# Implementation – CNN

Convolution Neural Network (CNN) performs well in image processing and prediction because of its special features. First, there is not all connections between the convolution neurons. This feature would lead to that CNN is sensitive to partial features in the images and not such sensitive to where the features are located. Thus, CNN performs friendly in image shifting and rotation which wouldn’t influence the object recognition. Meanwhile, CNN also has shared weights which can increase the ability of feature recognition and reduce the scale of the model.

In this project, Keras package is used to build this network in Python.

## Data Augmentation and Image Preprocessing

To increase the model prediction accuracy, it’s necessary to do the Data Augmentation and Image preprocessing. In this project, we find the images don’t have a predefined size. The smaller size could be 1000x1000 pixels and the larger one is 2000x2000. In the image augmentation part, the pixel value is rescaled from 1 to 255. And the images can do the width and height shift which is the 0.1 of the totals in training images. The validation and test images are only be rescaled the pixel values.

A screenshot of a computer

Description automatically generated

Function ‘ImageDataGenerator()’ works for image augmentation. In the following, ‘flow\_from\_directory’ will read the data from local directory and transform the RGB images to grayscale to reduce the complex of the model. In the preprocessing, the image size is also reshaped to the same size. The shape of a preprocessed image is [256, 256, 1].

A picture containing monitor, computer, screen, display

Description automatically generated

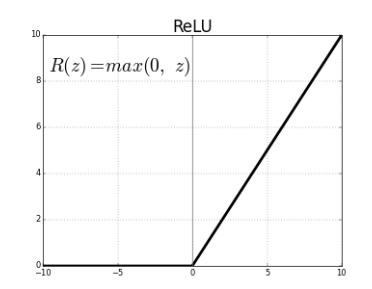
The image above shows the preprocessed results. These chest x-ray images are selected from a random batch and each batch contains 32 chest images. In the following steps, those images are processed under batch.

## Convolution Neural Network Model Building

A screenshot of a cell phone

Description automatically generated

In this section, CNN model was built referring from a chest classification project which build based on TensorFlow (Jang, 2020). This model contains 33 layers in total. The first layer is the input layer which defines the input images shape, [256, 256, 1]. In the following convolution layers, the kernel size is 3 which is the 3x3 matrix kernel to process image pixels sequentially. Activation function is all the convolution layers is The Rectified Linear Unit (ReLU). The image below is the plot graph of ReLU. In this graph, when the input is negative, the output will be 0. This function will solve the negative input pixel value problem and be rapid convergence.



The first two convolution layers use normal functions with 16 filters. The kernel is convolved with the input to product the tensor as the output (*Keras documentation: Conv2D layer*). The following 8 convolution layers use depthwise separable 2D convolution functions. The difference between this function and the previous is that this function performs depthwise convolution separately first. Then, it does the pointwise, 1x1 kernel convolution, to get the output. After every two convolution layers, a pooling layer is added to reduce the input the feature graph and simplify the network. The function MaxPooling2D is to find the max value in the covered region as the output feature. The dropout layer works on disconnecting neurons’ links randomly to prevent overfitting.

After finishing the convolution layers, the flatten layer works on convert 2D data into 1D which is the prerequisite input size of following dense layer. The following 4 dense layers work as most of the simple neural network with different unit size. The first 3 dense layers also have the ReLU activation function. The last activation function is Sigmoid because this is a binary classification problem.

## CNN Model Compiling and Training

Considering the size of normal and pneumonia training sample is different which may cause bias in the model. The weight of each category is added while training the model.

A screenshot of a cell phone

Description automatically generated A screen shot of a computer

Description automatically generated

In this part, the metrics of training model evaluation includes model accuracy, precision, and recall. Before training, model is compiled with the optimizer, Adam, and the loss function, binary\_crossentropy. The learning rate of this model is 0.0001. Learning rate is important but very hard to define. If the learning rate is higher, the model training wouldn’t take much time but probably cause very high loss. On the contrary, if the learning rate is lower than expect, it will take very long training time. At this time, I cannot find out a suitable learning rate in this project, but just set a very common value.

