Show and Tell: A Neural Image Caption Generator

Shizhao Wang, Chao Yu, Xinyuan Chen, Chutian Tai, Mingyu Zha

Model Details

Input: RGB images

Output: A sentence that describes the contents in the picture

Model Architecture: CNN (ResNet50) + LSTM (2 layers)

Datasets: MSCOCO + Flickr8k + Flickr30k

Hyperparameters: image_embedding_size =1000, batch_size=128, lr=0.001, hidden_units=500, vocabulary_size=12000

Optimizer: stochastic gradient descent is used with fixed learning rate without momentum

Implementation Details - Preprocessing

1. Create vocabulary:

- Count all distinct words appear in MSCOCO captions;
- Sort them in descending order according to the word frequency;
- Add four special tags into our vocabulary at the very beginning: <unknown>, <start>, <end>, <pad>;

2. Customize DataLoader:

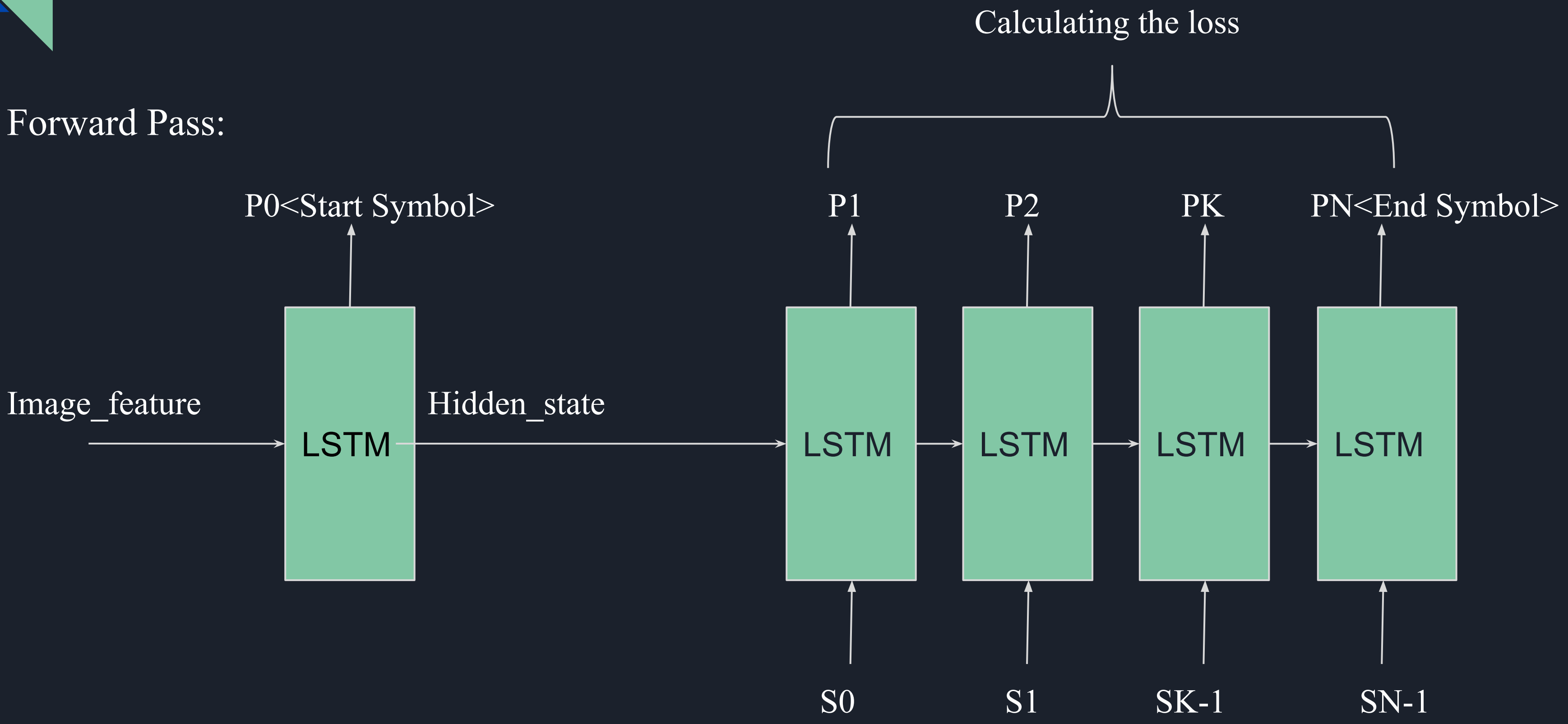
- Sort the batch in descending order according to the caption length; (in order to use pack_padded_sequence function)
- Convert the tokenized caption vector into id vector;
- Pad <start> flag at the front of caption, <end> flag at the end of caption, and zero paddings <pad> to ensure each caption have the same length;

