objects. The drawback of this paradigm is that it cannot handle the image's new object classes.

Lakhan H. Jadhav et.al have considered the data obtained mostly from a single static camera in place in which the data is available only in one view. This methodology helps in detecting the objects that are removed in between making it more useful for surveillance considerations. The set frames are sent to the model where the binary representations of the foreground are extracted by subtracting the background from the image. The object is finally classified using the attributes like size, height, etc.

The solution is implemented in a sequential manner consisting of three steps each performing a unique action resulting in an output that is required in the following steps. In the first step, the background is extracted by extracting the static regions in the sequence of frames. This, later on, helps in identifying the dynamically changing foreground and this is done by subtraction of the static regions from the image with another image. Shadow removal method is also involved as the shadow might affect the size attributes of the object. The features like coordinates, height, and width are extracted which will be used in object identification.

The later step involves the classification of the object as either stationary, moving, or unattended with the help of coordinates obtained from the previous step stored in the form of an array. The final step involves an alarming system where a certain category of objects is identified and alarmed.

With the increase in the number of classes in the second step and introduction of the object classification in the second step, it can be used as a positional reference for moving objects and answering queries about them. The designed model was only evaluated with the PETS2006 dataset which predominantly involved data biases towards a certain height and width which might have resulted in skewed results.

Mohammad Yasir Farhad et.al h presented a transfer learning model to categorize sports photos. They gathered information from the internet because there was no existing dataset. They classified them into 18 groups. It proved tough for them to have a huge dataset as they are generating it on their own. They overcame this by using data augmentation, which produced improved results.

For classification, a pre-trained VGG16 transfer learning model is employed. ImageNet dataset was used to train it. They removed the classification layer from the model and replaced it with two dense layers. They froze the weights for the four blocks and used their dataset to train the remaining block and dense layers. The relu activation function is used in each dense layer, and the softmax activation function is used in the output layer to calculate the probabilities for each category

Though the accuracy is good it can be improved with the proper dataset. Data augmentation may avoid overfitting and increase the dataset, it will show some data bias and affect the performance.

Thien Huynh-The et.al attempted to show CNN also presents noticeable performance with good extraction of features compared to LSTM which is generally used for action recognition. So, they used a new encoding technique to transform the skeletal moments into color images. Deep convolutional neural networks are used to capture high-level features. NTU RGB+D dataset is used for training the model.

Pose features such as joint-joint distance and orientation are retrieved from the skeleton sequence. It is then encoded as a chromatic RGB picture. The information is then loaded into a deep convolutional neural network. A pre-trained transfer learning model is employed instead of training CNN from scratch. Then it gets fine-tuned to ensure compatibility, input photos are resized to fit the associated network. And the last fully connected layer is modified to fit action classes.

Instead of converting the original skeleton data to the image directly, pose and motion features are extracted from skeleton sequences and transformed into a color image. It improves the performance as it is also considering motion features. Because of variations in the motion skeleton sequence, the encoding output can be insignificant.

III. Methodology

A. Overview

The overall idea of the method that has been followed, is to not only use the visual features in the image but also give an equal importance to the query given by the user and understand which specific words and parts of the query to put emphasis on, in order to result in a better answer. To achieve this, custom layers have been implemented to find the right parts to concentrate in the question and these layers are embedded into an architecture involving other dense and embedded layers which were hyperparameter tuned in such a way that the result is as accurate as possible. So, the model can be briefly explained as an architecture that attends to different regions of the image and different parts of the question.

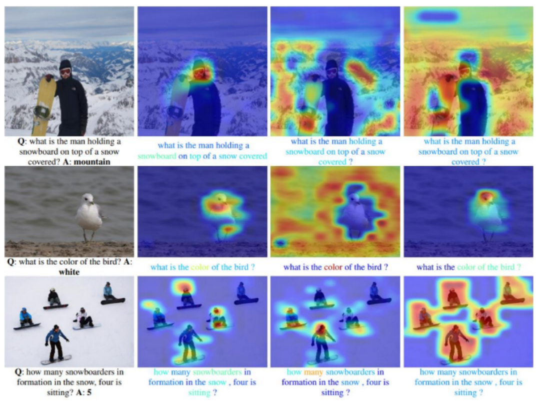

Q: what is the man holding a snowboard on top of a snow covered? A: mountain
what is the man holding a snowboard on top of a snow covered
what is the man holding a snowboard on top of a snow covered ?
what is the man holding a snowboard on top of a snow covered ?
Q: what is the color of the bird? A: white
what is the color of the bird ?
what is the color of the bird ?
what is the color of the bird ?
Q: how many snowboarders in formation in the snow, four is sitting? A: 5
how many snowboarders in formation in the snow , four is sitting ?
how many snowboarders in formation in the snow , four is sitting ?
how many snowboarders in formation in the snow , four is sitting ?

