NCDE-Diagnosis: a practical CAD software based on the Hyper Kvasir Dataset

Qiyu Zhang
School of Science and Engineering
The Chinese University of Hong Kong, Shenzhen
Shenzhen, GD 518172
qiyuzhang@link.cuhk.edu.cn;

Zihan Zhou
School of Science and Engineering
The Chinese University of Hong Kong, Shenzhen
Shenzhen, GD 518172
zihanzhou@link.cuhk.edu.cn;

Longda Tong
School of Data Science
The Chinese University of Hong Kong, Shenzhen
Shenzhen, GD 518172
longdatong@link.cuhk.edu.cn;

Longxiang Li
School of Data Science
The Chinese University of Hong Kong, Shenzhen
Shenzhen, GD 518172
longxiangli@link.cuhk.edu.cn;

Xinyu Fang
School of Data Science
The Chinese University of Hong Kong, Shenzhen
Shenzhen, GD 518172
xinyufang@link.cuhk.edu.cn

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Contents

Abstract

In this project, the author intends to build a practical computer-aided diagnosis (CAD) software based on the *Kvasir*[13] and *Hyper-Kvasir* [2] dataset, the largest image and video dataset of the gastrointestinal tract available today. When the doctor input a video and set the number of action localization n, the software will cut the video into n parts and output the keyframes of each part based on the neural controlled differential equation. The visualization results show that the dynamic of hidden state evolves with the motion of the probe in the gastrointestinal tract, which allows action localization according to clustering results of dynamic hidden states. we choose a keyframe with the highest prediction score as the representative frame. As the doctor play the video and select the target frame, the software will output the classification and segmentation results of this frame. The classification result is improved by semi-supervised learning and unsupervised learning. The segmentation result is based on UNet3+.

1 Introduction

In this project, our group intends to build a practical computer-aided diagnosis (CAD) software based on the *Kvasir*[13] and *Hyper-Kvasir* [2] dataset, the almost largest image and a video dataset of the gastrointestinal tract available today. Our goal is to do action localization and keyframes extraction to an intestinal testing medical video. Classification and segmentation to the selected frame should also be shown in our software.

A common approach to solving the video classification and clipping problem is to use features from a 3D Convolutional Neural Network (C3D) as input to train a recurrent neural network (RNN) [10]. The final hidden state of RNN determines the classification results and video action localization is determined by predicted frame labels, i.e. images with the same labels are clustered together. Natural video usually contains multiple shots and each shot corresponds to distinct labels. In each video of the Hyper-Kvasir dataset, there is only one shot and this shot contains distinct parts of the gastrointestinal tract and they may have the same label. Using different predicted labels for action localization does not work.

When the doctor selects the target frame, the software will output the classified disease and segmentation result. The *Hyper-Kvasir* dataset entails on almost 99417 unlabeled images, while the labeled image is just 1878. The corresponding labels are 4 kinds of the pathological findings of lower GI tract, which are polyps, ulcerative-colitis-grade-1, ulcerative-colitis-grade-2, and ulcerative-colitis-grade-3. The distribution of the number of images is in Figure 1. The sampled labeled image is in Figure 2:

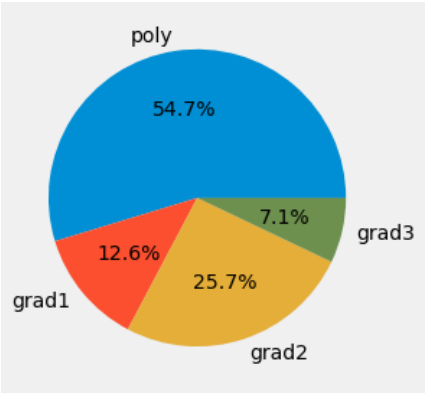

Figure 1: Labeled dataset distribution

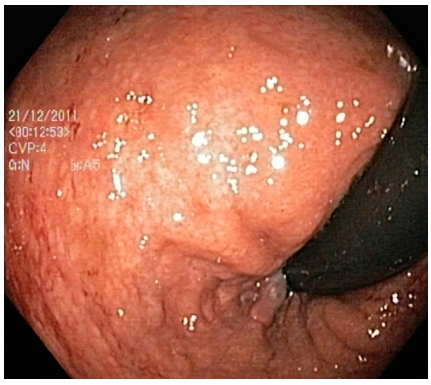

Figure 2: Sample images

The problems with the classification data set are the large size of unlabeled images and the unbalanced image labels. For the first problem, the pre-trained classification model would use semi-supervised

and unsupervised learning to utilize the information in these data. For the second problem, the weighted loss function is applied.

The segmentation model is pre-trained in the given segmentation dataset. The segmentation dataset contains 1,000 images and their corresponding masks. Then, the model is transferred to the task of segmentation in the medical video key frames.

2 Methods

2.1 Semi-supervised Learning

We train our model using the self-training framework [18] which has three main steps: 1) train a teacher model on labeled images, 2) use the teacher to generate pseudo labels on the unlabeled datasets, 3) train a student model on the combination of labeled images and pseudo labeled images. This algorithm goes several times by treating the student as a teacher to relabel the unlabeled data and training a new student model.

The key under this algorithm is the criteria to generate pseudo labels and how to train them. A wrong labeled image hurt the model very much. since we need to confirm the accuracy of pseudo labels, we set a threshold and only if predicted probability is larger than this threshold do we put it as a pseudo label. A recent paper tells that adding noise to the unlabeled image can improve the robustness of the student model and get better performance [21]. So every unlabeled image should first be noised and then predicted. The first teacher model will be Resnet 18, the student model will be chosen after considering the size of the labeled images and pseudo labeled images. The teacher-student algorithm will repeat twice.

The traditional cross-entropy loss strict the model to the direction led by the training set, but it shows poor performance facing some noisy labels. Pseudo labels cannot be always accurate. MAE loss is robust against noisy labels [6], but it doesn't believe in the labels and performs not well, with only tiny wrong labeled images. To reduce such influence, we apply Generalized Cross Entropy loss [22], defined by :

$$\textbf{GCE} = \frac{1 - P(\theta_j)^q}{q},\ q \in (0, 1)$$

The Cross Entropy loss is: $\textbf{CE} = -ln(P(\theta_j))$ and the MAE loss is: $\textbf{MAE} = 1 - P(\theta_j)$.

To understand the relationship between the three loss functions, We take a derivative of them:

$$\frac{\partial \textbf{GCE}}{\partial \theta_j} = -P(\theta_j)^{q-1}\frac{\partial P(\theta_j)}{\partial \theta_j}$$

$$\frac{\partial \textbf{CE}}{\partial \theta_j} = -P(\theta_j)^{-1}\frac{\partial P(\theta_j)}{\partial \theta_j}$$

$$\frac{\partial \textbf{MAE}}{\partial \theta_j} = -\frac{\partial P(\theta_j)}{\partial \theta_j}$$

When q is close to 0, the derivative of GCE is similar to CE. When q is close to 1, the derivative of GCE is MAE. GCE loss provides a balanced method between CE and MAE loss by choosing different q. In our experiment, q is selected to be 0.7.

