Reinforcement Learning with Transformers for Visual Question Answering

A project report submitted in partial fulfilment of

the requirements for the award of the Degree of

Master of Technology

In

Computer Science and Engineering

By

Bala Tripura Kumari (17011Pl0501)

Under the esteemed guidance of

DR. V. KAMAKSHI PRASAD
Professor - JNTUH
Department of Computer Science and Engineering

Department of Computer Science and Engineering

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
COLLEGE OF ENGINEERING HYDERABAD
Kukatpally, Hyderabad – 500085

2021 – 2022

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERISTY HYDERABAD
COLLEGE OF ENGINEERING HYDERABAD
Kukatpally, Hyderabad – 500085

2021-2022

Department of Computer Science and Engineering

DECLARATION

I, **Bala Tripura Kumari (17011P0501)**, hereby certify that the project report entitled **"Reinforcement Learning with Transformers for Visual Question Answering",** carried out under the guidance of **Dr. V. Kamakshi Prasad**, is submitted in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering. This is a record of bonafide work carried out by me and the results embodied in this project have not been reproduced/copied from any source and have not been submitted to any other University or Institute for the award of any other degree or diploma.

Bala Tripura Kumari (17011P0501)
Department of Computer Science & Engineering,
JNTUH College of Engineering Hyderabad,
Hyderabad – 500085

Date:

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERISTY HYDERABAD
COLLEGE OF ENGINEERING HYDERABAD
Kukatpally, Hyderabad – 500085

2021-2022

Department of Computer Science and Engineering

CERTIFICATE BY THE SUPERVISOR

This is to certify that the project report entitled **"Reinforcement Learning with Transformers for Visual Question Answering"** being submitted by **Bala Tripura Kumari(17011P0501)** in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering, is a record of bonafide work carried out by her. The results are verified and found satisfactory.

Dr. V. Kamakshi Prasad
Professor– JNTUH,
Department of Computer Science and Engineering,
JNTUH College of Engineering Hyderabad,
Hyderabad – 500 085.

Date:

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERISTY HYDERABAD
COLLEGE OF ENGINEERING HYDERABAD
Kukatpally, Hyderabad – 500085

2021-2022

Department of Computer Science and Engineering

CERTIFICATE BY THE HEAD OF THE DEPARTMENT

This is to certify that the project report entitled **“Reinforcement Learning with Transformers for Visual Question Answering”** being submitted by **Bala Tripura Kumari(17011P0501)** in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering, is a record of bonafide work carried out by her. The results are verified and found satisfactory.

Dr. D. Vasumathi
Professor & Head of the Department
Department of Computer Science and Engineering
JNTUH College of Engineering Hyderabad
Hyderabad – 500085

Date:

ACKNOWLEDGMENT

I take this opportunity to exhibit my deep gratitude to all those who have extended their help by every means possible for the successful completion of this project.

I am grateful to **Prof. Dr. A Prabhu Kumar**, Principal, JNTUH College of Engineering Hyderabad for his support while pursuing this project.

I thank **Prof. Dr. D Vasumathi,** Professor & Head, Department of Computer Science and Engineering for her constant support.

I would like to thank my guide **Prof**. **Dr. V. Kamakshi Prasad** for his timely advice and valuable suggestions while working on this project as well as throughout the M. Tech course. He has helped to keep me focused and pointed in the right direction.

I am grateful to the **Project Review Committee members** and Department of Computer Science and Engineering who have helped in successfully completing this project by giving their valuable suggestions and support.

I thank my family friends and all professors of CSE Department who have supported us all the way during the project.

Bala Tripura Kumari (17011P0501)

Department of Computer Science & Engineering,

JNTUH College of Engineering Hyderabad,

Hyderabad – 500085.

Abstract

Reinforcement Learning with Transformers for Visual Question Answering

The task of visual question answering (VQA) is to generate an answer for a question according to the content of an image being asked. In VQA, an algorithm needs to answer text-based questions about images. In this process, the critical problems of effectively embedding the question feature and image feature as well as transforming the features to the prediction of answer are still faithfully unresolved.

Existing models analyze the input images at a typically small resolution, often leading to discarding valuable fine-grained details. To overcome this limitation, in this work reinforcement learning based approach is proposed that works by applying a series of transformation operations on the images (translation) in order to facilitate answering the question at hand. This allows for performing fine-grained analysis, by only focusing on parts of the image that are relevant to the question in hand.

Visual Question Answering(VQA 2.0) dataset is used for the purpose of training and testing this proposed model.

Table of Contents

Table of Figures

1. CHAPTER

Introduction

1.1. Introduction

Visual Question Answering (VQA) is a computer vision task where a system is given a text-based question about an image, and it must infer the answer. Questions can be arbitrary and they encompass many sub-problems in computer vision. They can range from simple yes/no questions to complex counting questions, open-ended questions and question relating to spatial relationships among objects.

The broader idea of this task is to design systems that can understand the contents of an image like how humans do and communicate effectively about that image in natural language. It is a challenging task as both Image based models and natural language models are required.

This task can be useful for blind and visually impaired users. It can also be integrated into image retrieval systems. It also can help in providing information about an image on the Web or any social media.

1.2. Objective

The current attention mechanisms analyse the input visual data at a fixed resolution and use only entire image to extract features. This process, despite allowing for reducing the computational complexity of the models, has drawbacks. First, it restricts the fidelity of the input, leading to loosing several fine grained details, especially for smaller or thin objects that might end up covering only a few pixels. they are sensitive to the scale of the objects appearing in images. Therefore, if the same object, but in a different size, appear the employed model might fail to recognize it.

The objective of this study is to a provide a deep reinforcement learning (RL)-based approach that can overcome the above limitations by employing a methodology by applying transformations to image which facilitates to only focus on those parts of image relevant to the question.

1.3. Problem Statement

Visual Question Answering is a task that attempts to correctly answer questions in natural language regarding an image input. The broader idea of this problem is to design systems that can understand the contents of an image like how humans do and communicate effectively about that image in natural language. It is a challenging task as both Image based models and natural language models are required.

Currently, attention mechanisms are used for extracting image features. The computation complexity is less, but the images are processed at fixed resolution size and objects which cover only a few pixels can be left unrecognized. To solve this issue, the proposed system applies transformations on an image to focus on only those parts which are relevant to question. This will help in analyzing image at a finer level.

For this purpose, the proposed system trains an RL agent, which when given an image and a question, can transform the image such that only the part of image relevant to the question is used for answering the question.

