**Sprint 2**

**Medical Visual Question Answering**

**Group: Ruiling Zhang, Yuko Ishikawa, Zihao Shen**

**Overall:**

We duplicate and test an open-source python project, which is based on pytorch.

**The project architecture:**

These are important modules and their usage of the project

**Resize picture:**

Module: PIL.image

Method: image.resize([224,224],Image.ANTIALIAS)

Output: The 224\*224 picture with the same content, but maybe change aspect ratio.

**Make vocabulary for question and answer:**

Questions:

Data structure: set

Method: split a question and update the set.

Output: All tokens used in question dataset

Answers:

Data structure: defaultdict(lambda: 0)

Dictionary keys: content of answers

Dictionary values: times of use

Method: Increase the value of the key by 1 when find the key used. Reverse sort the dictionary and pick the top 1000 answers.

Output: the top 1000 answers.

**Build VQA inputs:**

Data Structure: list and dictionary

Dictionary keys: [image\_name, image\_path, question\_id, question\_str, question\_tokens], and may also have [all\_answers, valid\_answers] if the answer is given. All answers are all possible answer for this question, and valid answers are answers in both ‘All answers’ and ‘Answer vocabulary’ built before.

Output: a list of dictionaries, store it as .npy file.

**Train/valid Dataset:**

vqa: the list of dictionaries built by ‘build vqa inputs.py’

qst\_vocab/ans\_vocab: class, which read question/answer vocabulary file created before, and make the content as list and dictionary.

transform: ToTensor() and Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225)). The parameters are mean and standard derivation, and the value is referring ImageNet. But the value may only be used in "natural scenes" (people, buildings, animals, varied lighting/ angles/ backgrounds, etc.)

**Data loader:**

Method: torch.utils.data.DataLoader

Random choose #batch\_size data from dataset built before.

**ImgEncoder:**

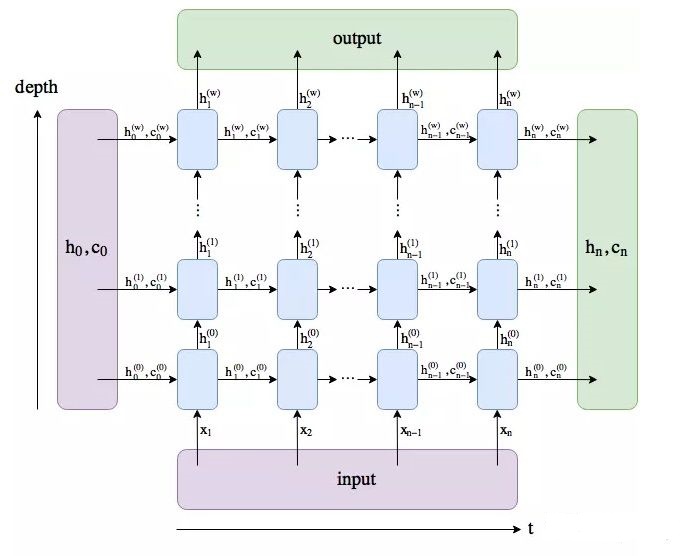
Using pretrained vgg19 model. Removing the last full connect layer, and using Linear to convert the output of the edited vgg19 network to a 1024 feature vector of the image



When forward propagation, calulate the norm of the output of the network. We cannot sure what the usage of this part.

**QstEncoder:**

Using LSTM model. Input size is 300, which is a word and it will be picked from the dictionary/ matrix built by pytorch.nn Embedding function. Hidden size is 512, which can be customized. And there are two layers, it can be recognized as depth in this picture.



Also using Linear to convert the parameters of LSTM to a 1024 feature vector of the sentence.

When forward propagation, the question will convert to the format of LSTM input and activate by tanh. And use the parameters of hidden and cell, we think that is the output and the long memory state. And the last Linear layer will convert the 2(hidden and cell) \* 2(layers) \* 512(hidden size) vector to a 1024 feature vector of the sentence.