We believe that the initially labeled data are accurate. Noticing that the training set is highly unbalanced, we apply weighted focal loss for initially labeled data. The loss function is weighted focal loss defined by

$$\textbf{Loss} = -w_j(1 - P(\theta_j))^2 ln(P(\theta_j)),\ w_j = \frac{1}{P(z_k = j|x_k)}$$

Here $P(z_k = j|x_k)$ is simplified as the percentage of the category j in the above pie graph. Even though the pseudo labeled images can expand the training set, it is not as reliable as the initial labeled images. For the pseudo labels, we apply the GCE loss. The loss is multiplied by 4 on the labeled set, to decrease the influence of the pseudo labels.

2.2 Unsupervised Learning

2.2.1 Related Work

Unsupervised learning methods focus on two main aspects: pretext tasks and loss functions. The term "pretext" means the tasks currently focused on is not the task that is concerned in actual application scenarios. It is only solved to get a good data application and serve the real purpose. The original MoCo focuses on the study of the loss function. Related studies of these two aspects will be discussed here.

Pretext tasks. Recent work has proposed many useful pretext tasks. A commonly used class of pretext tasks is to form pseudo-labels. The approaches include transformations of a single image [1], tracking[19] or segmentation objects in videos [5], clustering features[9], or patch orderings [11]. Other pretext tasks include recovering the input under some corruption, various auto-encoders were proposed to complete this task, such as context auto-encoders [4], cross-channel auto-encoders[16], or denoising auto-encoders [12].

Loss functions. One usual way to define loss functions is to measure the difference between a fixed target and a model's prediction. For example, using L1 or L2 norm to reconstruct input pixels, or using marginal-based losses or cross-entropy to do the classification task (e.g., color bins [15]). Contrastive losses [14] aim at measuring similarities between sample pairs in a representation space. In the contrastive loss formula, the target can change dynamically during training. The target can also be defined as the data representation generated by networks [14].

Clustering. Clustering algorithms can be basically divided into hierarchical and partitional. Hierarchical algorithms generate clusters based on previous clusters, while partitional algorithms determine all clusters at the same time. The procedure of the partitioning algorithm is to initialize a number of groups, and iteratively reallocate objects to groups until convergence. The clustering method, K-means, that we adopted when improving MoCo belongs to partitional algorithms.

2.2.2 Baseline Model: MoCo

In this project, Momentum Contrast (MoCo) is used for unsupervised virtual representation learning. MoCo is presented as a way to build large and consistent dictionaries for unsupervised learning with a contrastive loss. [7] This algorithm is aimed to train an encoder that defines and extracts the underlying features so as to maximize the distances between interclass features. These features are named as "keys" of dynamic dictionaries. The dictionary is dynamic in the sense that the keys are randomly sampled with the evolution of the key encoder during the training. In mathematics, learning is formulated as minimizing a contrast loss. Such an encoder would be used as a feature extractor for downstream classification tasks.

As stated in the name, the main idea of MoCo is contrastive learning. The model consists of two encoders, a query matrix q and a queue of keys. The main process is shown in *Figure 3*.

A notable thing is that the queue is maintained as a dictionary. This implementation allows the reusing of the encoded keys from previous batches. The size of the queue could be much larger than the size of the batch, so the immediately preceding data samples contribute to and refine the effects of training on the current batch. Using a queue makes the dictionary larger. However, it makes it intractable to update the parameters by back-propagation. In this case, a momentum update is introduced. Therefore, the workflow of MoCo is described as follows. The encoder will be randomly initialized, and the parameters will be copied to the momentum encoder. In other words, the encoder and the momentum encoder are equivalent in the initialization phase. The difference between the encoder and the momentum encoder lies in the way of updating parameters. Two sets of data are generated from a batch of data samples by different data augmentation. One set is passed to the encoder to extract the features q, and the other is passed to the momentum encoder to extract the features k. k will be enqueued, and queue updates are involved. After that, the contrastive loss will be calculated based on q, k and the queue. This contributes to the update of parameters in the encoder by back-propagation. Denote the parameters of the momentum encoder as $(\theta)_k$ and those of the momentum encoder as $(\theta)_q$, then we update $(\theta)_k$ by

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q \tag{1}$$

where $m \in [0, 1]$ is a momentum coefficient.

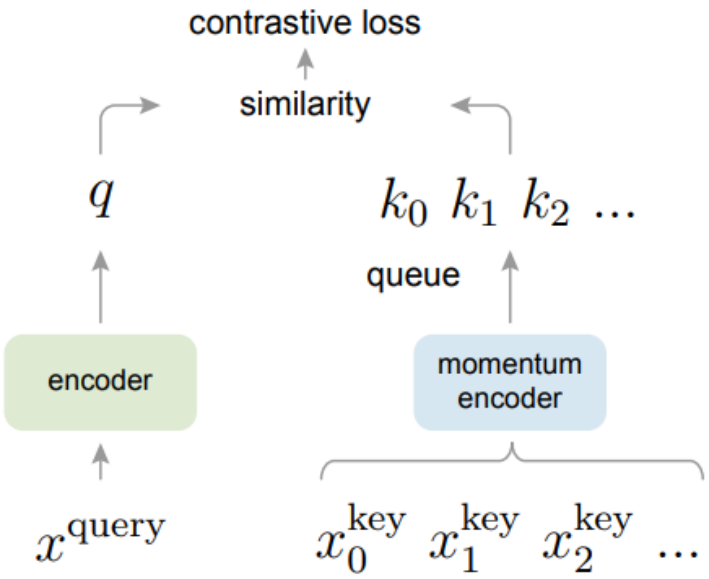

Figure 3: Traditional MoCo

The remaining part is the definition of contrastive loss. For an encoded query q, a key k matches q if they are augmented (probably differently) from the same batch of data. A key is positive for a query if they are matched; otherwise, the key is negative. The contrastive loss function in MoCo and our program is defined as:

$$L_q = -\log\left(\frac{exp(qk_+/\tau)}{\sum_{i=0}^{K} exp(qk_i/\tau)}\right) \tag{2}$$

where τ is a temperature hyper-parameter.

2.2.3 Improvement: MoCo with Clusters

MoCo is powerful as an underlying pattern extractor, but there is still room for improvement. The entry point is the use of a queue. On the one hand, the dynamic queue enlarges the size of the dictionary to improve the training effect with historical data. On the other hand, it also brings a huge burden to the memory. Especially when the program runs on the CUDA version in pursuit of high performance, the huge memory consumption is demanding on the GPU.

