Hybrid Deep Learning MRI Classification Using DenseNet201, EfficientNetB2, and Vision Transformer for Early Detection of Alzheimer

1. Introduction

This document describes the system requirements, software, hardware, and step-by-step configuration for the hybrid deep learning model developed for MRI classification. The goal of this model is to integrate DenseNet201, EfficientNetB2, and Vision Transformer to enhance classification accuracy by leveraging spatial, mid-level, and global features.

2. System Configuration

2.1 Software Specification

- Operating System: Windows 10/11 or Ubuntu 20.04+
- A Gmail account to access data uploaded to google drive.
- Google Colab for model training and evaluation using GPU support
- Cloud GPU ,Tesla T4 GPU with 16 GB VRAM (Google Colab Pro)

- **Libraries and Frameworks:**
  - TensorFlow 2.9
  - PyTorch 1.11 (for Vision Transformer)
  - Scikit-learn, NumPy, Pandas for data preprocessing and evaluation
  - Matplotlib, Seaborn for visualization
  - ImageNet Pretrained Models: DenseNet201 and EfficientNetB2
  - Hugging Face Transformers for Vision Transformer

2.2 Hardware Specification

- **Minimum Requirements:**
  - CPU: Intel Core i5 or equivalent
  - RAM: 8GB
  - GPU: NVIDIA GTX 1050 with 4GB VRAM
- **Recommended Requirements:**
  - CPU: Intel Core i7 or AMD Ryzen 7
  - RAM: 16GB or higher
  - GPU: NVIDIA RTX 3060 with 8GB VRAM or higher

3. Software Installation

Step 1 Create a Gmail account as shown below, and proceed to fill in the prompted requirements

