**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)**

**April 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)**

**April 2025**

**CERTIFICATE**

This is to certify that the Dissertation entitled “Digitization of Aircraft Engine Logbook using Large Language Model” and submitted by ANUBHAV KUMAR SAURAV having ID-No. 2023MT12232 for the partial fulfillment of the requirements of M.Tech - Software Systems degree of BITS, embodies the bonafide work done by him under my supervision.

A close up of a writing

AI-generated content may be incorrect.

Signature of the Supervisor

Place : Bangalore Renju Chandrasekhara Panicker,

Principal System Engineer,

Date : 20.04.2025 Honeywell Technology Solutions, Bangalore

**ABSTRACT**

The lifecycle of aircraft engines is approx. 20-30 years depending on the engine types – turboprop, turbojet, turboshaft, turboprop etc. 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 different locations, allowing better collaboration among maintenance crews and flight operations. 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 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.

![A close up of a note

AI-generated content may be incorrect.](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RwaRXhpZgAATU0AKgAAAAgABgALAAIAAAAmAAAIYgESAAMAAAABAAEAAAExAAIAAAAmAAAIiAEyAAIAAAAUAAAIrodpAAQAAAABAAAIwuocAAcAAAgMAAAAVgAAEV4c6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFdpbmRvd3MgUGhvdG8gRWRpdG9yIDEwLjAuMTAwMTEuMTYzODQAV2luZG93cyBQaG90byBFZGl0b3IgMTAuMC4xMDAxMS4xNjM4NAAyMDIzOjA3OjE3IDE0OjUzOjM5AAAIkAMAAgAAABQAABE0kAQAAgAAABQAABFIkpEAAgAAAAMwNwAAkpIAAgAAAAMwNwAAoAEAAwAAAAEAAQAAoAIABAAAAAEAAASooAMABAAAAAEAAAIV6hwABwAACAwAAAkoAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAyMzowNzoxMiAxMDoxODozNQAyMDIzOjA3OjEyIDEwOjE4OjM1AAAAAAYBAwADAAAAAQAGAAABGgAFAAAAAQAAEawBGwAFAAAAAQAAEbQBKAADAAAAAQACAAACAQAEAAAAAQAAEbwCAgAEAAAAAQAAClUAAAAAAAAAYAAAAAEAAABgAAAAAf/Y/9sAQwAIBgYHBgUIBwcHCQkICgwUDQwLCwwZEhMPFB0aHx4dGhwcICQuJyAiLCMcHCg3KSwwMTQ0NB8nOT04MjwuMzQy/9sAQwEJCQkMCwwYDQ0YMiEcITIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIy/8AAEQgAUACzAwEhAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A9kA3H/CneWelZjDyyKPK9qADyuenWjyyKADy/ajy8UAHl+1AiwOlAB5dNMZ9KAAxnpijyyaAF8s+/WjyiO2KADyqPK9qAF8s0eWaADyz6U0qQaQBuPaigB0fWp6YBRQAUnagBe9FABRQAlFACUd6ACigAozQAtAoABUMnU/WkAyii4D4+oqcdaYB3ooASj1oAKO1ABRQAUUAJSUAJ3FL+NACZpcjFJAAPalzRqPpcX1qGT71Ahg6df1NFADoutWKYBmkzSAaTRnmmAuaDQAZpNwoACaTNACFu3rRn3pAISeuO9OpgZNzqc8Piuw0wLF9nubSeZmIO/cjRgY5xjDHPHpV2/v4NMtPtVyWWPekY2qWJZ2CqAB6swrV0ruNuwr7mFd6+2qXWmafoF1Gft0JupLxAGMFuOAwVv4mb5RkcHORxW/aWzWyupup5wxBXzmBK8YODgcHrz6nHGALqxVOCj1ZMXdtkwmiM7QCRDKihmQMNyg9CR2Bwfypkn3q57WLIjnNFAEsXWp6AE7daQsFBZmCqBkk9AKEruyBvQ4q28YXv9sae2oW8UWj6xI0OnyqCGRx90SEn/lp24HOBznjsgfSt8TSVOVoakU5c0eZjxQa5yxufSsvStbh1afU44Y5EOn3rWcm/HzMqqxIx2wwqowck2O/Qq+M9TuNH8GarqNtN5M8EO5JcA7SWAzggjv6VtDO3nAPcD1q3FKmn1JTblZkH220F21obqH7SF3mLeNwXIGcfUgfiPWoIdZtZ9Zn09Li1ZooyWCzguGX76suONoZDnJ+9270sPPt5i5o7HP6HZ/8JLqV14g1SNZ7dLpk0qBzviijjbaJlXoXZgSG5IHQ4xXZDoc0YiylZdAjtfued/E261m01TRp/Dy7tSis79/uhtseyMM2D3BwR/tY7ZB1rK00G20mz1W01d4bWRo5nmvrtmS4Vc53rIcB+pyMEMq54XFdML+zi0T1tIz/AApDp2la/rNxpbz6ja3rqbVbOJ3jiTc7sPMIEa/PIeAx+7613sbOybnjMZ9CwJ/Tj9azxTUpcz3sVHYa1natdi7NtCbkDAmKAuB0wD1A9qSTqa5HLmsUMxRQBJF1qc0ANJ5rC8Xyyr4dmtYGUT3zpZRljgAyHafyXcfwrXDr94rky+FmT8Q9Kj/4V1cCzAik0lY7q0K/8sjDjBH/AAHNdTa3KXVrDcpjZNGsi/RgCP51dRuUL+bBLXQsKcjIHFQtdwfbFszPH9qaMyiEsNxQEAtj0yQPxrBRbKINP1GO+nvkWOSIWV0bZ2kAAYhEfcv+z8+PqDXB+ENaL6DqNxpKxy6hqF5c6mROG2xRMxCvIeMAhOBkEgZHArso0U1JT7oylLqXfGd3J4h+Ct7qEEbRNeWMNzsB5Ubkcj3GAfwq69/4i1e0utY8PNEsKF47S0ukG28VcgTK2QV3HlcnaQBnAO6nTUYxvIb5noYWjeEb7VtKtra9sbnSZbS1Gbm4ZZJLi8ZtzyEBjuQMoI3EZ3dscbOqfDmy1HT7aNL2WDUEd2uL5UG+5Ei7Zg65wdwA9cYHUZB0njeVKLWxEaUbtnX29tDaW8dvbxrHDEgSONRgKoGAKlArzZNuTkzdaK3Yz9VsLy/aCK3v5bKD5vPe3CiVhxhVY52g85IGemCKx9G+HugaBqDXWn2oVWXHkyqsoDf31ZgXBxxgNj2rqjXUYcqIceZ3Z1OP0pf84rlcm3dlJB61BL976Ur3GRlsHG39aKYEsfWpjQAnOcD8q898WeJ7X/hNvC+m21wl0sdzPcXMdqhnkVkjZVG1Twdxbg9Mc8A10YaEpTv2Im9LGpr1xrmqeHNTht9Fht4JbSVGa/ugG2lDnEaBucdiwrG8O23ie68LeGYrbVUhs5dOGZLe1AkQrGvlKS+4c4OW4zgDArpjGlBWepDlN7DLB9T8a+E7jxFY3FzbX32df7MSGchRLGoLkg8HdLuQ7uNqjsatgeK7nW7PxRb6QCJY2sxplzIIXt4WVWMjknr5q5IGW2heM5w3Onqu2guWW5qaN4a1ewnu4rrV4p9PuLlruREhKyu7Abk354jBHoWI7iktPh7pNlZTafDd6iNMmJMtiJwqSZHId1USMCOOW6DHTisp4lK6WzKUNEdM1lavYmxe2ha0MXkmAxjyymNu3b0xjjFTIioioiqqKAoVRgADoAK43Ns1FxRSAMZpeuaADFGKQBjvRTASoJetAEf4UUASx9amNAFa+sbfU7Gexu0Z7e4QxyqrshZT1GVINcva+HZrL4lC+trFIdJj0hoVkUj5pnmDN33EkAkk/wBa3o1ZQi4dCJK7udLqQ/4lF9/17yf+gmsjwKuzwH4fwc/6BCQf+AiknFUitb6mxZWFrptmlpZW8dvbxklYoxhQSSx4+pNWsVi5NtvuPXYO9LS6WY0xRRQIKKYBRQAtIaACigBKgl70ARZNFAEsVT0ALQaAM/XZGh8PapKihnSzmZQehIRqh8NWv2Hwro9pnd5NhBHn1xGorSy9mLqalLWYwooAM0ZpAGaTNMBc0ZoAWjtQA3NFIA5qvKeaYEVFK4DlJB4p/mmmA4SnFBkPPPSgBsu2WF4pAGR1Ksp7g8EUkYWKJIo12xooVVHYDoKLu1hdB4k570eYc0DE8w4FL5vvQA0SECjzSKVgF80kDmjzfemAvmH3o8w80AHmUeZ2oAC5B65pPNoAPMNRltxoAXH+cUUrAf/ZAP/tACxQaG90b3Nob3AgMy4wADhCSU0EJQAAAAAAENQdjNmPALIE6YAJmOz4Qn7/4THoaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLwA8P3hwYWNrZXQgYmVnaW49J++7vycgaWQ9J1c1TTBNcENlaGlIenJlU3pOVGN6a2M5ZCc/Pg0KPHg6eG1wbWV0YSB4bWxuczp4PSJhZG9iZTpuczptZXRhLyI+PHJkZjpSREYgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdG9yVG9vbD5XaW5kb3dzIFBob3RvIEVkaXRvciAxMC4wLjEwMDExLjE2Mzg0PC94bXA6Q3JlYXRvclRvb2w+PHhtcDpDcmVhdGVEYXRlPjIwMjMtMDctMTJUMTA6MTg6MzUuMDY3PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48L3JkZjpSREY+PC94OnhtcG1ldGE+DQogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwADAgIDAgIDAwMDBAMDBAUIBQUEBAUKBwcGCAwKDAwLCgsLDQ4SEA0OEQ4LCxAWEBETFBUVFQwPFxgWFBgSFBUU/9sAQwEDBAQFBAUJBQUJFA0LDRQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQU/8AAEQgAUACzAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A/QlI/NYZ78fL/X/PerH2Fun59/8AI/8Ar02zYbuvA9sVqcMMjOPWuIszPsZUYOcdOKVrHOfl5x2/nWj14Jo68d6AM02J3ZK43dfw+tH2FlyOfQitM4bPGRTedvTBBoAz1sz2Hek+xbcDkZGPyrS3fMcflQfrz2oAzVs+mR1pFsSqnA71p544H8+9H3u2aAM02Z9uR/n6VG1keuOBzWrnjrzSMepHTH+NAGU1iehH59RR9hLfga0v0PQ4oDfMO3TFAGcLF89GIDckc+n+fwoWxZTjbtPHT8/8/StHjke3Sl9O5zigDONieCB78D+mKFsTyNv1xWjk9jge9G7rzznPuKAKH2NuDyOMY9OvT/GnfYX4J9eQPf8A/VV/dz749KVehGMGgDPFmVxkZGenr/nn86gkiaNiDnPbgf1981rqxHqKz777zcdDnH4En+n0NS9gIVlfHHA/Gio/w6cdqKLoCxYn5l/zxWopG7rWVYn5uef0rU5OeKoA6MR1xS/XqDSFuufT1pN348Af/qoADnvSbvvAkE55/Co3kHOfrQHG4dv5UASelLuyvrn8qj3d80rN2P05oAfn8qTnimb+hpgmB759cGgCU9xnrxQ3f+eKikk7HngHFJ5nvn/GgCRvlzzj/P8An86Td3z+P41E03UDkt2z1FIJPfOT6f59qAHlsOpzjqDjp2p3tnHQGoJHOcheAw57VID36H86AF3buOhIz+NO3BV5OeK4DXPHV/pvxq8LeEkitDpWraJqWoSyMr/aBNby2qqEIYLt2zNkFSc7eRjnpPFnizT/AAPoZ1fVmlitBNb2y+TE0jvJPNHDEqqOpaSRBzwMkkgAkbvD1VypK/Mri5lq+xtK4+6c5p6tx2J7EV4p4i+Lkvj7WvBnhn4bavbyHxHZNrV34hgWOVtP0pSVEqRSD/WySgwrvQ7GDblyDj1Tw/ocuiR3Eb6tqGqpIyvH/aUiSPDhQrBWCKSrEbsMTgswUhdqrpWwsqMIyqOzey6/MmNRSbS6dTZBHzVn3bDzDjseR/n6VLHqVpJqM2npdQPfQRpNLarIpljjfIRnTOVVtrYJGDg4qveH94fSuOz6ljI8FAcj82/woqu28MRkfkaKLAXLH73TjP5c1qcd+B346e9ZVicMO2MY9q02b19KYDSflxnORj2+tRyTrCjSSSLFFGNzO5woUDkknsBzSs3zYz1rzH9ozULuL4V3+kafJHHqXiSeDw9bPM+1VN1J5bn3IiMpAHJIHTqOjC0PrFeNPa7RFSXLTcuxwOh/tIa8vjzwrN4l0uzsPh/48upNP8M30SOk1vOv+pW7Zmxm5AOwBVIYqPmBJH0NHJ1I5FfPv7ZHgG1/4ZT1VNERbG68Dx2ut6JJEB/ob2O3DKe2IRIOMfXrXt2ga1D4g0ew1aAqLfULaO7jI5GyRFkX9GFehj40qqVajHlV2rLy9fIyo80fdk7ve5tr1/PpSSN+FQwybl3AfKB1A6VQm8Qacuuw6I2oWw1qW1a+TT2lHnNbq6xtKE67A7oufVgK8mMJPY3NLcWB2kZrh/APxQsPiFqXjK1sra6tn8Ma9NoNz9oCYlmjjikZ0Kk/IRMoGcHjkCtPwf4ytfFeo+JoI7a5sU0DV30ieW8VUSZ1ggmMkZB5jxOACcHKnjpXy/8As5/E43Hwz8Xal4Njtr7xV4n1zV/GDLqiSeRY2cszJHPdkFdoYWxCR71ZlUsAVXFephsvnXhPTVNLV23MZYhRs+h7p+01451P4b/ADxv4n0m9/s7U9NsPMt7zYj+SzSom/a6spIDHAIIyBxXo0YfysvtWQj5lXoGxyPXtXzP+014gufjJ/wAE9fEXiXTraWxm13w7Yav9lR8mJTNbzyJk/eUKrjPdfrXS3Piz4ofEXQ9a8c/DKS0i06Fri00TQdbt4/K12GIuq38MuVaMyPzD5jeW6IgbYrmStYYT2lBRbUbN6v5aA52lzau62/U9fbxRokeuSaI2sWI1hIvPaxNwomEZdVDFM5A3Oinvl1H8QrM074laPqnj7UvDMGp6TNLZWzPIsOoq9yk0RH2mKWHb8gjWS2bdub/WkEJgFvmb4a/s6+IPiF4J0bStf8P6p8Pb7Q9GQPrWryQ3d3qevyyCa4u2RJW82BJYkdRIyljJjA25HoPjz9izQfGnhbSLaHXrzSfFUNxcTal4nhgXz9WW7hEN+s8YYKRMioADuCbFGHG8N1ywuW0kuao7vfTZ97mEaleTfuq3ruSfCzw3/wALw8Wa18SvF9rHq2l2+ry2/gzS7l/PsrK1tpDGmoRxY2NPNIpkEpDMoxsfbtx9CR8q27k+/WqukaJY+H9Ls9L021jstPs4I7a2tYVwkMKKFRAPQAAfhV5UGRn1rw8XXVao3FWitF8jrpx5Y679T5F/bm17xz4d8ZfD3UPhlEJfGNloXia4z5SyCO0+z2yySAE8ujbGQdPMC5yu5G73wz4d+HOi+CfD3jHRvG0+naJdSW2o3GoeJtalmg1WOPdu+0x3UmEmyWbcgR45oY85EZQ+ofEDwnrviyTTLPTfEd54Z00eaNRutKSJb+RSEKRxzOG8lWIbc0a7+EIZcc8B8M/2Ovht8IfFM2r+GtHWCCWEKdOvooryNJgADPHLKjTIxAwVWTZ6KK92liaMMPCMp2kl066vft5HO4y53ZXXmcD+z/pvhj4f/E34h6n4RuNQ8Z6J4injfRovD1lPc2lnD5txcTR/anSO1iHnXbgRrM3EYJwxIH1JZTTTwCSa1a0Y/wDLNpFZh9duV7nox7fhYC4x6Dp34pw7YP1X9K8nGYxYqfPbWyV3ubwjyoz5vDejza4msvpVk+souxNQaBWnVcAbVcjcq4GNoOOTxyaS+A3Pxjj8K0ieG/Osy/I8wjng5/WvOcnLd3saFbyz6fz/AMaKRrgqxAjOB/tf/Xoo1Au2P+sGelX2OPfFZ9n/AKwAYNaL9iTigCMbixVRknGFAPNfKH7Qvx00df2hvgp4W0vUodditNV1LVNXs9FgbUruGaC1mhhjEULEq4kebKt90oCw2qwP054r8K6Z448N6p4e1qCS50nU7drW7ginkt2kjbhl3xsrgEcHBGQSOhxXimg/Be98M/tdL4h0rw/Dp3gW18ES2MV3GyYk1Ce/WWUEFjI8jKhZnb2ySWFevl0qFNyqVW+ZJ2XT7znrc0rJbXH/ABb1r4g+PPhP4zsNM8A2OjaZeaJeQSy+KtXVJvKe3cOVtoElGQp4WSRcnqQBk+efBnRfi3r3wZ+DVnpfjCHTPD174VUveaRpCLdWzRWkP2GF2m85cMA++T5NxVVCrnn6g8brjwL4kxkH+y7oZx/0xeuD/ZRh+z/s0fCsI+/HhqwKtgjrChB9uMfTpXbTxcIUfaRhFK+2/wCZDpuTs5Pb0PFfCN14s/ae+CWq/FDw9qWp6J4m/s2IeErfT9QZIlvLaFHuGZCQjmW7E1uRICPLiX+Fq3FT4y638RPD3xe0zwUsiXlpLoC+DNZuU0+50yxljilN3OzuQCbuEM0agyeSsQCls7Po/wAMeEdH8E6FBougabb6PpNsztFZWqBI42d2kfA92diT3JNbKru46c9v8+9YSzZ3klBW2WnTt/WofV1o7u+/zPDvhr8D/G3hHUtcstX8b2eqeFNU1abXLm3trB476eeZU8y3M5fCWwZOAA0jKcbkAAEfh39jjwZ4Z8P3/hmx1nxNH4Lv2c3vhpNRSK3uw64dZpo4luZVZQEIaY/IoTO35a92A+b0znrT+hziuWeZYiV7O1910ZpGhBJLsZs3hfSLrw43h6fSrGXQntPsDaW1uhtTb7PL8nysbdgUBQuMAcYxV62tYbWCKGCKOC3jVY0iiQIiKoAVVUDgAAAAdAKlUcY64p2Dwetea5zluzcYEOcng/yoK5z3Ap+T0BxSc0gGFAxz1/GnFd24e9H6/hQOOD/nigBdvXgDmk24A+vSn7ux7UjEcj60AJtA+YHnsO9J6dh/Knbt2fXNN3bcc0AJzx6/Ssy+/wBZ6/j6Vp5Ix6459OtZd9g7u/FAFQfT9P8A61FN8w0UwL1h82K08jB/pXOws8b8frVr7cwFFgNpeBkelDfNxn6Vlx3zbf1H5UNeMN3PA5pAZHxWvZdM+FnjW8gjWa4t9Dv5Y45MFSy20hAOeME+pFZ/wR0AeFfgv8P9GD+b/Z/hzTbXfx85S2iUn8SM9+tb+oGLULG4s7lVlt7iNoZYW6OrAqyn2IOPxNJZrHp9nb2ltH5VtbxrFFGuSERBhV55wAAK39qvY8ltbkW15jZHXilYhePfvWct78xByPzprXrbuuO/XpWBZp9BgDrS5DDJ6elZIvH2rnp35xTvtxGPmHpmgDS8zgnHpx/n/PFHmdB7f4VlLeOq89cYP4ZpBflcDPakrgbG/jJFN3lce9Zn28sqjPPSk/tA44btx+VMDULj1xxTlY9+ayvtj8HJI9KVb1lDYPv/APXoA1d3T1o3fL046+/Sstrwtxk88daX7YWypOeQR+dAF/cex5o3E8E47VmtdFWyG3DAB9aT7eR06knt+lLUDT3N1znisnUG+YH2PT6/z4p/25u3TpVWSUzN161QEO7bwTg/SirIj9/0oqNQP//Z)A close up of a writing