Therefore, we improved MoCo by introducing clusters to replace the queue. The use of clustering is still aimed to utilize the preceding information. Compared with the queue, the information is obtained in the clusters of much smaller size. This reduces memory consumption. In our program, the clustering method is Kmeans.

There are two main modifications for MoCo with clustering. The first is the clustering method. The preceding keys are not recorded in our program. They show inexplicitly in the centroid of clustering. It should be noted that the importance of historical keys is not equivalent. The more recent data (acting as negative samples) should have a greater weight. Also, it should be noted that the importance of centroids and the keys of the current batch is not equivalent. Considering these two issues, one practical way is to repeat the current centroids several times as an input with the current keys to conduct the Kmeans in the next iteration. For example, if we replace the queue with the size of 2048 by Kmeans clusters with the size of 128, then the next Kmeans will take 20 repetitions of current centroids and the current keys as the input. It deals with the different weights of keys and centroids. During this procedure, the information of $K + 1$ batches plays the role of K batches in the next iteration. This means there is an important degeneration of the historical data. In this case, it also

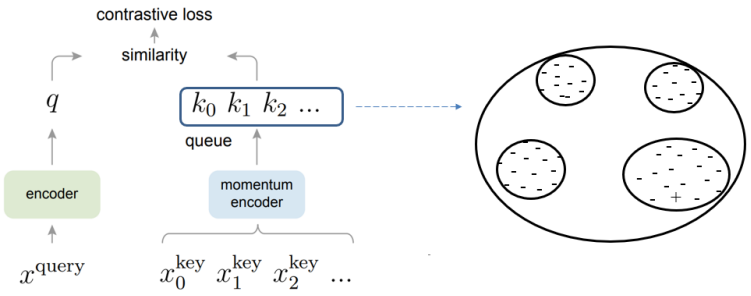

Figure 4: MoCo with clustering

deals with the different weights of historical data. Therefore, replacing the queue with Kmeans clusters are theoretically feasible.

The second is the definition of contrastive loss. More specifically, the definition of positive samples should be modified. In our program, the positive sample k_+ is defined as the centroid to which the encoded query is matched in the Kmeans. These matching pairs will be recorded before performing the fitting. The remaining calculation of the contrastive loss follows the *equation (2)*.

2.3 Action Localization

Standard RNN is a popular choice for time series data. The hidden states of standard RNN (i.e. RNN, LSTM, GRU) is calculated by matrix multiplication and activation function. Hidden states evolve discretely with time, which is inconsistent with the continuity property of video.

Neural ordinary differential equations (Neural ODE) are an attractive option for modelling temporal dynamics. The hidden states of Neural ODE evolve continuously but the main issue is that the solution of ODE function only depends on the initial inputted data, which can be resolved through CDE.

Let $T \in \mathbb{N}$ be the number of inputted timestamp. Let $x_i \in \mathbb{R}^v$ be the corresponding observation data and $X : [0, T] \to \mathbb{R}^v$ be a differentiable function of bounded variation. Let ζ be any neural network $\zeta : \mathbb{R}^v \to \mathbb{R}^w$ and $f : \mathbb{R}^w \to \mathbb{R}^{w \times v}$. Then we may define a continuous path $z : [0, T] \to \mathbb{R}^w$ and

$$z_T = z_0 + \int_0^T f(z_s) dX_s, \tag{3}$$

where $z_0 = \zeta_h(x_0)$ and the integral is a Riemann-Stieltjes integral. The Neural CDE is the continuous analogue of an RNN.

Suppose for simplicity that we can clipping video into n regular sampled time series but potential irregular sampled time series (pertubating clipped time to avoid fuzzy inputted images) $x = ((t_0, k_0), (t_1, k_1), ..., (t_n, k_n))$, where k_i denotes a keyframe of video. At each timestamp i, we obtained a feature vector $x_i \in \mathbb{R}^v$ representing the inputted image from a pretrained feature extractor. Let $X : [0, T] \to \mathbb{R}^v$ be the natural cubic spline with knots at $t_0, t_1, ..., t_n \in [0, T]$ such that $X(t_i) = x_t$. Let $f_\theta : \mathbb{R}^w \to \mathbb{R}^{w \times v}$ be any function approximated by a neural network depending on a trainable parameter θ. Then the final hidden state is the solution of the code

$$z_{t_n} = z_{t_0} + \int_{t_0}^{t_n} f_\theta(z_s) dX_s = z_{t_0} + \int_{t_0}^{t_n} f_\theta(z_s) \frac{dX_s}{ds}(s) ds = z_{t_0} + \int_{t_0}^{t_n} g_{\theta,X}(z_s, s) ds \tag{4}$$

We obtain the final classification probabilities by a linear mapping of hidden states followed by a softmax activation function, i.e. softmax$(\zeta(z_{t_n}))$. Action localization is solved by clustering hidden state dynamics by affinity propagation clustering and spectral clustering. In each cluster of video images, we choose the image with the highest confidence prediction as one of the keyframes of the video.

2.4 Segmentation

Segmentation method is also essential to help the doctor quickly diagnosis the illness part. We choose UNet [17], UNet++ [23] and UNet3+ [8] as our candidate model and pretrain them in the dataset for segmentation. The final model is chosen according to the dice score. Then, the model is transferred into the task of segmentation in the medical video key frame.

3 Experiments & Results

3.1 Semi-supervised Learning

3.1.1 Teacher Model1

We use the Resnet 18 for the basic 4-category classification method and the pre-trained model on the website. The data augmentation methods contain random crop, rotation, flip, and brightness. Here we split the data set into 5 parts. We train a teacher model on 4 parts of them and use the rest one for validation. The result accuracy is 0.816 and the Auc is 0.922 on the test set in Figure 5. The training process shows no overfitting problem and the test accuracy is even higher than the training set sometimes. The result comes from data augmentation, which makes the model has low confidence in determining some images and makes some mistakes on the training set sometimes. But on the test set, there is no such transform.

