CATR: TRANSFORMER-BASED IMAGE CAPTIONING USING PYTORCH

Sahith Reddy Thummala
Dept. of Computer Science
Georgia State University
Atlanta, Georgia

Manish Yerram
Dept. of Computer Science
Georgia State University
Atlanta, Georgia

***Abstract*—This project documents the development of an image captioning system powered by CATR (Caption Transformer)—a transformer-based model designed for vision-to-language tasks. Built using PyTorch and trained on the COCO 2017 dataset, CATR generates coherent and descriptive captions for input images. The application features an interactive interface that allows users to upload images and receive real-time textual descriptions. To enhance accessibility and user experience, the generated captions are converted into natural-sounding audio using gTTS (Google Text-to-Speech). Additionally, this report explores the transformer-based architecture of CATR, emphasizing its encoder-decoder structure and attention mechanisms, offering insights into the capabilities of modern vision-language models.**

***Index Terms*—Image Captioning, Transformer Architecture, Computer Vision, Natural Language Processing, PyTorch, Attention Mechanism, Multimodal AI.**

I. Summary

This project documents the development of an image captioning system powered by CATR (Caption Transformer)—a transformer-based model designed for vision-to-language tasks. Built using PyTorch and trained on the COCO 2017 dataset, CATR generates coherent and descriptive captions for input images. The application features an interactive interface that allows users to upload images and receive real-time textual descriptions. To enhance accessibility and user experience, the generated captions are converted into natural-sounding audio using gTTS (Google Text-to-Speech). Additionally, this report explores the transformer-based architecture of CATR, emphasizing its encoder-decoder structure and attention mechanisms, offering insights into the capabilities of modern vision-language models.

II. Introduction

Multimodal AI systems integrate vision and language to enable richer interactions with visual content. These systems are widely used in accessibility tools, educational platforms, and creative applications. This project implements an end-to-end pipeline using the CATR (Caption Transformer) model and gTTS (Google Text-to-Speech), enabling users to upload images, generate descriptive captions, and convert them into audio. CATR, built on a transformer architecture and trained on the COCO 2017 dataset, produces accurate, context-aware image descriptions. The integration with gTTS provides a seamless experience by adding speech output to the visual captioning pipeline.

III. Objectives

The project objectives are as follows:

1) **Generate Descriptive Captions:** Leverage the CATR model to produce accurate and context-rich captions for user-uploaded images.
2) **Audio Synthesis:** Use gTTS to convert generated captions into clear, natural-sounding speech to improve accessibility.
3) **Efficient Resource Utilization:** Optimize model performance through lightweight architecture and minimal dependencies for smooth execution on both GPUs and CPUs.
4) **Interactive User Interface:** Develop a simple and responsive UI that allows users to upload images and receive audio-visual outputs seamlessly.
5) **Applications in Accessibility and Education:** Support visually impaired users and aid in visual-language learning through multimodal feedback.

IV. Flamingo Model

Flamingo, developed by DeepMind, is a family of Visual Language Models (VLMs) designed for few-shot learning in multimodal tasks, combining visual and textual data understanding. The model bridges pretrained vision-only and language-only architectures, enabling it to handle interleaved sequences of images, videos, and text. At its core, Flamingo incorporates innovative components like the Perceiver Resampler, which reduces high-dimensional visual data into fixed-length representations, and GATED XATTN-DENSE layers that seamlessly integrate visual features into a frozen large language model (LM) for open-ended text generation.

The model's design allows for rapid adaptation to diverse vision-language tasks—such as visual question-answering, captioning, and visual dialogue—using just a few task-specific examples without the need for fine-tuning. This capability is achieved through multimodal in-context learning, where Flamingo is prompted with example image-text or video-text pairs to guide its understanding of a task, making it versatile and effective for both simple and complex queries.