Fig. 1. Graphical Representation of Working of the Model

The general workflow of the answer predictions to a user query to a video is as follows: 1. As the user hits pause and asks a query and gives in the timestamp of the interested scene. 2. The frame is extracted using certain video capture-aiding modules like OpenCV and other APIS's. 3. As the frame is extracted the frame is treated as an image. The image now and the query undergo preprocessing stage the features are extracted using VGG model for the image. 4. The query on the other hand first goes through several preprocessing NLP techniques like removing punctuations and replacing common words removing stop words and such. 5. The query and image features are now in the form of numerical matrices ready to be processed by the weights that are obtained after training the model. 6. The answer is encoded back into the text and given out.

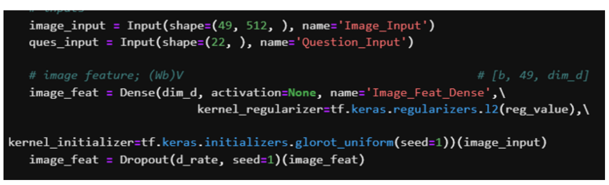
In the above-mentioned steps, the focus is on the stage between steps 4 and 5 because that is where the main idea behind the working of the model lies. Given a query, the phrase level, word level, and question embeddings are extracted and at each stage, the relevance between the image and question is used to obtain the region of interest. Further, the extraction of visual image features is one of the most important and crucial steps for relating the query and relevant regions of interest in the image. VGG16 and Faster RCNN have been used for extracting features for different functionalities. The VGG model was used with all the layers except the final softmax layer, which was implemented for predicting the probabilities of each class in the final prediction. Faster RCNN on the other was trained using the Pascal Voc dataset and was essentially used in detecting all the objects present in the image and was principally included for the counting functionality.

B. Architecture

The model has been built upon several parameters:

- max_answers: The number of output targets of the model that needs to be predicted.
- max_seq_len: The length of input sequences
- vocab_size: Size of the vocabulary which is generally the maximum integer + 1
- dim_d: Hidden dimension
- dim_k: Hidden attention dimension
- l_rate: Learning rate for the model
- d_rate: Dropout rate
- reg_value: Regularization value

These are the parameters that have been tuned to obtain a set of values that result in the best accuracy during evaluation.



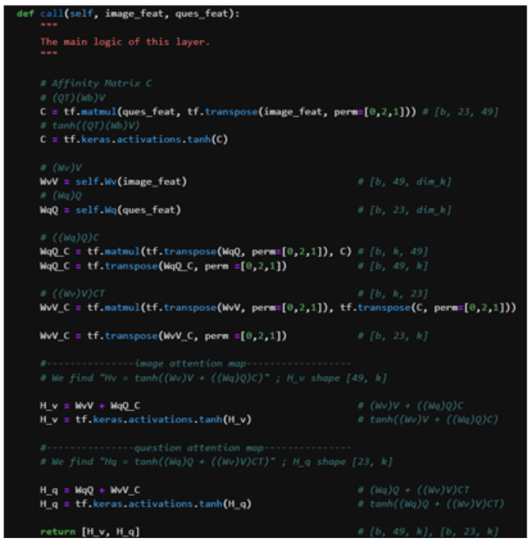
```
    image_input = Input(shape=(49, 512, ), name='Image_Input')
    ques_input = Input(shape=(22, ), name='Question_Input')

    # image feature; (Wb)V                                   # [b, 49, dim_d]
    image_feat = Dense(dim_d, activation=None, name='Image_Feat_Dense',\
                            kernel_regularizer=tf.keras.regularizers.l2(reg_value),\

kernel_initializer=tf.keras.initializers.glorot_uniform(seed=1))(image_input)
    image_feat = Dropout(d_rate, seed=1)(image_feat)
```

