

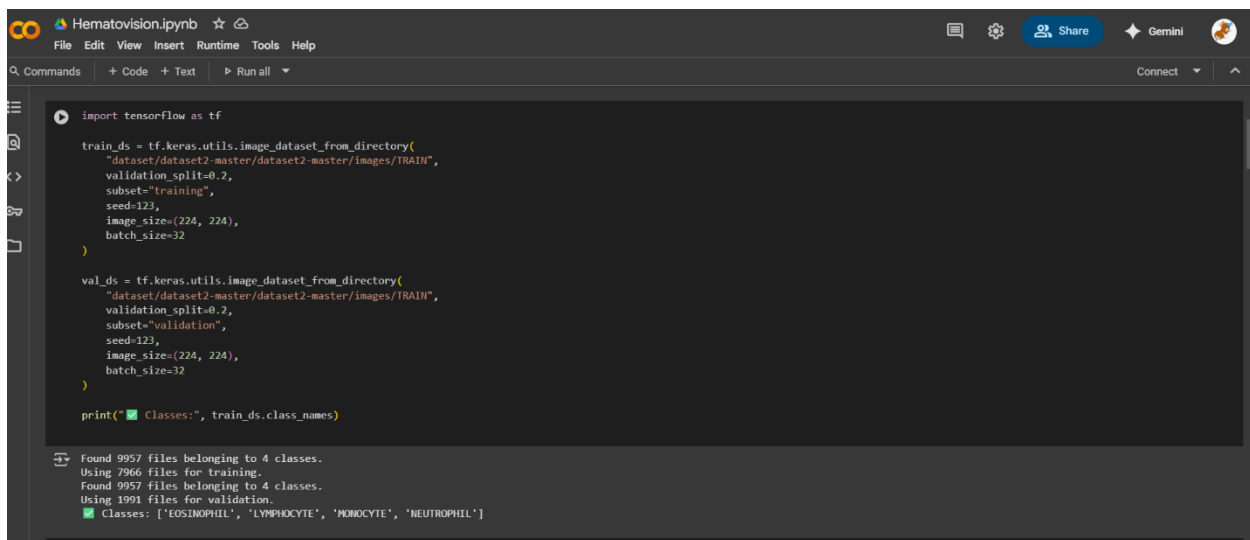
## Project Development Phase

### Model Performance Test

Date	29 June 2025
Team ID	LTVIP2025TMID46471
Project Name	Hematovision : advanced blood cell classification using transfer learning
Maximum Marks	

### Model Performance Testing:

## 1.Data rendered



```
import tensorflow as tf

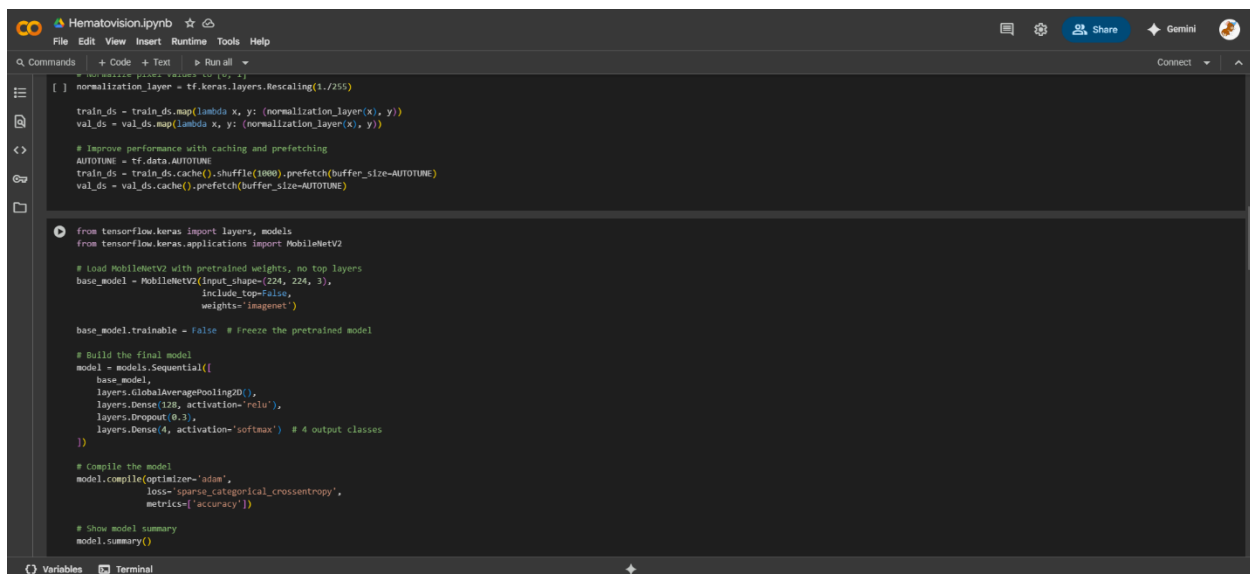
train_ds = tf.keras.utils.image_dataset_from_directory(
    "dataset/dataset2-master/dataset2-master/images/TRAIN",
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(224, 224),
    batch_size=32
)

val_ds = tf.keras.utils.image_dataset_from_directory(
    "dataset/dataset2-master/dataset2-master/images/TRAIN",
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(224, 224),
    batch_size=32
)

print("Classes:", train_ds.class_names)
```

