In [6]:

```
%%html
<style>
table {display: block;}
td {
  font-size: 18px
}
.rendered_html { font-size: 28px; }
*{ line-height: 200%; }
</style>
```

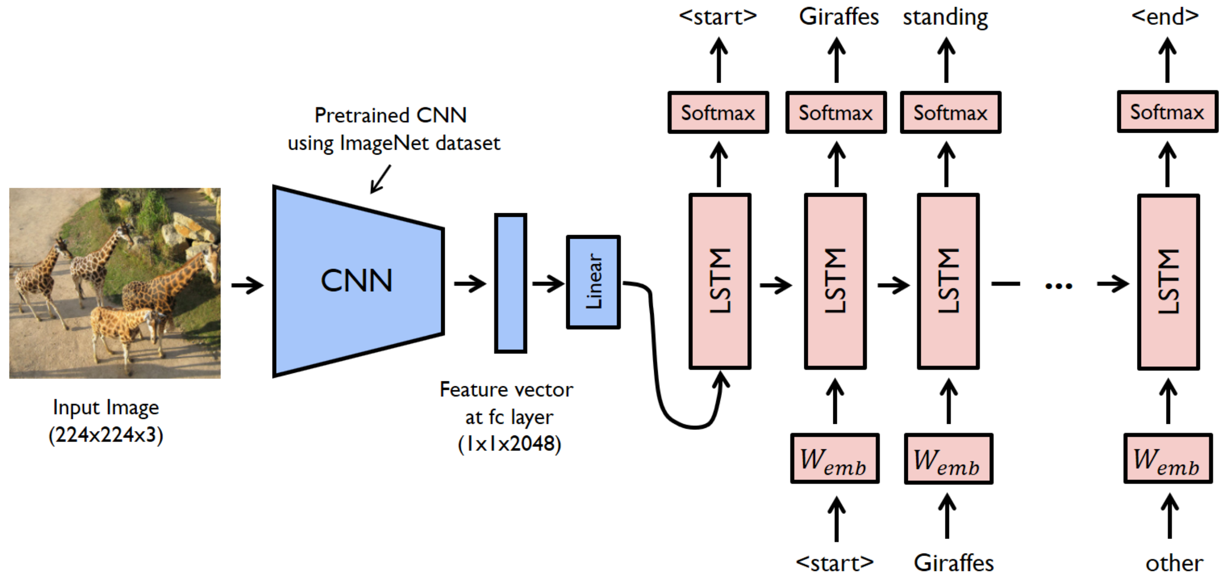
Welcome to the Natural Language Processing and the Web WS2022/23

Practical Class 7: Image Capitoning

Adapted from the practice class Xintong Wang delivered last year

Introduction:

Image captioning aims to convert a given input image into a natural language description. The encoder-decoder framework is widely used for this task. The image encoder is a convolutional neural network (CNN). The decoder is a long short-term memory (LSTM) network.



Training phase

For the encoder part, the pretrained CNN extracts the feature vector from a given input image. The feature vector is linearly transformed to have the same dimension as the input dimension of the LSTM network. For the decoder part, source and target texts are predefined. For example, if the image description is `Giraffes standing next to each other`, the source sequence is a list containing [`<start>`, `Giraffes`, `standing`, `next`, `to`, `each`, `other`] and the target sequence is a list containing [`Giraffes`, `standing`, `next`, `to`, `each`, `other`, `<end>`]. Using these source and target sequences and the feature vector, the LSTM decoder is trained as a language model conditioned on the feature vector.

Testing phase

In the test phase, the encoder part is almost same as the training phase. For the decoder part, there is a significant difference between the training phase and the test phase. In the test phase, the LSTM decoder can't see the image description. To deal with this problem, the LSTM decoder feeds back the previosly generated word to the next input.!!!

Inference phase

In this phase, users could input any images and get the caption for each image using the model we already trained.

Goal for this practice class:

1. Understand the standard process to implement image captioning model.

2. Understand the pipline for deep learning.
3. Understand the difference among training, testing, and inference.

Assignment for this class will be one question (5 scores) and two extra coding parts (10 scores).

GPU and Dataset Preparation

GPU Preparation

Colab provides you 12 hours GPU usage, 12G.

First of all, make sure this tutorial and your assignment is running on GPU environment. Otherwise, it would be pretty slow to train you model.

