**Digitization of Aircraft Engine Logbook using Large Language Model**

**SS ZG628T: Dissertation**

**by**

**ANUBHAV KUMAR SAURAV**

**2023MT12232**

**Dissertation work carried out at**

**Honeywell Technology Solutions, Bangalore**

A logo of a science and technology company

AI-generated content may be incorrect.

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

**March 2025**

**Digitization of Aircraft Engine Logbook using Large Language Model**

**SS ZG628T: Dissertation**

**by**

**ANUBHAV KUMAR SAURAV**

**2023MT12232**

**Dissertation work carried out at**

**Honeywell Technology Solutions, Bangalore**

**Submitted in partial fulfillment of M.Tech. Software Systems**

**Under the Supervision of**

**Renju C Panicker,**

**Honeywell Technology Solutions, Bangalore**

**A logo of a science and technology company

AI-generated content may be incorrect.**

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

**March 2025**

**ABSTRACT**

The lifecycle of aircraft engines is approx. 30-40 years. During their lifecycle, it undergoes for maintenance, repair and overhaul (MRO) operations at different intervals of flight hours depending on the engine type. The service logs of aircraft engines are currently maintained and recorded manually in paper-based form which is quite challenging to maintain, preserve and look for any particular data when needed. Digitizing service log records of aircraft engines reduces administrative workload and enhances collaboration and communication among stakeholders. By automating data entry and enabling real-time access to digital logs, critical data can be accessed from anywhere, fostering seamless collaboration between maintenance crews, flight operations, and regulatory authorities. This will help in strengthening safety, data preservation, compliance efforts and thus, improving overall operational efficiency.

The digitization of aircraft engine service logs presents a complex challenge due to the variability in handwritten, printed, and structured data formats. This dissertation focuses on an AI-driven framework that leverages Computer Vision (CV), Natural Language Processing (NLP), and Machine Learning (ML) to automate the extraction, classification, and analysis of aircraft engine service logs.

The proposed system integrates OCR-free multimodal transformer-based framework for visual document understanding (VDU) to interpret both textual content and spatial features of the document image, perform information extraction, NLP models for contextual understanding and key-value mapping, text prompt and visual question answering (VQA) features for querying on the extracted information. By structuring unstructured log data into a queryable format, this approach enhances data accessibility, error reduction, and predictive maintenance capabilities. The framework is further evaluated on real-world aviation datasets of engine logbook records, demonstrating its ability to improve information retrieval, maintenance forecasting, and compliance tracking. The dissertation highlights the potential of AI in transforming aviation maintenance by enabling automated, scalable, and intelligent service log management, ultimately paving the way for more efficient and proactive aircraft maintenance strategies.

Contents

[1. INTRODUCTION: LARGE LANGUAGE MODELS 5](#_Toc193753332)

[1.1 Vision Language Models 5](#_Toc193753333)

[1.2 State of the Art Large Language Models 6](#_Toc193753334)

[2. VISUAL DOCUMENT UNDERSTANDING 9](#_Toc193753335)

[2.1 Multimodal Transformer Architecture 10](#_Toc193753336)

[2.2 OCR free Document Understanding with DONUT model 10](#_Toc193753337)

[3. Dataset Preparation 11](#_Toc193753338)

[4. Pre-processing tasks 12](#_Toc193753339)

[5. ABBREVIATIONS 12](#_Toc193753340)

[6. REFERENCES 13](#_Toc193753341)

# INTRODUCTION: LARGE LANGUAGE MODELS

Large language models have emerged as a crucial technology with profound importance across various domains. They are huge and powerful machine learning models trained on massive datasets to process and generate human language and recognize patterns. Their advanced natural language processing capabilities make them invaluable tools in almost any industry. They have become indispensable assets for automating language-related tasks, freeing human resources, and enhancing organizational efficiency. Large language models can generate creative and contextually relevant content, assist decision-making processes, and enhance search engines, chatbots, virtual assistants, and recommendation systems. The use cases for LLM’s are streamlining operations, improving customer experience, content writing, language translation and drive innovation by automating tasks, personalizing interactions, and enabling data-driven insights, to name a few.