2. CHAPTER

Literature Survey

2.1. MUTAN :MultiModal Tucker Fusion for Visual Question Answering

AUTHORS: Hedi Ben-younes, Remo Cadene, Matthieu Cord, Nicolas Thome

Bilinear models provide an appealing framework for mixing and merging information in Visual Question Answering (VQA) tasks. They help to learn high level associations between question meaning and visual concepts in the image, but they suffer from huge dimensionality issues. In this paper MUTAN, a multimodal tensor-based Tucker decomposition to efficiently parametrize bilinear interactions between visual and textual representations is introduced. Additionally to the Tucker framework, a low-rank matrix-based decomposition to explicitly constrain the interaction rank is designed. With MUTAN, the complexity of the merging scheme is controlled while keeping nice interpretable fusion relations.

2.2 VISUALBERT

AUTHORS: Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, Kai-Wei-Chang

In this paper, VisualBERT, a simple and flexible framework for modeling a broad range of vision-and-language tasks is proposed. VisualBERT consists of a stack of Transformer layers that implicitly align elements of an input text and regions in an associated input image with self-attention. Further two visually-grounded language model objectives are provided for pre-training VisualBERT on image caption data. Experiments on four vision-and-language tasks including VQA, VCR, NLVR2 , and Flickr30K show that VisualBERT outperforms or rivals with state-of-the-art models while being significantly simpler. Further analysis demonstrates that VisualBERT can ground elements of language to image regions without any explicit supervision and is even sensitive to syntactic relationships, tracking, for example, associations between verbs and image regions corresponding to their arguments.

2.3 Hierarchical Object Detection with Deep Reinforcement Learning

AUTHORS : Miriam Bellver Bueno, Xavier Giro-i-Nieto, Ferran Marques, Jordi Torres

In this paper, a method for performing hierarchical object detection in images guided by a deep reinforcement learning agent is presented. The key idea is to focus on those parts of the image that contain richer information and zoom on them. An intelligent agent is trained, that, given an image window, can decide where to focus the attention among five different predefined region candidates (smaller windows). This procedure is iterated providing a hierarchical image analysis. Two different candidate proposal strategies to guide the object search are compared: with and without overlap. Moreover, this paper compares two different strategies to extract features from a convolutional neural network for each region proposal: a first one that computes new feature maps for each region proposal, and a second one that computes the feature maps for the whole image to later generate crops for each region proposal. Experiments indicate better results for the overlapping candidate proposal strategy and a loss of performance for the cropped image features due to the loss of spatial resolution. While this loss seems unavoidable when working with large amounts of object candidates, the much more reduced amount of region proposals generated by our reinforcement learning agent allows considering extracting features for each location without sharing convolutional computation among regions.

3. CHAPTER

Requirements Specifications

3.1. Tools used

3.1.1.TensorFlow:

TensorFlow is opensource software developed by Google Brains Team and was initially released on November 9, 2015. It is mainly used to counter problems related to Machine Learning and Artificial Intelligence. It is not only used in Python predominantly, but also can be used along with JavaScript, C++ and Java. Its flexible architecture enables us to deploy it across different myriad platforms taking from single CPU, GPU to cluster of CPUs, GPUs.

The autoDifferiantion is an effective feature of it, as it automatically calculates the gradient vector of a model having huge number of parameters but keeping the track of the operations done. The gradients are of high importance as they provide us a scope to optimize the algorithms. It also provides us various optimizers like ADAGRAD, Stochastic Gradient Descent (SDG) needed for getting the best possible outcome.

3.1.2.Keras:

Keras original authored by Francois Chollet and developed many other later. It is an Application Program Interface which works on top of TensorFlow primarily used for building deep learning models, making it more productive and simple to use it. It consists of Model APIs, Layers APIs, Callbacks APIs, Optimizers, Metrics, Built-in small Datasets etc.

3.1.3.NLTK:

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

3.2. System Specifications

3.2.1. Hardware Requirements

- **Processor:** Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz   2.80 GHz
- **RAM:** 16.0 GB
- **GPU:** NVIDIA GeForce GTX 1050 Ti

3.2.2. Software Requirements:

- **Operating System:** 64-bit operating system
- **Coding Language:** Python 3.8.8
- **TensorFlow Version:** TensorFlow 2.7.0
- **OpenCv Version:** cv2 4.5.4

4. CHAPTER

Proposed Work

4.1. Architecture

The following is the architecture of the proposed system:

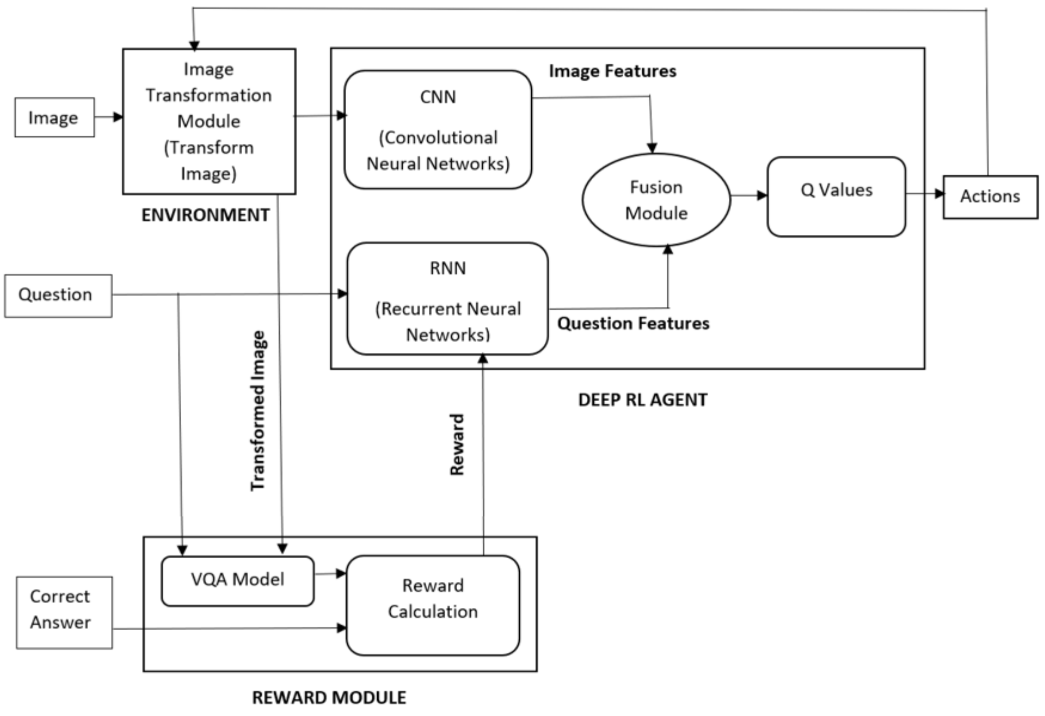


Fig.4.1 Proposed System Architecture

4.2. Dataset

VQA 2.0 (Balanced Real Images) dataset is used for training and testing. It contains open-ended questions about images.The following are the details of the dataset:

- 204,721 COCO images

- 1,105,904 questions
- 10 ground truth answers per question

4.3. Modules

4.3.1 Environment

This module is the environment with which the deep reinforcement learning agent interacts with. The environment consists of:

- State: A tuple consisting of image and question (image, question)

- Action: There are 5 actions that can be performed on the state:

  Action 1: Translate image to the right by ‘delta’ pixels

  Action 2: Translate image to the left by ‘delta’ pixels

  Action 3: Translate image downwards by ‘delta’ pixels

  Action 4: Translate image upwards by ‘delta’ pixels

  Action 5: Do nothing

4.3.1.1 Image Transformation Module

Given an image and the action to be taken, this module performs the action and return the transformed image and the questions as the new state to the Deep RL Agent.

