UNIVERSITAS
GADJAH MADA

Classifying Medical Images for Pneumonia with CNN-Transformer Hybrid Deep Learning Model

Sasha Annabel
22/496780/PA/21361

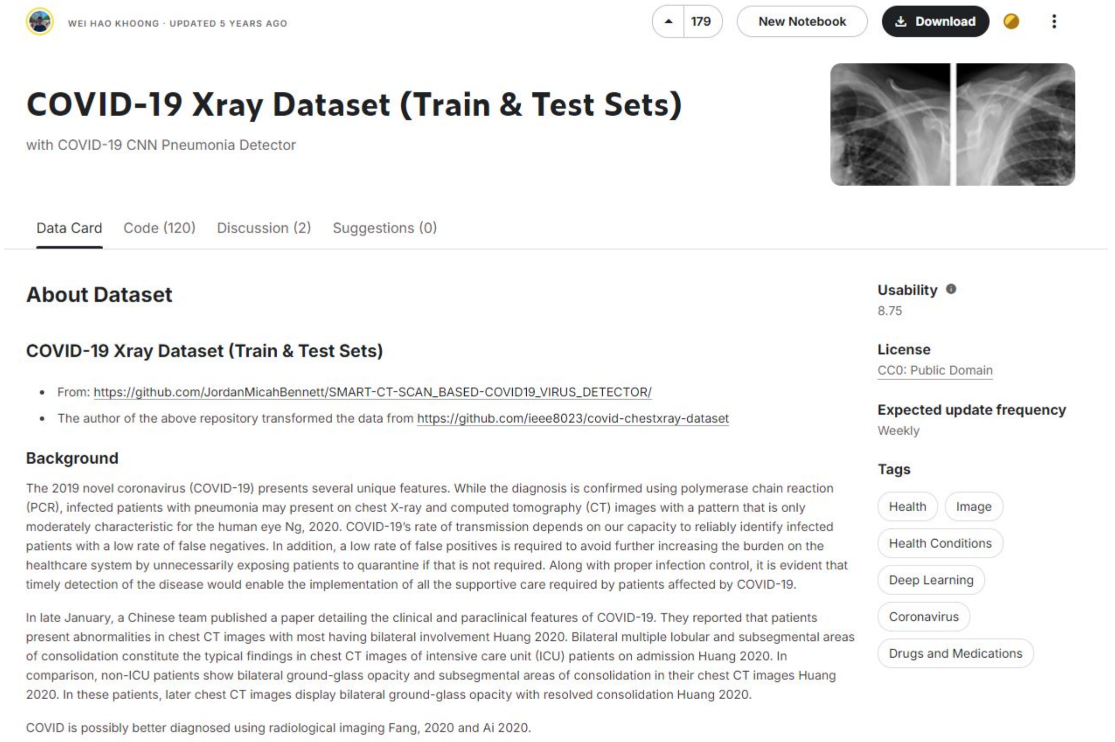
Department of Computer Science and Electronics
Universitas Gadjah Mada

Problem Statement

The case of **medical imaging for pneumonia detection**, especially in the context of COVID-19, is highly relevant due to the significant impact pneumonia has on patient outcomes. COVID-19 frequently leads to viral pneumonia, which can rapidly worsen without early and accurate detection. This has created a **demand for reliable diagnostic tools that can help healthcare providers quickly assess and diagnose pneumonia, particularly in high-risk patients.** Chest X-rays and CT scans are critical in this context, as they offer a non-invasive way to visualize lung inflammation and identify early signs of pneumonia.

Furthermore, the COVID-19 pandemic accelerated the development of AI tools to support overwhelmed healthcare systems by providing faster, automated analysis of medical images. Using a hybrid model that combines Convolutional Neural Networks (CNNs) and transformers can leverage the strengths of both architectures: CNNs for their spatial feature extraction and transformers for their ability to capture long-range dependencies in image data. This approach holds promise for improving the accuracy of pneumonia detection and classification, potentially leading to faster interventions and better patient outcomes.

