In the name of God

Sharif University of Tech

Electrical Engineering Department

Digital Image Processing Final Project

Prof: Dr.poreh

By:

Sajjad Hashembeiki

Student Number :9810707

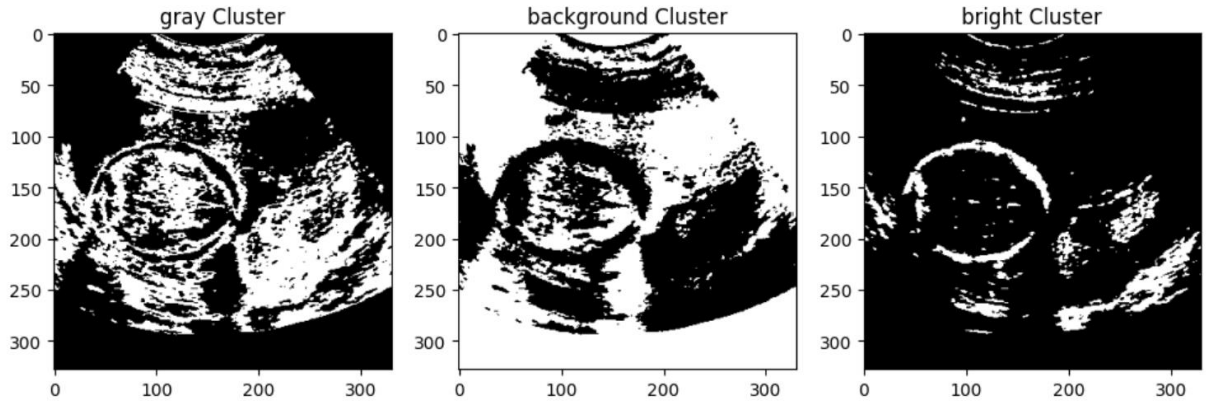
July 2023

Classical Methods

Problem1

In this part I clustered the example image by K-means. After the clustering I assign 1 and 0 to each pixel by considering the labels that obtained from the clustering method. So there are 3 binary images that belong to 3 clusters: **gray**, **background** and **bright**.

As we can see the **bright** cluster contains the fetal head.



Problem2

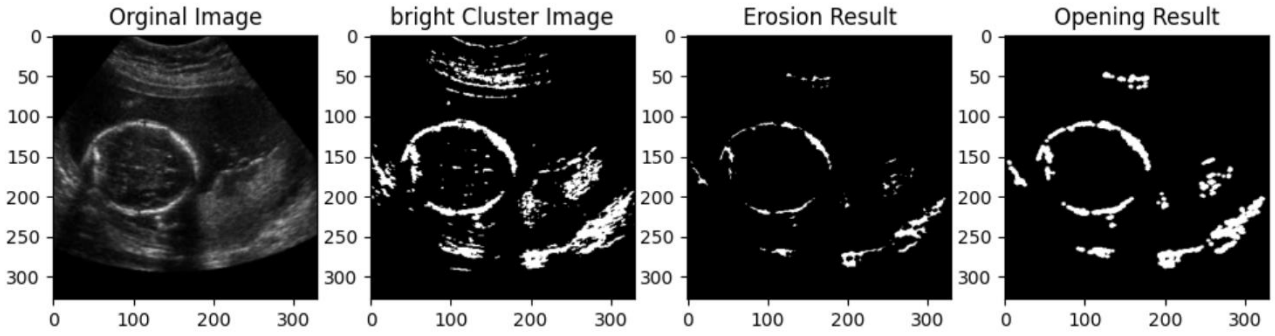
Now I perform Morphological methods on the binary bright image which obtained in previous part to remove small objects.

At first I test several method to find the best one. Finally I performed Opening method on the bright cluster and after that applied the skeletonize method.

