seq2seq-neural-image-caption-generation

July 7, 2024

still in progress

1 Imports

[69]:
```
import torch
from torch import nn, optim
from torch.nn.utils.rnn import pad_sequence
from torch.utils.data import DataLoader, Dataset

import torchvision
import torchvision.transforms as transforms
import torchvision.models as models

from torchmetrics import BLEUScore

import spacy

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # plotting
%matplotlib inline
import seaborn as sns

from PIL import Image  # load img
import statistics # self exmplanatory
from tqdm import tqdm # progress bar

import os  # when loading file paths
import warnings
warnings.filterwarnings("ignore")

print("Imported Successfully!")
```

```
Imported Successfully!
```

2 Data Collection & Preprocessing

2.1 Inspecting

[70]:
```
image_dir_path = "/kaggle/input/flickr8kimagescaptions/flickr8k/images/"
captions_dir_path = "/kaggle/input/flickr8kimagescaptions/flickr8k/captions.txt"
```

[71]:
```
captions = pd.read_csv(captions_dir_path)
captions.head()
```

[71]:

	image	caption
0	1000268201_693b08cb0e.jpg	A child in a pink dress is climbing up a set o...
1	1000268201_693b08cb0e.jpg	A girl going into a wooden building .
2	1000268201_693b08cb0e.jpg	A little girl climbing into a wooden playhouse .
3	1000268201_693b08cb0e.jpg	A little girl climbing the stairs to her playh...
4	1000268201_693b08cb0e.jpg	A little girl in a pink dress going into a woo...

2.2 Vocabulary

2.2.1 English Model for spaCy

[72]:
```
!python -m spacy download en
```

```
 As of spaCy v3.0, shortcuts like 'en' are deprecated. Please use the
full pipeline package name 'en_core_web_sm' instead.
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1-py3-none-
any.whl (12.8 MB)
     12.8/12.8 MB
73.1 MB/s eta 0:00:0000:0100:01
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in
/opt/conda/lib/python3.10/site-packages (from en-core-web-sm==3.7.1) (3.7.3)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
```

```
sm==3.7.1) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (8.2.2)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.0.10)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.3.4)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.9.0)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (4.66.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.5.3)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.10/site-packages
(from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.1.2)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.10/site-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (69.0.3)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (21.3)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in
/opt/conda/lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in /opt/conda/lib/python3.10/site-
```

```
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.26.4)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from
packaging>=20.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.1.1)
Requirement already satisfied: annotated-types>=0.4.0 in
/opt/conda/lib/python3.10/site-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.6 in
/opt/conda/lib/python3.10/site-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (2.14.6)
Requirement already satisfied: typing-extensions>=4.6.1 in
/opt/conda/lib/python3.10/site-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (4.9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-
packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-
sm==3.7.1) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from
requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2024.2.2)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in
/opt/conda/lib/python3.10/site-packages (from
thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.7.10)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in
/opt/conda/lib/python3.10/site-packages (from
thinc<8.3.0,>=8.2.2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.1.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in
/opt/conda/lib/python3.10/site-packages (from
typer<0.10.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (8.1.7)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in
/opt/conda/lib/python3.10/site-packages (from
weasel<0.4.0,>=0.1.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/opt/conda/lib/python3.10/site-packages (from jinja2->spacy<3.8.0,>=3.7.2->en-
core-web-sm==3.7.1) (2.1.3)
 Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

[73]:
```
spacy_eng = spacy.load("en_core_web_sm")
```

[74]:
```
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
        ]
```

2.3 Custom Dataset

[75]:
```
class FlickrDataset(Dataset):
    def __init__(self, root_dir, captions_file, transform = None, freq_threshold␣
↪= 6):
        self.root_dir = root_dir
        self.df = pd.read_csv(captions_file)
        self.transform = transform
```

```
        # Getting Images
        self.imgs = self.df["image"]
        self.captions = self.df["caption"]

        # Vocabulary
        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)
```

2.4 Custom Collate for Loader

In PyTorch, when you're working with datasets and DataLoader, collation refers to the process of combining individual samples into batches.

Now, why is collation necessary?

- Batching: Training neural networks is often done in batches rather than one sample at a time. This is because processing data in batches can make computations more efficient, especially on GPUs where parallelization can be leveraged effectively.
- Variable Length Data: In many cases, such as with text data or variable-length sequences, individual samples might have different lengths. However, to process them efficiently in batches, they need to be padded to the same length.

[76]:
```
class Custom_Collate:
    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.
↪pad_idx)

        return imgs, targets
```

2.5 Data Loading Pipeline

[77]:
```
def get_loader(
    root_folder,
    annotation_file,
    transform,
    batch_size=32,
    num_workers=8,
    shuffle=True,
    pin_memory=True,
):
    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=Custom_Collate(pad_idx=pad_idx),
    )
    print("Data Loaded Successfully!!")