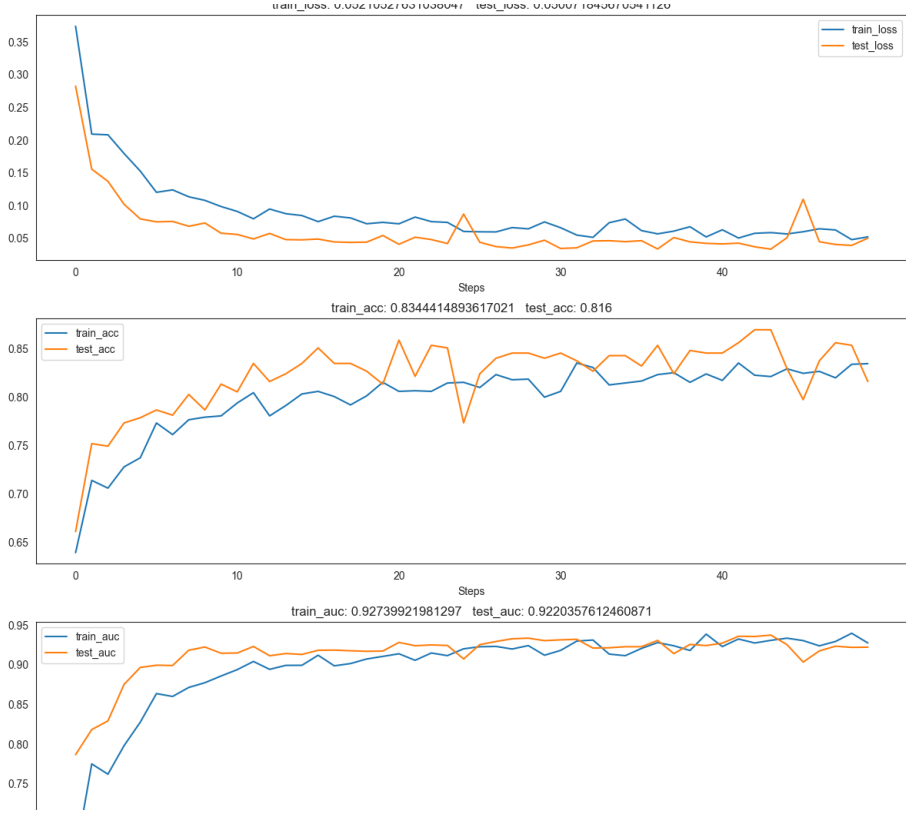

Figure 5: Semi-supervised Learning Results

3.1.2 Student Model1

After finishing finding the teaching model, we then set up a criterion to choose the pseudo labels. We first get a noise to every unlabeled images and set up a threshold of 0.8 and 0.9. The noise methods are rotating, flipping, brightness, blur, brightness. The distribution of the pseudo labels are in Figure 6, 7: Noticing that under this algorithm, the pseudo labels are also highly unbalanced and polyps are

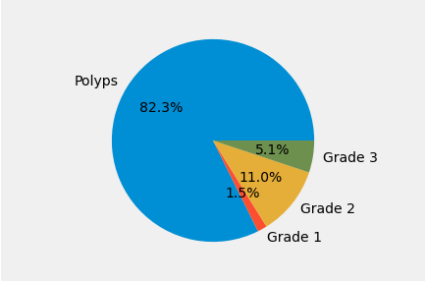

Figure 6: Threshold of 0.8

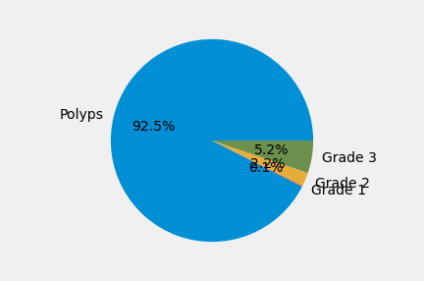

Figure 7: Threshold of 0.9

the major category. We intend to keep the chosen pseudo labels balanced by limiting the number of the chosen images with pseudo labels for each category. Here the upper bound of pseudo labels for each category is 1500. Under the threshold of 0.8, we first sort the pseudo labels by the probability for the corresponding label derived by the softmax layer. Next, we iterate all labels until the limit for each category has been reached. The distribution is in Figure 8 and the corresponding bar graph is in Figure 9.

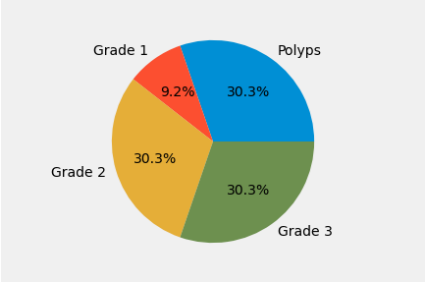

Figure 8: Distribution of labels

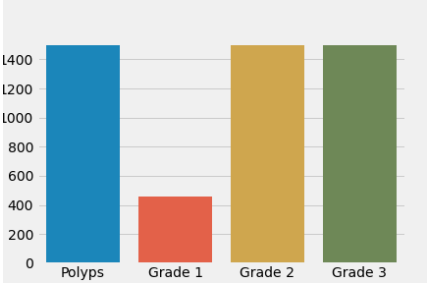

Figure 9: Number for each category

3.1.3 Teacher Model2

Then we update the weight in a weighted focal loss by combining the number of each pseudo label with the initial labels. The student model is set up by Resnet 101. The student model is first trained by the same labels in the previous teacher model and then trained by pseudo labels, then tested on the same test set in the previous teacher model. The training process of the teacher model 2 is in Figure 10: Here the accuracy and AUC increase to 0.838 and 0.944. The Auc on the test set is always larger than 0.91 and the accuracy is stably larger than 0.8 after the 10^{th} epoch. There is no overfitting problem.

3.1.4 Student Model2

We use the new teacher model to annotate the unlabeled set again. Nevertheless, the images that are already chosen as pseudo labels will be out from being chosen again. In a similar way, we set up thresholds and inspect the distribution of the pseudo labels in Figure 11, 12.

The percentage of polyps is 97.8%. And that becomes 99.6% after we choose a threshold of 0.9. Under this circumstance, the prediction of other categories is not reliable since it can be an error made by the model. Considering the unbalance problem, we don't choose any pseudo labels this time. This distribution comes from the behavior in building student model 1, where we nearly extract all images with high confidence in other labels.

