Cancer Image Datasets Classification

Huanjia Liu Anyi Wang Anfeng Peng

***Abstract*—With the development of artificial intelligence, computer image recognition technology is continuously applied to various fields. Our project applies image recognition technology to the identification of SIIM-ISIC Melanoma, with the aim of increasing the pre-confirmation rate of this type of cancer, achieving early detection and treatment, and at the same time reducing the work pressure of medical workers. Here we built a convolutional neural network with 7 convolutional layers based on TensorFlow Keras. After training with sgd optimizer, the training accuracy reaches up to 88.96%. Finally, importing 33128 images into the test, the accuracy rate is 85.72%, among which the accuracy of benign is 85.99% and the accuracy of malignant is 70.38%.**

I. Introduction

Cancer is a group of diseases involving abnormal cell growth with the potential to invade or spread to other parts of the body. Over 100 types of cancers, which always occur inadvertently, affect humans. The pain of advanced cancer is unbearable. However, it is not difficult to prevent. In general, there are two species of tumor: benign tumor and malignant tumor(cancer). A benign tumor is the early form of cancer. If we can deal with cancer in this period, the risk will be greatly reduced.

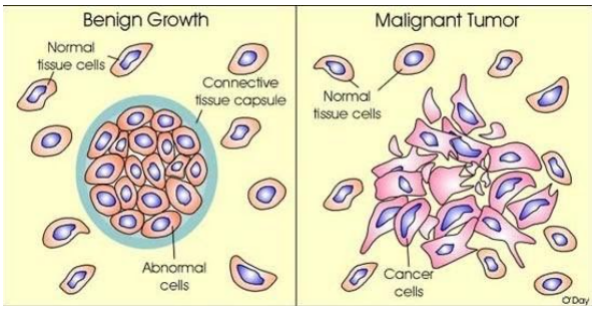

Fig. 1. Benign Tumor and Malignant Tumor

Thus, early detection is raised as our topic here. Thanks to the development of computer vision technology and artificial intelligence, using AI to determine the stage of cancer through images becomes possible in these years. This method is not only practical but also accurate. Our team is preparing to design a CNN (convolutional neural network) using TensorFlow, trained by lesion image to identify SIIM-ISIC Melanoma. The aim of our project is to build a convolutional neural network, which can accept the figure of lesion sites as input, and output the classification (benign or malignant), the accuracy can be achieved 80%-85%, and deploy the model on our website. The patient can upload the skin pathological pictures to the website by taking photos or uploading photos, then the website will analyze the pathological features of the pictures, and give the user the result of whether the pathology is benign or malignant. The website can quickly and easily alert people to skin cancer, so that they can be detected and treated early. At the same time, because it is convenient and easy to understand, ordinary people can easily understand it as well. Therefore, it can well reduce the burden on doctors, reduce the waste of resources, that medical resources can be better provided to patients in need, and provide important help for the future health development of human beings.

As for our team structure, there are three people in our team:

Huanjia Liu is the leader of this team, a senior student major in EE. He is mainly responsible for the organization of the team's weekly meetings and the communication between team members. He also leaded the writing and optimization of the convolutional neural network structure and the maintenance of the cloud server and domain name. At the beginning of the project, he contacted his teammates to discuss and work on the use and tricks of the Keras framework. In the middle of the project, he was responsible for experimenting with different convolutional neural network structures, such as different depths, different convolutional kernels, different activation functions. Later in the project, he was responsible for experimenting with various optimizers derived from group discussions, generating the final model h5 parameter files, and writing the final test program to test the final accuracy of the model. He also participated in the writing of various reports.

Anyi Wang is one of the group member, In the early stage, she was responsible for downloading relevant data of skin cancer required for Convolutional Neural Network training on Kaggle, and then she preprocessed the downloaded data for the Convolutional Neural Network, used Python tools to classify benign and malignant cancer data images, and set the image size parameters. In the middle stage, because there were fewer images of advanced skin cancer in the data downloaded earlier, the preliminary training accuracy deviated. Thus, she supplemented the relevant data of advanced skin cancer. She researched and learned Keras, Adam SGD and other CNN optimizers, and then discussed with team members, tested and adjusted the parameters of the optimizer, and finally decided to choose SGD as the optimizer of our CNN model. At a later stage, in order to deploy our Convolutional Neural Network model on a public website, she completed the construction of the server required by the project, programming and designing of the front end of the web page. Finally, she not only works to ensure that the weekly meetings run smoothly and that there is good communication between the members of the team, but also tracked the schedule of each process and wrote the outline of every paper. She was responsible for coordinating and setting time for weekly meetings as well, so as to ensure that the team could complete the project on time at the end of the term.