Sample Dataset



COVID-19 Xray Dataset (Train & Test Sets)

with COVID-19 CNN Pneumonia Detector

About Dataset

COVID-19 Xray Dataset (Train & Test Sets)

- From: https://github.com/JordanMicahBennett/SMART-CT-SCAN_BASED-COVID19_VIRUS_DETECTOR/
- The author of the above repository transformed the data from https://github.com/ieee8023/covid-chestxray-dataset

Background

The 2019 novel coronavirus (COVID-19) presents several unique features. While the diagnosis is confirmed using polymerase chain reaction (PCR), infected patients with pneumonia may present on chest X-ray and computed tomography (CT) images with a pattern that is only moderately characteristic for the human eye Ng, 2020. COVID-19's rate of transmission depends on our capacity to reliably identify infected patients with a low rate of false negatives. In addition, a low rate of false positives is required to avoid further increasing the burden on the healthcare system by unnecessarily exposing patients to quarantine if that is not required. Along with proper infection control, it is evident that timely detection of the disease would enable the implementation of all the supportive care required by patients affected by COVID-19.

In late January, a Chinese team published a paper detailing the clinical and paraclinical features of COVID-19. They reported that patients present abnormalities in chest CT images with most having bilateral involvement Huang 2020. Bilateral multiple lobular and subsegmental areas of consolidation constitute the typical findings in chest CT images of intensive care unit (ICU) patients on admission Huang 2020. In comparison, non-ICU patients show bilateral ground-glass opacity and subsegmental areas of consolidation in their chest CT images Huang 2020. In these patients, later chest CT images display bilateral ground-glass opacity with resolved consolidation Huang 2020.

COVID is possibly better diagnosed using radiological imaging Fang, 2020 and Ai 2020.

Usability
8.75

License
CC0: Public Domain

Expected update frequency
Weekly

Tags
Health · Image · Health Conditions · Deep Learning · Coronavirus · Drugs and Medications

The dataset is taken from Kaggle, and is focused on ***detecting pneumonia disease***, particularly pneumonia caused by COVID-19 infection.

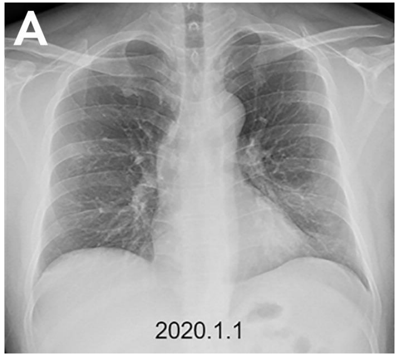
The data is separated into train/test sets, all in the form of medical images of chest X-rays.

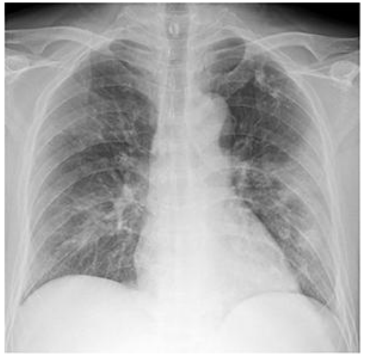
The 2 labels in the dataset :

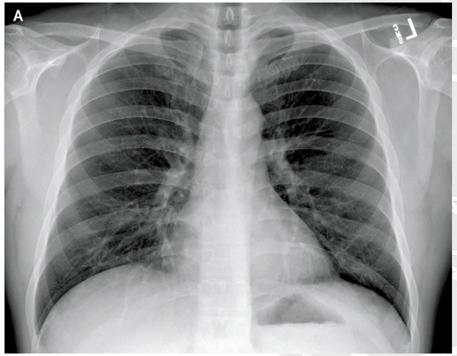
- **PNEUMONIA** : X-rays that show lung abnormalities consistent with pneumonia (both COVID-19 related or otherwise).
- **NORMAL** : X-rays without signs of pneumonia.

Sample Dataset (Pictures)