## Vision Language Models

Vision-Language Models (VLMs) are a subset of Large Language Models (LLMs) that can process and understand both text and images. These models combine natural language processing (NLP) with computer vision (CV) to perform tasks such as image captioning, visual question answering (VQA), and multimodal reasoning. They bridge the gap between visual and linguistic intelligence. They combine a large language model (LLM), which excels at understanding and generating text, with a vision encoder, which allows the LLM to "see" and process images or videos.

Vision language models are typically made up of 2 key components:

    ● **Language encoder:** A language encoder captures the semantic meaning and contextual associations between words and phrases and turns them into text embeddings for AI models to process.

    ● **Vision encoder:** A vision encoder extracts vital visual properties such as colors, shapes and textures from an image or video input and converts them into vector embeddings that machine learning models can process.

Earlier versions of VLMs used deep learning algorithms such as convolutional neural networks for feature extraction. More modern vision language models employ a vision transformer (ViT), which applies elements of a transformer-based language model. A vision transformer processes an image into patches and treats them as sequences, akin to tokens in a language transformer. The vision transformer then implements self-attention across these patches to create a transformer-based representation of the input image.

VLMs can be trained and instructed to perform a wide variety of tasks through natural language processing:

1. Visual questions-answering
2. Image and video summarization
3. Parsing text and handwritten documents

For the document images which are in the form of engine service logbooks, the application of vision language models will be used in the project to achieve the desired result of information extraction.

## State of the Art Large Language Models

Multimodal Large Language Models (MLLMs) are AI systems that process and generate text, images, audio, video, and other data types. These models have advanced image understanding, reasoning, and multimodal generation capabilities, making them useful for applications like chatbots, AI-powered search, robotics, content creation, and medical diagnostics.

These models have paved the way for various applications such as document parsing, image captioning, visual question answering, and multimodal content creation. Below are the examples of some of the state-of-the-art multimodal models under LLMs:

**1. OpenAI's Multimodal Models**

* **GPT-4V (GPT-4 Vision**)- It is a multimodal variant of OpenAI’s GPT-4 that can process both text and images. It extends the capabilities of standard GPT-4 by integrating visual understanding, allowing it to analyze and interpret images alongside text.
* **CLIP (Contrastive Language–Image Pretraining)** –This model can understand and correlate images and text, enabling tasks like image-text retrieval and zero-shot image classification.
* **DALL·E** – A generative model that creates images from textual descriptions, combining language and visual processing to produce novel and imaginative visuals.

**2. Google DeepMind's Models**

* **PaLI (Pathways Language and Image model)** – A large multimodal model trained on both image and text tasks. It can process both images and text for reasoning, captioning, and answering questions. It supports multiple languages, making it useful for cross-lingual vision-language tasks.
* **Flamingo** – A vision-language model that adapts to multimodal tasks with minimal training data. It is specifically designed for **few-shot learning**, meaning it can perform tasks with very few examples, making it highly adaptable across various vision-language applications.
* **Gemini** – A multimodal model that combines text, images, and other modalities. Gemini natively integrates text, images, audio, and video into a single model and can understand and generate content across multiple formats.

**3. Meta (Facebook) Models**

* **SEER (Self-supervised Emergent Representation)** – Learns visual concepts without labeled data. It is designed to learn from billions of unlabeled images, making it highly adaptable for various computer vision tasks without requiring labeled datasets.
* **LLaVa (Large Language and Vision Assistant)-** It combines Meta’s LLaMA language model with CLIP-ViT for vision-language tasks. Used for image captioning, visual Q&A, and multimodal reasoning.
* **ImageBind** – First AI model that binds six modalities, including text, audio, , images, video, thermal and motion sensors into a single embedding space. Usecase include Audio-visual content search, Multimodal scene understanding and Cross-modal retrieval.

**4. Microsoft Models**

* **Kosmos-2** – A multimodal model that integrates vision, language, and reasoning allowing for richer visual reasoning. It can locate objects in images based on textual descriptions.
* **BLIP (Bootstrapped Language-Image Pretraining)** – A model for captioning, retrieval, and visual question answering.

# VISUAL DOCUMENT UNDERSTANDING

Visual Document Understanding (VDU) is a method of extracting useful information from document images such as text, tables, spatial features of the document. The document images could be invoices, receipts, service logbook pages, scanned PDF etc which should be provided as an input to the multimodal transformer model. VDU combines artificial intelligence techniques or subfields such as computer vision (CV), natural language processing (NLP), and deep learning (DL) to understand document layouts, templates and extracts useful information from the semi-structured documents. Apart from these things, VDU is capable of performing various document related tasks such as entity grouping, sequence labeling, document classification and visual question answering.

