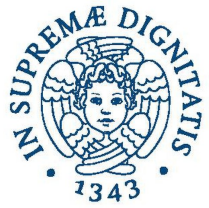


UNIVERSITÀ DI PISA

A first approach to Image Captioning with Neural Networks

Computational Intelligence and Deep Learning
Final project
Prof. Beatrice Lazzerini
Prof. Alessandro Renda

Francesco Campilongo

Msc. in Artificial Intelligence and Data Engineering

Contents

1 Introduction **2**

2 State of the art **2**

3 Dataset **3**
3.1 Train set . . . 3
3.2 Test set . . . 3

4 Convolution Neural Network **4**
4.1 Visual Geometric Group - VGGNet . . . 4
4.2 InceptionV3 . . . 5

5 Recurrent Neural Network **6**
5.1 Long Short - Term Memory Networks . . . 6

6 Work-flow **8**
6.1 Image Processing . . . 8
6.1.1 VGGnets . . . 8
6.1.2 InceptionV3 . . . 8
6.1.3 Output . . . 9
6.2 Text data processing . . . 9
6.2.1 Pre-processing . . . 9
6.2.2 Captions organizations . . . 9
6.2.3 Tokenization . . . 9
6.3 Deep Learning Model . . . 10
6.3.1 Defining the Model . . . 10
6.3.2 Fitting the Model . . . 13
6.3.3 Caption Prediction . . . 13
6.3.4 Evaluate the Model . . . 13

7 Conclusion **15**

8 Instructions **18**
8.1 Directories . . . 19
8.2 Jupyter Notebooks . . . 19

9 Reference **21**

1 Introduction

The smartphone almost saturated market has given to nearly every one a personal camera. The number of photos taken in an year is approximately 14 Billion, and a very good percentage of this number has no caption.
Image captioning is the process of generating a textual description of an image by using both natural language processing and computer vision applications. More specifically the Convolution Neural Network (CNN) for images features extractions for Computer Vision concern and the Recurrent Neural Network (RNN) precisely Long Short-Term Memory Network (LSTM) for Natural Language Processing.
This type of project has a lot of possible application, not just for auto captioning an old album but also for accessibility necessity or image searching using a textual search engine.

2 State of the art

There are different common techniques, like Attention Mechanism, Novel Objects paradigm, Semantics, but the most promising one is the Encoder-Decoder one, which is the same used in this project and that will be discussed later. It is interesting to take a look at the architectural pipeline for these type of tasks. For feature extraction, they have used CNN encoder structure with pre-trained models, ResNet152, AlexNet, VGG19-Net, DenseNet, SqueezeNet, to experiment with the performance on feature extractions. The choice was ResNet152 which performed slightly better. For the language model, LSTM and GRU recurrent neural networks are utilized by attention mechanism and teacher forcing technique. As a first step, the word embedding layer is used for word representation for sentence length captions.
Their best model according to BLEU Score and METEOR (the two different metric utilized) were the model with ResNet152 and GRU (Gated Recurrent Unit). This proves that the best approach is CNN followed by a RNN the same approach utilized here.
The link to this article is present into the reference section to the item "State of art".

3 Dataset

The selected dataset has 82783 general images and it is found on MSCOCO site. The 2014 dataset version.
Of course there is the need to have also the captions dataset for the train set, the latter was been downloaded directly from the MSCOCO site. It was organized in a json file, with a wide range of information available, like dimension, date, image id and of course the caption per every images, from this chaotic json a more straight forward one was been created, with the image name and the caption.

3.1 Train set

Starting from the upper mentioned dataset, 10000 random images are selected and utilized to train the RNN (a more detailed look later). From the JSON file was been retrieved the right caption for the right image.

