Automatic Image Captioning

a Show and Tell implementation

Final Project for the **Advanced Machine Learning** course @UnimiB

Author
Nassim Habbash

Task

Input

Output

Dog running through snow

Image caption **generation** in an **end-to-end** fashion

- **Definition** and **implementation** of a generative model for caption generation
- **Tuning** and **analysis** of its performances

Dataset

Flickr8k

Set of images and captions collected by different Flickr groups to contain a variety of scenes and situations

- **8091** images with 5 captions each
- **40453** image-caption pairs
- Comes with standard holdout splits:
  - Train: **6000** images
  - Validation: **1000** images
  - Test: **1000** images

Preprocessing

Caption cleaning

- Lowercasing
- Punctuation removal
- Partial stopwords removal
- Alphanumeric strings removal

A dog swimming in the pool.

⇩

dog swimming in pool

Preprocessing

Original image

Preprocessed image

Image data augmentation

- Random crop to 224x224 pixels
- Random horizontal flip
- Normalization by mean and stdev of ImageNet's images

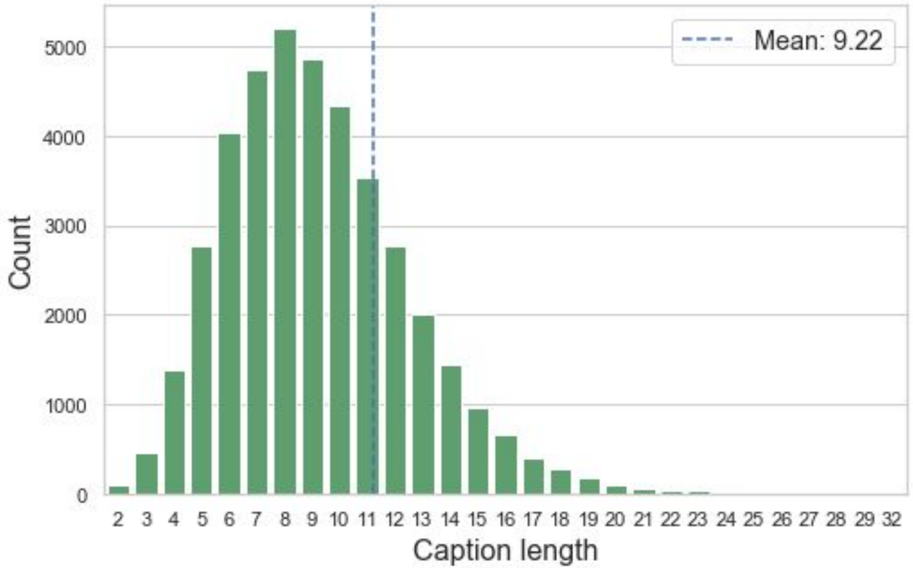
Note: augmentation is applied online

Exploratory Data Analysis

EDA Goal

Understand the **main trends** in the dataset

Caption length distribution

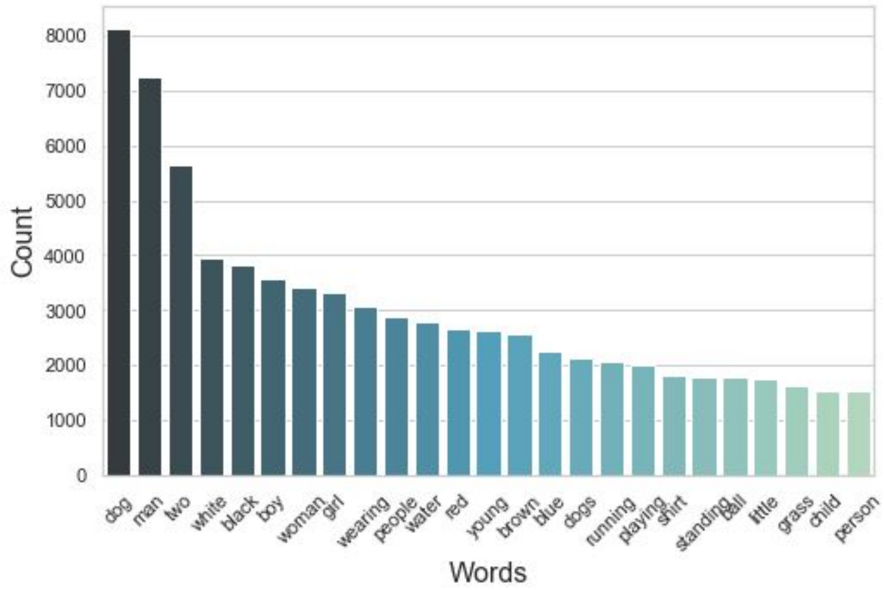


Exploratory Data Analysis

Caption terms wordcloud

Exploratory Data Analysis

Caption frequency distribution



EDA Results

The scenes in the dataset contain mostly **one or two subjects** engaging in **simple activities**, and are described by relatively **short sentences**