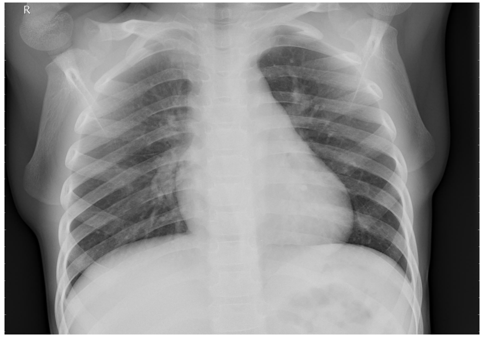
PNEUMONIA

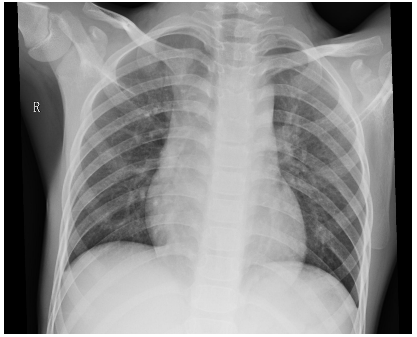


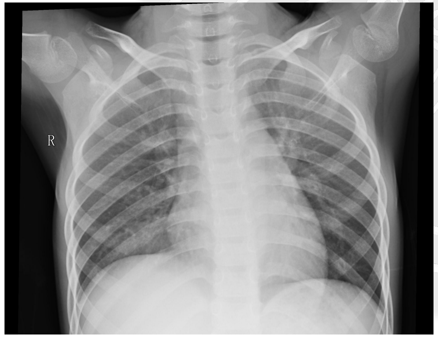




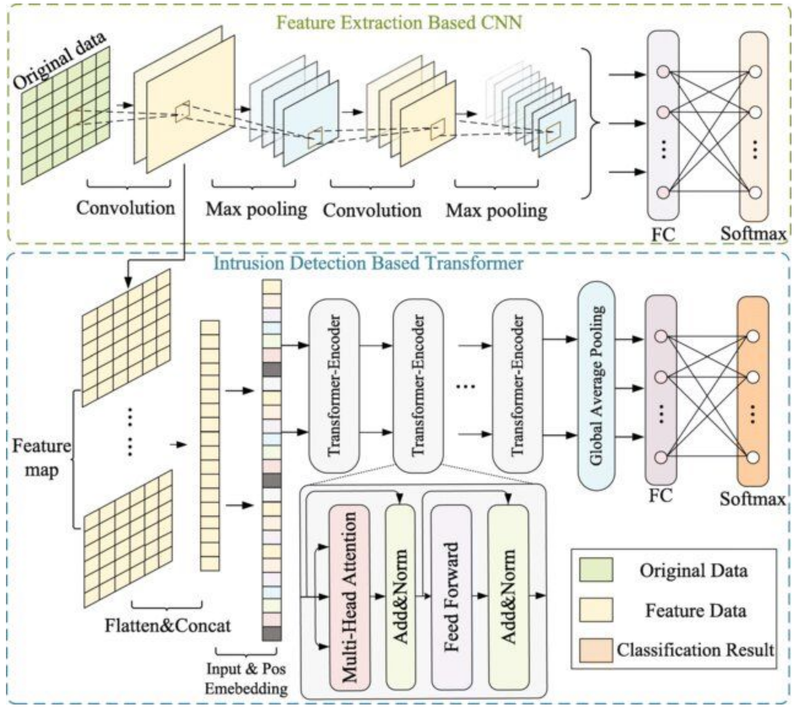
NORMAL







Proposed Deep Learning Architecture



*PLEASE NOTE THAT THE MODEL IN THIS PRESENTATION IS **A SIMPLIFIED VERSION** OF THE ILLUSTRATION*

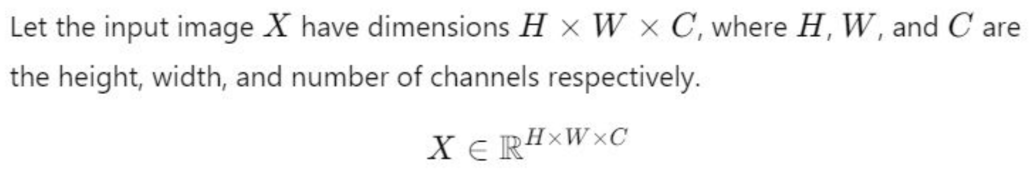
The illustration represents the model proposed for this pneumonia detection project. Here, CNN is for initial feature extraction of the images, and Transformer connects the relationship (feature map) between different feature aspects.

This is because CNN is good at capturing details / patterns, while Transformer has the ability to extract their interconnected meaning.

Proposed Deep Learning Architecture

Input Layer

The model expects images of shape (224, 224, 3), meaning the input size is a 224x224 pixel image with 3 color channels (RGB). This layer simply receives the data and passes it to the CNN for feature extraction.