3.2 Test set

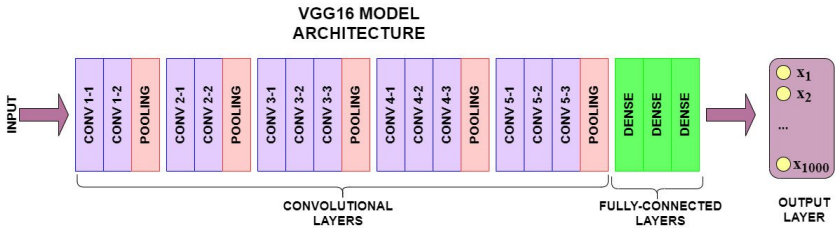
Just 1000 random images are selected from the original dataset to became the test set, with these images, every appropriate caption were selected from the original JSON file.

4 Convolution Neural Network

Neural networks that use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. They are specifically designed to process pixel data and are used in image recognition and processing.
The chosen Convolution Neural Network architectures for extracting features are VGG and InceptionV3.
In this project the CNNs are used to extract features from the images in order to get a representation of the image and use it as input for the RNN.

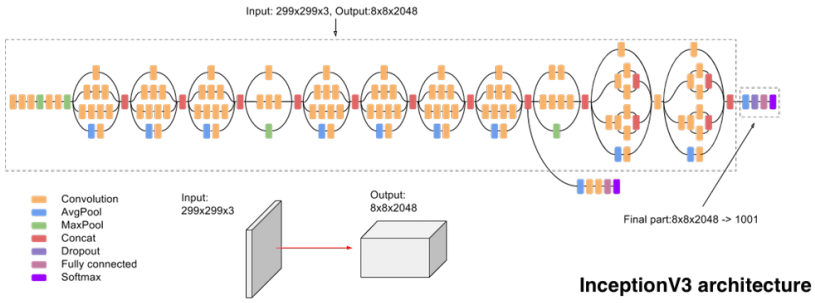
4.1 Visual Geometric Group - VGGNet

These Networks are a very deep. This approach is based on incorporate multiple non-linear rectification layers (the convolution layers seen in the image below) instead of a single rectification layer, this is done to help decreasing the number of parameters while keeping the performance. For example, using 2 layers of 3x3 filter is equal to 1 layer of 5x5 filter but using fewer parameters. The number of a parameter is reduced by 28 percent: 2x3x3=18 1x5x5=25.



The Convolution Neural Networks used in this project is the VGG16 and the VGG19, the difference between these two is that for VGG16 are expected 16 convolution layers instead for VGG19 there are 19 convolution layers.

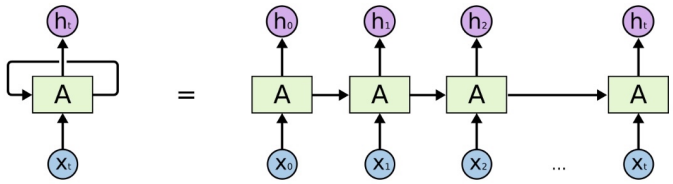
4.2 InceptionV3



Multiple deep layers of convolutions were used in a model it resulted in the overfitting of the data. To avoid this from happening the inception V1, first version of this architecture, presented in 2014, uses the idea of using multiple filters of different sizes on the same level. Thus in the inception models instead of having deep layers, there are parallel layers thus making the model wider rather than making it deeper. There are Convolution layers, 1x1, 3x3 and 5x5, and Pooling layers. The former are used to capture information directly from the image pixels using a wider or less wide filter (kernel), the latter is used to reduce the dimensions of the feature map, through max pooling and average pooling.

5 Recurrent Neural Network

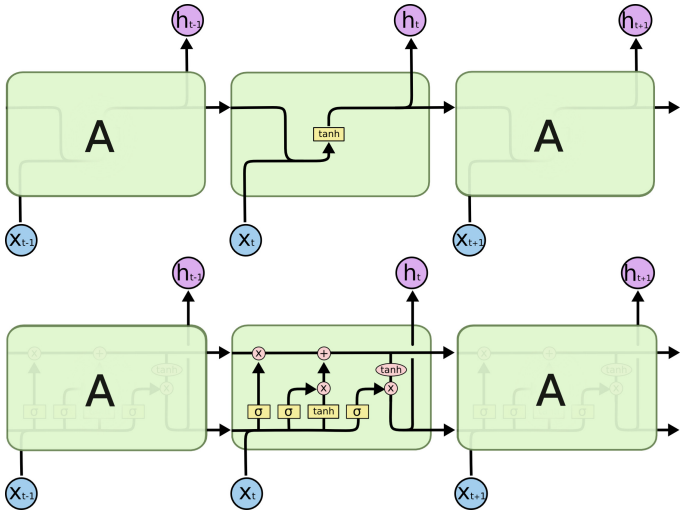
Recurrent Neural Networks are networks with loops in them, allowing information to persist. They can be thought of as multiple copies of the same network, each passing a message to a successor.