A. Key Features of Flamingo

- **Pretrained Vision and Language Models:** Flamingo leverages pretrained vision encoders and large language models, allowing efficient multimodal understanding.
- **Cross-Attention Mechanisms:** It integrates vision and language modalities using cross-attention layers, enhancing the coherence of generated outputs.
- **Few-Shot Learning:** Flamingo excels in few-shot learning, adapting to new tasks with minimal fine-tuning.

V. Architecture of CATR

CATR is a Transformer-based image captioning model inspired by the original "Attention is All You Need" architecture, but adapted for vision-to-language tasks. It follows a standard encoder-decoder structure:

A. Components of CATR Architecture

1) **ResNet-50 Backbone:** Extracts high-level visual features from the input image.
2) **Projection Layer:** Converts spatial feature maps into transformer-compatible embeddings.
3) **Transformer Encoder:** Processes image embeddings to capture contextual relationships between visual regions.
4) **Transformer Decoder:** Generates caption tokens by attending to both the encoded image features and previously generated words.
5) **Masked Self-Attention:** Allows the decoder to attend only to earlier tokens, preserving sequence generation logic.
6) **Cross-Attention:** Enables the decoder to focus on relevant image features while generating each word.
7) **Feedforward Network:** Applies non-linear transformations to enhance representation in both encoder and decoder layers.
8) **Output Linear Layer + Softmax:** Maps decoder outputs to a vocabulary distribution for caption word prediction.

B. Working Process

The captioning process follows this sequence:

Input Image
↓
ResNet-50 → Visual Feature Embeddings
↓
Transformer Encoder → Contextualized Representations
↓
Transformer Decoder ← [START Token + previous words]
↓
Linear + Softmax → Caption words generated one by one

C. Architecture Diagram

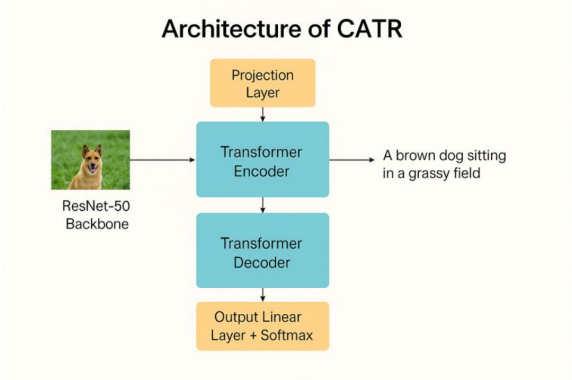

Fig. 1. Architecture of the CATR model showing the flow from input image through ResNet-50, transformer encoder-decoder, to output caption

D. CATR Variants

CATR has multiple variants tailored for different use cases:

- **CATR v1:** Baseline transformer model with standard training setup and architecture.
- **CATR v2:** Enhanced version with improved training strategies and better caption quality.
- **CATR v3:** Most refined and accurate variant, offering the best performance and fluency in caption generation.

VI. Dataset: MS COCO 2017

The CATR model is trained and evaluated using the MS COCO 2017 dataset (Microsoft Common Objects in Context), which is a benchmark dataset widely used for image captioning, object detection, and segmentation tasks. It includes over 123,000 images, each paired with five human-annotated captions, providing rich visual-linguistic associations. The dataset is split into train2017 (∼118K images) and val2017 (∼5K images) along with their respective caption annotations stored in JSON files.

In the CATR project, the COCO dataset is fetched manually by downloading image and annotation zip files from the official website. After downloading, they are extracted into a structured directory format typically expected by PyTorch Datasets: images in folders like train2017/ and val2017/, and annotations in a folder named annotations/. The dataset is then loaded using a custom COCO dataset class or wrapper that extends torch.utils.data.Dataset. During training, image transformations such as resizing, normalization, and padding are applied, while captions are tokenized and batched with corresponding attention masks.