AI-generated content may be incorrect.

**Signature of the Student Signature of the Supervisor**

**Name: Anubhav Kumar Saurav Name: Renju C Panicker**

**Date: 25-03-2025 Date: 25-03-2025**

**Place: Bangalore Place: Bangalore**

Contents

[1. INTRODUCTION 6](#_Toc195513448)

[2. LARGE LANGUAGE MODELS 8](#_Toc195513449)

[2.1 Evolution of LLM 8](#_Toc195513450)

[2.2 How a Large Language Model Works 10](#_Toc195513451)

[2.2 Classification of Large Language Models 11](#_Toc195513452)

[1.1.1 Architecture-Based LLMs 11](#_Toc195513453)

[2.2.2 Availability-Based LLMs 11](#_Toc195513454)

[2.2.3 Domain-Specific LLMs 12](#_Toc195513455)

[2.3 Vision Language Models 13](#_Toc195513456)

[2.4 State of the Art Large Language Models 14](#_Toc195513457)

[3. VISUAL DOCUMENT UNDERSTANDING 17](#_Toc195513458)

[3.1 Multimodal Transformer Architecture 18](#_Toc195513459)

[3.2 Comparison of LLM models for Visual Document Understanding 19](#_Toc195513460)

[3.2 Visual Document Understanding with OCR free DONUT model 19](#_Toc195513461)

[4. PRACTICAL IMPLEMENTATION 22](#_Toc195513462)

[4.1 Dataset Preparation 22](#_Toc195513463)

[4.2 Pre-Processing Tasks 24](#_Toc195513464)

[4.3 Training And Fine Tuning the Model 25](#_Toc195513465)

[4.4 Technical Considerations 27](#_Toc195513466)

[4.5 Performance Evaluation 29](#_Toc195513467)

[5. CHALLENGES AND FUTURE PROSPECTS 30](#_Toc195513468)

[6. CONCLUSION 30](#_Toc195513469)

[7. ABBREVIATIONS 31](#_Toc195513470)

[8. APPENDICES 31](#_Toc195513471)

[8.1 Appendix A: Environment Setup & Essential Python Libraries 31](#_Toc195513472)

[9. REFERENCES 32](#_Toc195513473)

[Figure 1: Two Stream multi-modal (left) and single stream multi-modal design (right) 10](#_Toc193842543)

[Figure 2: Transformer architecture of DONUT model 11](#_Toc193842544)

# INTRODUCTION

The dissertation work focuses on digitizing the aircraft engine maintenance logbook records that are currently maintained in paper-based form in the aviation industry, using the application of large language model that leverages computer vision, natural language processing and machine learning techniques. For decades, maintenance crews have relied on paper logbooks to record maintenance activities and compliance checks when an engine undergoes maintenance, repair or overhaul activities. Paper-based maintenance record keeping still dominates the aviation industry. While traditional method is effective, this method has limitations including the risk of damage or loss, difficulty in data retrieval, and the potential for human error. The objective of the project is to automate data extraction, document classification and visual document interpretation of engine service logbook records, mapping a desired structured output directly from an input document image. This will help in reducing the administrative workload, enhanced collaboration among stakeholders and aircraft engine data preservation in digital form.