An unrolled recurrent neural network.

5.1 Long Short - Term Memory Networks

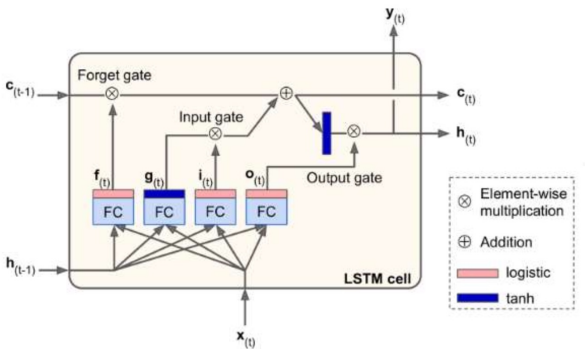
The most successful RNN are the Long Short - Term Memory networks, the most important difference with the "basic" RNN is that LSTM networks are capable of learning long term dependencies.



The picture above shows a repeating module for "basic" RNN (upper side), and a repeating module for a LSTM network (bottom side).

The picture below highlights the architecture, starting from the cell state C(t-1) — C(t), runs down for the entire chain with only some minor linear

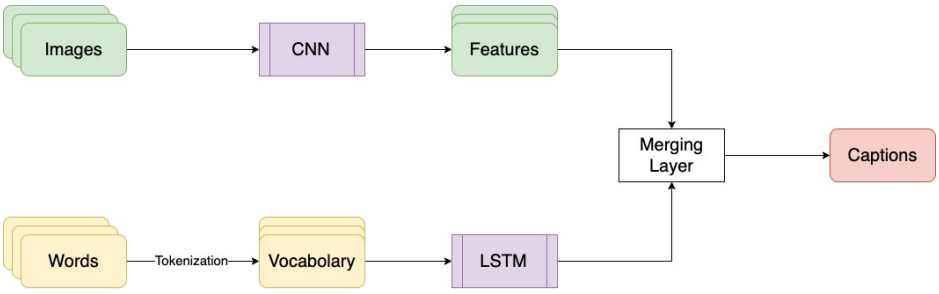
interactions. The informations removed/added to the cell state are regulated by structures called gates, composed out of a sigmoid neural net layer and a point-wise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through, with zero which means nothing let through and one let through everything.
During the training, the cell state passes from the forget gate, and so forgets something from the previous state, this gave the opportunity to learn something new after the input gate, this, without any others transformations became the output, simultaneously this goes to a tanh function so the results is filtered by the output gate, producing the short-term state, y(t) and h(t) in the picture below.



The picture above better emphasizes the different gates of the LSTM architecture.

6 Work-flow

The image below describe the passages and the structure utilized to obtain a caption for the image as input. Lets analyze it step by step.



6.1 Image Processing

In this section there are described the passages and the choices done for the features extraction. Two different CNN have been used in order to better understand what difference there are between those two different approaches. Earlier in this paper had been described these approaches in a more detailed manner, here will be discussed the practical points.

6.1.1 VGGnets

The VGGnets used were VGG16 and VGG19 keras provide a pre-trained model for both the versions. The images taken as input of the VGGnets must have a predefined dimension, which are 224 x 224 x 3 (height x wide x channels, RGB), this transformation is done using the keras VGG functions like image reshape. The output for this kind of CNN is a features vector of 4096 elements, one features vector per images, quite a lot of data.

