Supplemental Material
Deep Learning Accurately Distinguished Pancreatic Cancer from Non-cancerous Pancreas

Kao-Lang Liu[1], Tinghui Wu[2], Po-Ting Chen[1], Yuhsiang M. Tsai[2], Holger Roth[3], Ming-Shiang Wu[4, 5], Wei-Chih Liao[4, 5], and Weichung Wang[2]

[1]Department of Medical Imaging, National Taiwan University Cancer Center, National Taiwan University Hospital and National Taiwan University College of Medicine, Taipei, Taiwan
[2]Institute of Applied Mathematical Sciences, National Taiwan University, Taipei, Taiwan.
[3]NVIDIA, Bethesda, MD 20814, USA.
[4]Division of Gastroenterology and Hepatology, Department of Internal Medicine, National Taiwan University Hospital, National Taiwan University College of Medicine, Taipei, Taiwan.
[5]Internal Medicine, College of Medicine, National Taiwan University, Taipei, Taiwan.

February 27, 2020

1 Introduction

This supplemental material provides details of the analysis method used in the paper entitled "Deep Learning Accurately Distinguished Pancreatic Cancer from Non-cancerous Pancreas". For each patient, there would be one 3D CT volume constructed by several abdominal CT images along with pancreas and tumor segmentation manually labeled by radiologists. The analysis method includes the following steps.

1. Preprocess both CT volume and its corresponding segmentation label.
2. Generate cancerous and non-cancerous patches from CT images.
3. Train a convolution neural network (CNN) model to distinguish the cancerous and non-cancerous patches.
4. Select thresholds for patch-based and patient-based analysis from the model prediction.
5. Test the model on both local and external testing set.

Steps 5 is straightforward. We illustrate the technical details of Steps 1 to 4 in the following sections.

2 Image Preprocessing

We perform image preprocessing, including reconstruction, dilation, windowing, and normalization, for the CT volumes and their corresponding segmentation labels. Firstly, we reconstructed all the CT volumes and segmentation labels into 5 mm slices by linear interpolation and nearest-neighbor interpolation, respectively. Then, we performed dilation on both pancreas and tumor segmentations by adding 3x3 pixels on the x-y plane along the boundaries. That is, we expanded the boundaries of pancreases and tumors to include more information for analysis. The window width and window level of CT images were set as 250 Hounsfield unit (HU) and 75 HU, respectively. Finally, we normalized the pixel intensity values of images to $[0, 1]$.

3 Patch Generation

After preprocessing the images, we generated patches for CNN model training. A patch is a square sub-region of a two-dimensional CT image, along with a cancerous or non-cancerous label. To generate patches, we need a CT image as input and one target region Ω for patch localization and masking. Also, we need three parameters to decide how the patches would be generated: the patch size m, the maximum number of patches generated from each patient M, and the stride of the moving window s. In this study, the target region Ω is the union of pancreas and tumor area, which means all the non-pancreas areas will be ignored. We performed five-fold cross validation to find the optimized patch size m. The maximum number of patches generated from each patient is fixed as $M = 4000$. The stride s is originally set as $\frac{1}{2}m$ and would increase if too many patches have been generated.

Patches are generated slice by slice in the target region Ω by moving the $m \times m$ window from left to right and top to down on each slide. The process of generating patches is listed below.

1. Find the bounding box of Ω. Denote the bounding box as $B = [x_1, x_2) \times [y_1, y_2) \times [z_1, z_2)$.
2. For each CT slice $z \in [z_1, z_2)$, generate the patches on each slide as the following steps:
   (a) Find the number of the generated patch in x and y direction as n_x and n_y, respectively. Specifically, n_x and n_y are computed by

   $$n_x = \left\lfloor \frac{(x_2 - x_1) - m}{s} \right\rfloor + 1$$

   and

   $$n_y = \left\lfloor \frac{(y_2 - y_1) - m}{s} \right\rfloor + 1.$$

   (b) Generate the temporary patch list of this slide $P_z = \{p_{(i*n_y+j)}\}_{i=0,j=0}^{i=n_x-1,j=n_y-1}$ as

   $$p_{(i*n_y+j)} = \begin{cases} [x_2 - s, x_2) \times [y_2 - s, y_2) \times z & \text{if } x_1 + (i+1)s > x_2 \text{ and } y_1 + (j+1)s > y_2 \\ [x_1 + is, x_1 + (i+1)s)) \times [y_2 - s, y_2) \times z & \text{if } x_1 + (i+1)s \le x_2 \text{ and } y_1 + (j+1)s > y_2 \\ [x_2 - s, x_2) \times [y_1 + js, y_1 + (j+1)s) \times z & \text{if } x_1 + (i+1)s > x_2 \text{ and } y_1 + (j+1)s \le y_2 \\ [x_1 + is, x_1 + (i+1)s) \times [y_1 + js, y_1 + (j+1)s) \times z & \text{Otherwise} \end{cases}$$

   (c) Append all the patches in P_z behind P. (The list P is initialized as empty list.)
3. Generate classification label for each patches by the rules below.
   (a) If a patch contains tumor, label the patch as "cancerous".
   (b) For the rest of the patches, if the patch contains non-cancerous pancreas, label the patch as "non-cancerous".
   (c) Exclude the patches that are neither cancerous nor non-cancerous patches.
4. To reduce the variance of the patches number generated from each patient, exclude every other two patches if the number of generated patches n is larger than the maximum number M.

