HematoVision: Blood Cell Classification using Transfer Learning

1. Import necessary libraries

```
import tensorflow as tf from tensorflow.keras.applications import ResNet50 from
tensorflow.keras.models import Model from tensorflow.keras.layers import Dense,
GlobalAveragePooling2D from tensorflow.keras.preprocessing.image import
ImageDataGenerator import matplotlib.pyplot as plt
```

2. Define image size and batch size

```
IMAGE_SIZE = (224, 224) BATCH_SIZE = 32
```

3. Prepare the dataset using ImageDataGenerator

```
train_datagen = ImageDataGenerator( rescale=1./255, validation_split=0.2 )
```

Load training data (80%)

```
train_generator = train_datagen.flow_from_directory( 'dataset/', target_size=IMAGE_SIZE,
batch_size=BATCH_SIZE, class_mode='categorical', subset='training' )
```

Load validation data (20%)

```
val_generator = train_datagen.flow_from_directory( 'dataset/', target_size=IMAGE_SIZE,
batch_size=BATCH_SIZE, class_mode='categorical', subset='validation' )
```

4. Load pre-trained ResNet50 model without top layers

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

5. Add custom layers on top of ResNet50

```
x = base_model.output x = GlobalAveragePooling2D()(x)          # Global average pooling layer x
= Dense(128, activation='relu')(x)     # Fully connected layer predictions = Dense(3,
activation='softmax')(x)  # Output layer (3 classes)
```

6. Define the final model

```
model = Model(inputs=base_model.input, outputs=predictions)
```

7. Freeze base model layers (do not train)

```
for layer in base_model.layers: layer.trainable = False
```

8. Compile the model

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

9. Train the model

```
history = model.fit( train_generator, validation_data=val_generator, epochs=10, verbose=1 )
```

10. Evaluate the model on validation data

```
loss, accuracy = model.evaluate(val_generator) print(f"Validation Accuracy:
{accuracy*100:.2f}%")
```

11. Save the trained model

```
model.save("hemato_model.h5")
```

12. Plot training & validation accuracy and loss

```
plt.figure(figsize=(12, 4))
```

Accuracy plot

```
plt.subplot(1, 2, 1) plt.plot(history.history['accuracy'], label='Train Accuracy', color='blue')
plt.plot(history.history['val_accuracy'], label='Val Accuracy', color='green') plt.title('Model
Accuracy') plt.xlabel('Epoch') plt.ylabel('Accuracy') plt.legend()
```

Loss plot

```
plt.subplot(1, 2, 2) plt.plot(history.history['loss'], label='Train Loss', color='red')
plt.plot(history.history['val_loss'], label='Val Loss', color='orange') plt.title('Model Loss')
plt.xlabel('Epoch') plt.ylabel('Loss') plt.legend()
```

```
plt.tight_layout() plt.savefig('training_plot.png') plt.show()
```