Anfeng Peng is one of the team members during the project.

Early in the project, he was the first to test the feasibility of applying ordinary neural networks to this project, and helped the team compare the advantages and disadvantages of Keras and PyTorch. He discussed with team members how to use these different frameworks to build CNN, and after distinguishing the differences between them, testing and debug each framework. After discussing, testing and comparing the differences between the two frameworks, we chose the TensorFlow framework to complete the project, and in order to get a better result, he added some functions to CNN for getting a better result. At the same time in the final model optimization he also did a lot of testing. He attended group meetings every week and completed project tasks with other group members.

II. Process

A. Pre-processing the Data

With the help of With the help of Professor Wilson, we got the "SIIM-ISIC Melanoma Classification database" from Kaggle.[1] There are 44107 sample images over 100GB from Kaggle's database from different patients with different symptoms. And we separated them into two folders by class based on benign or malignant symptoms, this made it easier to extract images to the neural network.

All the image information is saved in a CSV file provided by Kaggle. After checking the JPEG file from the database, we found the size of the images is almost different. So before importing them into CNN, resizing should be processed to set all the image as 500*500 pixel. Also, to make the figures more random, random flip and shift operations will be applied. We can achieve them easily since the open source of library OpenCV has already provided all the functions we mentioned above.Thus, we wrote a short Python function to divide them into two folders (benign and malignant). Then we can use the "ImageDataGenerator" function from Keras to load data conveniently.

```
def processing_data(data_path, height, width,
    batch_size=32, test_split=0.1):


   train_data = ImageDataGenerator(

         rescale=1. / 255,
         shear_range=0.1,
         zoom_range=0.1,
         width_shift_range=0.1,
         height_shift_range=0.1,
         horizontal_flip=True,
         vertical_flip=True,
         validation_split=test_split
   )

   test_data = ImageDataGenerator(
         rescale=1. / 255,
         validation_split=test_split)

   train_generator =
       train_data.flow_from_directory(

         data_path,
         target_size=(height, width),
         batch_size=batch_size,
         class_mode='binary',
         classes=['benign','malignant'],

         subset='training',
         seed=0)
   test_generator =
       test_data.flow_from_directory(
         data_path,
         target_size=(height, width),
         batch_size=batch_size,
         class_mode='binary',
         classes=['benign','malignant'],
         subset='validation',
         seed=0)

   return train_generator, test_generator
```

In the midterm report, We made a very serious error in image preprocessing, as the ratio of early-stage cancer, to late-stage cancer was too imbalanced, causing us to run up to 98% correct with a neural network using only 4 layers of convolutional layers.

After checking, it was found that the ratio of benign to malignant for training was as high as 12:1, which is the reason for such an outrageously high accuracy rate. Therefore, in the subsequent improvement, we limit the ratio of benign and maligamant for training to about 1:1, and the accuracy rate returns to normal.

B. Building CNN using TensorFlow

Keras is an open-source software library that provides a Python interface for artificial neural networks. With the help of Keras, we can build a CNN easily. However, the exact structure of the neural network still has to be decided by us. At first, we tried to use a four convolutional layers network (Fig. 2) to solve this binary classification issues. This model is based on AlexNet, a model from the 2012 ImageNet competition, combing a convolutional layer with pooling layer, and adding a dense layer at the end.[2]

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 498, 498, 32)	896
max_pooling2d (MaxPooling2D)	(None, 249, 249, 32)	0
conv2d_1 (Conv2D)	(None, 247, 247, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 123, 123, 64)	0
conv2d_2 (Conv2D)	(None, 121, 121, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 60, 60, 64)	0
conv2d_3 (Conv2D)	(None, 58, 58, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 29, 29, 128)	0
conv2d_4 (Conv2D)	(None, 27, 27, 128)	147584
max_pooling2d_4 (MaxPooling2	(None, 13, 13, 128)	0
flatten (Flatten)	(None, 21632)	0
dropout (Dropout)	(None, 21632)	0
dense (Dense)	(None, 512)	11076096
dense_1 (Dense)	(None, 1)	513