Let the input image X have dimensions $H \times W \times C$, where H, W, and C are the height, width, and number of channels respectively.

$$X \in \mathbb{R}^{H \times W \times C}$$

Proposed Deep Learning Architecture

First Convolution + ReLU Activation + Max Pooling:

$$Z_1 = \mathrm{ReLU}(\mathrm{Conv2D}(X, W_1, b_1)), \quad Z_1 \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times 32}$$

$$P_1 = \mathrm{MaxPool}(Z_1), \quad P_1 \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times 32}$$

Second Convolution + ReLU Activation + Max Pooling:

$$Z_2 = \mathrm{ReLU}(\mathrm{Conv2D}(P_1, W_2, b_2)), \quad Z_2 \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4} \times 64}$$

$$P_2 = \mathrm{MaxPool}(Z_2), \quad P_2 \in \mathbb{R}^{\frac{H}{8} \times \frac{W}{8} \times 64}$$

Third Convolution + ReLU Activation + Max Pooling:

$$Z_3 = \mathrm{ReLU}(\mathrm{Conv2D}(P_2, W_3, b_3)), \quad Z_3 \in \mathbb{R}^{\frac{H}{8} \times \frac{W}{8} \times 128}$$

$$P_3 = \mathrm{MaxPool}(Z_3), \quad P_3 \in \mathbb{R}^{\frac{H}{16} \times \frac{W}{16} \times 128}$$

CNN Layer

Extracts relevant visual features from images.

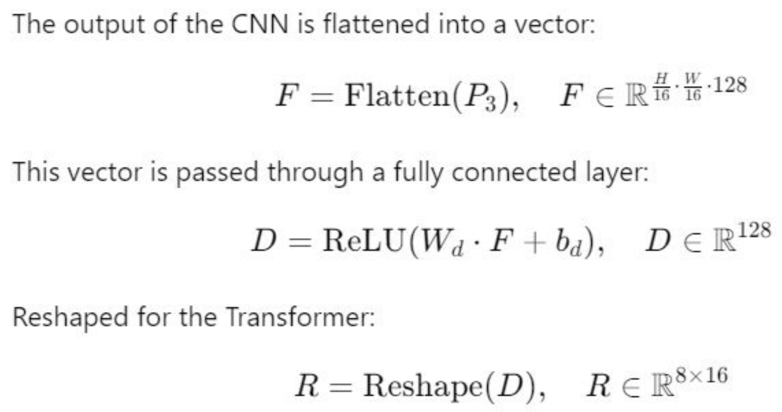
- ***Convolutional Layers*** : detect various features in the images (edges, textures, patterns, etc.)
    - Conv2D(32, (3, 3)), Conv2D(64, (3, 3)), Conv2D(128, (3, 3))
- ***Max Pooling Layers*** : reduces spatial dimensions and summarizes most important information from each region (focusing on prominent features).
- ***Flatten Layer*** : output from CNN layers is flattened (reshaped) to a 1-D vector so it can be passed to the fully connected (dense) layer, Transformer layer.
- ***Dense Layer*** (128 units) : this layer learns a compact representation of the features extracted by the CNN. It also helps adjust the features to a shape suitable for the Transformer layer.

Proposed Deep Learning Architecture

Reshaping

Reshape the 128-dimensional output into an (8, 16) tensor to allow the Transformer to interpret each sequence element as a separate "token" with a feature vector, much like words in a sentence, where:

- 8 represents the sequence length (or the number of "tokens" the Transformer will process),
- 16 is the dimension of each "token" (the feature representation).



The output of the CNN is flattened into a vector:

$$F = \mathrm{Flatten}(P_3), \quad F \in \mathbb{R}^{\frac{H}{16} \cdot \frac{W}{16} \cdot 128}$$

This vector is passed through a fully connected layer:

$$D = \mathrm{ReLU}(W_d \cdot F + b_d), \quad D \in \mathbb{R}^{128}$$

Reshaped for the Transformer:

$$R = \mathrm{Reshape}(D), \quad R \in \mathbb{R}^{8 \times 16}$$

Proposed Deep Learning Architecture

Multi-Head Attention: Let $Q = K = V = R$ (self-attention):

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

With multiple heads:

$$MHA(R) = \text{Concat}(\text{head}_1, \text{head}_2)W_o$$

Where each head computes:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Global Average Pooling:

$$G = \text{GlobalAveragePooling}(MHA(R)), \quad G \in \mathbb{R}^{16}$$

Transformer Encoder Layer (Attention Mechanism)

The Transformer layer is crucial for analyzing relationships between different regions of the image.