Fig. 2. Code level Architecture of the model

The thinking behind the architecture of the model was to put the emphasis on certain parts of the question using the customized layers so that, with the features from the question relevant regions of interest could be concentrated upon in the image to obtain max_answers number of answers. The input layer comprises two inputs, them being image input and questions input. Both are obtained from the scripts developed in the previous sections. The image input layer is followed by an image feature layer, where there is l2 regularization followed by a tuned parameter and the layer is initialized with the weights and uniform distribution values. The seed value here is set to a random value as each time the layer is commenced, we need to initialize the layer with the same value weights. This is crucial as training accuracy will vary if the initial values are tweaked each time. There is also a dropout layer in place to avoid overfitting the model with the training data. The main rationale of the model lies in the question being passed from the “Question_input” layer to the first level which has been named as word level. At the word level, the focus is on the question which is one-hot encoded and mapped into a vector space with the help of an embedding matrix in which weights lie and are learned through the specified number of epochs. The embedding layer is basically a layer that helps in converting the words to fixed-length vectors having real values instead of just ones and zeros. The reason behind using the embedding layer is that it is faster than dense as it makes more simplifying assumptions. The parameters in the embedding layer are as input shape and output shape, set to the maximum vocabulary size, and the output size is adjusted to the dimension calibrated to the size of a hidden layer’s input. The next layer through which the data flows is the custom layer called “AttentionMaps” which will be explained in further sections the input to this layer comes from both the image feature layer from previous sections and the embedding layer discussed above. The output from the AttentionMaps layer is hidden layers weights represented as matrices that are further flown through another custom layer having four different layers inputs. The resulting tensors are added to obtain an output. The obtained output is further passed through a Dense layer named “h_w_Dense” whose activation layer is tanh and has the dimension of hidden layer dimension. With all the above operations the layer of word-level extraction of the query comes to an end. The phrase level features of the question are extracted with the help of an embedding matrix which is obtained from the embedding layer from the previous level and the phrase level features are obtained by using the custom layer called Attention Maps and followed by Context Vector after which the tensor addition is performed. The phrase level features are mainly obtained using a 1-D convolution on the word embeddings, which were mostly zero padded before passing it on to the phrase level layers. After the convolution, it is then followed by a max–pooling layer being applied over n-grams of the query. The dense layer which is the last layer that is being used in the phrase level feature extraction of the query uses the activation function tanh having an l2 regularization to avoid over-fitting. The weights are initialized with uniformly distributed values and have a seed value so that each time the training restarts the values are initialized to the same values to provide ease in the understanding of the layers. The last and final extraction of the features from the query is at a sentence level where the features from the phrase level are sent into an LSTM layer having hidden layer dimensions. The reasoning to use an LSTM layer here is that even after several feature extractions that are happening, the query’s original length and order should be preserved to have a meaningful, full query feature extraction. The next two layers are typically customized to perform a co-attention between the image and the query to obtain the relevant regions of interest. This is done parallelly which will be discussed in the further explanations of the modules. It can be observed that the visual features and query sentence features are treated as tensors and concatenated together. Subsequently, the features are flattened out to be sent through a dense layer having two times the number of hidden layer units and a tanh activation function with the initial weights in a uniform distribution. There is also a drop-out layer included to avoid overfitting. The last layer where the final tensors are predicted has a softmax layer with max_answers number of neurons giving out the probability of each answer being correct.

1) Custom Layers: There are three custom layers that have been scripted to help the model in extracting the features from the user query and phase out the important features.

