TinyVLM: Scaling Down Vision-Language Models for the Edge

Anonymous ACL submission

Abstract

In this paper, we introduce TinyVLM, a compact and efficient Vision Language Model (VLM) designed for edge devices, which can be trained end-to-end in 106 A100 GPU hours or $159[1]. We introduce multiple adaptations to the classic ViT-LLM style VLMs, by introducing a convolution token pooler to reduce the number of visual tokens passed into the LLM by 4×, a cross-attention mechanism to fuse spatial features from a masked auto-encoder CNN model improving spatial understanding in tasks such as OCR, a patch zooming technique to capture fine-grained image details and a carefully curated fine-tuning dataset. Our final model has 0.6 B parameters and achieves a throughput of 18 toks/sec on a 8-core CPU machine, making it highly suitable for resource-constrained environments. TinyVLM achieves a good balance between performance and resource demands, advancing the capabilities of VLMs on the edge. We open source our complete training data, fully reproducible code and model weights for the community.

1 Introduction

Vision-Language Models (VLMs) have significantly advanced multimodal AI by enabling the joint understanding of visual and textual information, unlocking applications ranging from image captioning (Vinyals et al., 2015) to medical diagnostics (Rajpurkar et al., 2022; Yildirim et al., 2024). Pioneering architectures like CLIP (Radford et al., 2021) have demonstrated the effectiveness of aligning vision and language representations at scale. However, large-scale VLMs—such as LLaVA (Liu et al., 2023), BLIP (Li et al., 2023a) and Flamingo (Alayrac et al., 2022)—have also introduced significant computational bottlenecks, limiting their practicality for real-world deployment.

Despite their impressive performance, state-of-the-art VLMs face a few fundamental challenges. Training billion-parameter models demands massive computational resources, often consuming GPU hours equivalent to dozens of transatlantic flights in CO_2 emissions (Strubell et al., 2020). Even after training, inference remains expensive: running a 7B-parameter VLM requires 16GB+ VRAM, far exceeding the constraints of consumer GPUs and edge devices. Even after applying quantization techniques (Dettmers et al., 2022; Lin et al., 2024a), their latency and power consumption often exceed the constraints of real-world applications, such as assistive robotics and augmented reality.

Vision Transformers (ViTs) (Dosovitskiy et al., 2020), which serve as the primary vision encoders in most VLMs, excel at capturing global context but struggle with fine-grained spatial details. This limitation is particularly evident in tasks requiring pixel-level precision. Additionally, high-resolution image processing exacerbates computational inefficiency: a 448×448px image segmented into 14×14 patches produces 1,024 tokens, leading to quadratic complexity in self-attention (Touvron et al., 2021). Lastly, compact VLMs trained on web-crawled multimodal datasets (e.g., LAION (Schuhmann et al., 2022), CC3M (Changpinyo et al., 2021), WebLI (Chen et al., 2022)) suffer from high validation loss due to noisy labels and sparsity, reducing generalization ability.

This paper addresses these challenges by proposing a tiny yet powerful VLM, designed to strike a balance between performance and efficiency. Our approach introduces several contributions including:

Visual Feature Enrichment. To enhance the feature representation of ViT-based CLIP embeddings, we integrate a pre-trained Convolutional Neural Network (CNN) into the final stage of training. Using a cross-attention mechanism, we fuse CNN-extracted local details with ViT-based global em-

[1]Assuming A100 hour = 1.5$

beddings, enhancing fine-grained feature representation.

CNN-Guided Token Pooling. Extending the tile-based preprocessing introduced in Dragonfly (Thapa et al., 2024), we develop a CNN-guided token pooling mechanism to address the challenge of excessive visual tokens generated by the multiple high-resolution image patches. This pooling mechanism exploits local spatial redundancy in ViT embeddings, reducing computational overhead by 4× while preserving critical features.

Curated Multi-Modal Dataset for Tiny VLMs. To improve data efficiency, we construct a specialized multimodal dataset optimized for small-scale VLM training. This dataset mitigates overfitting and improves generalization, ensuring robust performance even with reduced model capacity.

We successfully train a 600M-parameter VLM that achieves comparable performance to much larger models while maintaining significantly lower computational costs. Our full training pipeline requires 106.3 A100 GPU hours, making it one of the most resource-efficient methods for training compact VLMs.

2 Related Work

Visual Language Model Architectures. VLMs follow an encoder-decoder framework, integrating a vision backbone (e.g., ViT or CNN) with a Large Language Model (LLM). These components are connected via a connector that aligns visual and textual modalities.