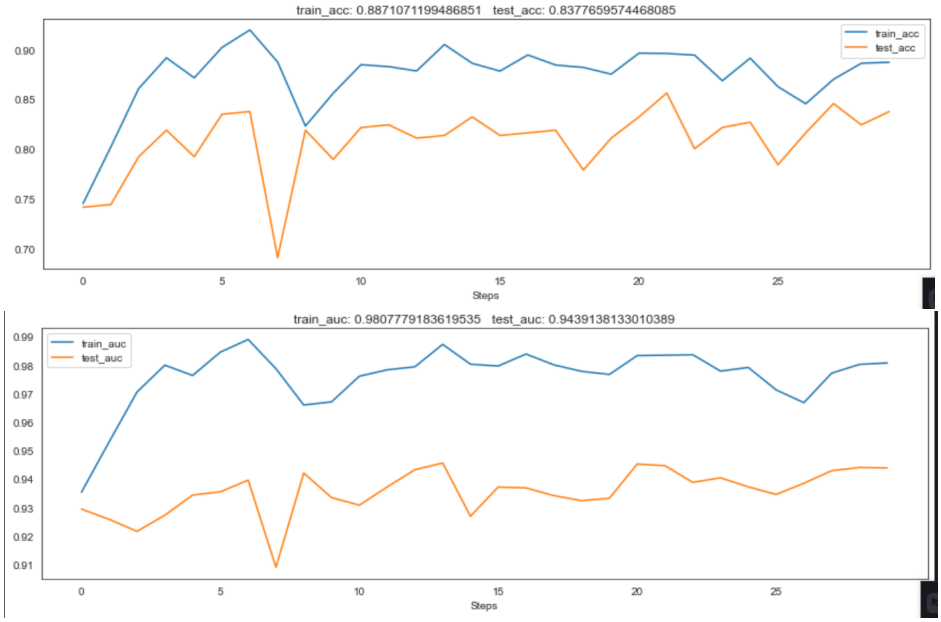

Figure 10: Training process teacher model 2

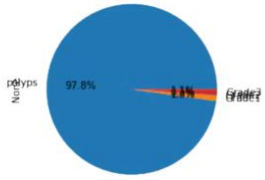

Figure 11: Threshold of 0.8

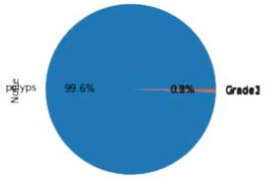

Figure 12: Threshold of 0.9

3.2 Unsupervised Learning

3.2.1 Setting

There are 99417 unsupervised images used for training the MoCo encoder. The validation data is based on labeled images. These data will be input to the well-trained encoder, and the extracted features will be used for downstream classification tasks. In this section, Resnet18 is chosen as the encoder and the momentum encoder. The downstream task is performed by KNN, where K is set to be 20. The momentum coefficient is set to be 0.999, which ensures the parameters of the momentum encoder change smoothly. The temperature hyperparameter is set to be 0.1. The dimension of features is set to 64. The size of Kmeans clustering is set to 128.

3.2.2 Result of Unsupervised Learning

We conduct several experiments to compare the performance of the original MoCo and the MoCo with clustering. We also vary the size of the queue in order to dig out how it influences the performance. For MoCo with clustering, the value (4096, 2048, or 1024) refers to the queue size of its corresponding normal MoCo version. The main difference here is the proportion of the differences in importance between the centroids and the current encoded data samples. More specifically, it influences the number of repetition times of cluster centroids in the input of next Kmeans.

The results of the training are shown in Figure 13. The training refers to the training of the MoCo encoder. The similarity between positive pairs embodies the encoder's ability to recognize the same

image with different augmentation, which is a manifestation of the data extraction ability for the same type of data. For a series of experiments on the same algorithm, the image curves follow a consistent trend. Performing the same epoch of experiment, the model with (or corresponding to) a larger queue size generates a higher similarity. This matches the institution since the larger queue can better utilize the historical data to correct the update of parameters. The curves for traditional MoCo keep increasing while the increasing rates gradually slow down. In contrast, the changes in the curves for MoCo with clusters are more complex. At the initial stage (epoch 0 to 20), the performance of MoCo with clustering is significantly lower than that of traditional MoCo. It is because MoCo with clustering cannot completely discard the influence of historical data before a certain period. The encoded keys generated by the randomly initialized encoder affect the training effect in the initialization stage. When the epoch was within 20 to 30, the training performance of MoCo surpassed that of traditional MoCo. This is due to the more consistent influence of the (rational) historical data in the algorithm of MoCo with clustering. After 30 epochs, the training effect tends to be stable. During this process, the curves of MoCo with clustering fall first and then keep rising, and they remain lower than those of traditional MoCo. This could be explained that, during the fine-tuning phase of the model, historical keys that were not extracted "correctly" brings limited facilitation or even negative impacts. In general, the training performance of these two algorithms tends to be consistent when the epoch is relatively large.

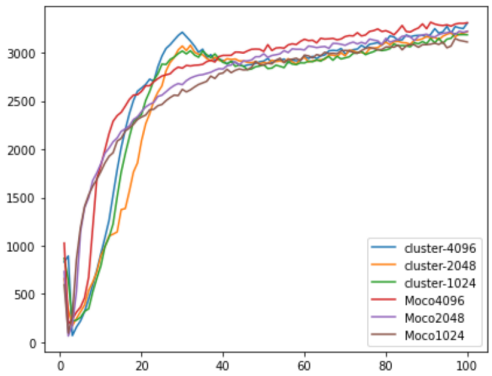

Figure 13: The variation of similarity between positive pairs with epoch

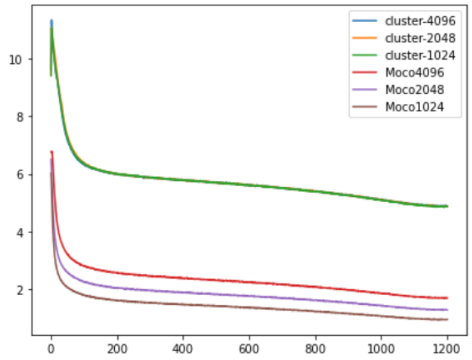

Figure 14: The variation of similarity between negative pairs with epoch

We also conduct the training for traditional MoCo and MoCo with clustering with different sizes. The performance is evaluated by the accuracy of the downstream KNN. The result shows in Figure 13. Similar to the analysis of the similarity curves, during periods of more drastic parameter changes, the wider view of historical data allows MoCo with clustering to ensure its superiority. In the fine-tune period, when the model tends to converge and the parameters change smoothly, historical wrong samples become an obstacle to performance improvement.

Generally speaking, there is no significant difference in the performance of traditional MoCo and MoCo with clustering in feature extraction. However, memory consumption differs dramatically. The number of cluster centroids is much less than that of the size of the queue. It is admitted that MoCo with clustering requires extraneous computation in the clustering process. Considering the traits of GPU computing, less memory consumption is far more acceptable than more computations. This justifies our improvement in the MoCo algorithm.