6.1.2 InceptionV3

The InceptionV3 architecture used for extracting feature from images was still provided by keras, which gave a pre-trained model.Also the images taken as input of the InceptionV3 net must have a predefined dimension 299 x 299 x 3 (height x wide x channels, RGB), also in this case the keras module provide functions to load and reshape the images in order to be taken as input. The output this time is a feature vector (one per images) of 2048 elements, basically the half of the previous approaches.

6.1.3 Output

The output of this "module" is basically a file of tuple with a the image key (name without the extension) and the features vector for each image; Through the python module Pickle the features vectors are already been calculated, saved and available in the project directory "features".

6.2 Text data processing

In this section will be discussed the operations done from a raw json file with annotations and other pictures informations to get a clean captions file.

6.2.1 Pre-processing

As mention before the JSON file is filled with informations, informations that are not important for the purpose of this project, so most of the text line in this file will be skipped. The annotations section of this JSON file will be examined and organized.

6.2.2 Captions organizations

The output for this section is a json file characterized from a image_id and a single caption per image, a similar organization already seen before, for the Image Processing module. A cleaned organized file is already present in the project directory "captions"

6.2.3 Tokenization

Now that there is a clean captions file, the takenization can take place.

- Converts all the word into lowercase.
- Remove all the punctuation.
- Remove all words that are one character or less in length (e.g. 'a').
- Remove all words with numbers in them.

After this operations it is possible to acquire the vocabolary dimension, which for the training set is 5114, the number of words into the vocabulary, the vocabulary creation has been done with the Tokenizer function provided by

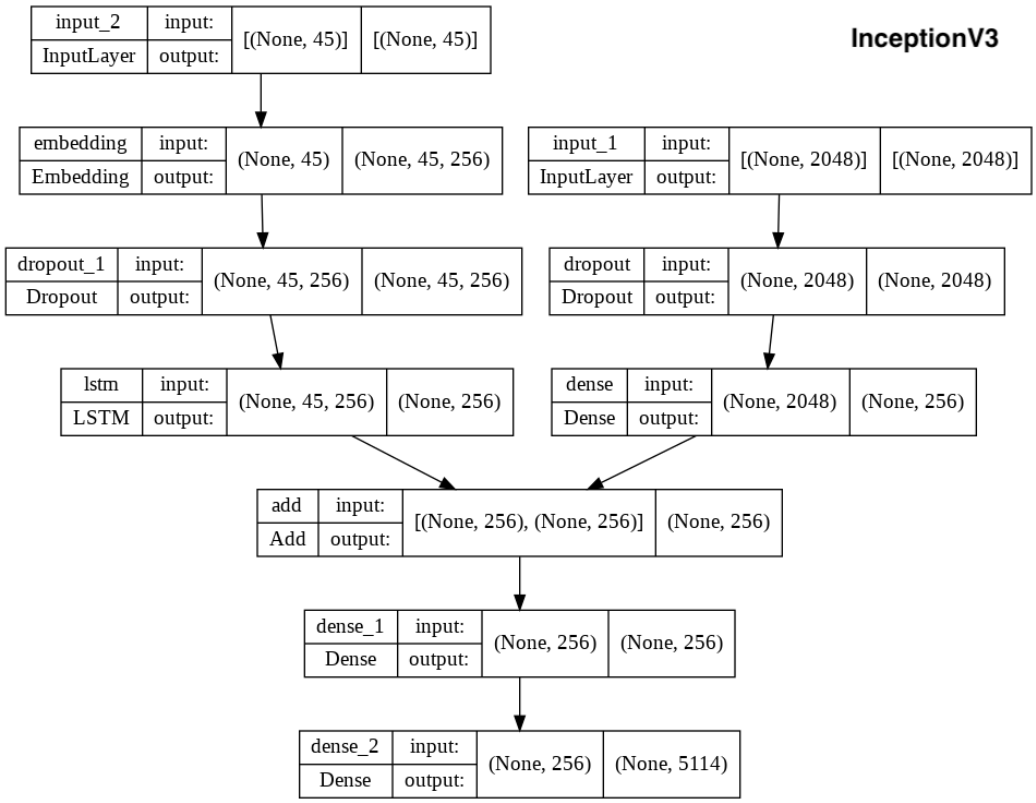
the keras processing text module. Another important dimension which can be extracted here is the max length of a single caption, in this case 45, these informations are used to create the model.

