SHBT 261 Final Project: Visual Understanding with TextVQA

Qianyu Fan, Jasper Liu, Lavinia Xu

December 12, 2025

1 Introduction

Vision–language models (VLMs) aim to bridge visual perception and natural language understanding, enabling systems to answer questions about images, documents, and real-world scenes. While recent models have achieved strong results on general multimodal benchmarks, tasks that require precise reading and reasoning over embedded text remain challenging. This project focuses on **TextVQA**, a representative benchmark designed to evaluate these capabilities.

1.1 Problem Statement

Visual Question Answering (VQA) is a core task in multimodal AI that requires a model to jointly understand visual content and natural language queries. Among existing benchmarks, TextVQA represents a particularly challenging setting, as answering each question typically requires reading text embedded in natural images—such as street signs, product labels, phone screens, or instrument readouts—rather than relying solely on object appearance or language priors. As a result, TextVQA lies at the intersection of computer vision, optical character recognition (OCR), and natural language understanding.

From a technical perspective, models evaluated on TextVQA must address three tightly coupled challenges. First, they must accurately recognize text under diverse visual conditions, including variations in font, scale, orientation, illumination, occlusion, and cluttered backgrounds. Second, they must correctly ground linguistic expressions in the visual scene, mapping words and phrases in the question to specific image regions or OCR tokens instead of hallucinating plausible but unsupported answers. Third, they must perform contextual reasoning that connects recognized text to the semantics of the question, such as identifying which number on a dashboard corresponds to "miles remaining" or which string on a receipt encodes the "store name".

Despite rapid progress in large vision–language models (VLMs), recent studies suggest that text-centric VQA remains a significant open problem. Models that perform strongly on conventional VQA datasets often degrade substantially when precise reading and fine-grained text grounding are required. This gap motivates the focus of this project. Specifically, we investigate whether **Low-Rank Adaptation** (LoRA), a parameter-efficient fine-tuning technique, can meaningfully improve the performance of a compact VLM on TextVQA.

Our study centers on the **Qwen2.5-VL-3B-Instruct** model and asks the following question: to what extent can LoRA fine-tuning enhance a mid-scale vision–language model's ability to read and reason over embedded text in real-world images, under realistic computational constraints? To answer this question, we fine-tune the model on the TextVQA training set using LoRA adapters, evaluate zero-shot and fine-tuned performance on the validation set, and analyze results across question types, with particular emphasis on OCR-heavy categories.

1.2 Background and Related Work

Recent advances in large-scale VLMs have enabled strong performance on tasks such as image captioning, generic VQA, and multimodal instruction following. However, text-heavy visual understanding, where success depends critically on accurate OCR, localization, and reasoning over embedded text tokens, remains comparatively underexplored. TextVQA was explicitly designed to probe these abilities by requiring models to answer questions whose solutions depend on reading text in natural, non-document images. In parallel, parameter-efficient fine-tuning (PEFT) methods such as LoRA

have emerged as practical tools for adapting large models in low-resource settings by updating only a small number of additional parameters. This project lies at the intersection of these two lines of work, studying how far PEFT can push a modern VLM on a text-centric benchmark.

1.2.1 Vision-Language Models for Text-Heavy VQA

We focus on widely used open-source VLMs recommended for the course project and evaluate their suitability for TextVQA. **BLIP-2** (1) introduces a Querying Transformer (Q-Former) that interfaces a frozen vision encoder with a frozen language model via a small set of learnable query tokens, achieving efficient performance on captioning and general VQA. However, its pre-training is largely general-purpose rather than OCR-oriented, and prior results report only moderate accuracy on text-centric tasks.

MiniGPT-4 (2) aligns a vision encoder with a 7B-parameter language backbone using a learnable projection, yielding strong multimodal conversational abilities. This capability comes at a substantial computational cost: even under 4-bit quantization, fine-tuning typically requires 24–28 GB of GPU memory, making systematic ablation studies impractical in a student-scale environment.

LLaVA-Phi-3-mini (3) follows the LLaVA paradigm with a lightweight Phi-3 language model, offering competitive efficiency and robust general visual reasoning. However, available evaluations suggest that its zero-shot performance on TextVQA remains below that of more OCR-focused models, reflecting a pre-training corpus not specifically tailored to text-heavy scenes.

Qwen2.5-VL (4) is a multilingual family of VLMs incorporating a dynamic-resolution visual encoder. By processing images at adaptive resolutions without aggressive aspect-ratio distortion, the model better preserves small or elongated textual elements such as receipts or prescription labels. Public benchmarks and early community reports indicate strong text-reading capabilities and competitive performance on text-centric tasks.

InternVL2.5 (5) provides a scalable suite of VLMs trained via a multi-stage curriculum integrating large-scale image–text, OCR, and instruction data. While larger variants achieve state-of-the-art results across many multimodal benchmarks, the framework is architecturally complex and smaller variants are less thoroughly documented.

1.2.2 Comparative Analysis and Model Selection

We conducted a systematic evaluation of candidate models based on four criteria, elaborated in Table 1:

(1) Text-reading capabilities (OCR performance)

(2) Computational feasibility (hardware requirements)

(3) Baseline appropriateness (zero-shot headroom for improvement)

(4) Engineering maturity (documentation and tooling support)

Model	Parameters	OCR Capability	Hardware	TextVQA Zero-shot	Engineering Support	Key Limitations
BLIP-2	2.7B	Moderate	10GB	$\sim 30\%$	High (HF native)	Not OCR-optimized
MiniGPT-4	7B	Good	24-28GB	$\sim 36\%$	Medium	Exceeds GPU budget
LLaVA-Phi-3	3.8B	Good	15GB	$\sim 32\%$	High (HF native)	Lower baseline than Qwen2.5
Qwen2.5-VL-3B	3B	Excellent	12GB	$\sim 35\%$	Very High	None (selected model)
InternVL2.5-1B	1-78B	Excellent	8GB+	$\sim 38\%$	Medium	Complex setup; limited docs

Table 1: Comparison of vision-language models for TextVQA

BLIP-2, while computationally efficient, demonstrates inadequate performance on text-centric tasks, achieving approximately 30% zero-shot accuracy on TextVQA, notably lower than Qwen2.5-VL's 35%. Prior work by Singh et al. (6) highlights the importance of OCR-oriented pre-training for this benchmark, a limitation corroborated by our preliminary experiments.

