National College of Ireland

Configuration Manual for Enhanced Liver Tumor Detection Using Deep Learning Techniques for Biomedical Image Segmentation

MSc Research Project
MSc in Data Analytics

Vigneswara Venkata Sai Nilesh Gurazada
Student ID: x23235985

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vigneswara Venkata Sai Nilesh Gurazada
Student ID:	X23235985
Programme:	MSc in Data Analytics **Year:** 2024/2025
Module:	MSc Research Project
Lecturer:	Vikas Tomer
Submission Due Date:	29/01/2025
Project Title:	Configuration Manual for Enhanced Liver Tumor Detection Using Deep Learning Techniques for Biomedical Image Segmentation.
Word Count:	398 **Page Count:** 8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Vigneswara Venkata Sai Nilesh Gurazada

Date: 28th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☑
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☑
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☑

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual for Enhanced Liver Tumor Detection Using Deep Learning Techniques for Biomedical Image Segmentation

Vigneswara Venkata Sai Nilesh Gurazada
Student ID: x23235985

1. Introduction

This LT-Net Model where it is a deep learning model customly designed for liver tumor segmentation in medical imaging, particularly for CT and MRI scans. This configuration manual provides the necessary steps to set up and run the LT-Net model, including system requirements, dataset configuration, and how to train and evaluate the model.

2. System Requirements

Before setting up the LT-Net model, ensure that your system meets the following requirements:

Hardware Requirements:

- CPU: Multi-core processor (Intel i7 or AMD Ryzen or better recommended)
- GPU: NVIDIA GPU with CUDA support (recommended: NVIDIA GTX 1080 Ti or higher for faster training)
- RAM: Minimum 16 GB RAM
- Storage: At least 50 GB of free space for the dataset and model weights

Software Requirements:

- Operating System: Linux (Ubuntu 20.04 or later) or Windows 10/11
- Python: Version 3.7 or later
- CUDA and cuDNN: For GPU acceleration, install CUDA 10.1 or later and cuDNN 7.6 or later
- Required Libraries: TensorFlow 2.x, Keras, NumPy, Matplotlib, OpenCV, scikit-learn, Fastai.

3. Dataset Configuration

The dataset used in this research is sourced from the Liver Tumor Segmentation (LiTS) hosted on Kaggle, which provides 3D CT scan images and their corresponding tumor segmentation masks. The CT scans are stored in NIfTI format (.nii), which is widely used in medical imaging.

- Load the Dataset and Manage to create Dataframe

```python
# Initialize an empty list to store file paths
file_paths = []

# Walk through the dataset directory to gather file information
for directory, _, files in os.walk('dataset'):
    for file in files:
        file_paths.append((directory, file))

# Create a DataFrame from the gathered file paths
data_frame = pd.DataFrame(file_paths, columns=['directory', 'file_name'])
data_frame.sort_values(by=['file_name'], ascending=True, inplace=True)  # Sort DataFrame by file names
```

Python

```python
# View the DataFrame
data_frame.head()  # Display the first few rows of the DataFrame
```

Python

	directory	file_name
0	dataset\segmentations	segmentation-0.nii
1	dataset\segmentations	segmentation-1.nii
2	dataset\segmentations	segmentation-10.nii
3	dataset\segmentations	segmentation-100.nii
4	dataset\segmentations	segmentation-101.nii

- Mapping the CT Scans image with their corresponding labels

```python
# Initialize new columns for mask information
data_frame["mask_directory"] = ""  # Placeholder for mask directory
data_frame["mask_file_name"] = ""  # Placeholder for mask file names
```

Python

```python
# Loop through the expected indices to populate mask file information
for index in range(131):
    ct_scan = f"volume-{index}.nii"  # Construct CT scan file name
    segmentation = f"segmentation-{index}.nii"  # Construct corresponding segmentation file name

    # Populate the DataFrame with mask information
    data_frame.loc[data_frame['file_name'] == ct_scan, 'mask_file_name'] = segmentation
    data_frame.loc[data_frame['file_name'] == ct_scan, 'mask_directory'] = "dataset/segmentations"
```

