A KAGGLE DATA SCIENCE BOWL 2018 PROJECT

Medical Image Semantic Segmentation using U-NET

CS-441
Computer Vision

Dr. Jignesh Patel

Team

Sr. No.	Name	Roll Number
1	Prasoon Pathak	202051143
2	Suyash Jain	202051189

Index

Problem Introduction

- Medical image segmentation is a critical task in the field of healthcare and diagnostics involving **identifying and delineating specific structures** or regions within medical images, providing invaluable insights for medical professionals.
- **Accurate nuclei segmentation** is essential for understanding **cellular structures**, which is vital in medical research and diagnosis. By **automating t**he process of identifying and segmenting nuclei in images, we aim to improve the **efficiency** of medical professionals.
- **Aim:** In this project, we focus on the **segmentation of nuclei in medical images** which is a fundamental step in **various applications such as pathology analysis and cancer diagnosis.** Thus helping in **reducing workload** and facilitating **quicker and more accurate analyses**, ultimately contributing to advancements in medical imaging and patient care.
- **U-Net**, a **convolutional neural network architecture**, is particularly well-suited for image segmentation tasks. It excels at capturing intricate details and **spatial relationships** within images, making it ideal for the complex and nuanced structures found in medical imagery. By leveraging U-Net, our objective is to d**evelop a robust and accurate model for the precise segmentation of nuclei**, thereby aiding medical professionals in their diagnostic endeavours.

Project Components

- **Data Loading and Exploration: Loaded and explored the dataset to understand its structure.**
  - Utilised Pandas for Data Processing, **explored** the Kaggle **dataset**'s directory structure, extracted training and testing data from **compressed files**, displayed **file paths and directory information.**
- **Data Pre-processing : Prepared the data for U-Net model training.**
  - Set the s**eed (42) for reproducibility** of the data, **resized** the training and test images to 128x128x3, **created arrays for images and masks**, conducted a **sanity check** by displaying random training samples.
- **Designed the U-Net Architecture : Designed the U-Net model for semantic segmentation.**
  - Implemented input, contraction, expansion, and output layers, used convolutional and transpose convolutional layers, applied activation functions and initialisers, compiled the model using **binary cross entropy loss and Adam optimiser.**
- **Model Training : Trained the U-Net model on the prepared dataset.**
  - Defined **callbacks** for model checkpointing and **early stopping**, specified **training parameters (batch size, epochs),** utilised a **validation split during training, monitored training progress with TensorBoard.**
- **Model Evaluation : Evaluated the trained model on training, validation, and test sets.**
  - Evaluated the model using **tensor-board** and plotted various graphs depicting model performance.

Libraries Used

- **TensorFlow**
  - **Purpose:** Deep-Learning Framework
  - **Usage:** Implemented U-Net architecture, model training, and evaluation.
- **NumPy**
  - **Purpose:** Numerical operations and array manipulation.
  - **Usage:** Creating and handling arrays for image data.
- **Pandas**
  - **Purpose:** Data manipulation and analysis.
  - **Usage:** Loading and processing dataset information.
- **Matplotlib**
  - **Purpose: Data visualisation.**
  - **Usage: Plotting graphs, displaying images, and visualising training results.**
- **Scikit-image**
  - **Purpose:** Image processing.
  - **Usage:** Resizing images, transforming masks, and handling image-related tasks.

Dataset Overview

- **Source:** Kaggle Data Science Bowl 2018. **Link**
- **Directory Structure**
  - **Training Data:** Images located in “/stage1_train/{image_id}/images/“, Corresponding masks in “/stage1_train/{image_id}/masks/".
  - **Testing Data:** Images located in “/stage1_test/{image_id}/images/“.
- **Contents**
  - The dataset comprises **medical images and corresponding masks** for nuclei segmentation.
  - Images and masks are organised within unique directories identified by "image_id."
  - Training data is provided in a compressed file named "stage1_train.zip."
  - Testing data is provided in a compressed file named “stage1_test.zip.”.
- **Displaying Images and Masks**
  - Utilised **Matplotlib** for visualisation.
  - **Randomly selected** and displayed a subset of training images along with their respective masks.
  - Images showcased during data exploration to understand the dataset's characteristics.
  - Aided in assessing the complexity and diversity of nuclei structures within the dataset.

