HematoVision: Advanced Blood Cell Classification

Introduction

HematoVision is a deep learning model designed for the classification of blood cells. This project utilizes transfer learning with a pre-trained MobileNetV model to achieve high accuracy in identifying red blood cells (RBCs) from microscopic images. This documentation provides a comprehensive guide to understanding, setting up, training, evaluating, and using the HematoVision model.

. Project Overview

HematoVision aims to automate and assist in the crucial task of blood cell analysis. Accurate and efficient classification of blood cells is vital for diagnosing various hematological conditions. By leveraging transfer learning, the model can achieve robust performance even with relatively smaller datasets, making it a practical solution for medical imaging applications. This project focuses on single-class classification, specifically identifying red blood cells (RBCs).

. Dataset

The HematoVision model was trained and evaluated using a publicly available dataset from Kaggle, titled "Blood Cell Images" []. This dataset contains images of individual blood cells, along with XML annotations providing bounding box coordinates and labels for each cell. For this specific project, due to the nature of the provided data, the model was configured for single-class classification, focusing solely on the identification of Red Blood Cells (RBCs).

Dataset Structure:

The dataset is organized into the following main directories:

- JPEGImages/ : Contains the blood cell images in JPEG format.
- Annotations/ : Contains XML files, each corresponding to an image in JPEGImages/ and providing annotations (bounding box coordinates and labels) for the blood cells within that image.

Data Challenges and Approach:

During the data preprocessing phase, it was observed that the dataset primarily contained annotations for a single class (RBCs) across the provided images. While a multi-class classification was initially considered, the available data steered the project towards a binary classification task: identifying whether a given image segment contains an RBC or not. This approach simplifies the problem given the dataset's characteristics and still provides a valuable proof-of-concept for automated blood cell detection.

. Environment Setup

To set up the environment for HematoVision, you will need Python .x and several libraries. It is highly recommended to use a virtual environment to manage dependencies.

Prerequisites:

- Python .x
- pip (Python package installer)

Steps to set up the environment:

. Create a virtual environment (recommended):

```
bash python3 -m venv hematovision_env
```

. Activate the virtual environment:

On Linux/macOS:

```
bash source hematovision_env/bin/activate
```

On Windows:

```
bash .\\hematovision_env\\Scripts\\activate
```

. Install the required Python libraries:

The following libraries are essential for running the HematoVision project:

- tensorflow : For building and training the deep learning model.
- keras : High-level neural networks API, running on top of TensorFlow.
- scikit-learn : For data splitting and evaluation metrics.
- matplotlib : For plotting and visualization.
- opencv-python (cv): For image preprocessing tasks.
- numpy : For numerical operations.

You can install all of them using pip:

```
bash pip install tensorflow keras scikit-learn matplotlib opencvpython numpy
```

Note: Depending on your system and TensorFlow version, you might need to install *tensorflow-cpu* if you don't have a compatible GPU, or *tensorflow-gpu* if you do. The *tensorflow* package usually installs the GPU version if compatible hardware is detected.

Once these steps are completed, your environment will be ready to run the data preprocessing, model training, and evaluation scripts.

. Data Preprocessing

The data preprocessing pipeline is crucial for preparing the raw image and annotation data for model training. The preprocess_data.py script handles the following steps:

1 Loading Data:

It reads image files from the JPEGImages directory and their corresponding XML annotation files from the Annotations directory.

2 Parsing Annotations:

For each image, it parses the XML file to extract bounding box coordinates (xmin, ymin, xmax, ymax) and labels for each object (blood cell) detected within the image. In this project, the primary label of interest is 'RBC'.

3 Image Cropping and Resizing:

Based on the extracted bounding box coordinates, the script crops the relevant blood cell regions from the original images. These cropped images are then resized to a uniform size (x pixels) to ensure consistency for the neural network input.

4 Pixel Normalization:

The pixel values of the images are normalized to a range between and . This is a common practice in deep learning to help the model converge faster and perform better.

5 Label Encoding:

The textual labels (e.g., 'RBC') are converted into numerical format using LabelEncoder . For single-class classification, this typically results in a binary encoding (or).

6 Data Splitting:

The preprocessed data (images and their corresponding labels) is split into training, validation, and test sets. This division is essential for proper model development:

- Training Set: Used to train the model.
- Validation Set: Used to tune the model's hyperparameters and prevent overfitting during training.
- Test Set: Used for final evaluation of the model's performance on unseen data.

7 Saving Preprocessed Data:

The preprocessed images and labels are saved as a compressed NumPy archive (preprocessed_data.npz). This allows for faster loading during subsequent training and evaluation phases, avoiding the need to re-process raw data every time.

Usage:

To run the data preprocessing script, execute the following command in your terminal:

```
python3 preprocess_data.py
```

Upon successful execution, you will find a preprocessed_data.npz file in your project directory, containing the prepared datasets and class information.

. Conclusion

HematoVision demonstrates the successful application of transfer learning for blood cell classification, achieving high accuracy in identifying Red Blood Cells (RBCs) using a pre-trained MobileNetV architecture. This project showcases the potential of AI in medical diagnostics and research, serving as a foundation for automated blood cell analysis. Future enhancements include expanding to multi-class classification, real-time inference, developing a user-friendly interface, and training on larger datasets to improve generalization and robustness.

. References

[] Paul, T. (). Blood Cell Images. Kaggle. https://www.kaggle.com/datasets/paultimothymooney/blood-cells