Early architectures such as LLaVA (Liu et al., 2023) and its successors (LLaVA-1.5 (Liu et al., 2024a), LLaVA-NeXT (Liu et al., 2024b)) employ lightweight MLP-based connectors to project flattened ViT embeddings into the LLM's embedding space. However, this approach results in a large number of visual tokens being stored in the LLM's KV Cache, significantly increasing memory consumption. This effect becomes even more pronounced with higher image resolutions, where the number of tokens scales quadratically with image size. Dragonfly (Thapa et al., 2024) introduces a tile-based pre-processing approach, where images are split into smaller patches. However, this does not reduce the total token count; rather, Dragonfly's novelty lies in its three-tiered resolution strategy combined with a similarity-based patch selection mechanism, enabling fine-grained high resolution feature learning.

In contrast, transformer-based connectors—used in models such as BLIP-2 (Li et al., 2023a) and Flamingo (Alayrac et al., 2022)—apply cross-attention or perceiver-style mechanisms to compress visual tokens before passing them to the LLM. For instance, Flamingo employs gated cross-attention layers, dynamically fusing image and text features to significantly reduce token count while maintaining strong performance and BLIP-2 utilizes a Q-Former which outputs a fixed and significantly smaller number of learned query vectors as tokens.

Beyond connectors, post-processing strategies also impact efficiency. LLaVA directly feeds projected tokens into the LLM's decoder, whereas Flamingo interleaves cross-attention within the LLM itself, allowing tighter modality fusion and improved visual-textual interactions. More recently, compact VLMs such as MobileVLM (Chu et al., 2023, 2024), DeepSeek-VL (Lu et al., 2024), and NanoLLaVA[2] have demonstrated the feasibility of small-scale architectures, offering a trade-off between performance and computational efficiency.

Data and Training Strategies. Training pipelines for VLMs typically follow a two-stage paradigm. The first stage focuses on aligning visual and textual embeddings, typically using datasets such as COCO Captions (Chen et al., 2015), CC3M (Changpinyo et al., 2021), LAION (Schuhmann et al., 2022), LLaVA-Pretrain (Liu et al., 2023) and OBELICS (Laurençon et al., 2023). During this phase, the connector is trained while freezing both the vision encoder and the LLM to establish a shared representation space between images and text. The second stage involves instruction fine-tuning using datasets such as LLaVA-Instruct (Liu et al., 2023), often augmented with synthetic data generated by GPT-4 (Achiam et al., 2023). This phase allows the model to adapt to real-world multimodal interactions, improving its ability to handle diverse instructions.

Certain models introduce enhanced training methodologies. For instance, IDEFICS-3 (Laurençon et al., 2024) employs a curriculum learning strategy, progressively training the model on captioning, question-answering (QA), and complex reasoning tasks. This structured learning approach enables VLMs to balance broad general-

[2] https://huggingface.co/qnguyen3/nanoLLaVA

ization with task-specific adaptation, improving efficiency while maintaining robust performance across a wide range of multimodal tasks.

3 Building VLM

3.1 Model Architecture

Vision Encoder: We adopt **SigLIP-base-patch16-384** (Zhai et al., 2023) as the primary visual backbone. Compared to the standard 224-resolution encoders, the 384-resolution variant improves fine-grained text recognition (e.g., +12.8% on TextVQA, Table 1) while maintaining manageable computational costs. We did not see massive improvements in performance upon using 512-resolution variant while the number visual tokens increased almost 2× as shown in Table 1. We explain later in this section on the importance of having lower number of visual tokens for better efficiency.

To incorporate localized feature extraction, we fuse features from a masked auto-encoder model - ConvNeXt-Tiny, leveraging the inductive biases of self-supervised pre-trained CNNs. These features are fused using a cross-attention mechanism with the CLIP-based fine-tuned vision tokens in the final phase of training. Our hybrid design introduces only 49 additional visual tokens but outperforms pure transformer-based baselines, particularly in dense-text scenarios such as TextVQA.

Language Model: **Qwen2.5-0.5B-Instruct** (Yang et al., 2024) serves as the text decoder, balancing inference speed and reasoning capabilities. It's instruction-tuning aligns with multimodal prompts, enabling zero-shot task generalization.

Connector: A lightweight **2-layer MLP with GeLU** aligns the clip-based visual embeddings with the LLM embeddings. Compared to Q-Former (Li et al., 2023a) or Perceiver resamplers (Jaegle et al., 2021; Alayrac et al., 2022; Bai et al., 2023; Laurençon et al., 2024), this design reduces parameters substantially. Recent works have shown that MLP-based connectors are more parameter-efficient while being equally representative of the visual to language transformation of the tokens. Using a simple linear projector forces the LLM to learn more and leads to better generalization (Lin et al., 2024b).