Model Architecture

Preliminary steps

Vocabulary building

- Word map of words to index
- Minimum word frequency cutoff
- Adds the <start>, <end> and <unk> tokens to the vocabulary

Caption encoding

- Transforms the caption into a variable size vector of numbers

dog swimming in pool

⇩

<start> dog swimming in pool <end>

⇩

0, 13, 49, 44, 99, 1

Model Architecture

$$\log p(C|I) = \sum_{t=0}^{N} \log \underbrace{p(C_t|I, C_0, ..., C_{t-1})}^{\textit{Sequence modeling!}} \qquad (1)$$

Given an image-caption pair, the model **maximizes** the **likelihood** of the **correct caption** given its image

Model Architecture

Encoder

$$I_{emb} = W_{ENC} \cdot CNN(I) \tag{2}$$

Decoder

$$x_{-1} = I_{emb} \tag{4}$$

$$p_t = LSTM(x_{t-1}) \tag{5}$$

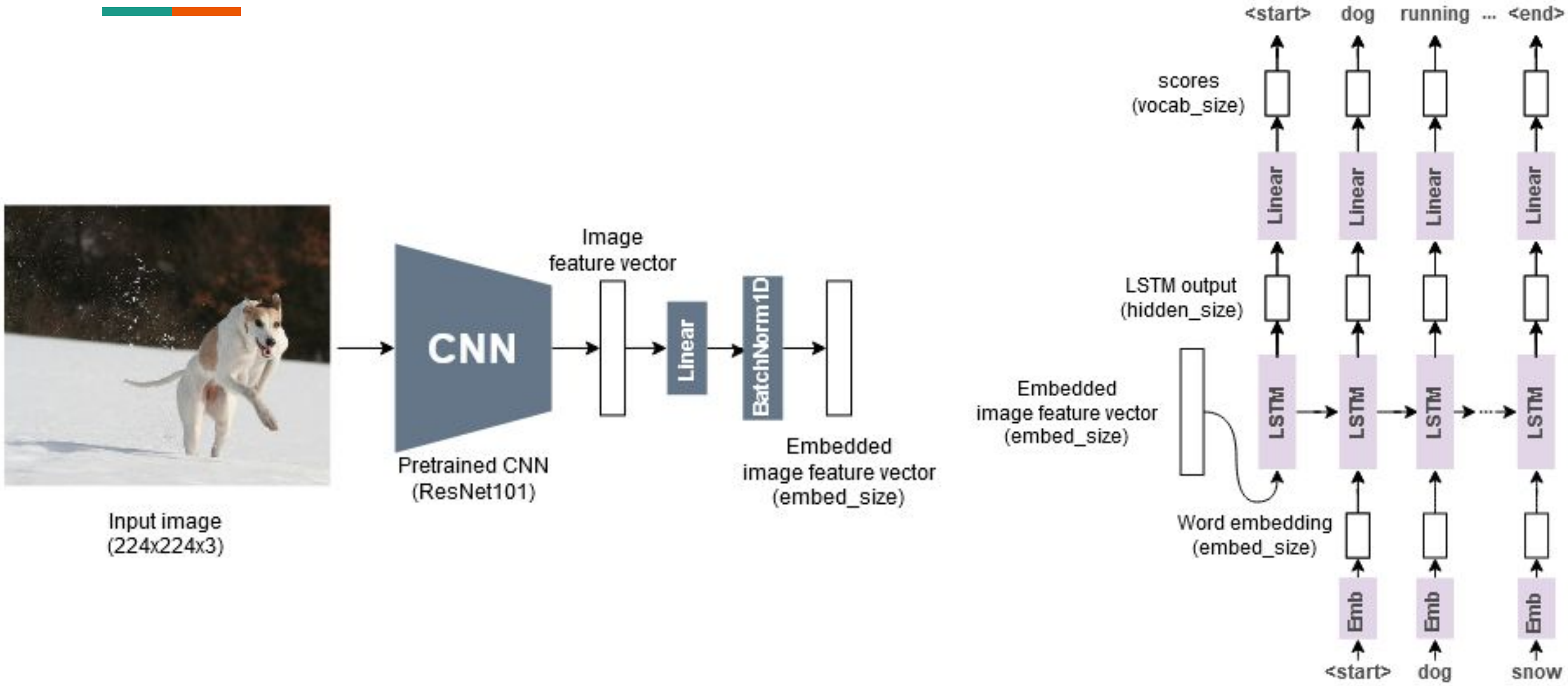
Loss function

$$L = -\sum_{t=1}^{N} log\ p_t(C_t) \tag{6}$$

How to **handle images** and **sequences together**?

