Section 2. Life Sciences

https://doi.org/10.29013/ELBLS-23-2-27-35

Zijian Wang,
Avon Old Farms, CT, USA

THE IDENTIFICATION OF CEREBRAL HAEMORRHAGE THROUGH HEAD CT IMAGES AND COMPARISON OF THREE CONVOLUTIONAL MODELS

Abstract

With modern computers and medical advancements, we can find a better way to increase efficiency and reduce human error in the healthcare system. In computer vision, machine learning models can analyze and categorize patients' head CTs. This diagnosis process is faster and retains the accuracy of experienced healthcare professionals. In this study, we used deep learning algorithms to identify cerebral haemorrhage in CT images with different CNN (convolutional neural network) architectures – exception and inception. Cerebral haemorrhage is one of the most complex diseases to diagnose and treat in the world. By comparing the performances of simple CNN, exception model, and inception model, we can find the best model for this task.

Keywords: Cerebral Haemorrhage, Head CT Interpretation, Convolutional Neural Network, Inception Model, Xception Model.

1. Introduction

AI (Artificial Intelligence) is a popular field continuously being developed nowadays. One of the main applications of AI is in processing and interpreting images by heavily studying medical interpretation, healthcare professionals, and AI scholars. Cerebral haemorrhage is severe bleeding that happens into or around the brain tissue. People who had cerebral haemorrhage could lose essential functions such as speaking, walking, or understanding others. Even though they can go through rehabilitation therapy and get better, it is still a painful for the patients and their family members. For example, my uncle has a loving family with a steady job and a cute daughter. Yet, a cerebral haemorrhage destroyed all of these. Their life became ominous; my poor aunt had to attend to her daughter and husband simultaneously. Watching them go through this, I become interested in researching how we can detect cerebral haemorrhage effectively and inflict less pain on happy families. AI is one way to do it. Trained machine learning models can quickly determine whether a cerebral haemorrhage is happening by comparing the scanned Head CT images to the training data. This approach will be much more efficient and accessible than having each image analyzed by doctors.

1.1 What is neural network

Neural Network is a structure for computers to learn to identify pictures like the human brain. A Neural Network contains three layers: an input, hidden, and output layer. Each layer has some neurons that contain values of the picture. The picture is first changed into black and white and then separated into several pixels; each pixel has a value between 0.0 to 1.0 based on its grayscale; 1.0 is white, and 0.0 is black. Then these values are stored in the input layer. The computer then will use an algorithm to calculate the values and transfer those to the hidden layer. A Neural Network could have several hidden layers; each hidden layer's values depend on the upper hidden layer. Then, the final hidden layer calculates a value converted to an output label (See Figure 1).

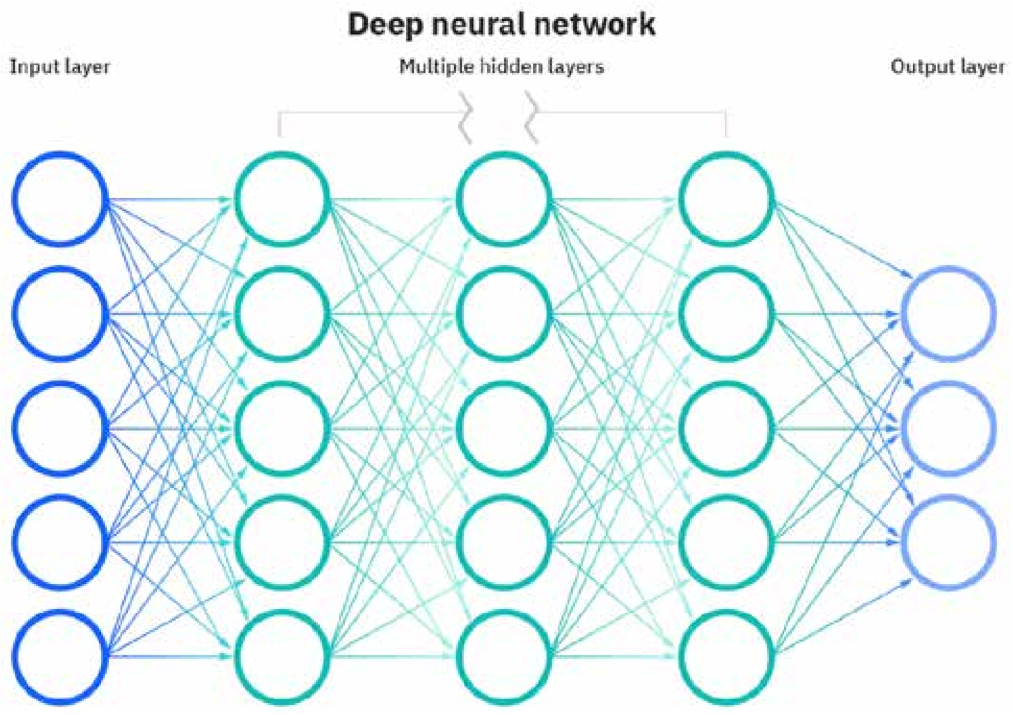


Figure 1. Simple Layers Structure of Neural Network

1.2 What is CNN

Convolutional Neural Network (CNN) is a deep learning algorithm that significantly contributes to image interpretation. There are mainly four parts to a CNN: Convolution, Padding, Striding, and Pooling. To begin with, Convolution is the most essential part of a CNN. It makes a feature map as the input image passes the filter layer. The new feature map decides the information gathered from the original image. The collected data depends on the dimension of the filter (See Figure 2). The filter moves at least one step for each pixel. For instance, if an m*m image passes through an n*n filter layer, the outcome image will be [(m–n) +1] *[(m–n) +1]. In other words, it picks some information from the original image and makes a new image containing these values. In this case, some of the information might be lost. Then, Padding helps to solve this problem. During the Padding process, the computer adds pixels on the margin of the filtered image that helps replenish some of the lost information based on the marginal pixels.