- ***Multi-Head Attention*** : focuses on different parts of the sequence simultaneously. Each "head" learns a different set of relationships between the "tokens" in the sequence, which allows the model to understand spatial dependencies in the feature map. Each head computes attention weights over the feature tokens, learning what parts of the input are important when making a classification.
- ***Global Average Pooling (1D)*** : to summarize all tokens into a single feature vector, giving a compact, fixed-size representation of the sequence before getting passed to dense layers for classification.

Proposed Deep Learning Architecture

Fully connected layer:

$$FC_1 = \mathrm{ReLU}(W_4 \cdot G + b_4), \quad FC_1 \in \mathbb{R}^{64}$$

Dropout (randomly zeroing some weights):

$$FC_1' = \mathrm{Dropout}(FC_1)$$

Output layer (sigmoid activation):

$$\hat{y} = \sigma(W_5 \cdot FC_1' + b_5), \quad \hat{y} \in [0, 1]$$

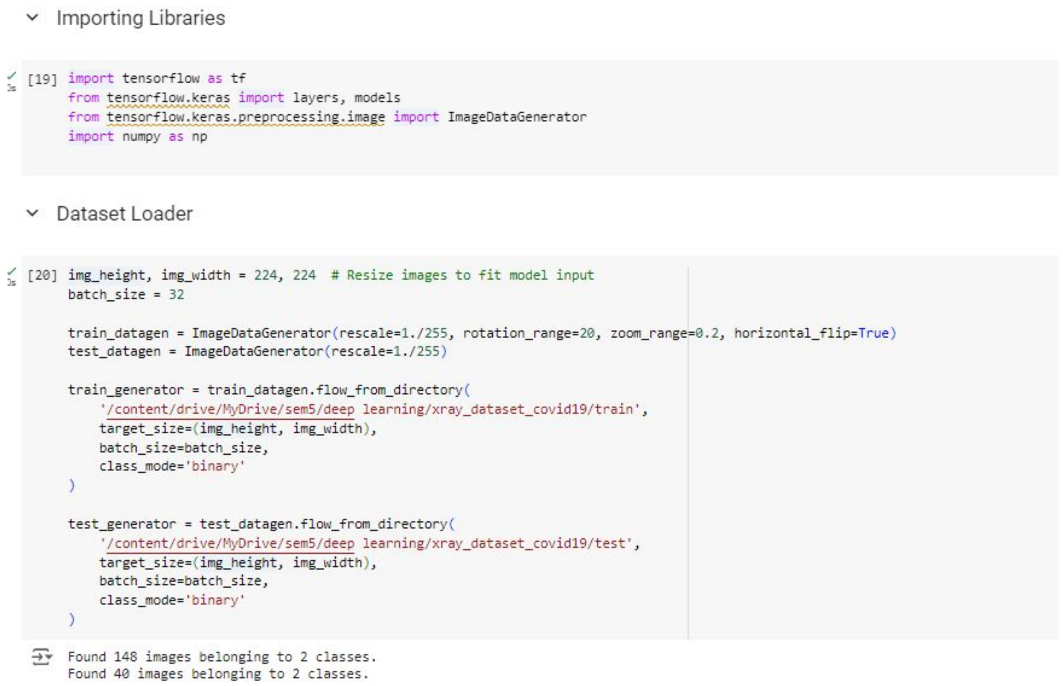
Fully Connected Layers

For classification task.

- ***Dense Layer (64 units)*** : provides an additional transformation to the pooled features from the Transformer. Also has a relu activation function, allowing the network to learn complex patterns in the final feature representation.
- ***Dropout Layer*** : 50% dropout rate helps prevent overfitting by randomly "dropping" nodes during training, forcing the network to learn more robust features.
- ***Output Layer*** (1 unit, sigmoid activation) : the final layer, with a single unit and sigmoid activation, that outputs a probability indicating whether the image shows pneumonia (1) or not (0).

Python Code for Model Creation

PART 1 : Importing libraries and dataset.