Python

	directory	file_name	mask_directory	mask_file_name
0	dataset\volume_pt1	volume-0.nii	dataset/segmentations	segmentation-0.nii
1	dataset\volume_pt1	volume-1.nii	dataset/segmentations	segmentation-1.nii
2	dataset\volume_pt1	volume-10.nii	dataset/segmentations	segmentation-10.nii
3	dataset\volume_pt2	volume-11.nii	dataset/segmentations	segmentation-11.nii
4	dataset\volume_pt2	volume-12.nii	dataset/segmentations	segmentation-12.nii
5	dataset\volume_pt2	volume-13.nii	dataset/segmentations	segmentation-13.nii
6	dataset\volume_pt2	volume-14.nii	dataset/segmentations	segmentation-14.nii
7	dataset\volume_pt2	volume-15.nii	dataset/segmentations	segmentation-15.nii
8	dataset\volume_pt2	volume-16.nii	dataset/segmentations	segmentation-16.nii
9	dataset\volume_pt2	volume-17.nii	dataset/segmentations	segmentation-17.nii
10	dataset\volume_pt2	volume-18.nii	dataset/segmentations	segmentation-18.nii
11	dataset\volume_pt2	volume-19.nii	dataset/segmentations	segmentation-19.nii

- Function to load the NIfTI files to preprocess them

```python
#Function to load the .nii format files
def load_nii(file_path):
    '''
    Load a .nii file and return its pixel array.
    '''
    ct_image = nib.load(file_path)  # Load the NIfTI file using nibabel
    pixel_array = ct_image.get_fdata()  # Extract the pixel data
    pixel_array = np.rot90(np.array(pixel_array))  # Rotate the array for correct orientation
    return pixel_array  # Return the pixel data
```

Python

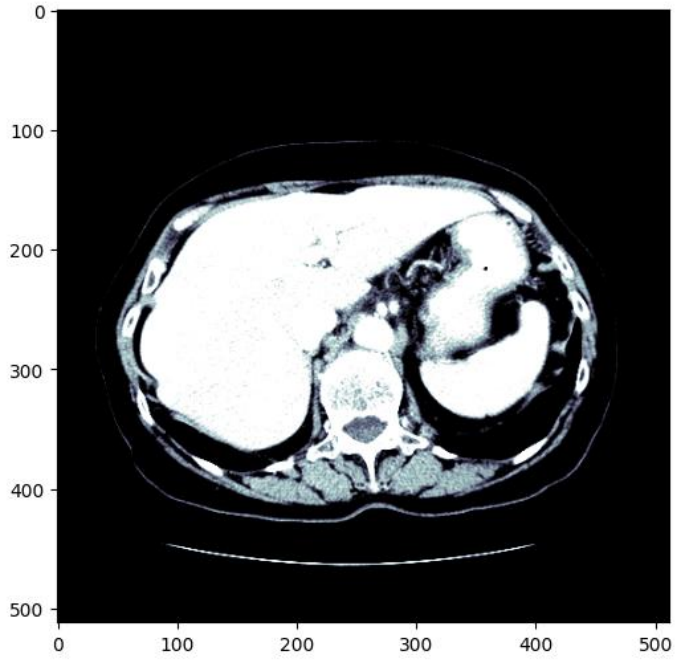
- Define the antonomical region for the liver tumor

```python
# Define window settings for liver tumor anatomical regions
window_settings = types.SimpleNamespace(
    liver=(150, 30),
    custom=(200, 60)
)
```

Python

```python
# Extend the Tensor class to include a windowing function

@patch
def apply_window(self: Tensor, width, level):
    pixel_data = self.clone()
    min_pixel = level - width // 2
    max_pixel = level + width // 2
    pixel_data[pixel_data < min_pixel] = min_pixel
    pixel_data[pixel_data > max_pixel] = max_pixel
    return (pixel_data - min_pixel) / (max_pixel - min_pixel)
```

