PS-FTSM-2023-025

DOUBLE MODEL IMPROVED ATTENTION MULTIPLE INSTANCE LEARNING TECHNIQUE REPORT

WEI QIANPU, VJ.KOK

Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia
43600 UKM Bangi, Selangor Malaysia.

p125223@siswa.ukm.edu.my, vj.kok@ukm.edu.my

ABSTRACT

Multiple Instance Learning (MIL) is one of the dominant methods for analyzing histopathology whole slide images (WSIs). These methods can primarily be categorized into key-instance-based MIL and embedding-based MIL. However, both types of MIL methods have their limitations. The key-instance-based method may assign incorrect pseudo-labels for key bags, and the embedding-based method aggregates feature from all bags to obtain slide features, hence may lead to model learning irrelevant or opposing labeled features. To address these limitations, we propose a MIL model combining the two methods, i.e., Dual-Model Improved Attention Multiple Instance Learning (DMIA-MIL). In the key instance identification stage, we introduce exponential attention. This can eliminate the influence of irrelevant parts within the bag, enabling the model to focus on distinguishing between positive (tumor cell) and negative (normal cell) instances. In the embedding stage, relevant features are selected based on attention scores from the key instance identification stage for embedding, thus enabling the model to learn the accurate instance information. Extensive experiments on the public benchmark histopathology CAMELYON16 dataset show that the proposed DMIA-MIL outperforms the current state-of-the-art MIL models in the histopathology whole slide images classification task. DMIA-MIL algorithm proposed in this study is primarily divided into four parts. The first part involves the segmentation and preprocessing of histopathology WSIs. The second part employs ResNet50 to extract features from all pseudo-bags and randomly group them. The third part utilizes Tier1 MIL to assign a preliminary attention to the pseudo-bag features, followed by feature selection. The fourth part engages Tier2 MIL to relearn and train the selected features, ultimately achieving the effect of predicting the WSI labels.

Keyword: Multiple Instance Learning, Histopathology WSIs, Medical Image Classification

I. INTANCE SEGMENTATION

The provided Python script performs instance segmentation on histopathological images by following a systematic process. Initially, the script is set up with user configurations, specifying parameters like the number of threads, magnification level, stride, patch size, and tissue mask threshold. It then identifies regions of interest (ROIs) in the slides by converting images to the HSV color space and using Otsu's thresholding. Morphological operations refine the mask, and contours are detected to draw bounding boxes around the segmented tissue. Subsequently, using the `Extract_Patch_From_Slide_STRIDE` function, the script iteratively moves over the mask to extract patches based on predefined stride and size parameters. These extracted patches are then saved as JPEG images in the specified directory. The script's modularity ensures precision in segmenting and extracting pertinent sections from histopathological slides.

1.1. Configuration Setup

- The number of threads is set to 4.

- The patch dimension level is set to 1, indicating a 20x magnification.

- A stride of 256 and patch size of 256 is used for extracting patches from the slide.

- A tissue mask threshold of 0.8 is defined.

- Slide images are read from a specified directory and patches are saved to another directory.

1.2. Region of Interest (ROI) Extraction

- A function `get_roi_bounds` is used to extract the region of interest from the slide.

- The function reads a region from the slide, converts it to HSV color space, and then applies Otsu's thresholding to segment tissue from the background.

- Morphological operations (close and open) are applied to refine the mask.

- Contours are detected on the mask and bounding boxes are drawn around them.

1.3. Patch Extraction

- A function `Extract_Patch_From_Slide_STRIDE` is used to extract patches from the slide.

- The function calculates the patch size and stride for the mask level and then iterates over the mask to extract patches based on the specified stride and size.

- Patches are then saved as JPEG images.

1.4. Helper Functions

- `getFolder_name` returns the folder name based on the slide name, patch level, and patch size.

- `read_tumor_mask` reads the tumor mask for a given slide.

1.5. Main Execution

- The `main` function orchestrates the entire process. It opens the slide, extracts the ROI, and then extracts patches from the slide.

- The script can be executed directly, where it reads slides from the specified directory, extracts patches, and saves them.

1.6. Conclusion

The provided script offers a comprehensive solution to segment regions of interest from histopathological slides and extract patches from them. The use of Otsu's thresholding and morphological operations ensures accurate segmentation, and the stride-based patch extraction ensures consistent patch sizes across the slide. The script is modular and can be easily adapted for different configurations or use cases.