6.3 Deep Learning Model

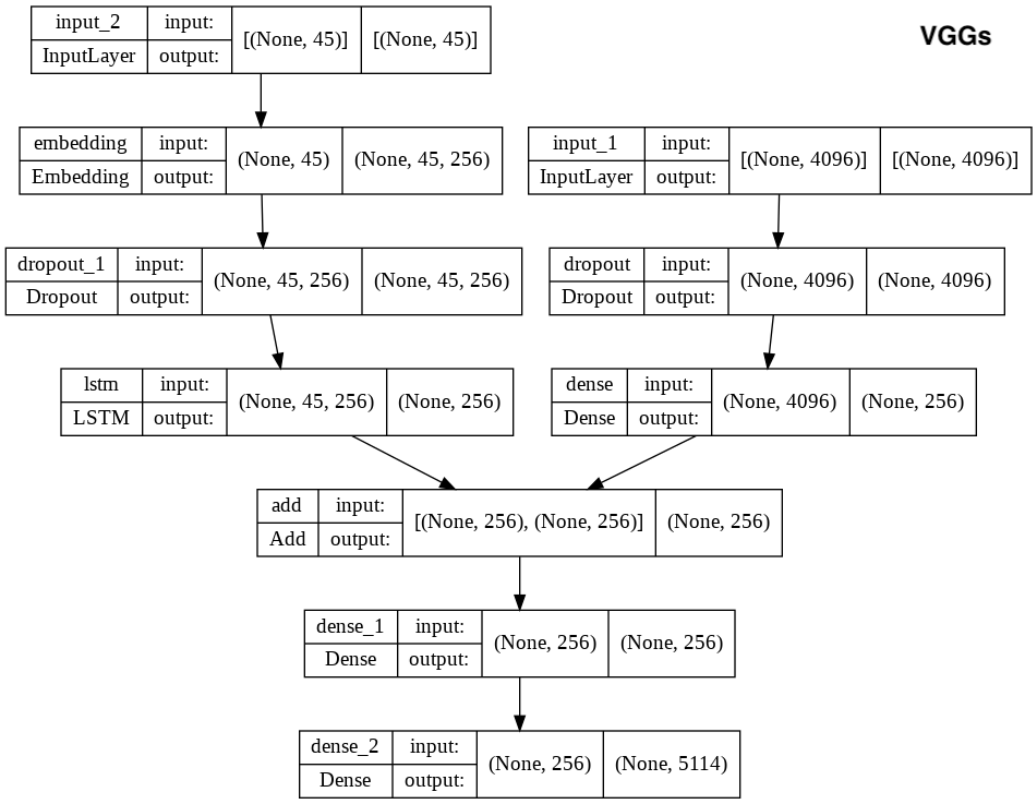
After loading up all the data, both images features and captions there are some operations to perform in order to get some results.

6.3.1 Defining the Model

There are several model to generate captions from a set of images, the model chosen here is the Encoder - Decoder; the former uses a neural network (Convolution Neural Network in this case) to encode the content of a photo using a fixed length vector (the features vector), the latter reads the encoded images and generates a textual description output.
Basically after the features extraction, there is a sequence processor that is a word embedding layer that manages the text input (the captions) followed by a LSTM recurrent neural network layer, these operations are basically the encoder part of the model. Finally the decoder part is a merge layer characterized by a dense layer that actually make the final prediction.



The image above shows the entire model with the InceptionV3 features extractor.