The final fine-tuning stage of our pipeline involves leveraging a pre-trained CNN to inject spatial features into the learning process and refine spatial understanding in tasks such as text understanding. To do this, we generate visual tokens from a CNN architecture and perform attention-guided token refinement using the ViT based fine-tuned tokens. These final tokens are passed directly into the LLM. This approach allows new spatial-aware feature infusion into the VLM while ensuring no massive performance drop due to the introduction of a new module in between the fine-tuning process. We define $E_g \in R^{N\times D}, E_{\text{cnn}} \in R^{M\times D}$ as the global visual tokens from fine-tuned ViT model and spatial tokens from the CNN model respectively. Our selected ViT and CNN models share the same embedding dimension D, ensuring compatibility for feature fusion. We project E_g and E_{cnn} using linear layers and then perform cross-attention using CNN-based queries and ViT-based key-value vectors.

$$Q_{\text{cnn}} = E_{\text{cnn}} \times W_q, \quad KV_g = E_g \times W_{kv}, \quad (1)$$

where $W_q \in R^{D\times d}, W_{kv} \in R^{D\times d}$ are learned projection matrices for the queries and key-value pairs respectively; and d denotes the embedding dimension of the chosen LLM. To enhance spatial correspondence, we incorporate positional encodings (PE) into both queries and key-value representations before computing the attention matrix:

$$\begin{aligned} A = \text{Softmax}\Big(&(Q_{\text{cnn}} + \text{PE}(Q_{\text{cnn}})) \\ &\times (KV_g + \text{PE}(KV_g))^\top \Big), \hat{A} \in R^{M\times N} \end{aligned} \quad (2)$$

Finally, we project the output using KV_g sharing the same set of vectors for key as well as value in the cross attention mechanism.

$$\hat{E}_{cnn} = A \times KV_g, \hat{E}_{cnn} \in R^{M\times d} \quad (3)$$

This fusion strategy enables the pre-trained CNN features to be enriched by fine-tuned ViT-based embeddings, capturing both localized spatial details and high-level semantic representations. ConvNeXt-Tiny (Woo et al., 2023) outputs a total of $M = 49$ tokens which is fairly small as compared to $N = 576$ tokens from the ViT, limiting the computational overhead introduced by $\hat{E}_{cnn}$.

Patch Zooming Strategy: Inspired by recent works on image processing for VLMs (Liu et al., 2024a; Thapa et al., 2024), we resize images to two resolutions: 384×384 and 768×768 denoted by I_g and I_l respectively. I_l is further broken down into four non-overlapping patches $I_l = \{I_{l1}, I_{l2}, I_{l3}, I_{l4}\}$,where I_{l1} , I_{l2} , I_{l3} , and I_{l4}