MiniGPT-4 is unsuitable due to resource constraints. Its 7B-parameter backbone exceeds available GPU memory, requiring 24–28 GB even under 4-bit NF4 quantization (7). Given its marginal performance gains over Qwen2.5-VL, this overhead cannot be justified.

LLaVA-Phi-3-mini offers competitive efficiency but underperforms on TextVQA, consistent with evidence that OCR-focused pre-training yields measurable improvements (8). In contrast, Qwen provides stronger documentation and fine-tuning support.

Although InternVL2.5 achieves strong results broadly, its complex training pipeline and limited evaluation of smaller variants raise reproducibility concerns. Larger models further exceed our computational budget.

In contrast, **Qwen2.5-VL-3B-Instruct** occupies a favorable accuracy–efficiency trade-off. Its dynamic-resolution encoder, strong text-reading capability, and modest parameter count enable LoRA fine-tuning and systematic ablation studies on a 12 GB GPU. Combined with mature tooling and documentation, these factors motivate our choice of Qwen2.5-VL-3B-Instruct as the base model for this project.

2 Methods

2.1 Dataset and Preprocessing

We conduct experiments on the TextVQA benchmark hosted on Hugging Face (`lmms-lab/textvqa`), derived from the original dataset by Singh et al. (6). The dataset contains 34,602 training, 5,000 validation, and 5,734 test question–answer pairs built on images from the Open Images dataset. Each question requires the model to read and reason about text embedded in real-world scenes such as signs, product labels, and digital displays. All experiments in this paper use the training and validation splits; the held-out test labels are not publicly available.

- ***Data Loading.*** We follow the official Hugging Face configuration and load the dataset in Parquet format, which stores JPEG-encoded image bytes together with the corresponding question text, candidate answers, and OCR tokens. All experiments in this paper are conducted on the training and validation splits; the held-out test labels are not publicly available.

- ***Image Processing.*** Images are processed using the `Qwen2.5-VL` visual preprocessor, which supports dynamic resolution. Instead of resizing every image to a fixed square resolution, we preserve the original aspect ratio and rescale the image so that the height and width become multiples of the Vision Transformer (ViT) patch size (14×14 pixels) and satisfy the model's token budget. This avoids the severe distortion that would arise from aggressive cropping or squaring, which is particularly important for elongated textual content such as banners, receipts, or scoreboard displays.

- ***Text Formatting and Prompt Construction.*** For each sample, we construct a single-turn multimodal prompt using the Qwen2.5-VL chat format:

  ```
  User: <image> + question text
  Instruction: Answer using a single word or short phrase.
  Assistant: model output
  ```

  The explicit instruction encourages concise answers and reduces the tendency of instruction-tuned models to generate verbose, conversational outputs that are incompatible with the TextVQA evaluation protocol.

- ***Batching and Collation.*** We implement a custom collation function that tokenizes question text, invokes the vision processor to obtain image patch embeddings, and pads variable-length sequences to form mini-batches. Because dynamic resolution produces different numbers of visual tokens per image, padding is applied only along the sequence dimension, and attention masks are used to exclude padded positions during computation.

2.2 Base Model and LoRA Fine-tuning

Our experiments are based on **Qwen2.5-VL-3B-Instruct**, a 3-billion-parameter multilingual vision–language model released by the Qwen team. The architecture consists of a Vision Transformer encoder with dynamic-resolution patch embedding (patch size 14×14) and a Transformer decoder-style language model initialized from Qwen2.5. Visual tokens produced by the encoder are projected into the language model's embedding space and integrated via cross-attention, allowing the decoder to jointly attend to textual and visual information.

We treat the off-the-shelf instruction-tuned checkpoint as our zero-shot baseline, which already achieves competitive performance on TextVQA due to large-scale image–text pre-training. This baseline provides sufficient headroom to assess whether parameter-efficient fine-tuning can further specialize the model for text-centric visual question answering. We adapt the base model to TextVQA using LoRA, a parameter-efficient fine-tuning method that freezes all original model weights and injects trainable low-rank matrices into selected linear projections. Specifically, a weight matrix W is modified as

$$W' = W + \frac{\alpha}{r} \cdot AB,$$

where $A \in R^{d \times r}$ and $B \in R^{r \times d}$ are trainable matrices, r is the LoRA rank, and α is a scaling factor. This approach substantially reduces the number of trainable parameters and memory usage while preserving the base model's pretrained representations.

LoRA is motivated by the intrinsic dimensionality hypothesis, which posits that task-specific updates to large pretrained models lie in a low-dimensional subspace of the full parameter space (9). By parameterizing weight updates as a rank-r decomposition $W+AB$, LoRA reduces trainable parameters from $O(d^2)$ to $O(2dr)$ with $r \ll d$.

For TextVQA, this provides a suitable inductive bias: the pretrained Qwen2.5-VL already encodes strong vision–language representations, so fine-tuning mainly requires learning task-specific routing (e.g., mapping question types to answer formats and attending to OCR tokens). Full fine-tuning risks catastrophic forgetting, whereas LoRA preserves the base weights while adding small, task-specific corrections.