- AttentionMaps Layer:



```
def call(self, image_feat, ques_feat):
    """
    The main logic of this layer.
    """

    # Affinity Matrix C
    # (QT)(Wb)V
    C = tf.matmul(ques_feat, tf.transpose(image_feat, perm=[0,2,1])) # [b, 23, 49]
    # tanh((QT)(Wb)V)
    C = tf.keras.activations.tanh(C)

    # (Wv)V
    WvV = self.Wv(image_feat)                             # [b, 49, dim_k]
    # (Wq)Q
    WqQ = self.Wq(ques_feat)                              # [b, 23, dim_k]

    # ((Wq)Q)C
    WqQ_C = tf.matmul(tf.transpose(WqQ, perm=[0,2,1]), C) # [b, k, 49]
    WqQ_C = tf.transpose(WqQ_C, perm =[0,2,1])            # [b, 49, k]

    # ((Wv)V)CT                                           # [b, k, 23]
    WvV_C = tf.matmul(tf.transpose(WvV, perm=[0,2,1]), tf.transpose(C, perm=[0,2,1]))

    WvV_C = tf.transpose(WvV_C, perm =[0,2,1])            # [b, 23, k]

    #---------------image attention map------------------
    # We find "Hv = tanh((Wv)V + ((Wq)Q)C)" ; H_v shape [49, k]

    H_v = WvV + WqQ_C                                     # (Wv)V + ((Wq)Q)C
    H_v = tf.keras.activations.tanh(H_v)                  # tanh((Wv)V + ((Wq)Q)C)

    #---------------question attention map---------------
    # We find "Hq = tanh((Wq)Q + ((Wv)V)CT)" ; H_q shape [23, k]

    H_q = WqQ + WvV_C                                     # (Wq)Q + ((Wv)V)CT
    H_q = tf.keras.activations.tanh(H_q)                  # tanh((Wq)Q + ((Wv)V)CT)

    return [H_v, H_q]                                     # [b, 49, k], [b, 23, k]
```

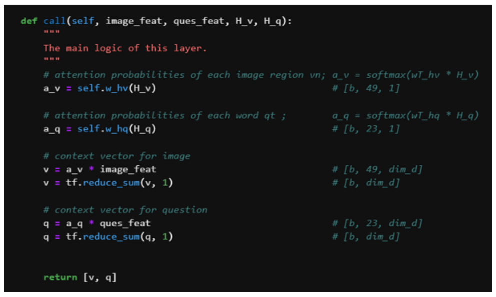
Fig. 3. AttentionMaps Layer code

The focus of this layer is, performing a parallel co-attention on the image and the query simultaneously, in either a parallel way or a sequential way by quickly shifting in between the image features and query region of interest and extracting and localizing the region of interest in the image trying to make the model learn where to look.

This is achieved with the help of the image feature matrix and query valued matrix. The two matrices are multiplied together to result in an affinity matrix. There are other weight matrices initialized with values that are in uniform distribution with each having a suitable dimension compatible with image features and others with query values matrix. Other affinity matrices are created for each image featured and query valued matrix. The final hidden valued matrix weights are applied with a tanh activation for both the image attention map and the question attention map. The resultant weight matrices will be used in further layers.

- Context Vector Attention Layer:

This layer has parameters of image features, question features, hidden valued matrix of image features, and hidden valued matrix of the question. The main motive of this layer is to provide an attention vector of the query consisting of

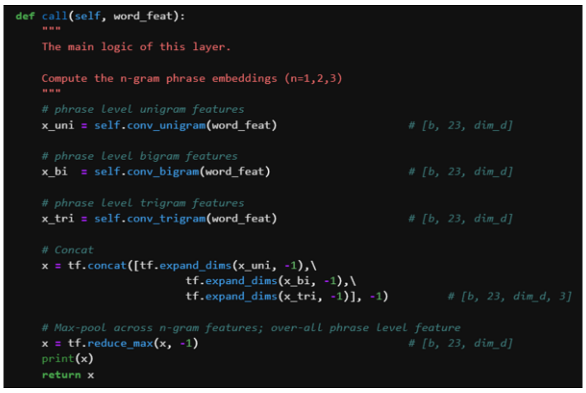


```
def call(self, image_feat, ques_feat, H_v, H_q):
    """
    The main logic of this layer.
    """
    # attention probabilities of each image region vn; a_v = softmax(wT_hv * H_v)
    a_v = self.w_hv(H_v)                              # [b, 49, 1]

    # attention probabilities of each word qt ;         a_q = softmax(wT_hq * H_q)
    a_q = self.w_hq(H_q)                              # [b, 23, 1]

    # context vector for image
    v = a_v * image_feat                              # [b, 49, dim_d]
    v = tf.reduce_sum(v, 1)                           # [b, dim_d]

    # context vector for question
    q = a_q * ques_feat                               # [b, 23, dim_d]
    q = tf.reduce_sum(q, 1)                           # [b, dim_d]


    return [v, q]
```

Fig. 4. Context Vector Layer Code

probabilities of each word in the query which will be obtained by passing it through a neuron consisting of the activation function softmax. After the values are obtained, the vectors are set to a particular dimension which will be spread over and multiplied with the image feature attention map to obtain the probability of the regions of interest like a matrix filled with probabilities. So, there would be a context vector for both the image and the query. The resultant matrix will be reduced to a vector sum of a matrix of dimension one and maximum length. The results are returned to be forwarded to the further layers.