By **embedding** the image features it into the same space!

Encoder-Decoder Architecture



[1] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell: A neural image caption generator,” 2014.

Inference

Greedy search

At each timestep, sample the caption token with the highest probability

Beam search

At each timestep, sample the top k caption tokens with the highest probability.

Returns the most likely sequence with the highest probability

Training and Validation

Mini-batching

Caption length sampling according to the length distribution in the dataset allows for fixed-size tensor batches (without padding)

Early Stopping with BLEU

Scoring the output captions to the ground-truths to ascertain training effectiveness with BLEU scores

If the BLEU doesn't increase for n epochs, stop the training process

Experiments

Implementation

Tools

- **PyTorch,** deep learning framework
- **Ax,** hyperparameter tuning
- **Weights and Biases,** experiment tracking

Baseline

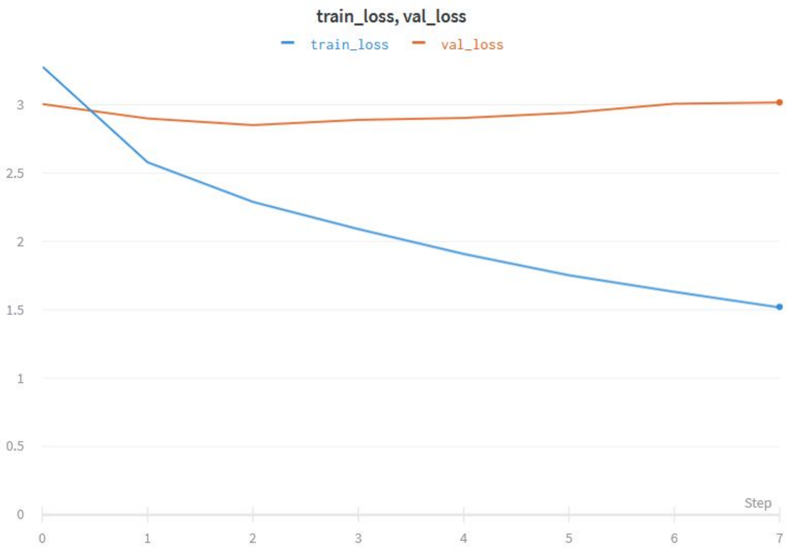
Hyperparameters

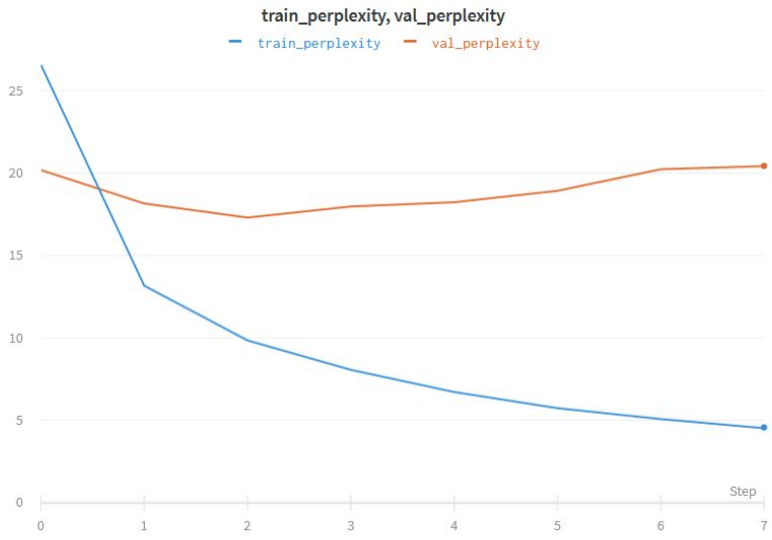
- Optimizer: Adam
- Learning rate: 0.001
- Momentum: 0.01
- Hidden size: 512
- Embed size: 512
- # Layers: 1
- Batch size: 32
- # Epochs: 10