In our setup, we use a LoRA rank of $r = 16$, a scaling factor of $\alpha = 32$, and a LoRA dropout of 0.05. Adapters are applied to both attention and feed-forward components of the language model, including the query, key, value, and output projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`) as well as the MLP projections (`gate_proj`, `up_proj`, `down_proj`). This placement enables the model to learn task-specific attention patterns that better align questions with relevant visual regions and OCR tokens, while leaving the majority of pretrained parameters unchanged.

2.3 Training Configuration

We fine-tune Qwen2.5-VL-3B-Instruct on the TextVQA training split using the LoRA adapters described above. Unless otherwise stated, all reported results use the same training configuration. Optimization is performed with **AdamW** for 3 epochs, using a peak learning rate of 2×10^{-4} applied to LoRA parameters only. Due to the memory cost of dynamic-resolution visual tokens, we use a per-device batch size of 2 and accumulate gradients for 8 steps, resulting in an effective batch size of 16. We adopt a linear warm-up followed by cosine decay for the learning rate schedule and do not apply additional data augmentation beyond the standard resizing performed by the vision processor.

Training follows a standard causal language modeling objective. For each example, the model is conditioned on the user message (image, question, and instruction) and trained to autoregressively generate the ground-truth answer string using teacher forcing, with tokens beyond the answer boundary masked from the loss. All experiments are conducted using 4-bit NF4 quantization for frozen base weights, while LoRA parameters are maintained in 16-bit precision.

2.4 Evaluation Metrics

2.4.1 Primary Metric: TextVQA Soft Accuracy

Following the official TextVQA evaluation protocol, we report soft accuracy as the primary metric. Each question is annotated with answers from 10 human annotators. Given a model prediction a and

the multiset of ground-truth answers $a_1, \ldots, a_{10}$, soft accuracy is defined as

$$\mathrm{Acc}(a) = \min(\frac{|i : a_i = a|}{3}, 1),$$

after applying the standard normalization pipeline, including lowercasing, punctuation removal, and whitespace trimming.

This scoring scheme assigns full credit when at least three annotators provide an answer identical to the model's prediction, while still rewarding predictions that match a smaller but non-negligible subset of annotators. It is designed to account for semantic agreement despite variation in surface forms, such as "$5," "5 dollars," or "5.00." Unless otherwise stated, **accuracy** in this paper refers to TextVQA soft accuracy evaluated on the validation split.

2.4.2 Secondary String-based Metrics

To provide a more nuanced view of model behavior beyond soft accuracy, we additionally report several complementary string-based evaluation metrics, all computed after the same normalization step described above.

- ***Exact Match (EM).*** A binary indicator equal to 1 if the model prediction exactly matches at least one ground-truth answer, and 0 otherwise. EM is stricter than soft accuracy and serves as a conservative estimate of performance.
- ***Token-level F1.*** Precision, recall, and F1 are computed based on overlapping whitespace-tokenized words between the prediction and reference. This metric is particularly informative for multi-word answers where the model may partially capture the correct content.
- ***BLEU.*** We compute BLEU-n as proposed by Papineni et al. (10), measuring n-gram overlap between predictions and references. Because many TextVQA answers are short (often 1–3 tokens), BLEU is not treated as a primary metric but is included for comparability with prior NLP work.
- ***METEOR.*** METEOR aligns predictions and references using surface forms, stems, and synonyms, and computes a harmonic mean of precision and recall with a fragmentation penalty (11). Its semantic matching makes it more tolerant to paraphrases than BLEU.
- ***ROUGE-L.*** ROUGE-L is based on the longest common subsequence between prediction and reference, from which an F1-like score is derived (12). It captures global sequence similarity while balancing precision and recall.
- ***Substring precision and recall.*** To diagnose over- and under-generation, we report two simple substring-based diagnostics:
  - **Substring precision** is fraction of predictions that appear as contiguous substrings of at least one reference answer.
  - **Substring recall** is the fraction of reference answers that contain the prediction as a substring.

  Low precision indicates hallucinated extra content, while low recall suggests overly short or underspecified predictions.

These secondary metrics are used for error analysis and ablation studies, rather than as primary optimization targets.

2.4.3 LLM-as-a-Judge Semantic Evaluation

String-matching metrics cannot fully capture semantic equivalence, particularly for open-class entities (e.g., "NYC subway card" vs. "MetroCard"). To better approximate human semantic judgments at scale, we adopt an **LLM-as-a-Judge** evaluation protocol inspired by Zheng et al. (13).

For each validation example, we construct a text-only evaluation prompt containing the question, the set of human reference answers, and the model's prediction. We use Qwen2.5-0.5B-Instruct (500M

parameters) as an independent judge to determine whether the prediction is semantically equivalent to the reference answers. The judge is prompted to output a binary response ("yes" or "no"), which we map to scores of 1 and 0, respectively. Averaging these scores over the validation set yields the LLM similarity metric.

To optimize computational efficiency, we implement a two-stage evaluation strategy: (1) exact string matching after normalization, which directly assigns a score of 1 for matches, and (2) LLM-based evaluation for non-matching pairs. This approach significantly reduces the number of LLM inference calls while maintaining evaluation quality.

We chose Qwen2.5-0.5B-Instruct for several reasons: (1) it is lightweight (∼1GB) and efficient for large-scale evaluation, (2) it is open-source and fully reproducible, eliminating concerns about API versioning, and (3) it demonstrates strong language understanding capabilities despite its compact size. While this metric is supplementary to TextVQA soft accuracy (Section 2.4.1), the LLM similarity scores provide valuable insights into semantic equivalence beyond exact string matching, particularly for cases involving paraphrases or different surface forms of the same answer.