Padding the image size to be n + 2p – f +1. In the formula, n is the standard matrix dimension, p is the padding size, and f is the filter dimension. However, in some cases, people want to avoid the original picture passing the filter one step at a time; striding comes in for this. Striding means the filter can move more than one pixel, and the outcome image shape will be (n + 2p – f) / s = 1 (here, n, p, and f are defined the same as the formula above, and s is the unit of strides). Lastly, pooling helps the machine compute faster. It cuts the image into pieces and does specific calculations. There are two classical instances: max pooling and average pooling. Max pooling means picking the maximum value in each piece. Average pooling means calculating the average value of each piece.

Using CNN has several advantages compared to the traditional artificial neural network. First, nowadays, machine learning algorithms usually have to process large images. By convoluting the image, it will give the computing process less pressure. Convoluting the image will give the computing process less pressure because of the convolution. Several pixels pass the filter and become one pixel for the feature map. As a result, it will make the size smaller than the previous input. Secondly, CNN can identify the same target at different positions, an improvement from the traditional neural network. Finally, having multiple filters (one filter outputs one feature map) in a convolutional layer, simultaneously, several filters are applied to its input. Thus, it is capable of detecting multiple features anywhere in the input.

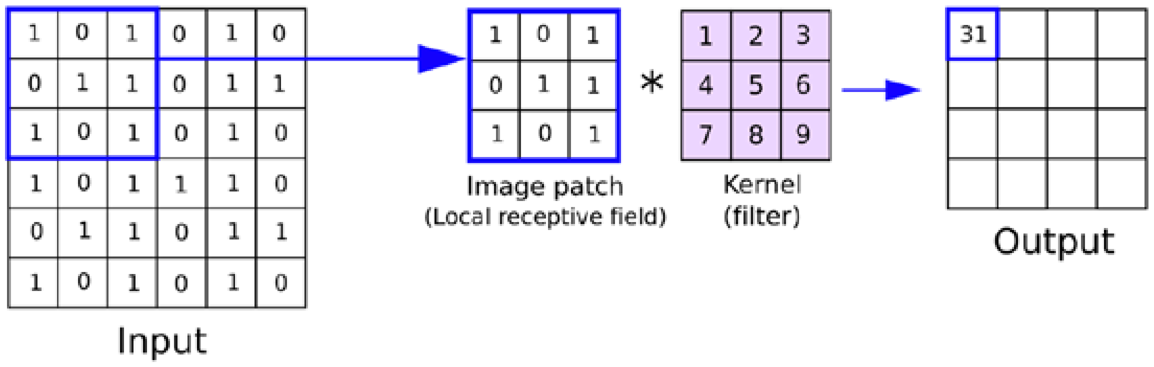


Figure 2. Simple Illustration of How CNN Layers Work in Convolution Process

1.3 How to Train a CNN Model

There are mainly two steps in training a CNN model: a forward phase and a backward phase, also known as backpropagation (backward propagation of errors). In the forward phase, it lets the image data go through the network from the input layer to the output layer. Each layer of the model will cache the image information. In backpropagation, it compares the output (prediction) to the original image label. Then it calculates the gradient for each layer of weights from the output layer back to the input layer – the neural-network version of gradient descent, which is to minimize the cost function. In the final layer, the activation function, SoftMax, calculates the possibility of the correct class. The loss function used is the cross-entropy loss. In other words, this algorithm helps to identify the labels.

$$L = -\ln(\rho_c)$$

The above formula is the Cross-entropy Loss. The P_c means the possibility of the correct class. Therefore, P is the possibility, and c is the suitable class.

1.4 Project Objective

This project aims to develop the best model to mark cerebral haemorrhages in CT pictures. Models included are regular CNN, Inception, and Xception, identifying the most efficient and accurate model by comparing their performance. In this project, by using transfer learning – using the model In this project, the accuracy of the model is improved by using transfer learning – using the model pre-trained by large amounts of data (imagenet). As a result, the absolute accuracy of the xception model is 0.93, which is the highest accuracy among all other models used in this project.

2. Methods

2.1 Preparation

The first step is to import the cerebral haemorrhage CT pictures and their corresponding labels to the Collaboratory notebook (the coding environment). Then, the heights and widths of the photographs were examined and unified. The original dataset included half normal brain CT and half cerebral haemorrhage CT images (See Figure 3).

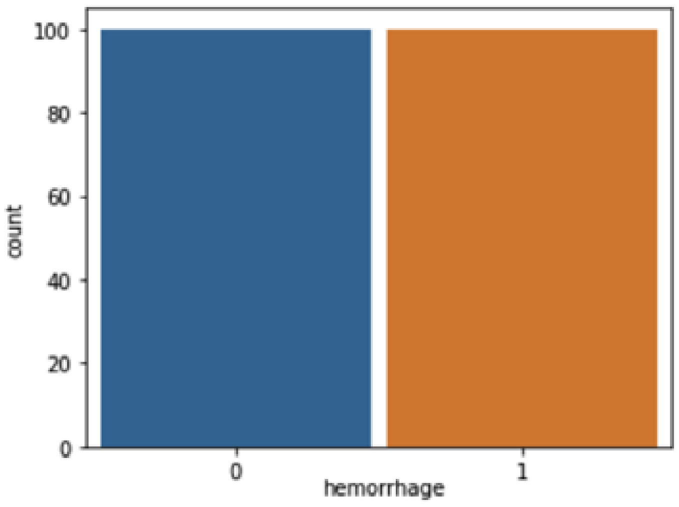


Figure 3. Count Plot of Labels

Selected and grouped at random, 60% of the pictures as the training data, and the rest were validation and test data.

