PixelParlance: Mitigating Hallucinations in Automatic Image Captioning

Mohammed Shurrab (40323793) **Patrick Liam O'Connor (40310129)**
Shariq Anwar (40321507) **Dan Munteanu (40351135)**

Abstract

Neural image captioning models, particularly those relying on strong language priors, frequently suffer from hallucination. They generate fluent but factually incorrect descriptions of visual content. In this work, we present PIXELPARLANCE, a grounded image captioning system that integrates a Vision Transformer (ViT) encoder with a Transformer decoder. To address the hallucination problem, we introduce a grounding-aware training objective that combines standard cross-entropy loss with a CLIP-based semantic similarity penalty. We evaluate our approach on the MS COCO 2017 dataset using a comprehensive suite of metrics, including BLEU-4 for fluency and CLIPScore and CHAIR for faithfulness. Our experiments demonstrate that incorporating the grounding loss significantly reduces object hallucinations and improves image-text alignment without compromising the readability of the generated captions. The code is publicly available on GitHub.[1]

1 Introduction

Modern digital life produces an overwhelming volume of visual content, where smartphones, social media, and large organizations continuously generate and store billions of images. To make this data accessible and useful, computer vision has gained attraction to convert images into meaningful text Hossain et al. [2019]. Automated captioning plays a crucial role in this process since it supports assistive technologies like screen readers, improves image search and indexing, and enables better content analysis at scale.

However, even advanced captioning machine learning (ML) models, such as convolutional neural networks (CNNs) He et al. [2016], and long short-term memory (LSTM) Hochreiter and Schmidhuber [1997], frequently suffer from hallucination, where the system describes objects, people, or attributes that do not actually exist in the scene. These errors weaken user trust, damage accessibility by providing misleading descriptions, and can negatively affect downstream applications. Current training and evaluation practices tend to reward fluent sentences rather than accurate grounding, meaning that a caption can sound correct but still fail to reflect the actual visual content.

Therefore, this paper proposes PixelParlance, a grounded image captioning system built on a modern Vision Transformer (ViT) Dosovitskiy et al. [2021] encoder – Transformer Vaswani et al. [2017] decoder architecture. The goal is to generate a single, readable caption for each image along with an implicit grounding signal. The main contributions of this paper are as follows:

- Develop an end-to-end image captioning model based on a modern Transformer-Transformer pipeline, where ViT is used as an encoder and a causal Transformer is used as a decoder.
- Utilize the publicly available MS COCO 2017 dataset Lin et al. [2014] for training and evaluating the model.

[1] https://github.com/shariqanwar20/PixelParlance-Comp6321.

- Introduce a grounding-aware objective by incorporating a CLIP-based Radford et al. [2021] similarity loss to discourage hallucination and encourage visual–text alignment
- Provide a rigorous evaluation combining fluency metrics with faithfulness metrics, enabling a balanced assessment of caption quality.
- Analyze hallucination behavior by quantifying object-level consistency, revealing common failure cases such as incorrect attributes and miscounting.

The proposed model is compared with a benchmark based on Xu et al. [2015] that uses CNN as an encoder and an LSTM with attention mechanism as a decoder. We also run our model without the grounding term in the loss function as another baseline. Our experiments show that introducing a lightweight grounding term can improve image–text consistency and reduce hallucinations, while largely preserving standard captioning metrics.

2 Literature Review

Classical encoder–decoder captioning: Early neural image captioning systems relied on a CNN–RNN encoder–decoder design. Vinyals et al. introduced Show and Tell, where a convolutional neural network encodes an image into a global feature vector that conditions an LSTM decoder to predict captions sequentially Vinyals et al. [2015]. This demonstrated that end-to-end training on MS COCO can produce fluent, relevant descriptions. Xu et al. extended this approach with soft visual attention in Show, Attend and Tell, enabling the decoder to dynamically focus on salient spatial regions and improving interpretability and caption quality Xu et al. [2015].

