Neural Network based Image & Query Search

Group 9

Shrinidhi Bhide,
Amisha Kelkar, Yashna Meher

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Motivation

- Natural language search is the easiest way people look for images today (shopping, media, etc.)
- Current solutions often depend heavily on pretrained models (like CLIP, BERT)
- Many companies need private, secure image search systems for internal photos
- Building from scratch (without pretrained models) gives deeper understanding of how multimodal systems work
- Future extension: Multilingual search and image-to-image search for broader usability

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Data

Flickr30k dataset contains:

- **31,000 images** collected from Flickr
- 5 reference sentences provided by human annotators.

Pre-processing:

- Resized and padded all images
- Tokenized the captions and padded them to uniform length
- Built a vocabulary

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Objective

Train a neural network from scratch to link images and captions

{01}

Explore different strategies for effective multi-modal learning

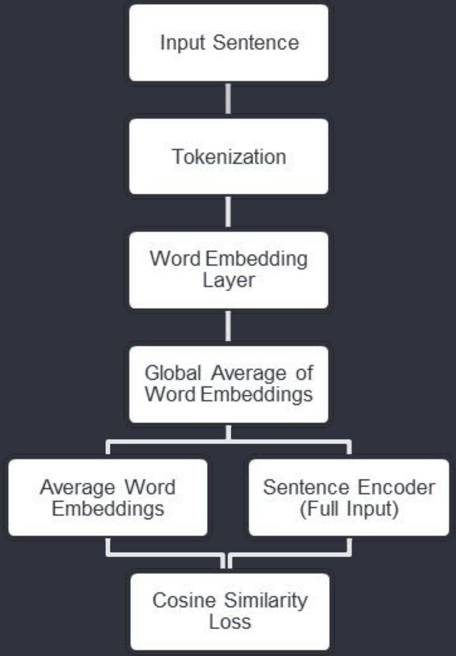
{02}

Achieve semantic retrieval capability with limited resources

{03}

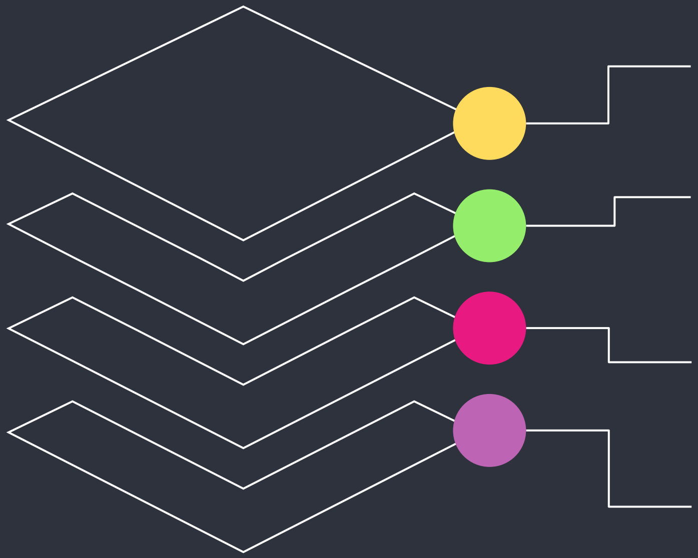
Trial 1: Caption-Only Retrieval

- Train a model that embeds captions like Bag of Words
- Find most similar caption for a query caption
- Search images based on caption match only



1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Trial 1: List of Basic Layers



Embedding Layer
Turns word tokens into dense vectors

Average Pooling
Averages word embeddings into a sentence vector

Dense Layer
Maps pooled sentence vector into learned embedding space

Cosine Distance Loss
Trains model to pull matching sentences together

Trial 1: Caption-Only Retrieval

- Sentence embeddings S
- Combined word embeddings $\hat{S} = 1/n \sum_{i=1}^{n} E(W_i)$
- Loss function $Loss = 1 - cosine\_similarity(S, \hat{S})$

Demo

Let’s take a look at how the model is performing

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

Trial 2: Introducing image embeddings

- Building on the model from Trial 1, we introduced image embeddings and used cosine similarity to make them closer to the caption embeddings from Trial 1.
- This was done initially using 2D CNNs, but it didn’t perform very well. We had to use Resnet to extract richer feature embeddings.
- We used a transformer model with cross attention. It took a caption (text) and image features (from CNN) and aligned them using attention.
- It learnt to focus on the most important parts of the image for a given caption.