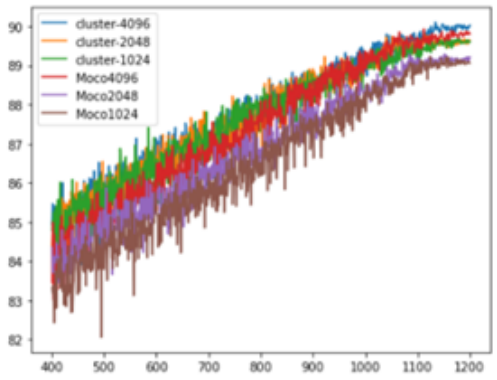

Figure 15: The accuracy variation with epoch

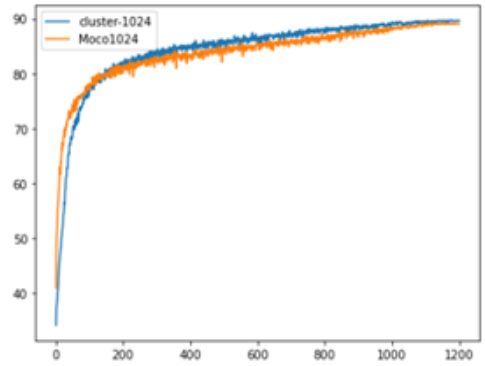

Figure 16: The accuracy variation with epoch for MoCo with size of 1024

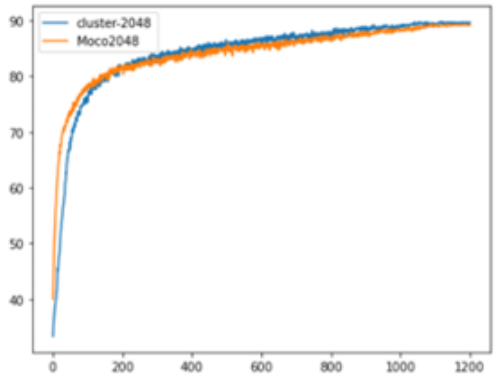

Figure 17: The accuracy variation with epoch for MoCo with size of 2048

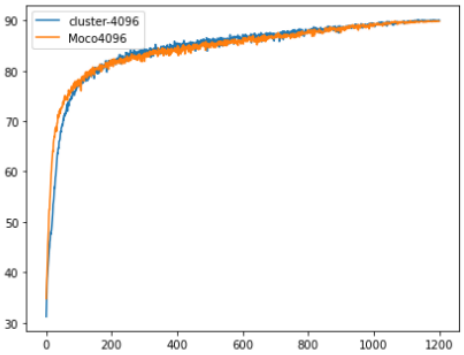

Figure 18: The accuracy variation with epoch for MoCo with size of 4096

3.3 Action Localization

We train a Resnet18 on the Kvasir dataset of 6 classes as the feature extractor of the Neural CDE model. Kvasir dataset contains 6 balanced classes of endoscopic pathological findings. Each class contains 500 images. Then we train the Neural CDE video classification model on the Hyper-Kvasir dataset which contains 13 videos of colitis and 66 videos of polyps. Each video has a length of 1-2 minutes and around 64 equidistant keyframes are extracted from each video as the input of the Neural CDE model. The final accuracy of the model reaches 87%, and the AUC score reaches 0.88.

We use the TSNE algorithm to reduce the hidden state dimension by Euclidean distance. Figure 19 and Figure 20 illustrate hidden state dynamics. The first subgraph shows how hidden states evolve with time. The second and third subgraphs show the corresponding prediction probability and label of each hidden state, respectively. We can see that model's output depends on the inputted images. In different periods of the video, our model may have different classification results.

No supervised label for action localization can be found in the Hyper-Kvasir dataset. We try to modify the model for video classification to a model for action localization. We assume that hidden states indeed represent the so far information of the inputted video. Then, the sudden change of hidden states indicates that a brand new image is inputted. Hence, we can do the action localization task by observing hidden state dynamics.

The above visualization results show that generally hidden states evolve smoothly and sometimes jump to another state and stabilize again. We say the video's action varies if the hidden state leaps. Both affinity propagation clustering and spectral clustering can identify hidden state leaps. The main difference is that affinity propagation clustering can identify the number of clusters by damping rate and spectral clustering needs a specified number of clusters. When applying clustering algorithms,

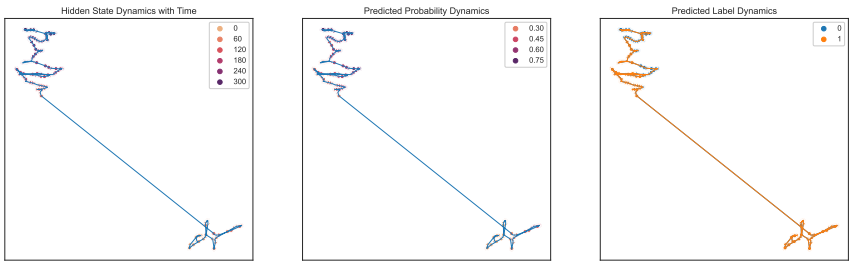

Figure 19: Hidden state dynamic of a colitis sample

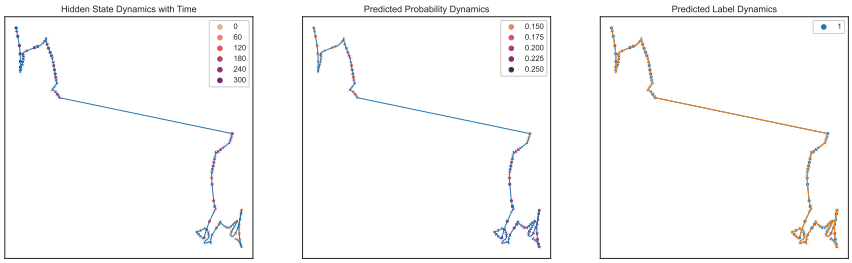

Figure 20: Hidden state dynamic of a polyps sample

timestamps are added to hidden vectors after normalizing into a similar magnitude, which enables the model to cluster hidden states "sequentially".

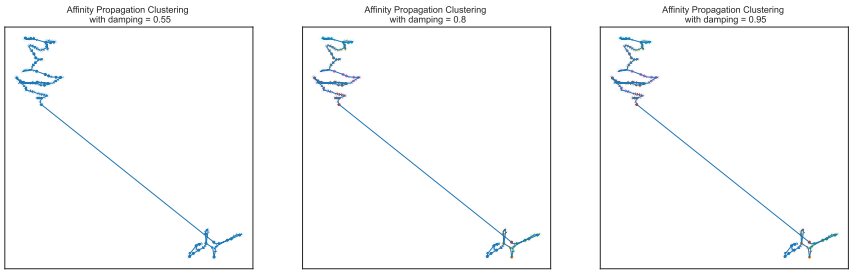

Figure 21: Hidden state cluster of affinity propagation clustering

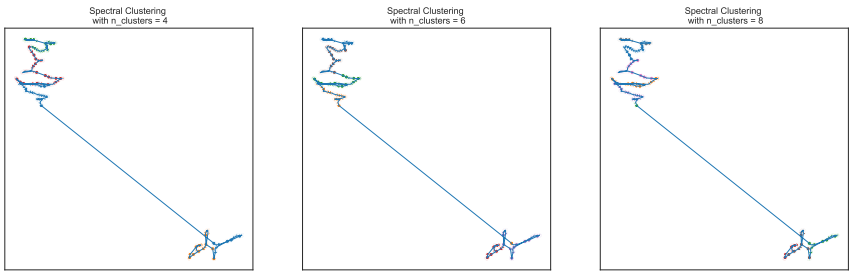

Figure 22: Hidden state cluster of spectral clustering

We use the first-time index of each hidden state cluster as the action localization start point and compare the results obtained from manually localizing the action of the endoscopic video. The results in Figure 23 show that when new noticeable objects appear, the hidden state leaps and hence results in a new action localization.