Transformer-based captioning and Vision Transformers: The introduction of the Transformer architecture prompted a shift away from recurrence for language generation. In captioning, Transformer decoders have shown stronger sequence modeling capabilities than LSTMs. Herdade et al. proposed the Object Relation Transformer, incorporating geometric attention to model object interactions explicitly, which improved standard captioning metrics on COCO Herdade et al. [2019]. In parallel, Dosovitskiy et al. introduced the Vision Transformer (ViT) as an attention-only alternative to CNNs for image encoding Dosovitskiy et al. [2021]. ViT pretrained on large-scale datasets has since become a powerful foundation for multimodal learning, and recent captioning systems pair ViT encoders with Transformer or GPT-based decoders to leverage large-scale vision–language pretraining Cornia et al. [2020], Li et al..

Hallucination in image captioning: Although captioning performance continues to improve, studies have shown that neural models can generate hallucinated content, confidently describing non-existent objects due to strong language priors. Rohrbach et al. formalized this failure mode through CHAIR (Caption Hallucination Assessment with Image Relevance), a metric assessing hallucination at caption and instance levels Rohrbach et al. [2018]. Their findings indicate that high BLEU (Bilingual Evaluation Understudy) or CIDEr (Consensus-based Image Description Evaluation) scores do not necessarily reflect faithful grounding.

CLIPScore and grounding-aware objectives: To address faithfulness in image captioning, recent work incorporates grounding-aware training and reference-free semantic evaluation. CLIP (Contrastive Language–Image Pretraining), trained via contrastive alignment of image–text pairs, produces a shared embedding space where aligned pairs have high cosine similarity Radford et al. [2021]. Hessel et al. introduced CLIPScore, which correlates well with human judgments of semantic alignment and can outperform traditional reference-based evaluations Hessel et al. [2021]. Emerging captioning approaches now integrate CLIP-based regularization to reduce hallucination by rewarding captions that better match image representations Deng et al. [2022].

3 Methodology

3.1 Data and preprocessing

We use the publicly available MS COCO 2017 Captions corpus Lin et al. [2014] as our training and evaluation resource. It contains 118,287 training images and 5,000 testing images, each paired with five human-written captions. To evaluate grounding and hallucination behavior, we additionally use the COCO instance annotations, which provide 80 object categories with bounding boxes and allow us to infer a ground-truth set of objects per image. Dataset acquisition is automated in our pipeline: the official image archives and annotation files are downloaded, extracted, and placed into a consistent directory structure. Integrity checks are then performed by validating the image count, opening random samples with PIL, and inspecting annotation JSON files for expected fields and formatting correctness. Image preprocessing follows standard Vision Transformer input conventions, including resizing to 224×224, tensor conversion, and ImageNet normalization. Caption preprocessing is intentionally minimal because the tokenizer manages casing and punctuation; we ensure proper UTF-8 encoding, remove unprintable characters, and discard only empty captions (rare in COCO). Text is tokenized into subword units using a learned byte-pair encoding (BPE) vocabulary and padded or truncated to a maximum length of 30 tokens with attention masks. These preprocessing steps are fully reproducible in our Jupyter Notebook workflow.

3.2 Model architecture

The proposed approach employs a Transformer-based encoder–decoder architecture for grounded caption generation, as shown in Fig. 1. The primary objective is to produce fluent captions while reducing hallucination, ensuring that generated text closely reflects visual evidence. The system receives an input RGB image $I \in \mathbb{R}^{3\times 224\times 224}$, which is first divided into 16×16 patches resulting in a sequence of $S = 196$ tokens. These patches are embedded and processed by a Vision Transformer (ViT-B/16) Dosovitskiy et al. [2021], pretrained on large-scale image datasets (ImageNet-1k Russakovsky et al. [2015]). The encoder outputs a sequence of visual feature embeddings $V \in \mathbb{R}^{S\times 768}$, which we project into a shared multimodal space $V' \in \mathbb{R}^{S\times 512}$ to align with the decoder dimensionality.