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Our CNN architecture

Conv2D + BatchNorm + ReLU [Image]

Extract low- and mid-level visual features from image pixels

MaxPooling

Reduce spatial dimensions & retain dominant features early in the network

Residual Blocks [ConvBlock + IdentityBlock]

Deep learning via shortcut connections

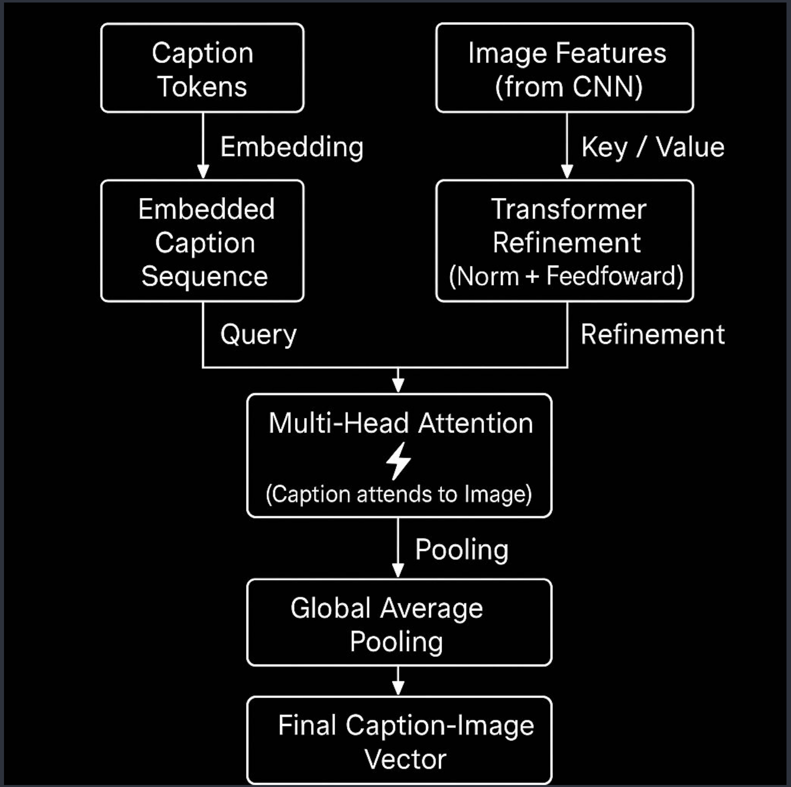
Final Output: 7×7×2048 Feature Map

Captures rich high-level visual features with deep receptive fields

Patch Flattening → [49, 2048]

Reshape image feature maps into patch sequences for our transformer

Our Transformer architecture



- Transformer model with cross modal attention

  Imagine the word “dog” in a caption. The attention mechanism enables the encoder to focus on image regions that likely contain a dog.

- Contrastive loss was used to bring similar image and caption embeddings together.

Transformer with YOLO

- Images features were still not getting learnt well enough
- Enter YOLO: You Only Look Once
  a. Localization Loss: Corrects bounding box coordinates.
  b. Confidence Loss: Detects object presence (object vs background).
  c. Classification Loss: Identifies object class (dog, tree, car, etc.).

$$L = \lambda_{loc} L_{loc} + \lambda_{conf} L_{conf} + \lambda_{cls} L_{cls}$$

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Trial 3: Query to Joint Embeddings

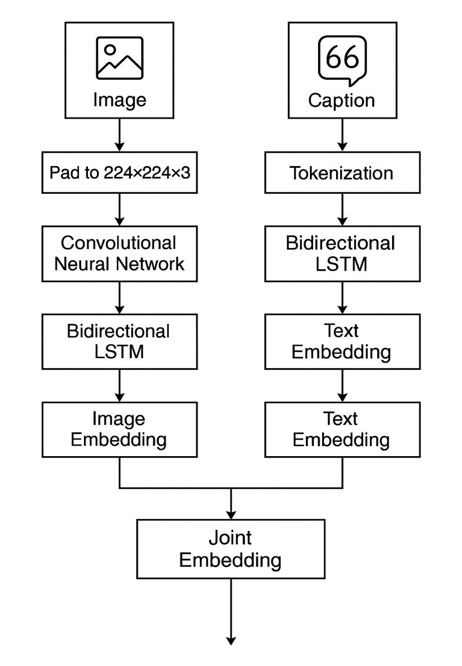


Image and text are processed independently, allowing flexible inputs during inference

Bidirectional LSTM improves caption understanding by capturing both past and future word context

Padding ensures consistent image shape, which is critical for batch training

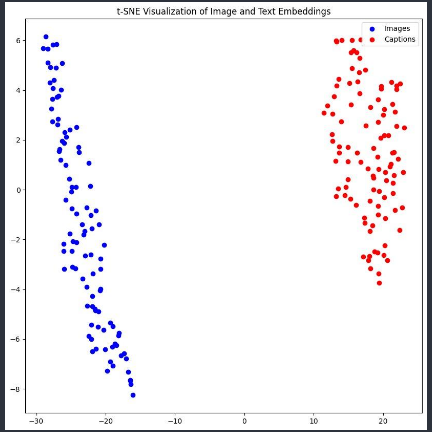
Joint embedding enables cross-modal retrieval

Model is trained using cosine similarity loss, helping align semantically similar inputs

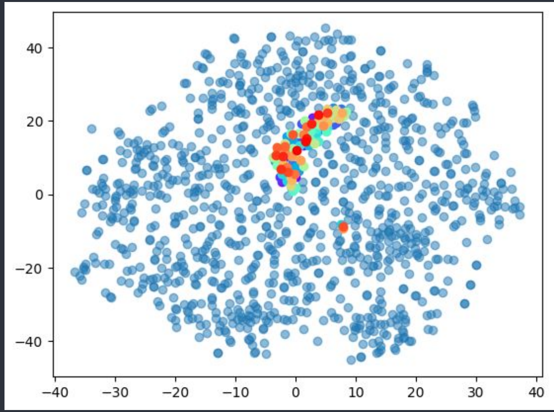
1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