Traditional approach to VDU was based on Optical Character Recognition based processing. It has two step process: (a)Text extraction with off the shelf OCR engines (b) Downstream processing of information. The OCR method primarily extracts text from images by converting the document’s visual text into machine-readable text. The extracted text is then processed using models like BERT or Layout LM, depending on the task to perform. This traditional OCR based method comes with several limitations like high computational cost, limited flexibility of OCR with different types of fonts, languages and document layout, error propagation in one step affects all subsequent processing steps. The problem becomes more severe in languages with complex character sets, such as Korean or Chinese, where the quality of OCR is relatively low.

The present state of the art VDU methods goes beyond OCR and leverages artificial intelligence techniques to understand the relationships between text elements, tables, and other visual cues within a document without OCR. OCR free visual document understanding models addresses the problems induced by OCR dependency. The current state of the art VDU models are based on multimodal transformer architecture which consists of encoder-decoder pipeline that combines computer vision (CV) and natural language processing (NLP) methods to analyse document and extract the information from document image.

## Multimodal Transformer Architecture

A multimodal transformer architecture is a deep learning framework designed to process and integrate multiple types of input data modalities such as text, images, audio, and video. These architectures are extension of the traditional Transformer model like BERT, GPT to handle multimodal information in a unified manner and enable cross-modal understanding and reasoning. They consist of modality specific encoders where each input modality is processed separately using specialized encoders:

* **Text Encoder:** Uses Transformer-based models like BERT or GPT to convert text into embeddings.
* **Image Encoder:** Uses Convolutional Neural Networks (CNNs) or Vision Transformers (ViT) to extract visual features and spatial understanding.
* **Audio Encoder:** Uses spectrogram-based CNNs or wav2vec-like architectures for speech processing.
* **Video Encoder:** Uses 3D-CNNs or TimeSformer to capture spatial-temporal features.

Each modality is converted into a sequence of tokens and then into embeddings that maps each token into a dense vector representation by the encoder module. To integrate information from different modalities, architecture employs various fusion techniques such as early fusion, late fusion, self-attention fusion etc. The decoder module generates a sequence of tokens from the encoded embeddings that is converted into the target output in a structured form.

## OCR free Document Understanding with DONUT model

DONUT, which stands for Document Understanding Transformer is a vision-language model designed for visual document understanding (VDU) tasks. It combines a Vision Transformer (ViT) backbone for image encoding with an autoregressive decoder that generates structured text (e.g., JSON) as output. In other words, it splits the input image into patches using a Swin Transformer and encodes the image into token vectors which it can then decodeinto an output sequence in the form of a data structure using the BART decoder model, publicly pretrained on multilingual datasets which can then be further parsed into JSON format. Any prompts fed into the model at inference time can also be decoded as well in the same architecture. The donut model can perform classification of documents, understanding the document layout, parsing and visual question answering.

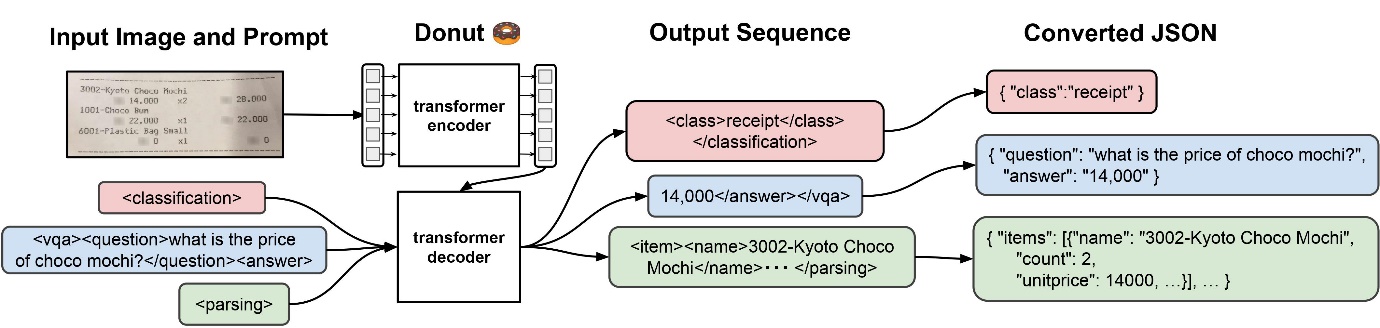


Figure 1: Transformer architecture of DONUT model

The donut model works in two phases. The first is Pretraining Phase where the model learns from document images paired with their corresponding text annotations. It predicts the next token in a sequence based on the document image and prior context, minimizing cross-entropy loss. The next phase is known as Fine tuning phase where the model learns to understand the documents on a new task.