Caption generation is performed autoregressively using a six-layer causal Transformer decoder that models the probability of each token conditioned on previous tokens and the encoded image. Each layer of the decoder includes a) Masked Multi-Head Self-Attention, b) Multi-Head Cross-Attention, c) Feed-Forward Network, and d) LayerNorm + residuals throughout. During decoding, masked self-attention ensures that the transformer can only access past tokens, while cross-attention allows the caption generator to attend to relevant visual patches from V' At training time, we use teacher forcing Williams and Zipser [1989] by supplying ground-truth tokens shifted right by one position, whereas at inference time captions are generated without ground-truth guidance. Token generation is driven by Vaswani et al. [2017]:

$$P\left(y_t \mid y_{<t},\, I\right) = \mathrm{Softmax}\left(W h_t\right) \tag{1}$$

where y_t is the token predicted at time step t, $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes all previously generated tokens, I is the input image, $h_t \in \mathbb{R}^d$ is the decoder hidden state at time step t,and $W \in \mathbb{R}^{V\times d}$ is the output projection matrix mapping the hidden representation into the vocabulary space of size V. Captions are tokenized into a fixed maximum length of 30 tokens, padded where necessary, and encoded using a custom subword vocabulary derived from MS COCO 2017.

3.3 Training Procedure

The proposed model is trained end-to-end using a two-stage optimization schedule designed to progressively balance linguistic fluency and visual grounding. In the initial warm-up stage, we optimize only the standard cross-entropy objective ($\mathcal{L}_{CE}$) to establish a strong language model. Given an input image $\boldsymbol{I}$ and its caption $\boldsymbol{y} = (y_1, \ldots, y_T)$, the decoder predicts each token autoregressively Williams and Zipser [1989]:

$$\mathcal{L}_{CE} = -\sum_{t=1}^{T} \log P\left(y_t \mid y_{<t},\, \boldsymbol{I}\right), \tag{2}$$

During this phase, the Vision Transformer encoder remains frozen to maintain stable pretrained visual features and prevent early reliance on language priors, which can lead to repetitive or generic captions. Once the decoder demonstrates fluent caption generation, we introduce a grounding-aware regularization term using a frozen CLIP model. After each training step, a provisional caption $\hat{\boldsymbol{y}}$ is decoded, and cosine similarity is computed between CLIP image and text embeddings, forming the grounding loss ($\mathcal{L}_{ground}$) Hessel et al. [2021]:

$$\mathcal{L}_{\text{ground}} = 1 - \cos\Big(f_{\text{img}}(\boldsymbol{I}),\, f_{\text{txt}}(\hat{\boldsymbol{y}})\Big), \tag{3}$$

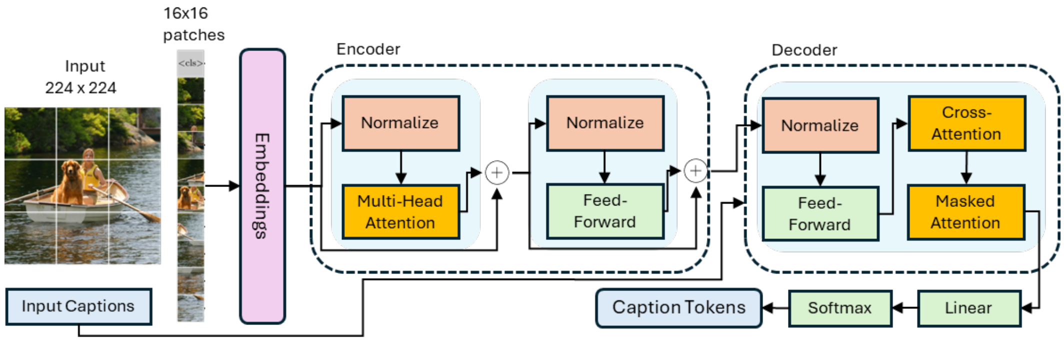


Figure 1: Proposed model architecture composed of ViT encoder and lightweight Transformer decoder.

where f_{img} and f_{txt} denote the frozen CLIP encoders that map each modality into a shared embedding space. The final training objective combines both terms:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \, \mathcal{L}_{\text{ground}}, \tag{4}$$

where λ is gradually increased to 0.5 throughout training. During this stage, only the last two ViT blocks are fine-tuned with a much smaller learning rate to maintain stable optimization. Beam search Vinyals et al. [2015] (size 3–5) is used for inference to improve caption quality and variation..

All training and experiments are conducted within a Kaggle Notebook environment using Numpy, PyTorch Paszke et al. [2019], HuggingFace Tokenizers, and `timm` libraries, running on NVIDIA Tesla T4 and P100 GPUs with mixed-precision acceleration. Dataset loading and image transforms rely on torchvision and PIL, while evaluation uses BLEU from NLTK and CLIP-based similarity metrics. Optimization employs AdamW with label smoothing (0.1), cosine learning-rate decay, batch size 32, and 10–15 epochs. We ensure reproducibility through fixed seeds, checkpoints, and a scripted end-to-end workflow. Some key code snippets are provided in Appendix A.