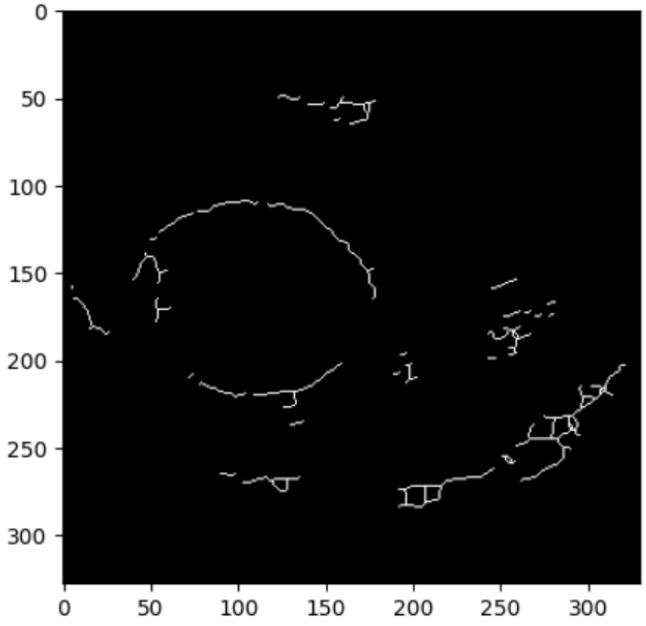
This method improves the ellipse fitting that will be perform by Hough transform.

For create the structure I used getStructure method and create an ellipse structure, Also the cross structure is useful and the result of this looks like the ellipse structure.

The result of the morphological methods are as follows:



Result of applying the skeletonize on the Opening result:



Problem3

After applying the morphological methods, I use iterative randomized Hough transform in order to fit the best ellipse in the image.

Here is a full description of the algorithm:

First the input image should be preprocessed to enhance edges or relevant features that correspond to ellipses. That is what I did in previous part by applying morphological techniques .

Random Point Selection: A random subset of points is selected from the preprocessed image. The number of points chosen depends on the desired trade-off between computational efficiency and accuracy. These points are selected uniformly at random or using other sampling techniques.

Parameter Space: Instead of the traditional parameter space used for line detection, ellipse fitting requires a different set of parameters. The parameters used for ellipse fitting include the center coordinates (x_c, y_c), major axis length (a), minor axis length (b), and orientation (θ). These parameters define an ellipse in the image.

Hough Transform Accumulation: For each randomly selected point (x, y), the algorithm computes the corresponding ellipse parameters (x_c, y_c, a, b, θ). The parameters are quantized to form bins in the parameter space. Each bin accumulates votes based on the presence of other points that could contribute to the same ellipse. The voting process can be as simple as incrementing the corresponding bin value.

Thresholding: Once all the points have been processed, a thresholding step is applied to identify the bins in the parameter space that have accumulated a significant number of votes.

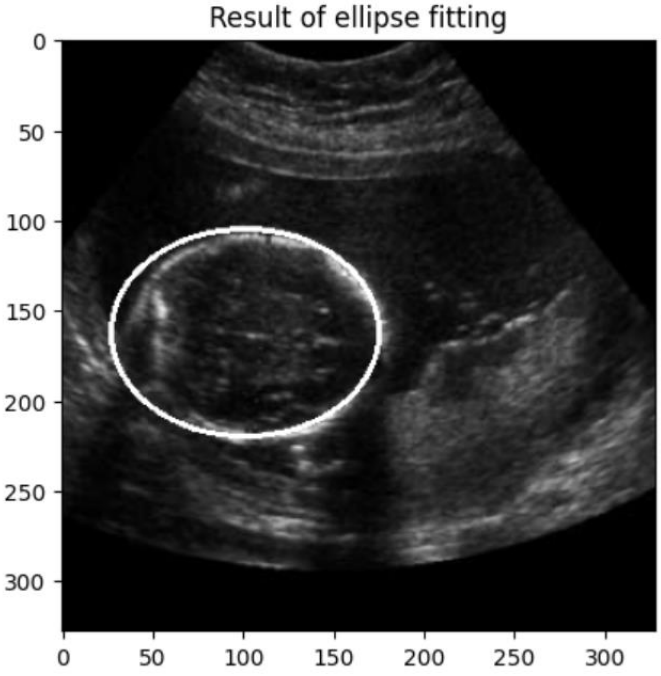
The threshold value is chosen empirically or using heuristics and controls the sensitivity of the algorithm.