The right one above is the training CNN model. In this model, times of iteration is 25. Meanwhile, validation data is set to validate the model while training. The validation data will not be included in training model to reduce the probability of model overfitting. In this case, there are 8 samples, 4 normal and 4 pneumonia, for validation. The weight of normal sample is 1.94, and pneumonia is 0.67.

## Model Finetuning

After training the model, next step is to do some fine tune of the model. In this project, the early stopping is set when the metrics in the 20 epochs are not improve while training, the training will stop. In this finetuning model, the number of epochs is 100.

A screenshot of a cell phone

Description automatically generated

# Data Analysis - CNN

This project CNN model would get two results, one for original CNN model, another for finetuning model. The accuracy, precision and recall we get are in the table below. Those metrics are evaluated by the test image data which contains 624 images.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Precision | Recall |
| CNN Model | 0.855 | 0.929 | 0.928 |
| Finetuning Model | 0.856 | 0.968 | 0.968 |

This table indicates the three metrics of the CNN model and Fine-tuning CNN model. In the Fine-tuning CNN model, the training stops at 96 epoch out of 100. The accuracy only improves about 0.001 which is very tiny. However, precision increases from 0.929 to 0.968 and Recall raises from 0.928 to 0.968.

The aim of this project is to recognize the pneumonia chest through x-ray. We want to predict real pneumonia chest cases into predicted pneumonia cases as much as possible. Thus, our target is only the pneumonia cases. Considering the confusion matrix, recall is True Positive (TP) cases divided by the total real positive cases (TP + FN). In this case, we want to predict all the pneumonia cases into pneumonia. To achieve this goal, this model would pay more attention to recall. In this model, recall value Improves 0.04 and the recall value is greater than 0.95 which performs well.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

The two screenshots above indicate the three metrics in the training, validation, and testing dataset. For the training part, it only records the last epoch. Comparing the validation and training results, the metrics is very close. However, for the testing results, the testing accuracy is lower than the training which means this model may overfitting.

A close up of a map

Description automatically generated

The image above shows the four plots from original CNN model. The blue line represents the training and orange one is validation. From the top two plots, the model precision and recall perform well in both training and validation data. When epoch is 25, the precision and recall of two datasets are very close. The bottom two plots show the change of model accuracy and loss. Those two orange lines are still not convergence in this model. But the trend of accuracy is start going up as training line shows. The trend of loss value is also going down.

The graph below is the four figures from finetune model training. It has the same meaning with the previous original CNN model one. In the top two figures, they show the precision and recall improve from the original CNN model which increase from 0.92 to 0.96. The bottom two also shows the accuracy and loss changing. Same with the previous CNN model, these two graphs are still not convergence in this case. However, we can easily observe the trends those two orange lines going which are more closing to the blue line. I believe in the further work, this model would become stable by modify the learning rate, batch size and structure of the model.

A close up of a map

Description automatically generated

# Discussion – CNN

Convolution Neural Network one of the most popular deep learning network nowadays which significantly performs well in images and videos recognition. In our class, I learnt a lot in Neural Network. And It’s the important basic knowledge for CNN. After finishing the Neural Network homework, I felt I have a deeper understanding of deep learning and it provided me a great start of CNN.

Comparing with Neural Network, CNN is much more complex but in the similar basic concept. The biggest difference between NN and CNN is that NN only uses the all connection to link the neurons. CNN only do all connect when the layer is close to output. Most of the layers in CNN are partial connection layer which perform well in edge or partial feature recognition.

It’s an interesting project. In the further work, I would target one learning rate and batch size analysis to reduce the model loss. Meanwhile, I would also want to analyze difference when using different convolution functions.

# Reference

Team, K. (n.d.). Keras documentation: Conv2D layer. Retrieved August 13, 2020, from https://keras.io/api/layers/convolution\_layers/convolution2d/

Jang, A. (2020, July 20). TensorFlow Pneumonia Classification on X-rays. Retrieved August 13, 2020, from https://www.kaggle.com/amyjang/tensorflow-pneumonia-classification-on-x-rays/notebook