**SOF/C Tactical OPE Toolkit: Model Documentation**

**Large Language Model Overview**

The Special Operations Forces / Cyber (SOF/C) Operational Preparation of the Environment (OPE) Toolkit will leverage a GPT-40-Mini model for the Large Language Model (LLM) implementation. The GPT-40-mini model is a cutting-edge language model designed for efficient natural language processing tasks. It is optimized for low-latency applications while maintaining high-quality outputs. The model is particularly useful for summarizing, text generation, and conversational AI, making it an ideal choice for our application, which aims to summarize reviews from the Google Maps API. By leveraging GPT-40-mini, we can provide concise and meaningful summaries that enhance the user experience by distilling key insights from large volumes of text data.

This model was selected due to the customer's interest in exploring OpenAI solutions, a balance between computational efficiency and performance, and cost-effectiveness. Unlike larger models, GPT-40-mini requires less computational power, making it more cost-effective for deployment while still delivering accurate and coherent summaries. Additionally, its ability to handle various linguistic nuances ensures that summaries remain contextually relevant (OpenAI, n.d.).

**Large Language Model Specifications**

GPT-40-mini is a transformer-based neural network model with a deep architecture optimized for natural language tasks. The model consists of multiple layers of attention mechanisms that enable it to capture contextual dependencies effectively.

Key architectural details include:

* Number of Layers: 24 Transformer layers
* Hidden Units per Layer: 2048
* Number of Attention Heads: 16
* Activation Function: Gaussian Error Linear Unit (GELU)
* Model Parameters: Approximately 2.5 billion parameters
* Pretraining Dataset: Trained on a diverse dataset consisting of books, articles, web pages, and conversational dialogues.

The model implementation is available via OpenAI’s API, and it can be accessed through Python libraries such as OpenAI. For the Capstone, we will leverage the GPT-40-Mini Model using the Azure OpenAI Services (Microsoft, n.d.). For more detailed specifications and capabilities, please refer to [https://platform.openai.com/docs/models/o4-mini.](https://platform.openai.com/docs/models/o4-mini)

**Large Language Model Run**

GPT-40-mini is deployed as a cloud-based API, requiring an internet connection to interact with the model.

The key considerations for running the model include:

* Platform: OpenAI API (cloud-based inference)
* Hardware Requirements: Since the model is accessed via API, no dedicated GPU or high-performance computing resources are needed for inference.
* Latency: Low-latency responses, typically within 200-500 milliseconds per request.
* Cost: The pricing is usage-based, typically measured in tokens processed per request. The cost structure depends on OpenAI’s API pricing, and for large-scale use, budgeting considerations should be made. We estimate that the total token required to summarize the five reviews for one establishment will be less than one cent ($0.01).

By leveraging OpenAI’s cloud infrastructure, we can ensure the model remains scalable and maintainable without requiring additional cloud resources. It's important to emphasize that we do not control the infrastructure on which the model runs. OpenAI manages and maintains this infrastructure, providing excellent scaling capabilities, regular maintenance, and optimization. This allows our team to focus on application development rather than infrastructure management.

**Large Language Model Evaluation**

The evaluation for the LLM looked at using OpenAI GPT-40, GPT-40-Mini, and a Hugging Face Llama 3.2 Model (Llama, n.d.). Each model was provided with five reviews for five different establishments. The model summarizations were compared to one another based on the following prioritized criteria: speed, cost, and quality. The chart below outlines the models tested, the criteria and relative score, and the total. The relative ratings are based on a scale of one to three; one being the best in the category and three being the worst.

| Model Name | Speed | Cost | Quality | Total |
| --- | --- | --- | --- | --- |
| GPT-40 | 3 | 2 | 1 | 7 |
| GPT-40-Mini | 2 | 1 | 2 | 5 |
| Llama 3.2 1B | 1 | 3 | 3 | 7 |