Total params: 11,354,369
Trainable params: 11,354,369
Non-trainable params: 0

Fig. 2. CNN version 1.0

```
def model():
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3),
        activation='relu', input_shape=(500,
        500, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64,(3,3),activation='relu'))
    model.add(MaxPooling2D((2,2)))
    model.add(Conv2D(64, (3, 3),
        activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3),
        activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3),
        activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dropout(0.5))
    model.add(Dense(512, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    return model
```

They used 5 convolutional layers to solve classification of hundreds of images. But since our task is just a binary classification problem, and in traditional binary classification problem, a 3 layer convolutional layer can solve it with very high accuracy. However after training we found the accuracy is only around 50%.

When we check the detail figure we can find some benign and maligamant are so similar, like the figure below(Fig). So we tired to make our neural network deeper, add more convolutional layers with pooling layers.

It works well, but when the number of convolutional layers is higher than 7, the contribution to the accuracy rate is not so obvious, and the result is an ever-increasing amount of operations. The number of layers of CNN needs to match the number of features provided by the image, perhaps our image cannot provide so many features to a CNN with more than seven convolutional layers.[3] Now the accuracy of our CNN stayed around 70%.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 498, 498, 32)	896
max_pooling2d (MaxPooling2D)	(None, 249, 249, 32)	0
conv2d_1 (Conv2D)	(None, 247, 247, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 123, 123, 64)	0
conv2d_2 (Conv2D)	(None, 121, 121, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 60, 60, 64)	0
conv2d_3 (Conv2D)	(None, 58, 58, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 29, 29, 128)	0
conv2d_4 (Conv2D)	(None, 27, 27, 256)	295168
max_pooling2d_4 (MaxPooling2	(None, 13, 13, 256)	0
conv2d_5 (Conv2D)	(None, 11, 11, 512)	1180160
max_pooling2d_5 (MaxPooling2	(None, 5, 5, 512)	0
flatten (Flatten)	(None, 12800)	0
dropout (Dropout)	(None, 12800)	0
dense (Dense)	(None, 1024)	13108224
dense_1 (Dense)	(None, 1)	1025

Total params: 14,714,753
Trainable params: 14,714,753
Non-trainable params: 0

Fig. 3. CNN version 2.0

```
def model():
    model = models.Sequential()
    model.add(Conv2D(32, (13, 13),
        activation='relu', input_shape=(500,
        500, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64,(11,11),activation='relu'))
    model.add(MaxPooling2D((2,2)))
    model.add(Conv2D(64, (7, 7),
        activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (5, 5),
        activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(256, (3, 3),
        activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(512, (3, 3),
        activation='relu'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dropout(0.5))
    model.add(Dense(1024, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))

    return model
```

So we still had to do some other methods to improve the accuracy of our convolutional network. The way we used here is expand the receptive field. In the original LeNet5 convolutional neural network [3], they didn't always use small convolutional kernels, they had layers of smaller kernels, the first few layers, to get more information about the region by using large convolutional kernels. We tried this method and create a model structure below.(Fig. 4)

This didn't help our accuracy rate, so we had to confirm our final neural network structure with seven convolutional layers here and look for breakthroughs in other areas.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 488, 488, 32)	16256
max_pooling2d (MaxPooling2D)	(None, 244, 244, 32)	0
conv2d_1 (Conv2D)	(None, 234, 234, 64)	247872
max_pooling2d_1 (MaxPooling2	(None, 117, 117, 64)	0
conv2d_2 (Conv2D)	(None, 111, 111, 64)	200768
max_pooling2d_2 (MaxPooling2	(None, 55, 55, 64)	0
conv2d_3 (Conv2D)	(None, 51, 51, 128)	204928
max_pooling2d_3 (MaxPooling2	(None, 25, 25, 128)	0
conv2d_4 (Conv2D)	(None, 23, 23, 256)	295168
max_pooling2d_4 (MaxPooling2	(None, 11, 11, 256)	0
conv2d_5 (Conv2D)	(None, 9, 9, 512)	1180160
max_pooling2d_5 (MaxPooling2	(None, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
dropout (Dropout)	(None, 8192)	0
dense (Dense)	(None, 1024)	8389632
dense_1 (Dense)	(None, 1)	1025