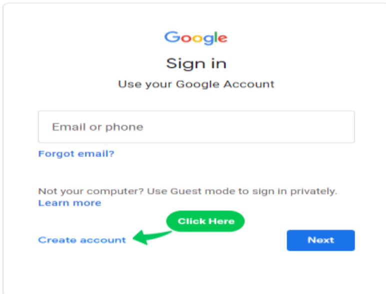


Figure 1: How to create gmail account

Step 2 After Successful account creation, On your browser open Google Colab

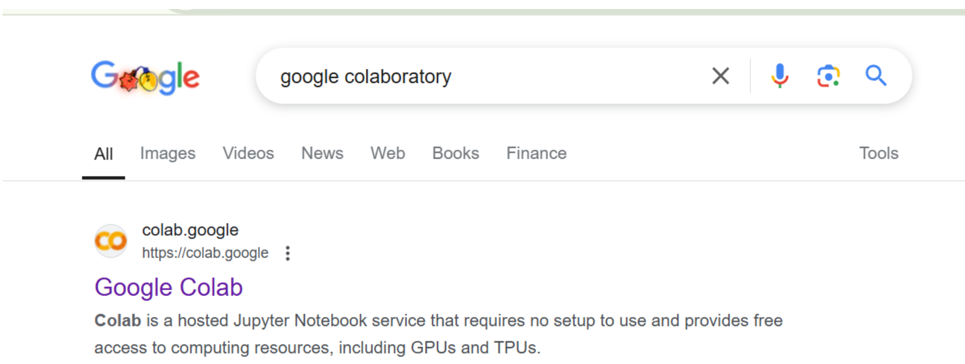


Figure 2: How to access Colab

Step 3. Open Google Colab and Subscribe to Pro to access T4 GPU with 16 GB VRAM

I. On settings tab, click on colab pro as shown

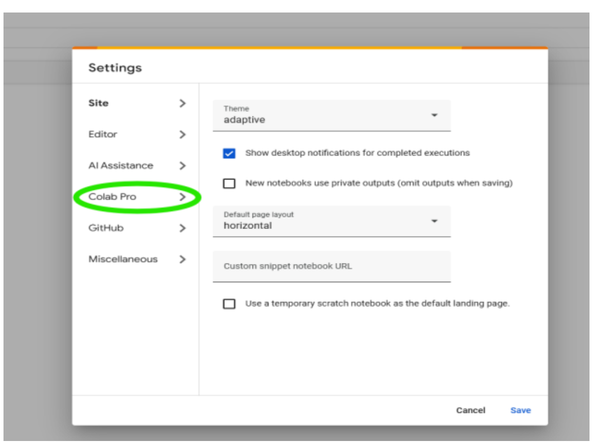


Figure 4: Colab Pro

II. Subscribe to Colab Pro as   highlighted in figure() below

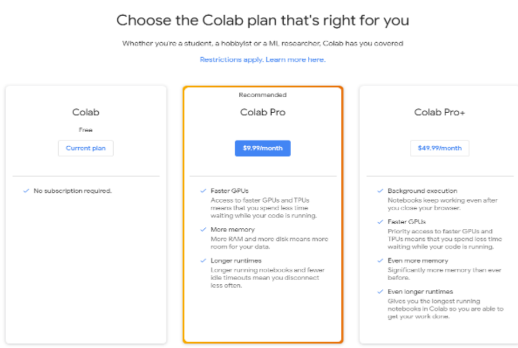


Figure 5: Subscribe to Colab Pro

Iii Verify Subscription

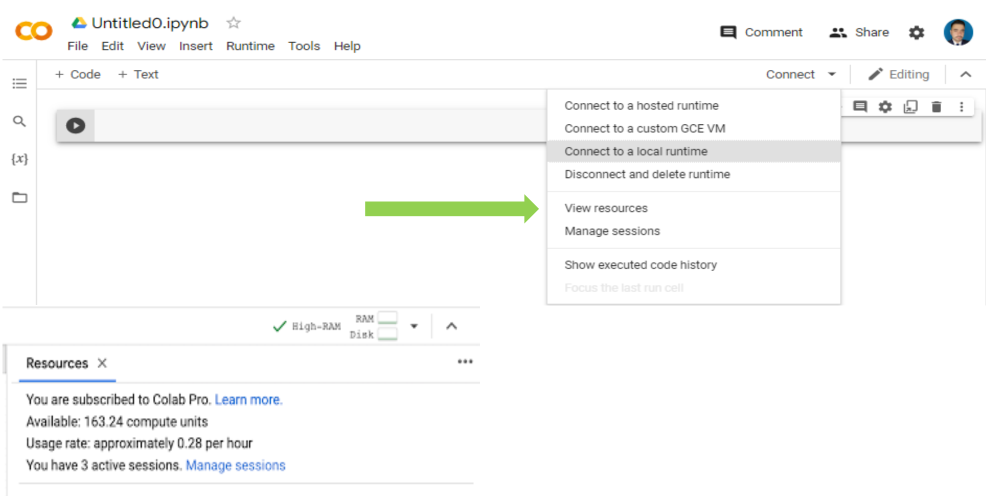


Figure 6: Verification of subscription

4.Software Configurations

To configure the T4 GPU on google colab

Step1. Select Change run type on the drop-down menu as illustrated

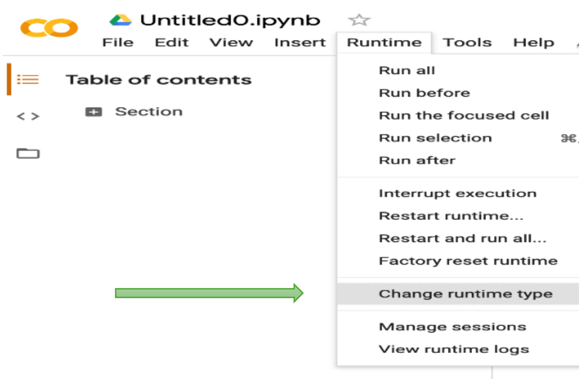
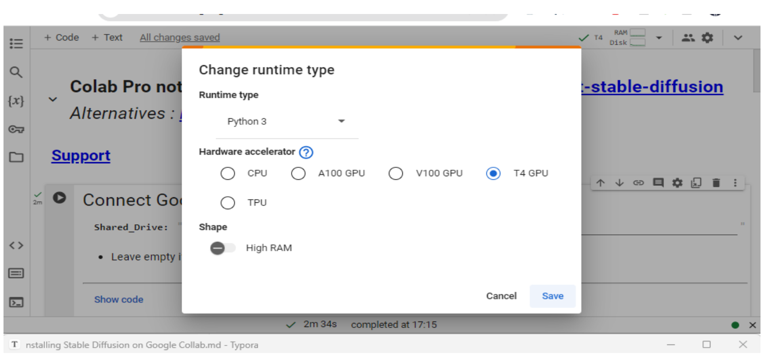


Figure 7: Set runtime

Step 2 click on T4 GPU



Project Development

Install Required Libraries

```
import os
import zipfile
import random
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.utils.class_weight import compute_class_weight
from tensorflow.keras.applications import DenseNet201, EfficientNetB2
from tensorflow.keras.layers import Input, GlobalAveragePooling2D, Dense, Dropout, Concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from transformers import ViTFeatureExtractor, ViTModel
from tensorflow.keras.utils import Sequence
from PIL import Image
import torch
```

Figure 9: Code showing how to import necessary libraries

Data Extraction

Step2; Extract the image file paths from the zip file

```
RANDOM_STATE = 42
ZIP_FILE = "cnn.zip"  # Zip file
EXTRACTION_PATH = "OriginalDataset"  # Path where the dataset will be extracted
```

```
[ ] # Extract Dataset
    if not os.path.exists(EXTRACTION_PATH):
        with zipfile.ZipFile(ZIP_FILE, "r") as zip_ref:
            zip_ref.extractall(EXTRACTION_PATH)
    print(f"Dataset extracted to: {EXTRACTION_PATH}")

    Dataset extracted to: OriginalDataset
```