Different types of state-of-the art large language models, their underlying architecture and capabilities in solving specific use cases pertaining to industry needs are surveyed to arrive at the decision of selecting a model for the practical implementation of the solution that meets our requirements of Visual Document Understanding (VDU) tasks. VDU aims at understanding digital documents either born as PDF’s or as images and focuses on varied document related tasks like entity grouping, sequence labeling, document classification [2].

The proposed system integrates OCR-free multimodal transformer-based Document Understanding Transformer (DONUT) model framework for visual document understanding (VDU) tasks to interpret both textual content and spatial features of the document image using computer vision (CV), perform information extraction, contextual understanding and key-value pair mapping using natural language processing (NLP) methods. It also features text prompt and visual question answering (VQA) features for querying on the extracted information. The dataset for training the Donut model is prepared consisting of aircraft engine logbook records in jpeg image form. Pre-processing tasks such as image annotation, tokenization, image preprocessing, dataset formatting etc. are performed on the prepared custom dataset before it is used for training and fine-tuning the model.

By structuring unstructured engine log data into a queryable format, this approach enhances data accessibility, error reduction, data preservation and better collaboration among maintenance personnel. The fine-tuned Donut model achieves state-of-the-art performances on various VDU tasks performed on the engine logbook dataset in terms of both speed and accuracy [1]. The output of the trained model is evaluated and compared with a few benchmark models which excel in visual document understanding tasks. We have further identified the challenges and limitations of the current model and how further its performance could be enhanced. The implemented 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.

# LARGE LANGUAGE MODELS

Large language models (LLM) 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. LLMs have revolutionized natural language processing (NLP) and artificial intelligence (AI). 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 driving innovation by automating tasks, personalizing interactions, and enabling data-driven insights, to name a few.

## Evolution of LLM

The advent of neural networks in the late 20th century marked a pivotal moment in developing LLMs. Researchers began exploring recurrent neural networks (RNNs) and later long short-term memory (LSTM) networks to handle sequential data. These architectures enabled models to capture dependencies in text, albeit with computational efficiency and scalability limitations.

A breakthrough came with the introduction of transformer-based architecture by Vaswani et al. in 2017. Transformers utilize self-attention mechanisms to process input sequences in parallel, which improves the training efficiency and performance significantly [10]. This innovation laid the foundation for subsequent large-scale models, including BERT (Bidirectional Encoder Representations from Transformers), which achieved state-of-the-art results across various NLP tasks [11].

**Growth of Pre-trained Models**

Following the success of BERT model, the LLM field witnessed an explosion in the development of pre-trained models. These models are trained on massive datasets of text corpora like Wikipedia, BooksCorpus, WebText2, Github etc. using unsupervised learning techniques, allowing them to learn rich language representations. Notable examples include OpenAI's GPT (Generative Pre-trained Transformer) series and Google's T5 (Text-to-Text Transfer Transformer). These models demonstrated impressive capabilities in generating coherent and contextually relevant text, spurring interest in LLMs. The journey of LLMs from early theoretical concepts to today's sophisticated models exemplifies the remarkable progress made in AI research. Since the launch of ChatGPT by OpenAI on November 30, 2022, large language models (LLMs) have become popular more quickly across the industries [8].

**Multimodal Large Language Model**

In 2023, OpenAI released GPT model 4, which is a multimodal large language model (MLLM). MLLM are usually trained on different types of datasets comprising of text, image, audio and video [9]. MLLM is a robust model that integrates multiple modalities of input data such as text, images, audio, and video. Therefore, it is considered a more advanced version of LLM [8]. Unlike traditional language models that rely solely on textual inputs, MLLM’s leverage the power of multimodal data to enhance comprehension and generate more expressive and nuanced responses.

Multimodal models, like CLIP (OpenAI), LLaVa (Meta), Qwen (Alibaba) have demonstrated impressive capabilities in tasks such as image captioning, text-to-image generation, image-to-text generation and multimodal translation. These models represent a significant advancement in understanding and processing multimodal data, paving the way for more sophisticated human-computer interaction and content generation application [9]. The evolution journey of language models with prominent framework developments is depicted in Figure 1.

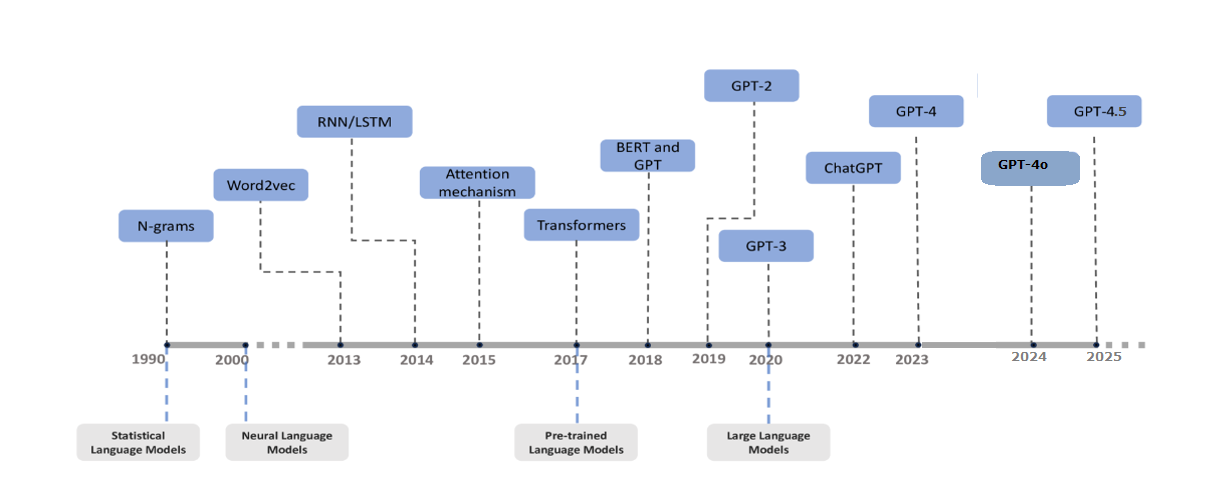


Figure 1: Evolution of Language Models

## How a Large Language Model Works

A transformer model is at the core of architecture of a large language model that undergoes extensive training on huge datasets consisting of textual data. A transformer model consists of an encoder and a decoder module [10]. The encoder module processes the input data by first tokenizing it, then generates numerical representations of input text known as embeddings, forming mathematical equations to derive relationships between tokens and at the end the decoder module generates output sequences of text by analyzing these embeddings. Transformer model has self-attention mechanism that enables to weigh the importance of different words in a sentence while processing the data and distributes the computational load efficiently.