4.3.2 Deep Reinforcement Learning Agent

The agent, when given the state as an input which consists of image and question, predicts the action to be performed on the state to maximize the reward.The following is the architecture of the model:

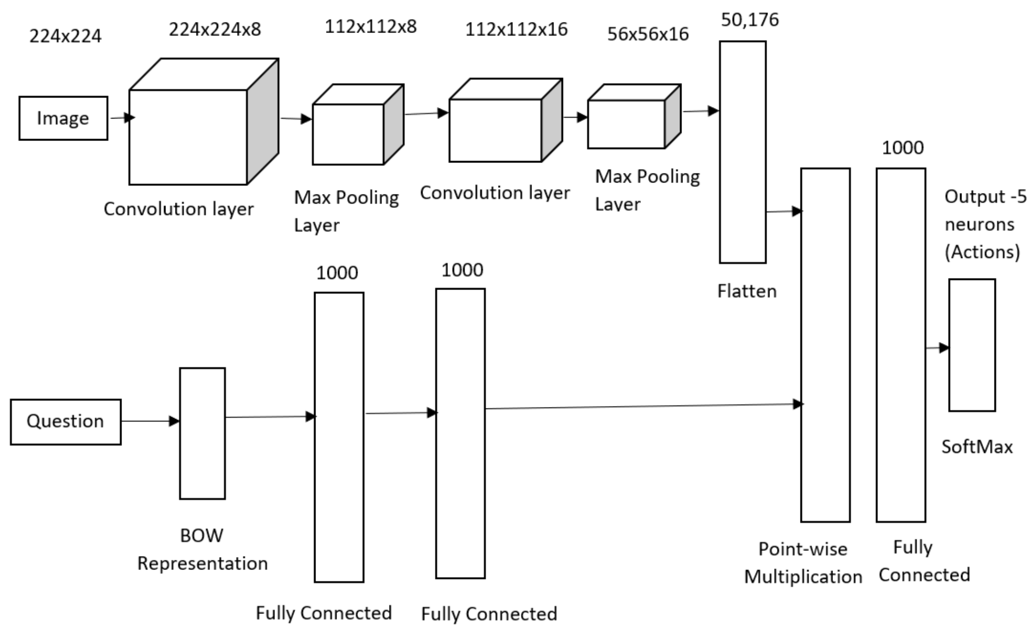


Fig.4.3.2 Deep RL Agent Architecture

The steps performed in the above model are:

- Image Feature Extraction: Convolutional Neural Network is used to extract features from the image.It consists of 2 pairs of convolutional and max pooling layers followed by a flatten layer

- Question Feature Extraction: Bag Of Words module is used to extract BOW feature from question, followd by 2 fully connected layer

- Fusion:Point-wise multiplication is used to fuse the image and question features

- Prediction of action:A fully connected layer followd by a softmax layer consisting of 5 neurons is used to predict the next action to be performed.

4.3.3 Reward Module

This module calculated the reward for every step taken given transformed image and the question and actual answer as inputs.Pretrained VQA models like VisualBert and MUTAN are used to predict the answer given an image and question.Along with reward, the confidence with which answer is predicted is also returned by this module.The reward is calculated as follows:

- If predcited answer does not match actual answer a negative reward of -1 is returned.

- If predicted answer matches the actual answer reward is the change in confidence of the predicted answer from the previous step:

```
Reward=(confidence of answer in current step)-(confidence of answer in previous step)
```

4.4 Procedure

1. The image,question,answer pairs are given input to the model. The number of images is equal to number of episodes the model runs.

2. For every episode, the Deep RL Agent will run 10 steps, wherein in every step the agent will predict the action to be taken in the current state to maximise the reward.

3. Initially, since the agent is not trained, exploration of the environment is done(selecting random actions).After few episodes eploitation is done(predicting action from trained model weights). To control this trade-off between exploration and expoitation, ‘epsilon’ is used which is modified every episode.

4. Once, the action to be performed is predicted, the action is performed by Image transformation module which becomes the new state of the system.

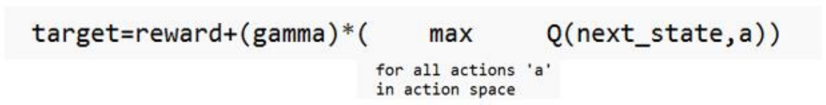
5. The new state (transformed image,question) is passed to Reward module . It uses pretrained VQA models to predict answer and return the reward based on above reward formula.

6. The current state,action,new state and the reward are returned to the agent.This tuple is called an experince.

7. After every step , the experiences are stored in replay buffer.

8. After every 20 episodes, the RL agent network is trained. For this, 100 experiences are sampled form replay buffer.Then for every

experience (state,action,reward,next_state)target Q values is calculated as follows:



```
target=reward+(gamma)*(    max      Q(next_state,a))
                      for all actions 'a'
                      in action space
```