2.4.4 Per-Category Performance Analysis

To probe model strengths and weaknesses across different reasoning types, we perform a per-category analysis on the validation split. We partition questions into nine semantic categories shown below in Table 2 based on rule-based pattern matching over the question text, followed by manual verification on a random subset.

Category	Samples	Description
Brand	589	Queries about brand names, logos, or identifiers (e.g., "What brand is this phone?")
Text	379	Questions requiring reading free-form text (e.g., "What does the sign say?")
Number	905	Numerical quantities, prices, or counts (e.g., "What is the price?")
Time	197	Clock times, dates, or years (e.g., "What time is shown?")
Person	294	Names of people associated with objects in the image (e.g., "Who is the author?")
Yes/No	266	Binary decision questions (e.g., "Is this a Nike shoe?")
Color	39	Questions about the color of text or objects (e.g., "What color is the text?")
Location	189	Geographic locations or place names (e.g., "Where is this located?")
Other	2,142	Remaining questions not covered by the above categories

Table 2: Semantic categories used for per-category TextVQA analysis

For each category, we report TextVQA soft accuracy along with selected secondary metrics. This breakdown enables fine-grained analysis of where LoRA fine-tuning yields the largest gains—particularly in OCR-centric categories such as Text, Brand, and Number—and where performance remains limited, such as in more compositional Other questions.

3 Experimental Design

3.1 Overall Setup

Because the official TextVQA test labels are not publicly available, we treat the validation split (5,000 question–answer pairs) as our held-out evaluation set and report all results on this split. We consider two primary experimental settings:

(1) **zero-shot baseline**, in which the off-the-shelf `Qwen2.5-VL-3B-Instruct` checkpoint is evaluated directly using the preprocessing and prompting scheme described in Section 2; and

(2) **LoRA fine-tuning**, in which the same model is adapted on the full training split (34,602 samples) using parameter-efficient updates.

This experimental design allows us to quantify both the intrinsic zero-shot capability of the base model and the performance gains achieved through task-specific adaptation under realistic computational constraints.

3.2 Implementation and Reproducibility

All experiments are implemented in Python using PyTorch, Hugging Face Transformers/Datasets, and the `peft` library for LoRA. Training is performed on a single 12-GB NVIDIA GPU using 4-bit quantization and gradient accumulation to accommodate dynamic-resolution visual tokens. To ensure reproducibility, all stochastic components, including Python, NumPy, PyTorch, and CUDA, are initialized with a fixed random seed of 42.

The codebase is organized as a modular, end-to-end pipeline that separates data processing, model definition, training, and evaluation. The main components are listed below, following the execution flow of an experiment:

- `src/data_loader.ipynb` loads the TextVQA dataset and performs preprocessing, prompt construction, and batching.
- `src/model.ipynb` defines the base Qwen2.5-VL model and applies LoRA adapters to the specified modules.
- `src/train_lora.ipynb` implements the fine-tuning loop, including optimization, scheduling, and checkpointing.
- `src/evaluate_zeroshot.ipynb` and `src/evaluate_finetuned.ipynb` run evaluation for zero-shot and fine-tuned models, respectively.
- `src/metrics.ipynb` implements all primary and secondary evaluation metrics.
- `src/analyze_results.ipynb` performs error analysis, aggregation, and visualization of results.
- `src/run.ipynb` serves as a lightweight entry point that configures experiments and launches training or evaluation.
- `src/metric.py` provides a standalone evaluation tool that uses Qwen2.5-0.5B-Instruct as an independent LLM judge to assess semantic equivalence.

This modular design enables systematic ablation studies, simplifies debugging, and allows individual components to be reused or modified without affecting the rest of the pipeline.

3.3 Training Procedure

For LoRA fine-tuning, we adopt the hyperparameters described above. Training is conducted for three epochs over the full TextVQA training split using AdamW with a learning rate of 2×10^{-4} applied exclusively to LoRA parameters. Due to the memory cost of dynamic-resolution visual tokens, we use a per-device batch size of 2 with 8-step gradient accumulation, yielding an effective batch size of 16. LoRA adapters are configured with rank $r = 16$, scaling factor $\alpha = 32$, and dropout rate 0.05. At each optimization step, the model is conditioned on the multimodal prompt (image, question, and instruction) and optimized using an autoregressive cross-entropy loss over answer tokens.

During training, we monitor TextVQA soft accuracy on the validation set every 500 optimization steps and select the checkpoint achieving the highest validation accuracy as the final model. After training completes, we run a unified evaluation pipeline that computes all metrics described in Section 2.4, including secondary string-based metrics, LLM-as-a-Judge semantic evaluation, and per-category performance breakdowns.

3.4 Ablation Studies

In addition to the main zero-shot and full-data fine-tuning experiments, we conduct three controlled ablation studies to isolate the effects of prompt design, adapter capacity, and training-data scale.

3.4.1 Ablation 1: OCR Token Augmentation

We first examine whether prompt engineering alone can improve zero-shot performance. Using the frozen `Qwen2.5-VL-3B-Instruct` model, we compare the standard prompt (image + question) with an OCR-augmented variant that appends dataset-provided OCR tokens to the user message:

```
User: <image>; OCR tokens: $t_1$, $\cdots$, $t_n$; Question: $q$.
Answer using a single word or short phrase based on the OCR tokens.
```

No parameters are updated; any performance difference is attributable purely to the additional textual context.