If your GPU environment is ready to use, use command `!nvidia-smi` to monitor as following.

```
!nvidia-smi
```

In [2]:

```
1 !nvidia-smi
```

```
Sat Nov 20 23:16:13 2021
+-----------------------------------------------------------------------
--------+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 1
1.4     |
|-------------------------------+----------------------+---------------
--------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Unco
rr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Com
pute M. |
|                               |                      |
MIG M. |
|===============================+======================+===============
========|
|   0  NVIDIA GeForce ...  On   | 00000000:04:00.0 Off |
N/A |
| 27%   25C    P8    18W / 250W |   4920MiB / 11019MiB |      0%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+---------------
--------+
|   1  NVIDIA GeForce ...  On   | 00000000:05:00.0 Off |
N/A |
| 59%   76C    P2   226W / 250W |   9909MiB / 11178MiB |     98%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+---------------
--------+
|   2  NVIDIA GeForce ...  On   | 00000000:08:00.0 Off |
N/A |
| 27%   27C    P8     1W / 250W |   6968MiB / 11019MiB |      0%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+---------------
--------+
|   3  NVIDIA GeForce ...  On   | 00000000:09:00.0 Off |
N/A |
| 55%   71C    P2   264W / 250W |   9577MiB / 11178MiB |     66%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+---------------
--------+
|   4  NVIDIA GeForce ...  On   | 00000000:84:00.0 Off |
N/A |
| 29%   25C    P8     8W / 250W |      1MiB / 11178MiB |      0%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+---------------
--------+
|   5  NVIDIA TITAN Xp     On   | 00000000:85:00.0 Off |
```

```
N/A |
| 23%   19C    P8     8W / 250W |    787MiB / 12196MiB |      0%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+--------------
--------+
|   6  NVIDIA GeForce ...  On   | 00000000:88:00.0 Off |
N/A |
| 29%   19C    P8     8W / 250W |    775MiB / 11178MiB |      0%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+--------------
--------+
|   7  NVIDIA GeForce ...  On   | 00000000:89:00.0 Off |
N/A |
| 42%   79C    P2   246W / 250W |   7140MiB / 11019MiB |     96%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+--------------
--------+

+---------------------------------------------------------------------
--------+
| Processes:
|
|  GPU   GI   CI        PID   Type   Process name                  GPU
Memory |
|        ID   ID                                                   Usa
ge      |
|=====================================================================
========|
|    1   N/A  N/A    381721      C   python
9905MiB |
|    2   N/A  N/A    990296      C   /usr/bin/python3
6965MiB |
|    3   N/A  N/A   1079315      C   python3
9573MiB |
|    5   N/A  N/A    771734      C   ...cal/miniconda3/bin/python
783MiB |
|    6   N/A  N/A   1148442      C   ...cal/miniconda3/bin/python
771MiB |
|    7   N/A  N/A   2822267      C   python3
7137MiB |
+---------------------------------------------------------------------
--------+
```

In this practical class, we will use PyTorch Framework.

Make sure PyTorch is installed in your environment. To install the appropriate PyTorch, check cuda version first.

```
# !nvcc --version
```

OR in your local linux machine as:

```
!cat /usr/local/cuda/version.txt
```

In [ ]:

```
!cat /usr/local/cuda/version.txt
```

This result tells us that cuda that is running on our server is version 11

But please ensure to check it yourself in your Google Colab env. Because your running environment may different from where the notebook is running here.

[PyTorch Install Link: (https://pytorch.org/get-started/locally/)](https://pytorch.org/get-started/locally/)

If your environment is the same with this notebook, then the install command will be:

```
!pip install torch==1.10.0+cu111
 torchvision==0.11.1+cu111 torcha
udio==0.10.0+cu111 -f https://dow
nload.pytorch.org/whl/cu111/torch
_stable.html
```

Sidenote: If you want to use conda package management, you should install miniconda or anaconda first. Otherwise, you should use pip.

In [ ]:

```
!pip install torch==1.10.0+cu111 torchvision==0.11.1+cu111 torchaudio==0.10.0+cu
```

How to check if pytorch is installed successfully?

```
import torch
```

If you could import it, then you succesfully install it.

Note: the package of PyTorch is called torch.

In [7]:

```
import torch
```

Dataset Preparation:

For this practice class we will use flickr8k to train our models.

flickr 8k (~1G) is a captioning dataset. You can download it from Kaggle using the link [https://www.kaggle.com/adityajn105/flickr8k/activity (https://www.kaggle.com/adityajn105/flickr8k/activity)](https://www.kaggle.com/adityajn105/flickr8k/activity). But we will share the dataset for the practice class in our datas server.

As Colab will delete all your uploading files when 12hs limit comes. Now, we use the drive.mount method to make our notebook reach our datasets without uploading.

