A Deep Learning-based Implementation of U-Net: Convolutional Networks for Biomedical Image Segmentation

*Project Report for *Indian Institute of Technology, Bombay* - DS 303: Introduction to Machine Learning (2023)

K. Solanki
Department of Physics
kandarp.solanki@iitb.ac.in

S. Suresh
Department of Physics
210260046@iitb.ac.in

H. Shah
Department of MEMS
21d180016@iitb.ac.in

***Abstract*—A successful deep-learning model requires a huge amount of dataset for training usually, but data augmentation these days has emerged to be a helpful tool to effectively reduce the dataset used by giving small perturbations to our small dataset and making it invariant with respect to these changes. This augmented data could be thought or modelled as some natural phenomena which causes variance in the data taken. This is especially helpful when it comes to biomedical imaging. The U-Net model architecture uses symmetric contraction and expanding paths to capture contexts and accurate localization at the same time.**

I. Introduction

A. What is Biomedical Imaging and Biomedical Image Segmentation?

Biomedical imaging is a broad field that encompasses a range of imaging modalities used to visualize the internal structures and functions of the human body. These modalities include techniques such as X-ray, computed tomography (CT), magnetic resonance imaging (MRI), ultrasound, and positron emission tomography (PET). Biomedical imaging has revolutionized the diagnosis and treatment of various diseases, providing doctors and researchers with an unprecedented view of the body's inner workings.

Biomedical image segmentation is a critical step in the analysis of biomedical images. It involves dividing an image into multiple regions or segments based on various criteria, such as tissue type, shape, or texture. Biomedical image segmentation has applications in many fields, including medical diagnosis, image-guided therapy, and biomedical research. Accurate segmentation of biomedical images is essential for identifying and quantifying abnormalities, tracking disease progression, and developing new treatment strategies.

B. How does deep-learning help here?

Deep learning has emerged as a powerful tool in biomedical imaging and image segmentation. Deep learning algorithms, such as convolutional neural networks (CNNs), can automatically learn hierarchical representations of image features, allowing for more accurate and efficient segmentation of biomedical images.

CNNs are particularly effective in biomedical image analysis because they can handle large amounts of image data and learn complex features without requiring manual feature engineering. They can also adapt to new data and provide accurate predictions, making them valuable for applications such as medical diagnosis and treatment planning.

C. U-Net Model Architecture

We will be using a special type of architecture known as U-Net built specifically to train data faster and with the same or even better accuracy with relatively lesser amount of data, which is usually difficult to find. This will be more or less achieved by augmenting the data and making it prone to some perturbations in the data fed to it as input. Moreover, our network will be a fully convolutional network with no fully-connected layers but just 2-D convolutions, deconvolutions and max-pooling (upsampling operators to be precise).

FLOWCHART

Apply 3x3 convolution (our modified version includes Batch Normalization)
↓
ReLU (Rectified Linear Unit) after each convolution
↓
2x2 max-pooling with stride 2 for downsampling
↓
Repeat ... until features = 1024
↓
Upsampling by 2x2 convolutions (transpose) makes features half
↓
Concatenate with the feature map of same size in downsampling path
↓
Two 3x3 convolutions followed by ReLU
↓
1x1 convolutions, maps n-dimensional feature vector to desired number of classes

II. DATASET AND METHODOLOGY

The advantage of the U-net model is that it can be expanded to a variety of other datasets beyond biomedical imaging. To demonstrate its functionality, the dataset we used for our project is the Carvana dataset, sourced from Kaggle. This dataset contains a large number of car images (as .jpg files). Each car has exactly 16 images, each one taken at different angles. Each car has a unique id and images are named accordingly in chronological order from 1 to 16. We are also provided with basic information about each car, such as the year, make, model and trim. Besides this, the dataset contains the manually cutout mask for each image, pertaining to the training data. In machine learning, a mask is an image where some pixels of the original image are set to zero. The original image is modified either by changing the background or hiding some parts of the image with text, another image, cropping or editing. Masking thus plays a crucial role in enhancing the visibility of the image in a non-destructive manner, by maintaining the originality of the image.

Overall, the contents of the dataset utilised are:

- /train/ - This folder contains the training set images
- /train_masks/ - this folder contains the training set masks in .gif format
- train_masks.csv - for convenience, this files gives a run-length encoded version of the training set masks.
- metadata.csv - contains basic information about all the cars in the dataset. Some values are missing.

This data has been sourced from kaggle.com and it is free to use.