2.2 Image Augmentation

Image augmentation can help CNN learn better by the invariant transformation pictures to feed the model for training, meaning that a photo will copy itself and make some shifts horizontally or vertically (See Figure 4). The CT scan produces nine images. It helps to increase the diversity of the training samples for the model. This step is crucial because deep learning requires a lot of data to train. The more training sample it has, the less overfitting there will be and the more accurate judgment it will make – just like the human brain completing the preparation.

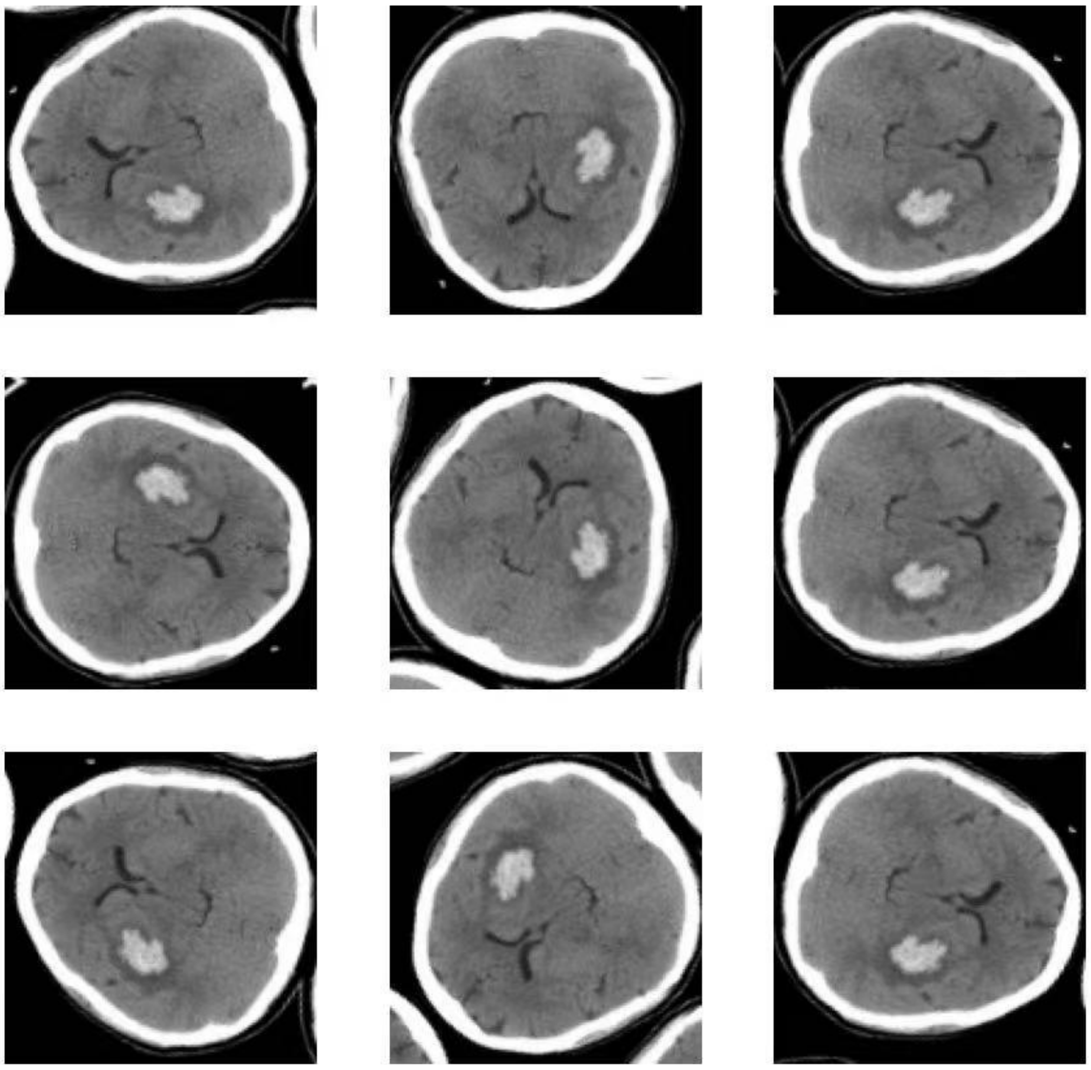


Figure 4. Image of Head CT After Image Augmentation

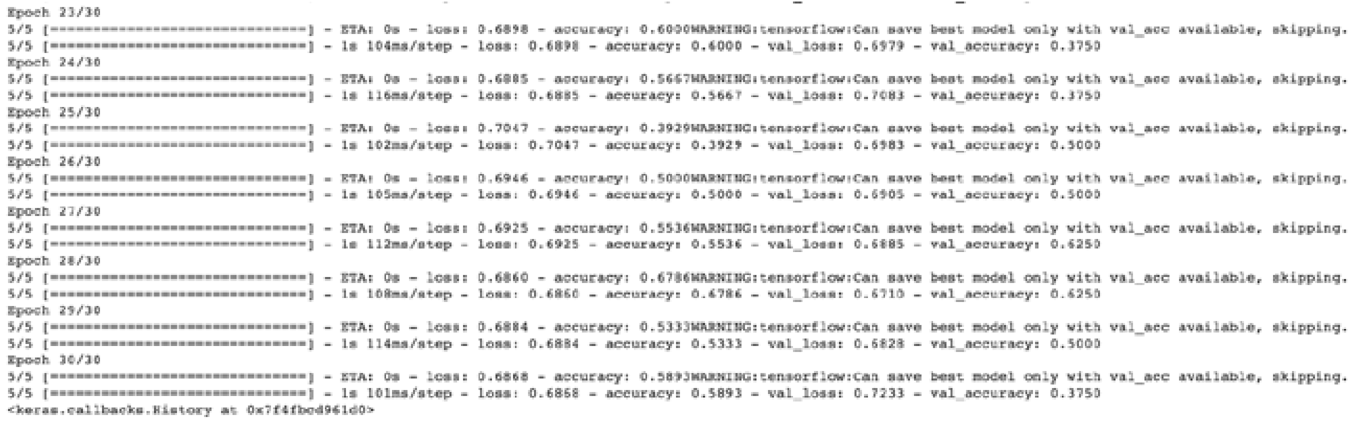
2.3 Transfer Learning

Transfer learning means to pre-train the model using a large database (imagenet) and applying the pre-trained model to similar problems. Transfer learning can also add specific criteria to solve complicated issues, especially computer vision problems. Moreover, not all transfer learning model is suitable for the project. Therefore, researchers need to determine which part of the research is valuable and if any necessary modifications and retesting might be required. In other words, it will transfer the knowledge that that applies to the current project and be trained with the existing data to become a new model to solve a specific problem. In my research, transfer learning helps to reduce training time, solve the small dataset problem, and to adapt to new images.

2.4 Simple CNN Model

A simple CNN model is the first model used in the computer vision problem. As mentioned before CNN model will convolute the input images from the input layers and then transfer them to other layers.

In the project, the simple CNN model used transfer learning to pre-train the base model. Unfortunately, in the outcome, the simple CNN model only has an accuracy of 0.59 (See Figure 5), which is the worst model in this project.

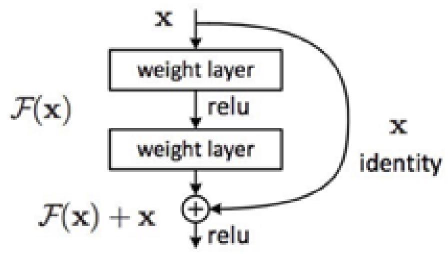


```
Epoch 23/30
5/5 [==============================] - ETA: 0s - loss: 0.6898 - accuracy: 0.6000WARNING:tensorflow:Can save best model only with val_acc available, skipping.