What you should do:

Download the dataset from here (http://ltdata1.informatik.uni-hamburg.de/flickr8k/flickr8k.zip). Then uploading it to your own google drive. Run the code below, using the token your own google drive gives back to you.

Note: Please do not share your notebook, because it would be potential risks. We don't know who can reach your files.

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

In [9]:

```
!pwd
```

```
/raid/seid/par4sem/tmp
```

In [8]:

```
! ls /content/drive/MyDrive/
```

```
ch7.ipynb
```

Now we have flickr8k folder under our directory

Get the dataset from our server

The dataset is available in our server. Download it to your colab directory as follows `wget - http://ltdata1.informatik.uni-hamburg.de/flickr8k/flickr8k.zip`

In [10]:

```
!wget -  http://ltdata1.informatik.uni-hamburg.de/flickr8k/flickr8k.zip
```

```
--2021-11-20 23:28:42--  http://-/ (http://-/)
Resolving - (-)... failed: Name or service not known.
wget: unable to resolve host address ‘-’
--2021-11-20 23:28:42--  http://ltdata1.informatik.uni-hamburg.de/flic
kr8k/flickr8k.zip (http://ltdata1.informatik.uni-hamburg.de/flickr8k/f
lickr8k.zip)
Resolving ltdata1.informatik.uni-hamburg.de (ltdata1.informatik.uni-ha
mburg.de)... 134.100.15.200
Connecting to ltdata1.informatik.uni-hamburg.de (ltdata1.informatik.un
i-hamburg.de)|134.100.15.200|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1112971163 (1.0G) [application/zip]
Saving to: ‘flickr8k.zip’

flickr8k.zip        100%[===================>]   1.04G  51.7MB/s    in
16s

2021-11-20 23:28:58 (67.2 MB/s) - ‘flickr8k.zip’ saved [1112971163/111
2971163]

FINISHED --2021-11-20 23:28:58--
Total wall clock time: 16s
Downloaded: 1 files, 1.0G in 16s (67.2 MB/s)
```

unzip the file to a folder

unzip flickr8k.zip -d flickr8k

In [11]:

```
!unzip flickr8k.zip -d flickr8k
```

```
Archive:  flickr8k.zip
  inflating: flickr8k/Images/1000268201_693b08cb0e.jpg
  inflating: flickr8k/Images/1001773457_577c3a7d70.jpg
  inflating: flickr8k/Images/1002674143_1b742ab4b8.jpg
  inflating: flickr8k/Images/1003163366_44323f5815.jpg
  inflating: flickr8k/Images/1007129816_e794419615.jpg
  inflating: flickr8k/Images/1007320043_627395c3d8.jpg
  inflating: flickr8k/Images/1009434119_febe49276a.jpg
  inflating: flickr8k/Images/1012212859_01547e3f17.jpg
  inflating: flickr8k/Images/1015118661_980735411b.jpg
  inflating: flickr8k/Images/1015584366_dfcec3c85a.jpg
  inflating: flickr8k/Images/101654506_8eb26cfb60.jpg
  inflating: flickr8k/Images/101669240_b2d3e7f17b.jpg
  inflating: flickr8k/Images/1016887272_03199f49c4.jpg
  inflating: flickr8k/Images/1019077836_6fc9b15408.jpg
  inflating: flickr8k/Images/1019604187_d087bf9a5f.jpg
  inflating: flickr8k/Images/1020651753_06077ec457.jpg
  inflating: flickr8k/Images/1022454332_6af2c1449a.jpg
  inflating: flickr8k/Images/1022454428_b6b660a67b.jpg
```

move the file to your Gdrive

In [ ]:

```
!mv flickr8k  /content/drive/MyDrive/
```

Data Loader

There will be one folder containing all the images and one text file under your directory.

Images: 8091 pictures.

Caption: description for each image

The start point for any deep learning models is data processing. Be patience, because most of the bugs will happen in the future if you make mistakes in this step.

In Pytorch, the interface for training to use is called DataLoader.