**VQAModel:**

Using ImgEncoder and QstEncoder. Multiplying the two outputs of the two models, and two full connected layers to convert the 1024 vector to a 1000 vector.

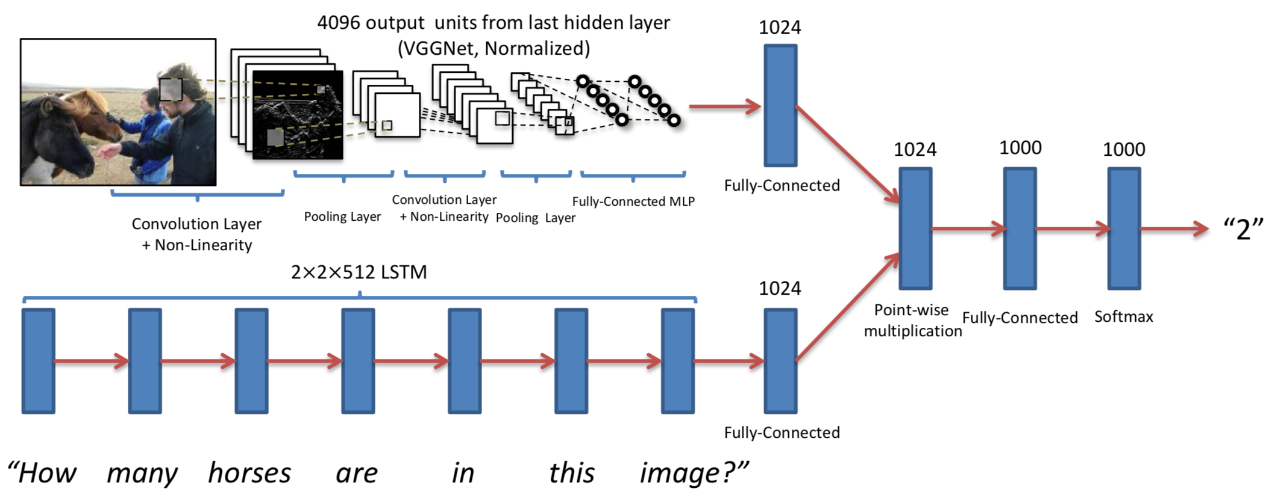
**Train:**

Loss function: CrossEntropyLoss()

Optimizer: Adam()

Change learn rate: StepLR()

Use the VQAModel to predict, the output will be a 1000 vector, which indicate the possibility of the answer depend on the answer vocabulary built before. Then pick the max possibility to calculate the accuracy.



