POLITECNICO DI MILANO

ARTIFICIAL NEURAL NETWORK AND DEEP LEARNING

Visual Question Answering

Third Challenge

Authors

Enrico Gherardi

Ludovico Righi

January 31, 2021

Contents

Chapter 1

Setting Up the Environment

1.1 Prepare the Training

Firstly, we opened the training JSON file and extracted the total number of questions and we tokenized all the used words.
Then, we created our custom DataGenerator class to handle the datasets. Its main methods are:

- __getitem__(self, index): retrieves the i-th batch ([RGBimage0, ...], [input_question0, ...]), [answer0, ...]).
- _generate_X(self, indexes): indexes is the list of indexes to take the values from. This function generates as output the couple [RGBimages, questions].
- _generate_Y(self, indexes): indexes is the list of indexes to take the values from. This function generates the output in categorical form.
- def _load_image(self, img_name, img_w, img_h): loads the img, resizes and normalizes it.

We divided the dataset in 70% for training and 30% for validation. We extracted from the JSON the list of questions, images and answers both for training and for validation with the function readTrainJson. Instead, for the test set we extracted the list of questionids, images and questions with the function readTestJson.
Finally, we extracted from the tokenizer the tokens for the training, validation and test set and we padded them. In this way, we have all the element we need to create the train, val and test generator with the custom DataGenerator class.

1.2 Create the Final CSV

We create an external script to generate the final csv to submit our model. We made this decision to try different checkpoints of the same training. The function is very simple and it is shown below:

```
def create_csv(results, results_dir='./drive/MyDrive'):
    csv_fname = 'results_'
    csv_fname += datetime.now().strftime('%b%d_%H-%M-%S') + '.csv'
```

```
    with open(os.path.join(results_dir, csv_fname), 'w') as f:
        f.write('Id,Category\n')
        for key, value in results.items():
            f.write(key + ',' + str(value) + '\n')

pred = VQA_net.predict(test_generator)
results = {}

for i in range(len(pred)):
    results[test_generator.answers[i]] = np.argmax(pred[i])

create_csv(results)
```

Chapter 2

Model Experiments

The final model multiplies a CNN with a RNN or a Transformer-based net or a BERT and connects it to a FFNN to classify the right answer to corresponding the couple (image, question).

2.1 Networks for the Images (CNN)

We used as a base some famous CNN models (with fine tuning). After this, we added on top some dense layers with batch-normalization, ReLu and drop out. The CNN used are relatively simple because the total model has to fit in the GPU and to reduce the training time.
The net that reached the best performance are:

- VGG19: with this CNN we reached a test accuracy of 55%
- MobileNetV2: this net has only 3,538,984 parameters (compared to the 143,667,240 of the VGG19) and the training with that is much faster. This has permitted to use data argumentation. With this CNN we reached a test accuracy of: 57%

2.2 Networks for the Questions (LSTM, Transformer, BERT)

For this part, we experimented with different models:
The first network that we have implemented is a LSTM-based net, it is formed by:

- Embedding layer
- LSTM layer with return_sequence=True
- Dropout layer
- LSTM layer with return_sequence=False
- Dropout layer
- Dense layer

With this net we reached a test accuracy of 57%.
The second model is a Transformer-based net, it was formed by:

- Token and position Embedding layer
- Transformer Block
- GlobalAveragePooling1D layer
- A series of dense layers and dropout

With this net we reached our best result of 62%.
Finally, we try to use the BERT net with a final test accuracy of 55%.

2.3 Final VQA net

To create this network:

- Multiply the CNN and the LSTM or Transformer-based net or BERT
- Add a dense layer
- Add a batch-normalization, relu and dropout layer
- Add the last classification layer with the number of neurons equals to the number of answer-classes and with the softmax as activation function.

Chapter 3

Final Result

Our best test accuracy is: 62%.
We reached this result with:

- MobileNetV2-based CNN
- Transformer-based net
- Data Argumentation
- Input size = (200x350)
- Loss = Categorical Cross Entropy
- Optimizer = Adam