9. The Deep RL Agent is trained for 50 epochs given the input and target values

10. This process is done for every episode

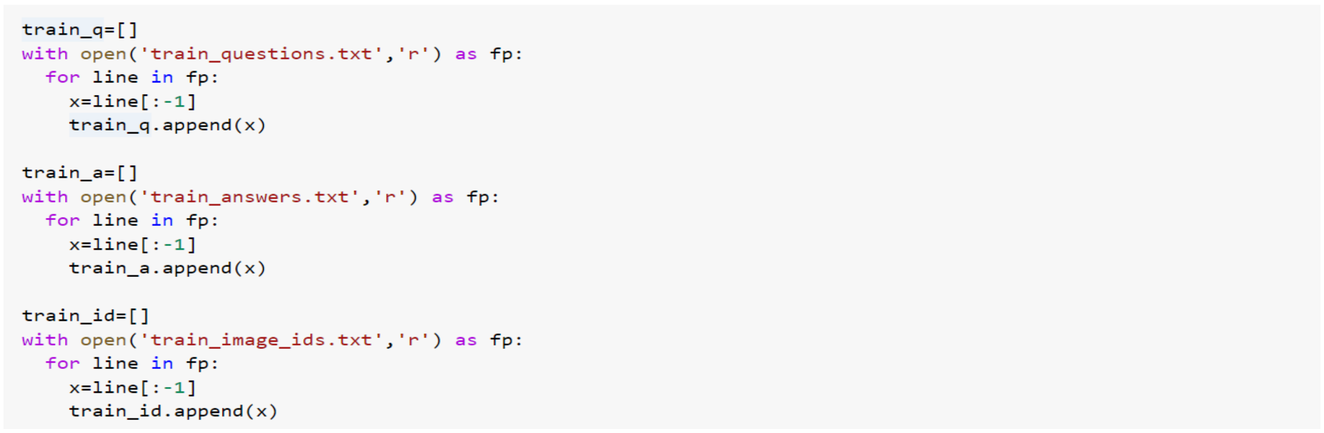
5. CHAPTER

Implementation

5.1. Dataset

For training, 10000 (image, question, answer) pairs are downloaded from VQA 2.0 training set. All the questions are stored in ‘train_questions.txt’, answers in ‘train_answers.txt’, image id’s in ‘train_image_ids.txt’.

For testing, 7000 (image, question, answer) pairs are downloaded from VQA 2.0 testing set. All the questions are stored in ‘test_questions.txt’, answers in ‘test_answers.txt’, image id’s in ‘image_ids.txt’.Then the data is pre-processed using following code:

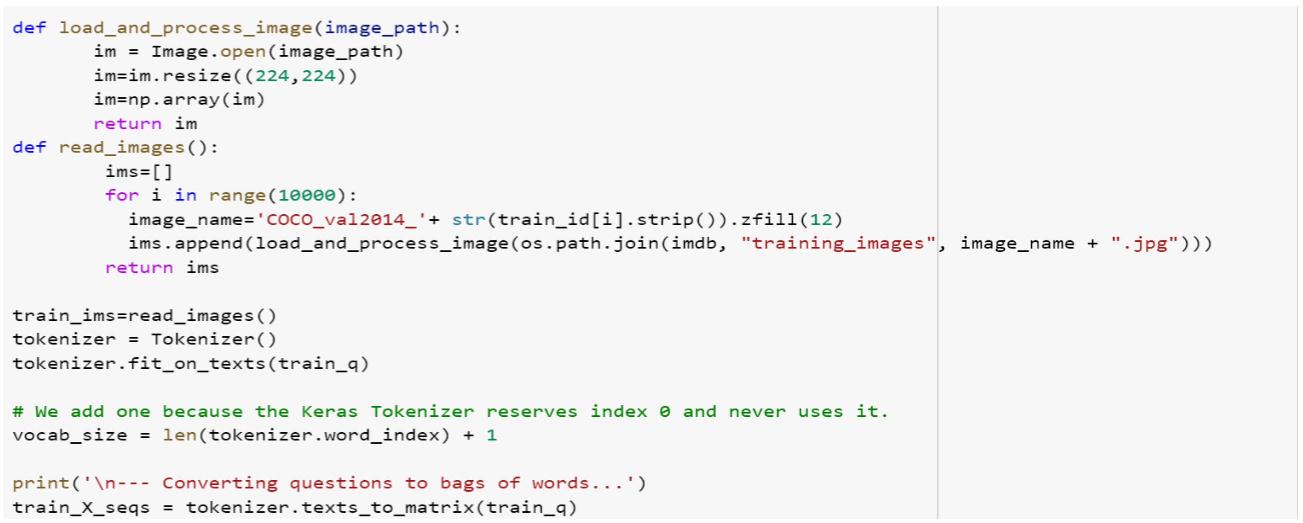


```
train_q=[]
with open('train_questions.txt','r') as fp:
  for line in fp:
    x=line[:-1]
    train_q.append(x)

train_a=[]
with open('train_answers.txt','r') as fp:
  for line in fp:
    x=line[:-1]
    train_a.append(x)

train_id=[]
with open('train_image_ids.txt','r') as fp:
  for line in fp:
    x=line[:-1]
    train_id.append(x)
```

Fig.5.1.1 Pre-process dataset



```
def load_and_process_image(image_path):
       im = Image.open(image_path)
       im=im.resize((224,224))
       im=np.array(im)
       return im
def read_images():
        ims=[]
       for i in range(10000):
         image_name='COCO_val2014_'+ str(train_id[i].strip()).zfill(12)
         ims.append(load_and_process_image(os.path.join(imdb, "training_images", image_name + ".jpg")))
        return ims

train_ims=read_images()
tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_q)

# We add one because the Keras Tokenizer reserves index 0 and never uses it.
vocab_size = len(tokenizer.word_index) + 1

print('\n--- Converting questions to bags of words...')
train_X_seqs = tokenizer.texts_to_matrix(train_q)
```

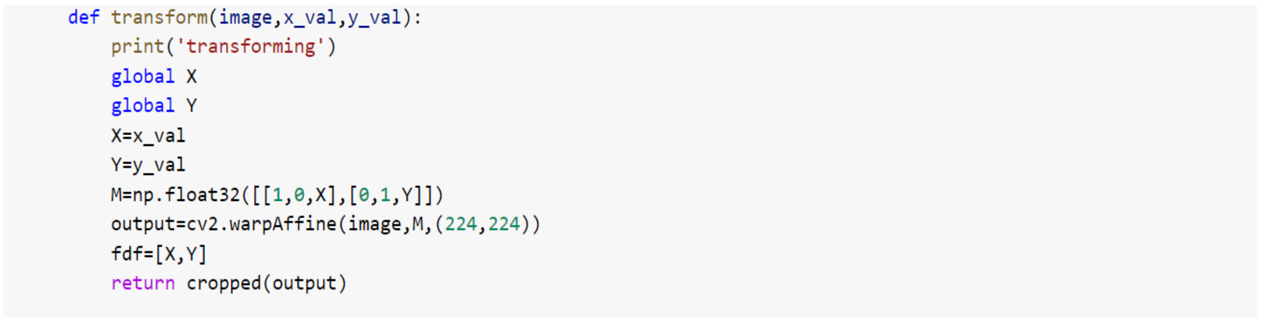
Fig.5.1.2 Load images

5.2. Implementation of Training Phase

5.2.1 Image transformation module

This module performs action on the image and returns new image.