Metrics

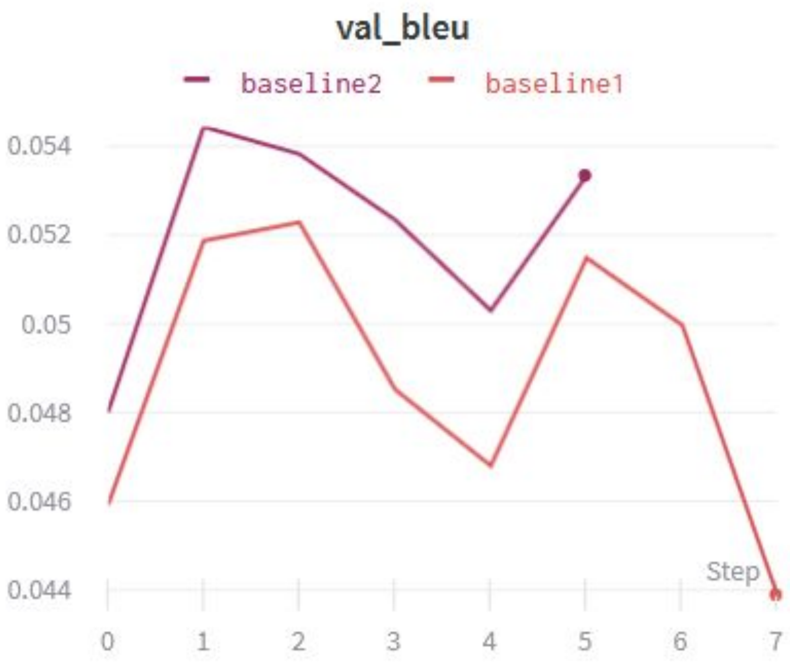
Weights and Biases dashboard

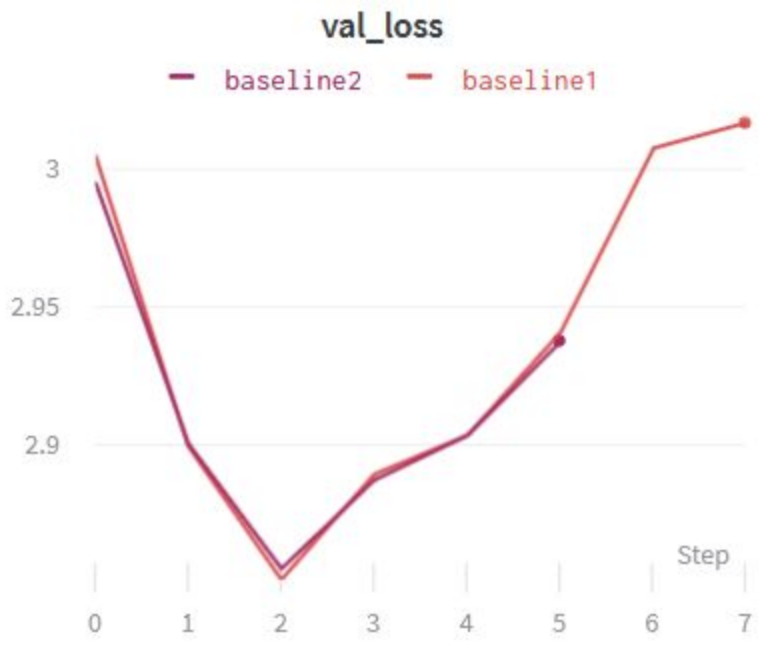
Baseline - Training metrics





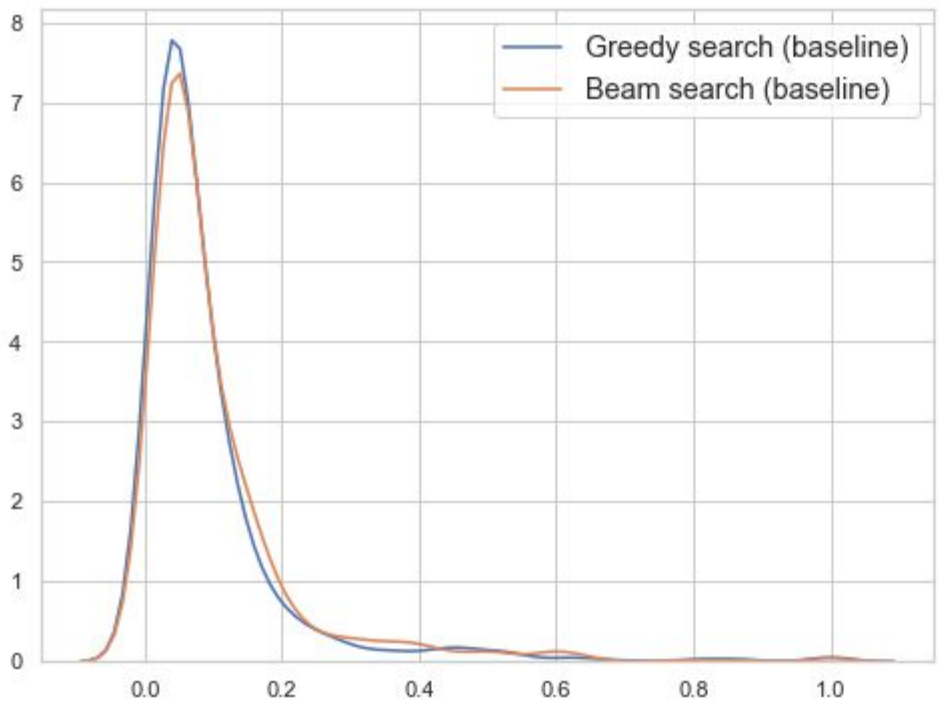
Baseline - Training metrics





Baseline - Test metrics

Beam search obtains a **marginal improvement** compared to Greedy search



Hyperparameter tuning

Search Space

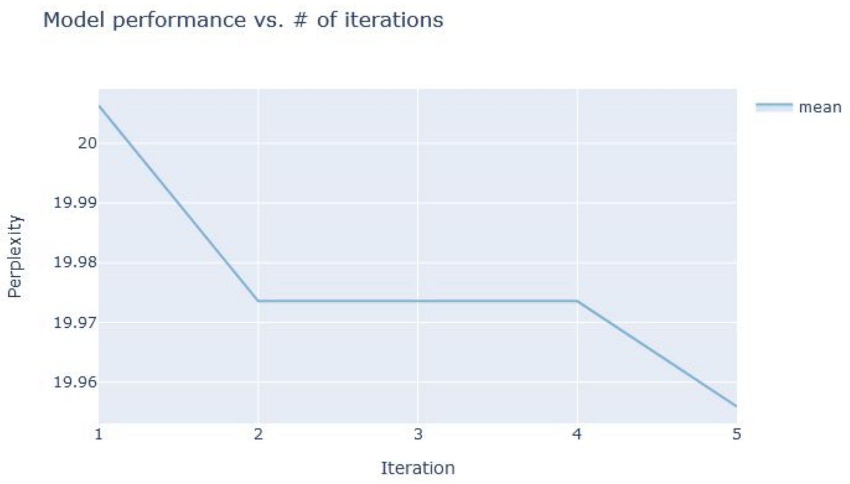
- Learning rate: [0.0005, 0.002]
- Momentum: [0.005, 0.02]
- Hidden size: {128; 256; 512}
- Embed size: {128; 256; 512}
- # Layers: {1; 2; 3}