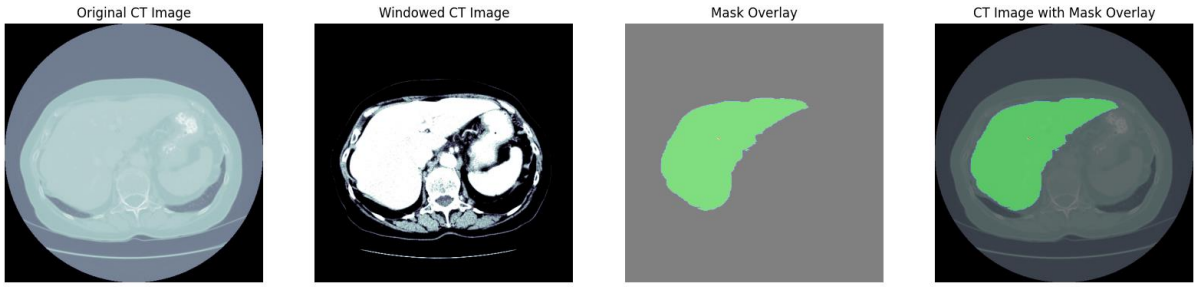
Python



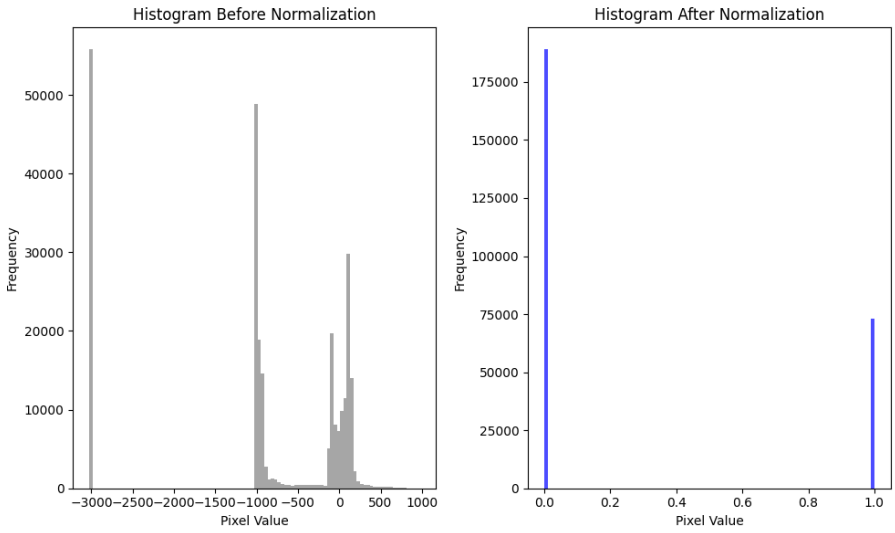
- Plot the Liver Tumor with its segmented Region

```python
def visualize_samples(image_arrays, color_mapping='nipy_spectral'):
    '''
    Visualizes a slice of CT scan along with its annotations.
    '''
    fig = plt.figure(figsize=(20, 16), dpi=100)

    # Original CT Image
    plt.subplot(1, 4, 1)
    plt.imshow(image_arrays[0], cmap='bone')
    plt.title('Original CT Image')
    plt.axis('off')

    # Windowed CT Image
    plt.subplot(1, 4, 2)
    plt.imshow(tensor(image_arrays[0].astype(np.float32)).apply_window(*window_settings.liver), cmap='bone')
    plt.title('Windowed CT Image')
    plt.axis('off')

    # Mask Overlay
    plt.subplot(1, 4, 3)
    plt.imshow(image_arrays[1], alpha=0.5, cmap=color_mapping)
    plt.title('Mask Overlay')
    plt.axis('off')

    # Combined CT and Mask Image
    plt.subplot(1, 4, 4)
    plt.imshow(image_arrays[0], cmap='bone')
    plt.imshow(image_arrays[1], alpha=0.5, cmap=color_mapping)
    plt.title('CT Image with Mask Overlay')
    plt.axis('off')
    plt.show()
```

Python



4. Model Initialization and Preprocessing



- Model Data Initialization and Proccesing

```python
# Set parameters for data loading and processing
BATCH_SIZE = 16  # Number of samples per batch
IMAGE_SIZE = 128  # Target size for images

# Define class codes for segmentation
class_codes = np.array(["background", "liver", "tumor"])

# Function to return the filename as a Path object
def get_image_filename(file_path: Path) -> Path:
    return file_path

# Function to generate the corresponding mask file path
def get_mask_filename(image_file: Path) -> Path:
    return output_path / 'train_masks' / f'{image_file.stem}_mask.png'

# Define transformations for data processing
transformations = [IntToFloatTensor(), Normalize()]  # Convert to float tensor and normalize

# Create a DataBlock for organizing the dataset
data_block = DataBlock(
    blocks=(ImageBlock(), MaskBlock(class_codes)),  # Specify the types of blocks (images and masks)
    batch_tfms=transformations,  # Apply transformations in batches
    splitter=RandomSplitter(),  # Randomly split the dataset into training and validation sets
    item_tfms=[Resize(IMAGE_SIZE)],  # Resize each item to the specified size
    get_items=get_image_files,  # Function to retrieve image files
    get_y=get_mask_filename  # Function to retrieve corresponding mask file paths
)

# Create the dataset from the specified image source
dataset = data_block.datasets(source=output_path / 'train_images')
```