3.4.2 Ablation 2: LoRA Rank

To study the capacity–efficiency trade-off, we fine-tune models with different LoRA ranks $r \in 8, 16, 32$ on the same randomly sampled subset of 5,000 training examples (14.5% of the full dataset). Each model is trained for one epoch with a learning rate of 2×10^{-4} and without OCR augmentation during training. All configurations are evaluated on the full 5,000-example validation split, isolating the effect of adapter rank under a fixed data budget.

3.4.3 Ablation 3: Training-data Scale

Finally, we examine data efficiency by contrasting a small-data regime with full-data training while keeping the LoRA configuration fixed at $r = 16$. The small-scale setting uses 5,000 training examples for one epoch (approximately 312 update steps), whereas the full-scale setting uses all 34,602 examples for three epochs (approximately 6,500 updates). Both settings share the same learning rate and evaluation protocol. The resulting performance gap quantifies the marginal benefit of additional supervision for TextVQA under LoRA fine-tuning.

This design uses different epochs (1 for 5k, 3 for full), resulting in different training steps (312 vs 6,500). While this confounds data scale with training duration, we observe from validation curves that the 5k model plateaus within 1 epoch, suggesting the performance gap is primarily driven by data quantity rather than training length.

4 Results

This section presents the empirical results of our experiments on the TextVQA validation set. We begin with an overview of overall performance across different configurations, followed by ablation studies that examine the effects of OCR token augmentation, LoRA rank selection, and training data scale. All results reported below use the evaluation metrics described in Section 2.4.

4.1 Overall Performance

We first compare the zero-shot baseline against various fine-tuned configurations to evaluate the overall effectiveness of LoRA adaptation on TextVQA. The main results are summarized in Table 3, which reports accuracy as the primary metric along with several secondary string-based and semantic metrics. As shown, the zero-shot Qwen2.5-VL-3B-Instruct model achieves an accuracy of 67.03% on the validation set. Incorporating OCR tokens into the prompt without updating any model parameters already leads to a substantial improvement, raising accuracy to 74.94%. This indicates that explicit textual information alone provides a strong performance boost for text-centric VQA tasks.

Metric	Zero-shot	Zero-shot + OCR	FT (5k, r=8)	FT (5k, r=16)	FT (5k, r=32)	FT (full, r=16)
Accuracy (%)	67.03	74.94	78.29	77.89	77.86	**80.40**
Exact Match (%)	71.42	79.26	82.18	81.82	81.78	**84.32**
F1 Score	0.778	0.831	0.851	0.847	0.846	**0.867**
BLEU	0.03	0.03	0.03	0.03	0.03	0.03
METEOR	0.794	0.834	0.853	0.849	0.848	**0.869**
ROUGE-L	0.777	0.830	0.850	0.847	0.846	**0.867**
Substring Precision	0.740	0.827	0.846	0.844	0.842	**0.863**
Substring Recall	0.803	0.832	0.853	0.848	0.847	**0.870**
LLM-as-a-Judge	0.805	0.850	0.865	0.862	0.861	**0.880**

Table 3: Overall TextVQA performance across configurations

The strongest performance is achieved by the fully fine-tuned model with LoRA rank $r = 16$ trained on the full TextVQA training set. This configuration reaches an accuracy of **80.40%**, corresponding to an absolute improvement of **13.37%** over the zero-shot baseline. This result is listed at the bottom of Table 1 and represents the primary performance outcome of this project.

Beyond accuracy, all secondary metrics show consistent gains after fine-tuning. Exact Match increases from 71.42% to **84.32%**, while token-level F1 improves from 0.778 to **0.867**. METEOR and ROUGE-L also increase by approximately 0.08–0.09, and the LLM-as-a-Judge score rises from 0.805 to **0.880**, indicating improved semantic alignment across evaluation criteria.

For a more direct comparison between zero-shot and fully fine-tuned performance, Table 4 reports changes in the number of correct predictions along with improvement and regression statistics. As shown, fine-tuning converts **679** previously incorrect predictions into correct ones, while introducing only 10 regressions, resulting in an improvement-to-regression ratio of **68:1**. This demonstrates that LoRA fine-tuning yields overwhelmingly positive net gains.

Metric	Zero-shot	Fine-tuned (full, r=16)	Absolute Δ	Relative Δ
Correct Predictions	3,331 / 5,000	4,010 / 5,000	+679	+20.4%
Accuracy	66.62%	80.20%	+13.58%	+20.4%
Exact Match	71.42%	84.32%	+12.90%	+18.1%
F1 Score	0.778	0.867	+0.089	+11.4%
METEOR	0.794	0.869	+0.075	+9.4%
Improvements (wrong → correct)	–	679 cases	–	–
Regressions (correct → wrong)	–	10 cases	–	–
Improvement / Regression Ratio	–	68:1	–	–
Net Gain	–	+669 correct	+13.38%	–

Table 4: Comparison between zero-shot and fully fine-tuned performance

To visualize these trends, Figure 1 shows overall accuracy across configurations, including zero-shot, OCR-augmented zero-shot, partial- and full-data fine-tuning. Performance improves monotonically with increased task-specific adaptation, with the largest gains occurring after fine-tuning.