    return loader, dataset
```

2.6 Transforms

[78]:
```
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)),
    ]
)
```

2.7 Data Pipeline

[79]:
```
train_loader, dataset = get_loader(image_dir_path, captions_dir_path,␣
↪transform=transform)
```

```
Data Loaded Successfully!!
```

3 Model

3.1 Encoder

[81]:
```
class EncoderCNN(nn.Module):
    def __init__(self, embed_size, train_CNN=False):
        super(EncoderCNN, self).__init__()
        self.train_CNN = train_CNN
        self.resnet = models.resnet50(pretrained=True)

        # Freeze the parameters of the pre-trained ResNet
        if not train_CNN:
            for param in self.resnet.parameters():
                param.requires_grad = False

        # Modify the last fully connected layer to match the embed_size
        self.resnet.fc = nn.Linear(self.resnet.fc.in_features, embed_size)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.5)

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

3.2 Decoder

[82]:
```
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
```

3.3 Encoder-Decoder Architecture

putting it all together

[83]:
```
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_layers)

    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]
```

4 Utility Functions for Training

4.0.1 Printing Examples

[84]:
```
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)),
        ]
    )
```

```
    # To Evaluation Mode
    model.eval()

    # First Image
    test_img1 = transform(Image.open("/kaggle/input/flickr8kimagescaptions/
↪flickr8k/images/1001773457_577c3a7d70.jpg").convert("RGB")).unsqueeze(0)
    print("Example 1 CORRECT: A black dog and a spotted dog are fighting")
    print("Example 1 OUTPUT: " + " ".join(model.caption_image(test_img1.
↪to(device), dataset.vocab)))
    print("\n")

    # Second Image
    test_img2 = transform(Image.open("/kaggle/input/flickr8kimagescaptions/
↪flickr8k/images/102351840_323e3de834.jpg").convert("RGB")).unsqueeze(0)

    print("Example 2 CORRECT: A man is drilling through the frozen ice of a pond␣
↪.")
    print("Example 2 OUTPUT: " + " ".join(model.caption_image(test_img2.
↪to(device), dataset.vocab)))
    print("\n")

    # Third Image
    test_img3 = transform(Image.open("/kaggle/input/flickr8kimagescaptions/
↪flickr8k/images/1007320043_627395c3d8.jpg").convert("RGB")).unsqueeze(0)

    print("Example 3 CORRECT: A little girl climbing on red roping .")
    print("Example 3 OUTPUT: " + " ".join(model.caption_image(test_img3.
↪to(device), dataset.vocab)))
    print("\n")

    # Fourth Image
    test_img4 = transform(Image.open("/kaggle/input/flickr8kimagescaptions/
↪flickr8k/images/1015118661_980735411b.jpg").convert("RGB")).unsqueeze(0)

    print("Example 4 CORRECT: A young boy runs aross the street .")
    print("Example 4 OUTPUT: "+ " ".join(model.caption_image(test_img4.
↪to(device), dataset.vocab)))
    print("\n")

    # Fifth Image
    test_img5 = transform(Image.open("/kaggle/input/flickr8kimagescaptions/
↪flickr8k/images/1052358063_eae6744153.jpg").convert("RGB")).unsqueeze(0)
    print("Example 5 CORRECT: A boy takes a jump on his skateboard while another␣
↪boy with a skateboard watches .")
    print("Example 5 OUTPUT: "+ " ".join(model.caption_image(test_img5.
↪to(device), dataset.vocab)))
```

```
    print("\n")

    # Back to Training Mode
    model.train()
```

4.0.2 Saving Checkpoint

[85]:
```
def save_checkpoint(state, filename="my_checkpoint.pth.tar"):
    print("=> Saving checkpoint")
    torch.save(state, filename)
```

4.0.3 Loading Checkpoint

[86]:
```
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
```

4.1 Tokenizer

[87]:
```
def tokenize(text):
    return [token.text for token in spacy_eng.tokenizer(text)]
```

5 Training

5.1 Device Agnostic Code

[88]:
```
torch.backends.cudnn.benchmark = True
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

if device == 'cuda':
    !nvidia-smi
else:
    print("CUDA device is not available.")
```

```
CUDA device is not available.
```