# Dataset Preparation

We have created a dataset of engine service logbooks comprising of different invoices, templates and data in order to train the transformer-based donut model on our custom dataset. The dataset directory will consist of two folders. The first folder contains the images of the service logbooks and the other folder contains the corresponding text annotations of the corresponding images in JSON format.

The next step in dataset preparation is to process the images and their corresponding text annotations to create a ‘metadata.json’ file that contains information about the dataset images including their text annotations.

The *“metadata.json”*file will consist of information like this as given in example below.

{"file\_name": "0001.jpg", "text": "This is a ship crossing the ocean"}

{"file\_name": "0002.jpg", "text": "A german shepherd dog playing with a ball"}

# Pre-processing tasks

After loading the dataset, there are further pre-processing tasks that need to be carried out before it can be passed as an input to train and fine tune the model.

1. **Text Pre-processing:**

* **Tokenization:** Models cannot process raw text, so we need to convert the text into numbers. The tokenization process divides the text into individual words called tokens. Tokens are finally converted to numbers. The Donut model typically uses a BART-style tokenizer for the text decoder.
* **Add Special tokens:** Using special tokens like <s> (start), </s> (end), and <sep> (separator) to format the JSON output properly.

1. **Image Pre-processing:**

* **Resize and Normalize:** Resize the images to a consistent resolution (e.g., 256x256 or 480x480) depending on the Donut model variant to ensure uniformity.
* **Convert to Tensor:** Convert images to PyTorch tensors for GPU processing. Ensure the shape is in **CHW format** (Channels, Height, Width).

1. **Dataset formatting and alignment:**

Donut requires paired image-text data, where each image corresponds to a structured label. We have ensured each image has a corresponding JSON-structured label and stored the pairs in a dictionary-like format:

Example:

{

"image": "path/to/image.jpg",

"ground\_truth": '{"invoice\_number": "12345", "date": "2025-03-24", "total": "$1,234.56"}'

}

# Training and fine tuning

After we have loaded our dataset and performed the pre-processing tasks on it, we can start training our model. In-order to start the training, we will use a pre-trained donut base model with the VisionEncoderDecoderModel class from Hugging face library to further train it on our specific dataset. This pre-trained model contains pre-trained weights which means it has already been trained on a large dataset and can be used as a starting point for fine-tuning on specific dataset. The dataset is split into training (e.g. 85%) and test data (e.g. 15%). The hyper parameters like learning rate, number of epochs, weight decay, batch size, output directory, max steps etc. need to be defined as arguments for the training.

After the training is completed, we will get a processor and a fine-tuned model as an output that need to be saved on git repo or local disk. This saved model will be used for prediction tasks and evaluation on new test data.

# Technical Considerations

|  |  |  |
| --- | --- | --- |
| Software Requirements | Programming Languages | Python (version 3.8 or higher) |
|  | Software Libraries | PyTorch (version ≥ 1.12): A deep learning framework for building and training models.  Hugging Face Transformers (version ≥ 4.30): A library that provides pre-trained models and tools for various NLP tasks, including Donut.  Donut Repository: Clone the official Donut repository from GitHub |
| Hardware Requirements | GPU | A GPU with at least 16 GB of memory for efficient training, especially with larger models. |
|  | Cloud Platforms | Google Colab Pro or Amazon SageMaker for scalable and powerful training environment |
|  | RAM, Hard Disk | 16-32 GB RAM and 256 GB hard disk. |

# ABBREVIATIONS

|  |  |
| --- | --- |
| VDU | Visual Document Understanding |
| ViT | Vision Transformer |
| NLP | Natural Language Processing |
| CV | Computer Vision |
| ML | Machine Learning |
| OCR | Optical Character Recognition |
| DONUT | Document understanding Transformer |
| LLM | Large Language Model |
| DL | Deep Learning |