- Phrase Level Features Layer

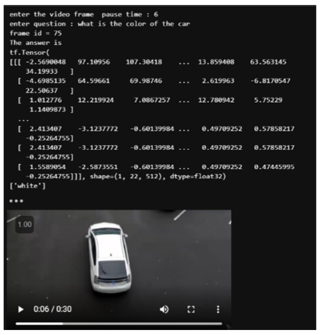


```
def call(self, word_feat):
    """
    The main logic of this layer.

    Compute the n-gram phrase embeddings (n=1,2,3)
    """
    # phrase level unigram features
    x_uni = self.conv_unigram(word_feat)                    # [b, 23, dim_d]

    # phrase level bigram features
    x_bi  = self.conv_bigram(word_feat)                     # [b, 23, dim_d]

    # phrase level trigram features
    x_tri = self.conv_trigram(word_feat)                    # [b, 23, dim_d]

    # Concat
    x = tf.concat([tf.expand_dims(x_uni, -1),\
                   tf.expand_dims(x_bi, -1),\
                   tf.expand_dims(x_tri, -1)], -1)          # [b, 23, dim_d, 3]

    # Max-pool across n-gram features; over-all phrase level feature
    x = tf.reduce_max(x, -1)                                # [b, 23, dim_d]
    print(x)
    return x
```

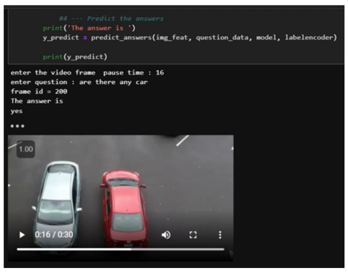
Fig. 5. Phrase Level Feature Extraction Layer Code

This layer is solely responsible for computing the n-gram embedding of the query. This feature layer extracts the phrase-level features of the query. The phrase features are computed by applying 1-D convolution on the word embedding vectors with filters of three window sizes: unigram, bigram, and trigram. The word-level features, Qw, are appropriately 0-padded before feeding into bigram and trigram convolutions to maintain the length of the sequence after convolution. Given the convolution result, max-pooling is applied across different n-grams at each word location to obtain phrase-level features. It can be deduced that as the query is passed as a parameter and all the unigram, bigram, and trigram embeddings of the query are calculated and concatenated as a matrix after converting them to tensors. The size of the matrix is reshaped and returned to be used in phrase-level extraction of the model.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 07, July-2023



```
enter the video frame  pause time : 6
enter question : what is the color of the car
frame id = 75
The answer is
tf.Tensor(
[[[ -2.5690048   97.10956    107.30418    ...  13.859408    63.563145
    34.19933   ]
  [ -4.6985135   64.59661     69.98746    ...   2.619963    -6.8170547
    22.50637   ]
  [  1.012776    12.219924     7.0867257  ...  12.780942     5.75229
     1.1409873 ]
  ...
  [  2.413407    -3.1237772   -0.60139984 ...   0.49709252   0.57858217
    -0.25264755]
  [  2.413407    -3.1237772   -0.60139984 ...   0.49709252   0.57858217
    -0.25264755]
  [  1.5589054   -2.5873551   -0.60139984 ...   0.49709252   0.47445995
    -0.25264755]]], shape=(1, 22, 512), dtype=float32)
['white']
```



```
        #4 --- Predict the answers
    print('The answer is ')
    y_predict = predict_answers(img_feat, question_data, model, labelencoder)

    print(y_predict)
```

```
enter the video frame  pause time : 16
enter question : are there any car
frame id = 200
The answer is
yes
```

Fig. 6. Some of the Outputs received for color detection and object presence

IV. Results

As stated previously, the proposed methodology is to answer the queries with the concept of co-attention maps. Various performance metrics such as accuracy, specificity, and sensitivity are applied to the training dataset which is 80% of the coco dataset and the rest of the 20% is the validation set. While validated against this dataset the global accuracy achieved is 58% which is a noticeable percentage in the research conducted on visual question answering when compared to other works. The questions are classified into different categories, with yes/no questions having the highest rate of accuracy (about 67%). When compared to other state-of-the-art models, traditional object counting in visual question answering improves to 62% accuracy. And some categories showed significant improvement compared to other state of art models. However, overall performance is good and latency is low, despite the fact that extraction of the specific image from the video where the user wants to ask the question is necessitated, and then processing it to provide an answer.

