*IB Subject(s): Computer Science (Group 4 Science)*

*Extended Essay*

**The Effect of Using Different Activation Function in Convolutional Neural Network**

**To what extent is activation function ReLU more effective than Sigmoid in reducing cost function in image classification?**

**Word count:**

**Introduction: (tied to the question)**

Since the last century, the amount of computation required for research in various subjects has skyrocketed. Researchers often need to observe large amounts of data and find out trends in the data in a model; or classify the input data by features. To help people accomplish this, machine learning has emerged.

Machine learning is a branch of artificial intelligence, abbreviated as AI. AI is a very broad topic, but it is also limited in its capabilities. It is only capable of classifying data according to features set by the program writer, like determining whether a person is going to ask a question based on whether he raises his hand or not. Machine learning, on the other hand, is to allow the program itself to adjust bias and weights to find correlations between data through training with large amounts of data.

But at this point, machine learning's ability to analyze and process data is somewhat weak in the face of image classification. This is when the idea of deep learning was introduced, and for image classification, the most used neural network is the Convolutional Neural Network, or CNN.

Deep learning, based on machine learning, uses neural networks to analyze data, and its procedure is very similar to that of machine learning. They all collect and prepare the data first, removing missing values or outliers; perform feature engineering; train the model; and evaluate the model.

In a neural network, the input of the activation function is the output of a neuron from the previous layer, and the output of the activation function is the inputs of neurons in the next layer. Using different activation functions will hence lead to different outputs with the same input. For CNN, it uses ReLU, or Rectified Linear Unit, as the activation function. While there is another widely used activation function called Sigmoid. Since both activation functions are widely used in the world, yet CNN uses only ReLU, ReLU must have its superiority compared to Sigmoid.

As mentioned, the model trained by CNN will be evaluated, and the value of cost function is one of the metrics. A cost function measures the performance of a model and return a number representing the difference between the predicted value and the actual value. higher the value of cost function, the lower the effectiveness and accuracy of the model. If ReLU is used in a wider range than Sigmoid in CNN, then to what extent is activation function ReLU more effective than Sigmoid in reducing cost function in image classification?

**Theory:**

Image classification refers to the process of extracting information from raster images and identify the category of the input image. One single image is usually composed of hundreds of thousands of pixels with three channels representing the color. When the image is put into the model as one input, the information for each pixel needs to be processed by the model. In other words, if one black and white image is , then there should be neurons in the input layer of the CNN to process the information. If the image’s resolution is even higher, the number of neurons will be unmanageable. This is where CNN appears and solve the issue.

Before putting the image into the neural network, CNN extracts the feature of image and convert it into lower dimension without losing its characteristics, so that the number of pixels need to processed decreases.

A typical Convolutional Neural Network has six main layers: input layer, convo layer, pooling layer, fully connected layer, SoftMax layer, and output layer. They follow the flowchart of this: Input -> Convolution (ReLU)-> Pooling -> Fully connected layer -> SoftMax -> Output.

In the input layer, the input image will be reshaped and form a single column of pixels, where a picture originally will be reshaped into .

Shape

Description automatically generatedIn the convolution layer, features of the image are extracted. The image will undergo convolution operation, where the information of the map will be compressed onto a filter map by calculating the dot product between the receptive field and the filter. The filter is a matrix composed of s and s. The result will be sent to the next layer. During the convolution operation, activation functions will input data values and output values according to their characteristics.

Pooling layer reduces the spatial volume of input image after convolution. This layer is necessary in reducing the computational cost. After the pooling layer, the data is set to fully connected layer and SoftMax layer for training.

Chart, line chart

Description automatically generatedAs mentioned, activation functions lie within the convolution layer. The main impact of using different activation functions is the output. For ReLU, if the input is a negative number, then all output will be 0, whereas any positive input will lead to the exact same output, as shown in the diagram below.

Chart

Description automatically generatedAs for sigmoid function, it follows the exponential function of *,* where all input values will be recalculated and assigned a value between 0 and 1. Due to the mathematical characteristic of logistic function, the larger the input, or x value, the closer the output to 1, yet the output will only infinitely approach to 1.

There are both advantages and disadvantages for using ReLU and Sigmoid.

ReLU is more computational efficient compared to sigmoid since it does not require calculation involved with exponential equations, and it only calculate input within and output 0 for the rest. This makes ReLU a more computational efficient activation function compared to sigmoid.

While those beneficial characteristics of ReLU also leads to a serious shortcoming, the dying ReLU problem. If too many activations’ results get below the value zero, and negative inputs are involved, then most of the neurons in network with ReLU will simply output zero as well since the input from the previous layer are mostly zero. If that is the case, the process of backpropagation will be hindered since the adjust in weights will not be reflected in the change of output value as weight acts like a multiplier and is always 0. Ultimately a large part of the network will become inactive, and the model can no longer learn further as neurons stuck in negative range and giving output 0.