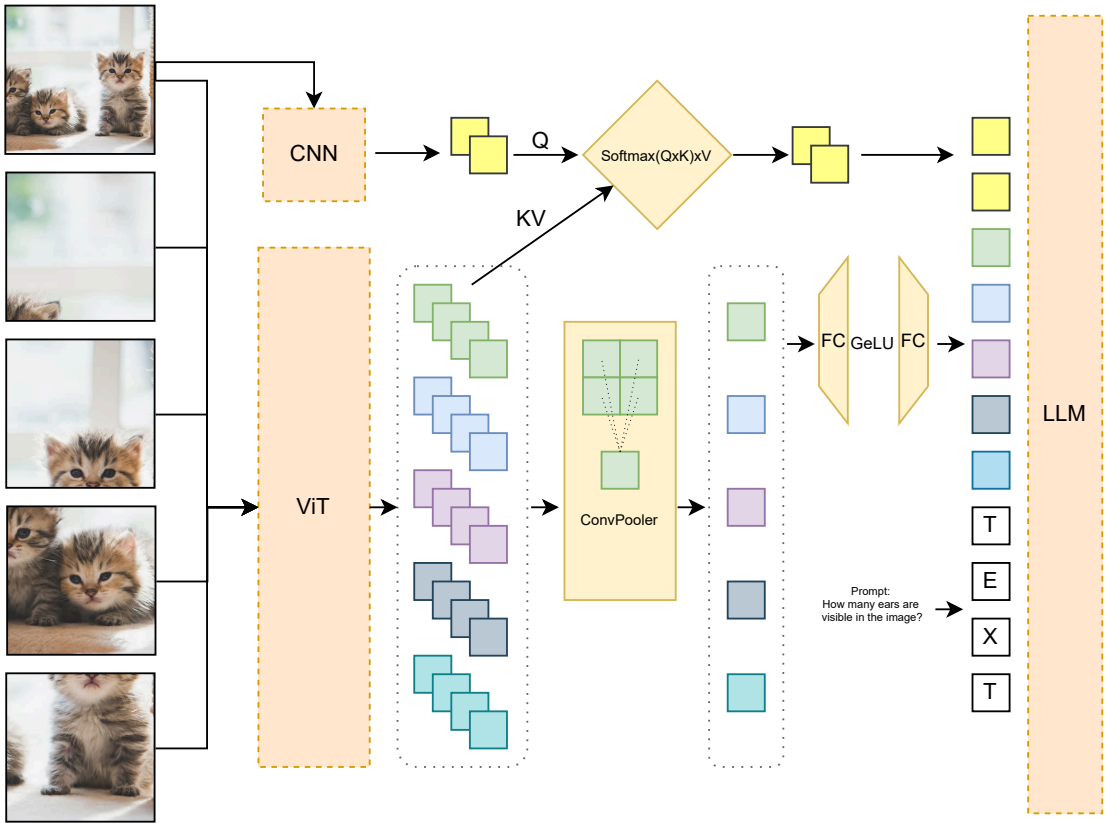

Figure 1: Schematic representation demonstrating the working of different components of TinyVLM. The original image is zoomed and split into 4 sub-images which are then fed into the ViT. Post which they are processed by the ConvPooler and connector in order to get the final ViT-based input tokens to the LLM. The global image is also passed into the CNN in parallel and the spatial CNN tokens undergo cross attention with the global tokens generated by ViT to get the final CNN-based input tokens to the LLM. We finally concatenate the CNN tokens, ViT tokens and prompt tokens as the final input to the LLM.

correspond to the top-left, top-right, bottom-left, and bottom-right regions of I_l , each with a resolution of 384×384 . During both pre-training and fine-tuning, we process the global image I_g alongside the four local patches I_{li} in parallel, generating a total of 5×576 visual tokens. This multi-resolution strategy allows our model to capture fine-grained, high-resolution features while preserving global context. All extracted tokens are then passed through the ConvPooler for further processing.

Visual Token Pooler: Recent work has shown that high-resolution images with token downsampling work better than low-resolution images (Lin et al., 2024b). A single 384-resolution image processed by the ViT generates 576 visual tokens. For high-resolution processing, multiple patches of the image are used, concatenating their respective visual tokens before passing them into the LLM. However, simple high-resolution approaches—such as splitting an image into four patches significantly increase the number of visual tokens (Table 2), leading to higher pre-filling latency and KV Cache size expansion.

Our visual token pooler is based off a simple fact that spatially local tokens exhibit high similarity, while distant tokens have lower correlation. This motivated the design of a spatial visual token pooler, which applies a convolution layer to pool a 2×2 grid of visual tokens into a single visual token:

$$E_g^{\text{pooled}} = \text{ConvPool}(E_g) \quad (4)$$

$$E_l^{\text{pooled}} = \text{ConvPool}(E_l) \quad (5)$$

Our visual token pooler is implemented as a single convolutional layer with kernel size = stride = 2, allowing it to capture locally important spatial features while reducing the number of visual tokens by 4×. We initialize the kernel weights to

0.25, mimicking an average pooling operation at initialization, and allow the model to learn the optimal kernel values during training. ConvNeXt outputs are excluded from pooling, as their 49 tokens (7×7 grid) already represent condensed spatial hierarchies. We process both global as well as local tokens using the token pooler reducing the number of tokens from 5×576 to 5×144.

The overall pipeline is presented in Figure 1. Finally we use E_g^{pooled}, E_l^{pooled} and $\hat{E}_{cnn}$ as inputs to the LLM.

3.2 Data Preparation

Pretraining Data. We curated a dataset by combining the LAION subset of ALLaVA (Chen et al., 2024a) and the LLaVA-pretrain subset of ShareGPT4V-PT (Chen et al., 2024b), resulting in a total of 1.03 million image-text pairs. Both datasets contain synthetically generated captions, providing higher-quality descriptions. Given that TinyVLM is a compact model, training on smaller datasets with shorter captions led to unstable convergence and poor generalization. By selecting a balanced dataset with both diversity and high-quality captions, we ensured effective learning of vision-language representations.

Finetuning Data. For fine-tuning, we curated a dataset by selecting specific sections from The Cauldron (Laurençon et al., 2024), LNQA[3], ShareGPT4o[4], and Docmatix (Laurençon et al., 2024), resulting in 2.70 million image-text pairs. Each dataset contributes a unique aspect to the model's training: *The Cauldron* is a large collection of 50 vision-language datasets used in Idefics2 and Idefics3, covering science, mathematics, documents, charts, and tables. *Docmatix* focuses on OCR and document understanding, enhancing text extraction capabilities. *LNQA* provides real-world environmental knowledge, improving generalization. *ShareGPT4o* is a smaller dataset with high-quality, detailed captions that refine captioning ability.