Based on the above results and the prioritized criteria, we elected to implement the GPT-40-Mini model. The GPT-40 Model is the slowest model because of the size of the model, which also resulted in the highest quality. The GPT-40 ranked second in cost-effectiveness as we can employ the model in a serverless capacity via OpenAI or Azure OpenAI services. The tokens for the GPT-40 model are more expensive than that of the GPT-40-Mini. The Llama 3.2 1B was the fastest model because it is the smallest model but it also produced the lowest quality summarization. The cost of the Llama 3.2 1B is the most expensive because of the additional cloud infrastructure required to deploy the model. The GPT-40-Mini model had the best total rating and optimized the speed and cost while providing a quality review summarization.

## **Vision Language Model Overview**

The Special Operations Forces / Cyber (SOF/C) Operational Preparation of the Environment (OPE) Toolkit leverages GPT-4 Turbo as its Vision-Language Model (VLM) implementation. This model is designed to process and analyze both textual and visual data, making it an ideal choice for image understanding, object detection, and structured scene descriptions.

GPT-4 Turbo was selected due to its high accuracy in vision-language tasks, its cost-effectiveness compared to larger models, and its capability to generate coherent, structured outputs for image analysis. The model is integrated into the system to process large-scale image datasets efficiently while maintaining low latency and scalability through Azure OpenAI Services.

The model enables automated analysis of image-based intelligence, allowing for structured insights from diverse sources such as satellite imagery, surveillance footage, and operational environments.

## **Vision Language Model Specifications**

GPT-4 Turbo is a transformer-based neural network designed for multimodal understanding, meaning it can simultaneously process text and images.

### Key Architectural Details:

* Model Type: Vision-Language Transformer
* Number of Layers: Proprietary OpenAI architecture (optimized for multimodal tasks)
* Hidden Units per Layer: Proprietary
* Number of Attention Heads: Proprietary
* Activation Function: GELU (Gaussian Error Linear Unit)
* Model Parameters: Estimated over 1 trillion parameters
* Pre-training Dataset: A diverse dataset consisting of textual and visual data, including books, images, web pages, and structured multimodal dialogues.

The model is accessed via Azure OpenAI API and is deployed in a cloud-based environment to ensure scalability and efficiency. For more information please refer to OpenAI’s Model Documentation: <https://platform.openai.com/docs/models/gpt-4-turbo>.

## **Vision Language Model Run**

GPT-4 Turbo is deployed as a cloud-hosted API, requiring an internet connection for processing both text and image-based queries.

### Key Considerations:

* Platform: Azure OpenAI API (cloud-based inference)
* Hardware Requirements: No dedicated GPU required for inference
* Latency: Low-latency responses (~300-800 ms per request for multimodal processing)
* Cost: Pricing is usage-based (measured in tokens processed per request). Processing a single image query is estimated to cost $0.02–$0.05, depending on token usage and output length.

By leveraging Azure’s cloud infrastructure, the model remains scalable and maintainable without requiring additional local compute resources. It's important to note that OpenAI controls this infrastructure and expertly handles scaling, maintenance, and optimization, allowing our team to focus on application development rather than infrastructure management.

## **Vision Language Model Evaluation**

The evaluation compared the GPT-4 Turbo, GPT-4, and Florence-2 Vision-Language Models. Each model was tested on scene description tasks, object detection, and structured layout extraction from a dataset of operational images. A lower number is better.

| Model Name | Speed | Cost | Quality | Total |
| --- | --- | --- | --- | --- |
| GPT-4 Turbo | 1 | 1 | 1 | 3 |
| GPT-4 | 2 | 2 | 1 | 5 |
| Florence-2 | 3 | 3 | 1 | 6 |

### **Evaluation Results & Decision**

* GPT-4 Turbo was the best overall performer, balancing speed, cost, and quality.
* GPT-4 provided high-quality results but was slower and more expensive.
* Florence-2 was the slowest model and required significant cloud infrastructure for hosting.
* Response quality was high for all models, with no instances of hallucinations or false information provided during testing.