Sigmoid function, in other hand, will not run into the dying ReLU problem since all input values will be compressed within the range 0 and 1 with no information lost for calculation within a single neuron.

Yet there will be information lost for neurons with sigmoid function when there are multiple hidden layers, that is, the vanishing gradient. Since the essence of deep learning is gradient decent, where the model tries to the find the “path” to the local minimum. And this path follows the points with maximum gradients. Gradient decent is based on a mechanism called backpropagation, where it finds the derivatives of the network by moving layer by layer from the final layer to the initial one. According to the chain rule, the derivatives of each layer are multiplied down the network.

Sigmoid function, however, follow the function of , by looking at its first derivative, which is , all results calculated are less than 1 since is always greater than for all in put . If there are hidden layers in the network, then derivatives that is less than 1 are multiplied together. Thus, the gradient decreases exponentially as the model propagate down to the initial layers. Since the weights and bias are adjusted according to the gradient, a small gradient value means that the weights and biases of the initial layers will not be adjusted effectively.

Hence, if sigmoid is used in neural network, the computational cost will be higher due to more calculated operations; and the model might not converge that well due to the vanishing gradient issue.

In the process of determining whether ReLU is better in CNN, a distinction needs to be made. In deep learning there are two main types of models, one is the regression model, and the other one is the classification model. Regression model aims to find predict the output of a continuous value, like the price of a house or a probability. Classification model aims to select a class from a list of classes, like identifying whether the picture shows a dog or cat. In this essay that discuss image classification, the main focus is on classification model. The accuracy and the loss value will be measured and compared.

**Methodology:**

This experiment serves to evaluate the effectiveness of ReLU activation function compared to sigmoid activation function in Convolutional Neural Network. To collect sufficient data and reduce uncertainties, multiple trials of experiment are needed for the same set up to avoid the situation where the gradient decent process leads the training model to a local minimum point that is still very inaccurate.

The independent variable in the experiment is the different combination of ReLU and sigmoid function in the convolution layer. The dependent variable is the cost function value and the accuracy.

The code used in the experiment comes from towardsdatascience.com, article *“Convolutional Neural Network”* by dshahid380. The data used in the experiment is the CIFAR-10 dataset, with 10 different labels and 60000 image data. The ratio between training data set and testing data set is 5 : 1.

During the experiment, a total of 6 sets of combinations of activation functions will be tested. Since there is a total of six convolution layer in the network, the combination will be shown in the list below, with R representing ReLU and S representing sigmoid.

To reduce the result difference due to different orders of ReLU and sigmoid, two additional sets of experiment will be tested. They are and . Hence all combinations are:

There will be 5 trials in each set of experiment, the cost function value and accuracy for each of them will be recorded. The procedure is shown below.

1. Text

   Description automatically generatedConstruct Convolutional Neural Network based on the tutorial on towardsdatascience.com, replace all activations for convolution layer (Conv2d) according to the combination. Examples are shown in the picture below, noticed that there are a total of six convolution layers and all of them need to be changed accordingly.
2. Run the program with epoch equals to , record the cost function value (val\_loss) and accuracy for the result.
3. Repeat step two for 4 more times, record the data.
4. Switch to the next combination of ReLU and sigmoid, repeat from step two to step three.

As for clarification, the amount of data inputted is constant, the content will be the same as well to make sure if the cost function goes into a local minimum point, it will be the same for each set or trial of the experiment.

Noticed that in the experiment, the val\_loss value is recorded instead of the loss value. The loss value is the cost function value during training, the value is calculated based on the performance of the model in the train data set. While the val\_loss value is the cost function value during testing, the value is calculated based on the performance of the model in the test data set.

**Experiment Result 1:**

Raw Data Table:







Processed Data Table:







**Analysis 1:**

From the experiment, an interesting situation happens. The third, fifth and sixth combination, representing RSSRSS, SSRSSR and SSSSSS, have an extremely low accuracy and high val\_loss value. They both have the same accuracy of 0.1, meaning that the model doesn’t learn at all since there are 10 labels in the data, the probability of getting one correct with pure guess is 1/10, or 0.1. Besides, the val\_loss value remains at around 2.3 throughout the training of 100 epoch, meaning that the model does not converge.