5/5 [==============================] - 1s 104ms/step - loss: 0.6898 - accuracy: 0.6000 - val_loss: 0.6979 - val_accuracy: 0.3750
Epoch 24/30
5/5 [==============================] - ETA: 0s - loss: 0.6885 - accuracy: 0.5667WARNING:tensorflow:Can save best model only with val_acc available, skipping.
5/5 [==============================] - 1s 116ms/step - loss: 0.6885 - accuracy: 0.5667 - val_loss: 0.7083 - val_accuracy: 0.3750
Epoch 25/30
5/5 [==============================] - ETA: 0s - loss: 0.7047 - accuracy: 0.3929WARNING:tensorflow:Can save best model only with val_acc available, skipping.
5/5 [==============================] - 1s 102ms/step - loss: 0.7047 - accuracy: 0.3929 - val_loss: 0.6983 - val_accuracy: 0.5000
Epoch 26/30
5/5 [==============================] - ETA: 0s - loss: 0.6946 - accuracy: 0.5000WARNING:tensorflow:Can save best model only with val_acc available, skipping.
5/5 [==============================] - 1s 105ms/step - loss: 0.6946 - accuracy: 0.5000 - val_loss: 0.6905 - val_accuracy: 0.5000
Epoch 27/30
5/5 [==============================] - ETA: 0s - loss: 0.6925 - accuracy: 0.5536WARNING:tensorflow:Can save best model only with val_acc available, skipping.
5/5 [==============================] - 1s 112ms/step - loss: 0.6925 - accuracy: 0.5536 - val_loss: 0.6885 - val_accuracy: 0.6250
Epoch 28/30
5/5 [==============================] - ETA: 0s - loss: 0.6860 - accuracy: 0.6786WARNING:tensorflow:Can save best model only with val_acc available, skipping.
5/5 [==============================] - 1s 108ms/step - loss: 0.6860 - accuracy: 0.6786 - val_loss: 0.6710 - val_accuracy: 0.6250
Epoch 29/30
5/5 [==============================] - ETA: 0s - loss: 0.6884 - accuracy: 0.5333WARNING:tensorflow:Can save best model only with val_acc available, skipping.
5/5 [==============================] - 1s 114ms/step - loss: 0.6884 - accuracy: 0.5333 - val_loss: 0.6828 - val_accuracy: 0.5000
Epoch 30/30
5/5 [==============================] - ETA: 0s - loss: 0.6868 - accuracy: 0.5893WARNING:tensorflow:Can save best model only with val_acc available, skipping.
5/5 [==============================] - 1s 101ms/step - loss: 0.6868 - accuracy: 0.5893 - val_loss: 0.7233 - val_accuracy: 0.3750
<keras.callbacks.History at 0x7f4fbcd961d0>
```