Training a large language model comprises of two-step process which includes pre-training and fine-tuning. In the pre-training phase, the model tries to learn from vast dataset of unlabeled text data in an unsupervised learning method where it predicts the next word in a sentence and captures the language’s statistical patterns and linguistic structures. The next phase i.e Fine-tuning where the model specializes for specific tasks and get trained in a specific dataset with labeled examples, that further refine the abilities of the model to generate coherent and context-relevant text.

## Classification of Large Language Models

LLM’s are classified based on their architecture, availability, and domain specific usage.

### Architecture-Based LLMs

1. **Autoregressive Model**

Autoregressive model is a type of generative model that predicts the next token (word, character, or subword) in a sequence based on the previous tokens. The model learns from large amounts of text data and uses probability distributions to predict the most likely next token. Example: GPT (Generative Pre-trained Transformer) series, Claude etc.

1. **Autoencoding Model**

Autoencoding model uses an encoder-decoder architecture and works by encoding input data, such as text, into a more compact form, and then decoding it back into its original form. It learns from both past and future words (bidirectional context) rather than only predicting the next token. Example: BERT (Bidirectional Encoder Representations from Transformers)

1. **Seq2Seq Model**

Sequence-to-sequence (Seq2Seq) model transforms one sequence (input) into another sequence (output). It is widely used in machine translation, text summarization, question answering, and dialogue generation. They mostly use an encoder-decoder architecture. The encoder processes the input sequence, and the decoder generates the output sequence. Example: T5 (Text-To-Text Transfer Transformer), BART

### Availability-Based LLMs

1. **Open-Source Model**

Open-source LLMs offer transparency, flexibility, and community-driven improvements and are publicly available meaning that anyone can download, view and modify the raw code that powers the model's algorithms. These models are developed and supported by a community of developers and researchers. Their main advantages are transparency, flexibility, and the ability to customize the models for specific needs. Examples: LLaMA (by Meta), Qwen (by Alibaba Cloud) and Falcon (by Technology Innovation Institute).

1. **Proprietary Model**

Proprietary LLMs are developed, maintained and owned by a specific organization or company, using their private training data and resources. They are made available for usage through commercial licenses or subscriptions. The company retains exclusive control over access to the source code, data and are not available to public. Examples: GPT-4 (by OpenAI), PaLM (Pathways Language Model by Google), and Claude (by Anthropic).

### Domain-Specific LLMs

1. **General-Purpose LLM**

These models are versatile AI models designed to meet the requirements of a wide range of natural language processing (NLP) tasks without being specialized for any single domain. These models are trained on diverse and massive datasets of books, articles, code, conversations etc. and can perform tasks like text generation, translation, summarization, code generation, reasoning, and more. Examples: GPT-4, LLaMa, Gemini, Qwen, DeepSeek etc.

1. **Domain-Specific LLM**

Domain-specific LLM is a specialized model that is trained or fine-tuned on domain-specific datasets to improve accuracy, relevance, and performance for specialized tasks meeting the business requirements of particular industry or organization. The training process is the key differentiating element between a general purpose model and a domain-specific model. Few domain specific LLM’s with their use cases are mentioned here:

* + *Healthcare:* Use cases include medical question answering, diagnosis, research, clinical decision support, biomedical text processing etc. Ex: Med-PaLM 2(Google DeepMind), BioGPT (Microsoft).
  + *Finance:*Use cases include financial report analysis, market prediction and economic trend analysis, fraud detection, financial sentiment analysis, risk assessment. Ex: BloombergGPT (Bloomberg), FinBERT (AI2 Labs).
  + *Legal:*Use cases includes legal texts understanding, contract review and compliance, legal document processing and legal research. Ex: LegalBERT (OpenAI), LEXLM (Thomson Reuters).
  + *Coding & Software Development:* Use cases include AI assisted coding, code reviews, code generation in multiple programming languages, algorithms analysis etc. Ex: Github Copilot (Microsoft), Starcoder (BigCode), Codex (OpenAI).

## 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, similar 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 implementation to achieve the desired result of information extraction.

## State of the Art Large Language Models

Large Language Models are advanced AI systems that process and generate text, images, audio, video, and other data types. The recent advancement in LLM’s can handle data from various modalities and are called multi-modal LLM’s. 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.

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** – This model can generate 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.

**5. Alibaba LLM Model**

* **QWEN2.5 VL** – An advanced vision language model from Qwen vision-language series developed by Alibaba cloud team. It has state of the art visual recognition, document and diagram understanding, precise object detection, document parsing, multilingual support and video comprehension capabilities.

**6. IBM LLM Model**

* **SmolDocling** – An open-source ultra-light weight vision language model developed jointly by IBM and Hugging Face for visual document understanding tasks such as parsing documents and convert them into formats like Markdown or JSON. It is the smallest multimodal model in the world having only 256 million parameters.

# 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 [2].

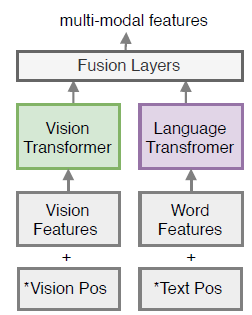
Traditional approach to VDU was based on Optical Character Recognition based processing. It has two step process: (a)Text extraction with off the shelf OCR engines (b) Downstream processing of information. The OCR method primarily extracts text from images by converting the document’s visual text into 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 [1].

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 [1]. The current state of the art VDU models is 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 [9]. 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.

****  **A diagram of a diagram

AI-generated content may be incorrect.**

Figure 2: Two Stream multi-modal (left) and single stream multi-modal design (right)

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.

## Comparison of LLM models for Visual Document Understanding

We have studied the use cases and underlying architecture of different large language models used in industry for visual document understanding task. For analysis and evaluation purpose, we selected SmolDocling and Qwen2.5 VL as benchmark models.

|  |  |  |
| --- | --- | --- |
| Donut | SmolDocling | Qwen 2.5 VL |
| 1. Consists of image Transformer encoder (SWIN transformer) and an autoregressive text Transformer decoder (BART mode) 2. Base model is trained on approximately 11 million parameters. 3. The base model is trained using 64 A100 GPU. Number of layers {2,2,14,2} for encoder and 4 for decoder. | Base model is trained on approximately 256 million parameters | 1.It is a huge model based on ViT transformer with window attention mechanism trained on billion of parameters.  2. Qwen 2.5 VL is trained on 3B, 7 B and 72 billion parameters. Its available in three series.  3. Along with document parsing capabilities, it has long video comprehension, fine grained video grounding, agentic capabilities and precise object grounding. |

## Visual Document Understanding with OCR free DONUT model

For the implementation of training methods and fine tuning the model for visual document understanding, the donut model is chosen, which is an OCR free transformer-based architecture model. The donut model has simple architecture which has the advantage of efficient memory utilization and time cost.

The reasoning behind choosing the Donut model is mentioned below:

1. **OCR-Free Operation:** Donut bypasses the need for Optical Character Recognition (OCR) engines, which are often prone to errors and require additional processing steps.
2. **End-to-End Approach:** Donut directly maps document images to structured outputs, eliminating the need for intermediate text extraction or post-processing methods.
3. **Scalability and Cost-Effectiveness:** Donut's transformer-based architecture allows for efficient scaling to handle large volumes of documents, and its OCR-free nature reduces operational costs.
4. **Memory Efficiency:** Donut is designed to be memory efficient, making it suitable for deployment on devices with limited resources.

DONUT, which stands for Document Understanding Transformer is a vision-language model designed for visual document understanding (VDU) tasks developed by Naver Clova AI Research. 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 [1] 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 [12]. 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.