Ellipse Extraction: The identified bins above the threshold represent the detected ellipses. The corresponding parameters can be converted back to image space coordinates to obtain the ellipses found in the original image. Techniques like interpolation or least squares fitting can be used to refine the ellipse parameters based on the votes accumulated in the Hough Transform.

The Randomized Hough Transform for ellipse fitting provides a computationally efficient method for detecting and fitting ellipses in an image. By randomly selecting a subset of points and performing the Hough Transform only on these points, the algorithm reduces the computational complexity while maintaining accuracy. However, as with other randomized approaches, there is a trade-off between efficiency and accuracy, and the algorithm may miss less prominent ellipses or produce slightly different results on each run due to the random sampling.

Problem4

The result of the Hough transform on the skeletonize opening image:



Idea to improve this algorithm:

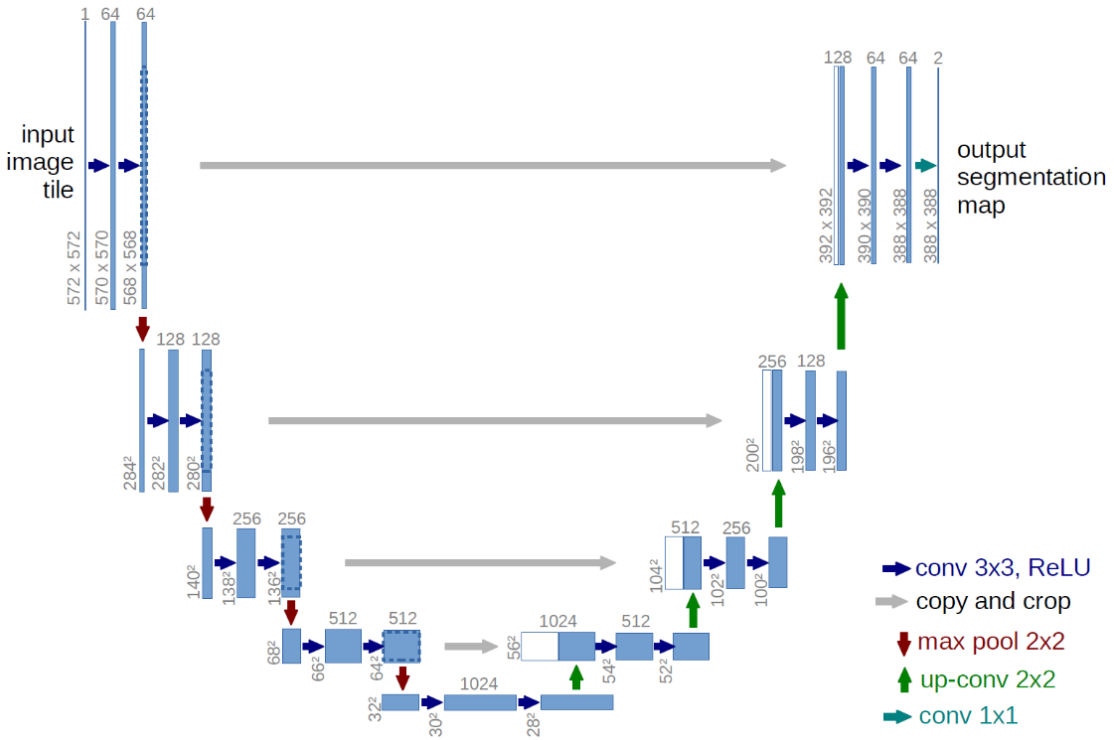
I think performing edge detectors like canny is useful, also for reduce the computationally cost we can use image pyramids like Gaussian pyramids and Laplacian pyramids.

By down sampling and reduce the image resolution the algorithm will be run more efficient and faster.