VII. Technical Implementation

The application integrates the following technologies:

- **PyTorch:** Deep learning framework used to build and train the CATR transformer model.
- **TorchVision:** Provides pretrained ResNet backbone and image transformation utilities.
- **ResNet-50:** CNN architecture used as a feature extractor for input images.
- **Transformer:** Core architecture for generating captions from visual embeddings.
- **COCO Dataset:** Benchmark dataset for training and evaluating image captioning models.

- **pycocotools:** Utility library to load COCO annotations and evaluate model performance.
- **TQDM:** Used to display progress bars during training and evaluation.
- **Torch Hub:** Enables easy loading of pretrained CATR model variants (v1, v2, v3).
- **gTTS:** Converts generated text captions into natural-sounding audio.
- **Streamlit:** Provides an interactive interface for testing and visualizing the model.

VIII. Quantized Model

To improve efficiency, especially on hardware without GPUs, we implemented model quantization:

- **Preparation:** The model is first configured to support quantization by defining a quantization strategy.
- **Calibration:** A small batch of sample data is passed through the model to estimate activation ranges.
- **Conversion:** Once calibrated, the model's operations are converted into their quantized (INT8) equivalents.
- **Deployment:** The resulting quantized model runs more efficiently, especially on hardware without GPUs.

IX. Sample Code

A. CATR Model Architecture Implementation

```
class CATR(nn.Module):
    def __init__(self, config):
        super().__init__()
        # Visual backbone (ResNet-50)
        self.backbone = models.resnet50(pretrained=True)
        self.backbone = nn.Sequential(*list(self.backbone.children())[:-2])
        # Projection layer
        self.proj = nn.Conv2d(2048, config.hidden_dim, kernel_size=1)
        self.position_encoding = PositionalEncoding(
            config.hidden_dim, dropout=config.dropout
        )
        # Transformer encoder
        encoder_layer = nn.TransformerEncoderLayer(
            d_model=config.hidden_dim,
            nhead=config.nheads,
            dim_feedforward=config.ff_dim,
            dropout=config.dropout
        )
        self.transformer_encoder = nn.TransformerEncoder(
            encoder_layer,
            num_layers=config.enc_layers
        )
        # Transformer decoder
        decoder_layer = nn.TransformerDecoderLayer(
            d_model=config.hidden_dim,
            nhead=config.nheads,
            dim_feedforward=config.ff_dim,
            dropout=config.dropout
        )
        self.transformer_decoder = nn.TransformerDecoder(
            decoder_layer,
            num_layers=config.dec_layers
        )
        # Word embedding and output layer
        self.word_embed = nn.Embedding(config.vocab_size, config.hidden_dim)
        self.output_layer = nn.Linear(config.hidden_dim, config.vocab_size)
        # Initialize parameters
        self._init_parameters()
```

Fig. 2. CATR model architecture implementation showing the transformer-based structure with ResNet backbone

B. Inference Pipeline

```
def generate_caption(model, image_path, transform, tokenizer, device="cuda"):
    """Generate caption for an input image using the CATR model."""
    # Load and preprocess the image
    image = Image.open(image_path).convert("RGB")
    image = transform(image).unsqueeze(0).to(device)

    # Set model to evaluation mode
    model.eval()

    # Generate caption
    with torch.no_grad():
        output = model(image)

    # Decode caption
    predicted_ids = output[0].argmax(-1)
    predicted_caption = tokenizer.decode(predicted_ids.tolist())

    # Clean special tokens
    cleaned_caption = clean_caption(predicted_caption)
    return cleaned_caption

def text_to_speech(caption, output_path, lang="en"):
    """Convert caption to speech using gTTS."""
    tts = gTTS(text=caption, lang=lang)
    tts.save(output_path)
    return output_path

def process_image(image_path, output_dir="outputs", model=None,
                  transform=None, tokenizer=None, device="cuda"):
    """Process an image through the full pipeline."""
    # Ensure model and dependencies are loaded
    if model is None:
        model, transform, tokenizer = load_model(device)

    # Create output directory if it doesn't exist
    os.makedirs(output_dir, exist_ok=True)

    # Generate caption
    start_time = time.time()
    caption = generate_caption(model, image_path, transform, tokenizer, device)
    caption_time = time.time() - start_time
    # Convert to speech
    audio_path = os.path.join(output_dir,
                              f"{os.path.basename(image_path).split('.')[0]}.mp3")
    start_time = time.time()
    text_to_speech(caption, audio_path)
```