I, love, Deep, Learning
I, love, Deep, Learning, so, much
I, love, study

<start>, I, love, Deep, Learning, so, much, <end>
<start>, I, love, Deep, Learning, <end>, <pad>, <pad>
<start>, I, love, study, <end>, <pad>, <pad>, <pad>

Implementation Details - RNN (Train)

Forward Pass:



Implementation Details - Loss Function

We use Negative Log-likelihood as Loss Function.[1]

```
def my_loss(y_prob, y, y_length):
    y_prob = F.log_softmax(y_prob, dim=2)
    y = y.contiguous()
    y = y.view(-1)
    y_prob = y_prob.view(-1, vocab_size)
    mask = ((y != 3) * (y != 1)).float()

    count = int(torch.sum(mask).data.item())
    total = mask * y_prob[range(y.shape[0]), y]
    return -torch.sum(total) / count
```

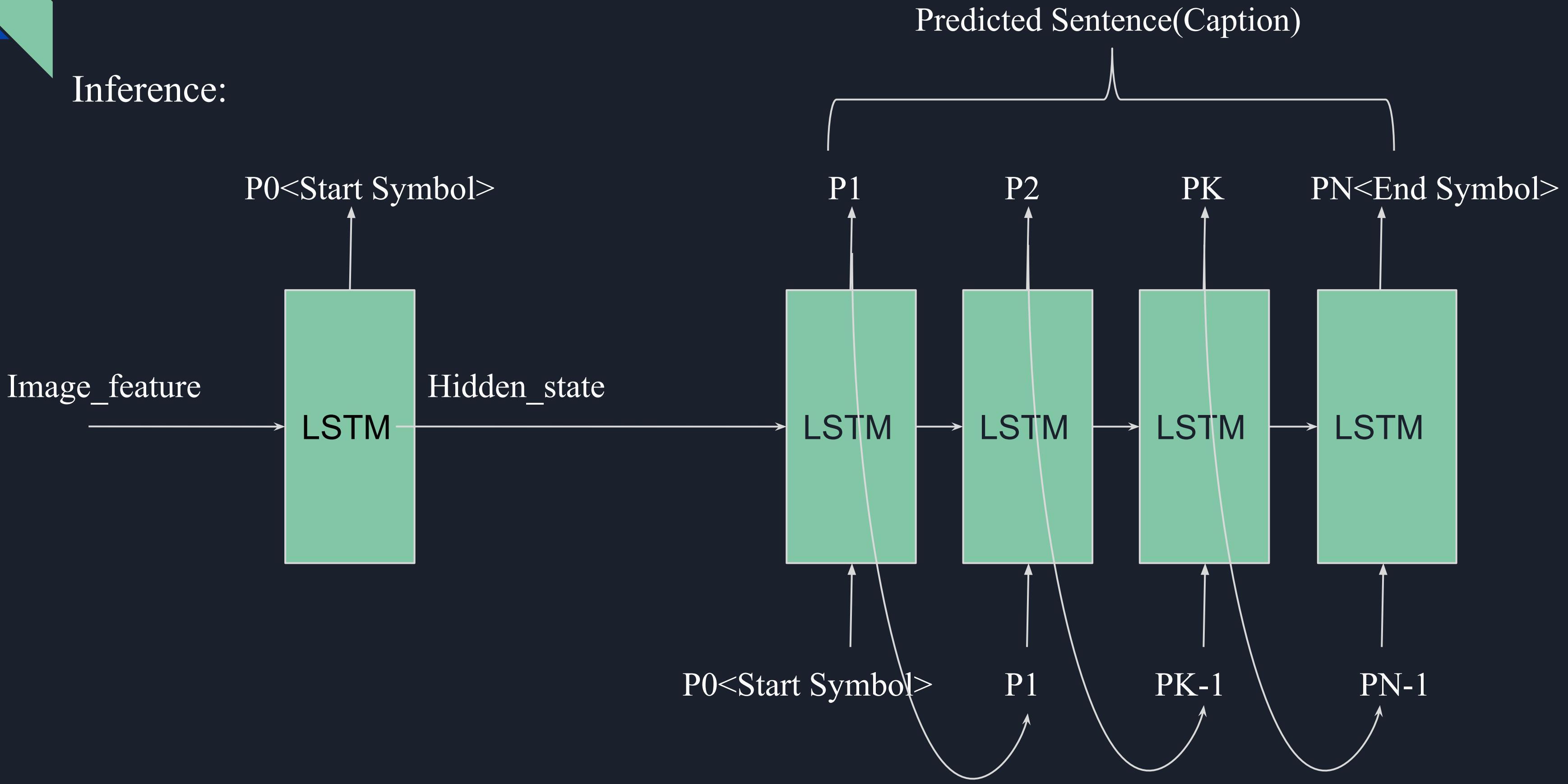
Flatten all network outputs and labels. Then calculate the loss on that ONE sequence.