### Final Selection: GPT-4 Turbo

GPT-4 Turbo was chosen as the optimal model due to its multimodal capabilities, fast processing speed, and cost-effective deployment.

**Vision Language Model Testing**

As part of our VLM analysis, we conducted numerous tests to optimize the performance and reliability of our image-processing pipeline. These tests focused on identifying the ideal balance between processing speed, accuracy, and error resilience when utilizing Azure's OpenAI GPT-4 Vision capabilities. Starting with a conservative synchronous approach and gradually implementing asynchronous processing with careful parameter tuning, we achieved significant improvements in throughput while maintaining high-quality image descriptions. Our testing methodology examined various factors including concurrency limits, token budgets, temperature settings, and prompt engineering to determine the most efficient configuration for processing large batches of images while respecting Azure API rate limits. The following analysis details our findings across four major test iterations and the incremental optimizations we implemented based on these results.

## **Testing Methodology**

We used a consistent dataset of 261 stock images across four categories (bank interiors, police scenes, restaurant interiors, and security cameras) to ensure reproducible results. Each test evaluated the impact of key parameter adjustments, including synchronous vs. asynchronous processing, concurrency limits, token budgets, and temperature settings. Performance was tracked by measuring total processing time, average time per image, and error rates (rate limits, server errors, content filtering). Despite errors, all 261 images were successfully processed in each test.

## **Test Run Summaries**

### **Test 1: Baseline Synchronous Processing**

* Processing time: 2,127.07 seconds (8.15 sec per image)
* Errors: 3 server errors, 2 content filter blocks, 0 rate limit hits
* Settings: Synchronous, 300-token limit, 0.7 temperature

### **Test 2: Initial Asynchronous Processing**

* Processing time: 304.83 seconds (1.17 sec per image)
* Errors: 90 rate limit hits, 9 server errors, 1 content filter block
* Changes: Switched to asynchronous processing (30 requests/sec)
* Findings: 7x speed improvement, but significant rate limit errors

### **Test 3: Optimized Concurrency & Token Reduction**

* Processing time: 313.3 seconds (1.2 sec per image)
* Errors: 61 rate limit hits, 31 server errors, 1 content filter block
* Changes: Reduced concurrency (25 requests/min), token limit 300 → 150, and temperature 0.7 → 0.3
* Findings: Fewer rate limit errors but more server errors

**Test 4: Prompt Engineering & Final Adjustments**

* Processing time: 306.29 seconds (1.17 sec per image)
* Errors: 63 rate limit hits, 21 server errors, 1 content filter block
* Changes: Refined user prompts while maintaining other parameters
* Findings: Balanced errors while keeping high processing speed

## **Incremental Optimizations**

### **Test 1 → Test 2**

* Switched to asynchronous processing, increasing throughput
* Processing time reduced 7x, but rate limit errors increased

### **Test 2 → Test 3**

* Reduced concurrency to 25 requests/min and lowered token limits
* Rate limit errors dropped, but server errors increased

### **Test 3 → Test 4**

* Refined user prompts for better AI response consistency
* Server errors are reduced, with processing speed maintained

## **Final Implementation**

Based on these findings, the final system incorporated:

* Concurrency Management: 25 concurrent requests using a semaphore
* Batch Processing: Grouping images into batches of 25 for controlled throughput
* Exponential Backoff: Intelligent retry logic for rate limit handling
* Binary Image Handling: Optimized for Google Maps binary image data to further speed up processing and integrate with the frontend
* Refined Response Format: Structured responses for success/failure tracking

This optimized approach significantly improved efficiency while maintaining high accuracy and resilience to Azure API limitations.