Figure 5. Result of Simple CNN Model

2.5 Res Net

With the development of deep learning, models are getting deeper and deeper. However, one problem appeared: sometimes more layers a model has, the worse it will perform. In simple words, residual learning helps the machine have more layers without losing information during deep learning. Initially, a model with x layers is as accurate as the x + 1 layer's model when just copying previous layers and identities of the last layer. However, it does not; the accuracy is gradually decreasing. To solve this issue, the author of ResNet started with the hypothesis that it's hard for the machine to learn direct information. In this case, ResNet helps it to learn a function by learning the differences. For instance, instead of learning a function F(x) directly, ResNet blocks will attempt to learn G(x) + x when F(x) = G(x) – x. Each block of the ResNet can add the features element-wise (See Figure 6). This method essentially helped solve the problem of vanishing gradient, which means losing the gradient signals since they cannot trace back to previous layers. By using ResNet, signals are allowed to travel back to early layers.



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 6. ResNet Block

2.6 Inception

From the regular convolutional neural network, each output image comes from the previous layers. If there are lots of hidden layers, it increases the computational cost. In this case, the Inception model solves this problem by using parameters more efficiently. Instead of convolute each layer for one filter board, Inception does this more comprehensively, which means the input signal is fed to four different layers. Each layer uses a different kernel size, allowing them to capture patterns at different scales, as shown in (Figure 7). There is a one-by-one convolution layer, a three-by-three convolution layer, a five-by-five convolution layer, and a three-by-three max pooling layer. The final depth concatenation layer stacks the feature maps from all four previous layers. In simple words, the inception module can output feature maps that capture complex patterns at various scales.

In the project, using transfer learning, the inception model has a final result of 0.91 accuracy (See Figure 8). The result is the second-highest accuracy compared to other models.

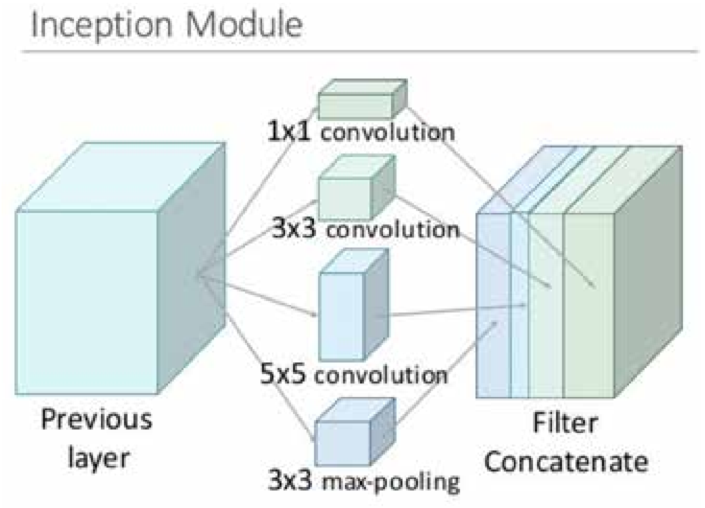
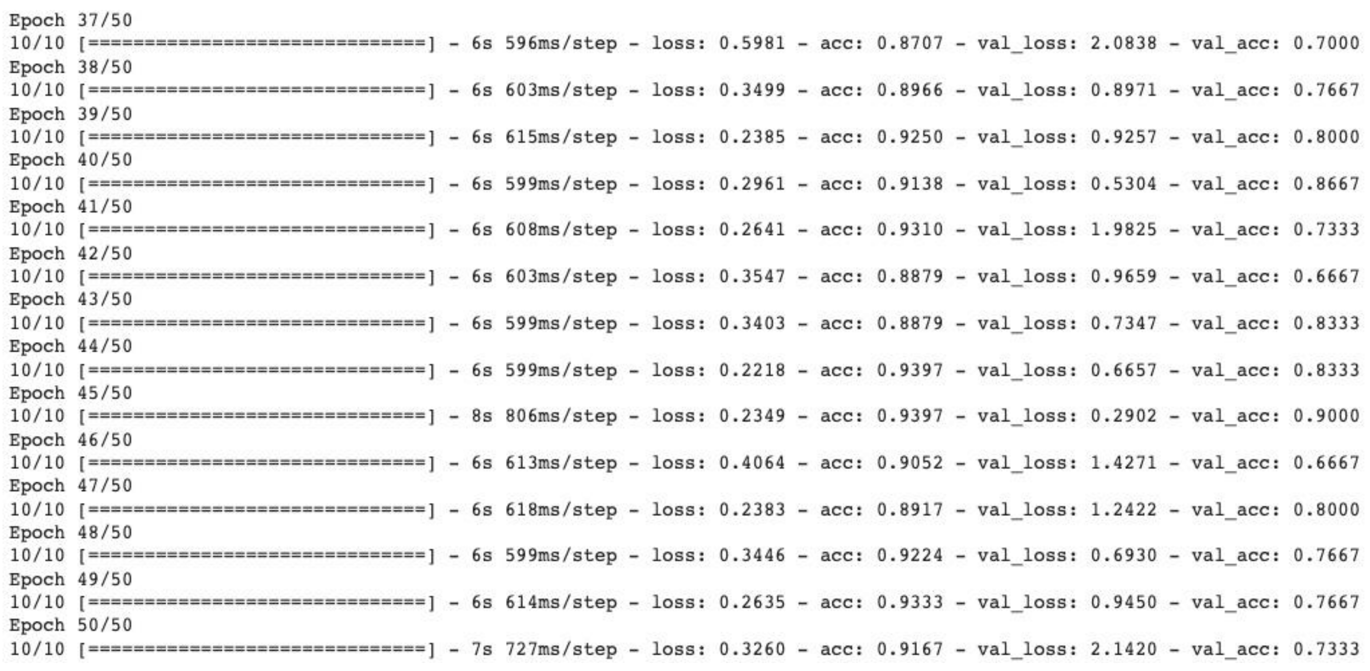


