

# Image Captioning Generator

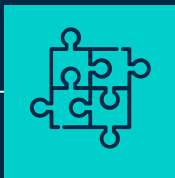
By  
Sahith Damera  
Prudhviraaj Boddu

# CONTENTS

Here's what you'll find

1. Decoding the Challenge
2. Introduction to Image Captioning Generator
3. Understanding the problem
4. Essential Building Blocks
5. From Pixels to Text
6. Training Process
7. Evaluation
8. Implementation
9. Future Work and Conclusion

# Decoding the Challenge



01

## PROBLEM & SOLUTION

Describing Images  
From Silent Images  
to Descriptive  
Narratives



02

## OUR PROCESS

Tokenizing and  
extracting image  
features with  
ResNet-18 and  
Inspection V3



03

## TARGET

Fine-Tuning  
and  
Adaptability

# Introduction

What do you see in the picture?

Image captioning involves generating descriptive text to describe the content of an image

In the era of AI, understanding images is crucial for machines to interact with the visual world. Image captioning bridges the gap between visual data and natural language



# UNDERSTANDING THE PROBLEM

## Generating Captions for Images

The central challenge lies in developing models that can effectively transform visual information into coherent and relevant textual descriptions. This requires overcoming complexities in understanding the context and semantics of images.

We are addressing the challenge by conducting a comparative analysis between ResNet-18 and Inception V3 to determine the superior solution



# Essential Building Blocks

**Data  
Preprocessing**  
For  
Tokenization



**Implementation**  
Using Transformer  
based Captioning  
model for text



**Image Feature  
Extraction**  
Using  
ResNet-18

**Deploying**  
Creating a Flask  
Application

# From Pixels to Text

## Data Preprocessing

This involves tokenization and vocabulary building. Here, we leverage the capabilities of ResNet-18 to extract rich image features and train a Transformer-based model for caption generation.

## Data Exploration

Exploring sample captions and associated images in Flickr 8k dataset. This step provides insights into the preprocessing techniques and the diversity of image content we are working with.

## Image Feature Extraction

The selection of ResNet-18 for image feature extraction is pivotal. Its deep architecture captures intricate visual patterns, enhancing our model's ability to understand and interpret visual content effectively.

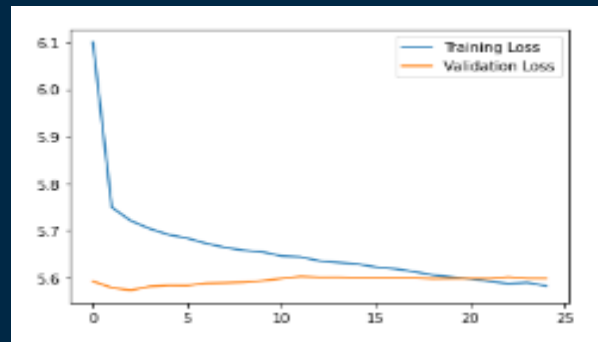
## Model Architecture

Our model adopts a Transformer-based architecture, a cutting-edge approach for handling sequential data. Incorporating positional encoding and attention mechanisms,

# Training Process

Training involves an iterative process, optimizing our model using the Adam optimizer and minimizing the Cross Entropy loss. Techniques to address challenges like sequence padding are employed, ensuring robust learning.

```
Epoch -> 20 Training Loss -> 5.597502708435059 Eval Loss -> 5.598044395446777  
Epoch 21: reducing learning rate of group 0 to 2.6214e-06.  
Epoch -> 21 Training Loss -> 5.591978073120117 Eval Loss -> 5.597818851470947  
Epoch -> 22 Training Loss -> 5.586987018585205 Eval Loss -> 5.6010541915893555  
Epoch -> 23 Training Loss -> 5.5891523361206055 Eval Loss -> 5.598156452178955  
Epoch 24: reducing learning rate of group 0 to 2.0972e-06.  
Epoch -> 24 Training Loss -> 5.582265377044678 Eval Loss -> 5.59806489944458
```





# Evaluation

The Model generating captions that closely align with ground truth annotations. Performance metrics, including BLEU scores, highlight the effectiveness of our approach. Visual comparisons further validate the quality of generated captions."

Training

Phase-I



Learning the  
Features of data

Validation

BLEU Score



metric used for  
evaluating the quality  
of machine-generated  
text

Testing

Evaluation



Generated relevant  
Caption's for Image

# Implementation – Gradio App

```
22 max_seq_len = 33
23 vocab_size = 8360
24
25 class PositionalEncoding(nn.Module):
26
27     def __init__(self, d_model, dropout=0.1, max_len=max_seq_len):
28         super(PositionalEncoding, self).__init__()
29         self.dropout = nn.Dropout(p=dropout)
30
31         pe = torch.zeros(max_len, d_model)
32         position = torch.arange(0, max_len, dtype=torch.float).unsqueeze(1)
33         div_term = torch.exp(torch.arange(0, d_model, 2).float() * (-math.log(10000.0) / d_model))
34         pe[:, 0::2] = torch.sin(position * div_term)
35         pe[:, 1::2] = torch.cos(position * div_term)
36         pe = pe.unsqueeze(0)
37         self.register_buffer('pe', pe)
38
39
40     def forward(self, x):
41         if self.pe.size(0) < x.size(0):
42             self.pe = self.pe.repeat(x.size(0), 1, 1).to(device)
```

PS D:\NLP\ImageCaptionGenerator> C:/Users/prudh/miniconda3/Scripts/activate  
PS D:\NLP\ImageCaptionGenerator> conda activate D:\conda\envs\ml  
(ml) PS D:\NLP\ImageCaptionGenerator> & D:/conda/envs/ml/python.exe d:/NLP/ImageCaptionGenerator/app.py  
D:\conda\envs\ml\lib\site-packages\torchvision\models\\_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.  
warnings.warn(  
D:\conda\envs\ml\lib\site-packages\torchvision\models\\_utils.py:223: UserWarning: Arguments other than a weight enum or 'None' for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is equivalent to passing 'weights=ResNet18\_Weights.IMAGENET1K\_V1'. You can also use 'weights=ResNet18\_Weights.DEFAULT' to get the most up-to-date weights.  
warnings.warn(msg)  
Running on local URL: http://127.0.0.1:7860  
  
To create a public link, set 'share=True' in 'launch()'.

# Future Work and Conclusion



In Future, we work could explore additional architectural enhancements or the utilization of diverse datasets to further elevate between different model's performance.

In conclusion, our work successfully integrates ResNet-18 features with a Transformer-based model for image captioning.



Do you have any questions?

[sdame1@unh.newhaven.edu](mailto:sdame1@unh.newhaven.edu)

[pboddu@unh.newhaven.edu](mailto:pboddu@unh.newhaven.edu)

# THANKS