Deep learning Methods

Problem5

5.1-Description of the U-net



Fugure5.1

The U-Net is a convolutional neural network (CNN) architecture commonly used for image segmentation tasks.

The U-Net architecture is characterized by its U-shaped design, which consists of an encoder path and a decoder path. The encoder path captures the context and spatial information from the input image, while the decoder path performs up-sampling and reconstructs the segmented image.

Here's a detailed description of the U-Net architecture:

1. Encoder Path:

- The input to the U-Net is an image, typically of variable size.

- The encoder path starts with a series of convolutional layers followed by rectified linear unit (ReLU) activations, which extract features from the input image.

- These convolutional layers are typically configured to have a small filter size (e.g., 3x3) and are followed by max pooling layers with stride 2, which down-sample the feature maps and increase their receptive field.

2. Bridge:

- The bridge connects the encoder and decoder paths and helps in preserving the spatial information.

- It consists of a stack of convolutional layers with the same configuration as the encoder path, typically without the max pooling layers. This ensures that the spatial resolution of the feature maps remains unchanged.

3. Decoder Path:

- The decoder path begins with up-sampling operations, such as transposed convolutions or bilinear up-sampling, which increase the spatial resolution of the feature maps.

- Concatenation: At each up-sampling step, the feature maps from the corresponding encoder path level are concatenated with the up-sampled feature maps.

- The concatenated feature maps are then passed through a series of convolutional layers with ReLU activations.

4. Output:

- The final layer of the U-Net is a 1x1 convolutional layer followed by a sigmoid or softmax activation function, depending on the number of classes for segmentation.

- This layer produces the segmented image or probability map, where each pixel represents the probability of belonging to a particular class or being part of the segmented object.

5.2-Preprocessig methods

As preprocessing first of all I resize the images in to 128*128 for the implemented CNN.

Also after resizing the images I normalized them. As we know normalizing the pixel values of the images can help in reducing the effects of lighting variations and improve the convergence of the network during training.

Common normalization techniques include subtracting the mean and dividing by the standard deviation of the image dataset or scaling the pixel values to a specific range, such as [0, 1]. I scaled the images into [0,1] by dividing the pixels by 255.

Problem6

After data preprocessing that mentioned in detail in previous part now I implement U-net by Keras. The structure of the network is mentioned completely in previous part and you can see the layer architecture in figure5.1.

For training the model I use 80% of the dataset as training and the 20% for test set.

I set the batch size for 16 as mentioned in the article and also use Adam optimizer for training. The loss function that I use is binary-cross-entropy.

This model is trained in 25 epochs and as you see in the figure6.1 the accuracy of the model is about 88.97% for test set and 88.7% for validation set.

The learning rate is set to 10^{-4} .