In [38]:

```
# Check the content
!ls /content/drive/MyDrive/flickr8k/
```

```
captions.txt   Images
```

In [39]:

```
# How many images we have
!ls /content/drive/MyDrive/flickr8k/Images | wc -l
```

```
8091
```

In [40]:

```
# See how the description file looks like, image_name, description
!tail /content/drive/MyDrive/flickr8k/captions.txt
```

```
997338199_7343367d7f.jpg,A person stands near golden walls .
997338199_7343367d7f.jpg,a woman behind a scrolled wall is writing
997338199_7343367d7f.jpg,A woman standing near a decorated wall writes
.
997338199_7343367d7f.jpg,The walls are covered in gold and patterns .
997338199_7343367d7f.jpg,"Woman writing on a pad in room with gold , d
ecorated walls ."
997722733_0cb5439472.jpg,A man in a pink shirt climbs a rock face
997722733_0cb5439472.jpg,A man is rock climbing high in the air .
997722733_0cb5439472.jpg,A person in a red shirt climbing up a rock fa
ce covered in assist handles .
997722733_0cb5439472.jpg,A rock climber in a red shirt .
997722733_0cb5439472.jpg,A rock climber practices on a rock climbing w
all .
```

In [41]:

```
# lets see how the image for the file '997722733_0cb5439472.jpg' looks like, for
from PIL import Image
Image.open('/content/drive/MyDrive/flickr8k/Images/997722733_0cb5439472.jpg')
```

Out[41]:

IMPORTANT THING We should know in this step!

We want to convert text to numerical values

1. We need a Vocabulary mapping each word to a index
2. We need to setup a Pytorch dataset to load the data
3. Setup padding of every batch (all examples should be of same seq_len and setup dataloader)

We will look at itos --> index_to_sentence and stoi-->sentence_to_index here.

In [14]:

```
# use spaCy for preprocessing of the description text
!python -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.0.0
  Downloading https://github.com/explosion/spacy-models/releases/downl
oad/en_core_web_sm-3.0.0/en_core_web_sm-3.0.0-py3-none-any.whl (http
s://github.com/explosion/spacy-models/releases/download/en_core_web_sm
-3.0.0/en_core_web_sm-3.0.0-py3-none-any.whl) (13.7 MB)
     |████████████████████████████████| 13.7 MB 3.0 MB/s eta 0:00:01
Requirement already satisfied: spacy<3.1.0,>=3.0.0 in /srv/home/yimam/
anaconda3/lib/python3.7/site-packages (from en-core-web-sm==3.0.0) (3.
0.3)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /srv/home/yima
m/anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-
core-web-sm==3.0.0) (0.8.2)
Requirement already satisfied: typer<0.4.0,>=0.3.0 in /srv/home/yimam/
anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-co
re-web-sm==3.0.0) (0.3.2)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /srv/home/yima
m/anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-
core-web-sm==3.0.0) (3.0.5)
Requirement already satisfied: packaging>=20.0 in /srv/home/yimam/anac
onda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-core-w
eb-sm==3.0.0) (20.4)
Requirement already satisfied: catalogue<2.1.0,>=2.0.1 in /srv/home/yi
mam/anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->e
n-core-web-sm==3.0.0) (2.0.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.0 in /srv/hom
e/yimam/anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.
0->en-core-web-sm==3.0.0) (3.0.1)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /srv/home/
yimam/anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0-
>en-core-web-sm==3.0.0) (1.0.5)
Requirement already satisfied: numpy>=1.15.0 in /srv/home/yimam/anacon
da3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-core-web
-sm==3.0.0) (1.18.5)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /srv/home/yimam/
anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-co
re-web-sm==3.0.0) (4.47.0)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /srv/home/yimam/a
naconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-cor
e-web-sm==3.0.0) (0.7.4)
Requirement already satisfied: setuptools in /srv/home/yimam/anaconda
3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-core-web-s
m==3.0.0) (49.1.0.post20200710)
Requirement already satisfied: thinc<8.1.0,>=8.0.0 in /srv/home/yimam/
anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-co
re-web-sm==3.0.0) (8.0.1)
Requirement already satisfied: importlib-metadata>=0.20 in /srv/home/y
imam/anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->
en-core-web-sm==3.0.0) (3.10.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /srv/home/yi
mam/anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->e
n-core-web-sm==3.0.0) (2.25.1)
```

```
Requirement already satisfied: pydantic<1.8.0,>=1.7.1 in /srv/home/yim
am/anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en
-core-web-sm==3.0.0) (1.7.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.0 in /srv/home/yimam/
anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-co
re-web-sm==3.0.0) (2.4.0)
Requirement already satisfied: jinja2 in /srv/home/yimam/anaconda3/li
b/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==
3.0.0) (2.11.2)
Requirement already satisfied: pathy in /srv/home/yimam/anaconda3/lib/
python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-core-web-sm==3.
0.0) (0.4.0)
Requirement already satisfied: typing-extensions>=3.7.4 in /srv/home/y
imam/anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->
en-core-web-sm==3.0.0) (3.7.4.3)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /srv/home/yimam/
anaconda3/lib/python3.7/site-packages (from spacy<3.1.0,>=3.0.0->en-co
re-web-sm==3.0.0) (2.0.5)
Requirement already satisfied: zipp>=0.5 in /srv/home/yimam/anaconda3/
lib/python3.7/site-packages (from importlib-metadata>=0.20->spacy<3.1.
0,>=3.0.0->en-core-web-sm==3.0.0) (3.1.0)
Requirement already satisfied: six in /srv/home/yimam/anaconda3/lib/py
thon3.7/site-packages (from packaging>=20.0->spacy<3.1.0,>=3.0.0->en-c
ore-web-sm==3.0.0) (1.15.0)
Requirement already satisfied: pyparsing>=2.0.2 in /srv/home/yimam/ana
conda3/lib/python3.7/site-packages (from packaging>=20.0->spacy<3.1.0,
>=3.0.0->en-core-web-sm==3.0.0) (2.4.7)
Requirement already satisfied: idna<3,>=2.5 in /srv/home/yimam/anacond
a3/lib/python3.7/site-packages (from requests<3.0.0,>=2.13.0->spacy<3.
1.0,>=3.0.0->en-core-web-sm==3.0.0) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /srv/home/yima
m/anaconda3/lib/python3.7/site-packages (from requests<3.0.0,>=2.13.0-
>spacy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (1.25.9)
Requirement already satisfied: chardet<5,>=3.0.2 in /srv/home/yimam/an
aconda3/lib/python3.7/site-packages (from requests<3.0.0,>=2.13.0->spa
cy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /srv/home/yimam/a
naconda3/lib/python3.7/site-packages (from requests<3.0.0,>=2.13.0->sp
acy<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (2020.12.5)
Requirement already satisfied: click<7.2.0,>=7.1.1 in /srv/home/yimam/
anaconda3/lib/python3.7/site-packages (from typer<0.4.0,>=0.3.0->spacy
<3.1.0,>=3.0.0->en-core-web-sm==3.0.0) (7.1.2)
Requirement already satisfied: MarkupSafe>=0.23 in /srv/home/yimam/ana
conda3/lib/python3.7/site-packages (from jinja2->spacy<3.1.0,>=3.0.0->
en-core-web-sm==3.0.0) (1.1.1)
Requirement already satisfied: smart-open<4.0.0,>=2.2.0 in /srv/home/y
imam/anaconda3/lib/python3.7/site-packages (from pathy->spacy<3.1.0,>=
3.0.0->en-core-web-sm==3.0.0) (3.0.0)
Installing collected packages: en-core-web-sm
Successfully installed en-core-web-sm-3.0.0
WARNING: You are using pip version 21.0.1; however, version 21.3.1 is
 available.
You should consider upgrading via the '/srv/home/yimam/anaconda3/bin/p
ython -m pip install --upgrade pip' command.
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

In [15]:

```
import os  # when loading file paths
import pandas as pd  # for lookup in annotation file
import spacy  # for tokenizer
import torch
from torch.nn.utils.rnn import pad_sequence  # pad batch
from torch.utils.data import DataLoader, Dataset
from PIL import Image  # Load img
import torchvision.transforms as transforms


# Download with: python -m spacy download en
# spacy_eng = spacy.load("en")

# if not in colab, use the following
# Download with: python -m spacy download en_core_web_sm
spacy_eng =  spacy.load("en_core_web_sm")
```

1. Build vocabulary and numericalize

Tokenizer:

```
I attend NLP4Web Course.

--- Tokenizer ---

['i', 'attend', 'nlp4web', 'clas
s']
```

Build Vocab.: Counting the frequency. If meet the freq_threshold then add to our vocabulary.

Index plus 1 (start from 4, because we have `pad`, `sos`, `eos`, `unk` tokens)

In [16]:

```
class Vocabulary:
    def __init__(self, freq_threshold):
        self.itos = {0: "<PAD>", 1: "<SOS>", 2: "<EOS>", 3: "<UNK>"}
        self.stoi = {"<PAD>": 0, "<SOS>": 1, "<EOS>": 2, "<UNK>": 3}
        self.freq_threshold = freq_threshold

    def __len__(self):
        return len(self.itos)

    @staticmethod
    def tokenizer_eng(text):
        return [tok.text.lower() for tok in spacy_eng.tokenizer(text)]

    def build_vocabulary(self, sentence_list):
        frequencies = {}
        idx = 4

        for sentence in sentence_list:
            for word in self.tokenizer_eng(sentence):
                if word not in frequencies:
                    frequencies[word] = 1

                else:
                    frequencies[word] += 1

                if frequencies[word] == self.freq_threshold:
                    self.stoi[word] = idx
                    self.itos[idx] = word
                    idx += 1

    def numericalize(self, text):
        tokenized_text = self.tokenizer_eng(text)

        return [
            self.stoi[token] if token in self.stoi else self.stoi["<UNK>"]
```