Figure 7. Inception Model



```
Epoch 37/50
10/10 [==============================] - 6s 596ms/step - loss: 0.5981 - acc: 0.8707 - val_loss: 2.0838 - val_acc: 0.7000
Epoch 38/50
10/10 [==============================] - 6s 603ms/step - loss: 0.3499 - acc: 0.8966 - val_loss: 0.8971 - val_acc: 0.7667
Epoch 39/50
10/10 [==============================] - 6s 615ms/step - loss: 0.2385 - acc: 0.9250 - val_loss: 0.9257 - val_acc: 0.8000
Epoch 40/50
10/10 [==============================] - 6s 599ms/step - loss: 0.2961 - acc: 0.9138 - val_loss: 0.5304 - val_acc: 0.8667
Epoch 41/50
10/10 [==============================] - 6s 608ms/step - loss: 0.2641 - acc: 0.9310 - val_loss: 1.9825 - val_acc: 0.7333
Epoch 42/50
10/10 [==============================] - 6s 603ms/step - loss: 0.3547 - acc: 0.8879 - val_loss: 0.9659 - val_acc: 0.6667
Epoch 43/50
10/10 [==============================] - 6s 599ms/step - loss: 0.3403 - acc: 0.8879 - val_loss: 0.7347 - val_acc: 0.8333
Epoch 44/50
10/10 [==============================] - 6s 599ms/step - loss: 0.2218 - acc: 0.9397 - val_loss: 0.6657 - val_acc: 0.8333
Epoch 45/50
10/10 [==============================] - 8s 806ms/step - loss: 0.2349 - acc: 0.9397 - val_loss: 0.2902 - val_acc: 0.9000
Epoch 46/50
10/10 [==============================] - 6s 613ms/step - loss: 0.4064 - acc: 0.9052 - val_loss: 1.4271 - val_acc: 0.6667
Epoch 47/50
10/10 [==============================] - 6s 618ms/step - loss: 0.2383 - acc: 0.8917 - val_loss: 1.2422 - val_acc: 0.8000
Epoch 48/50
10/10 [==============================] - 6s 599ms/step - loss: 0.3446 - acc: 0.9224 - val_loss: 0.6930 - val_acc: 0.7667
Epoch 49/50
10/10 [==============================] - 6s 614ms/step - loss: 0.2635 - acc: 0.9333 - val_loss: 0.9450 - val_acc: 0.7667
Epoch 50/50
10/10 [==============================] - 7s 727ms/step - loss: 0.3260 - acc: 0.9167 - val_loss: 2.1420 - val_acc: 0.7333
```

Figure 8. The Result of Inception Model

2.7 Xception Model

Xception means the extreme inception. In other words, it allows the inception model to be the most efficient at training and testing. The difference between the inception model and the exception model is the depth-wise separable convolution layer based on the idea of spatial and cross-channel patterns can be modeled separately. The separable convolution layers have one spatial filter per input channel. In detail, it filters the input image first and then compresses it into the 1 * 1 deep convolutional compression to catch the Cross-Chanel correlation (See Figure 9). It also includes a depth-wise convolution and a point-wise convolution which means to break mapping 3D models by mapping it by 1D model plus 2D model. As a result, it decreases the computational cost. Secondly, after the operation, the exception model will not introduce any non-linearity. Therefore, the exception model increases the image interpretation accuracy and performs better in big data analysis.

In this project, using transfer learning- improves the model's accuracy. As a result, the final accuracy of the xception model in the project is 0.93 (See Figure 2.7.2), which is the highest accuracy among all other models tried in this project.