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
mobilenetv2_1.00_224_input (Inp	[(None, 200, 350, 3)	0	
mobilenetv2_1.00_224 (Functiona	(None, 7, 11, 1280)	2257984	mobilenetv2_1.00_224_input[0][0]
global_average_pooling2d_1 (Glo	(None, 1280)	0	mobilenetv2_1.00_224[0][0]
dense_6 (Dense)	(None, 1024)	1311744	global_average_pooling2d_1[0][0]
batch_normalization_4 (BatchNor	(None, 1024)	4096	dense_6[0][0]
activation_4 (Activation)	(None, 1024)	0	batch_normalization_4[0][0]
dropout_6 (Dropout)	(None, 1024)	0	activation_4[0][0]
dense_7 (Dense)	(None, 512)	524800	dropout_6[0][0]
batch_normalization_5 (BatchNor	(None, 512)	2048	dense_7[0][0]
embedding_1_input (InputLayer)	[(None, 20)]	0	
activation_5 (Activation)	(None, 512)	0	batch_normalization_5[0][0]
embedding_1 (Embedding)	(None, 20, 512)	256000	embedding_1_input[0][0]
dropout_7 (Dropout)	(None, 512)	0	activation_5[0][0]
lstm_2 (LSTM)	(None, 20, 512)	2099200	embedding_1[0][0]
dense_8 (Dense)	(None, 1024)	525312	dropout_7[0][0]
dropout_9 (Dropout)	(None, 20, 512)	0	lstm_2[0][0]
batch_normalization_6 (BatchNor	(None, 1024)	4096	dense_8[0][0]
lstm_3 (LSTM)	(None, 512)	2099200	dropout_9[0][0]
activation_6 (Activation)	(None, 1024)	0	batch_normalization_6[0][0]
dropout_10 (Dropout)	(None, 512)	0	lstm_3[0][0]
dropout_8 (Dropout)	(None, 1024)	0	activation_6[0][0]
dense_9 (Dense)	(None, 1024)	525312	dropout_10[0][0]
multiply_1 (Multiply)	(None, 1024)	0	dropout_8[0][0] dense_9[0][0]
dense_10 (Dense)	(None, 1024)	1049600	multiply_1[0][0]
batch_normalization_7 (BatchNor	(None, 1024)	4096	dense_10[0][0]
activation_7 (Activation)	(None, 1024)	0	batch_normalization_7[0][0]
dropout_11 (Dropout)	(None, 1024)	0	activation_7[0][0]
dense_11 (Dense)	(None, 58)	59450	dropout_11[0][0]

Figure 3.1: CNN-LSTM Model

```
__________________________________________________________________________________________________
mobilenetv2_1.00_224 (Functiona (None, 7, 11, 1280)  2257984     mobilenetv2_1.00_224_input[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_2 (Glo (None, 1280)         0           mobilenetv2_1.00_224[0][0]
__________________________________________________________________________________________________
dense_20 (Dense)                (None, 1024)         1311744     global_average_pooling2d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 1024)         4096        dense_20[0][0]
__________________________________________________________________________________________________
activation_8 (Activation)       (None, 1024)         0           batch_normalization_8[0][0]
__________________________________________________________________________________________________
dropout_20 (Dropout)            (None, 1024)         0           activation_8[0][0]
__________________________________________________________________________________________________
dense_21 (Dense)                (None, 512)          524800      dropout_20[0][0]
__________________________________________________________________________________________________
input_6 (InputLayer)            [(None, 500)]        0
__________________________________________________________________________________________________
batch_normalization_9 (BatchNor (None, 512)          2048        dense_21[0][0]
__________________________________________________________________________________________________
token_and_position_embedding_2  (None, 500, 512)     512000      input_6[0][0]
__________________________________________________________________________________________________
activation_9 (Activation)       (None, 512)          0           batch_normalization_9[0][0]
__________________________________________________________________________________________________
transformer_block_2 (Transforme (None, 500, 512)     5253120     token_and_position_embedding_2[0]
__________________________________________________________________________________________________
dropout_21 (Dropout)            (None, 512)          0           activation_9[0][0]
__________________________________________________________________________________________________
global_average_pooling1d_2 (Glo (None, 512)          0           transformer_block_2[0][0]
__________________________________________________________________________________________________
dense_22 (Dense)                (None, 1024)         525312      dropout_21[0][0]
__________________________________________________________________________________________________
dropout_27 (Dropout)            (None, 512)          0           global_average_pooling1d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_10 (BatchNo (None, 1024)         4096        dense_22[0][0]
__________________________________________________________________________________________________
dense_26 (Dense)                (None, 20)           10260       dropout_27[0][0]
__________________________________________________________________________________________________
activation_10 (Activation)      (None, 1024)         0           batch_normalization_10[0][0]
__________________________________________________________________________________________________
dropout_28 (Dropout)            (None, 20)           0           dense_26[0][0]
__________________________________________________________________________________________________
dropout_22 (Dropout)            (None, 1024)         0           activation_10[0][0]
__________________________________________________________________________________________________
dense_27 (Dense)                (None, 1024)         21504       dropout_28[0][0]
__________________________________________________________________________________________________
multiply_2 (Multiply)           (None, 1024)         0           dropout_22[0][0]
                                                                 dense_27[0][0]
__________________________________________________________________________________________________
dense_28 (Dense)                (None, 1024)         1049600     multiply_2[0][0]
__________________________________________________________________________________________________
batch_normalization_11 (BatchNo (None, 1024)         4096        dense_28[0][0]
__________________________________________________________________________________________________
activation_11 (Activation)      (None, 1024)         0           batch_normalization_11[0][0]
__________________________________________________________________________________________________
dropout_29 (Dropout)            (None, 1024)         0           activation_11[0][0]
__________________________________________________________________________________________________
dense_29 (Dense)                (None, 58)           59450       dropout_29[0][0]
==================================================================================================
Total params: 11,540,110
Trainable params: 11,495,470
Non-trainable params: 44,640
```