Python

- Visualize the Segmented Dataset

```python
# Select an index to visualize images from the dataset
index = 200  # Index of the image to display

# Retrieve the image and its corresponding mask from the dataset
images = [dataset[index][0], dataset[index][1]]

# Create a figure with two subplots for displaying the images side by side
fig, axes = plt.subplots(1, 2)  # 1 row, 2 columns

# Iterate through the axes and images to display them
for i, ax in enumerate(axes.flatten()):
    ax.axis('off')  # Hide the axis for a cleaner presentation
    ax.imshow(images[i])  # Display the image or mask in the corresponding subplot
```

Python

- LT-Net Model Building and Training

```python
# Calculates the non-background accuracy for multiclass segmentation
def compute_foreground_accuracy(inputs, targets, background_index=0, axis=1):
    targets = targets.squeeze(1)  # Remove any singleton dimensions from targets
    mask = targets != background_index  # Create a mask to exclude background pixels
    return (inputs.argmax(dim=axis)[mask] == targets[mask]).float().mean()  # Calculate mean accuracy

# Includes the background in the accuracy metric by using a dummy value.
def custom_foreground_accuracy(inputs, targets):
    return compute_foreground_accuracy(inputs=inputs, targets=targets, background_index=3, axis=1)  # 3 is a
```

Python

```python
# Initialize a U-Net learner with the enhanced data loaders and model architecture
learner = unet_learner(
    data_loaders,  # Data loaders for training
    resnet50,  # Backbone architecture
    loss_func=CrossEntropyLossFlat(axis=1),  # Loss function for multiclass segmentation
    metrics=[compute_foreground_accuracy, custom_foreground_accuracy]  # Metrics for evaluation
)

# Fine-tune the model for 10 epochs with weight decay and model saving callback
learner.fine_tune(10, wd=0.1, cbs=SaveModelCallback())
```

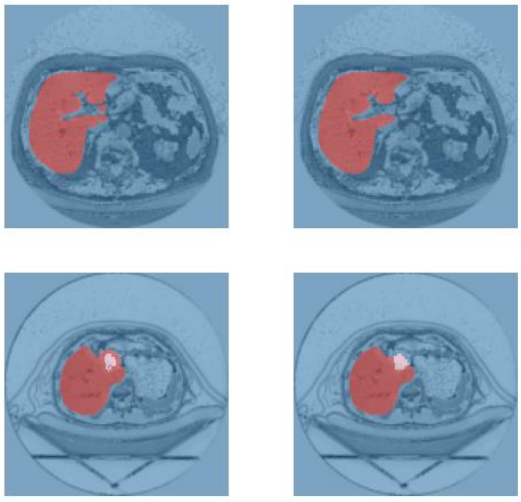
Python

epoch	train_loss	valid_loss	compute_foreground_accuracy	custom_foreground_accuracy	time
0	0.217112	0.054248	0.920119	0.985010	02:56

```
Better model found at epoch 0 with valid_loss value: 0.05424789711833.
```

epoch	train_loss	valid_loss	compute_foreground_accuracy	custom_foreground_accuracy	time
0	0.017466	0.012723	0.871375	0.995736	03:07
1	0.011359	0.009035	0.927754	0.996830	03:10
2	0.011427	0.007700	0.904527	0.997149	03:10
3	0.006926	0.005919	0.945737	0.997774	03:10
4	0.006051	0.006394	0.951457	0.997656	03:10
5	0.004645	0.004532	0.951800	0.998292	03:10
6	0.004072	0.004313	0.960855	0.998414	03:10
7	0.003320	0.003862	0.967538	0.998560	03:10
8	0.002959	0.003791	0.964670	0.998599	03:10
9	0.002695	0.003843	0.966276	0.998605	03:10