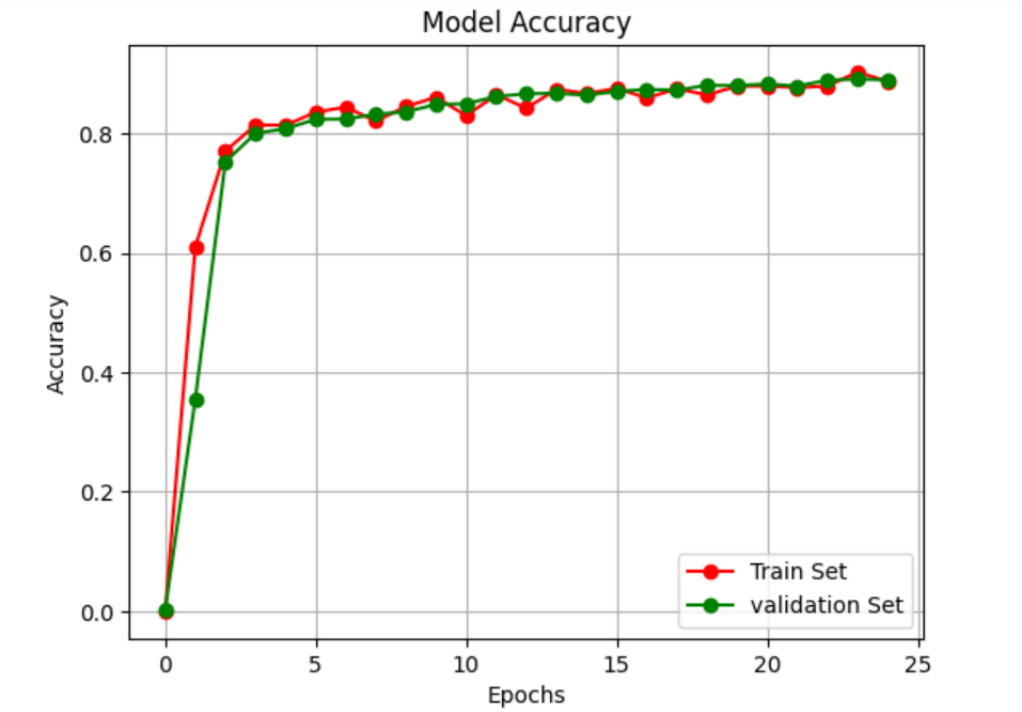
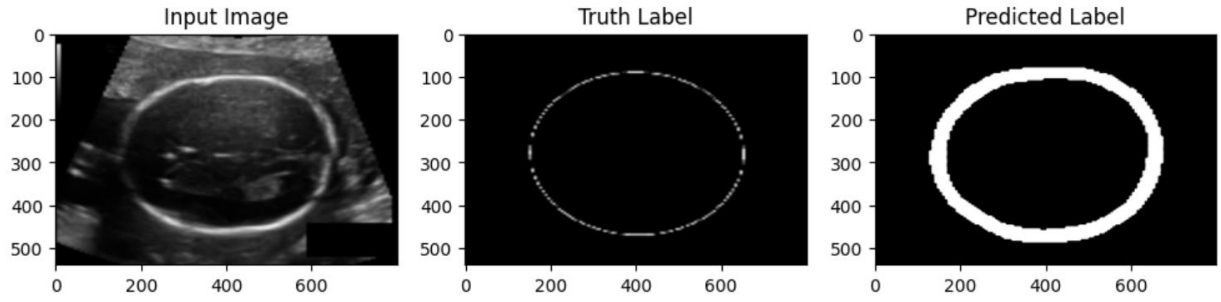