5.2 Hyperparamters & Others

[89]:
```
embed_size = 256
hidden_size = 256
vocab_size = len(dataset.vocab)
num_layers = 1
learning_rate = 3e-4
num_epochs = 2
```

number of epochs is less on purpose

[90]:
```
load_model = False
save_model = False
train_CNN = False
```

5.3 Training Function

[91]:
```
# 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)
```

[95]:
```
def train():

    # Only finetune the CNN
    for name, param in model.encoderCNN.resnet.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()

    for epoch in tqdm(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, mininterval= 10):
            imgs = imgs.to(device)
            captions = captions.to(device)

            outputs = model(imgs, captions[:-1])
            loss = criterion(
```

```
                outputs.reshape(-1, outputs.shape[2]), captions.reshape(-1)
            )


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

[96]:
```
# Uncomment to train the below
train()
```

```
  0%|          | 0/2 [00:00<?, ?it/s]
  0%|          | 0/1265 [00:00<?, ?it/s]
  5%|         | 66/1265 [00:10<03:11,  6.27it/s]
 12%|        | 146/1265 [00:20<02:34,  7.25it/s]
 18%|        | 226/1265 [00:31<02:22,  7.30it/s]
 24%|       | 307/1265 [00:41<02:06,  7.58it/s]
 31%|       | 388/1265 [00:52<01:55,  7.59it/s]
 37%|      | 465/1265 [01:02<01:46,  7.52it/s]
 43%|     | 545/1265 [01:12<01:34,  7.60it/s]
 49%|     | 624/1265 [01:23<01:23,  7.63it/s]
 55%|    | 702/1265 [01:33<01:14,  7.54it/s]
 62%|   | 783/1265 [01:43<01:02,  7.66it/s]
 68%|   | 863/1265 [01:54<00:52,  7.65it/s]
 75%|  | 945/1265 [02:05<00:41,  7.68it/s]
 81%|  | 1023/1265 [02:15<00:31,  7.62it/s]
 87%| | 1103/1265 [02:25<00:20,  7.72it/s]
 94%|| 1183/1265 [02:35<00:10,  7.70it/s]
100%|| 1260/1265 [02:46<00:00,  7.65it/s]
 50%|     | 1/2 [02:47<02:47, 167.10s/it]
  0%|          | 0/1265 [00:00<?, ?it/s]
  5%|         | 65/1265 [00:10<03:07,  6.42it/s]
 11%|        | 145/1265 [00:20<02:32,  7.33it/s]
 18%|        | 225/1265 [00:31<02:21,  7.36it/s]
 24%|       | 303/1265 [00:41<02:07,  7.52it/s]
 30%|       | 381/1265 [00:51<01:57,  7.52it/s]
 36%|      | 457/1265 [01:01<01:48,  7.46it/s]
 43%|     | 538/1265 [01:11<01:35,  7.64it/s]
 49%|     | 619/1265 [01:22<01:24,  7.68it/s]
 55%|    | 697/1265 [01:33<01:15,  7.54it/s]
 62%|   | 779/1265 [01:43<01:02,  7.74it/s]
 68%|   | 861/1265 [01:53<00:52,  7.69it/s]
 74%|  | 937/1265 [02:04<00:43,  7.60it/s]
 80%|  | 1016/1265 [02:14<00:32,  7.66it/s]
 86%| | 1094/1265 [02:24<00:22,  7.69it/s]
 93%|| 1172/1265 [02:34<00:12,  7.61it/s]
 99%|| 1250/1265 [02:44<00:01,  7.65it/s]
100%|| 2/2 [05:33<00:00, 166.96s/it]
```

6 Plotting Results

[ ]:

7 Prediction

[97]:
```
print_examples(model, device, dataset)
```

```
Example 1 CORRECT: A black dog and a spotted dog are fighting
Example 1 OUTPUT: <SOS> a dog is running through the grass . <EOS>


Example 2 CORRECT: A man is drilling through the frozen ice of a pond .
Example 2 OUTPUT: <SOS> a man is riding a bike on a skateboard . <EOS>


Example 3 CORRECT: A little girl climbing on red roping .
Example 3 OUTPUT: <SOS> a young girl in a blue shirt and a blue shirt is jumping
into a pool . <EOS>


Example 4 CORRECT: A young boy runs aross the street .
Example 4 OUTPUT: <SOS> a young girl in a blue shirt and a blue shirt is
standing on a swing . <EOS>


Example 5 CORRECT: A boy takes a jump on his skateboard while another boy with a
skateboard watches .
Example 5 OUTPUT: <SOS> a man in a red shirt is standing on a bench with a <UNK>
in the background . <EOS>
```

obviously needs to be run for more epochs