Report on Automated Measurement of Fetal Head Circumference

Nguyen Cong Quoc

March 20, 2024

Abstract

Measuring the fetal head circumference (HC) is an important technique for assessing gestational age and fetal development. In this work, we present an automated pipeline based on the U-net architecture for computing the HC from two-dimensional ultrasound images. Additionally, we implement the nnU-net model, which achieves state-of-the-art performance in HC measurement on the test set of the HC18 challenge. We discuss and analyze the performance of both models and describe the methods employed to achieve these results.

1 Introduction

The fetal head circumference (HC) is an important measurement used to monitor gestational age and fetal growth. In this report, we present an implementation of a deep learning model to automate the measurement of HC from 2D ultrasound images. The goal is to predict the HC in millimeters (mm) from the input ultrasound images.

2 Dataset

The dataset used in this project is provided by the HC18 Grand Challenge website (https://hc18.grand-challenge.org/). It consists of 1334 two-dimensional (2D) ultrasound images of the standard plane of fetuses. The dataset is split into 999 training images and 335 test images.

Each image in the dataset has a resolution of 800 by 540 pixels, and the pixel sizes range between 0.052 and 0.326 mm. The dataset also includes a CSV file (`training_set_pixel_size_and_HC.csv`) that provides additional information about each image, such as the filename, pixel size, and the ground truth head circumference in millimeters.

3 Data Preprocessing

The data preprocessing steps are as follows:

1. Load the CSV file containing annotations for the training set.
2. Iterate through each row in the CSV file:
    - Load the corresponding image from the provided path.
    - Resize the image to a fixed size of 800 by 540 pixels.
    - Append the resized image and the corresponding head circumference value to separate lists.
3. Convert the lists of images and head circumference values to NumPy arrays.
4. Normalize the pixel values of the images to the range [0, 1].
5. Split the dataset into training and validation sets using the train$_{t}est_{s}plitfunctionfromscikit - learn.$

4 Model Implementation

The model is implemented using the TensorFlow and Keras libraries in Python. It is a convolutional neural network (CNN) with the following architecture:

- Convolutional layer with 16 filters of size 3x3, ReLU activation
- Max pooling layer with a 2x2 window
- Convolutional layer with 32 filters of size 3x3, ReLU activation
- Max pooling layer with a 2x2 window
- Convolutional layer with 64 filters of size 3x3, ReLU activation
- Flatten layer
- Fully connected layer with 64 units, ReLU activation, and 40% dropout
- Output layer with a single unit (for regression)

The model is compiled with the Adam optimizer and mean squared error loss function. The mean absolute error (MAE) is also tracked as a metric during training.

5 Training and Evaluation

The model is trained for 10 epochs with a batch size of 32, using the training and validation sets created during data preprocessing. The training and validation loss, as well as the MAE, are monitored during training.

After training, the model is evaluated on the validation set, and the validation MAE is reported.

6 Results and Visualization

The training and validation loss, as well as the MAE, are plotted to visualize the model's performance during training.

Additionally, a plot is generated to compare the predicted head circumference values with the actual ground truth values for a subset of the validation set.

The mean absolute error (MAE) between the predicted and actual head circumference values is calculated and reported.

Finally, a histogram is plotted to visualize the distribution of prediction errors (difference between predicted and actual values).