Supplementary Material: Contrastive Learning for Generating Optical Coherence Tomography Images of the Retina

Sinan Kaplan[1 [0000−0002−0849−934X]] and Lasse Lensu[1 [0000−0002−7691−121X]]

[1]Department of Computational Engineering, Lappeenranta-Lahti University of Technology LUT, P.O. Box 20, 53850 Lappeenranta, Finland
sinan.kaplan@student.lut.fi, lasse.lensu@lut.fi

1 Introduction

In this supplementary material, we provide detailed information regarding the data used experiments, training environment and hyperparameters. In addition, we present randomly generated samples from each disease category explained in the original paper. The randomly generated samples can also be found on Github[1] alongside the source code.

2 Dataset details

This section provides information about the data used for training. It covers number of instances per class and aspect ratio stats. The data is collected from patients from the Shiley Eye Institute of the University of California San Diego, the California Retinal Research Foundation, Medical Center Ophthalmology Associates, the Shanghai First People's Hospital, and Beijing Tongren Eye Center between July 1, 2013 and March 1, 2017 [1]. As the images in the set are acquired by different cameras with different orientations, there is considerable amount of variation in the set.

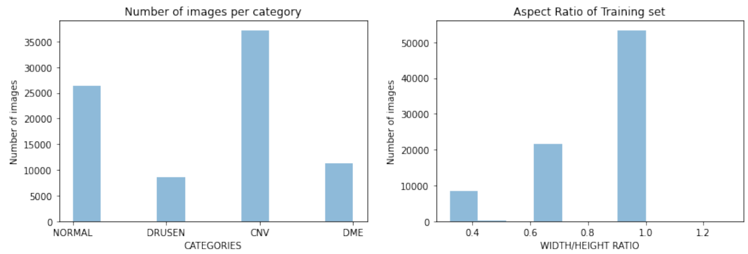
2.1 Raw data details

The data stats without any pre-processing are shown in Figure 1. We demonstrate number of samples per each class and aspect ration across all the set.

2.2 Pre-processed data details

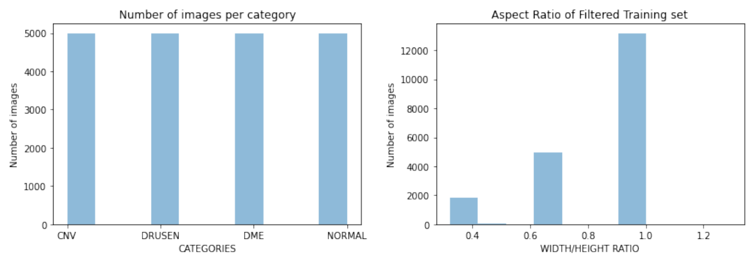
The training set is further filtered by representative sampling to solve issue of class imbalance. Figure 2 demonstrates the number of samples per each class and the aspect ratio across the filtered set. Also, we provide the algorithm used for representative sampling in Algorithm 4 in the appendix.

[1] https://github.com/kaplansinan/OCTRetImageGen_CLcVAE



(a) (b)

Fig. 1: Raw data statistics: (a) number of samples per each category; (b) the histogram of the aspect ratio of images.



(a) (b)

Fig. 2: Filtered data statistics: (a) number of samples per each category;(b) the histogram of the aspect ratio of images.

3 Training details and hyperparameters

We choose Tensorflow framework to train models on a computer with NVIDIA GeForce 3060 6GB GPU, Ubuntu 20.14. Training hyperparameters are given in Figure 3.

4 Randomly generated samples for each class

In this section, randomly generated samples from each class are presented in figures 4 (CNV), 5 (DME) and 6 (DRUSEN), respectively.

References

1. Kermany, D., Zhang, K., Goldbaum, M., et al.: Labeled optical coherence tomography (oct) and chest x-ray images for classification. Mendeley data **2**(2) (2018)

```
training_parameters = {
    # stage-1 contrastive learning training details
    "contrastive_learning_model": [{
        'input_image_size': (256, 256, 3),
        'output_vector_size': (1, 128),
        'epochs': 50,
        'batch_size': 8,
        'optimizer': [{
            'name': 'Adam',
            'learning_rate': 0.001,
            'beta_1': 0.9,
            'beta_2': 0.99
        }],
    }],
    # stage-2 conditional variational autoencoder details
    "cvae_model": [{
        'input_image_size': (256, 256, 3),
        'input_conditional_vector_size': (1, 128),
        'output_image_size': (256, 256, 3),
        'latent_dimension_size': (1, 128),
        'epochs': 150,
        'batch_size': 4,
        'optimizer': [{
            'name': 'Adam',
            'learning_rate': 0.001,
            'beta_1': 0.9,
            'beta_2': 0.99
        }],
    }]
}
```