Total params: 10,535,809
Trainable params: 10,535,809
Non-trainable params: 0

Fig. 4. CNN version 3.0 (unsuccessful)

```
def model():
    model = models.Sequential()
    model.add(Conv2D(32, (3, 3),
        activation='relu', input_shape=(500,
```

```
    500, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(64, (3, 3),
    activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3),
    activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(256, (3, 3),
    activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(512, (3, 3),
    activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(1024, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

return model
```

C. Optimizer

We then turned our attention to the optimizer, and after reviewing the documentation, we chose and tested among the following three optimizers.

Adam is an effective stochastic optimization method, which can calculate the adaptive learning rate of the stochastic objective function by estimating the low-order gradient.[5] This method not only has high computation rate, small memory demand, but also does not change the rescaling of the gradient diagonal. Therefore, it is more suitable to deal with a large number of data and parameters.

RMSProp is a gradient-based optimization method that standardizes the gradient by dividing the learning rate of a CNN weight by the moving average of the square of the gradient of that weight This normalization can well balance the step size and avoid the step size of gradient being too large or too small.[6] Therefore, it can be used for random techniques of small amount of learning.

SGD is an iterative method in gradient descent[7], which estimates a gradient from randomly selected data subsets and replaces the actual gradient, so that it can better reduce the computational load in high-dimensional optimization problems, and achieving faster iteration with faster convergence speed. Therefore, SGD is often used to train linear regression models.

After testing, SGD performed the best among these three, and as iteration was extended, the accuracy rate brought by SGD kept improving, finally reaching about 85%. It seems to be better at the field of dichotomous classification, while Adam and Rmsprop are more dominant in multiclassification image recognition.

D. Other Setting

To help the model training, I added the parameters reduce learning rate and early stopping.

1) Reduce Learning Rate: Reduce learning rate is generate from the Keras library "ReduceLROnPlateau()", the code is below.

When there are three training sessions in which the change in accuracy rate is less than 1, the learning rate is reduced by half.

```
reduce_lr = ReduceLROnPlateau(
                monitor='accuracy',
                factor=0.5,
                patience=3,
                verbose=1
            )
```

E. Early Stopping

Early stopping is generate from the Keras library "EarlyStopping", here I let it monitor the validation loss to prevent over fitting.

```
early_stopping = EarlyStopping(
                    monitor='val_loss',
                    min_delta=0,
                    patience=10,
                    verbose=2
                )
```

III. Training

In my training part, there were about I imported about 1000 images and divided them into a test machine and a training set based on a split ratio of 0.2. And set a total of 180 iterations.(We divided them into three training, the code and the figure below is the last time, from iteration 120 to iteration 180). Also we set loss as "binary_crossentropy" and initial training rate of 0.001.

```
train_generator,test_generator =
    processing_data(data_path, height=500,
    width=500, batch_size=batch_size,
    test_split=0.2)

model.compile(loss='binary_crossentropy',
          optimizer= SGD(lr=0.001),
          metrics=['accuracy'])

history = model.fit_generator(train_generator,
              epochs=180,

              steps_per_epoch= train_size//
                  batch_size,
              validation_data=test_generator,
              validation_steps=max(1,
                  test_size//batch_size),
              initial_epoch=120,
              callbacks=[checkpoint_period,
                  reduce_lr])
```

After training, our accuracy is around 84%, the highest training accuracy is around 88.96%.