*Donut = Visual Transformer Encoder + Textual Transformer Decoder*

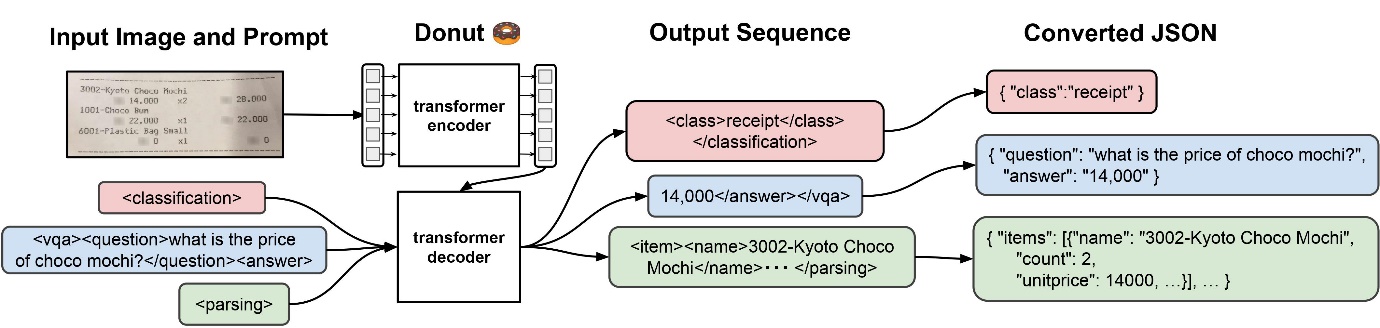


Figure 3: Transformer architecture of DONUT model

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 second phase is known as Fine tuning phase where the model learns to understand the documents on a new task.

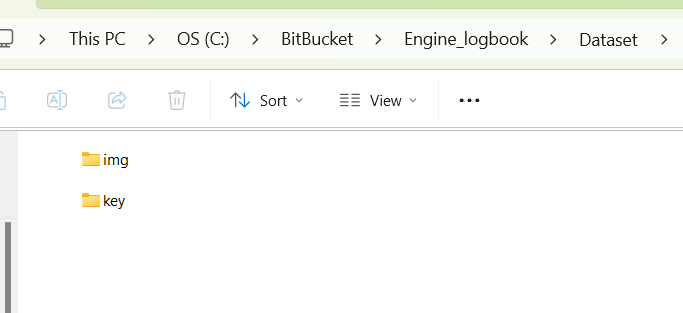
# PRACTICAL IMPLEMENTATION

## Dataset Preparation

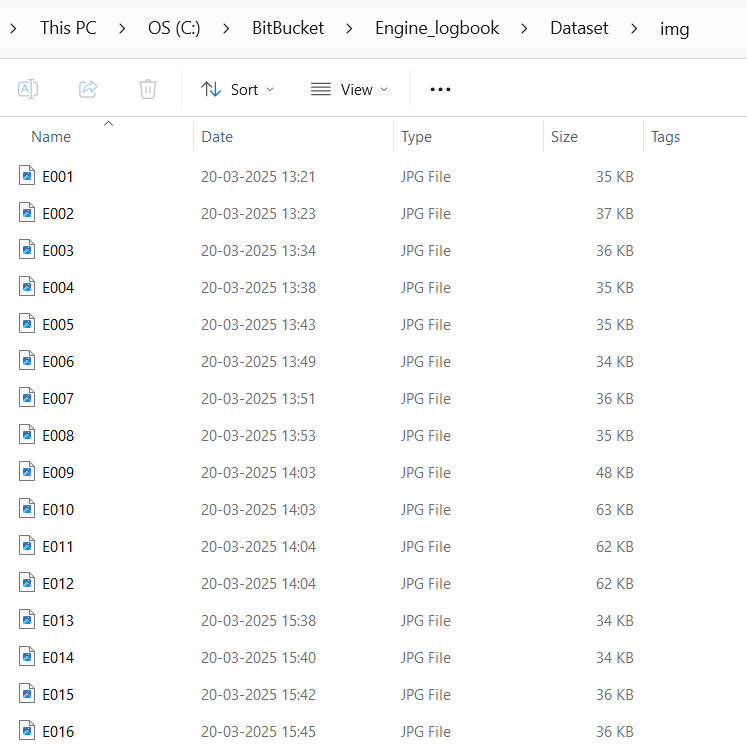
We have created a dataset of engine service logbooks comprising of different invoices, templates and data in-order to train the transformer-based donut model on the custom dataset. The dataset comprises of 200 annotated data images which are utilized for training, fine tuning and testing the model. The dataset directory consists of two folders- *img* and *key*. The first folder contains the images of the engine maintenance logbooks and the other folder contains the 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 in a single file. The ‘metadata.json’ file contains two columns: text and filename. The “text” column contains the text annotations of the image and the “filename” column contains the image file name.

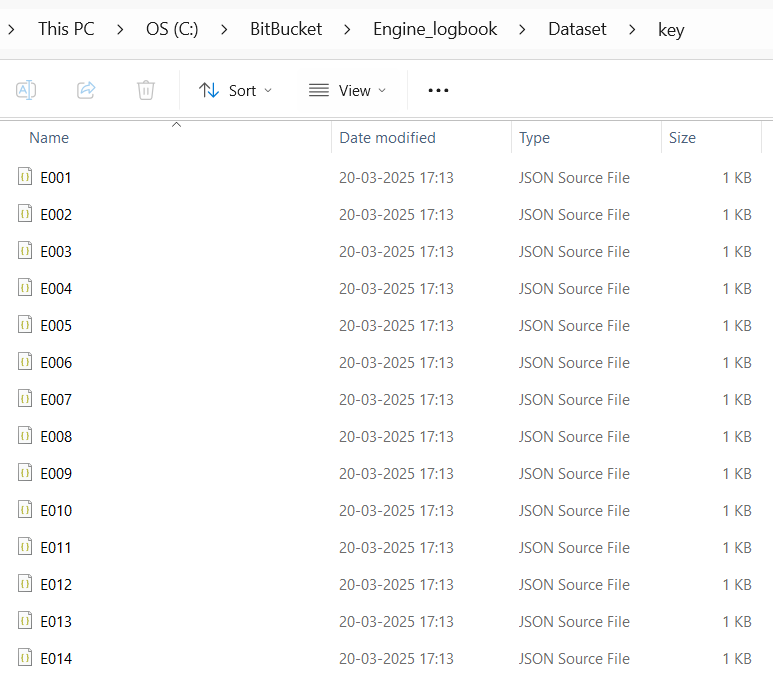
The dataset directory structure looks like this:



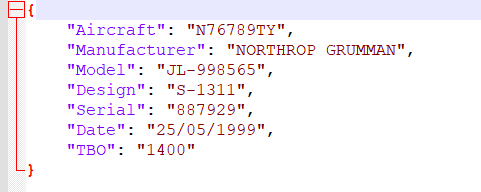
The ‘*img’* folder contains the images of the engine logbook in jpg image format.



The ‘*key’* folder contains the annotations of the images in JSON file format.



A sample individual image JSON file contains annotations like this.

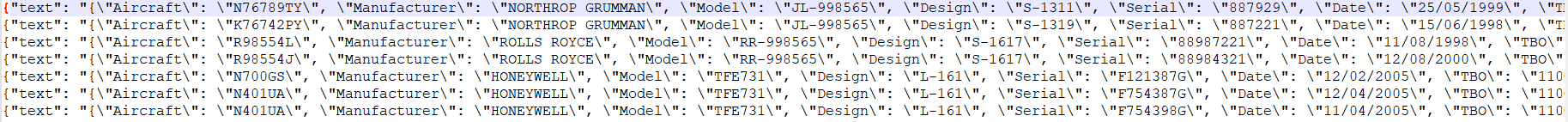


The *“metadata.json”*file will consist of information like this as given in example below.

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

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

A snippet of the metadata file is shown in figure below.