```
            for token in tokenized_text
        ]
```

In [43]:

```
#spaCy based tokenizer
examplesent = ["the old fox jumps over the lazy dog."]
spacy_eng.tokenizer(examplesent[0])
```

Out[43]:

```
the old fox jumps over the lazy dog.
```

In [44]:

```
vocab = Vocabulary(1)
vocab.build_vocabulary(examplesent)
print("index to sentence==>",vocab.itos)
print("sentence to index==>", vocab.stoi)
```

```
index to sentence==> {0: '<PAD>', 1: '<SOS>', 2: '<EOS>', 3: '<UNK>',
4: 'the', 5: 'old', 6: 'fox', 7: 'jumps', 8: 'over', 9: 'lazy', 10: 'd
og', 11: '.'}
sentence to index==> {'<PAD>': 0, '<SOS>': 1, '<EOS>': 2, '<UNK>': 3,
'the': 4, 'old': 5, 'fox': 6, 'jumps': 7, 'over': 8, 'lazy': 9, 'dog':
10, '.': 11}
```

In [45]:

```
vocab.stoi["<PAD>"],vocab.stoi["<SOS>"],vocab.stoi["<EOS>"],vocab.stoi["<UNK>"]
```

Out[45]:

```
(0, 1, 2, 3)
```

In [46]:

```
vocab.numericalize("fox and diffirent dog")
```

Out[46]:

```
[6, 3, 3, 10]
```

2. Make Flickr Dataset/DataLoader, image and caption

Image part is more easy than the caption part

In [17]:

```
class FlickrDataset(Dataset):
    def __init__(self, root_dir, captions_file, transform=None, freq_threshold=5
        self.root_dir = root_dir
        self.df = pd.read_csv(captions_file)
        self.transform = transform

        # Get img, caption columns
        self.imgs = self.df["image"]
        self.captions = self.df["caption"]

        # Initialize vocabulary and build vocab
        self.vocab = Vocabulary(freq_threshold)
        self.vocab.build_vocabulary(self.captions.tolist())

    def __len__(self):
        return len(self.df)

    def __getitem__(self, index):
        caption = self.captions[index]
        img_id = self.imgs[index]
        img = Image.open(os.path.join(self.root_dir, img_id)).convert("RGB")

        if self.transform is not None:
            img = self.transform(img)

        numericalized_caption = [self.vocab.stoi["<SOS>"]]
        numericalized_caption += self.vocab.numericalize(caption)
        numericalized_caption.append(self.vocab.stoi["<EOS>"])

        return img, torch.tensor(numericalized_caption)
```

We want to load data in the way of batch not just a single pair.

In [18]:

```
class MyCollate:
    def __init__(self, pad_idx):
        self.pad_idx = pad_idx

    def __call__(self, batch):
        imgs = [item[0].unsqueeze(0) for item in batch]
        imgs = torch.cat(imgs, dim=0)
        targets = [item[1] for item in batch]
        targets = pad_sequence(targets, batch_first=False, padding_value=self.pa

        return imgs, targets
```

Dataloader is the interface for PyTorch to reach data after processing

In [19]:

```
def get_loader(root_folder, annotation_file, transform, batch_size=32, num_worke

    dataset = FlickrDataset(root_folder, annotation_file, transform=transform)

    pad_idx = dataset.vocab.stoi["<PAD>"]

    loader = DataLoader(
        dataset=dataset,
        batch_size=batch_size,
        num_workers=num_workers,
        shuffle=shuffle,
        pin_memory=pin_memory,
        collate_fn=MyCollate(pad_idx=pad_idx),
    )

    return loader, dataset
```

In [20]:

```
transform = transforms.Compose(
    [transforms.Resize((224, 224)),
     transforms.ToTensor(),])
```

In [24]:

```
loader, dataset = get_loader(
  #  "./flickr8k/Images/", "./flickr8k/captions.txt", transform=transform
     "/content/drive/MyDrive/flickr8k/Images/", /content/drive/MyDrive/flickr8k/

    )

for idx, (imgs, captions) in enumerate(loader):
    print(imgs.shape)
    print(captions.shape)
    break
```

