**FMS User Manual**

**Introduction**

This manual provides detailed instructions on how to use the FMS (Fusion Medical System) software for training, classification, evaluation, and fusion diagnosis of medical image data. The system supports multiple datasets, automates model training, classification, evaluation, and performs fusion diagnosis on test data.

**System Requirements**

- Python 3.x

- Required libraries:

- `psutil`

- `scikit-learn`

- `Keras` (with `InceptionResNetV2` model)

- Custom modules: `classification\_module`, `classification\_evaluator`, `ERTrain`, `multi\_mfusiond`

- Necessary data folders and files structured as described below.

**Directory Structure**

- model\_appendix/

Contains subfolders for each dataset (e.g., AODC, CC, CAE, etc.). Each dataset folder should contain:

- `train/` directory with training images

- `models/` directory for saving trained models

- `BestModel/` directory for storing best-performing models

- `json/` directory for model configuration files

- `Testresults/` directory for classification and evaluation results

- `TestModel/` directory for test models (optional)

- TestImage\_crop/

Directory containing cropped test images used during classification.

- Testdata\_184.xlsx

Excel file containing test data labels and metadata.

- Test\_data.xlsx

Excel file used for the fusion diagnosis module.

- TestImage/

Directory containing test images used in fusion diagnosis.

- Testresults/

Output directory for fusion diagnosis results.

**Main Functionalities**

1. Processing a Single Data Folder

The main function `process\_data\_folder(data\_folder)` automates the following steps for a given dataset:

- Training Phase\*

- Performs ER training on the dataset.

- Counts the number of training samples.

- Executes model training using the `InceptionResNetV2` architecture.

- Stores trained models in the appropriate folders.

- Classification Phase:

- Runs classification on test images using the trained model.

- Uses dataset-specific label pairs for classification.

- Evaluation Phase:

- Evaluates classification results.

- Provides performance metrics based on true labels and predicted probabilities.

2. Fusion Diagnosis

The `run\_FusionDiagnosis` function performs fusion diagnosis by combining results from multiple data sources.

Usage Instructions

Step 1: Prepare Data

- Organize your data folders inside `model\_appendix/` following the structure mentioned above.

- Ensure the Excel files `Testdata\_184.xlsx` and `Test\_data.xlsx` are in the working directory.

- Prepare test images and place them in `TestImage\_crop/` and `TestImage/` as required.

Step 2: Run Processing for Each Dataset

Uncomment and run the following loop in the main script to process all datasets sequentially:

```python

data\_directories = ['AODC', 'AWC', 'BFC', 'CAE', 'CC', 'FAC', 'GAC', 'MSC', 'PFCC']

for folder in data\_directories:

process\_data\_folder(folder)

```

This will:

- Train models for each dataset.

- Perform classification.

- Evaluate results.

Step 3: Run Fusion Diagnosis

After processing individual datasets, run fusion diagnosis with the command:

```python

run\_FusionDiagnosis(

excel\_file\_path='Test\_data.xlsx',

source\_directory='TestImage',

output\_directory='Testresults'

)

```

This will generate combined diagnostic results based on multiple datasets.

Dataset Label Mappings

Each dataset uses specific label pairs for classification and evaluation:

| Dataset | Positive Label | Negative Label |

|---------|-------------------------|--------------------------|

| AODC | Thickening | No thickening |

| CC | Presence of fecal stones| No fecal stones |

| CAE | Presence of fluid accumulation | No fluid accumulation |

| AWC | Abnormal appendix wall | Continuous appendix wall |

| GAC | Presence of gas accumulation | No gas accumulation |

| FAC | Presence of fluid accumulation | No fluid accumulation |

| PFCC | Presence of fluid accumulation | No fluid accumulation |

| MSC | Presence of swelling | No swelling |

| BFC | Presence of blood flow | No blood flow |

---

Functions Overview

| Function | Description |

|-----------------------|-------------------------------------------------------|

| `proc\_exist(name)` | Checks if a process with the given name is running. |

| `del\_file(path)` | Deletes all files and folders inside the specified path. |

| `get\_filenum(dir)` | Returns the total number of files in the directory and subdirectories. |

| `get\_all(dir)` | Returns a list of all files (including subdirectories) in the given directory. |

| `process\_data\_folder(folder\_name)` | Runs training, classification, and evaluation for the specified dataset folder. |

---

**Notes**

- The system uses the `InceptionResNetV2` model by default.

- Training parameters such as batch size and number of experiments can be modified in the `process\_data\_folder` function.

- Warning messages are suppressed to keep the output clean.

- Make sure custom modules (`classification\_module`, `classification\_evaluator`, `ERTrain`, `multi\_mfusiond`) are installed and accessible in your Python environment.

---

Troubleshooting

- Missing Training Directory:

If the `train` folder is missing in a dataset directory, processing will skip that dataset.

- Module Import Errors\*

Ensure all custom modules are correctly installed or available in the Python path.

- Data Format Issues:

Verify that Excel files and image folders are correctly formatted and paths are accurate.

---

Contact and Support

For further assistance, please contact the development team or refer to the project documentation repository.

---

End of FMS User Manual