4 Experimental Results

In this section, we evaluate the proposed grounded Transformer captioning model against a strong attention-based baseline, Show, Attend and Tell (SAT) Xu et al. [2015], using the MS COCO 2017 validation split. Following standard captioning evaluation practice, we report corpus-level scores for BLEU-4, CLIPScore, and CHAIR.

4.1 Evaluation Metrics

BLEU-4 Papineni et al. [2002] measures n-gram precision between a generated caption $\hat{y}$ and a set of human-written ground-truth references Y. Higher BLEU-4 indicates stronger language fluency and content overlap. To assess grounding, we compute CLIPScore Hessel et al. [2021], which uses a pretrained CLIP encoder to compare cosine similarity between image and caption embeddings Additionally, hallucination is measured using the CHAIR (Caption Hallucination Assessment with Image Relevance) metrics Rohrbach et al. [2018] where lower values indicate stronger grounding and fewer hallucinated object mentions. The detailed equations are provided in Appendix B.

4.2 Baseline

The proposed model is compared to a benchmark based on Xu et al. [2015], where they utilize a CNN composed of ResNet pretrained on ImageNet as an encoder to extract spatial visual features, which are then fed to an attention-LSTM decoder that generates captions one word at a time. This baseline reflects classical captioning pipelines that emphasize fluency while offering limited robustness against semantic hallucination

4.3 Results

The results comparison between the proposed model and the benchmark is summarized in Table 1 for the aforementioned metrics. The classical baseline achieves reasonable results but exhibits notable hallucination rates. Our Transformer–Transformer baseline without grounding loss improves BLEU-4 slightly and yields stronger CLIPScore, indicating improved alignment between visual and textual representations. Introducing the CLIP-guided grounding loss further enhances semantic consistency and reduces hallucination, demonstrating that even a lightweight regularizer can nudge the model toward more truthful caption generation.

Beam search decoding (beam size = 3) boosts performance across all metrics, producing more coherent and visually grounded captions than greedy decoding. Under this configuration, our system achieves the highest BLEU-4 and CLIPScore values, while also obtaining the lowest CHAIR_s score, showing a relative reduction in hallucination frequency compared to both the baseline and our non-grounded model.

Qualitative examples further illustrate the effect of grounding. For several images, we compare baseline and grounded captions, given the ground truth: in a motorcycle scenario, the Baseline produces: “a motorcycle parked on a street with cars”, whereas the grounded give: "a red motorcycle parked in a small garage”, and the ground truth is “a motorcycle with red seat sits parked in a garage” The grounded caption more accurately reflects the indoor garage and color details, whereas the baseline introduces generic “street with cars” language. Office scene: Baseline: “a kitchen with a stove and a sink”, our proposed model: “a computer desk with a monitor and a chair”, and the Ground Truth: “an office cubicle with four different types of computers”. Here, The baseline misclassifies the entire scene category as a kitchen. Our grounded model correctly recovers the office semantics, greatly reducing hallucination. More examples are given in Appendix C.

Overall, our experiments demonstrate that a modern ViT encoder coupled with a Transformer decoder forms a competitive baseline for image captioning and that incorporating a simple grounding constraint can meaningfully improve caption accuracy and trustworthiness without requiring additional detection supervision.

Table 1: Comparison of captioning models on the MS COCO 2017 dataset. Higher BLEU-4 and CLIPScore indicate better fluency; lower CHAIR indicate fewer hallucinations.