TABLE I
Output Metrics

	Precision	*Recall*	*F1-score*
Accuracy	Na	Na	0.58
Macro avg	0.39	0.55	0.44
Weighted avg	0.57	0.58	0.58

V. Conclusion and Future Work

The co-attention maps were highly useful in answering the queries more precisely since they took into account the image locations very well. The mapping of questions to images at each level, i.e. word level, and phrase level, worked effectively in recognizing the things in the images. It also made certain that the image is extracted in good quality from the video in which the query is raised. Significant improvements in accuracy have been noticed in some question categories. It can be concluded from this research that there is a lot of future potential in which it can detect and answer queries very well, and can be extremely beneficial for identifying things through CCTV cameras. Another direction in which this work could be further continued is to embed VQA into all kinds of images in a database, to make image retrieval systems exponentially more efficient. This could have a massive impact on search engines of e-commerce platforms. VQA can also be used for data-heavy computations like self-driving automobiles or for automating trivial tasks like finding an empty parking spot or an empty seat at an event.

References

[1] Y. Ma et al.,"Knowing what it is: Semantic-enhanced Dual Attention Transformer," in IEEE Transactions on Multimedia, doi: 10.1109/TMM.2022.3164787.
[2] A. Mishra, A. Anand and P. Guha,"Dual Attention and Question Categorization based Visual Question Answering,",in IEEE Transactions on Artificial Intelligence, doi:10.1109/TAI.2022.3160418.
[3] S. S. Samant, N. L. Bhanu Murthy and A. Malapati, "Improving Term Weighting Schemes for Short Text Classification in Vector Space Model,"in IEEE Access, vol. 7, pp. 166578-166592, 2019, doi: 10.1109/ACCESS.2019.2953918.
[4] https://doi.org/10.1186/s13638-020-01826-x
[5] L. H. Jadhav and B. F. Momin, "Detection and identification of unattended/removed objects in video surveillance," 2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), 2016, pp. 1770-1773, doi: 10.1109/RTEICT.2016.7808138.
[6] K. Patel et al., "Facial Sentiment Analysis Using AI Techniques: State-of-the-Art, Taxonomies, and Challenges," in IEEE Access, vol. 8, pp. 90495-90519, 2020, doi: 10.1109/ACCESS.2020.2993803.
[7] J. Chen, Q. Mao and L. Xue, "Visual Sentiment Analysis With Active Learning," in IEEE Access, vol. 8, pp. 185899-185908, 2020, doi: 10.1109/ACCESS.2020.3024948.
[8] K. J. Shih, S. Singh and D. Hoiem, "Where to Look: Focus Regions for Visual Question Answering," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4613-4621, doi: 10.1109/CVPR.2016.499.
[9] Y. Huang, J. Chen, W. Ouyang, W. Wan and Y. Xue, "Image Captioning With End-to-End Attribute Detection and Subsequent Attributes Prediction," in IEEE Transactions on Image Processing, vol. 29, pp. 4013-4026, 2020, doi: 10.1109/TIP.2020.2969330.
[10] M. Yasir Farhad, S. Hossain, M. Rezaul Karim Tanvir and S. Ameen Chowdhury, "Sports-Net18: Various Sports Classification using Transfer Learning," 2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI), 2020, pp. 1-4, doi: 10.1109/STI50764.2020.9350415..
[11] T. Huynh-The, C. Hua and D. Kim, "Learning Action Images Using Deep Convolutional Neural Networks For 3D Action Recognition," 2019 IEEE Sensors Applications Symposium (SAS), 2019, pp. 1-6, doi: 10.1109/SAS.2019.8705977.
[12] https://towardsdatascience.com/color-identification-in-images-machine-learning-application-b26e770c4c71