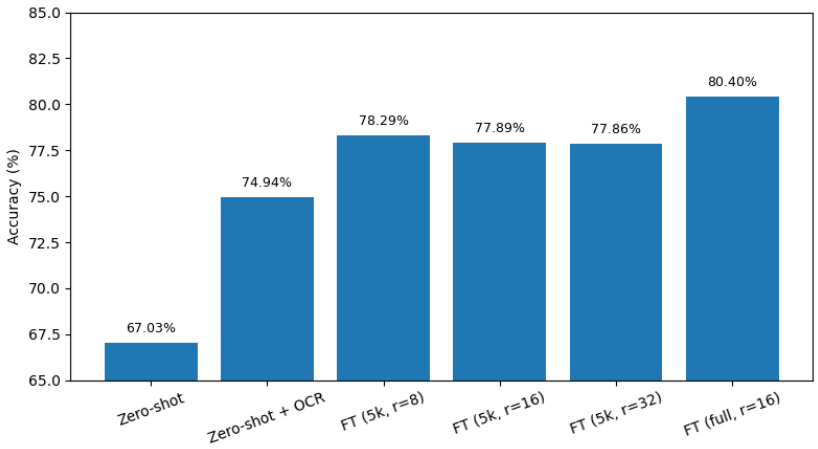

Figure 1: Overall TextVQA accuracy across configurations

4.2 Ablation Studies

To better understand which factors contribute most to the observed improvements, we conduct three ablation studies that isolate the effects of prompt design, LoRA adapter capacity, and training data scale. Unless otherwise stated, all ablations are evaluated on the TextVQA validation set using the same metrics as in Section 3.1.

4.2.1 Ablation Study 1: OCR Token Augmentation

The first ablation examines whether explicitly providing OCR tokens in the prompt improves zero-shot performance. The results of this comparison are reported in Table 5.

Configuration	Accuracy	Exact Match	F1	METEOR	ROUGE-L	LLM-as-a-Judge
Zero-shot	67.03%	71.42%	0.778	0.794	0.777	0.805
Zero-shot + OCR Tokens	74.94%	79.26%	0.831	0.834	0.830	0.850
Absolute Δ	+7.91%	+7.84%	+0.053	+0.040	+0.053	+0.045

Table 5: OCR token augmentation performance comparison

As shown, adding OCR tokens increases accuracy from 67.03% to 74.94%, an absolute gain of **7.91%**. Exact Match improves by **7.84%**, and content-preserving metrics such as F1, METEOR, and ROUGE-L also show consistent gains. BLEU exhibits a slight decrease, which is expected given the short length of most TextVQA answers and does not affect the overall conclusions. Overall, this ablation shows that OCR token augmentation alone can substantially strengthen zero-shot performance by providing explicit textual cues.

4.2.2 Ablation Study 2: LoRA Rank Selection

The second ablation investigates the effect of LoRA adapter rank under a fixed training budget of 5,000 samples. The results are listed in Table 6. As shown, LoRA rank $r = 8$ achieves the highest accuracy at **78.29%**, slightly outperforming both $r = 16$ (77.89%) and $r = 32$ (77.86%). Similar trends are observed for Exact Match and F1 score, while training time and GPU memory usage increase with higher ranks.

Rank	Trainable Parameters	% of Total	Accuracy	Exact Match	F1	METEOR	Training Time	GPU Memory
$r = 8$	22M	0.73%	78.29%	82.18%	0.851	0.853	∼2 hours	∼12GB
$r = 16$	45M	1.50%	77.89%	81.82%	0.847	0.849	∼2 hours	∼12GB
$r = 32$	90M	3.00%	77.86%	81.78%	0.846	0.848	∼2.4 hours	∼13GB

Table 6: Ablation study on LoRA rank using 5k training samples

These results are visualized in Figure 2, which plots accuracy as a function of LoRA rank and highlights the non-monotonic relationship between adapter capacity and performance in the limited-data setting.

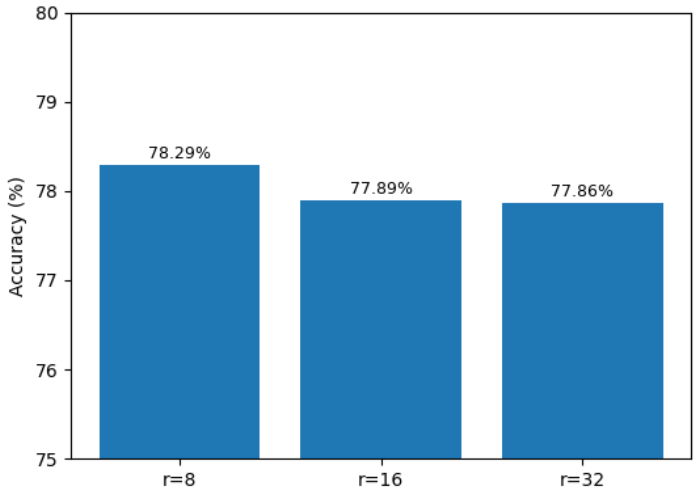

Figure 2: TextVQA accuracies from the ablation study on LoRA rank

4.2.3 Ablation Study 3: Training Data Scale

The final ablation examines how performance scales with the amount of training data when LoRA rank is fixed at $r = 16$. Results are summarized in Table 7, with the corresponding trend shown in Figure 3. As shown, fine-tuning on 5,000 samples (14.5% of the full dataset) already achieves an accuracy of 77.89%, capturing most of the improvement over the zero-shot baseline. Training on the **full dataset** further increases accuracy to **80.40%**, but the additional gain is relatively modest.

Training Samples	% of Full	Epochs	Accuracy	Exact Match	F1	Training Time	Δ vs Zero-shot
0 (Zero-shot)	0%	0	67.03%	71.42%	0.778	0	–
5,000	14.5%	1	77.89%	81.82%	0.847	∼2 hours	+10.86%
34,602	100%	3	80.40%	84.32%	0.867	∼18 hours	+13.37%
Δ (Full vs 5k)	+85.5%	+2	+2.51%	+2.50%	+0.020	+16 hours	+2.51%

Table 7: Ablation study on training data scale with LoRA rank fixed at $r = 16$

Figure 3 illustrates this pattern clearly, showing diminishing returns beyond the first 5,000 samples. This result highlights the data efficiency of LoRA fine-tuning for TextVQA and motivates the use of small, targeted training subsets in resource-constrained settings.