Found 9957 files belonging to 4 classes.  
Using 7966 files for training.  
Found 9957 files belonging to 4 classes.  
Using 1991 files for validation.  
Classes: ['EOSINOPHIL', 'LYMPHOCYTE', 'MONOCYTE', 'NEUTROPHIL']

## 2.Data processing



```
[ ] normalization_layer = tf.keras.layers.Rescaling(1./255)

train_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
val_ds = val_ds.map(lambda x, y: (normalization_layer(x), y))

# improve performance with caching and prefetching
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

from tensorflow.keras import layers, models
from tensorflow.keras.applications import MobileNetV2

# Load MobileNetV2 with pretrained weights, no top layers
base_model = MobileNetV2(input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet')

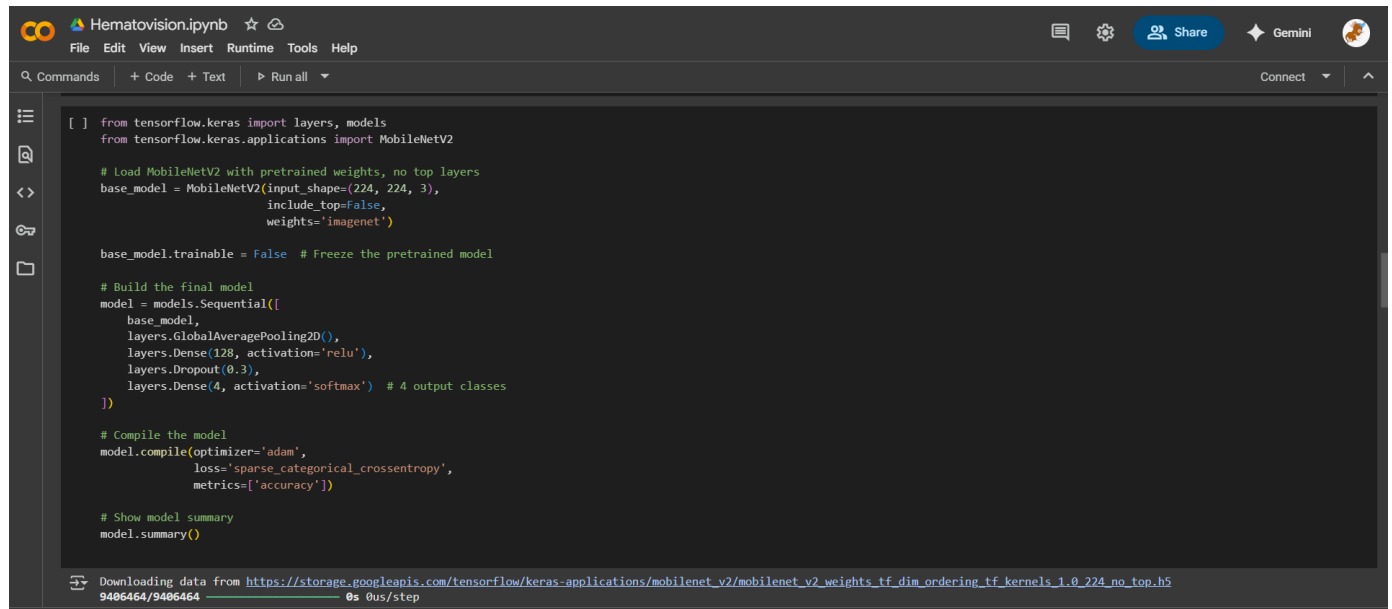
base_model.trainable = False # Freeze the pretrained model

# Build the final model
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(4, activation='softmax') # 4 output classes
])

# Compile the model
model.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])

# Show model summary
model.summary()
```