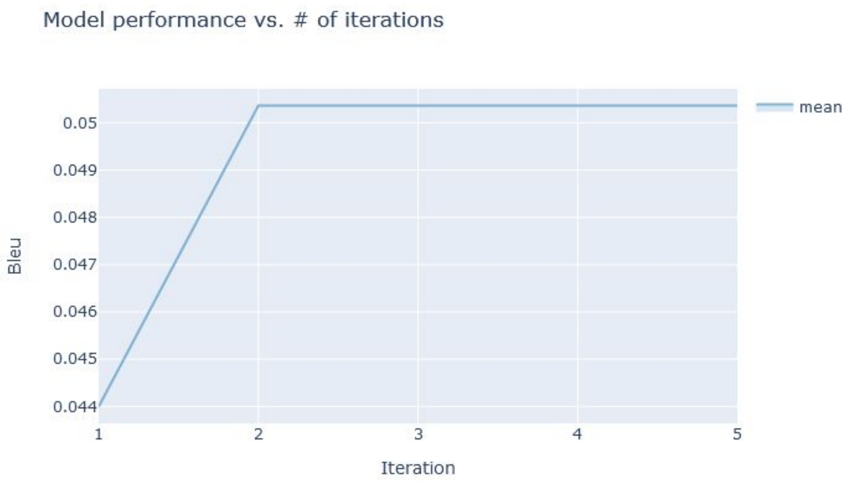
Model selected

- Sobol Sequences
- Budget: 5 trials

```
def _should_use_gp(search_space: SearchSpace, num_trials: Optional[int] = None) -> bool:
    """We should use only Sobol and not GPEI if:
    1. there are less continuous parameters in the search space than the sum of
    options for the choice parameters,
    2. the number of total iterations in the optimization is known in advance and
    there are less distinct points in the search space than the known intended
    number of total iterations.
    """
```