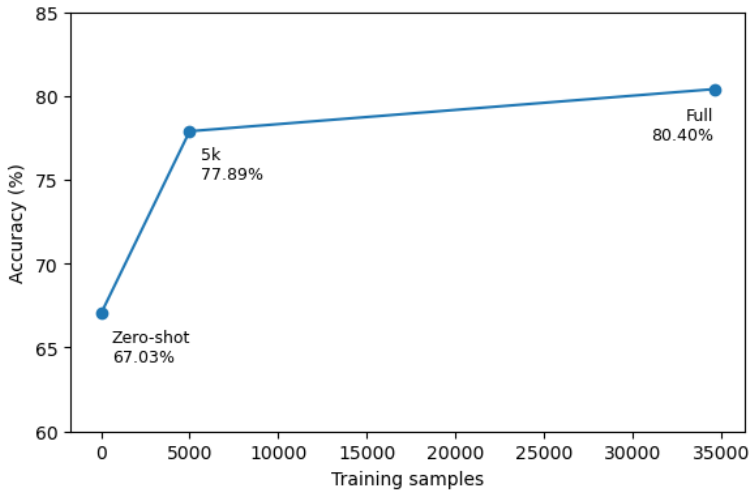

Figure 3: Effect of Training Data Scale ($r = 16$)

4.3 Error Analysis and Key Findings

While aggregate metrics provide a high-level view of performance, they do not fully capture how and why the model succeeds or fails. We therefore conduct an error analysis to better understand common failure modes and the effects of fine-tuning at a more granular level.

4.3.1 Error Type Distribution

We first analyze the distribution of error types for the zero-shot model. The results are summarized in Table 8, which categorizes errors into hallucinations, format errors, reasoning errors, OCR misses, and other miscellaneous cases. As shown, the largest error category is **format errors**, accounting for **44.4%** of all errors. These cases correspond to predictions where the model identifies the correct semantic content but outputs it in an incorrect format, such as providing a text string instead of a yes/no response. This suggests that many zero-shot failures stem from misalignment between question intent and output structure rather than from visual perception errors.

Error Type	Count	% of Total	% of Errors	Description
Correct	3,571	71.42%	–	Prediction matches ground truth
Hallucination	445	8.90%	31.1%	Invented non-existent information
Format Error	635	12.70%	44.4%	Correct content, wrong format
Reasoning Error	60	1.20%	4.2%	Saw information, selected wrong one
OCR Miss	142	2.84%	9.9%	Ignored available OCR tokens
Other	147	2.94%	10.3%	Miscellaneous errors
Total Errors	1,429	28.58%	100%	–

Table 8: Error type distribution for the zero-shot model on the TextVQA validation set.

Hallucinations and OCR misses together account for a smaller portion of errors, while reasoning errors represent a relatively minor fraction. Overall, the error distribution indicates that instruction-following and output formatting are dominant challenges in the zero-shot setting.

4.3.2 OCR-Grounding Bias and Qualitative Examples

A recurring failure pattern observed in the zero-shot model is a strong OCR-grounding bias, where predictions directly copy visible text without considering the semantic intent of the question. This behavior is illustrated through several representative examples shown in Figure 4.

Figure 4: Qualitative error examples comparing zero-shot and fine-tuned predictions

In the first example, the model answers "Denny's" to the question *"Is this Denny's?"*, copying the detected text instead of producing a yes/no response. In the second example, the model outputs partial on-screen text rather than identifying the intended action requested by the question. In the third example, the model responds with a brand name ("pepsi") instead of answering a yes/no question about whether the bottles are Pepsi.

In all three cases, the fine-tuned model produces the correct answer. These examples demonstrate that fine-tuning improves the model's ability to interpret question structure and generate appropriately formatted responses, rather than simply reproducing OCR tokens.

4.3.3 Category-wise Performance Breakdown

To further analyze performance differences, we examine accuracy across nine semantic question categories. The top five most improved categories are listed in Table 9, while the three least improved categories are shown in 10.

Rank	Category	Samples	Zero-shot	Fine-tuned	Absolute Δ	Relative Δ	Key Factor
1	Yes/No	266	32.33%	94.36%	+62.03%	+192%	Question type learning
2	Color	39	58.97%	83.76%	+24.79%	+42%	Region focus
3	Location	189	72.31%	90.48%	+18.17%	+25%	Spatial reasoning
4	Other	2,142	65.41%	78.93%	+13.52%	+21%	General improvement
5	Text	379	64.12%	74.93%	+10.81%	+17%	OCR integration

Table 9: Top-5 most improved semantic categories after fine-tuning.

As shown in Table 9, the **Yes/No** category exhibits the largest improvement, with an absolute accuracy increase of **62.03%**. This aligns with the qualitative observations in Section 3.3.2, as yes/no questions are particularly sensitive to OCR-grounding bias in the zero-shot setting. Substantial gains are also observed for **Color** and **Location** questions, which benefit from improved visual-text alignment.

Rank	Category	Samples	Zero-shot	Fine-tuned	Absolute Δ	Relative Δ	Challenge
1	Time	197	45.18%	47.88%	+2.70%	+6%	Small text, analog clocks
2	Brand	589	80.70%	83.53%	+2.83%	+4%	Already near ceiling
3	Number	905	75.80%	84.13%	+8.33%	+11%	Moderate challenge

Table 10: Bottom-3 least improved semantic categories after fine-tuning.