Model	Decoder	BLEU-4 $\uparrow$	CLIPScore $\uparrow$	$\text{CHAIR}_s \downarrow$	$\text{CHAIR}_i \downarrow$
SAT baseline	Greedy	0.053	0.22	0.22	0.28
SAT baseline	Beam = 3	0.06	0.23	0.22	0.28
Ours (no grounding)	Greedy	0.063	0.23	0.15	0.15
Ours (with grounding)	Greedy	0.065	0.23	0.13	0.15
Ours (with grounding)	Beam = 3	**0.071**	**0.25**	**0.12**	**0.15**

5 Conclusion

In this work, we developed and evaluated PIXELPARLANCE, an end-to-end image captioning architecture designed to mitigate the prevalence of object hallucinations. By replacing traditional CNN encoders with a pre-trained Vision Transformer (ViT) and augmenting the training objective with a CLIP-guided consistency loss, we successfully improved the semantic alignment between input images and generated text. Our results indicate that while standard likelihood-based training prioritizes fluency, often at the cost of accuracy, our grounded approach strikes a better balance. The reduction in CHAIR metric scores confirms that our model is less prone to fabricating non-existent objects, marking a step forward in building more reliable vision technologies.

Limitations and Future Work. Our training was limited to a subset of the dataset due to computational constraints, reducing achievable BLEU-4 performance versus full-corpus models. Grounding depends on CLIP, meaning any biases or failures in its embedding space propagate into our captions. The loss currently uses a single global sentence-level grounding term, missing fine-grained alignment. Future work includes token-level contrastive grounding and constrained beam search to better enforce visual fidelity.

References

Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10578–10587, 2020.

Yihong Deng, Yan Li, Juncheng Zhang, and Quanquan Gu. Length-controllable image captioning with clip reward. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2491–2503, 2022.

Alexey Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. Image captioning: Transforming objects into words via attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 11135–11145, 2019.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7514–7528, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural Computation*, volume 9, pages 1735–1780. MIT Press, 1997.

Md Zakir Hossain, Ferdous Sohel, Mohd Fauzi Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys*, 51(6):118, 2019.

Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Xuedong Dong, and Furu Wei. Oscar: Object-semantics aligned pre-training for vision-language tasks.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

Kishore Papineni et al. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002.

Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.

Alec Radford et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763, 2021.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Marcus Rohrbach. Object hallucination in image captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4035–4045, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. volume 115, pages 211–252, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2015.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. In *Neural Computation*, volume 1, pages 270–280, 1989.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, pages 2048–2057, 2015.

A Code snippets

– Dataset Download

```
import os

#CHANGE THIS to where you want the dataset
COCO_ROOT = "/kaggle/input/coco-2017-dataset"

IMAGES_DIR = COCO_ROOT  # train2017/ and val2017/ will live directly under this
ANN_DIR = os.path.join(COCO_ROOT, "annotations")

os.makedirs(COCO_ROOT, exist_ok=True)
os.makedirs(ANN_DIR, exist_ok=True)

COCO_ROOT, IMAGES_DIR, ANN_DIR

print("Downloading MS COCO 2017 train/val + annotations to", COCO_ROOT)

# Train images
!cd "$COCO_ROOT" && wget -c http://images.cocodataset.org/zips/train2017.zip

# Val images
!cd "$COCO_ROOT" && wget -c http://images.cocodataset.org/zips/val2017.zip

# Train/Val annotations (includes captions)
!cd "$COCO_ROOT" && wget -c http://images.cocodataset.org/
                                    annotations/annotations_trainval2017.zip

# Unzip train and val images into COCO_ROOT
!cd "$COCO_ROOT" && unzip -q train2017.zip
!cd "$COCO_ROOT" && unzip -q val2017.zip

# Unzip annotations into COCO_ROOT/annotations
!cd "$COCO_ROOT" && unzip -q annotations_trainval2017.zip -d "$ANN_DIR"
```