Importing Libraries

```
[19] import tensorflow as tf
     from tensorflow.keras import layers, models
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     import numpy as np
```

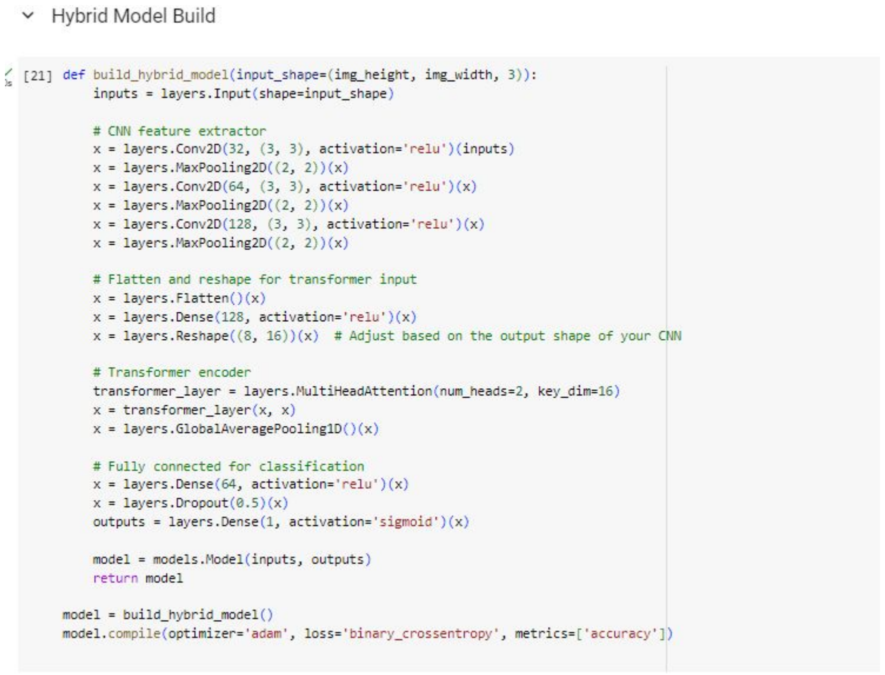
Dataset Loader

```
[20] img_height, img_width = 224, 224  # Resize images to fit model input
     batch_size = 32

     train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20, zoom_range=0.2, horizontal_flip=True)
     test_datagen = ImageDataGenerator(rescale=1./255)

     train_generator = train_datagen.flow_from_directory(
         '/content/drive/MyDrive/sem5/deep learning/xray_dataset_covid19/train',
         target_size=(img_height, img_width),
         batch_size=batch_size,
         class_mode='binary'
     )

     test_generator = test_datagen.flow_from_directory(
         '/content/drive/MyDrive/sem5/deep learning/xray_dataset_covid19/test',
         target_size=(img_height, img_width),
         batch_size=batch_size,
         class_mode='binary'
     )
```

```
Found 148 images belonging to 2 classes.
Found 40 images belonging to 2 classes.
```

Python Code for Model Creation

PART 2 : Hybrid Deep Learning Model combining CNN and Transformer.

