Theoretical Tasks

Image captioning can be regarded as an end-to-end Sequence problem, as it converts images, which are regarded as a sequence of pixels, to a sequence of words. For this purpose, we need to process both the language or statements and the images. We use recurrent neural networks for the language part, and for the image part, we use convolutional neural networks to obtain the feature vectors. Image Captioning

We are dealing with two types of information, a language one and another image one. So, the question arises of how or in what order we should introduce the information into our model. Elaborately speaking, we need a language RNN model to generate a word sequence, so when should we introduce the image data vectors in the language model? A paper by Marc Tanti and Albert Gatt [Comparison of Architectures], Institute of Linguistics and Language Technology, University of Malta covered a comparison study of all the approaches. Image Captioning

You can read the post and Andrej Karpathy's Architecture.
Then, you should be able to complete the following tasks.

Task 3.1

Answer the following questions:

Task 3.1.1 - Explain the pros and cons of utilising Concatenation for combining embeddings

Task 3.1.2 - Explain the pros and cons of utilising Addition for combining embeddings

Task 3.1.3 - Explain the pros and cons of utilising Multiplication for combining embeddings

Task 3.1.4 - Explain the pros and cons of utilising Attention for combining embeddings

Task 3.1.5 - Explain the pros and cons of utilising Difference for combining embeddings

Task 3.2 Practical Assessment

Task Description

This exercise involves implementing an image-captioning network. You can use any Deep Learning Framework of your choice. We will provide you with the general steps, and you will implement them as you see fit.

You can choose training data for the images and captions. You will also choose how to combine the embeddings and answer the questions at the end.

As a base, you can use this example Image captioning in Pytorch.

The basic high-level steps to follow are:

Image Feature Extraction:

The first step in an image captioning network is to extract the features from the image. This is usually done using a Convolutional Neural Network (CNN). The CNN takes the image as input and outputs a feature vector that represents the image's content. This feature vector serves as the input to the next part of the network.

Example: Using Keras.

```python
# Extract features from the image
base_model = VGG16(weights='imagenet', include_top=False)
image_features = base_model.predict(img_array)
image_features = image_features.reshape(image_features.shape[0], -1)
```

Sequence Model for Language Processing:

The next part of the network is a sequence model, usually some variant of LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit). This part of the network is responsible for generating the caption. It takes the feature vector from the CNN as input and generates a sequence of words as output.

```python
# Two networks, one for images and one for captions
image_input = Input(shape=(image_features.shape[1],))
image_embedding = Dense(256, activation='relu')(image_input)

caption_input = Input(shape=(max_length,))
caption_embedding = Embedding(input_dim=vocab_size, output_dim=256, input_length=max_length
caption_embedding = LSTM(256)(caption_embedding)
```

Combining Image and Text Data:

The feature vector from the CNN and the output from the RNN need to be combined to generate the final caption. There are several ways to do this, such as Concatenation, Addition, attention mechanisms, etc.

```python

combined_embeddings = you_choose_how([image_embedding, caption_embedding])
 # And you need to create the model, for example:
 Model(inputs=[image_input, caption_input], outputs=output)
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.001), metrics=['accuracy
```

Where output is a standard FCC:

```python
# For N classes, we are using Softmax
output = Dense(vocab_size, activation='softmax')
```

Training the Network:

You will need a dataset of images and their corresponding captions. Then, do a standard training process.

In Keras:

```python
model.fit([image_features, X_captions], y, epochs=10, batch_size=1)
```

Generating Captions:

When the process is completed, it should be able to generate captions on the test set; please show them.

Evaluation:

The quality of the generated captions is typically evaluated using metrics like BLEU, METEOR, ROUGE, or CIDEr, which compare the generated caption to a set of reference captions.

Note: This section is optional as long as you can see the loss is decreasing; your model will not be penalized on this.