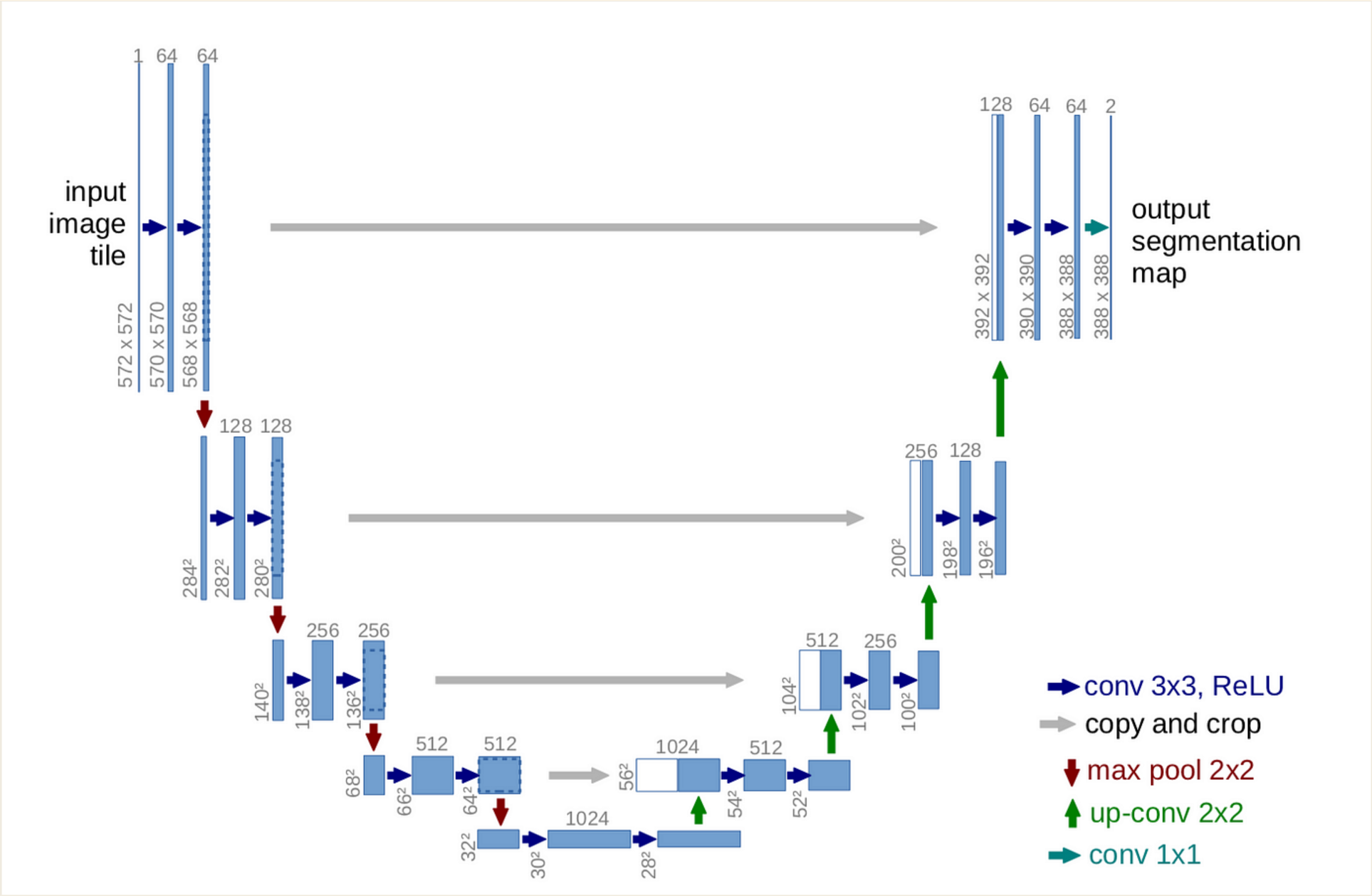
Data Pre-Processing

- **Extraction of Images and Masks from Zip Files**
  - **Objective:** Unpack data from compressed files for further processing.
  - **Procedure:** Used Python's **'zipfile'** library to extract contents.
  - **Outcome:** Access to individual images and masks within the dataset.
- **Resizing Images to a Common Size (128x128 pixels)**
  - **Objective:** Standardise image dimensions for model compatibility.
  - **Procedure:** Utilised **'Scikit-Image'** resize function.
  - **Outcome:** Ensured uniformity in image size, facilitating model training.
- **Creation of Training and Testing Datasets**
  - **Objective:** Prepare data for model training and evaluation.
  - **Procedure:** Organised resized images and masks into arrays.
  - **Outcome:** Created two distinct datasets:
    A. X_train: Array of resized training images
    B. Y_train: Array of corresponding resized training masks
  - Also prepared X_test for the resized testing images.
- **Note:** These steps are essential for ensuring that the input data is in a suitable format for the U-Net model, facilitating effective training and evaluation.

