Deep Learning Interactive Visualization

Project Documentation

Table of Contents

Project Overview

Purpose

This project implements an interactive visualization system for understanding and interpreting 3D Convolutional Neural Networks (CNNs) used in Alzheimer's disease detection. The system allows medical professionals and researchers to visualize which brain regions contribute most significantly to the model's decisions.

Publication

The project is published in Alzheimer's Research & Therapy (2021):

- Title: "Improving 3D convolutional neural network comprehensibility via interactive visualization of relevance maps"
- DOI: 10.1186/s13195-021-00924-2

Key Features

- 3D CNN model for Alzheimer's disease detection
- Interactive visualization of brain regions
- Real-time relevance mapping
- Web-based interface using Bokeh
- Docker containerization for easy deployment

Technical Architecture

MVC Architecture

The project follows the Model-View-Controller (MVC) pattern:

- **Model** (`datamodel.py`): Handles data processing and CNN operations
- **View** (`view.py`): Manages the visualization interface
- **Controller** (`main.py`): Coordinates user interactions and updates

Core Technologies

1. **Backend**
   - Python for core implementation
   - TensorFlow 1.15 for CNN model
   - Bokeh for web visualization
   - NumPy for numerical computations
2. **Frontend**
   - Bokeh web application
   - Interactive HTML5 interface
   - Real-time data visualization
3. **Deployment**
   - Docker containerization
   - Web service deployment
   - Local installation support

Implementation Details

CNN Model

The system uses a 3D Convolutional Neural Network specifically designed for brain image analysis:

```
def apply_thresholds(self, relevance_map, threshold=0.5,
cluster_size=20):
    # Apply threshold to relevance map
    self.overlay = np.copy(relevance_map)
    self.overlay[np.abs(self.overlay) < threshold] = 0

    # Cluster size filtering
    labelimg = np.copy(self.overlay)
    labelimg[labelimg > 0] = 1  # binarize img
    labelimg = label(labelimg, connectivity=2)

    # Calculate cluster properties
    lprops = regionprops(labelimg, intensity_image=self.overlay)
```

Visualization System

The visualization component converts neural network outputs into interpretable visual representations:

```
def overlay2rgba(relevance_map, alpha=0.5):
    # Convert relevance map to RGBA visualization
    alpha_mask = np.copy(relevance_map)
    alpha_mask[np.abs(alpha_mask) > 0] = alpha

    # Scale to color range
    relevance_map = relevance_map / 2 + 0.5
    ovl = np.uint8(overlay_colormap(relevance_map) * 255)
    ovl[:, :, :, 3] = np.uint8(alpha_mask * 255)
    return ovl.view("uint32").reshape(ovl.shape[:3])
```

Interactive Features

The system provides real-time interaction with brain visualizations:

```
def click_frontal_callback(event):
    # Handle user interaction with frontal brain view
    if event.x < 1:
        x = 1
    elif event.x > slice_slider_sagittal.end:
        x = slice_slider_sagittal.end
    else:
        x = int(round(event.x))

    # Update other views
    slice_slider_sagittal.update(value=x)
    slice_slider_axial.update(value=y)
    plot_sagittal()
```

Core Components

1. Data Model (`datamodel.py`)

- Handles data preprocessing
- Manages CNN model interactions
- Processes relevance maps
- Coordinates data flow between components

2. View System (`view.py`)

- Implements the user interface
- Manages visualization layouts
- Handles real-time updates
- Provides interactive controls

3. Controller (`main.py`)

- Coordinates system components
- Handles user input
- Manages application state
- Orchestrates updates

Installation & Setup

Prerequisites

- Python <3.8
- TensorFlow 1.15
- Bokeh ·2.2.3

- CUDA support for GPU acceleration

Installation Steps

1. Clone the repository:

   ```
   git clone

   https://github.com/martindyrba/DeepLearningInteractiveVis
   ```

2. Create a Python environment:

   ```
   conda create -n InteractiveVis python=3.7
   conda activate InteractiveVis
   ```

3. Install dependencies:

   ```
   pip install -r requirements.txt
   ```

Docker Deployment

```
docker pull martindyrba/interactivevis
./run_docker_intvis.sh
```

Usage Guide

Starting the Application

1. Local Installation:

   ```
   bokeh serve InteractiveVis --show
   ```

2. Docker Container:

   ```
   ./run_docker_intvis.sh
   ```

3. Web Service:

   - Access the demo at: https://explaination.net/demo

Using the Interface

1. **Subject Selection**

   - Choose from internal dataset
   - Upload custom brain scans
   - Enter subject information

2. **Visualization Controls**

   - Adjust relevance thresholds
   - Control transparency
   - Navigate brain regions
   - View cluster statistics

3. **Analysis Features**

   - Real-time relevance mapping

- Region identification
- Statistical analysis
- Export capabilities

Code Documentation

Key Classes and Methods

Model Class

```
class Model:
    def set_subject(self, subject_id):
        """Sets the current subject and loads their data"""

    def apply_thresholds(self, relevance_map, threshold):
        """Applies thresholds to relevance maps"""

    def process_visualization(self):
        """Processes data for visualization"""
```

View Class

```
class View:
    def update_visualization(self):
        """Updates the visualization display"""

    def handle_interaction(self, event):
        """Handles user interaction events"""

    def render_overlay(self):
        """Renders the relevance map overlay"""
```

Research Impact

Clinical Applications

- Improved understanding of CNN decisions
- Enhanced diagnostic support
- Better interpretation of results
- Validation across multiple datasets

Technical Contributions

- Novel visualization approach
- Interactive relevance mapping
- Real-time analysis capabilities
- Extensible architecture

Future Directions

- Enhanced model interpretability
- Extended dataset support
- Advanced visualization features
- Improved clinical integration

Resources

Project Links

- GitHub: github.com/martindyrba/DeepLearningInteractiveVis
- Docker Hub: docker pull martindyrba/interactivevis
- Demo: explaination.net/demo
- Documentation: Available in project README

Support

For issues and questions:

- GitHub Issues
- Project Documentation
- Research Paper References