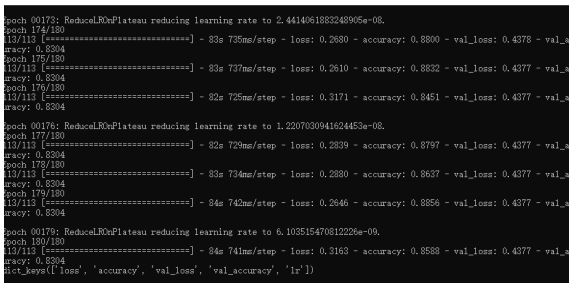

Fig. 5. Final Training Result

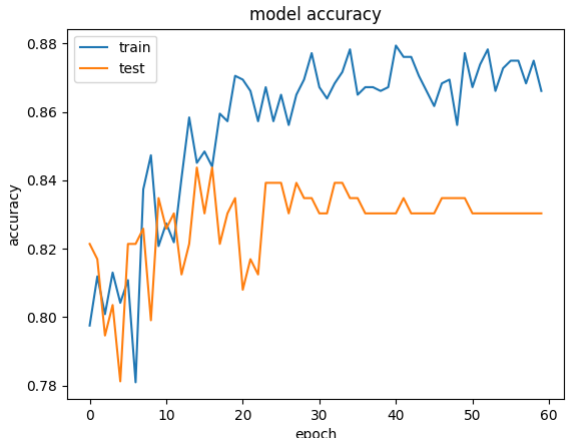

Fig. 6. Accuracy Curve

IV. Experiment Result

Finally, we loaded around 33128 (benign is 32542, malignant is 584) figure into our model. The accuracy of benign is 28085/32542 = 0.8599656, the accuracy of the malignant is 441/584 = 0.7037672. The total accuracy is 28526/33128 = 0.58721. We can find that we actually did not meet the requirements in malignant detection, and the overall correct rate is high because the sample with malignant is relatively too small.

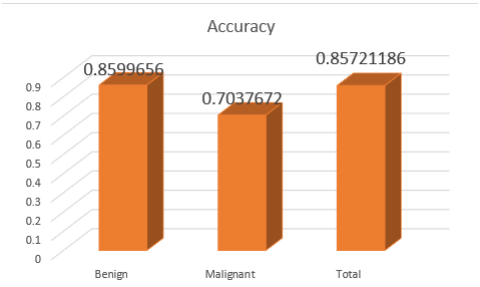

Fig. 7. Final Result

V. Website

A. Server and domain

We created a Ubuntu 18.04 Linux server on DigitalOcean, a cloud server provider. The server is used for users to upload their images and display the classification result. Apache2 was installed to enable https protocol, and domain was bound for user access convenience. (http://ece431.loserleo.com/)

B. Website Front-end and Bcak-end

We have finished the building of the server and the preliminary front-end code writing of the webpage required by the project. The interface of the website is as follows figure(Fig. 8) . In addition, we have tentatively completed a simple web design and made the website back-end and front-end connection, and we are debugging. On our website, the user can select the picture and upload the picture, the webpage interface will show the picture you uploaded and the result after the image recognition. In the future, we will optimize the interface.

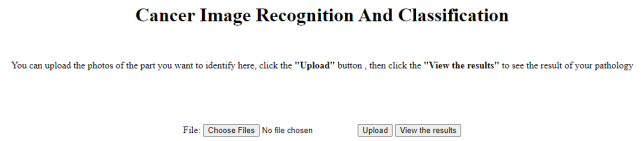

Fig. 8. Website view

VI. Summary

Overall, although the overall accuracy reached the minimum standard we set before, the 70% accuracy in malignant was not as good as it could have been, and the project became more like a classifier model of whether it was benign or not. Still, we learned a lot about the structure of convolutional neural networks, optimizers, and algorithms through this semester's project, and it was still a very interesting project!

VII. Reference

[1] "SIIM-ISIC Melanoma Classification", Kaggle, August 2020 https://www.kaggle.com/c/siim-isic-melanoma-classification/data

[2] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. "ImageNet classification with deep convolutional neural networks".May 2017.Communications of the ACM. 60 (6): 84–90. doi:10.1145/3065386. ISSN 0001-0782. S2CID 195908774.

[3] Pawar, Dipti. “Improving Performance of Convolutional Neural Network!” Medium, Medium, 3 Oct. 2020, medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7.

[4] LeCun, Y.; Bottou, L.; Bengio, Y. Haffner, P. Gradient-based learning applied to document recognition. (1998). Proceedings of the IEEE. 86(11): 2278 - 2324

[5] Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. 2014. arXiv:1412.6980v9

[6] Bushaev, Vitaly. “Understanding RMSprop — Faster Neural Network Learning.” Medium, 2 Sept. 2018, towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a.

[7]Bottou, Léon; Bousquet, Olivier. "The Tradeoffs of Large Scale Learning". In Sra, Suvrit; Nowozin, Sebastian; Wright, Stephen J. (eds.). Optimization for Machine Learning. Cambridge: MIT Press.(2012) pp. 351–368. ISBN 978-0-262-01646-9