III. BUILDING THE MODEL

First, we built the machine learning model. The U-net architecture looks as follows:

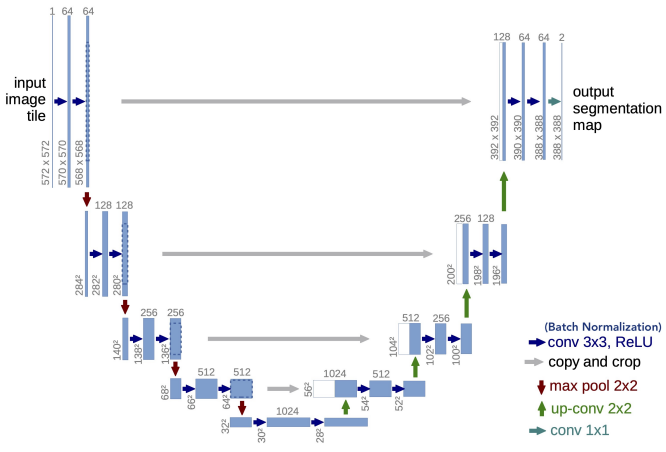

Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution)

This is a modification of a so-called 'fully convolutional network' such that it performs adequately well with very few training images and still produces precise segmentations. Each blue box corresponds to a multi-channel feature map. The number of channels is mentioned on each box. The white boxes represent copied feature maps. The main idea behind this architecture is to supplement a usual contracting network by successive layers, where pooling operators are replaced by upsampling operators. Hence, these layers increase the resolution of the output.

One important modification in this architecture is that in the upsampling part we have also a large number of feature channels, which allow the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path, and yields a U-shaped architecture. The network does not have any fully connected layers and only uses the valid part of each convolution. In other words, the segmentation map only ontains the pixels, while the full context is available in the input image. This strategy allows the seamless segmentation of arbitrarily large images by an overlap-tile strategy.

The network architecture is comprised 23 layers in total. It starts with a contracting followed by an expansive path. The contracting path represents a typical convolutional network, consisting of repetitions of successive 3X3 convolutions and batch normalisation, which is eventually succeeded by a rectified linear unit (ReLU). This is succeeded by a max pooling operation and a downsampling using a stride of 2. The expansive path uses an upsampling of the feature map at every step. A 2X2 convolution is concatenated with a batch normalisation to result in a resizing of the feature space. This resulting alteration in the feature space is necessary since border pixels are lost with every convolution. Finally, a 1X1 convolution layer is used to map each 64-component feature vector to the required number of classes. This forms the complete U-net architecture.

IV. OVERVIEW OF FILES

We created four files named

- `Carvana.py`
- `train.py`
- `U_net.py`
- `utils.py`

The **Carvana.py** file contains a function named **Carvana-Dataset** which inputs images and returns augmented images and masks in the form of numpy arrays

The **train.py** file trains the model with all the hyperparameters like learning rate, device, batch-size, epochs, etc specified in it. When we run the model, it gives a real-time model training status with the number of epochs, accuracy and a dice-score.

The **U_net.py** file contains the definition to the architecture of the model. The model is slightly modified to improve the performance in the following ways:

- Usage of **Batch Normalization** in the double convolution layer which allows us to use much **faster learning rates**
- **Concatenation** using **resizing** instead of centre crop so that **dimensions are not disturbed during upsampling**

The **utils.py** file contains some user-defined collection of utilities/definitions that are commonly used in various tasks such as data preprocessing, model training, and evaluation. We used it for the following:

- **Data loading** and **preprocessing** functions, such as loading images, resizing, normalization, and data augmentation.
- **Model creation** and **training functions**, such as creating model architectures, defining loss functions, and setting up training and validation loops.
- Evaluation functions, such as calculating accuracy, precision, recall, and F1 score. We used only **accuracy**.

V. Training & Results

After training the model and validating it with about 20% of the total dataset chosen randomly, the final results are as follows:

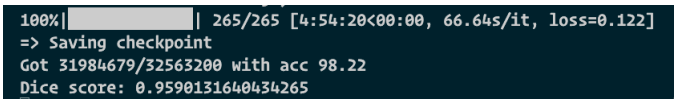

Fig. 2. Training Results

The model was trained in nearly 5 hours and ended up with an accuracy of **98.22%** at the finish with a loss valued at **0.122** which is quite good. The loss function we used is

BCEWithLogitsLoss which combines a Sigmoid layer and the BCELoss in one single class. This is more numerically stable than using a plain **Sigmoid** followed by a **BCELoss** as, by combining the operations into one layer, we take advantage of the log-sum-exp trick for numerical stability.
Mathematically the loss function is given by:

$$l(x, y) = L = \{l_1, l_2, ...l_N\}^T,$$
$$l_n = -w_n[y_n \cdot log\sigma(x_n) + (1 - y_n) \cdot log(1 - \sigma(x_n))]$$

References

[1] stackoverflow.com
[2] kaggle.com (for dataset and analysis ideas)
[3] PyTorch Documentation - https://pytorch.org/docs/stable/index.html
[4] U-Net: Convolutional Networks for Biomedical Image Segmentation by Olaf Ronneberger, Philipp Fischer, and Thomas Brox