```
torch.Size([32, 3, 224, 224])
torch.Size([20, 32])
```

One more thing we should finish here before we start our model part.

In order to have a look at the current performance, we sample some cases from our dataset and generate related captions comparing with the ground truth.

What you should do: (Assignment 2)

1. **Make a test_examples folder and put five images/rename in this folder**
2. **Copy each caption to the code below**

In [49]:

```
import torch
import torchvision.transforms as transforms
from PIL import Image
```

In [50]:

```
def print_examples(model, device, dataset):
    transform = transforms.Compose(
        [
            transforms.Resize((299, 299)),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
        ]
    )

    model.eval()
    test_img1 = transform(Image.open("test_examples/dog.jpg").convert("RGB")).ur
        0
    )
    print("Example 1 CORRECT: Dog on a beach by the ocean")
    print(
        "Example 1 OUTPUT: "
        + " ".join(model.caption_image(test_img1.to(device), dataset.vocab))
    )
    test_img2 = transform(
        Image.open("test_examples/child.jpg").convert("RGB")
    ).unsqueeze(0)
    print("Example 2 CORRECT: Child holding red frisbee outdoors")
    print(
        "Example 2 OUTPUT: "
        + " ".join(model.caption_image(test_img2.to(device), dataset.vocab))
    )
    test_img3 = transform(Image.open("test_examples/bus.png").convert("RGB")).ur
        0
    )
    print("Example 3 CORRECT: Bus driving by parked cars")
    print(
        "Example 3 OUTPUT: "
        + " ".join(model.caption_image(test_img3.to(device), dataset.vocab))
    )
    test_img4 = transform(
```

```python
        Image.open("test_examples/boat.png").convert("RGB")
    ).unsqueeze(0)
    print("Example 4 CORRECT: A small boat in the ocean")
    print(
        "Example 4 OUTPUT: "
        + " ".join(model.caption_image(test_img4.to(device), dataset.vocab))
    )
    test_img5 = transform(
        Image.open("test_examples/horse.png").convert("RGB")
    ).unsqueeze(0)
    print("Example 5 CORRECT: A cowboy riding a horse in the desert")
    print(
        "Example 5 OUTPUT: "
        + " ".join(model.caption_image(test_img5.to(device), dataset.vocab))
    )
    model.train()
```

Save and loading checkpoints

In [51]:

```python
def save_checkpoint(state, filename="my_checkpoint.pth.tar"):
    print("=> Saving checkpoint")
    torch.save(state, filename)


def load_checkpoint(checkpoint, model, optimizer):
    print("=> Loading checkpoint")
    model.load_state_dict(checkpoint["state_dict"])
    optimizer.load_state_dict(checkpoint["optimizer"])
    step = checkpoint["step"]
    return step
```

Model

In [52]:

```
import torch
import torch.nn as nn
import statistics
import torchvision.models as models
```

For the image understanding part, we name it as EncoderCNN.

Here we use pretrained inception model, also you could change it to VGG or ResNet.

Two parameters: embed_size, train_CNN

In [53]:

```
class EncoderCNN(nn.Module):
    def __init__(self, embed_size, train_CNN=False):
        super(EncoderCNN, self).__init__()
        self.train_CNN = train_CNN
        self.inception = models.inception_v3(pretrained=True, aux_logits=False)
        self.inception.fc = nn.Linear(self.inception.fc.in_features, embed_size)
        self.relu = nn.ReLU()
        self.times = []
        self.dropout = nn.Dropout(0.5)

    def forward(self, images):
        features = self.inception(images)
        return self.dropout(self.relu(features))
```

For the caption generation part, we name it as DecoderRNN.

Here we use LSTM

Three parameters: vocab_size, embed_size, num_layers.

In [54]:

```python
class DecoderRNN(nn.Module):
    def __init__(self, embed_size, hidden_size, vocab_size, num_layers):
        super(DecoderRNN, self).__init__()
        self.embed = nn.Embedding(vocab_size, embed_size)
        self.lstm = nn.LSTM(embed_size, hidden_size, num_layers)
        self.linear = nn.Linear(hidden_size, vocab_size)
        self.dropout = nn.Dropout(0.5)

    def forward(self, features, captions):
        embeddings = self.dropout(self.embed(captions))
        embeddings = torch.cat((features.unsqueeze(0), embeddings), dim=0)
        hiddens, _ = self.lstm(embeddings)
        outputs = self.linear(hiddens)
        return outputs
```

Now we have encoder and decoder, what we would do next is to obtain them in a way of end to end learning.

Pay attention to caption_image function:

```
max_length, '<EOS>', itos
```

In [55]:

```
class CNNtoRNN(nn.Module):
    def __init__(self, embed_size, hidden_size, vocab_size, num_layers):
        super(CNNtoRNN, self).__init__()
        self.encoderCNN = EncoderCNN(embed_size)
        self.decoderRNN = DecoderRNN(embed_size, hidden_size, vocab_size, num_la

    def forward(self, images, captions):
        features = self.encoderCNN(images)
        outputs = self.decoderRNN(features, captions)
        return outputs

    def caption_image(self, image, vocabulary, max_length=50):
        result_caption = []

        with torch.no_grad():
            x = self.encoderCNN(image).unsqueeze(0)
            states = None

            for _ in range(max_length):
                hiddens, states = self.decoderRNN.lstm(x, states)
                output = self.decoderRNN.linear(hiddens.squeeze(0))
                predicted = output.argmax(1)
                result_caption.append(predicted.item())
                x = self.decoderRNN.embed(predicted).unsqueeze(0)

                if vocabulary.itos[predicted.item()] == "<EOS>":
                    break

        return [vocabulary.itos[idx] for idx in result_caption]
```

Training

In [56]:

```
import torch
from tqdm import tqdm
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
from torch.utils.tensorboard import SummaryWriter
```

First of all, we need creat a model and set parameters

In [57]:

```
transform = transforms.Compose(
    [
        transforms.Resize((356, 356)),
        transforms.RandomCrop((299, 299)),
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
    ]
  )

train_loader, dataset = get_loader(
       # root_folder="./flickr8k/Images",
       #  annotation_file="./flickr8k/captions.txt",
        root_folder="/content/drive/MyDrive/flickr8k",
        annotation_file="/content/drive/MyDrive/flickr8k/captions.txt",
        transform=transform,
        num_workers=2,
    )

torch.backends.cudnn.benchmark = True
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
load_model = False
save_model = False
train_CNN = False

# Hyperparameters
embed_size = 256
hidden_size = 256
vocab_size = len(dataset.vocab)
num_layers = 1
learning_rate = 3e-4
num_epochs = 100

# for tensorboard
writer = SummaryWriter("runs/flickr")
step = 0
```

```python

# initialize model, loss etc
model = CNNtoRNN(embed_size, hidden_size, vocab_size, num_layers).to(device)
criterion = nn.CrossEntropyLoss(ignore_index=dataset.vocab.stoi["<PAD>"])
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

# Only finetune the CNN
for name, param in model.encoderCNN.inception.named_parameters():
    if "fc.weight" in name or "fc.bias" in name:
        param.requires_grad = True
    else:
        param.requires_grad = train_CNN

if load_model:
    step = load_checkpoint(torch.load("my_checkpoint.pth.tar"), model, optimizer

model.train()
```

Out[57]:

```
CNNtoRNN(
  (encoderCNN): EncoderCNN(
    (inception): Inception3(
      (Conv2d_1a_3x3): BasicConv2d(
        (conv): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), bias
=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, t
rack_running_stats=True)
      )
      (Conv2d_2a_3x3): BasicConv2d(
        (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), bia
s=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, t
rack_running_stats=True)
      )
      (Conv2d_2b_3x3): BasicConv2d(
        (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), pad
```

TRAINING: Let see what happens!

In [ ]:

```python
for epoch in range(num_epochs):
    # Uncomment the line below to see a couple of test cases
    # print_examples(model, device, dataset)

    if save_model:
        checkpoint = {
            "state_dict": model.state_dict(),
            "optimizer": optimizer.state_dict(),
            "step": step,
        }
        save_checkpoint(checkpoint)

    for idx, (imgs, captions) in tqdm(
        enumerate(train_loader), total=len(train_loader), leave=False
    ):
        imgs = imgs.to(device)
        captions = captions.to(device)

        outputs = model(imgs, captions[:-1])
        loss = criterion(
            outputs.reshape(-1, outputs.shape[2]), captions.reshape(-1)
        )

        writer.add_scalar("Training loss", loss.item(), global_step=step)
        step += 1

        optimizer.zero_grad()
        loss.backward(loss)
        optimizer.step()
```

```
45%|████▌     | 564/1265 [01:38<02:00,  5.81it/s]
```

Assignment: (15 points)

1. Summary the difference of LSTM decoder in the training and test phrases? (2 Points)
2. Coding1: Prepare test samples in order to see the performance during training. (3 Points)

3. Coding2: Write the inference code.(10 Points)

   Input any image, using our trained model to generate sentence for the image.

Question 1: 2 Pts

Question 2: 3 Pts

In [ ]:

```
# Question 3: 10Pts Write you code here
```

In [ ]:

```

```