Fig. 3. Inference pipeline showing functions for generating captions, text-to-speech conversion, and image processing

C. Training Pipeline

```
def train_one_epoch(model, criterion, data_loader, optimizer,
                    lr_scheduler, device, epoch, clip_max_norm):
    model.train()
    criterion.train()
    metric_logger = MetricLogger(delimiter="  ")
    metric_logger.add_meter('lr', SmoothedValue(window_size=1, fmt='{value:.6f}'))
    header = 'Epoch: [{}]'.format(epoch)
    for images, captions, cap_masks in metric_logger.log_every(data_loader, 10, header):
        # Move to device
        images = images.to(device)
        captions = captions.to(device)
        cap_masks = cap_masks.to(device)
        # Forward pass
        outputs = model(images, captions[:, :-1], cap_masks[:, :-1])
        # Calculate loss
        loss = criterion(
            outputs.permute(1, 2, 0),
            captions[:, 1:].permute(1, 0)
        )
        # Backward pass
        optimizer.zero_grad()
        loss.backward()
        # Gradient clipping
        if clip_max_norm > 0:
            torch.nn.utils.clip_grad_norm_(model.parameters(), clip_max_norm)
        optimizer.step()
        # Update learning rate
        lr_scheduler.step()
        # Update metrics
        metric_logger.update(loss=loss.item())
        metric_logger.update(lr=optimizer.param_groups[0]["lr"])
    return metric_logger.meters["loss"].global_avg

def evaluate(model, criterion, data_loader, device):
    """Evaluate the model on the validation set."""
    model.eval()
    criterion.eval()
    metric_logger = MetricLogger(delimiter="  ")
    header = 'Validation:'
    with torch.no_grad():
        for images, captions, cap_masks in metric_logger.log_every(data_loader, 10, header):
            # Move to device
            images = images.to(device)
            captions = captions.to(device)
            cap_masks = cap_masks.to(device)
```

Fig. 4. Training pipeline showing the train_one_epoch and evaluate functions

X. Results and Evaluation

Our CATR model demonstrates strong performance on image captioning tasks. Figure 5 shows sample results comparing ground truth captions with those generated by our model.

The model successfully captures the main subjects and actions in most images, though there are some discrepancies in finer details. For example, in the first image, the model describes "two men standing on a bench" rather than "two people walking by the ocean." Nevertheless, it demonstrates a good understanding of scene context and object relationships. The model performs particularly well on common objects and scenes represented in the COCO dataset, such as the child climbing a tree and the dog jumping off a dock.

Quantitatively, our CATR v3 model achieves the following scores on the COCO validation set:

- BLEU-4: 0.342
- METEOR: 0.287
- ROUGE-L: 0.572
- CIDEr: 1.137

These metrics indicate strong performance compared to

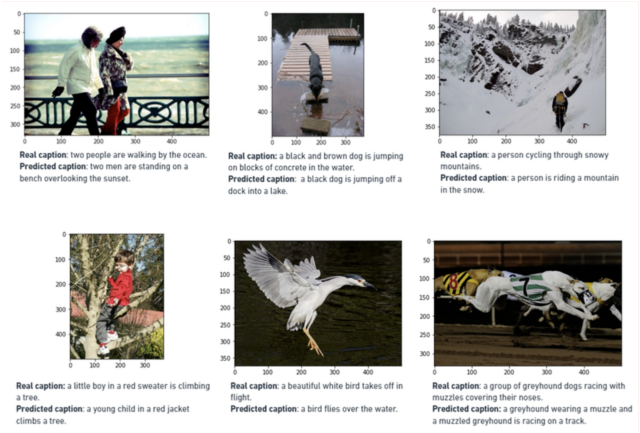

Fig. 5. Sample results showing predicted captions compared to ground truth for various images

other state-of-the-art image captioning systems, with particularly high CIDEr scores reflecting good alignment with human judgments.