Hyperparameter tuning





Run #1: Perplexity minimization

Run #2: BLEU score maximization

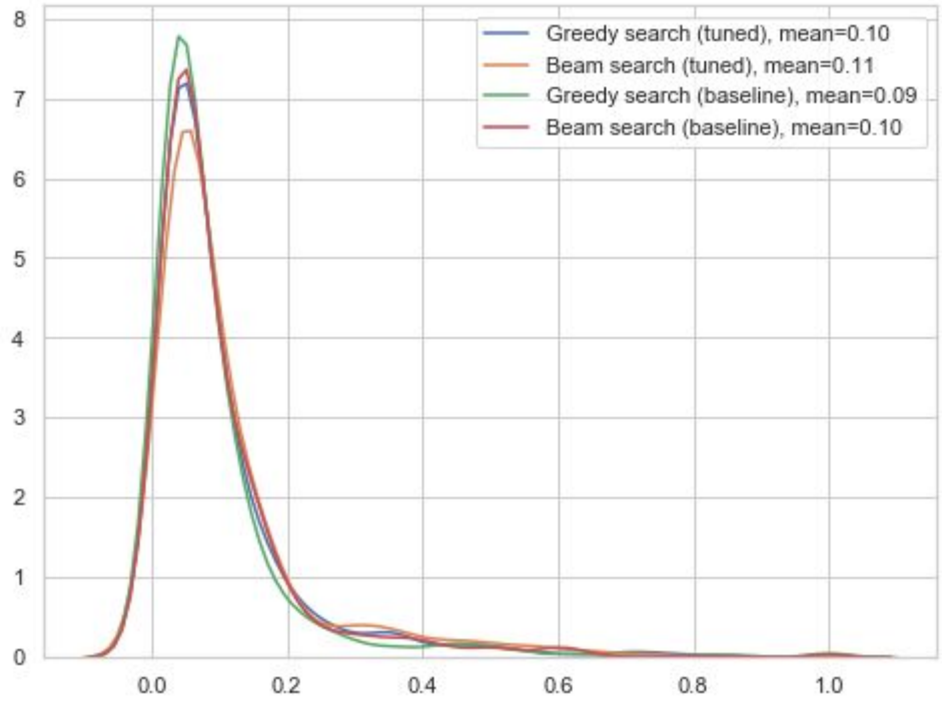
Hyperparameter tuning

Best configuration between runs

Run #	1	2
Embed Size	512	128
Hidden Size	256	512
LR	1.938e-3	5.577e-4
Momentum	1.626e-2	1.481e-2
# Layers	1	1
Perplexity	19.95	19.39
BLEU	0.044	0.05

Hyperparameter tuning - Test metrics

Hyperparameter tuning brings a **marginal improvement** to both Greedy search and Beam search compared to the baseline



Results

Presented model

Best BLEU score: 11

Dataset: Flickr8k (8k images)

Short hyperparameter tuning

Limited hardware capabilities

Original paper model

Best BLEU score: 27.2

Dataset: MSCOCO (330k images)

(Possibly) extensive hyperparameter tuning

Better hardware available

Considerations

- Every training instance seem to get trapped into a **local minima** (0.05 BLEU/19 Perplexity)
- Better **exploration** of the search space during tuning might improve optimum convergence
- **Changes** in the architecture (additional dropout layers) might improve **overfitting**
- Better yet, training on a **bigger dataset** might solve all the issues altogether

- A more structured and effective **experiment tracking** workflow might solve a lot of headaches

Considerations

Man and woman pose for picture

BLEU = 0.0

BLEU scores **correlate** with human judgement **but are not a completely reliable** validation measure alone

Man is on the ground by his arms trees

BLEU = 0.0

Conclusions

- Encoder-decoder architectures show great effectiveness in **cross-domain translations**
- Caption generation is a field that has seen rapid progress in recent years, as encoder-decoder architectures have already been **outclassed**
- AutoML pipelines added to complex models make training extremely **computationally expensive,** but progress is being made towards less taxing/same performance models

Repository: *https://github.com/nhabbash/autocaption*