The image above shows the entire model with the VGGnets features extractor. The difference are basically none, only the dimension for the input of the images features since the number of features extracted with vgg are 4096 instead the number of features extracted with inceptionV3 are 2048. It is important to highlights the presence of a dropout layer for both part of the net that manages the images features and the text data, in order to prevent the overfitting, after the dropout a dense layer was utilized to produce a 256 element representation of the image, 256 are also the number of element of the output of the LSTM layer, those two are merged into the decoder part of the net using an addition operation, this is then merged into a dense layer (relu), the output goes into an another dense layer with softmax activation function that takes care of the prediction. The loss is calcolated using the categorical crossentropy the optimizer utilized was Adam.

6.3.2 Fitting the Model

Defined the model, it is now possible to fit the model with the training set. The models was trained for 20 epochs reaching a loss of 2.02 for the model with inceptionV3 features and 1.83 for the model with features extracted with VGGs. Using Google Colab and the GPU runtime this operation reaches and passes the 10 minutes of execution time. A data generator approach was used in order to fed the network with the right amount of data instead of risking to saturate the memory available, the approach utilized is more memory optimized, using this expedient the data is better organized, every images with its caption, is well grouped in this batch of data required to the network to fit the model, without fitting the memory.

6.3.3 Caption Prediction

In order to get the predicted caption, the known captions are written between a "startseq" and "endseq" to signal the beginning and the end of a single caption, this is needed for every caption prediction, since once passed the startseq token, this is a signal to start generating new words, exploiting the model to predict the new ward until the endseq token or the max length of caption is reached.

6.3.4 Evaluate the Model

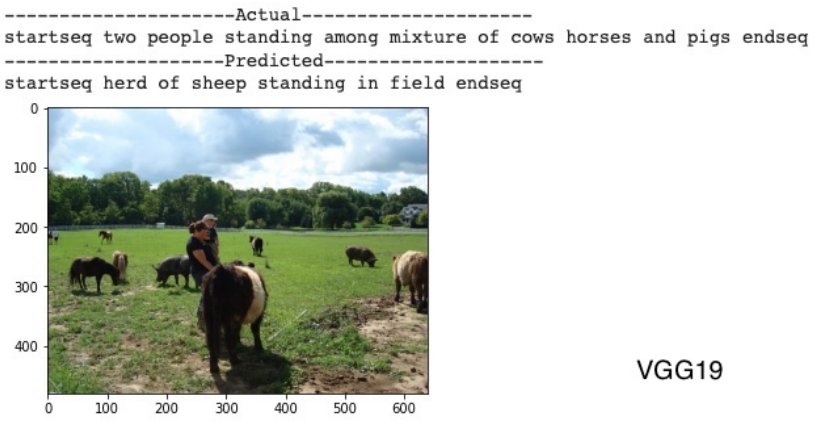
A test set is used to evaluate the model, all the images in the test set have a caption from the previously discussed JSON file, the original caption and one predicted by the model, the metric used for this type of evaluations are multiple, the one chosen in this project is the BLEU Score, based on precision, mostly used to evaluate the translated text against more reference translations, but it can be used for different Natural Language Processing task, the python library which provide this score calculation is NLTK. This score gave a perfect match of 1.0 when the two phrases are identical and a perfect mismatch of 0.0 when every sentence words are different from each other. Here the function utilized to calculate the Bleu Score is corpus_bleu() which gave the possibility to evaluate more reference sentences for a single predicted sentence. More then one version of BLEU Score was calculated by just modifying the weights, which basically are necessary to describe what kind of N-Gram is taken in consideration, with BLEU-1 is considered the 1-Gram, considering one word at time, for BLEU-2 is considered the 2-Gram,