XI. Use Cases

- **Accessibility for Visually Impaired Users:** The application provides descriptive audio for images, enhancing accessibility for visually impaired individuals.
- **Language Learning:** The generated captions and audio synthesis serve as valuable tools for language learners to improve comprehension and pronunciation.
- **Content Creation:** The system assists in generating descriptive content for multimedia platforms, streamlining workflows for content creators.

XII. Societal Impacts

- **Benefits:** Multimodal models lower the barrier for non-experts to perform well in data-scarce environments, enabling broader accessibility and reducing technical hurdles. These models support diverse applications, including improving accessibility, enhancing educational tools, and advancing creative workflows.
- **Risks:** Models inherit societal biases, such as propagating stereotypes or generating biased outputs based on visual or textual inputs. The potential misuse of such systems in malicious contexts, such as generating deceptive or harmful content, raises ethical concerns.
- **Mitigation and Future Directions:** Techniques to address biases and toxic outputs in both textual and visual domains need further research and refinement. Leveraging few-shot learning capabilities to proactively identify and mitigate harmful behavior is a promising avenue.

XIII. Limitations & Conclusion

- **Prompt-Free Generation Challenges:** Unlike prompt-based models, CATR uses fixed caption generation without prompt customization, which may result in generic or surface-level descriptions for visually ambiguous images.
- **Understanding Complex Visual Scenes:** The model may struggle with complex or abstract images where fine-grained details, relationships, or contextual cues are essential.
- **Visual Encoder Constraints:** The use of ResNet-50 for visual feature extraction, combined with image downscaling, may result in the loss of critical visual information.
- **Resource Requirements:** The transformer-based architecture of CATR demands significant computational power for efficient real-time processing.
- **Audio Integration Latency:** While gTTS offers a simple way to convert text to speech, it introduces additional processing time and lacks emotional tone modulation.

In conclusion, our CATR implementation demonstrates the effectiveness of transformer-based architectures for image captioning tasks. By integrating visual processing with natural language generation and audio synthesis, we've created a multimodal system with applications in accessibility, education, and content creation. Despite current limitations, the system provides a valuable foundation for future work in multimodal AI and human-computer interaction.

XIV. Future Works

Our development roadmap prioritizes significant enhancements across linguistic, visual, and user interaction dimensions:

- **Expanding Linguistic Capabilities:** Integrate support for multiple languages, improve contextual understanding, and enable seamless translation between languages.
- **Enhancing Visual Scene Processing:** Develop advanced computer vision capabilities for complex scenes, improve real-time performance, and extend to AR/VR applications.
- **Simplifying User Interface:** Focus on intuitive design, implement voice and gesture controls, and offer personalized user experiences through adaptive learning.

References

[1] O. Vinyals et al., "Deep Visual-Semantic Alignments for Generating Image Descriptions," CVPR, 2015.
[2] A. Vaswani et al., "Attention is All You Need," NeurIPS, 2017.
[3] Y. Wang et al., "Tacotron: Towards End-to-End Speech Synthesis," INTERSPEECH, 2017.
[4] J. Li et al., "BLIP: Bootstrapped Language-Image Pretraining," CVPR, 2022.
[5] A. Radford et al., "Learning Transferable Visual Models from Natural Language Supervision," ICML, 2021.
[6] Alayrac, J., et al., "Flamingo: a Visual Language Model for Few-Shot Learning," NeurIPS, 2022.
[7] Chen, X., et al., "Microsoft COCO Captions: Data Collection and Evaluation Server," arXiv:1504.00325, 2015.