### 3. Utilization of Filters



The screenshot shows a Jupyter Notebook titled 'Hematovision.ipynb'. The code defines a MobileNetV2 model with pre-trained weights, freezes the base model, and builds a final model with Global Average Pooling, Dense layers, and Dropout. It then compiles the model with Adam optimizer and sparse categorical crossentropy loss, and shows the model summary.

```
[ ] from tensorflow.keras import layers, models
    from tensorflow.keras.applications import MobileNetV2

    # Load MobileNetV2 with pretrained weights, no top layers
    base_model = MobileNetV2(input_shape=(224, 224, 3),
                             include_top=False,
                             weights='imagenet')

    base_model.trainable = False # Freeze the pretrained model

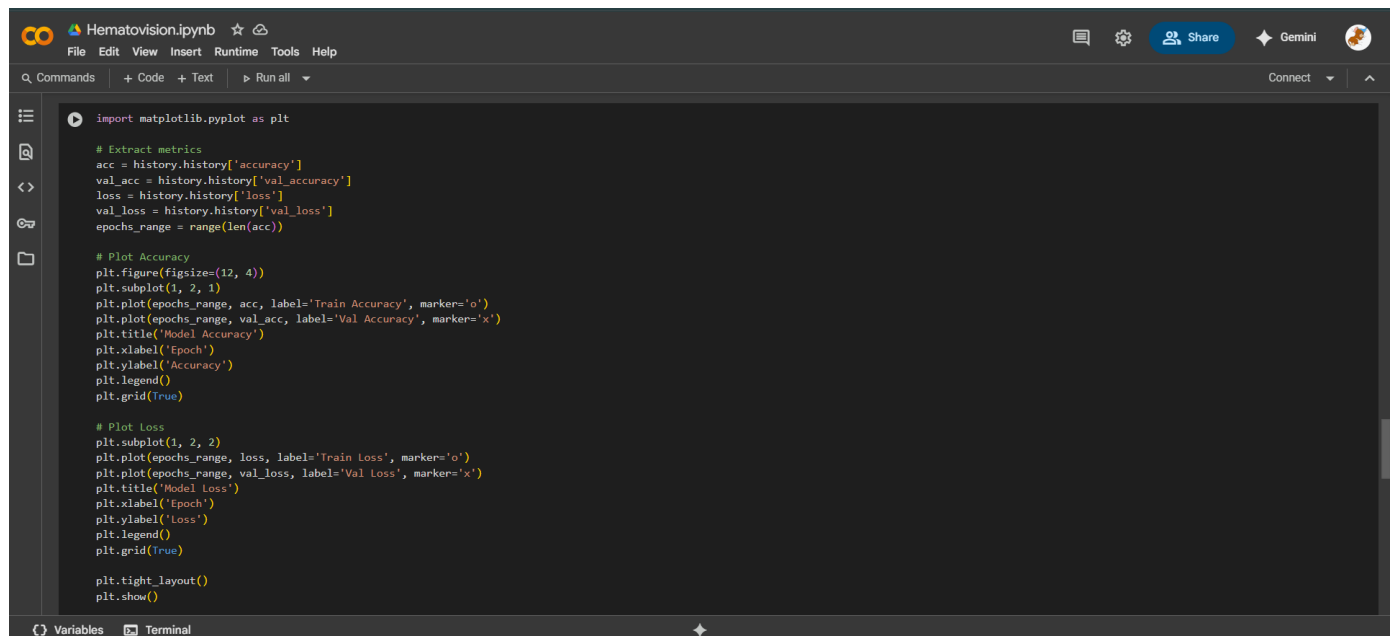
    # Build the final model
    model = models.Sequential([
        base_model,
        layers.GlobalAveragePooling2D(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.3),
        layers.Dense(4, activation='softmax') # 4 output classes
    ])

    # Compile the model
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    # Show model summary
    model.summary()
```

Downloading data from [https://storage.googleapis.com/tensorflow/keras-applications/mobilenet\\_v2/mobilenet\\_v2\\_weights\\_tf\\_dim\\_ordering\\_tf\\_kernels\\_1.0\\_224\\_no\\_top.h5](https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5)  
9486464/9486464 0s 0us/step

