Language and Image Processing
Project Report

Kirill Milintsevich, Kateryna Peikova

Generating product name from an image[1]

Data

For our project, we used the well-known dataset "Amazon Product Data"[2]. This is a quite big dataset, however, it might be a bit outdated compared to currently existing products at Amazon. The dataset consists of 9.4 million products metadata. Besides, it contains 350 000 unique products from the Clothing category.
We data splitted into Train-Validation-Test sets with sizes 60k-30k-30k.

Data preprocessing

Before building our model we did some preprocessing for textual and image data. This included:

- Select products in the "Clothing" category
- Filter out products with empty names or no images
- Remove products with duplicate images with Image-Match and ElasticSearch
- Shuffle data

Data transformation

Now we list transformations we applied as a part of our model.

Text transformations:

- Remove punctuation
- Numbers to <num>
- Build a numeric dictionary Size 6547 Occurred > 5 times
- Transform into a numerical representation
- Added <pad>
- All captions of size 52

Image transformations:

- Normalization
  mean = [0.485, 0.456, 0.406]
  std = [0.229, 0.224, 0.225]
  This values are for pretrained ImageNet models that we use as part of our Encoder.
- Resize to 64x64

[1] https://github.com/501Good/LTAT.01.005-LANGUAGE-AND-IMAGE-PROCESSING-Spring-2019-.git

[2] http://jmcauley.ucsd.edu/data/amazon/

Method

Our model implemented traditional and well-known Encoder-Decoder architecture. Main features of them are listed next.

Encoder

- **Input**: 3-colour channel image 64x64
- Pretrained ResNet101 on ImageNet
- Discarded the last two layers: pooling and linear layers
- Fine-tuning
- **Output**: 14x14 with 2048 channels

Decoder with attention

- **Attention** to generate weights
- LSTM
- Decoder's output -> **Beam search** (to generate several possible sentences and choose best among them).
- **Loss function** - Cross entropy loss

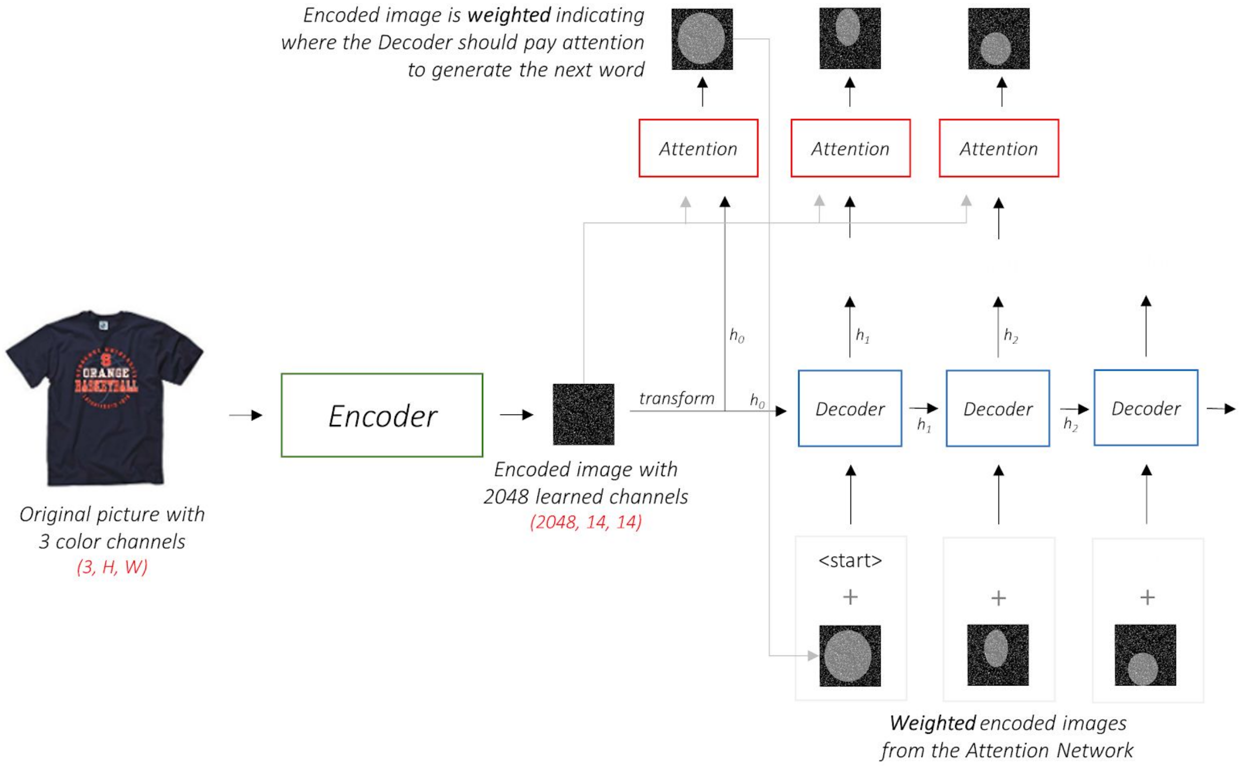
Encoded image is weighted indicating where the Decoder should pay attention to generate the next word
Attention
Attention
Attention
ORANGE BASKETBALL
Encoder
transform
h_0
h_0
h_1
h_2
Decoder
h_1
Decoder
h_2
Decoder
Original picture with 3 color channels
(3, H, W)
Encoded image with 2048 learned channels
(2048, 14, 14)
<start>
+
+
+
Weighted encoded images from the Attention Network

Figure 1. The architecture of image captioning model.