The code is as follows:



```
def transform(image,x_val,y_val):
    print('transforming')
    global X
    global Y
    X=x_val
    Y=y_val
    M=np.float32([[1,0,X],[0,1,Y]])
    output=cv2.warpAffine(image,M,(224,224))
    fdf=[X,Y]
    return cropped(output)
```

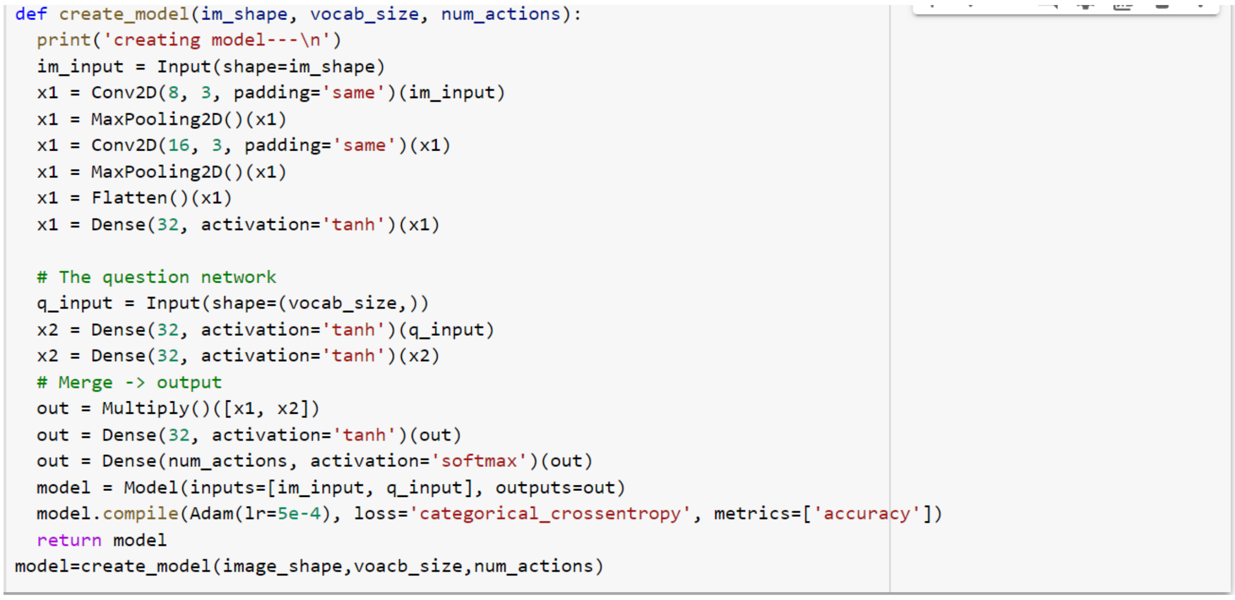
Fig.5.2.1 Transform Image

Here ‘x_val’ and ‘y_val’ are the pixels by which the image must be translated in x and y direction.

5.2.2 Deep RL Agent

1. Given the size of input image, the vocabulary size and number of actions, the model is created which extract image features, text features, combine them and predict the action to be taken.

2. The code illustrating on creating the model architecture is shown below:

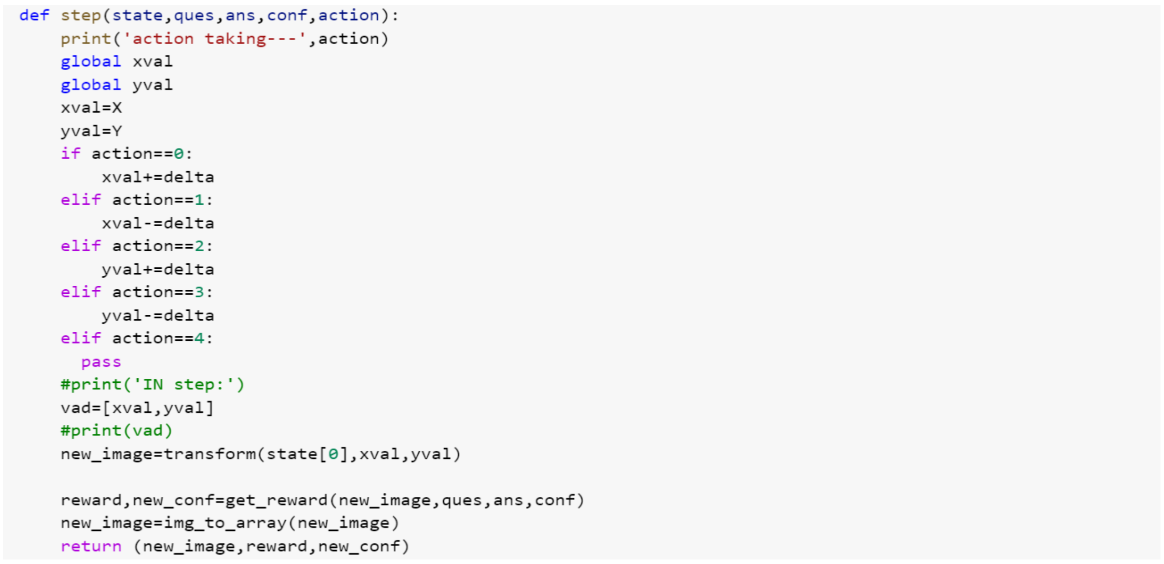


```
def create_model(im_shape, vocab_size, num_actions):
  print('creating model---\n')
  im_input = Input(shape=im_shape)
  x1 = Conv2D(8, 3, padding='same')(im_input)
  x1 = MaxPooling2D()(x1)
  x1 = Conv2D(16, 3, padding='same')(x1)
  x1 = MaxPooling2D()(x1)
  x1 = Flatten()(x1)
  x1 = Dense(32, activation='tanh')(x1)

  # The question network
  q_input = Input(shape=(vocab_size,))
  x2 = Dense(32, activation='tanh')(q_input)
  x2 = Dense(32, activation='tanh')(x2)
  # Merge -> output
  out = Multiply()([x1, x2])
  out = Dense(32, activation='tanh')(out)
  out = Dense(num_actions, activation='softmax')(out)
  model = Model(inputs=[im_input, q_input], outputs=out)
  model.compile(Adam(lr=5e-4), loss='categorical_crossentropy', metrics=['accuracy'])
  return model
model=create_model(image_shape,voacb_size,num_actions)
```

Fig.5.2.2.1 Create Model

3. For every step in each episode, the agent will perform the action, calculate the reward using Reward Module which takes new state and question as inputs. Then it returns the new state and the reward to store in the replay memory. To transform the image delta value used is 30 pixels. The code illustrating this is as follows:

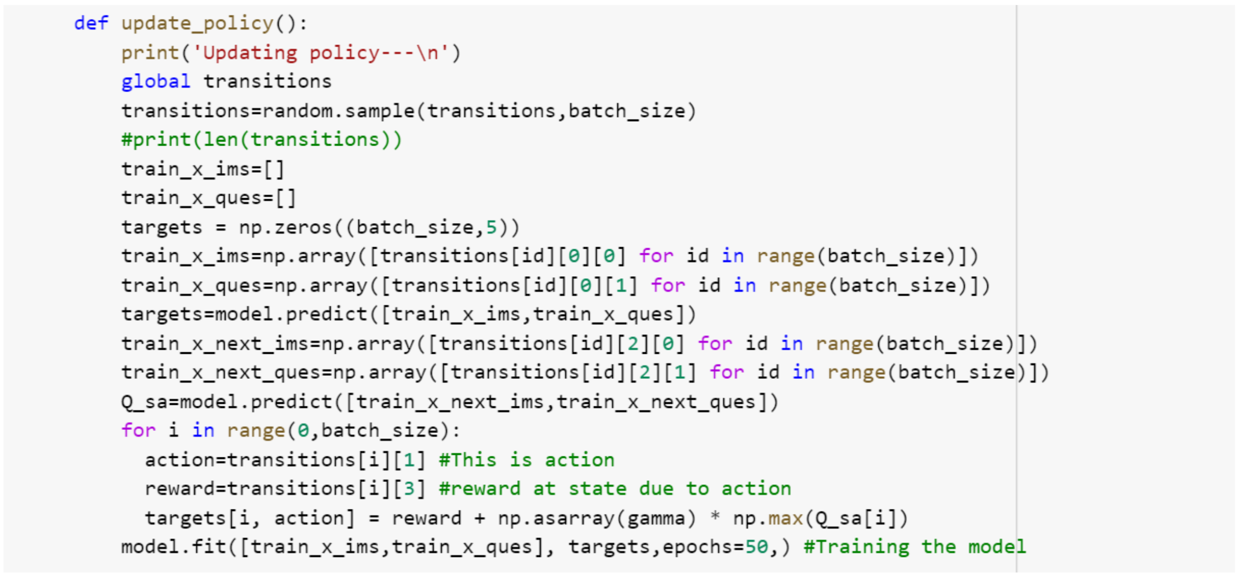


```
def step(state,ques,ans,conf,action):
    print('action taking---',action)
    global xval
    global yval
    xval=X
    yval=Y
    if action==0:
        xval+=delta
    elif action==1:
        xval-=delta
    elif action==2:
        yval+=delta
    elif action==3:
        yval-=delta
    elif action==4:
      pass
    #print('IN step:')
    vad=[xval,yval]
    #print(vad)
    new_image=transform(state[0],xval,yval)

    reward,new_conf=get_reward(new_image,ques,ans,conf)
    new_image=img_to_array(new_image)
    return (new_image,reward,new_conf)
```

Fig.5.2.2.2 Executing a single step in episode

4. After every 20 episodes the RL Model is trained by sampling 100 experiences from the replay memory.

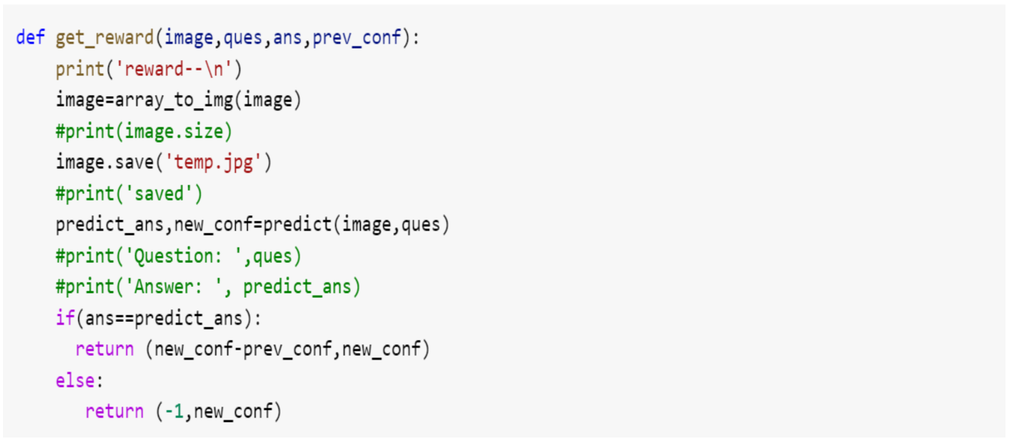


```
def update_policy():
    print('Updating policy---\n')
    global transitions
    transitions=random.sample(transitions,batch_size)
    #print(len(transitions))
    train_x_ims=[]
    train_x_ques=[]
    targets = np.zeros((batch_size,5))
    train_x_ims=np.array([transitions[id][0][0] for id in range(batch_size)])
    train_x_ques=np.array([transitions[id][0][1] for id in range(batch_size)])
    targets=model.predict([train_x_ims,train_x_ques])
    train_x_next_ims=np.array([transitions[id][2][0] for id in range(batch_size)])
    train_x_next_ques=np.array([transitions[id][2][1] for id in range(batch_size)])
    Q_sa=model.predict([train_x_next_ims,train_x_next_ques])
    for i in range(0,batch_size):
      action=transitions[i][1] #This is action
      reward=transitions[i][3] #reward at state due to action
      targets[i, action] = reward + np.asarray(gamma) * np.max(Q_sa[i])
    model.fit([train_x_ims,train_x_ques], targets,epochs=50,) #Training the model
```

Fig.5.2.2.3 Updating policy

5.2.3 Reward Module

Given a state and question, the answer is predicted using pretrained VQA models. The VQA models used are ‘VisualBERT’ and ‘MUTAN’ Based on the predicted answer reward is calculated:

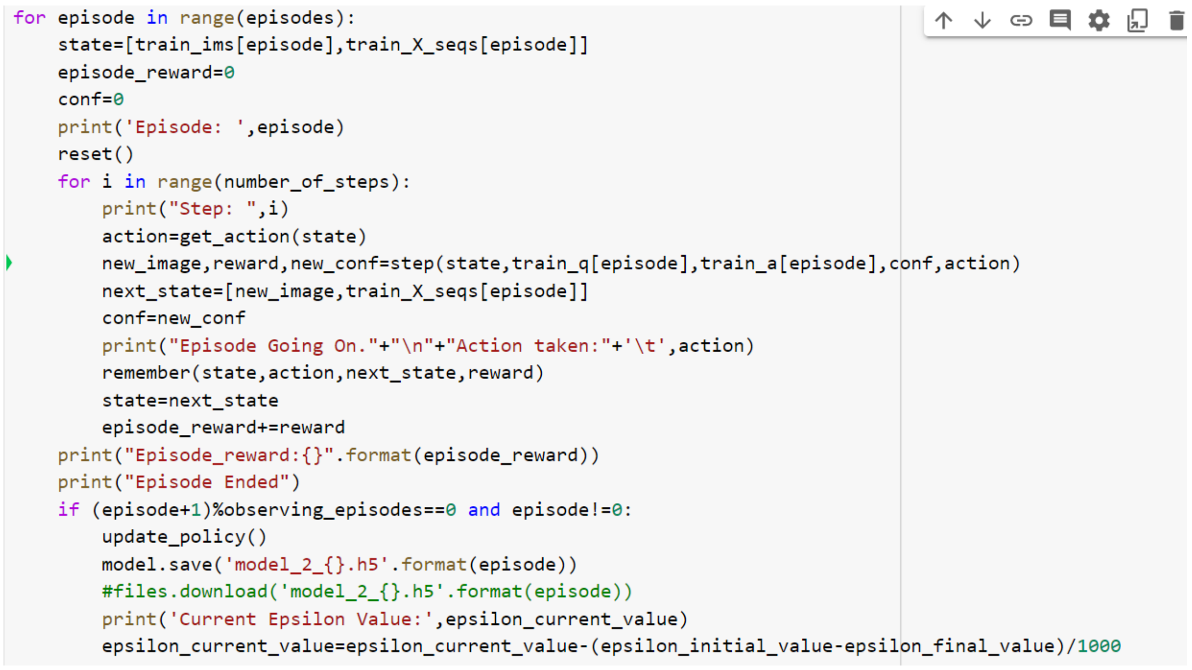


```
def get_reward(image,ques,ans,prev_conf):
    print('reward--\n')
    image=array_to_img(image)
    #print(image.size)
    image.save('temp.jpg')
    #print('saved')
    predict_ans,new_conf=predict(image,ques)
    #print('Question: ',ques)
    #print('Answer: ', predict_ans)
    if(ans==predict_ans):
      return (new_conf-prev_conf,new_conf)
    else:
       return (-1,new_conf)
```

Fig.5.2.3.1 Reward Calculation

5.2.4 Training the Model

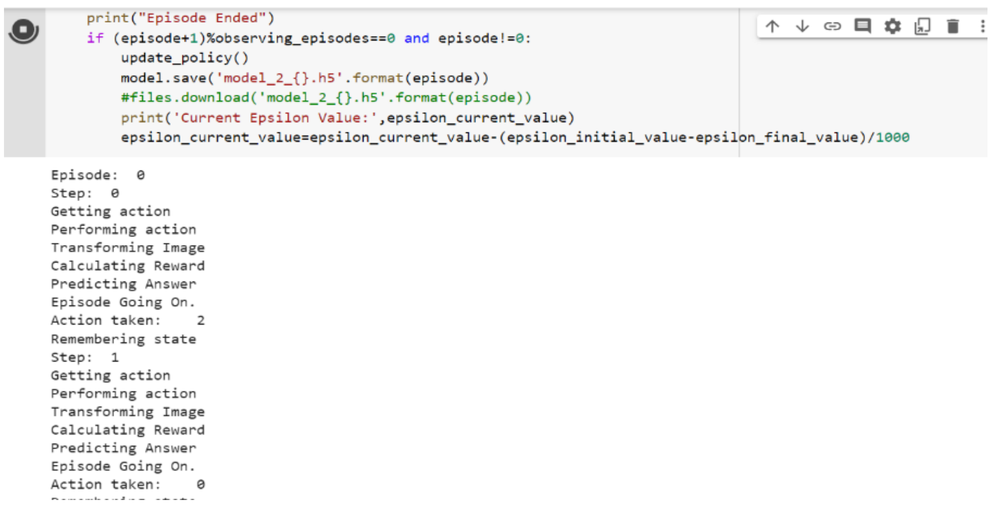
- The model is run for 10000 episodes, where in every episode 10 steps are run.
- In every step, given the current state, the model will predict the action to be performed, return the new state along with the reward.
- After every 20 episodes RL model is updated and is saved to load while testing
- Also for exploration-exploitation trade-off during action selection, epsilon value is updated every iteration.



```
for episode in range(episodes):
    state=[train_ims[episode],train_X_seqs[episode]]
    episode_reward=0
    conf=0
    print('Episode: ',episode)
    reset()
    for i in range(number_of_steps):
        print("Step: ",i)
        action=get_action(state)
        new_image,reward,new_conf=step(state,train_q[episode],train_a[episode],conf,action)
        next_state=[new_image,train_X_seqs[episode]]
        conf=new_conf
        print("Episode Going On."+"\n"+"Action taken:"+'\t',action)
        remember(state,action,next_state,reward)
        state=next_state
        episode_reward+=reward
    print("Episode_reward:{}".format(episode_reward))
    print("Episode Ended")
    if (episode+1)%observing_episodes==0 and episode!=0:
        update_policy()
        model.save('model_2_{}.h5'.format(episode))
        #files.download('model_2_{}.h5'.format(episode))
        print('Current Epsilon Value:',epsilon_current_value)
        epsilon_current_value=epsilon_current_value-(epsilon_initial_value-epsilon_final_value)/1000
```

Fig.5.2.4.1 Training model

- The following is a snapshot at the beginning of training when the episode started:



```
print("Episode Ended")
if (episode+1)%observing_episodes==0 and episode!=0:
    update_policy()
    model.save('model_2_{}.h5'.format(episode))
    #files.download('model_2_{}.h5'.format(episode))
    print('Current Epsilon Value:',epsilon_current_value)
    epsilon_current_value=epsilon_current_value-(epsilon_initial_value-epsilon_final_value)/1000
```

```
Episode:  0
Step:  0
Getting action
Performing action
Transforming Image
Calculating Reward
Predicting Answer
Episode Going On.
Action taken:     2
Remembering state
Step:  1
Getting action
Performing action
Transforming Image
Calculating Reward
Predicting Answer
Episode Going On.
Action taken:     0
```