Figure 10: Extraction of filepaths from zipped file

Modelling

Step1: Initialize Pretrained models; Load DenseNet201 and EfficientNetB2 from TensorFlow's applications module. Load Vision Transformer from Hugging Face.

```
[ ] # Initialize Pretrained Models
    densenet_base = DenseNet201(weights="imagenet", include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))
    efficientnet_base = EfficientNetB2(weights="imagenet", include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))
    vit_feature_extractor = ViTFeatureExtractor.from_pretrained("google/vit-base-patch16-224-in21k")
    vit_model = ViTModel.from_pretrained("google/vit-base-patch16-224-in21k")

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet201_weights_tf_dim_ordering_tf_kernels_notop.h5
    74836368/74836368 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
    Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb2_notop.h5
    31790344/31790344 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
    /usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
    The secret `HF_TOKEN` does not exist in your Colab secrets.
    To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
    You will be able to reuse this secret in all of your notebooks.
    Please note that authentication is recommended but still optional to access public models or datasets.
      warnings.warn(
    preprocessor_config.json: 100% 160/160 [00:00<00:00, 13.9kB/s]
    /usr/local/lib/python3.10/dist-packages/transformers/models/vit/feature_extraction_vit.py:28: FutureWarning: The class ViTFeatureExtractor is deprecated and will be removed in version 5 of Transformers. Please use ViTImageProcessor i
      warnings.warn(
    config.json: 100% 502/502 [00:00<00:00, 42.3kB/s]
    model.safetensors: 100% 346M/346M [00:01<00:00, 230MB/s]
```

Figure 11: Code Showing initializing pretrained base models

Create Model

Step 1 : Define and verify the full hybrid model

```
[13] hybrid_model = Model(inputs=[densenet_input, efficientnet_input, vit_input], outputs=output)
     # Print model summary
     hybrid_model.summary()
```

Figure 12: Code Showing Hybrid Model Definition

Training

Step 1; Phase 1: Freeze pre-trained layers, train only dense layers.

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
# Optimizer and learning rate scheduler
optimizer = Adam(learning_rate=1e-5)
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True, verbose=1)
# Freeze pretrained layers
for layer in densenet_base.layers:
    layer.trainable = False
for layer in efficientnet_base.layers:
    layer.trainable = False

# Compile the model
hybrid_model.compile(optimizer=Adam(learning_rate=1e-4), loss="categorical_crossentropy", metrics=["accuracy"])

# Train for a few epochs
hybrid_model.fit(train_generator, validation_data=val_generator, epochs=5, class_weight=class_weights_dict)
```

Step 2:Phase 2: Unfreeze pre-trained layers, fine-tune entire model.

```
# Unfreeze and fine-tune
for layer in densenet_base.layers:
    layer.trainable = True
for layer in efficientnet_base.layers:
    layer.trainable = True

# Compile again with a lower learning rate
hybrid_model.compile(optimizer=Adam(learning_rate=1e-6), loss="categorical_crossentropy", metrics=["accuracy"])

# Train again
hybrid_model.fit(train_generator, validation_data=val_generator, epochs=10, class_weight=class_weights_dict)
```

Save the hybrid model

```
# Save the trained model
hybrid_model.save("hybrid_model.h5")
print("Model saved as hybrid_model.h5")

```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered lega
Model saved as hybrid_model.h5
```

Evaluation

Step1: Generate the Classification Report with Precision, recall, F1-score

```
report = classification_report(y_true, y_pred_classes, target_names=label_encoder.classes_)
print(report)
```

Step 2: Generate Predictions

```python
!pip install scikit-learn tensorflow pandas

import numpy as np
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.models import load_model
import pandas as pd

# Load your hybrid model
loaded_model = load_model("hybrid_model.h5")

# Extract image paths and labels from the DataFrame
test_image_paths = test_df["filepaths"]
test_labels = test_df["encoded_labels"]
# 1. Preprocess and predict on all test images
predictions = []
true_labels = []

for image_path, true_label in zip(test_image_paths, test_labels):
    preprocessed_data = preprocess_single_image(image_path)
    prediction = loaded_model.predict(preprocessed_data)
    predicted_class = np.argmax(prediction)

    predictions.append(predicted_class)
    true_labels.append(true_label)

# 2. Convert predictions and true labels to NumPy arrays
predictions = np.array(predictions)
true_labels = np.array(true_labels)
```

Step 2: Confusion Matrix

```python
cm = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

Troubleshooting

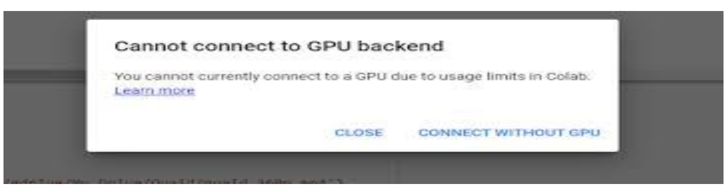


Figure 16: Error Alert

Possible Cause and Solution

Cause	Solution
GPU Quota Limit Reached	- Upgrade to Colab Pro or Pro+ for extended GPU limits.
	- Reduce GPU usage by optimizing batch sizes or clearing caches during training.
High Server Load	- Wait for 1–2 hours and retry connecting to the GPU backend.
	- Switch to a different runtime (e.g., TPU or CPU) temporarily.
Connectivity Issues	- Check your internet connection and ensure it is stable.
	- Restart the runtime via **Runtime > Manage Sessions**.