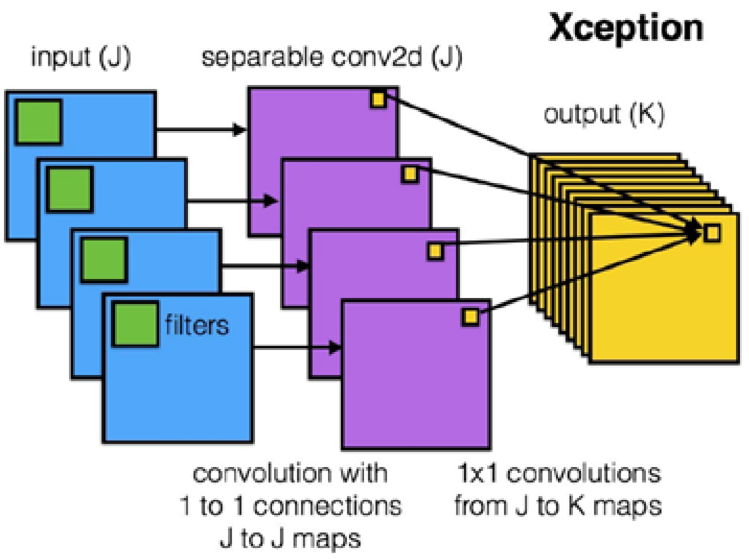
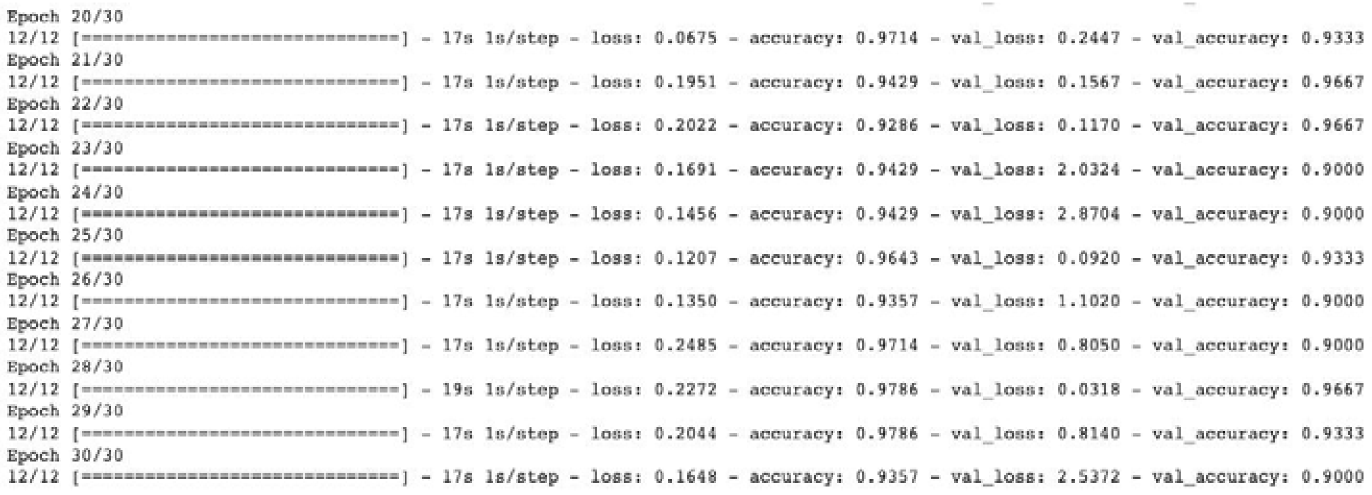

Xception
input (J)
separable conv2d (J)
output (K)
filters
convolution with 1 to 1 connections J to J maps
1x1 convolutions from J to K maps

Figure 9. The Simple Structure of Xception Model



```
Epoch 20/30
12/12 [==============================] - 17s 1s/step - loss: 0.0675 - accuracy: 0.9714 - val_loss: 0.2447 - val_accuracy: 0.9333
Epoch 21/30
12/12 [==============================] - 17s 1s/step - loss: 0.1951 - accuracy: 0.9429 - val_loss: 0.1567 - val_accuracy: 0.9667
Epoch 22/30
12/12 [==============================] - 17s 1s/step - loss: 0.2022 - accuracy: 0.9286 - val_loss: 0.1170 - val_accuracy: 0.9667
Epoch 23/30
12/12 [==============================] - 17s 1s/step - loss: 0.1691 - accuracy: 0.9429 - val_loss: 2.0324 - val_accuracy: 0.9000
Epoch 24/30
12/12 [==============================] - 17s 1s/step - loss: 0.1456 - accuracy: 0.9429 - val_loss: 2.8704 - val_accuracy: 0.9000
Epoch 25/30
12/12 [==============================] - 17s 1s/step - loss: 0.1207 - accuracy: 0.9643 - val_loss: 0.0920 - val_accuracy: 0.9333
Epoch 26/30
12/12 [==============================] - 17s 1s/step - loss: 0.1350 - accuracy: 0.9357 - val_loss: 1.1020 - val_accuracy: 0.9000
Epoch 27/30
12/12 [==============================] - 17s 1s/step - loss: 0.2485 - accuracy: 0.9714 - val_loss: 0.8050 - val_accuracy: 0.9000
Epoch 28/30
12/12 [==============================] - 19s 1s/step - loss: 0.2272 - accuracy: 0.9786 - val_loss: 0.0318 - val_accuracy: 0.9667
Epoch 29/30
12/12 [==============================] - 17s 1s/step - loss: 0.2044 - accuracy: 0.9786 - val_loss: 0.8140 - val_accuracy: 0.9333
Epoch 30/30
12/12 [==============================] - 17s 1s/step - loss: 0.1648 - accuracy: 0.9357 - val_loss: 2.5372 - val_accuracy: 0.9000
```

Figure 10. The Result of Xception Model

3. Conclusion

The results show that some models, such as the Xception and inception models, performed much better than the simple CNN model in identifying whether the Head CT images have a cerebral haemorrhage. Comparing their performances, Simple CNN only had 46% accuracy while the other two (inception model and Xception model) had about 91% and 93%. Hence the Xception model is the model of choice for this research.

Based on the National Institute of Health, ten in every 100.000 people in the United States get cerebral haemorrhage. It is a considerable number in total and still increasing every year. Several reasons, such as head trauma, blood vessel anomalies, cerebral aneurysm, or some liver illness, may cause cerebral haemorrhage. The symptom will happen imminently with severe headaches and loss of balance. In addition, people with cerebral haemorrhage might lose consciousness, mobility, and linguistic abilities. In this project, by using machine learning models, we can identify whether a patient has cerebral haemorrhage through the CT image with faster and more accurate results. It will significantly increase the efficiency of hospitals in the diagnosis phase, which is particularly important in remote areas where the lack of medical professionals is a problem.