This diverse dataset enables TinyVLM to acquire broad multimodal capabilities while maintaining a compact architecture. To ensure the model retains the ability to generate coherent text, we also included OpenHermes(Teknium, 2023) and MathInstruct (Xiang Yue, 2023), which are text-only datasets.

[3] https://huggingface.co/datasets/vikhyatk/lnqa

[4] https://huggingface.co/datasets/OpenGVLab/ShareGPT-4o

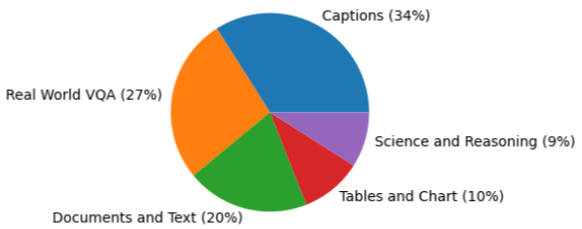

Figure 2: Category wise distribution of the fine-tuning dataset. Note that this data is sampled from the Cauldron (Laurençon et al., 2024), LNQA[5], ShareGPT4o[6], and Docmatix (Laurençon et al., 2024)

3.3 Training Setup

The training of TinyVLM is conducted in three sequential stages: pre-training, fine-tuning, and CNN-augmented fine-tuning. Each stage is designed to progressively enhance the model's ability to process and generate responses based on both visual and textual inputs.

Stage 1 - Pre-training. In the pre-training phase, the model is trained on 1.03 million image-caption pairs sourced from two synthetically generated caption datasets. The architecture of TinyVLM comprises four primary components: vision encoders, connectors, ConvPoolers, and LLM. During pre-training, only the connector and ConvPooler weights are updated, while the rest of the model remains frozen. Note that the CNN and it's cross-attention connector is not introduced in the pre-training phase at all.

The model is optimized using the next-token prediction objective to generate coherent captions and responses based on image embeddings. This stage is crucial for establishing a foundational understanding of vision-language relationships. The pre-training phase required 14.8 A100 GPU hours.

Stage 2 - Fine-tuning. Following pre-training, the model undergoes instruction fine-tuning on a diverse dataset of 2.7 million image-text pairs derived from multiple instruction datasets. Unlike pre-training, all model weights are updated in this phase, allowing the model to fully adapt to instruction-based interactions.

To maintain strong instruction-following and reasoning capabilities, the model is periodically trained on text-only data. Specifically, we incorporate OpenHermes and MathInstruct, two instruction-tuning datasets, every 25 and 100 iterations, respectively. This ensures that the LLM retains its ability to perform structured reasoning

Resolution	# Visual Tokens	KV Size (MB)	TTFT (ms)	MMMU	TVQA	POPE	RWQA	Avg.
224	49×5	3.0	24.6	34.2	22.3	78.2	41.6	44.1
384	144×5	8.8	28.1	32.8	35.1	82.6	41.4	48.0
512	256×5	15.7	33.9	33.2	34.2	81.1	42.2	47.7

Table 1: Comparison of different resolutions and their effect on memory usage, model performance and efficiency. Note that the ViT patch size is 16 across these experiments. TTFT denotes the Time to First Token and KV Size denotes the per sample KV Cache size contributed by the visual tokens.

Method	# Visual Tokens	KV Size (MB)	TTFT (ms)
Single image	576	7.0	25.3
+ 4 zoom-images	2880	35.3	40.4
+ ConvPooler	720	8.8	28.1

Table 2: Compute comparison of Single image and our ConvPooler. ConvPooler successfully models zoomed images while limiting the total number of visual tokens substantially.

Pooler	Δ Param	MMMU	TVQA	POPE	RWQA
Mean Pooling	0	32.4	45.9	85.4	45.9
ConvPooler	2.4 M	34.4	44.81	85.7	46.8

Table 3: Comparison of ConvPooler with the baseline Mean Pooling.

and general instruction following.

A key modification in this stage is that the loss is computed only on answer tokens, ensuring the model prioritizes generating high-quality responses rather than replicating the entire input structure. Additionally, NEFTune (Jain et al., 2023) is applied to the language model embeddings, introducing controlled noise to enhance generalization and robustness. The fine-tuning phase required 77.5 A100 GPU hours.