The main point here is that we don't want to take into account the network output for padded elements and start flag.

[1]https://towardsdatascience.com/taming-lstms-variable-sized-mini-batches-and-why-pytorch-is-good-for-your-health-61d35642972e for customizing loss function

Implementation Details - RNN (Inference)

Inference:



Implementation Details - Inference

1. Sampling

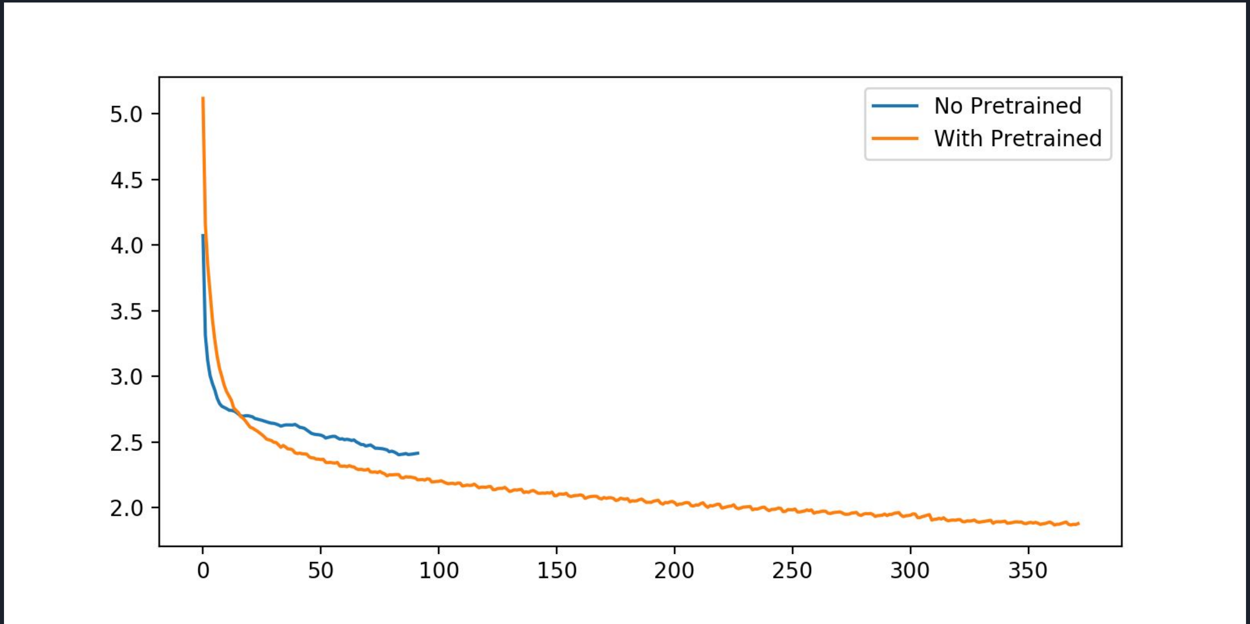
- Set a maximum caption length (20 in our implementation)
- Whenever generate an <end> symbol, stop the generating process

2. Beam Search

- Iteratively consider the set of the k (5 in our implementation) best sentences up to time t as candidates to generate sentences of size t+1
- Only keep the best k results

Results - Loss

We trained the model for 350 steps (Step size = 500 batches, each epoch has around 3200 batches) on MSCOCO Dataset (Using the machine NVIDIA Tesla V100 on Google Cloud Platform) . The loss plot looks like this:



Result - Generated Caption Example (Good one)

an old car parked in a field

an airplane is flying through the air

Result - Generated Caption Example (Good one)

a man in a hat and glasses

a giraffe standing in a grassy field

Result - Generated Caption Example (Related, Still Recognized key object)

a clock on the side of a wall

a cat sitting on a toilet seat

Result - Generated Caption Example (Totally Nonsense)

black and white photo of a man riding a bike

flowers sitting in a glass vase

Result - Generated Caption With Beam Search

(1) motorcycles are parked in a row
(2) rows of motorcycles parked in a lot
(3) motorcycles are parked along the street
(4) rows of motorcycles parked in a parking lot
(5) group of motorcycles parked in a parking lot

Result - Evaluation Scores

Metric	Our Model*	Result From Paper
BLEU_4	21.7%	27.7%
METEOR	20.9%	23.7%
CIDER	52.7%	85.5%

*Model trained on MSCOCO dataset and test on MSCOCO

Metric	Our Model*
BLEU_4	9.9%
METEOR	13.9%
CIDER	12.8%

*Test on Flickr30K dataset

*Calculated using the API provided by MSCOCO dataset

Thank You