## 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:** The tokenization process divides the text or sentence into individual words, characters or sub-words called tokens so that it becomes easier for machines to understand and process. 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. For this ‘DonutProcessor’ is used from the pre-trained model and new tokens are added to it.

1. **Image Pre-processing:**

* **Resize and Normalize:** Resizing the images to a consistent resolution ensures uniformity before being processed by the model. We have resized the images to a resolution of 720 \* 960 (width x height).
* **Convert to Tensor:** The images are converted to PyTorch tensors for GPU processing. We have ensured that the shape is in CHW format (Channels, Height, Width). For

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 the Model

Fine-tuning the donut model means additional training of a pre-existing model, which has already learnt the features and patterns from a huge-massive dataset consisting of millions of parameters, using a custom domain-specific engine logbook dataset. Training a large language model from scratch is extremely expensive and it requires extensive high-end resources for computational power and time. Fine tuning leverages the existing acquired knowledge embedded in the pre-trained model and allows to achieve enhanced performance on specific tasks with substantially reduced data and computational requirements. This reduces hallucination, irrelevant information and produces more consistent output. A top-level overview of LLM fine-tuning process is illustrated in Figure 3.

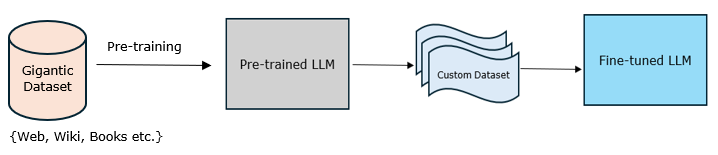


Figure 4: Fine tuning LLM

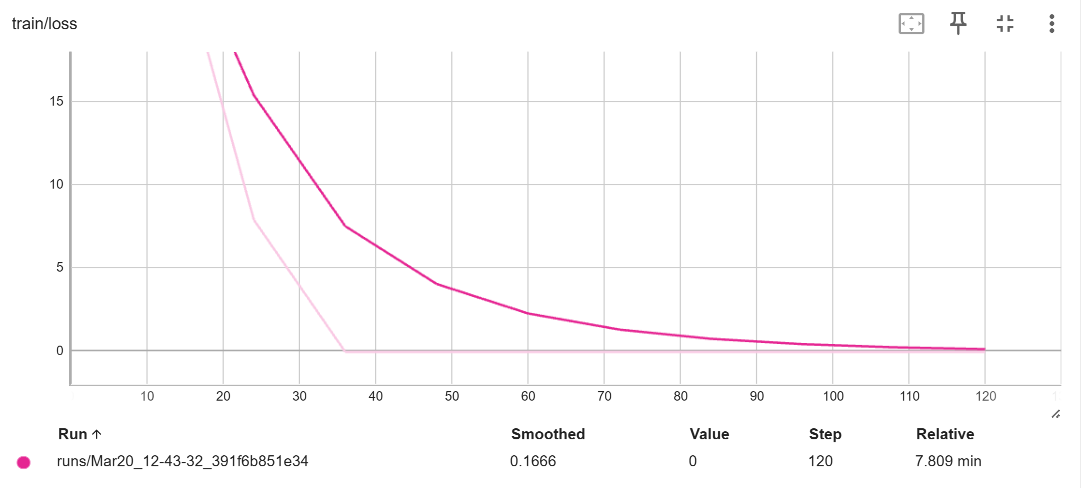
Once we have loaded our custom dataset and performed the pre-processing tasks on it, the training arguments or hyperparameters are defined which are passed to the Sequence to Sequence (Seq2Seq) trainer that uses transformer architecture. A Seq2Seq trainer is a specialized neural network model used for tasks where the input and output are sequences of varying lengths. It has transformer architecture consisting of encoder-decoder pipeline that leverages the power of self-attention mechanism, making it highly efficient for tasks like machine translation, text summarization, and question answering.

After initializing the training hyperparameters, the donut base model is ready for training. In-order to start the training, we have used the pre-trained donut base model with VisionEncoderDecoderModel class from Hugging face library to fine tune it on our specific dataset. The donut base model is pre-trained on two main datasets: IIT-CDIP (11 million images) and SynthDoG (English, Chinese, Japanese, Korean, 0.5 million x 4). SynthDoG is a synthetic dataset and IIT-CDIP (Illinois Institute of Technology Complex Document Information Processing) dataset is a real-world dataset having collection of scanned document images, consisting of 400,000 grayscale images in 16 classes. The donut base model contains pre-trained weights which means it has already been trained on a larger dataset and can be used as a starting point for fine-tuning on specific dataset.

The engine logbook custom 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, logging steps etc. are defined as arguments for the training as mentioned in Table X. After the training of the model is completed, we will get a processor and a fine-tuned model as output of the training that is saved on hugging face repository. This fine-tuned model is used further for prediction tasks and evaluation on new test data. Refer Appendix A for code reference.

Table 1: Training Hyper-parameters

A training accuracy of 84 % is achieved while fine tuning the donut model on the custom dataset of engine logbooks which is depicted in Figure X.



## Technical Considerations

Training and fine-tuning the large language model requires significant computing resources for flawless performance. This includes high-performance GPUs or TPUs with CUDA support, ample RAM memory, and substantial hard disk storage to store large datasets and to save the trained model. Further, the software stack requires python environment with necessary packages, machine learning frameworks (such as TensorFlow or PyTorch), transformer libraries and repository access to manage and scale the implementation.

The software and hardware requirements are mentioned in Table 1 which is used for data pre-processing tasks, training and fine tuning the model and predicting the output.

Table 2: Software and Hardware Requirements

|  |  |  |
| --- | --- | --- |
| 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 ≥ 4.45): A library that provides pre-trained models and tools for various NLP tasks, including Donut.  Hugging Face Repository: To save the trained models and processed dataset on cloud repo so that it can be accessed from any location. |
| 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. |

## Performance Evaluation

We have evaluated the performance of the trained Donut model on different scenarios. The test evaluation includes comparison with other open source LLM models such as SmolDocling from IBM and QWEN from Alibaba Cloud which is a vast model pre-trained on massive datasets.

**Information Extraction:** The model is able to extract data from the scanned input images of engine logbooks. The model is tested on different types of service logbook formats, layout and templates that includes various types of data of aircraft engine. It also includes manually written words by the maintenance personnel. The model has shown satisfactory performance on the information extraction task from the input document images.

As compared to SmolDocling model, a light-weight model from IBM particularly designed for visual document understanding tasks, the donut model is able to match the performance.

On handwritten texts, QWEN-2.5VL model has shown remarkable performance in recognizing the text and converting into markup language.

**Document Classification**: Given an input document, the model is able to identify and classify the type of document it belongs. The trained model donut has shown good performance on this.

**Visual Question Answering:** We have evaluated the performance of the donut model for document visual question answering where the model needs to answer questions asked on document images. This task involves a computer understanding and answering questions about the content of a document image, going beyond simple Optical Character Recognition (OCR). VQA becomes challenging on mixed text and image content like charts, diverse document formats, and layouts.

We have asked questions to the model like- (what is the aircraft number? What is the engine serial number? Date of maintenance performed?) The donut model shows better performance on visual question answering tasks in comparison to SmolDocling model. However, QWEN model shows better performance also.

# CHALLENGES AND FUTURE PROSPECTS

The application of large language model in automating the task of digitizing the aircraft engine service logbook shown a remarkable performance in various visual document understanding tasks including information extraction, visual question answering, document classification which makes the underlying method OCR free and goes beyond text extraction by the application of computer vision and natural language processing techniques.