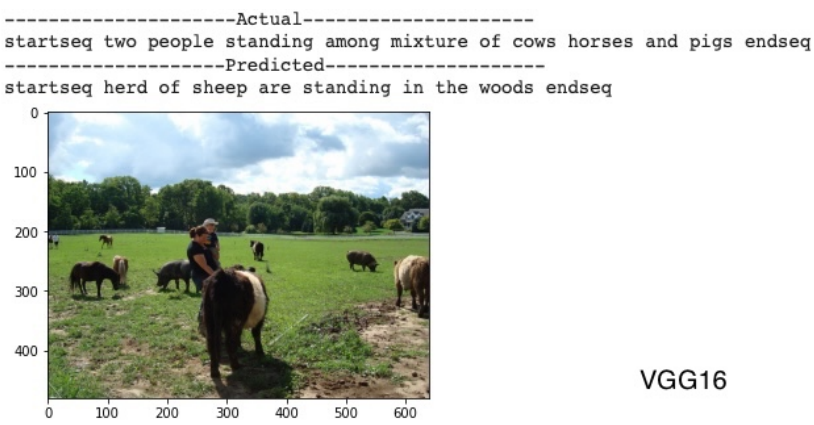
considering two words at time and so on. These versions are useful to better understand the type of prediction obtained.

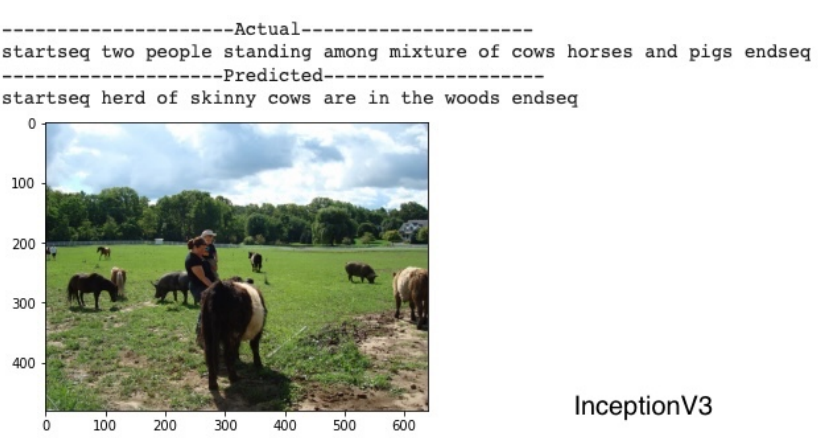
7 Conclusion

	InceptionV3	**VGG16**	**VGG19**
BLEU-1	0.350733	0.337931	0.340216
BLEU-2	0.144897	0.134527	0.133923
BLEU-3	0.093793	0.083037	0.083836
BLEU-4	0.037027	0.031675	0.031644

In the table above there are all the BLEU Score calculated for the various version of the model. The InceptionV3 is the best performer even with higher loss and lower number of elements in the features value, in fact the MB for the features file are actually half the MB for the same files of the VGG models. This because the inceptionV3 architecture tends to extract better quality features respect to VGGnets, since with the inception layers, basically filters of 1x1 dimension can extract more information from an image. The VGGnets works very well with simpler images, the chosen dataset has no simple images but actually very random images with very random contexts and subjects.
The values of this BLEU Score are pretty low, this is also because for every images the number of captions available where just one, with an higher number of captions per images led to a better and higher number of vocables in the vocabulary, this could had let a better and wider choice for the word prediction and also an higher connection between image features and text data.
For the VGGs networks, the results are quite similar, slightly better results for the VGG19 were expected since the three more convolution layers, this is not always true or at least not for this dataset and not by too much. This means that for this type of work with this type of dataset using VGG16 and using VGG19 gave very similar results and so for the three less convolution layers the VGG16 is a better choice between these two.
Now lets talk about actual results, how about the predicted captions?