4 Model Training

We describe how the CNN model was trained by introducing the model structure, software, hardware, and other training details.

4.1 Model Structure

We revised the VGG network[1] to construct a cancerous/non-cancerous patch classification CNN. The input images were $m \times m$ in grayscale. The model contained two parts, convolution part and fully connected part. The are three blocks in the convolution part. Each block consisted of two convolution layers followed by the rectified linear unit (ReLU) as the activation function, and then connected with a max-pooling layer before the next block. After passing the first part, all the values will be flatten to an array. Then, there are two fully connected layers in the second part. Table 1 shows the detailed information of the model structure when the patch size $m = 50$.

Layer	Kernel size	Channel	Output size
Convolution 1a	5×5	16	(50,50,16)
Convolution 1b	5×5	32	(50,50,32)
Max-pooling 1	2×2	-	(25,25,32)
Convolution 2a	3×3	64	(25,25,64)
Convolution 2b	3×3	64	(25,25,64)
Max-pooling 2	2×2	-	(12,12,64)
Convolution 3a	3×3	128	(12,12,128)
Convolution 3b	3×3	128	(12,12,128)
Max-pooling 3	2×2	-	(6,6,128)
Flatten	-	-	(4608)
Dense 1	-	-	(32)
Dense 2	-	-	(32)
Dense 3	-	-	(1)

Table 1: Structure of the convolution nural network (input: 50×50)

Weighted binary cross-entropy was used as the loss function to account for imbalance between the number of cancerous and non-cancerous patches. That is, the loss function

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} w_1 y_i \dot{\log}(p(y_i)) + w_2(1-y_i)\dot{\log}(1-p(y_i)),$$

where N is the total amount of training patches. The classification label of the i-th patch y_i is 1 for cancerous patch, and 0 for non-cancerous patches. The model prediction of the i-th patch is set as $p(y_i)$. The weights w_1 and w_2 are given from the inversion proportion of the sample number in the two classes. That is,

$$w_1 : w_2 = N_{cancerous} : N_{non-cancerous},$$

where $N_{cancerous}$ and $N_{non-cancerous}$ are the amounts of cancerous and non-cancerous patches in the training set, respectively.

4.2 Software and Hardware

We used Python (version 3.6.8), Keras (version 2.2.4) and Tensorflow libraries (version 1.7.0) to compose the codes. The model was trained on a computer equipped with one NVIDIA Tesla P100 with 16 GB memory and Linux OS (Ubuntu 16.04).

4.3 Training Details

During the training process, we set hyperparameters and added callbacks to optimize model performance. For the hyperparameters, we set the batch size as $2,560$, which means the model would receive $2,560$ patches every iteration. Before every training step, the input image would randomly flip horizontally or vertically. Furthermore, we used two callbacks listed below to monitor validation losses to adjust the training movement.

1. Reducing learning rate: if the validation loss did not decrease for ten iterations, the learning rate would be reduced to 10%.
2. Early stopping: after the validation loss remained stable after 40 iterations, the model would stop training to avoid overfitting.

5 Threshold Selection

We describe how we selected the thresholds for patch-based and patient-based analysis in this section. Once the CNN model is trained, we select the cutoff thresholds ρ_{patch} and $\rho_{patient}$ for the patch-based and patient-based analysis, respectively. If the risk of a patch is larger than ρ_{patch}, it will be predicted as a cancerous patch, otherwise a non-cancerous patch. A similar concept is applied to patient-based analysis. The risk of patches and the thresholds ρ_{patch} are computed as follows.

1. Apply the command `model.predict_proba` to all the patches in the validation set `X_valid` and get the probability of prediction as cancerous for each patch. We denote this probability as the risk of the patch.
2. Plot the receiver operating characteristic (ROC) curve based on the risk of patches in the validation set.
3. Determine the patch-based cutoff threshold ρ_{patch} as the one that maximizes the Youden index, i.e. the value of (sensitivity + specificity − 1), of the ROC curve in the previous step.

The risk of patient and the thresholds $\rho_{patient}$ are computed as follows.

1. Compute the risk of each patient in the training and validation set in the following way.
    - For the patient with PC, the risk of the patient is defined as the ratio of the number of cancerous patches that predicted as cancerous to the total number of cancerous patches.
    - For the patient without PC, the risk of the patient is defined as the ratio of the number of patches that predicted as cancerous to the total number of patches.
2. Plot the ROC curve based on the risk of patients in the training and validation sets.
3. Determine the cutoff threshold for patient-based $\rho_{patient}$ as the one that maximizes the Youden index of the ROC curve in the previous step.

References

[1] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv [cs. CV]. 2014Available from: `https://arxiv.org/abs/1409.1556`. (Accessed: 2020-02-26).