This study shows machine learning potential to decrease the damage and total rate of patients who suffer from this disease. However, some questions remain for further research, such as whether other models or combining different models achieve better accuracy and whether a larger dataset with more variations will change the result. I hope to keep working on this topic so that machine-learning models for cerebral haemorrhage identification can become a reality and be widely used by the public.

Acknowledgment

I want to thank my mentor Dr. Ying Qin Taylor, for the professional advice and encouragement for the project. Also, thanks to my parents, Yang Wang and Yang Sun, for their support throughout this project.

References:

1. Reynolds Anh H. "Convolutional Neural Networks (Cnns)." Anh H. Reynolds, 15 Oct. 2017. URL: https://anhreynolds.com/blogs/cnn.html.
2. IBM Cloud Education. "What Are Neural Networks?" IBM. URL: https://www.ibm.com/cloud/learn/neural-networks.
3. Seldon. "Transfer Learning for Machine Learning." Seldon, 29 June 2021.URL: https://www.seldon.io/transfer-learning#:~: text=Transfer%20learning%20means%20taking%20the, to%20solve%20a%20specific%20task.
4. Xu Joyce. "An Intuitive Guide to Deep Network Architectures." Medium, Towards Data Science, 15 Aug. 2017. URL: https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41#:~: text=Xception%20stands%20for%20%E2%80%9Cextreme%20inception, of%20Inception%20to%20an%20extreme
5. Peixeiro Marco. "Introduction to Convolutional Neural Networks (CNN) with Tensorflow." Medium, Towards Data Science, 19 May 2020. URL: https://towardsdatascience.com/introduction-to-convolutional-neural-networks-cnn-with-tensorflow-57e2f4837e18
6. Peixeiro Marco. "Step-by-Step Guide to Building Your Own Neural Network from Scratch." Medium, Towards Data Science, 19 May 2020. URL: https://towardsdatascience.com/step-by-step-guide-to-building-your-own-neural-network-from-scratch-df64b1c5ab6e

7. Quetscher Felizia. "A Comprehensible Explanation of the Dimensions in Cnns." Medium, Towards Data Science, 24 June 2021. URL: https://towardsdatascience.com/a-comprehensible-explanation-of-the-dimensions-in-cnns-841dba49df5e
8. Zhou Victor. "Training a Convolutional Neural Network from Scratch." Medium, Towards Data Science, 6 June 2019. URL: https://towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754
9. Felman Adam. "Brain Hemorrhage: Causes, Symptoms, and Treatments." Medical News Today, Medi-Lexicon International, 2019. URL: https://www.medicalnewstoday.com/articles/317080#symptoms.
10. Zhou Victor. "CNNs, Part 1: An Introduction to Convolutional Neural Networks." Victor Zhou, Victor Zhou, 22 May 2019. URL: https://victorzhou.com/blog/intro-to-cnns-part-1/#41/implementing-pooling
11. Kwiatkowski Robert. "Gradient Descent Algorithm – a Deep Dive." Medium, Towards Data Science, 13 July 2022. URL: https://towardsdatascience.com/gradient-descent-algorithm-adeep-dive-cf04e8115f21
12. "Backpropagation." Brilliant Math & Science Wiki. URL: https://brilliant.org/wiki/backpropagation
13. Techslang. "What Is a Convolutional Neural Network? – Definition by Techslang." *Techslang*, Techslang – Today's Most Spoken Tech Explained, 12 Nov. 2020. URL: https://www.techslang.com/definition/what-is-a-convolutional-neural-network
14. Paradi Himarka et al. "Multi Modal Learning with Deep Neural Networks – Anits." Multi Modal Learning with Deep Neural Networks A PROJECT REPORT, 2022. URL: http://cse.anits.edu.in/projects/projects1920B17.pdf.
15. ost@lincoln.ac.uk, Online Services Team; "Volumetric Estimation of Cystic Macular Edema in OCT Scans." Volumetric Estimation of Cystic Macular Edema in OCT Scans – The, 1 Jan. 1970. URL: https://eprints.lincoln.ac.uk/id/eprint/38920//1/Volumetric_Estimation_Of_Cystic_Macular_Edema_In_OCT_Scans.pdf.
16. Zhou Shuangxi et al. "Crack Texture Feature Identification of Fiber Reinforced Concrete Based on Deep Learning." Materials (Basel, Switzerland) – Vol. 15.– 113940. 1 Jun. 2022. Doi:10.3390/ma15113940
17. Wang Zhu. "A Rom-Accelerated Parallel-in-Time Preconditioner for Solving All-at …" A ROM-Accelerated Parallel-in-Time Preconditioner for Solving All-at-Once Systems in Unsteady Convection-Diffusion PDEs, 2021. URL: https://people.math.sc.edu/wangzhu/reference/Liu2021ParaDIAG_PinT_ROM_zw.pdf.
18. Géron Aurélien. "3. Convolutional Neural Networks." Ebookreading.Net. URL: https://ebookreading.net/view/book/EB9781492037354_5.html.
19. Xia Xiao. "CNN Architectures – Googlenet/Resnet/Vggnet." CNN Architectures – GoogLeNet/ResNet/VGGNet, 2022. URL: https://zhuanlan.zhihu.com/p/426100290