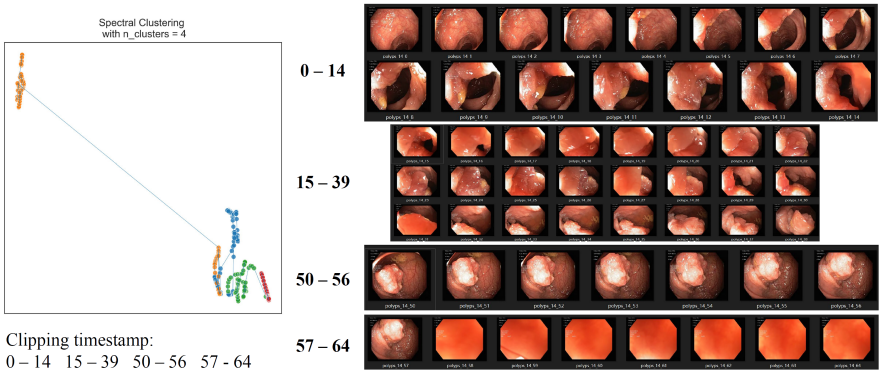

Figure 23: Video clipping results

We also use the Grad-CAM to visualize the model's attention. Figure 24 shows that in different clusters of images, the Neural CDE model pays attention to different locations.

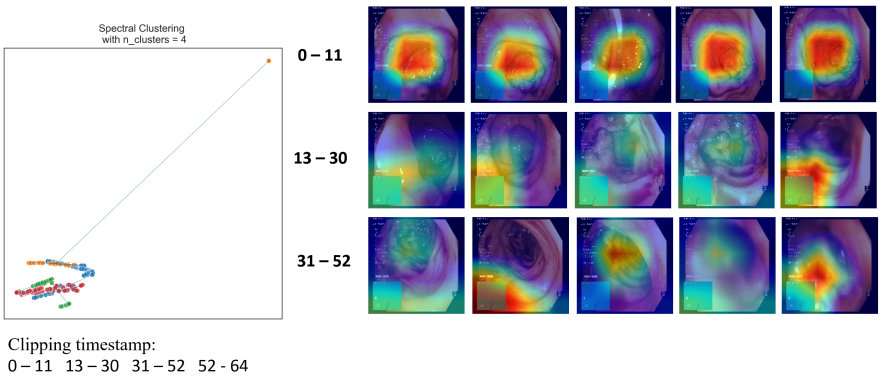

Figure 24: Video attention results

After clipping videos into several parts, we pass frames in each part to a classification model and choose the image with the highest confidence prediction score as one keyframe of this video.

3.4 Segmentation

A segmentation result would also help to clarify the illness part after the doctor selects a frame when playing the medical video. Since there is no mask data for the video, we pre-train a segmentation model in the given segmentation dataset and transfer this model to our task. An example of the given segmentation image and mask are shown below:

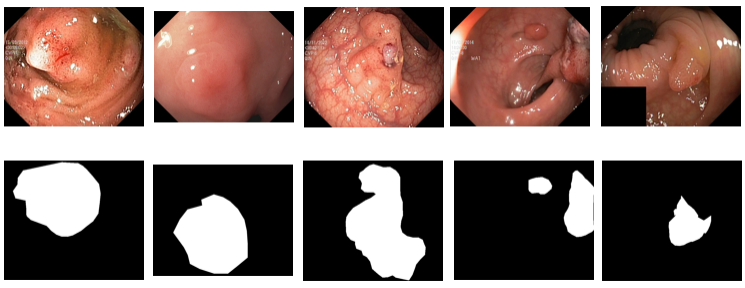

Figure 25: Example of segmentation dataset

With its high performance in medical image segmentation, the author choose UNet [17] as the candidate model, and compare its results with two variations: UNet++ [23] and UNet3+ [8], then select a final model by comparing the dice score.

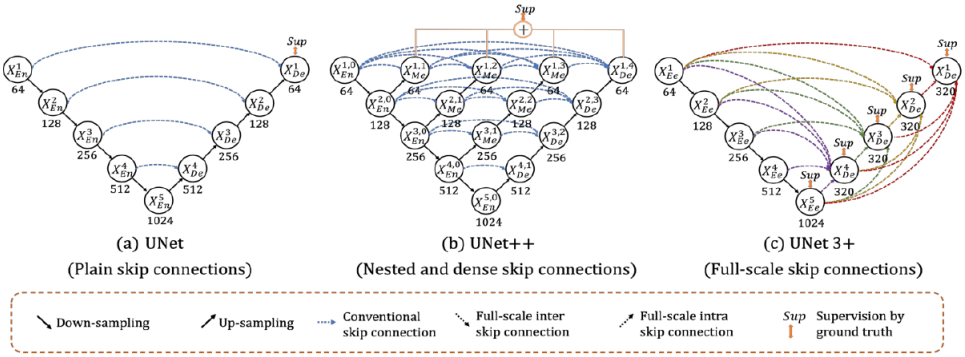

Figure 26: Candidate model structure [8]

The above figure 26 shows the network structure of each candidate model.

Since there are 1,000 segmentation images and mask pairs, the author randomly chooses 800 data as the training set, while the rest is the validation set. Naturally, the one with the highest dice score is chosen as the final segmentation model.

The final dice score results are shown below:

Model	IOU Loss	Validation Dice Score
UNet	0.2620	0.8321
UNet++	0.2352	0.8532
UNet3+	0.1221	0.9329

Thus, given its highest dice score, the software use UNet3+ trained in the segmentation dataset as the segmentation model.

4 Discussions & Conclusion

In this project, we build a practical CAD software based on the *Hyper-Kvasir* dataset.

4.1 Semi-supervised Learning

Compared with the baseline model Resnet 18, teacher model 2 in semi-supervised learning has better performance. The accuracy and Auc have both increased by about 2%. The GCE loss function is more robust with noisy labels than the CE loss and has better performance than MAE loss. The limitation of my method is that the teacher models are all Resnet networks and the cross-validation is not applied. Future work will focus on the trial of more kinds of models and apply the cross-validation method to the self-training framework.

4.2 Unsupervised Learning

The modified version of MoCo achieved the goal of reducing memory cost while maintaining similar accuracy performance. In this work, we've used a clustering method to quantify the distribution of keys. A potential future research direction is to utilize the distribution parameters to generate key samples that conform to the original distribution. It is expected by applying data generation, the accuracy of downstream tasks can be further improved.

4.3 Action Localization

The hidden state of standard RNN evolves by matrix multiplication and activation functions, which results that two adjacent hidden states may have a large Euclidean distance. This leads to the hardness of identifying which two hidden states are similar to each other in terms of the provided information. The Neural CDE model's hidden state evolves by a bounded ODE function and controlled signal. Two hidden states at adjacent timestamps must have a close Euclidean distance so that we can use Euclidean distance as the metric for dimension reduction and clustering. Also, the continuous involving hidden state matches the continuity property of medical video. Hence, this model has stronger interpretative abilities.