Figure 3.2: CNN-Transformer Model

```
________________________________________________________________________________________________
mobilenetv2_1.00_224 (Functiona (None, 7, 11, 1280)  2257984     mobilenetv2_1.00_224_input[0][0]
________________________________________________________________________________________________
global_average_pooling2d_1 (Glo (None, 1280)         0           mobilenetv2_1.00_224[0][0]
________________________________________________________________________________________________
dense_10 (Dense)                (None, 1024)         1311744     global_average_pooling2d_1[0][0]
________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 1024)         4096        dense_10[0][0]
________________________________________________________________________________________________
activation_4 (Activation)       (None, 1024)         0           batch_normalization_4[0][0]
________________________________________________________________________________________________
dropout_10 (Dropout)            (None, 1024)         0           activation_4[0][0]
________________________________________________________________________________________________
dense_11 (Dense)                (None, 512)          524800      dropout_10[0][0]
________________________________________________________________________________________________
input_4 (InputLayer)            [(None, 500)]        0
________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 512)          2048        dense_11[0][0]
________________________________________________________________________________________________
token_and_position_embedding_1  (None, 500, 512)     512000      input_4[0][0]
________________________________________________________________________________________________
activation_5 (Activation)       (None, 512)          0           batch_normalization_5[0][0]
________________________________________________________________________________________________
transformer_block_1 (Transforme (None, 500, 512)     5253120     token_and_position_embedding_1[0]
________________________________________________________________________________________________
dropout_11 (Dropout)            (None, 512)          0           activation_5[0][0]
________________________________________________________________________________________________
global_average_pooling1d_1 (Glo (None, 512)          0           transformer_block_1[0][0]
________________________________________________________________________________________________
dense_12 (Dense)                (None, 1024)         525312      dropout_11[0][0]
________________________________________________________________________________________________
dropout_17 (Dropout)            (None, 512)          0           global_average_pooling1d_1[0][0]
________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 1024)         4096        dense_12[0][0]
________________________________________________________________________________________________
dense_16 (Dense)                (None, 20)           10260       dropout_17[0][0]
________________________________________________________________________________________________
activation_6 (Activation)       (None, 1024)         0           batch_normalization_6[0][0]
________________________________________________________________________________________________
dropout_18 (Dropout)            (None, 20)           0           dense_16[0][0]
________________________________________________________________________________________________
dropout_12 (Dropout)            (None, 1024)         0           activation_6[0][0]
________________________________________________________________________________________________
dense_17 (Dense)                (None, 1024)         21504       dropout_18[0][0]
________________________________________________________________________________________________
multiply_1 (Multiply)           (None, 1024)         0           dropout_12[0][0]
                                                                 dense_17[0][0]
________________________________________________________________________________________________
dense_18 (Dense)                (None, 1024)         1049600     multiply_1[0][0]
________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 1024)         4096        dense_18[0][0]
________________________________________________________________________________________________
activation_7 (Activation)       (None, 1024)         0           batch_normalization_7[0][0]
________________________________________________________________________________________________
dropout_19 (Dropout)            (None, 1024)         0           activation_7[0][0]
________________________________________________________________________________________________
dense_19 (Dense)                (None, 58)           59450       dropout_19[0][0]
================================================================================================
Total params: 11,540,110
Trainable params: 11,495,470
Non-trainable params: 44,640
```

Figure 3.3: CNN-BERT Model