II. FEATURE EXTRACTION

This charter outlines the implementation of feature extraction using a modified ResNet-50 architecture. The Python script, which is provided, is structured to process histopathological images, extract relevant features, and save the results in an organized manner for further analysis. The architecture leverages a bottleneck block, defined in the `Bottleneck_Baseline` class, comprising three sequential convolutions interspersed with batch normalization and ReLU activation functions. The primary structure, `ResNet_Baseline`, integrates these blocks, beginning with an initial convolution, followed by three sets of bottleneck blocks of varying depths. An adaptive average pooling layer finalizes the forward propagation, yielding the desired feature tensor. Initialization of the model's weights employs the Kaiming normalization for convolutional layers and constants for batch normalization layers. The model can optionally harness pre-trained weights from ImageNet, facilitating transfer learning. For the dataset, organized hierarchically with primary folders representing slides containing multiple image patches, a specific class `Patch_dataset_SlideFolder_noLabel` has been created. Images undergo transformations, including center cropping, tensor conversion, and normalization, before being fed into the model. The primary function, `main`, coordinates the feature extraction, which is detailed in the `extract_save_features` function: the model operates in evaluation mode, processes batches of images to extract features, and subsequently stores these features alongside associated metadata in a dictionary. Lastly, the features are serialized and saved per slide using the pickle format, ready for downstream tasks like image classification or clustering.

2.1. ResNet-50 Model

- The script includes a class named `Bottleneck_Baseline` that defines a single bottleneck layer with convolutional, batch normalization, and ReLU operations. The class also contains methods for forward propagation.

- Another class named `ResNet_Baseline` defines the ResNet architecture with layers and down-sampling, utilizing the previously defined bottleneck layers.

- The function `resnet50_baseline` constructs a modified ResNet-50 model using the defined architecture. Pre-trained weights can be loaded from the provided URL.

2.2. Data Processing

- The script uses command-line arguments to set parameters for data processing, including the data directory, device type (CPU or GPU), number of workers, crop size, batch size, and log directory.

- The `main` function reads the arguments and initializes the ResNet feature extractor, as well as data transformation and loading.

2.3. Feature Extraction

- The function `extract_save_features` takes an extractor, data loader, and parameters as inputs. It evaluates the extractor model, processes data in batches, extracts features, and saves them as dictionaries in a `.pkl` file.

2.4. Dataset Preparation

- A custom dataset class named `Patch_dataset_SlideFolder_noLabel` is defined for loading patches from a slide directory. It reads image files, resizes them, and prepares the data for further processing.

2.5. Main Execution

- The script executes the `main` function, initializing the ResNet model, data transformation, and loading. It then performs feature extraction using the trained model and saves the results to the specified directory.

2.6. Conclusion

The presented script demonstrates an effective approach to instance segmentation by utilizing a modified ResNet-50 architecture for feature extraction. The script processes histopathological image data, extracts meaningful features, and organizes them for analysis. The modular design and usage of command-line arguments enhance the script's flexibility and adaptability for various research and practical applications.

III. DOUBLE MIL MODEL

This charter showcases a double model improved attention (DMIA) approach using Multiple Instance Learning (MIL) for medical image analysis, specifically slide image classification. This method comprises a two-tiered classification: The first tier involves group-based instance classification while the second tier uses distilled features for slide-level predictions.

3.1. Model Architectures

Classifier_1fc: A straightforward linear classifier with optional dropout.

residual_block: A simple residual block to support deeper feature extraction without additional complexity.

DimReduction: A dimensionality reduction module that can incorporate multiple residual blocks.

Attention Mechanisms:

- Attention2: A basic attention mechanism that determines instance importance.
- Attention_Gated: An enhanced attention mechanism with gating.
- AttentionLayer: A complex attention mechanism that can handle multiple heads.
- Attention_with_Classifier: An integrated module combining attention and classification.

3.2. Training and Testing Approach

The main procedure, main(), first initializes the model components: a classifier, attention mechanism, dimensionality reduction module, and an attention-based classifier. Training data is loaded and reorganized into slide names, feature lists, and labels.

Training is conducted in dual stages:

⑴ First-Tier Training: Groups of instances are processed, and their attention-weighted features are classified.

⑵ Second-Tier Training: Features from the first tier are distilled and used to make slide-level predictions.

The two tiers are trained alternately, ensuring that the second tier benefits from the refined features of the first. The test_attention_DTFD_preFeat_MultipleMean function evaluates the model. It uses the features processed by the first-tier for slide-level predictions, consolidating them through averaging across multiple inferences. This approach enhances the robustness of predictions.

3.3. Utilities and Data Handling

AverageMeter: Computes and stores average values. Useful for tracking metrics like loss during training.

print_log: A utility function for logging.

reOrganize_mDATA & reOrganize_mDATA_test: Functions that restructure the input data into slide names, feature lists, and labels.

3.4. Implementation Details

Data: Slide images are split into training, validation, and testing sets. Slide-level labels are inferred based on naming conventions: slides named with 'tumor' are labeled as 1, and 'normal' as 0.

Optimization: Adam optimizer is used with an optional learning rate scheduler.

Regularization: Gradient clipping ensures stability during training.

Distillation Types: The code supports various distillation strategies: 'MaxMinS', 'MaxS', and 'AFS'.

Logging and Model Saving: Training progress is logged, and the best model is saved based on validation AUC.

3.5. Conclusion

The DMIA-MIL method offers a systematic two-tier approach to slide image classification. By leveraging group-level attention mechanisms and feature distillation, it aims to achieve robust and accurate slide-level predictions.