Stage 3 - CNN-Augmented Fine-tuning. In the final stage, a masked auto-encoder model - ConvNeXT-Tiny (Woo et al., 2023) is integrated into the model architecture to enhance spatial feature extraction. We wanted to leverage spatial features learned through self-supervised learning and hence choose this 28M CNN based masked auto-encoder model. This CNN-based component generates 49 spatial visual tokens, which are processed through the cross-attention connector alongside the already fine-tuned ViT's output (acting as keys and values). The resulting tokens are then appended to the original image tokens, enabling the model to capture fine-grained visual details more effectively.

During this phase, the model is fine-tuned on 400K image-text pairs, refining its performance with this enhanced vision-text representation. This stage required 14 A100 GPU hours.

Total Training Cost: The complete training process takes 106.3 A100 hours and $159 making our proposed pipeline one of the most efficient approaches for training compact VLMs.

4 Evaluation

We evaluate the multimodal capabilities of VLMs on five datasets: POPE (Li et al., 2023b), which detects object hallucinations; TextVQA (Singh et al., 2019), which assesses the ability to read and reason about text in images; MMMU (Yue et al., 2024), which tests college-level subject knowledge and deliberate reasoning across six core disciplines—Art & Design, Business, Science, Health & Medicine, Humanities & Social Science, and Tech & Engineering; RealWorldQA, a benchmark designed for real-world understanding; and VQAv2 (Goyal et al., 2017), which requires a combination of vision, language, and commonsense knowledge to generate accurate answers.

Image Resolution. Input image resolution plays a crucial role in balancing fine-grained feature learning and inference latency. Table 1 presents the performance of our method across three different resolutions. In order to efficiently study multiple paradigms of our proposed model, we limit the fine-tuning steps to 25K out of a total of 169K for Table 1. We qualitatively observed that 25K steps are a good proxy for the final fine-tuned model and hence sufficient to make decisions regarding the model architecture. Note that we pre-train all the models with the same number of steps and only use a proxy for the fine-tuning stage.

The $\times 5$ factor in the visual token count accounts for both global and local tokens, resulting from our zooming strategy. We observe that the 384-resolution model significantly outperforms the 224-resolution variant, while further scaling to 512 resolution provides only marginal improvements. Higher resolutions are particularly beneficial for fine-grained tasks such as text recognition, as evidenced by the performance gains in TextVQA.

However, the increased number of visual tokens at higher resolutions negatively impacts model ef-

	TTFT (ms)	KV Size (MB)	MMMU	TVQA	POPE	RWQA	VQAv2	Average
without CNN	28.1	8.8	34.4	44.8	85.7	46.8	66.36	52.94
with CNN	35.3	9.4	34.7	47.7	84.5	47.4	69.74	56.80

Table 4: Effect of visual feature enrichment by a CNN. With minimal rise in KV Size, CNN improves performance by upto 4%.

Model	Parameters	TTFT (ms)	MMMU	TVQA	POPE	RWQA	VQAv2	Average
DeepseekVL	2.0 B	30.1	32.2	-	87.6	-	-	-
MobileVLM	1.7 B	27.3	30.3	41.5	84.5	42.9	68.1	53.4
MobileVLM-V2	1.7 B	25.5	30.7	52.1	84.3	46.3	73.0	57.3
NanoLLaVA	1.1 B	71.9	30.4	46.7	84.1	44.0	70.8	55.2
Ours	0.6 B	35.3	34.7	47.7	84.5	47.4	69.7	56.8

Table 5: Comparison with existing compact VLMs. TinyVLM achieves performance comparable to state-of-the-art models while maintaining a significantly smaller parameter footprint.

ficiency. To mitigate this, our visual token pooler effectively reduces the number of tokens passed to the LLM. As shown in Table 1, at 384 resolution, rather than processing 576 tokens per image, we utilize only 144 pooled tokens per image, significantly reducing computational overhead.

Based on these findings, we fix 384 as the default input resolution across all our experiments.

Visual Token Pooler. As described in the previous section, limiting the number of visual tokens is an important factor in building VLMs. Table 2 shows that adding zoomed images into the pipeline substantially increases the number of visual tokens. Additionally, processing all 2880 tokens from the original and zoomed-in sub-images leads to a quadratic increase in TTFT and results in a very large context size for a small language model.

To address this, we explore two simple pooling strategies, with results presented in Table 3. ConvPooler improves upon a simple 2×2 mean pooling approach by learning an optimized token pooling convolutional kernel. This allows TinyVLM to aggregate high-resolution features while maintaining a reasonable token count, thereby reducing TTFT and KV cache size substantially.