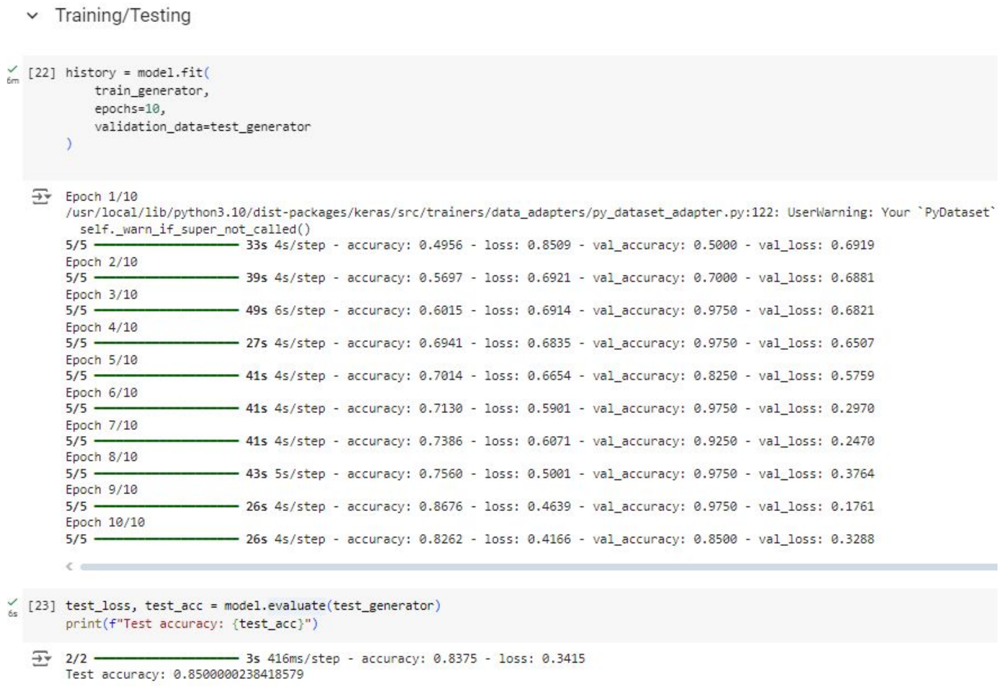


Hybrid Model Build

```
[21] def build_hybrid_model(input_shape=(img_height, img_width, 3)):
         inputs = layers.Input(shape=input_shape)

         # CNN feature extractor
         x = layers.Conv2D(32, (3, 3), activation='relu')(inputs)
         x = layers.MaxPooling2D((2, 2))(x)
         x = layers.Conv2D(64, (3, 3), activation='relu')(x)
         x = layers.MaxPooling2D((2, 2))(x)
         x = layers.Conv2D(128, (3, 3), activation='relu')(x)
         x = layers.MaxPooling2D((2, 2))(x)

         # Flatten and reshape for transformer input
         x = layers.Flatten()(x)
         x = layers.Dense(128, activation='relu')(x)
         x = layers.Reshape((8, 16))(x)  # Adjust based on the output shape of your CNN

         # Transformer encoder
         transformer_layer = layers.MultiHeadAttention(num_heads=2, key_dim=16)
         x = transformer_layer(x, x)
         x = layers.GlobalAveragePooling1D()(x)

         # Fully connected for classification
         x = layers.Dense(64, activation='relu')(x)
         x = layers.Dropout(0.5)(x)
         outputs = layers.Dense(1, activation='sigmoid')(x)

         model = models.Model(inputs, outputs)
         return model

     model = build_hybrid_model()
     model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Python Code for Model Creation

PART 3 : Model training and testing processes, along with accuracy.



Training/Testing

```
[22] history = model.fit(
         train_generator,
         epochs=10,
         validation_data=test_generator
     )
```

```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset`
  self._warn_if_super_not_called()
5/5 ━━━━━━━━━━━━━━━━━━━━ 33s 4s/step - accuracy: 0.4956 - loss: 0.8509 - val_accuracy: 0.5000 - val_loss: 0.6919
Epoch 2/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 39s 4s/step - accuracy: 0.5697 - loss: 0.6921 - val_accuracy: 0.7000 - val_loss: 0.6881
Epoch 3/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 49s 6s/step - accuracy: 0.6015 - loss: 0.6914 - val_accuracy: 0.9750 - val_loss: 0.6821
Epoch 4/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 27s 4s/step - accuracy: 0.6941 - loss: 0.6835 - val_accuracy: 0.9750 - val_loss: 0.6507
Epoch 5/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 41s 4s/step - accuracy: 0.7014 - loss: 0.6654 - val_accuracy: 0.8250 - val_loss: 0.5759
Epoch 6/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 41s 4s/step - accuracy: 0.7130 - loss: 0.5901 - val_accuracy: 0.9750 - val_loss: 0.2970
Epoch 7/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 41s 4s/step - accuracy: 0.7386 - loss: 0.6071 - val_accuracy: 0.9250 - val_loss: 0.2470
Epoch 8/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 43s 5s/step - accuracy: 0.7560 - loss: 0.5001 - val_accuracy: 0.9750 - val_loss: 0.3764
Epoch 9/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 26s 4s/step - accuracy: 0.8676 - loss: 0.4639 - val_accuracy: 0.9750 - val_loss: 0.1761
Epoch 10/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 26s 4s/step - accuracy: 0.8262 - loss: 0.4166 - val_accuracy: 0.8500 - val_loss: 0.3288
```

```
[23] test_loss, test_acc = model.evaluate(test_generator)
     print(f"Test accuracy: {test_acc}")
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 3s 416ms/step - accuracy: 0.8375 - loss: 0.3415
Test accuracy: 0.8500000238418579
```

The model's test dataset (new data) accuracy is 85%

Link to Code Platform

The code on CNN+Transformer Model for this project can be found at :

https://colab.research.google.com/drive/1Oryb0diXAUOrbfEmS2VOtUa2Op59l7Ft?usp=sharing

UNIVERSITAS
GADJAH MADA

Thank you for your attention.