Table 10 shows that **Time**, **Brand**, and **Number** questions exhibit smaller improvements. In particular, Time questions remain challenging due to the need for geometric or temporal reasoning, such as interpreting analog clocks. Brand questions show limited gains because the zero-shot model already performs relatively well when copying OCR text is sufficient.

4.3.4 Regression Analysis

Although fine-tuning yields large overall improvements, it also introduces a small number of regressions. As reported in Table 4, only 10 cases transition from correct to incorrect after fine-tuning, compared to **679 cases** that improve.

Closer inspection of these regression cases reveals a common pattern involving rare or uncommon vocabulary, particularly proper nouns and brand names. Examples include cases where less common brand names are incorrectly normalized to more frequent tokens observed during training.

While these regressions represent a very small fraction of the dataset, they highlight a trade-off introduced by fine-tuning and motivate potential mitigation strategies discussed in the next section.

5 Discussions and Lessons Learned

This section discusses the key takeaways from our experiments and reflects on broader lessons learned from applying parameter-efficient fine-tuning to a vision–language model for TextVQA.

A central lesson from this project is that parameter-efficient fine-tuning can be highly effective for task adaptation. By updating only a small fraction of model parameters through LoRA, we were able to substantially improve TextVQA performance without modifying the full model. This confirms that much of the gap between zero-shot and task-specific performance can be closed through lightweight adaptation rather than large-scale retraining. For a course-scale setting, this approach proved especially practical, allowing experimentation with multiple configurations under limited computational resources.

The results highlight **OCR-grounding bias** as a key limitation of zero-shot vision–language models on TextVQA. In the absence of task-specific supervision, the model often defaults to copying visible text, even when the question requires reasoning about intent or output format. Fine-tuning helps mitigate this behavior by encouraging the model to better align question structure with appropriate answer types. This suggests that supervision on downstream tasks plays an important role not only in improving accuracy but also in shaping how models interpret and respond to different question forms.

Another important takeaway is the strong data efficiency observed in LoRA fine-tuning. A relatively small subset of the training data captures most of the achievable performance gains, while additional data provides diminishing returns. This pattern suggests that targeted supervision is sufficient to correct major failure modes in TextVQA. At the same time, the adapter rank ablation shows that larger capacity is not always better, particularly when data is limited. Smaller LoRA ranks can generalize more effectively and reduce overfitting risk, reinforcing the importance of matching model capacity to available data.

Finally, this project underscores the value of complementing quantitative metrics with qualitative analysis. While aggregate scores provide a useful summary of performance, qualitative examples reveal systematic behaviors, such as formatting errors and grounding failures, that are not fully captured by

accuracy alone. Together, these analyses provide a more complete understanding of model behavior and help identify directions for further improvement.

6 Limitations and Future Work

This section summarizes the main limitations of the current study and outlines directions for future work.

While the results demonstrate that LoRA fine-tuning can substantially improve TextVQA performance, the scope and design of this project impose several limitations that should be acknowledged when interpreting the findings:

- **Limited evaluation scope:** Experiments are conducted on a single dataset (TextVQA) and a single base model. While results are consistent across metrics, they may not generalize to other vision–language tasks or datasets with different reasoning requirements.
- **Restricted model and configuration exploration:** Due to computational constraints, this project focuses on a small set of LoRA configurations. Other adaptation methods, adapter placements, or full fine-tuning strategies are not explored.
- **Persistent difficulty in specific categories:** Certain question types, such as Time and Number questions, remain challenging even after fine-tuning. These cases often require geometric or numerical reasoning that is not explicitly addressed in the current setup.
- **Dependence on OCR quality:** The approach relies on externally provided OCR tokens. Errors or omissions in OCR extraction can directly affect downstream performance, limiting robustness in scenarios with noisy text detection.

Despite these limitations, the findings of this project suggest several promising directions for extending and improving the current approach in future work:

- **Evaluation across models and datasets:** Future work could test the approach on additional vision–language datasets and base models to assess generality and robustness.
- **Improved reasoning for challenging question types:** Incorporating targeted supervision or prompting strategies for time, counting, and numerical reasoning may help address remaining error cases.
- **Joint OCR and reasoning optimization:** Rather than treating OCR as a fixed input, future work could explore jointly optimizing OCR extraction and model reasoning, or incorporating OCR confidence information.
- **Comparison with alternative adaptation methods:** Exploring other parameter-efficient fine-tuning techniques, such as adapters or prefix tuning, could provide insight into trade-offs between efficiency and performance.

7 Conclusion

This project demonstrates that parameter-efficient LoRA fine-tuning can substantially improve TextVQA performance starting from a zero-shot vision–language model. Simple OCR token augmentation and lightweight adaptation correct common failure modes and yield strong gains under limited computational resources. Overall, the results show that effective task adaptation is possible within a course-scale setting without full model retraining. The complete implementation, including code, trained models, and generated figures, is available on GitHub.

References

[1] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 19730–19742, 2023.

[2] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiyang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[3] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[4] Qwen Team. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[5] Zhe Chen, Jiapeng Wu, Wenhai Wang, Wenhan Su, Guangyu Chen, Shilin Xing, Mingyu Zhong, Qi Zhang, Xizhou Zhu, Lewei Lu, Bo Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24185–24198, 2024.

[6] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8317–8326, 2019.

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115, 2023.

[8] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Yukun Lu, Zicheng Liu, and Liwei Wang. An empirical study of gpt-3 for few-shot knowledge-based vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3081–3089, 2022.

[9] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 7319–7328, 2021.

[10] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, 2002.

[11] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005.

[12] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, 2004.

[13] Lianmin Zheng, Wei-Lin Chiang, Yi Sheng, Siyuan Zhuang, Ziqi Wu, Yan Zhuang, Zihang Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623, 2023.