– Dataloaders

```
class COCODataset(Dataset):
    def __init__(
        self,
        images_root: str,
        captions_json: str,
        vocab: Vocabulary,
        max_len: int = 30,
        transform=None,
        debug_limit: int = None,
    ):
        self.images_root = images_root
        self.vocab = vocab
        self.max_len = max_len
        self.transform = transform or self._default_transform()
```

```python
        with open(captions_json, "r") as f:
            ann = json.load(f)

        self.imgs = {img["id"]: img for img in ann["images"]}

        self.samples = []
        for a in ann["annotations"]:
            img_id = a["image_id"]
            caption = a["caption"]
            tokens = tokenize_caption(caption)
            self.samples.append((self.imgs[img_id]["file_name"], tokens))

        if debug_limit is not None:
            self.samples = self.samples[:debug_limit]
            print(f"[COCODataset] Debug limit: {len(self.samples)} samples")

        print(f"Loaded {len(self.samples)} (image, caption) pairs from {captions_jsor

    def _default_transform(self):
        return transforms.Compose([
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize(
                mean=[0.485, 0.456, 0.406],
                std=[0.229, 0.224, 0.225],
            ),
        ])

    def __len__(self) -> int:
        return len(self.samples)

    def __getitem__(self, idx: int) -> Dict[str, Any]:
        file_name, tokens = self.samples[idx]
        img_path = os.path.join(self.images_root, file_name)

        image = Image.open(img_path).convert("RGB")
        image = self.transform(image)

        bos_id = self.vocab.stoi["<bos>"]
        eos_id = self.vocab.stoi["<eos>"]
        pad_id = self.vocab.stoi["<pad>"]

        seq_ids = [bos_id] + self.vocab.numericalize(tokens) + [eos_id]
        if len(seq_ids) < self.max_len:
            seq_ids += [pad_id] * (self.max_len - len(seq_ids))
        else:
            seq_ids = seq_ids[:self.max_len]

        caption = torch.tensor(seq_ids, dtype=torch.long)

        return {
            "image": image,
            "caption": caption,
            "file_name": file_name,
        }
```

– Model Architecture

```python
class ViTEncoder(nn.Module):
```

```python
    def __init__(
        self,
        model_name: str = "vit_base_patch16_224",
        pretrained: bool = True,
        trainable: bool = False,
        d_model: int = 512,
    ):
        super().__init__()
        self.vit = timm.create_model(
            model_name,
            pretrained=pretrained,
        )
        self.vit.reset_classifier(0)
        vit_dim = self.vit.num_features
        if vit_dim != d_model:
            self.proj = nn.Linear(vit_dim, d_model)
        else:
            self.proj = nn.Identity()

        for p in self.vit.parameters():
            p.requires_grad = trainable

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        feats = self.vit.forward_features(x)  # [B, S, C]
        feats = self.proj(feats)              # [B, S, d_model]
        return feats

class TransformerCaptionDecoder(nn.Module):
    def __init__(
        self,
        vocab_size: int,
        d_model: int = 512,
        num_layers: int = 6,
        num_heads: int = 8,
        dim_feedforward: int = 2048,
        max_len: int = 30,
        dropout: float = 0.1,
    ):
        super().__init__()
        self.vocab_size = vocab_size
        self.d_model = d_model
        self.max_len = max_len

        self.token_embed = nn.Embedding(vocab_size, d_model)
        self.pos_embed = nn.Embedding(max_len, d_model)
        self.dropout = nn.Dropout(dropout)

        decoder_layer = nn.TransformerDecoderLayer(
            d_model=d_model,
            nhead=num_heads,
            dim_feedforward=dim_feedforward,
            dropout=dropout,
            batch_first=True,
        )
        self.decoder = nn.TransformerDecoder(decoder_layer, num_layers=num_layers)
        self.out_proj = nn.Linear(d_model, vocab_size)

    def forward(self, tgt, memory, tgt_key_padding_mask=None):
        B, T = tgt.shape
```

```
        positions = torch.arange(0, T, device=tgt.device).unsqueeze(0).expand(B, T)
        x = self.token_embed(tgt) * (self.d_model ** 0.5)
        x = x + self.pos_embed(positions)
        x = self.dropout(x)

        causal_mask = torch.triu(
            torch.ones(T, T, device=tgt.device, dtype=torch.bool),
            diagonal=1,
        )

        x = self.decoder(
            tgt=x,
            memory=memory,
            tgt_mask=causal_mask,
            tgt_key_padding_mask=tgt_key_padding_mask,
        )
        logits = self.out_proj(x)
        return logits

class Captioner(nn.Module):
    def __init__(self, vocab_size: int, max_len: int = 30,
                 d_model: int = 512, vit_trainable: bool = False):
        super().__init__()
        self.encoder = ViTEncoder(d_model=d_model, trainable=vit_trainable)
        self.decoder = TransformerCaptionDecoder(
            vocab_size=vocab_size,
            d_model=d_model,
            max_len=max_len,
        )
        self.max_len = max_len

    def forward(self, images: torch.Tensor, captions_in: torch.Tensor) -> torch.Tens
        memory = self.encoder(images)
        logits = self.decoder(captions_in, memory)
        return logits
```