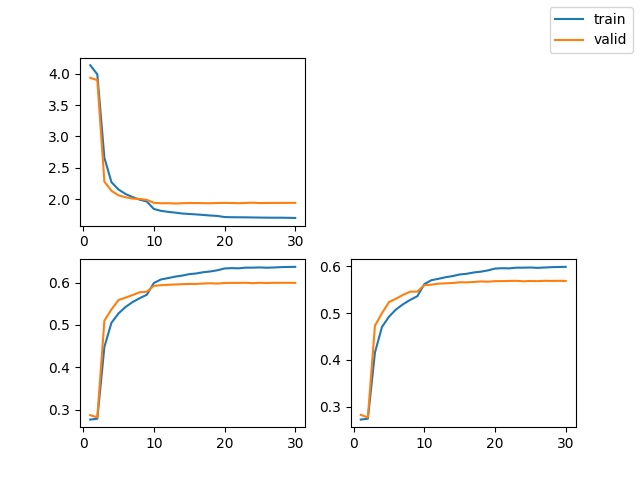
**Technology:**

Tool: Pytorch

Model: VGG19, LSTM

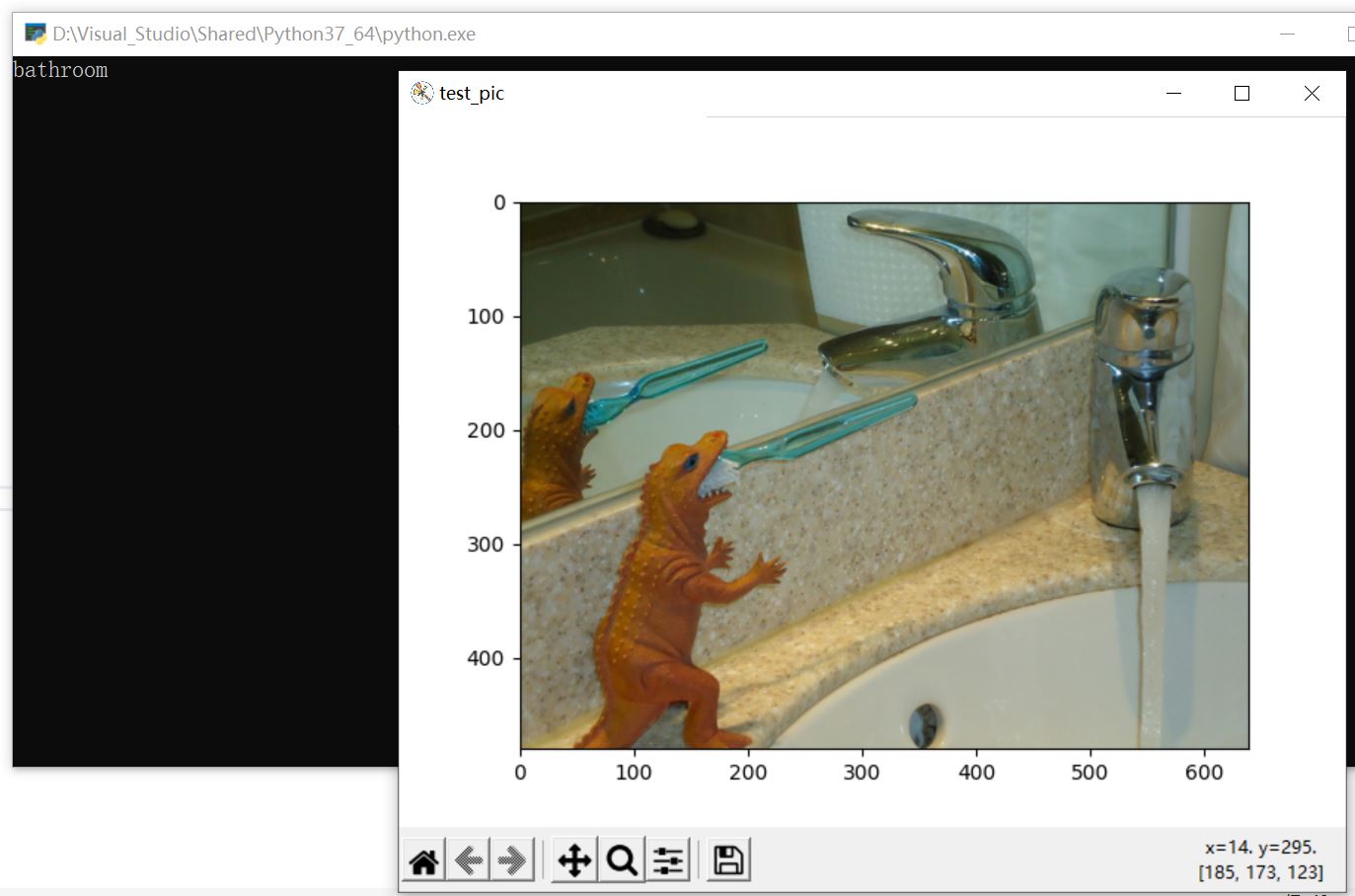
And some necessary functions to train and evaluate.