U-Net Architecture

- **Input Layer :** Receives the input image data dimensions of which match the size of input images (e.g., 128x128 pixels) and passes the input data to the subsequent layers for feature extraction.
- **Contraction Path with Convolutional and Pooling Layers :** Captures and consolidates high-level features and contains multiple convolutional layers with a small **receptive field (e.g., 3x3).** Each convolutional layer is followed by a **Rectified Linear Unit (ReLU)** activation. **Pooling layers (MaxPooling)** reduce **spatial-dimensions,** enhancing feature-extraction. Also have **optional dropout layers** for regularisation.
- **Expansion Path with Transposed Convolutional Layers :** It **enables precise localisation of features and segmentation.** The transposed convolutional layers perform **upsampling and concatenation** with corresponding feature maps from the contraction path. Convolutional layers also refine and expand feature maps.
- **Output Layer with Sigmoid Activation :** Produces the final segmentation mask having single convolutional layer with a 1x1 receptive field. **Sigmoid activation function** outputs **pixel-wise probability of belonging to the target class** (nuclei or background). The binary segmentation achieved by thresholding the output.
- Information flow is from the input layer through the contraction and expansion paths to the output layer.

U-NET Architecture



Model Training

- **Compilation of the Model**
  - **Optimiser:** Utilised the **Adam optimiser**, an adaptive learning rate optimisation algorithm.
  - **Loss Function:** Chose binary **cross-entropy as the loss function**, suitable for binary segmentation tasks.
  - **Metrics: Accuracy** was chosen as the evaluation metrics to assess model performance.
- **ModelCheckpoint and EarlyStopping Callbacks**
  - **ModelCheckpoint:** Saves the model at **specified intervals** during training, allows retrieval of the best-performing model, preventing potential overfitting, saved the model as 'model_for_nuclei.h5' with the best weights.
  - **EarlyStopping:** Halts training if the **validation loss plateaus and prevents overfitting** by stopping training when no improvement is observed, we set a patience of 2 epochs and monitored the validation loss.
- **Training on Resized Images and Masks**
  - **Input Data:** Utilised the X_train (resized training images) and Y_train (corresponding resized training masks) datasets.
  - **Training Parameters:** Batch Size: Set to 16 for efficient processing, Epochs: Defined as 25 to iterate through the entire dataset multiple times.
  - **Callbacks:** Incorporated the ModelCheckpoint and EarlyStopping callbacks during training, ModelCheckpoint **saved the best weights,** while EarlyStopping halted training when validation loss stagnated.
  - **TensorBoard Logging:** Monitored training progress with TensorBoard for visualising metrics and loss.

Training Results and Model Evaluation

- **Displaying Training and Validation Loss Curves**
  - Plotted **loss curves** using Matplotlib to visualise how training and validation losses evolve over epochs.
- **ModelCheckpoint and EarlyStopping to Prevent Overfitting**
  - Ensured that the model state with the **lowest validation loss was preserved**.
  - **Prevented overfitting** by stopping training when further optimisation did not lead to a reduction in validation loss.
- **Randomly Selected Training Sample : Displaying the original image, ground truth mask, and predicted mask**
  - Evaluated how well the model learned to segment nuclei in training data.
  - Provided insights into the model's ability to generalise.
- **Randomly Selected Validation Sample : Displaying the original image, ground truth mask, and predicted mask.**
  - Evaluated the model's generalisation to unseen data.
  - Examined whether the model could accurately segment nuclei in validation samples.

Conclusion and Future Enhancements

- **Conclusion**
  - In conclusion, this project successfully applied the U-Net architecture for nuclei segmentation in medical images, showcasing its potential for automating critical tasks in healthcare.
  - The model demonstrated commendable accuracy in segmenting nuclei, laying the foundation for further advancements in medical image analysis. The journey through data preprocessing, model training, and evaluation highlighted the intricacies involved in creating robust segmentation models.
- **Future Enhancements**
  - Looking ahead, there are several avenues for improvement and exploration.
  - Fine-tuning hyper-parameters, experimenting with **alternative architectures**, and incorporating advanced techniques for medical image segmentation present exciting opportunities.
  - By continuously refining the model and exploring cutting-edge methodologies, we aim to push the boundaries of segmentation accuracy and contribute to the ongoing advancements in the field of medical imaging.

References

References

The following references were used by the team for preparing the presentation:

- **Research Paper :** U-Net: Convolutional Networks for Biomedical Image Segmentation, https://arxiv.org/abs/1505.04597
- **Article :** Image Segmentation with U-Net, https://www.analyticsvidhya.com/blog/2022/10/image-segmentation-with-u-net/
- **Dataset :** 2018 Data Science Bowl, https://www.kaggle.com/c/data-science-bowl-2018/data
- **Kaggle Documentation :** Unet +Augmentation+segmentation models, https://www.kaggle.com/code/capecrusader27/unet-augmentation-segmentation-models
- **YouTube: DigitalSreeni YouTube Channel,** https://www.youtube.com/@DigitalSreeni
- Class Notes and other research papers related to the field.

THE END