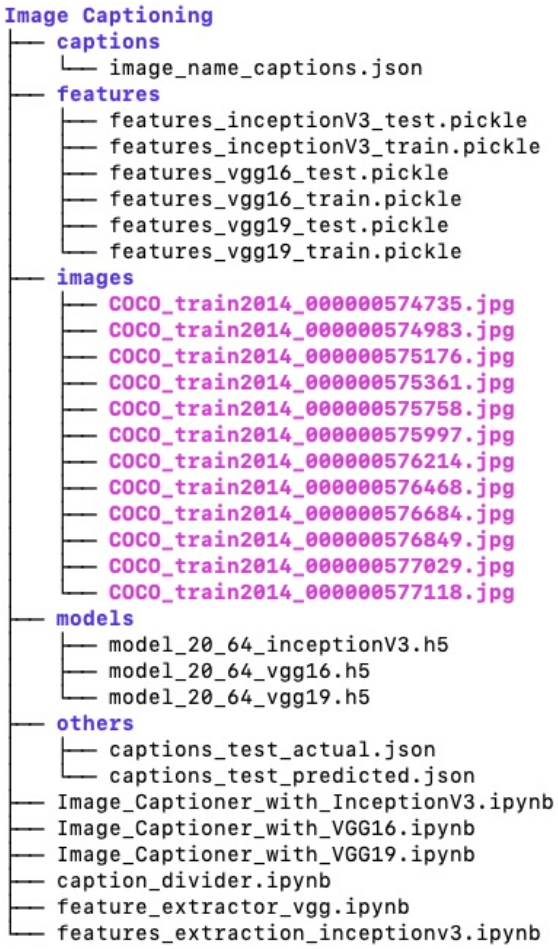


The pictures above shows the images present into the test set, the actual caption and the predicted one for the tree different model analyzed. It is

possible to quite discriminate from just looking the image and the predicted caption what type of features extraction was used in the picture. For example in the inceptionV3 caption there is an adjectives that in the other two captions is not present "skinny". Anyway I believe that all these captions are quite good and could pass the Turing test, even if the BLEU Score were not so great, a new proof that try to evaluate Image captioning isn't easy.
As the state of art section mentioned, the best approach today for image captioning is the combination of CNN and RNN, there are some optimization that could led to a better captions like a better vocabulary without misspelling words or using others features extractors, as mentioned on the state of art the ResNet152 gave interesting results, or trying to fine tuning the model for a better optimization and so better results.

8 Instructions

The project directory structure is organized as the image below



8.1 Directories

As the images shows there are five sub directory into the project directory lets analyze them:

- captions: containing the cleaned JSON file with image_id and caption.
- features: containing all the features extracted for the different models.
- images: containing the sample images used for trying the model.
- models: containing all the models already compiled and fitted in order to save time.
- others: containing extra stuff.

8.2 Jupyter Notebooks

There are six different notebook, three using colab the three main ones, and three that are auxiliaries to the colab notebook. More in specific:

- caption_diveder: utilized to clean all the captions of the dataset and obtain an ordered file saved into captions directory.
- features_extractor_vgg: utilized for extracting features with both VGG16 and VGG19, features extracted for both the test set and the dataset, available into the features directory.
- features_extractor_inceptionv3: utilized for extracting features with inceptionV3, features extracted for both the test set and the dataset, available into the features directory.
- Image_Captioner_with_InceptionV3 : code to obtain predicted captions with image features extracted with InceptionV3.
- Image_Captioner_with_VGG16 : code to obtain predicted captions with image features extracted with VGG16.
- Image_Captioner_with_VGG19 : code to obtain predicted captions with image features extracted with VGG19.

The code for the three colab notebook is a little redundant, but was written this way for clarity sake.
The three colab notebook works in the same way, just run all the cell in the notebook and make sure to enable the GPU hardware acceleration runtime. There are some cell all commented since for the normal evolution of the code could be useless or a little problematic, like the cell for saving/loading the model, compromise all the right models, anyway in order to skip the model fitting run and execution time the load module cell can be uncommented.
To try the model with different images, it is a bit tricky, the images that can be utilized are in the directory "images", and in order to utilize them, copy the name, with the file extension, from one of the images in the images directory and past replacing the already present image name in the last cell of the colab notebook based on what type of model to use.

9 Reference

- State of art - towardsdatascience.com/image-captioning-by-translational-visual-to-language-models-d728bced41c3
- Images dataset - cocodataset.org/
- Caption dataset - cocodataset.org/
- InceptionV3 - iq.opengenus.org/inception-v3-model-architecture/
- VGG - becominghuman.ai/what-is-the-vgg-neural-network-a590caa72643
- RNN/LSTM - colah.github.io/posts/2015-08-Understanding-LSTMs/
- Project Ispiration - machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/