Visual Feature Enrichment. While ViTs excel at capturing global context, they often struggle with fine-grained spatial details due to the absence of localized inductive biases. In contrast, CNNs inherently model spatial hierarchies through locality and translation equivariance, making them an effective complement to ViTs. To leverage these advantages, we integrate ConvNeXt-Tiny as an auxiliary feature extractor, generating 49 additional tokens. These tokens enrich the model's visual representation by capturing fine-grained spatial information and reinforcing structured visual patterns, effectively complementing transformer-based global features.

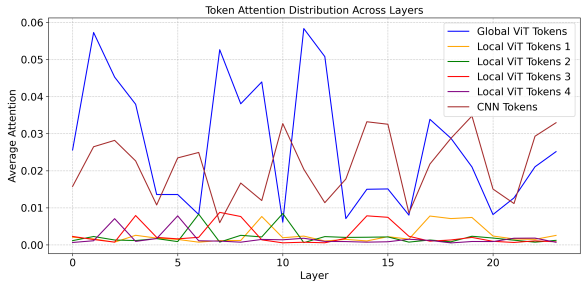

Figure 3: Average attention across layers and token types computed with respect to the answer tokens.

Table 4 demonstrates the substantial performance improvements achieved by incorporating a CNN alongside a ViT as the vision encoder. This enhancement is particularly beneficial for text understanding tasks. However, adding a small CNN introduces a slight increase in hallucinations, as indicated by the POPE score.

We analyze the average attention distribution across different types of visual tokens during the decoding of answer tokens. As illustrated in Figure 3, the CNN-generated tokens receive significantly higher attention. This indicates that the CNN tokens play a more influential role in shaping the model's generated responses.

Finally, Table 5 compares our model with existing compact vision-language models, including DeepSeek-VL (Lu et al., 2024), MobileVLM (Chu et al., 2023), MobileVLM V2 (Chu et al., 2023), and NanoLLaVA. our model achieves competitive performance while maintaining a significantly smaller footprint. It outperforms NanoLLaVA and MobileVLM substantially while being nearly half its size, demonstrating a strong balance between accuracy and efficiency. Compared to larger mod-

Precision	MMMU	TVQA	POPE	RWQA	VQAv2	Average
W16A16	34.7	47.7	84.5	47.4	69.7	56.8
W4A16	33.5	46.2	85.1	47.2	69.3	56.3
W4A8	31.1	39.2	83.7	40.1	66.4	52.1

Table 6: TinyVLM is compatible with existing quantization algorithms while retaining performance at high precision.

Device	TTFT	Throughput (toks/s)
NVIDIA A100	35.3 ms	1880.1
Intel Xeon - 8 Core CPU	3.5 s	18.0

Table 7: Latency and Througput of TinyVLM on A100 GPU machine and 8-core CPU machine.

els like MobileVLM-V2 and DeepSeek-VL, our approach offers improved feasibility for real-world deployment, particularly in resource-constrained environments.

Deployment on the Edge. Deploying VLMs on edge devices presents significant challenges due to limited compute power, memory constraints, and latency requirements. TinyVLM is designed with efficiency in mind, enabling deployment on resource-constrained platforms while maintaining strong performance.

We benchmark inference latency on both a high-performance A100 GPU and an Intel Xeon 8-core CPU Machine with Platinum 8370C CPU. As shown in Table 7, TinyVLM achieves low-latency inference on the CPU machine, making it a practical solution for real-time applications such as robotics, assistive technologies, and mobile AI systems. TinyVLM runs at 18 tokens/sec on a CPU-only system enabling real world visual-language applications on edge hardware. Note that these numbers are presented for a fp16 model and quantization can further improve the same.

To assess its feasibility for edge deployment and compatibility with existing quantization methods, we quantize TinyVLM using AWQ (Lin et al., 2024a) and QoQ (Lin et al., 2024c) for W4A16 and W4A8 precision respectively. Table 6 compares these quantizations with the baseline model, showing that while W4A8 quantization introduce some performance degradation, W4A16 maintains competitive accuracy while significantly reducing computational overhead. At 4-bit precision, TinyVLM takes a mere 300 MB memory, enabling efficient edge deployment.

Limitations

Despite the efficiency and competitive performance of TinyVLM, several limitations remain. First, while our model employs a CNN-based feature extractor and visual token pooling to reduce computational overhead, it still relies on a Vision Transformer (ViT) backbone, which can be resource-intensive for extremely low-power edge devices. Although quantization techniques such as W4A16 and W4A8 (Table 6) mitigate this to some extent, further exploration of distillation-based approaches or hardware-aware optimizations could improve deployment feasibility on constrained hardware.

Second, our visual token compression strategy effectively reduces the number of tokens fed into the LLM, improving inference efficiency. However, aggressive token reduction may lead to a loss of fine-grained spatial details, particularly in tasks that require precise text recognition or dense visual reasoning. While our CNN-guided token pooling retains critical features, further refinements in adaptive token selection strategies could help balance efficiency and spatial fidelity.