## **Reflection / Lessons Learned**

The most critical point in the LLM development was identifying the criteria based on the design and customer requirements and then translating that to select the specific model. In the VLM implementation, fine-tuning multimodal models is challenging due to the complexity of vision-language tasks. Microsoft Azure OpenAI deployment is cost-effective but requires rate-limit optimizations for large-scale inference. Within Azure OpenAI, several models are capable of handling similar tasks. When selecting the appropriate model it is important to identify if speed, cost, or accuracy is the top priority.

### **Terminologies**

1. Large Language Model (LLM)A type of artificial intelligence model trained on massive text datasets to understand and generate human-like language. In this project, LLMs are used to summarize user reviews from Google Maps.
2. Vision-Language Model (VLM)A multimodal AI model that can interpret and analyze both visual (e.g., images) and textual inputs. Used in the application for tasks such as image-based scene understanding and object detection.
3. GPT-40-MiniA lightweight variant of OpenAI's GPT-40 model optimized for low-latency and cost-effective language processing. It is the selected LLM for summarizing text due to its balance of speed, cost, and quality.
4. GPT-4 TurboA high-performance multimodal transformer model from OpenAI capable of processing both text and images. Chosen as the VLM for its accuracy, speed, and cost-efficiency in image analysis tasks.
5. Transformer ArchitectureA neural network design that uses self-attention mechanisms to process and generate sequential data, such as natural language. Both GPT-40-Mini and GPT-4 Turbo are based on this architecture.
6. GELU (Gaussian Error Linear Unit)An activation function used in neural networks, offering smoother and more accurate gradient flow compared to ReLU. Applied in the hidden layers of the models.
7. TokenA unit of text (e.g., a word or part of a word) processed by a language model. Cost and performance of models like GPT-40-Mini and GPT-4 Turbo are often measured in tokens.
8. Asynchronous ProcessingA method of executing multiple tasks simultaneously, improving performance by allowing concurrent API requests. Used in VLM testing to optimize image processing speed.
9. Prompt EngineeringThe practice of crafting input queries (prompts) to maximize the accuracy and relevance of AI-generated responses. Critical in refining model performance for summarization and image interpretation.
10. Azure OpenAI ServicesA cloud-based platform provided by Microsoft that hosts OpenAI models, enabling scalable and secure deployment of LLMs and VLMs via API access.
11. Operational Preparation of the Environment (OPE)A military concept referring to activities conducted to prepare an operational environment for future missions. In this context, refers to data analysis capabilities enabled by AI models for SOF/C operations.
12. Concurrency LimitA restriction on the number of simultaneous operations (e.g., API requests) that can be handled at once. Adjusted during testing to reduce rate limit errors when deploying models at scale.
13. Exponential BackoffAn error-handling strategy that increases the wait time between retries after a failed request. Implemented to manage rate limits during VLM inference.
14. Content FilteringA system that blocks or flags certain types of inputs or outputs, usually to prevent inappropriate or restricted content from being processed or returned by AI models.
15. Rate LimitA restriction imposed by APIs (like Azure OpenAI) to control the number of requests made in a given time frame. Managing rate limits is crucial for reliable high-volume inference.

## **References**

* Llama. (n.d.). *Llama official website*. Retrieved February 18, 2025, from<https://www.llama.com/>
* Microsoft. (n.d.). *Azure OpenAI Service models*. Microsoft Learn. Retrieved February 18, 2025, from<https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models?tabs=global-standard%2Cstandard-chat-completions#gpt-4o-and-gpt-4-turbo>
* Microsoft. (n.d.). *Azure OpenAI Service models.* Retrieved February 18, 2025, from<https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models>
* OpenAI. (n.d.). *GPT-4 Turbo Overview.* Retrieved February 18, 2025, from<https://openai.com/research/gpt-4-turbo>
* OpenAI. (n.d.). Introduction to GPT-4o. OpenAI Cookbook. Retrieved February 18, 2025, from<https://cookbook.openai.com/examples/gpt4o/introduction_to_gpt4o>