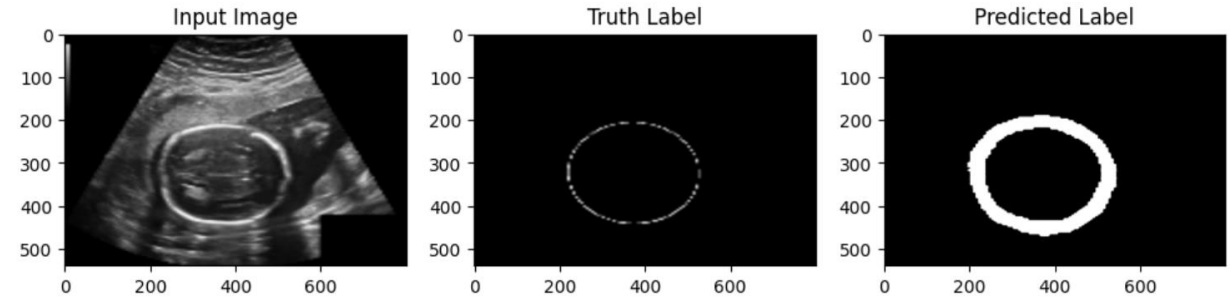
Figure6.1

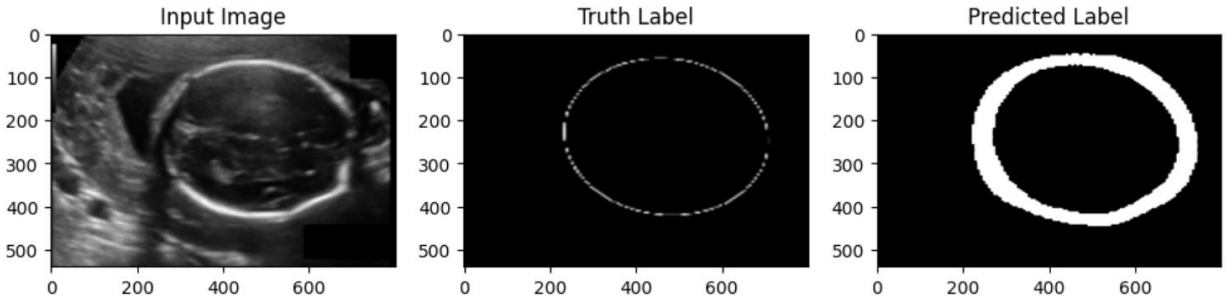
The result of the model for random datapoint are as follows:

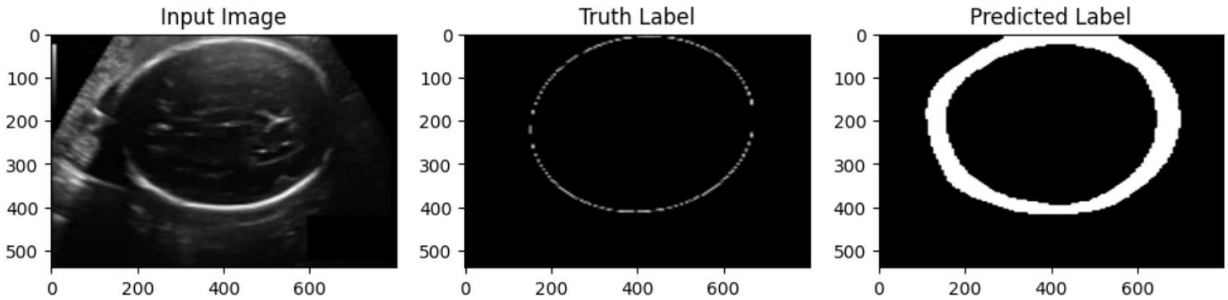
A randomly selected datapoint from training set:



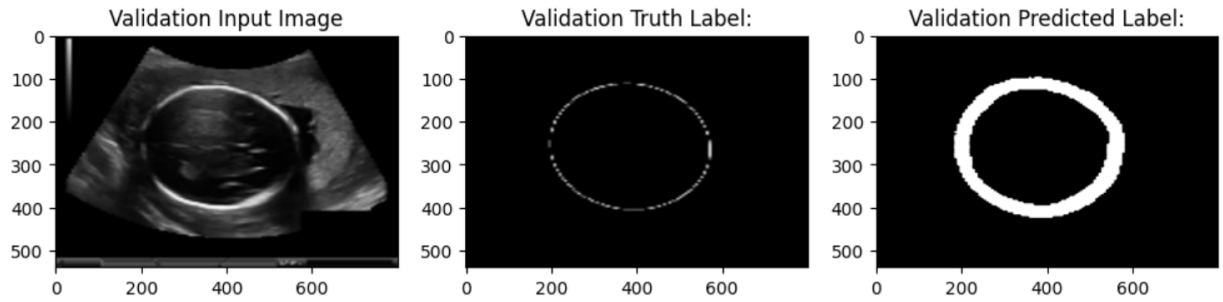
Examples from test set:





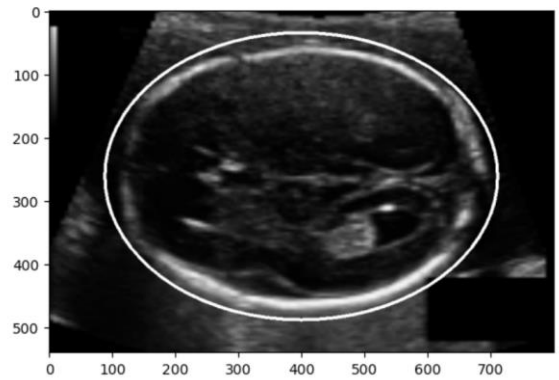


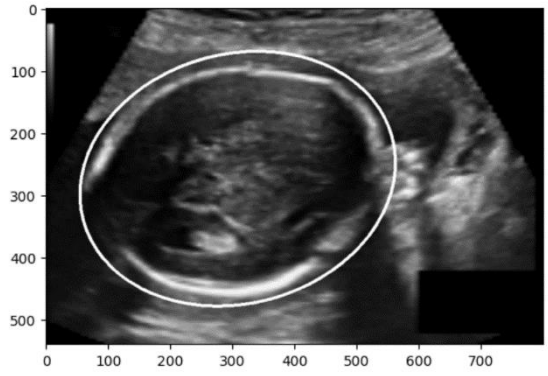
And one from validation set:

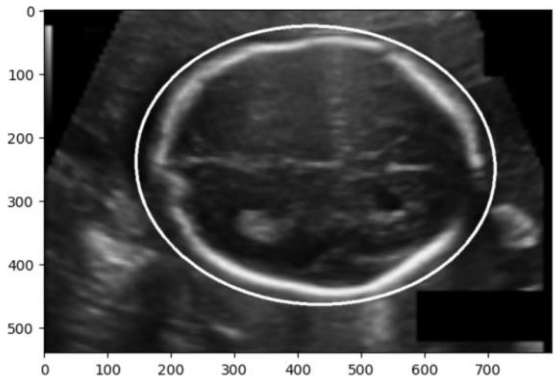


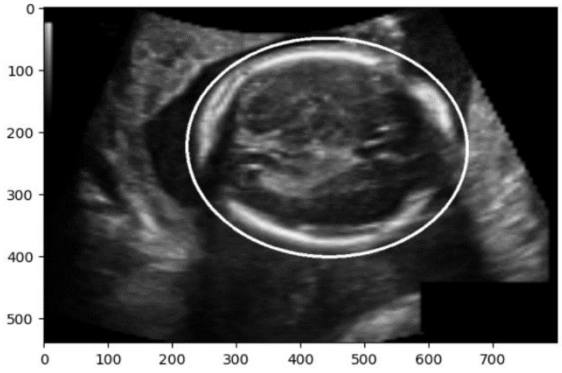
Problem7

In this section I use RHT to fit ellipse to the predicted labels. The results on some test images are as follows:









Problem8

8.1-description of the VGG16

The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer.

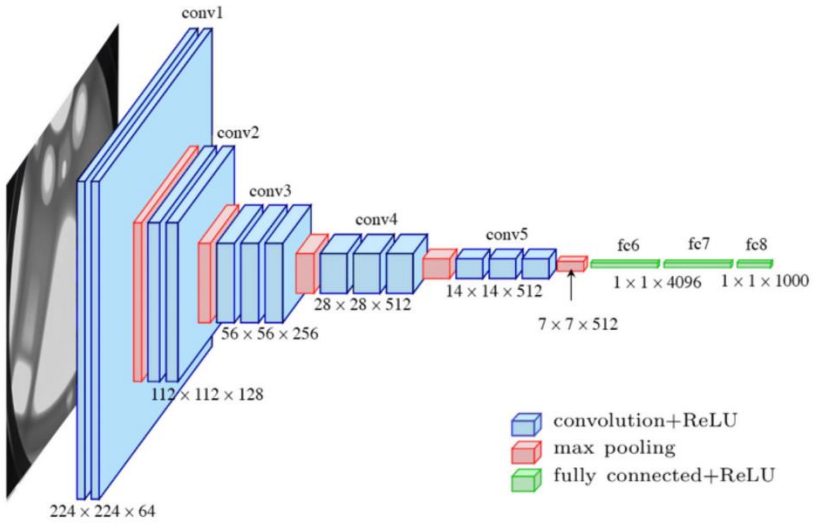


Figure8.1

VGG16 takes input tensor size as 224, 244 with 3 RGB channel.

Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2.

Figure8.2

The convolution and max pool layers are consistently arranged throughout the whole architecture

Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters.

Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

8.2-preprocessing method:

As preprocessing first of all I normalized the HC values by divide them by the max value thus the HC values are in [0,1].

Also I resize the images in to 224*224 because of the structure of the VGG16.

8.3-Results:

In dense layers I use Relu as activation function and finally a linear layer.

For training the model I use Adam optimizer and MSE as a loss function.

I split the data into test and train set.80% for train and 20% for test

I use the pretrained model on the ImageNet. Because of the downloading the weights I used google colab for training and for the instability of the internet I'm sorry but I couldn't train the network completely. The results are for just 3 epochs.

The predicted HC values for test set are as follows:

	head circumference (mm)	HC preds (mm)
722	0.535797	0.403102
77	0.201386	0.275388
877	0.723441	0.420933
613	0.536114	0.465001
903	0.800837	0.582872
...	...	...
150	0.266628	0.619085
299	0.446016	0.493073
239	0.439954	0.603380
981	0.927540	0.675548
435	0.520987	0.531651

200 rows × 2 columns

For example the second datapoint values are 0.20 mm for the truth value and 0.27 mm for predicted one. Note: the values are normalized by dividing by max value.

Problem9

The classical method finds the ellipse and HC good enough but the computational cost of the Hough transform is the disadvantage of this method, also the preprocessing of the image and find the suitable thresholds and methods are hard and this is a big defect of the method.

On the other hand, the U-net approach is so accurate and very faster and easier for applying. The second deep learning approach, regression CNN, needs more time and computational cost for training but the U-net is so easy to train. I train the U-net om my local system and got the accuracy of about 90% just after 25 epochs and less than several minutes.

Segmentation-based methods provide interpretable results for the HC estimation because the segmentation result is visible, they often require dedicated post-processing steps. On the other hand, regression approaches based on CNN are end-to-end, less costly, and prone to error and even though they do not offer explicit interpretability.

Bunos Part

Problem10

Generative Adversarial Networks (GANs) and Pix2Pix networks are both deep learning architectures that have made significant contributions to the field of image generation and image-to-image translation tasks. Let's delve into each of them:

Generative Adversarial Networks (GANs):

GANs are a class of generative models. GANs consist of two key components: a generator network and a discriminator network, which are trained simultaneously in an adversarial manner.

The generator network takes random noise as input and tries to generate synthetic data samples (e.g., images) that resemble real data from a given distribution. The discriminator network, on the other hand, acts as a binary classifier, distinguishing between real and generated samples. The goal of the generator is to fool the discriminator into classifying its generated samples as real, while the discriminator aims to correctly classify real and generated samples.

During training, the generator and discriminator play a game against each other. The generator tries to improve its ability to generate realistic samples by learning from the feedback of the discriminator, while the discriminator gets better at distinguishing between real and generated samples. This adversarial training process leads to a competition that eventually pushes the generator to generate increasingly realistic samples.

Pix2Pix Network:

Pix2Pix is a specific type of GAN architecture proposed by Phillip Isola et al. in 2016, specifically designed for image-to-image translation tasks. While traditional GANs generate new images from random noise, Pix2Pix networks learn to translate input images into corresponding output images.

The Pix2Pix network comprises two main components: an encoder-decoder generator network and a discriminator network. Unlike the typical GAN architecture, the generator in Pix2Pix is an encoder-decoder network that takes an input image and produces an output image of the same size.

The encoder part of the generator encodes the input image into a lower-dimensional representation, while the decoder part reconstructs the output image from the encoded representation.

The discriminator network in Pix2Pix is similar to the discriminator in GANs, but with a modification. It takes pairs of images as input either a real input-output image pair or a generated input-output image pair and attempts to distinguish between the two. The discriminator's goal is to correctly identify whether the pairs are real or generated.

The training of Pix2Pix networks involves an adversarial loss, which encourages the generator to produce output images that are indistinguishable from the real ones, as well as a pixel-wise L1 or L2 loss that measures the similarity between the generated output and the ground truth output.

10.2-Results on Satellite-Map dataset