Fig. 3: Training hyperparameters

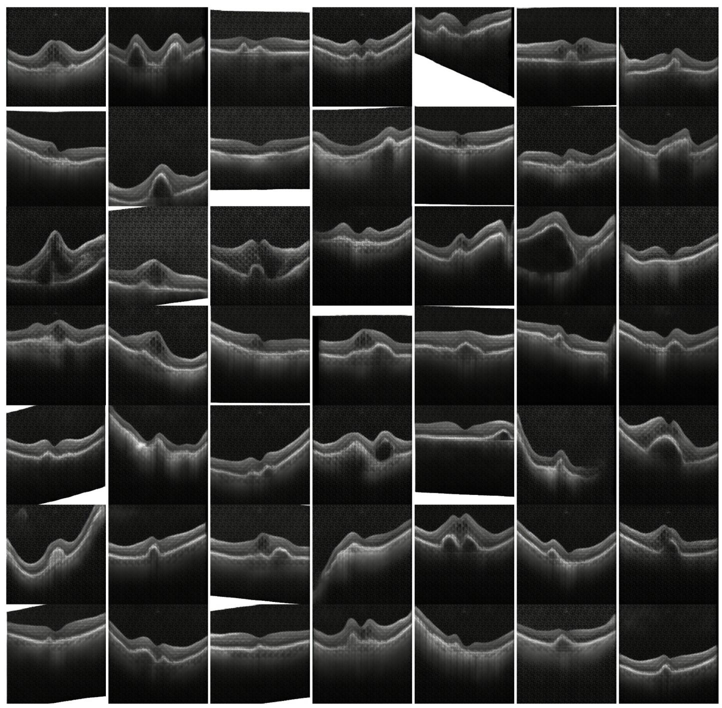


Fig. 4: Randomly generated choroidal neovascularization (CNV) samples.

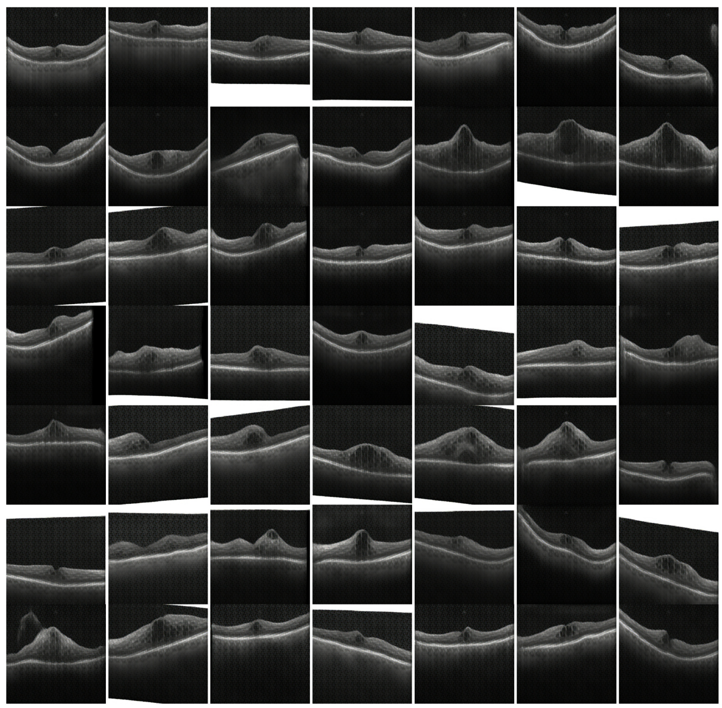


Fig. 5: Randomly generated diabetic macular edema (DME) samples.

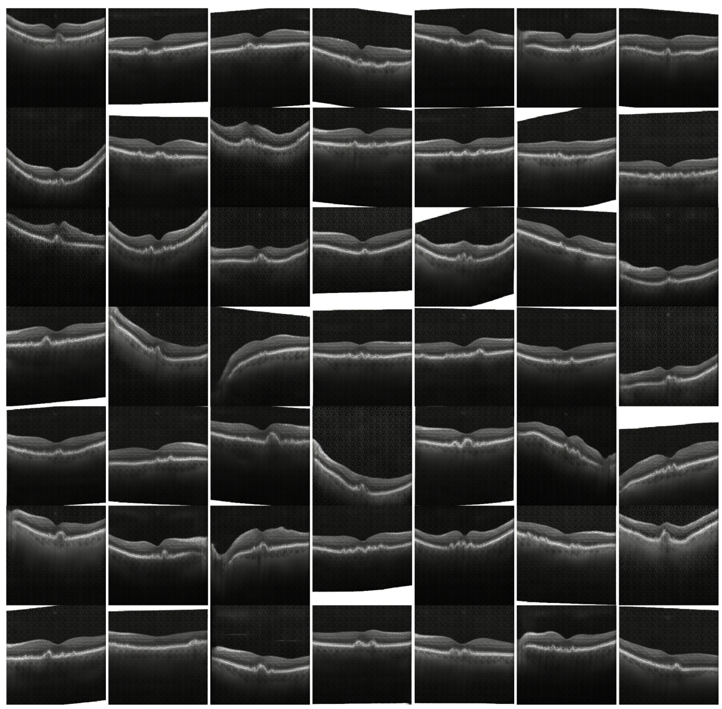


Fig. 6: Randomly generated DRUSEN samples.

Algorithm 1 Class Imbalance Solver (Representative Sampling)

```
DATASET = D
CLASS_LIST = ['NORMAL', 'DME', 'DRUSEN', 'CNV']
# resnet50 model trained on Imagenet
EMBEDDING_MODEL = RESNET50
#number of instances to be sampled from each class
DESIRED_NUMBER_OF_SAMPLES_PER_CLASS = N

def class_imbalance_solver():
    SAMPLE_LIST = []
    for class in CLASS_LIST:
        # get all samples of a class in dataset D
        data = get_class_samples(class, D)
        # extract embeddding features from the data
        data_emb_features = get_embedding_features(data, EMBEDDING_MODEL)
        # run hierarchical clustering over the embedding features
        clusters = run_hierarchical_clustering(data_emb_features)
        # sample representative samples from each cluster
        repr_samples = get_n_representative_samples(clusters,N)
        # add representative samples to sampling bucket
        SAMPLE_LIST.append(repr_samples)

    return SAMPLE_LIST
```