### 4. Calculation Fields Used



The screenshot shows a Jupyter Notebook titled 'Hematovision.ipynb'. The code imports matplotlib.pyplot as plt, extracts accuracy and loss metrics from the model history, and plots them. It creates two subplots: one for accuracy (Train Accuracy and Val Accuracy) and one for loss (Train Loss and Val Loss). The plots are titled 'Model Accuracy' and 'Model Loss' respectively, and the x-axis is labeled 'Epoch'.

```
import matplotlib.pyplot as plt

# Extract metrics
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(len(acc))

# Plot Accuracy
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Train Accuracy', marker='o')
plt.plot(epochs_range, val_acc, label='Val Accuracy', marker='x')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)

# Plot Loss
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Train Loss', marker='o')
plt.plot(epochs_range, val_loss, label='Val Loss', marker='x')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

## 5. Model Evaluation

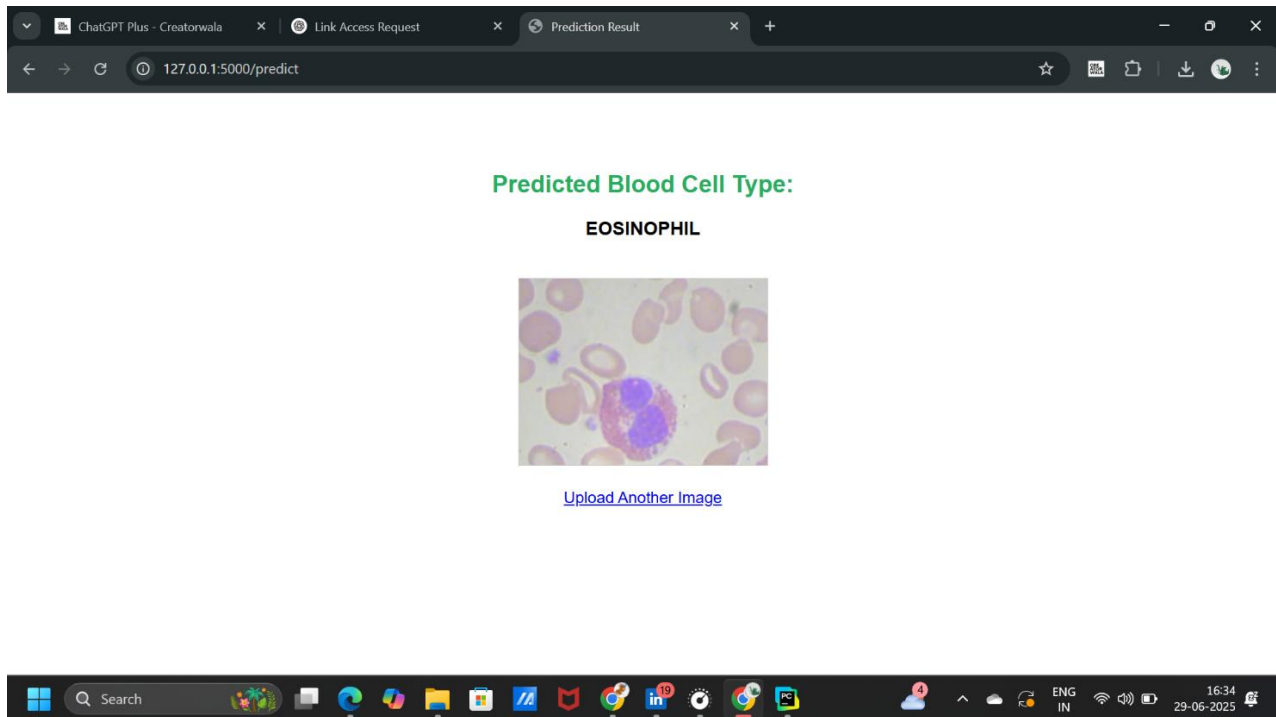
```
Click to go back, hold to see history
rematovisionupynio
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text Run all Connect ^
[ ] # Train the model
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=10
)

Epoch 1/10
249/249 436s 2s/step - accuracy: 0.5144 - loss: 1.1451 - val_accuracy: 0.7509 - val_loss: 0.6368
Epoch 2/10
249/249 421s 2s/step - accuracy: 0.7397 - loss: 0.6497 - val_accuracy: 0.7971 - val_loss: 0.5121
Epoch 3/10
249/249 436s 2s/step - accuracy: 0.8079 - loss: 0.4909 - val_accuracy: 0.8443 - val_loss: 0.4128
Epoch 4/10
249/249 476s 2s/step - accuracy: 0.8375 - loss: 0.4136 - val_accuracy: 0.8523 - val_loss: 0.3737
Epoch 5/10
249/249 484s 2s/step - accuracy: 0.8688 - loss: 0.3461 - val_accuracy: 0.8845 - val_loss: 0.3020
Epoch 6/10
249/249 434s 2s/step - accuracy: 0.8829 - loss: 0.3043 - val_accuracy: 0.9026 - val_loss: 0.2746
Epoch 7/10
249/249 436s 2s/step - accuracy: 0.8922 - loss: 0.2830 - val_accuracy: 0.9046 - val_loss: 0.2519
Epoch 8/10
249/249 391s 2s/step - accuracy: 0.9144 - loss: 0.2278 - val_accuracy: 0.9211 - val_loss: 0.2309
Epoch 9/10
249/249 393s 2s/step - accuracy: 0.9199 - loss: 0.2077 - val_accuracy: 0.9071 - val_loss: 0.2315
Epoch 10/10
249/249 448s 2s/step - accuracy: 0.9296 - loss: 0.1930 - val_accuracy: 0.8970 - val_loss: 0.2735

[ ] # Save the trained model
model.save("Blood Cell.h5")
print("Model saved successfully as 'Blood Cell.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 legacy. We recommend using instead the native Keras format, e.g.
```

## 6. Dashboard Design



## 7. Visualizations