B Evaluation Metrics

BLEU-4 Papineni et al. [2002] measures n-gram precision between a generated caption $\hat{y}$ and a set of human-written ground-truth references Y. Higher BLEU-4 indicates stronger language fluency and content overlap. It is defined as:

$$\text{BLEU-4} = \exp\left(\min\left(1 - \frac{r}{c}, 0\right) + \frac{1}{4}\sum_{n=1}^{4}\log p_n\right) \tag{5}$$

where p_n denotes the modified n-gram precision, c is the candidate caption length, and r is the reference length heuristic. To assess grounding, we compute CLIPScore Hessel et al. [2021], which uses a pretrained CLIP encoder to compare cosine similarity between image and caption embeddings:

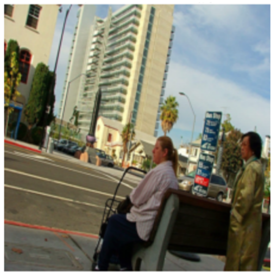
$$\text{CLIPScore}(\hat{y}, I) = \cos\left(f_{\text{img}}(I), f_{\text{txt}}(\hat{y})\right) \tag{6}$$

Additionally, hallucination is measured using the CHAIR (Caption Hallucination Assessment with Image Relevance) metrics Rohrbach et al. [2018]:

$$\text{CHAIR}_s = \frac{\#\text{ hallucinating captions}}{\#\text{ total captions}}, \qquad \text{CHAIR}_i = \frac{\#\text{ hallucinated object tokens}}{\#\text{ total object mentions}} \tag{7}$$

where lower values indicate stronger grounding and fewer hallucinated object mentions.

C Qualitative Results



Baseline (beam=3): "A man sitting on a street with a city street"
Ours (beam=3): "A group of people waiting in a city street"
Ground Truth: "A woman sitting on a bench and a woman standing waiting for the bus"

(a) Pedestrian scenario

Baseline (beam=3): "a kitchen with astove and a sink"
Ours (beam=3): "a computer desk with a monitor and a chair"
Ground Truth: "an office cubicle with four different types of computers"

(b) Office scenario

Figure 2: Examples of generated captions compared to the ground truth.

Checklist (modelled after NeurIPS Paper Checklist)

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

Justification: The abstract and introduction clearly state that we propose a ViT-based image captioning model with CLIP-guided training to reduce hallucinations, evaluated on COCO 2017. This is also supported by the results.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We explicitly discuss limitations, including dependence on COCO-style photographs, the computational cost of training transformer-based captioners, and remaining hallucinations and failure cases, in the Future works

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper is empirical and does not introduce new theoretical results or formal proofs; it focuses on model design and experimental evaluation for image captioning.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We share the GitHub link to our code. We also describe the model architecture, training details, data preprocessing, evaluation metrics (BLEU, CLIPScore, CHAIR), and dataset splits, and we provide enough hyperparameter, optimization, and implementation details for another researcher to reproduce the main results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We build on the publicly available COCO 2017 dataset and we have instructions on how to run the code on our GitHub.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The experimental section specifies the COCO 2017 train/validation splits, preprocessing steps, model sizes, optimization settings (optimizer, learning rate, batch size, number of epochs), and selection of hyperparameters, which also included in the GitHub link.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We report single-seed performance for each method due to computational constraints and do not include error bars or statistical significance tests.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the type of hardware used (e.g., single GPU, memory), the python libraries used, approximate training time per run, and the number of runs performed.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Broader impacts**

   Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

   Answer: [NA]

   Justification: We do not explicitly mention the impacts however, one of the main objectives is to improve accessibility to the users.

   Guidelines:

   - The answer NA means that there is no societal impact of the work performed.
   - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
   - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
   - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
   - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

10. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The models and datasets used in this work are standard image captioning components trained on COCO and are not large, general-purpose foundation models or scraped web datasets with high dual-use risk; therefore, no special safeguards beyond standard licensing and usage conditions are required.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

11. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not introduce new datasets or large pretrained models; our experiments rely on the existing COCO 2017 dataset and standard backbones.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

12. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method and experiments do not rely on large language models as an important or novel component; any LLM tools were used only for auxiliary tasks such as drafting or editing text, debugging codes and help criticizing ideas

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.