Training

We trained our model for 24 epochs (fine-tuning after 10th epoch).

- Bach size = 64
- Bach size for fine-tuning = 32

Additionally, we used pretrained FastText embeddings with fine-tuning.

The evaluation metrics are:

Accuracy - 62.901

Bleu - 12.88

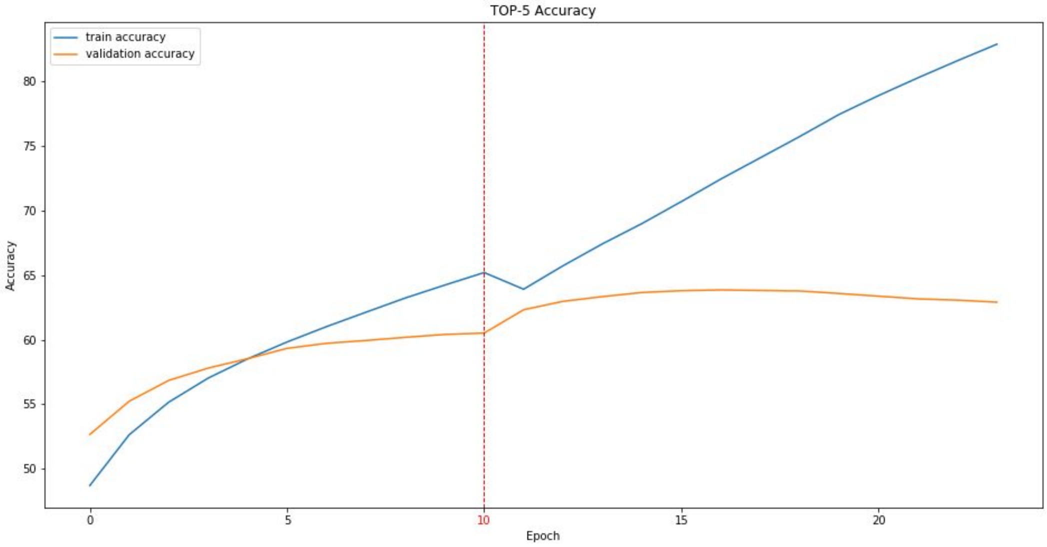

Figure 2. Accuracy metrics during training.

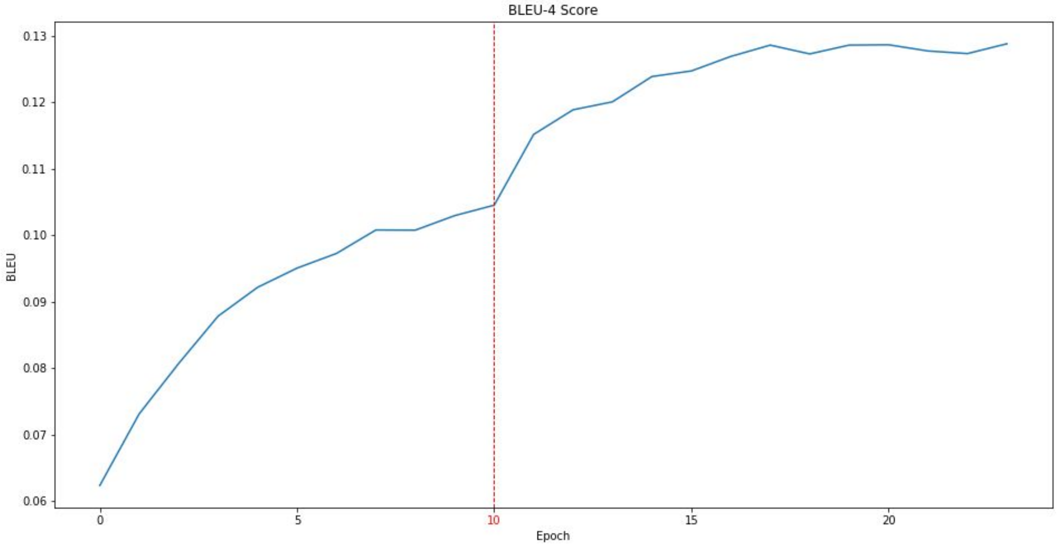

Figure 3. BLEU-4 score

Output example