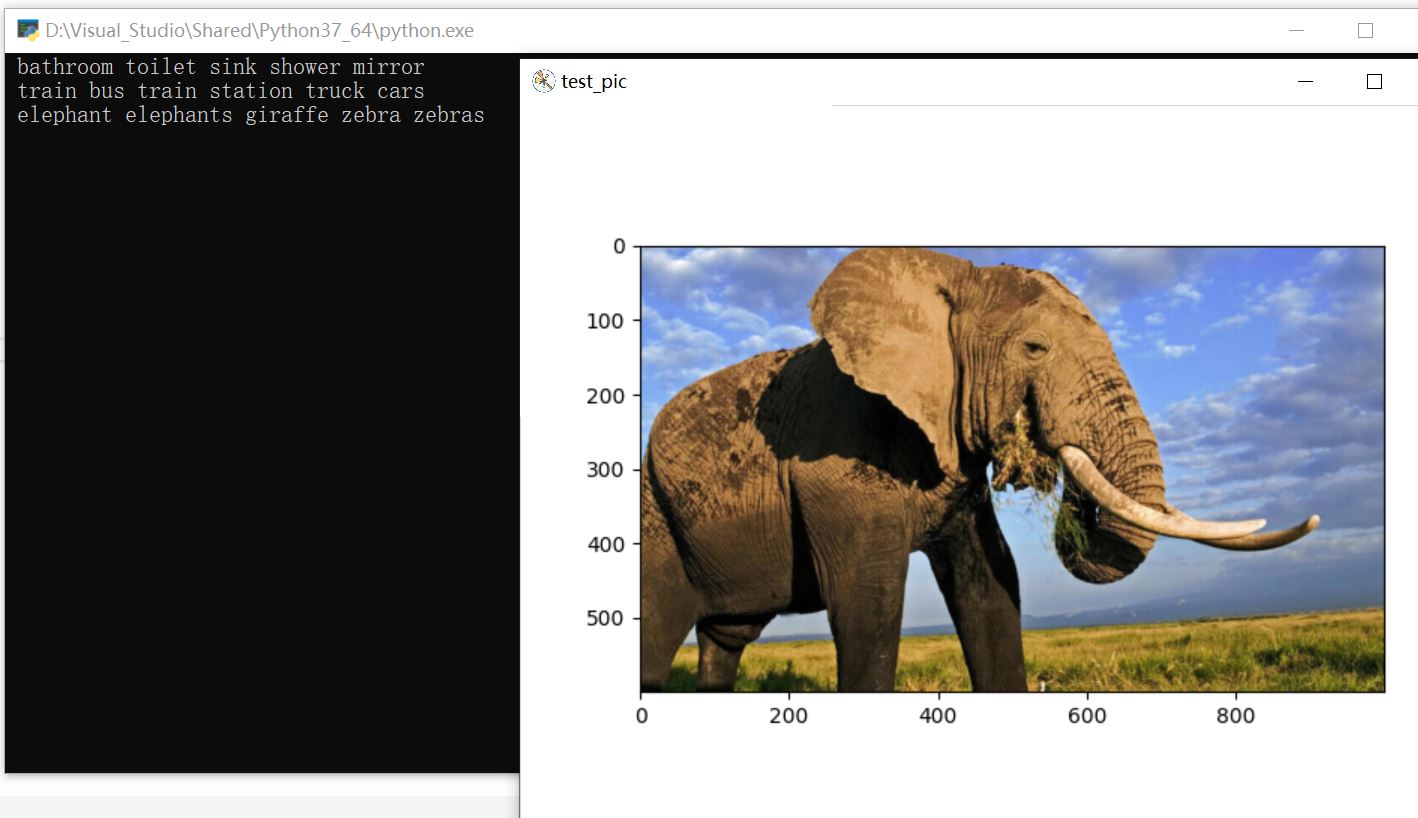
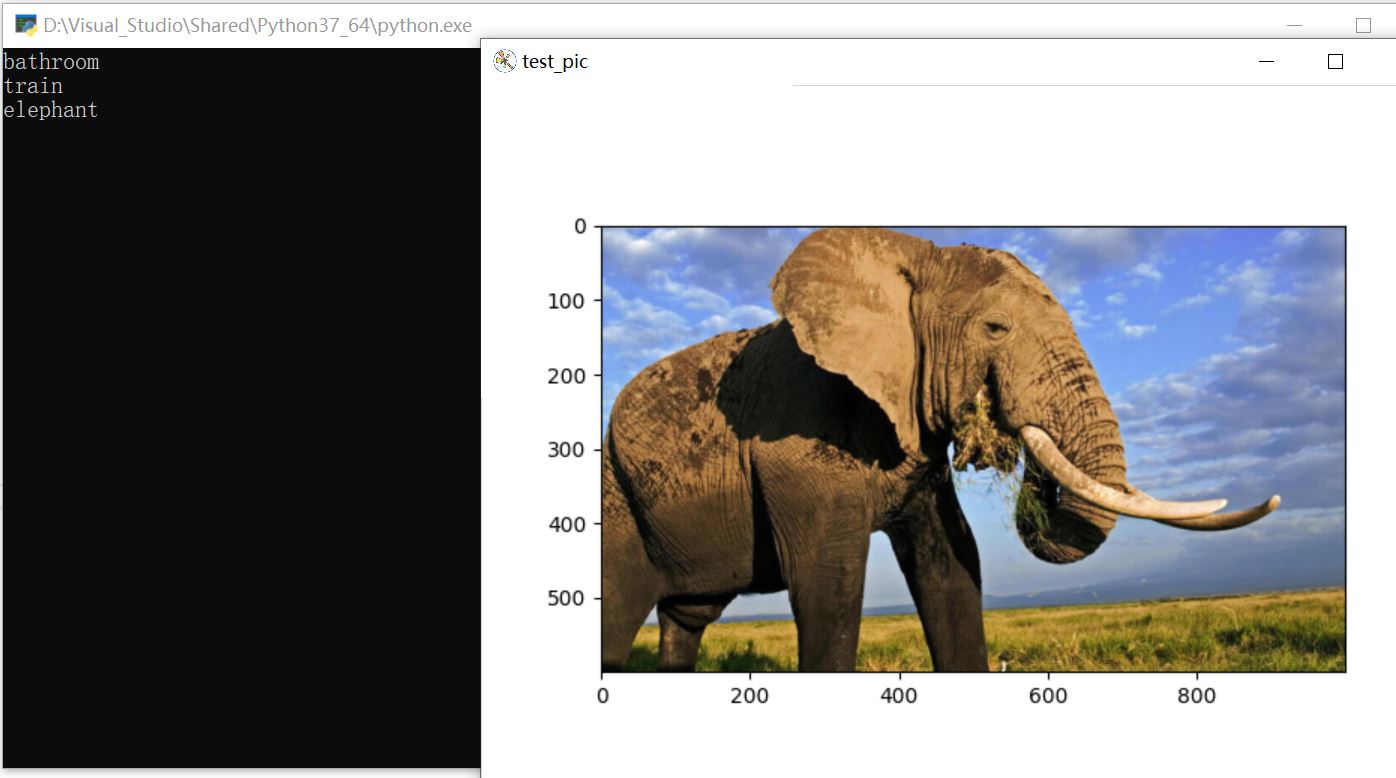
**Our duplicate project output:**



**Use the model to predict:**

The architecture of our code is simple. Firstly, convert the picture and the question to tensor that fit the input size of the model. Then, load the checkpoint file which saved by the open-source project. After that, change the model to evaluate mode and feed the input into the model to get the output. Finally, pick the answer depends on the output possibility of the model.





So, as we mentioned before, the answers always are picked in the top 1000 answers, we are not sure why the author of the open-source project planned to implement like this.

**Future work:**

Convert the model to medical input

Maybe increase the answer field