Fig.5.2.4.2 Training Snapshot

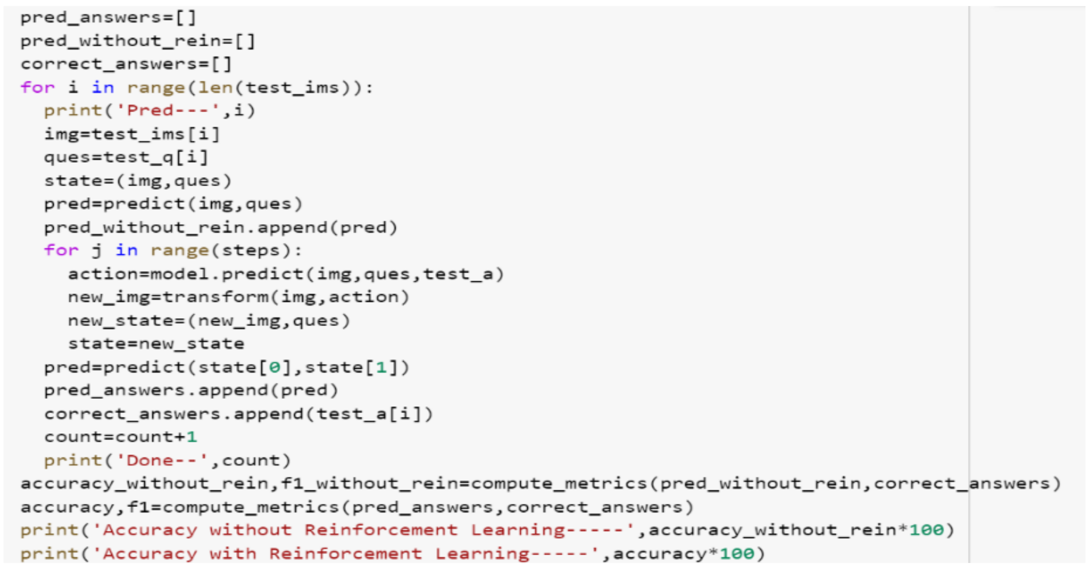
5.3. Implementation of Testing Phase

The testing questions, images and answers are pre-processed same as of training set. The trained RL Agent model is loaded. Then, for every image, question pair in testing, the agent will iterate through 10 steps, perform the transformations by predicting actions from trained RL model. Using the last state, it predicts answer using pretrained VQA model and returns it.

Then, the predicted answer and correct answers are passed to calculate accuracy. The accuracies are calculated for:

- Predicting answers using only pretrained VQA model

- Predicting answers with reinforcement learning using pretrained VQA model



```
pred_answers=[]
pred_without_rein=[]
correct_answers=[]
for i in range(len(test_ims)):
  print('Pred---',i)
  img=test_ims[i]
  ques=test_q[i]
  state=(img,ques)
  pred=predict(img,ques)
  pred_without_rein.append(pred)
  for j in range(steps):
    action=model.predict(img,ques,test_a)
    new_img=transform(img,action)
    new_state=(new_img,ques)
    state=new_state
  pred=predict(state[0],state[1])
  pred_answers.append(pred)
  correct_answers.append(test_a[i])
  count=count+1
  print('Done--',count)
accuracy_without_rein,f1_without_rein=compute_metrics(pred_without_rein,correct_answers)
accuracy,f1=compute_metrics(pred_answers,correct_answers)
print('Accuracy without Reinforcement Learning-----',accuracy_without_rein*100)
print('Accuracy with Reinforcement Learning-----',accuracy*100)
```

Fig.5.3.1 Testing Code

6. CHAPTER

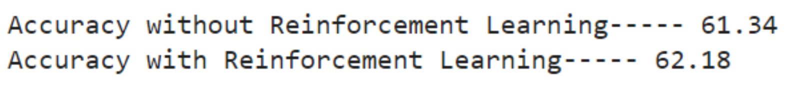
Results and Discussions

6.1 MUTAN as pre-trained model:

Multimodal Tucker Fusion for Visual Question Answering (MUTAN) pre-trained model is used for predicting the answer given an image and question. The dataset is tested for 2 models:

- MUTAN pre-trained model
- Reinforcement Learning with MUTAN pre-trained model

On executing the testing code, the following is the result:



```
Accuracy without Reinforcement Learning----- 61.34
Accuracy with Reinforcement Learning----- 62.18
```

Fig.6.1 Accuracy MUTAN

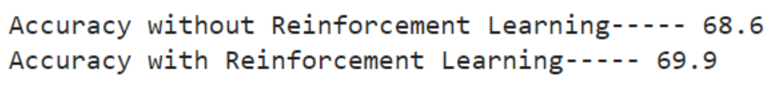
6.2 VisualBERT as pre-trained model:

VisualBERT pre-trained model is used for predicting the answer given an image and question.

The dataset is tested for 2 models:

- VisualBERT pre-trained model
- Reinforcement Learning with VisualBERT pre-trained model

On executing the testing code, the following is the result:



```
Accuracy without Reinforcement Learning----- 68.6
Accuracy with Reinforcement Learning----- 69.9
```

Fig.6.2 Accuracy VisualBERT

From the above results the following is the summary for the 2 models used:

S.no	**Model**	**Accuracy**
1	MUTAN	61.34
2	Reinforcement Learning with MUTAN	62.18
3	VisualBERT	68.6
4	Reinforcement Learning with VisualBERT	69.9

Fig.6.3 Results Summary

7. CHAPTER

Conclusion and Future Scope

7.1. Conclusion

Using Reinforcement Learning for transforming images, the resolution at which the images are processed is increased .This allows for performing fine-grained analysis of the images. This is advantageous especially when there are very thin objects which occupy small pixels in the image. As a result of this there is a slight improvement in the accuracy. The number of episodes trained did affect the accuracy. The number of steps trained did not affect the accuracy much.

Using the Reinforcement Learning combined with MUTAN produced accuracy of 62.18 while combining with VisualBERT produced accuracy of 69.9. In both cases there is an increase in accuracy using Reinforcement Learning rather than only the VQA model.

7.2. Future Scope

The model can be augmented with two extra actions: Zoom in and Zoom out. This will help analyze the images with finer granularity.

In the Deep RL agent, instead of CNN, Visual Transformer can be used to extract image features and BERT to extract text features. This increases the complexity but also increases accuracy.

Instead of using pretrained VQA model for prediction, a VQA model can be trained simultaneously along with the RL Agent

References

- Bellver, Miriam & Giró-i-Nieto, Xavier & Marques, Ferran & Torres, Jordi. (2016). Hierarchical Object Detection with Deep Reinforcement Learning. Advances in Parallel Computing. 31. 10.3233/978-1-61499-822-8-164.

- Gordon, Daniel & Kembhavi, Aniruddha & Rastegari, Mohammad & Redmon, Joseph & Fox, Dieter & Farhadi, Ali. (2018). IQA: Visual Question Answering in Interactive Environments. 4089-4098. 10.1109/CVPR.2018.00430.

- Kafle, Kushal & Kanan, Christopher. (2016). Visual Question Answering: Datasets, Algorithms, and Future Challenges. Computer Vision and Image Understanding. 163. 10.1016/j.cviu.2017.06.005.

- https://www.tensorflow.org/

- https://keras.io/

- Computer Vision: Algorithms and Applications, by Richard Szeliski

- Concise Computer Vision: An Introduction into Theory and Algorithms by Reinhard Klette

- Deep Learning for Vision Systems by Mohamed Elgendy