However, they also come with certain significant limitations and challenges. The quality of the input image plays a significant role in the output result. We have tested on less resolution images and the performance of the model reduces significantly. This implies that dataset preparation requires intensive pre-processing tasks to be performed. On handwritten texts, the donut model sometimes fails to recognize few letters of the words which are not clear and requires intensive training to recognize the manual or handwritten texts on the aircraft engine service logbooks. Further on templates which has complex layout including charts and multiple tables, it becomes challenging for the model to infer the data correctly and extract the right key value pair.

Adequate training of large language models requires massive amounts of high-quality training data. Creating such extensive and diverse datasets can be challenging, particularly for specialized domains or languages with limited resources. Obtaining and curating large-scale training data can be time-consuming and resource intensive.

# CONCLUSION

Large Language Models offers immense potential of transforming the landscape of aviation industry, offering groundbreaking capabilities and use cases that can ultimately redefine the functioning of industries bringing more efficiency and better collaboration among the stakeholders.

In this dissertation work, we have implemented a use case of digitizing the aircraft engine logbooks which are currently maintained in paper-based form across different Maintenance Repair Overhaul (MRO) units of aviation industry through the application of multimodal large language models (MLLM) that leverages computer vision and natural language processing techniques of AI. The MLLM is trained and fine-tuned on our custom dataset of engine logbook. The result of the implemented solution is evaluated and compared against few state-of-the art LLM models such as Qwen-2.5VL and SmolDocling. This project work helped in understanding the effectiveness of industry specific custom dataset, data pre-processing tasks which are crucial for the training and fine-tuning the model to predict the output with higher accuracy and minimal loss.

# 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 |
| MLLM | Multimodal Large Language Model |
| GPT | Generative Pre-trained Transformer |
| CLIP | Contrastive Language Image Pretraining |
| LSTM | Long Short Term Memory |
| BERT | Bidirectional Encoder Representations from Transformers |
| LLaVa | Large Language and Vision Assistant |
| CLIP | Contrastive Language-Image Pretraining |
| PaLI | Pathways Language and Image model |
| BLIP | Bootstrapped Language-Image Pretraining |

# APPENDICES

## Appendix A: Development Environment Setup & Essential Python Libraries

The development environment setup for project implementation, execution and evaluation of the output results is done using “Google Colab Pro” that has hosted Jupyter Notebook in the cloud and offers compute units for faster processing, priority access to powerful GPUs such as A100 GPU, L4 GPU, T4 GPU, V2-8 TPU etc. and background execution. I have particularly used A100 & T4 GPU’s for the training of the model and execution of the project code.

The dataset of engine logbook is hosted in my BITS-WILP google drive which is accessed by the code running in Colab hosted Jupyter notebook for loading the dataset and pre-processing tasks.

The trained, fine-tuned model and processor model is saved in a cloud repository offered by Hugging Face which is quite similar to Github repository. I have created an account on Hugging face and created private repository on it in order to push the fine-tuned model to the repository, save it and access it when required for evaluation purpose.

The essential python libraries that need to be installed in the development system environment to execute the implemented python code are mentioned below:

|  |  |
| --- | --- |
| **Python Libraries** | **Usage** |
| pip install -q datasets | This library from Hugging Face provides features to load and process public datasets and custom datasets. It also offers efficient data pre-processing and integration with machine learning frameworks. |
| pip install -q sentencepiece | This library is used for tokenization and sub-word unit generation, for text processing tasks. It provides methods to handle raw text directly. |
| pip install -q tensorboard | This is a toolkit which enables to track, visualize metrics, understand training progress and improve model performance by updating the hyperparameters. |
| pip install transformers == 4.45.2 | Transformers library developed and maintained by Hugging Face community provides framework for working with state-of-the-art pre-trained models in various modalities like natural language processing, computer vision, audio, and multimodal tasks. They have built-in capabilities for fine-tuning models on custom datasets and seamless integration with popular machine learning frameworks like PyTorch and TensorFlow. Transformers supports a multitude of tasks, including Text classification, Named entity recognition (NER), Question answering, Text generation, Summarization, Translation |
| pip install sentence-transformers == 3.3.1 | This library provides framework for creating and using sentence, text, and image embeddings which are dense vector representations and used for various NLP tasks such as semantic search, semantic textual similarity and paraphrase mining |
| Pip install torch | PyTorch is a open-source Python library primarily used for deep learning computer vision, natural language processing applications. It has ability to accelerate computation on both CPUs and GPUs facilitating building and training various neural network models. |

## Appendix B: Training Hyperparameters and Fine tuning

We have tried various combinations of training hyperparameters while training the donut model on top of a pre-trained donut base model which significantly impact training performance by influencing how a model learns and generalizes. Appropriate hyperparameter settings lead to improved accuracy, reduced training time, and better generalization on unseen data, while incorrect settings can result in underfitting or overfitting. The training hyperparameters values which are used for fine tuning the donut model is mentioned in Table 1

|  |  |
| --- | --- |
| **Training Parameters** | **Value** |
| num\_train\_epochs | 10 |
| learning\_rate | 1e-5 |
| per\_device\_train\_batch\_size | 2 |
| weight\_decay | 0.01 |
| logging\_steps | 100 |
| predict\_with\_generate | True |

The snippet of the Seq2Seq trainer is given below:

# REFERENCES

1. *Kim, Geewook et al, “OCR-free Document Understanding Transformer”, Computer Vision – ECCV 2022: 17th European Conference.* *https://doi.org/10.48550/arXiv.2111.15664*
2. *Appalaraju, Srikar et al, “Docformer: End-to end transformer for document understanding”, IEEE/CVF International Conference on Computer Vision (ICCV)(October 2021).* [*https://doi.org/10.48550/arXiv.2106.11539*](https://doi.org/10.48550/arXiv.2106.11539)
3. *Mohammad Minouei, Mohammad Reza Soheili, and Didier Stricker. Layout in Text for Document Understanding Using Large Language Models ,Document Analysis and Recognition - ICDAR 2024.*
4. *Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, Furu Wei. Improving Text Embeddings with Large Language Models, ACL 2024.*
5. *Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering, CVPR 2018 IEEE.*
6. *Teakgyu Hong, Donghyun Kim, Mingi Ji, Wonseok Hwang, Daehyun Nam, and Sungrae Park. 2022. BROS: A Pre-Trained Language Model Focusing on Text and Layout for Better Key Information Extraction from Documents, AAAI 2022.*
7. *Kai Hu, Zhuoyuan Wu, Zhuoyao Zhong, Weihong Lin, Lei Sun and Qiang Huo.2023: A Question-Answering Approach to Key Value Pair Extraction from Form-like Document Images, AAAI 2023.*
8. *Bhattacharya, Manojit et al,”**Large language model to multimodal large language model: A journey to shape the biological macromolecules to biological sciences and medicine”,* *Molecular Therapy Nucleic Acids, Volume 35, Issue 3, 102255.*
9. *Liang, Xin Chia et al, “A Comprehensive Survey and Guide to Multimodal Large Language Models in Vision-Language Tasks”* *arXiv:2411.06284.* [*https://doi.org/10.48550/arXiv.2411.06284*](https://doi.org/10.48550/arXiv.2411.06284)
10. *Vaswani, Ashish et al, “Attention Is All You Need”, Advances in Neural Information Processing Systems 30 (NIPS 2017).* [*https://doi.org/10.48550/arXiv.1706.03762*](https://doi.org/10.48550/arXiv.1706.03762)
11. *Devlin, Jacob et al, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, Conference of the North American Chapter of the Association for Computational Linguistics, 2018.* [*https://doi.org/10.48550/arXiv.1810.04805*](https://doi.org/10.48550/arXiv.1810.04805)
12. *Feyisa, Degaga et al, “The future of document indexing: GPT and Donut revolutionize table of content processing”, 2024. https://doi.org/10.48550/arXiv.2403.07553*