Final Project Report (December, 2021)

Ankita Ghosh[1] and Sahil Khose[2]

[1]Research Assistant, ghoshankita0907@gmail.com , CSE, MIT Manipal
[2]Research Assistant, sahilkhose18@gmail.com, ICT, MIT Manipal

ABSTRACT

We discuss the results obtained for deep learning based solutions of two problen statements in the domain of opthalmology–fovea segmentation and macular degeneration classification

Fovea Segmentation

Data Pre-processing

- We have trained the model on five datasets, total of **484 datapoints** to perform binary segmentation and extract the fovea:

    1. Drive: 40 images with ground truth
    2. Messidor: 180 images with ground truth
    3. IDRiD: 58 images with ground truth
    4. diaretdb0: 127 images with ground truth
    5. diaretdb1: 79 images with ground truth

- To expedite the process of feature extraction for the deep learning model, we apply **gaussian filter**, **binary thresholding** and **morphological open** to the mask.
- The images and masks are resized to **512 × 512 dimensions** while training to strike a balance between processing efficiency gained by the lower dimensional images and information retrieval of the high-resolution images.
- We also added **cropping** to increase the number of datapoints and for image **augmentation** we perform shuffling, rotation, scaling, shifting and brightness contrast.

Training

- **Loss**: We also added the Twersky Loss as a weighted loss along with BCE Loss for our final loss. We found that this converges the models to obtain a considerable improvement in our results. We fine tune the weights for the final loss formula through a grid search method, resulting in best weights to be **0.7 for Twersky** and **0.5 for BCE** and **1.5 for Gamma**.

    - **Binary Cross Entropy Loss**
      $L_{BCE} = y\log(y_{hat}) + (1-y)\log(1-y_{hat})$
    - **Dice Loss**
      $L_{Dice} = 1 - \frac{2TP}{2TP+FN+FP}$
    - **Tversky Loss**
      $L_{Tversky} = 1 - \frac{TP}{TP+\alpha FN+\beta FP}$ where, $\alpha + \beta = 1$
    - **Focal Tversky Loss**
      $L_{FTL} = (L_{Tversky})^{\gamma}$ where, γ controls the non-linearity of the loss.
    - **Final Loss**
      $L_f = \lambda L_{BCE} + (1-\lambda)L_{FTL}$ where, λ is the weight parameter.

- **Model**:
  - We train the **DeepLabV3+** model with **EfficientNet-B3** as the backbone.
  - For DeepLabV3+ we use **encoder depth of 5** which refers to the number of stages used in encoder. The number of **convolution filters (decoder channels) used is 256**.
  - Batch size was set to **8**. Learning rate is set at **5e-4**.
- **Semi-supervised learning**: We add **unlabelled Messidor data (1200 datapoints)** to the existing labelled data (484 datapoints) and trained it on a semi-supervised algorithm as shown in **Figure 1**.

Algorithm 1: Semi-supervised classification train loop

Input: Sample image
Output: Class of the given image
for $epoch \leftarrow 0$ **to** E **do**
  if $epoch < E_i^\alpha$ **then**
    $\alpha \leftarrow \alpha_i$
  else if $epoch < E_f^\alpha$ **then**
    $\alpha \leftarrow \frac{\alpha_f - \alpha_i}{E_f^\alpha - E_i^\alpha} * (epoch - E_i^\alpha) + \alpha_i$
  else
    $\alpha \leftarrow \alpha_f$
  end if
  Run the model on train set
  $loss \leftarrow BCE(l, \hat{l}) + \alpha * BCE(u_{epoch}, u_{epoch-1})$
  Generate the pseudo labels for unlabeled data
  Evaluate the model on validation set
end for

Figure 1. Semi-supervised Algorithm

Results

- We evaluate our results based on the **metrics**: Dice, Jaccard, Sensitivity, Specificity and Accuracy.
- **Table 1** shows a comparison between our model and the other methods based on the metrics stated above.

Method	Dice(F1score)	Jaccard(MIoU)	Sensitivity	Specificity	Accuracy
Traditional Method (non-DL)	0.8044	0.6881	0.8162	**0.9984**	0.996
Deep Learning (ours)	**0.8243**	**0.7052**	**0.9174**	0.9975	0.9957
Deep Learning (Tan et al)	-	-	0.8853	0.9914	-
Deep Learning (Sedai et al)	0.81	-	-	-	-

Table 1. Metrics Comparison

Macular Degeneration Classification

Data Augmentation

We implement online augmentation and offline augmentation to enhance the dataset.

- **Offline Augmentation**
  - Label 1 has only 58 datapoints. To increase the number of datapoints we perform different types of augmentation to generate and store more data.
  - Augmentation techniques include horizontal flip, vertical flip, brightness, contrast and rotation.
  - The label 1 data is increased four-folds thus making a total of 290 datapoints.
  - Fig 2 shows the dataset post augmentation. Datapoints across label 0, 1 and 2 are now almost equally distributed.

```
----------------Dataset-----------
IDRiD Testing Set  : 103
IDRiD Training Set  : 413
drive  : 31
messidor2  : 80
augData Label1  : 232
Total datapoints:  859


----------------Labels-----------
Label  0 :      295
Label  1 :      290
Label  2 :      274
Total datapoints:  859
```

Figure 2. Dataset post augmentation

- **Online Augmentation**
  We apply augmentation during training which introduces the model to more variations in the dataset so that the results are generalized.

Training

- **EfficientNet-B3** (12M parameters) is implemented to be trained for this 3-class classification problem.
- We use pre-trained **ImageNet** weights and train the entire network over the dataset.
- The optimizer that we have used is **Adam optimizer** with a learning rate of **5e-4**.
- The batch size found optimal is **8**.

Result Comparison

- Table 2 and 3 demonstrates our results compared to other methods. Our model performs significantly better after addition of augmentation and pre-processing techniques.
- We obtain an accuracy of **93.6%** which surpasses majority of the methods listed in the table.

Method	No. of datapoints	Classes	Accuracy
Ours (augmentation)	627	3	**93.6%**
Zapata et al	306,302	2	86.3%
Gonzalez-Gonzalo et al	134,421	2	85.9%
Burlina et al	133,821	2	91.6%
Grassmann et al	120,656	13	63.6%
Bhuiyan et al	116,875	4	**96.1%**
Govindaiah et al	116,875	4	86.1%
Ting et al	108,558	2	88.8%
Keenan et al	59,812	2	**96.5%**
Peng et al	59,302	6	67.1%
Keel et al	56,113	2	**96.5%**
Burlina et al	5664	4	79.4%
Phan et al	279	2	87.7%
Kankanahalli et al	2772	3	81.8%
Mookiah et al	784	4	90.2%

Table 2. Classification Results

Method	Accuracy	No. of datapoints	Classes
Ours	**93.6%**	627	3
Burlina et al	79.4%	5664	4
Govindaiah et al	86.1%	116,875	4
Kankanahalli et al	81.8%	2772	3
Mookiah et al	90.2%	784	4
Bhuiyan et al	**96.1%**	116,875	4

Table 3. Classification Results (3 or 4 number of classes)

Discussion

- This report gives a summary of our approach and results which we will detail in our manuscript draft.
- We will try to train segmentation and classification in a multi-task learning setup to observe enhancement in performance and build upon our current results as a future project. Compute remains a concern.