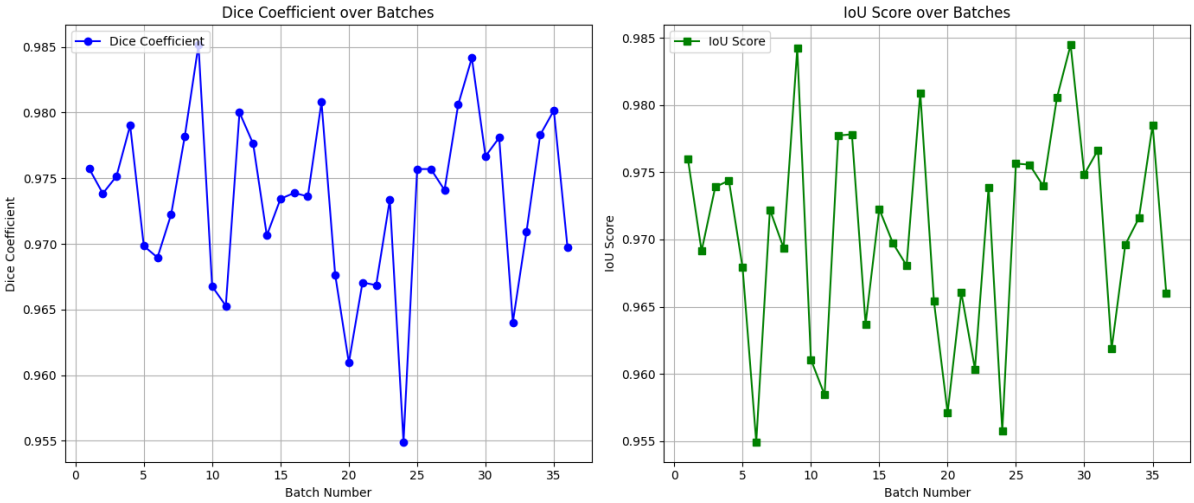
- Predictions of LT-Net Model



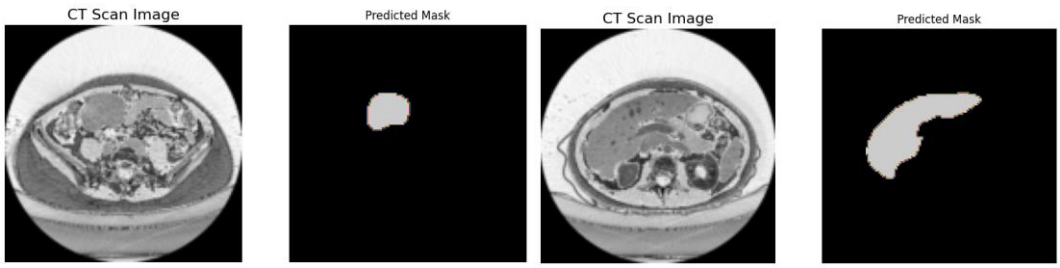
- Evaluation of the Model

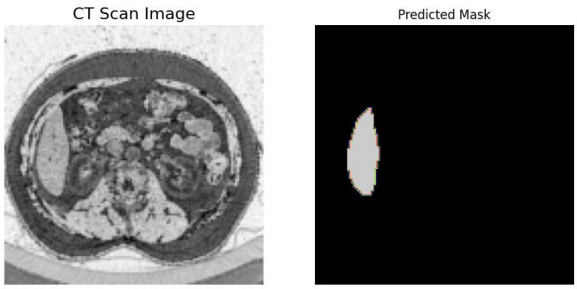
```python
# Plot Dice Coefficient and IoU
def plot_metrics(dice_values, iou_values):
    epochs = range(1, len(dice_values) + 1)

    # Create subplots
    fig, ax = plt.subplots(1, 2, figsize=(14, 6))

    # Plot Dice Coefficient
    ax[0].plot(epochs, dice_values, label='Dice Coefficient', color='b', marker='o')
    ax[0].set_title('Dice Coefficient over Batches')
    ax[0].set_xlabel('Batch Number')
    ax[0].set_ylabel('Dice Coefficient')
    ax[0].legend(loc='upper left')
    ax[0].grid(True)

    # Plot IoU Score
    ax[1].plot(epochs, iou_values, label='IoU Score', color='g', marker='s')
    ax[1].set_title('IoU Score over Batches')
    ax[1].set_xlabel('Batch Number')
    ax[1].set_ylabel('IoU Score')
    ax[1].legend(loc='upper left')
    ax[1].grid(True)

    # Show the plot
    plt.tight_layout()
    plt.show()
```

Python



- Predict and visualize the output mask for a given image from the dataset.





- Save the Trained LT-Net Model for Future Use.

```
# Export the trained model for future use
learner.export(output_path / 'Liver_segmentation')  # Save the model to the specified path
```

Python

- **After training the model, where save the model so that it can be used later without needing to retrain. This is particularly useful for the model in production, sharing it with others, or resuming training later.**

This structured implementation performs through the entire process of preparing, training, and evaluating a deep learning model for liver tumor detection. Each section is critical for ensuring that the data is correctly handled, processed, and used to train an effective model.

Conclusion

This manual provides all the necessary steps to set up, train, and evaluate the LT-Net model for liver tumor segmentation. By following the instructions carefully, you will be able to use this model for precise tumor detection, aiding clinicians in the diagnosis and treatment of liver diseases.

References

Python: https://www.python.org
Dataset Source: https://www.kaggle.com/datasets/andrewmvd/liver-tumor-segmentation