The bounded hidden state ODE function also has drawbacks. The hidden states of the Neural CDE model evolve more stably than that of standard RNN. Hence, it is more difficult for Neural CDE to catch the keyframes within a short appearing time. For endoscopic pathological finding classification and action localization, due to the smooth movement of the probe (i.e., no sudden shot change as that in the nature video) and no drastic changes in gastrointestinal tract health state as the probe move, we do not need to focus on a single keyframe but choose to use a few adjacent keyframes for diagnosing.

Continuously evolving hidden states of ODE have trouble with making high confidence predictions. In able to shift the prediction label quickly when a different keyframe is inputted, hidden states of the Neural CDE model must lie around the classification boundary so that they can go through the boundary in a short time. Hence, the Neural CDE model may have a low AUC score.

Standard RNN, including LSTM, suffers from gradient exploding and vanishing problems. At each input timestamp, matrix multiplication is performed. If the eigenvalue of the matrix is far from 1, then the gradient will be unstable or negligible. In Neural CDE, hidden states can be solved by the adjoint method which requires less matrix multiplication and less memory [3, 20].

In a video, there may be frames that have totally different features from other adjacent frames but should not be used as the action localization separating keyframes. Neural CDE has the ability to eliminate such distractions by a carefully chosen weighted timestamp. In other words, when clustering the hidden states (one dimension of the hidden states is the timestamp), how to balance the weight of the feature vector and timestamp vector. If we pay more attention to time, this video is more likely to be cut equally because, at this time, the time feature is the dominant variable for clustering. When we pay less attention to time, this video is more likely to be cut according to the feature vector, but the distraction from an outlier image may have more damage to model performance. We grid search some weights of feature vector and timestamp vector, compare their performance, and finally choose the best one.

4.4 Segmentation

Though UNet3+ works well in the segmentation dataset as it gives the 0.9329, the model's performance in the keyframe segmentation task still needs some improvements. The below Figure 27 shows sample keyframe segmentation results. Although the chosen part and the GradCAM somehow coincide with each other, the area of segmentation is far smaller. This illustrates that the model tends to choose the illness part conservatively. For future work, a method that could adopt both the segmentation dataset and the video dataset is suggested.

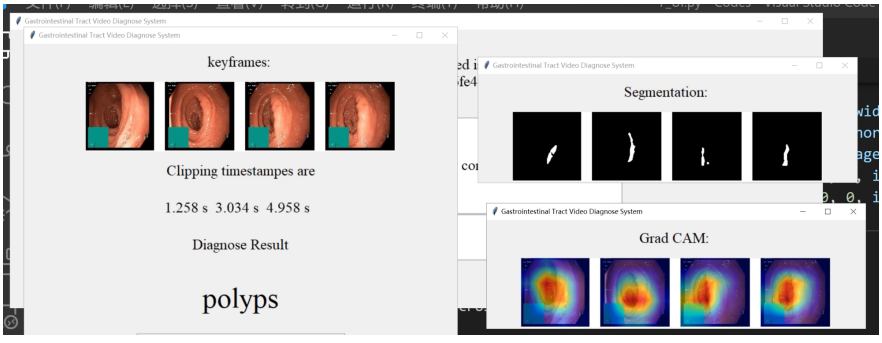

Figure 27: A sample demo result

References

[1] Martin Riedmiller Alexey Dosovitskiy, Jost Tobias Springenberg and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. *NeurIPS*, 2014.

[2] Hanna Borgli, Vajira Thambawita, Pia H Smedsrud, Steven Hicks, Debesh Jha, Sigrun L Eskeland, Kristin Ranheim Randel, Konstantin Pogorelov, Mathias Lux, Duc Tien Dang Nguyen, Dag Johansen, Carsten Griwodz, Håkon K Stensland, Enrique Garcia-Ceja, Peter T Schmidt, Hugo L Hammer, Michael A Riegler, Pål Halvorsen, and Thomas de Lange. HyperKvasir, a comprehensive multi-class image and video dataset for gastrointestinal endoscopy. *Scientific Data*, 7(1):283, 2020.

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[4] Jeff Donahue Trevor Darrell Deepak Pathak, Philipp Krahenbuhl and Alexei A Efros. Context encoders: Feature learning by inpainting. *CVPR*, 2016.

[5] Piotr Dollar Trevor Darrell ´ Deepak Pathak, Ross Girshick and Bharath Hariharan. Learning features by watching objects move. *CVPR*, 2017.

[6] Aritra Ghosh, Naresh Manwani, and P. S. Sastry. Making risk minimization tolerant to label noise. *CoRR*, abs/1403.3610, 2014.

[7] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. pages 9726–9735, 2020.

[8] Huimin Huang, Lanfen Lin, Ruofeng Tong, Hongjie Hu, Qiaowei Zhang, Yutaro Iwamoto, Xianhua Han, Yen-Wei Chen, and Jian Wu. Unet 3+: A full-scale connected unet for medical image segmentation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1055–1059. IEEE, 2020.

[9] Julien Mairal Mathilde Caron, Piotr Bojanowski and Armand Joulin. Unsupervised pre-training of image features on non-curated data. *ICCV*, 2019.

[10] Alberto Montes, Amaia Salvador, Santiago Pascual, and Xavier Giro-i Nieto. Temporal activity detection in untrimmed videos with recurrent neural networks. *arXiv preprint arXiv:1608.08128*, 2016.

[11] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *ECCV*, 2016.

[12] Yoshua Bengio Pascal Vincent, Hugo Larochelle and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. *ICML*, 2008.

[13] Konstantin Pogorelov, Kristin Ranheim Randel, Carsten Griwodz, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Concetto Spampinato, Duc-Tien Dang-Nguyen, Mathias Lux, Peter Thelin Schmidt, Michael Riegler, and Pål Halvorsen. Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, MMSys'17, pages 164–169, New York, NY, USA, 2017. ACM.

[14] Sumit Chopra Raia Hadsell and Yann LeCun. Dimensionality reduction by learning an invariant mapping. *ICCV*, 2019.

[15] Phillip Isola Richard Zhang and Alexei A Efros. Colorful image colorization. *ECCV*, 2016.

[16] Phillip Isola Richard Zhang and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *CVPR*, 2017.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[18] H. J. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inf. Theory*, 11:363–371, 1965.

[19] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. *ICCV*, 2015.

[20] Timo C Wunderlich and Christian Pehle. Eventprop: Backpropagation for exact gradients in spiking neural networks. *arXiv preprint arXiv:2009.08378*, 15:16–86, 2020.

[21] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020.

[22] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *CoRR*, abs/1805.07836, 2018.

[23] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. *CoRR*, abs/1807.10165, 2018.