Third, our pretraining dataset, sourced from web-scale multimodal corpora, introduces inherent biases present in synthetically generated captions and internet-scraped image-text pairs. This may affect model robustness, particularly in specialized domains such as medical imaging, scientific document understanding, or low-resource languages. Addressing this requires better dataset curation, domain-adaptive training techniques, and controlled synthetic data generation to reduce spurious correlations.

Finally, while TinyVLM performs well on vision-language benchmarks, it has not been extensively tested on long-form reasoning tasks, instruction-following in low-data regimes, or few-shot generalization scenarios. Future work could explore in-context learning adaptations, retrieval-augmented generation (RAG), or meta-learning techniques to improve performance in settings where labeled multimodal data is scarce.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 1(2):3.

Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. 2021. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3558–3568.

Guiming Hardy Chen, Shunian Chen, Ruifei Zhang, Junying Chen, Xiangbo Wu, Zhiyi Zhang, Zhihong Chen, Jianquan Li, Xiang Wan, and Benyou Wang. 2024a. Allava: Harnessing gpt4v-synthesized data for a lite vision-language model. *arXiv preprint arXiv:2402.11684*.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. 2024b. Sharegpt4v: Improving large multi-modal models with better captions. In *European Conference on Computer Vision*, pages 370–387. Springer.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. 2022. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, et al. 2023. Mobilevlm: A fast, reproducible and strong vision language assistant for mobile devices. *arXiv preprint arXiv:2312.16886*.

Xiangxiang Chu, Limeng Qiao, Xinyu Zhang, Shuang Xu, Fei Wei, Yang Yang, Xiaofei Sun, Yiming Hu, Xinyang Lin, Bo Zhang, et al. 2024. Mobilevlm v2: Faster and stronger baseline for vision language model. *arXiv preprint arXiv:2402.03766*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words. *arXiv preprint arXiv:2010.11929*, 7.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR.

Neel Jain, Ping-yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, et al. 2023. Neftune: Noisy embeddings improve instruction finetuning. *arXiv preprint arXiv:2310.05914*.

Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. 2024. Building and better understanding vision-language models: insights and future directions. In *Workshop on Responsibly Building the Next Generation of Multimodal Foundational Models*.

Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander Rush, Douwe Kiela, et al. 2023. Obelics: An open web-scale filtered dataset of interleaved image-text documents. *Advances in Neural Information Processing Systems*, 36:71683–71702.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *Preprint*, arXiv:2405.02246.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023a. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023b. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024a.

Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. 2024b. Vila: On pre-training for visual language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26689–26699.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. 2024c. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. 2024b. Llava-next: Improved reasoning, ocr, and world knowledge.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, et al. 2024. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Pranav Rajpurkar, Emma Chen, Oishi Banerjee, and Eric J Topol. 2022. Ai in health and medicine. *Nature medicine*, 28(1):31–38.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294.

Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. 2019. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13693–13696.

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

Rahul Thapa, Kezhen Chen, Ian Connick Covert, Rahul Chalamala, Ben Athiwaratkun, Shuaiwen Leon Song, and James Zou. 2024. Dragonfly: Multi-resolution zoom-in encoding enhances vision-language models.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. 2023. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16133–16142.

Ge Zhang Yao Fu Wenhao Huang Huan Sun Yu Su Wenhu Chen Xiang Yue, Xingwei Qu. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Nur Yildirim, Hannah Richardson, Maria Teodora Wetscherek, Junaid Bajwa, Joseph Jacob, Mark Ames Pinnock, Stephen Harris, Daniel Coelho De Castro, Shruthi Bannur, Stephanie Hyland, et al. 2024. Multimodal healthcare ai: identifying and designing clinically relevant vision-language applications for radiology. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–22.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986.

Figure 4: Qualitative samples demonstrating wide abilities of the proposed TinyVLM.

Appendix

	Stage 1	**Stage 2**	**Stage 3**
Number of Steps	16K	169K	25K
Learning rate (max, min)	$(1e^{-5}, 5e^{-6})$	$(1e^{-6}, 0)$	$(5e^{-7}, 5e^{-7})$
LR Scheduler	Linear	Linear	Constant
Batch Size	64	16	16
Train Vision Encoder	✗	✓	✓
Train Connector and Pooler	✓	✓	✓
Train Language Model	✗	✓	✓
Train CNN	NA	NA	✓
Data	ALLaVA ShareGPT4V-PT	The Cauldron, LNQA, Docmatix, ShareGPT4o	Subset of Stage 2

Table 8: Training stages and their corresponding parameters and datasets.