Trial 3: Query to Joint Embeddings

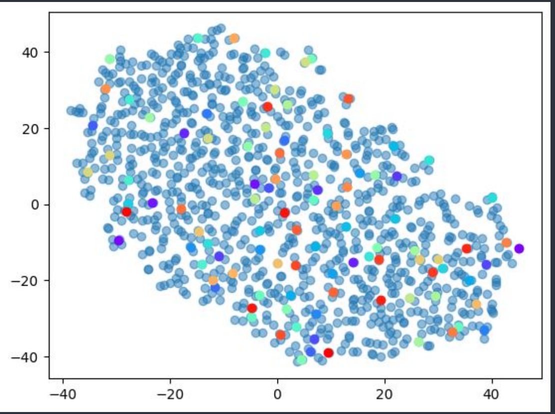
Text and image embeddings in space



Not in the same space



In the same space but not distributed well



Well distributed joint embeddings

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

Future: CLIP

CLIP (Contrastive Language-Image Pretraining) learns to link images and texts

It encodes entire images and captions into dense embedding vectors

Training Objective: Pull matching image-text pairs closer and push non-matching pairs apart

Focuses on global meaning: understands the overall scene, not specific objects

No object detection: Doesn't predict bounding boxes or classify individual items

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

YOLO vs CLIP

If you use YOLO	If you use CLIP
You detect all objects (e.g., "dog", "car", "tree") in an image first	You encode the entire image as a "scene meaning"
Then match detected objects to caption words	Then match full caption meaning to full image meaning
Needs multi-class loss if many objects must match caption	Needs contrastive loss between image and caption embeddings
Easier to control (you know what was detected)	Hard to control (latent meaning)
Good for multi-label captions ("a cat and a dog")	Good for overall descriptions ("a sunny day at the beach")

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1

Future: List of YOLO Layers

- **YOLO Backbone** — Detect objects and extract bounding boxes
- **Object Embedding Layer** — Map detected object classes into dense vectors
- **Multi-class Dense Projection** — Combine multiple object vectors into one image-level vector
- **Caption Encoder (Embedding + LSTM)** — Same caption processing as earlier
- **Multiclass Contrastive Loss** — Match object-rich image vectors with text vectors allowing multi-label matching

Future: List of CLIP Layers

CLIP Image Encoder

Encodes whole image into a single semantic embedding

CLIP Text Encoder

Encodes query caption into a matching embedding

L2 Normalization Layer

Normalizes both embeddings

Cosine Similarity Layer

Calculates similarity between text and image embeddings

Contrastive Loss
[already pre-trained]

Fine-tuning or zero-shot matching

Overall Challenges

{01}

Data scarcity

hard to generalize on small custom datasets

{02}

Efficient training

batch handling, augmentation required

{03}

Multi-modal alignment

tricky without heavy pretraining

{04}

Loss balancing

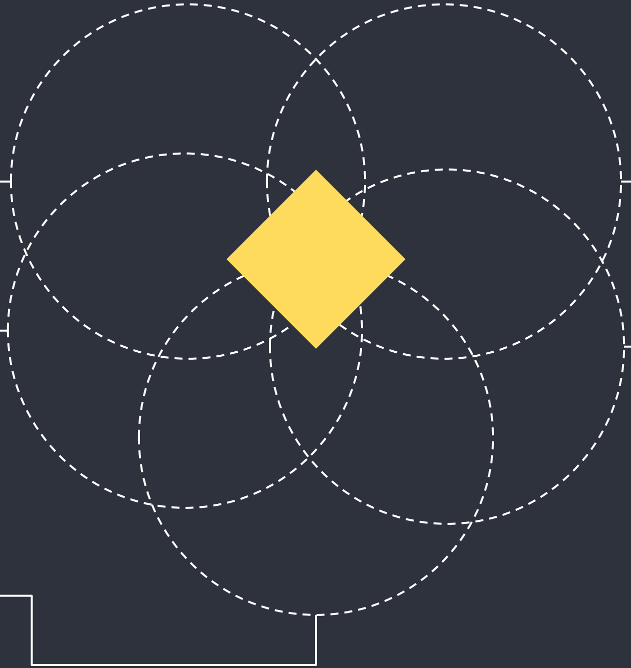
simple cosine loss not always sufficient

Future Work

Scale to larger datasets with more diverse captions

Explore hard negative mining for better contrastive learning

Data Augmentation for better image recognition



Experiment with multimodal transformers (like CLIP-style)

Real-world deployment for caption-based image search

Thank You

Questions?

1 0 1 1   0 1 1   0 1   1 0 1 1 0 0 1   1 0   1 1 0 1 1   0 1 1   0 1   1 1 0 1 1 0   1 1 0 1 1 1   1 1 0 1