A picture containing text

Description automatically generated

On the other hand, the other three combinations, RRRRRR, RRSRRS and SRRSRR present a relatively well-trained model. With combination RRRRRR having the lowest average val\_loss of and highest accuracy of 0.79656. Interestingly, although combination RRSRRS and SRRSRR have the exact same number of ReLU and sigmoid function inside the combination, and there are two RS and one RR combination, the RRSRRS combination has a lower val\_loss of 0.77086 and accuracy of 0.77434, compared to SRRSRR with val\_loss of 0.967825 and accuracy of 0.68226.

From the experiment, a hypothesis can be concluded that if a combination doesn’t have at least one RR combination of convolutional layer before each pooling layer, the model will not learn and perform any progress. While if two model has the same number of RR combination, the more advanced the order of occurrence of RR combinations in model, the better the learning ability of that model. And there are theories and clues that support this hypothesis.

In the code, six activation functions are divided into three sections, where there will be one pooling layer after each two convolutional layer. It is known that the process of multiplying derivatives happens between convolutional layer but stopped and restarted after going through a pooling layer.

A screenshot of a computer

Description automatically generated with medium confidence

Hence, the derivatives in two convolutional layers will multiply in this experiment. Sigmoid function itself will already output a low derivative value. As mentioned, the sigmoid function is represented as , and the first derivative of it is , or simply . The maximum value for is 0.5 when , hence the maximum derivative value is . Hence the gradient value that is provided by the loss function will get smaller and smaller as the model backpropagates it through the network since the gradient is multiplied with 0.25 every time. If two sigmoid functions are activation function in a pair of convolutional layers, it’s likely that vanishing gradient happens and the model loss the ability to backpropagate normally and can’t learn. While a combination of two ReLU doesn’t have the vanishing gradient issue, it is unclear that whether a combination of RS will perform well or not. Additionally, the reason why the order of ReLU and sigmoid in one combination matters is probably because of the process of backpropagation. In simple words, backpropagation will start at the last layer and multiply its derivatives with the previous layers’ derivatives. When the RR combination are in the front of the forward phase where outputs are calculated, it is at the back of backward phase, where backpropagation happens. If backpropagation process functions well, the cost function can be reduced in a more effective way.

To valid the hypothesis, additional experiments are performed. To eliminate the effect of the same ReLU and sigmoid combination but in different order of arrangement in the original experiment, three additional combinations are being tested. They are RRRRSS, RRSSRR and SSRRRR. If there is a difference between their val\_loss value and accuracy, then it is validated that the order of each section, each including two convolutional layers, will affect the effectiveness of the model. If the val\_loss value and accuracy is higher for those combinations that has its RR combination in the front of the forward phase, then probably ReLU combinations will have a higher positive impact on the backpropagation process compared to sigmoid combinations, proving that activation function ReLU is more effective in reducing cost function compared to sigmoid function.

**Experiment Result 2:**

Raw Data Table:





Processed Data Table:







**Analysis 2:**

From the experiment, three combinations all have two RR and one SS combinations. Yet the average val\_loss value and average accuracy for RRRRSS is the lowest, with RRSSRR in second and SSRRRR in third. It is also worth noticing that the standard deviation for RRRRSS is also the lowest, with an incredibly low value of . This represents that the RRRRSS combination provides a stable learning environment for the model, where the fluctuation for each trial is little. In other words, the model is almost converge using the RRRRSS combination because all experimental results using this combination has a low and close val\_loss value, meaning that it is closed to the local minimum point, at least closer to models using other combinations.

Since in the second experiment, models with RR combination in the front of its forward phase still perform better, as same as in the first experiment, a comparison of effectiveness in reducing cost function, or val\_loss value, between all combinations of ReLU and sigmoid can be made, as showed below.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Conclusion:**

After several experiments, it is clear that activation function ReLU is more effective than sigmoid in reducing cost function in image classification in terms of lower val\_loss value and higher accuracy. The main reason this situation happens is because the nature of sigmoid function leads to the issue of vanishing gradient in the process of backpropagation and cause the model not to learn.

The secrets of activation functions in Convolutional Neural Network are still a mystery yet. This essay only considers the situation where there are two convolutional layers in front of each pooling layer. For further investigation, it is reasonable to investigate the effectiveness of ReLU and sigmoid with single convolutional layer.

In general, the reduction of the value of cost function is strongly related to the activation function used within the model. It is clearly not enough to understand the impact of only two activation functions on the overall model. Famous activation functions such as leaky ReLU, tanh, etc. have not been covered yet. There is still a lot of room to explore.

**Reference:**

“Backpropagation.” *Brilliant Math & Science Wiki*, https://brilliant.org/wiki/backpropagation/.

“Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network.” *UpGrad Blog*, 16 Sept. 2022, https://www.upgrad.com/blog/basic-cnn-architecture/#:~:text=Convolutional%20Layer,-This%20layer%20is&text=In%20this%20layer%2C%20the%20mathematical,of%20the%20filter%20(MxM).

dshahid380. “Convolutional Neural Network.” *Medium*, Towards Data Science, 26 Feb. 2019, https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529#:~:text=Examples%20of%20CNN%20in%20computer,i.e%2C%20weights%2C%20biases%20etc.

Hindawi. “Image Classification Model Based on Deep Learning in Internet of Things.” *Image Classification Model Based on Deep Learning in Internet of Things*, https://www.hindawi.com/journals/wcmc/2020/6677907/.

Jeremy Jordan. “Neural Networks: Training with Backpropagation.” *Jeremy Jordan*, Jeremy Jordan, 15 June 2019, https://www.jeremyjordan.me/neural-networks-training/.

Kathuria, Ayoosh. “Intro to Optimization in Deep Learning: Gradient Descent.” *Paperspace Blog*, Paperspace Blog, 18 Dec. 2020, https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/.

Kim, Suhyun. “A Beginner's Guide to Convolutional Neural Networks (Cnns).” *Medium*, Towards Data Science, 15 Feb. 2019, https://towardsdatascience.com/a-beginners-guide-to-convolutional-neural-networks-cnns-14649dbddce8#:~:text=In%20convolutional%20networks%2C%20multiple%20filters,%2C%20say%2C%20a%20dark%20edge.

Leung, Kenneth. “The Dying Relu Problem, Clearly Explained.” *Medium*, Towards Data Science, 23 Sept. 2021, https://towardsdatascience.com/the-dying-relu-problem-clearly-explained-42d0c54e0d24.

M, Shruti. “Discover the Differences between AI vs. Machine Learning vs. Deep Learning [2022 Edition].” *Simplilearn.com*, Simplilearn, 13 July 2022, https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/ai-vs-machine-learning-vs-deep-learning.

Middleton, Michael. “Deep Learning vs. Machine Learning - What's the Difference?” *Flatiron School*, 9 Aug. 2022, https://flatironschool.com/blog/deep-learning-vs-machine-learning/.

Ramadhan, Luthfi. “Neural Network: The Dead Neuron.” *Medium*, Towards Data Science, 9 Jan. 2022, https://towardsdatascience.com/neural-network-the-dead-neuron-eaa92e575748.

RockTheStarRockTheStar 11.9k3131 gold badges6666 silver badges9090 bronze badges, et al. “What Are the Advantages of ReLU over Sigmoid Function in Deep Neural Networks?” *Cross Validated*, 1 Apr. 1962, https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks#comment825545\_126238.

Terra, John. “Regression vs. Classification in Machine Learning for Beginners: Simplilearn.” *Simplilearn.com*, Simplilearn, 15 Sept. 2022, https://www.simplilearn.com/regression-vs-classification-in-machine-learning-article.

Wang, Chi-Feng. “The Vanishing Gradient Problem.” *Medium*, Towards Data Science, 8 Jan. 2019, https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484.

**Appendix:**

CNN testing codes, combination RRRRRR:

from \_\_future\_\_ import print\_function

import tensorflow as tf

device\_name = tf.test.gpu\_device\_name()

if device\_name != '/device:GPU:0':

raise SystemError('GPU device not found')

print('Found GPU at: {}'.format(device\_name))

import numpy as np

#import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten

# load dataset and split it into train and test sets

from keras.datasets import cifar10

(train\_images, train\_labels), (test\_images, test\_labels) = cifar10.load\_data()

print('Training data shape: ', train\_images.shape, train\_labels.shape)

print('Testing data shape: ', test\_images.shape, test\_labels.shape)

# find unique numbers from train labels

classes = np.unique(train\_labels)

nClasses = len(classes)

print('Total number of outputs: ', nClasses)

print('Output classes: ', classes)

'''

plt.figure(figsize=[4,2])

# display the first image in training data

plt.subplot(121)

plt.imshow(train\_images[0,:,:], cmap='gray')

plt.title('Ground Truth: {}'.format(train\_labels[0]))

# display the first image in testing data

plt.subplot(122)

plt.imshow(test\_images[0,:,:], cmap='gray')

plt.title('Ground Truth: {}'.format(test\_labels[0]))

'''

# reshape data

nRows, nCols, nDims = train\_images.shape[1:]

train\_data = train\_images.reshape(train\_images.shape[0], nRows, nCols, nDims)

test\_data = test\_images.reshape(test\_images.shape[0], nRows, nCols, nDims)

input\_shape = (nRows, nCols, nDims)

# convert all data into float

train\_data = train\_data.astype('float32')

test\_data = test\_data.astype('float32')

# normalize data

train\_data /= 255

test\_data /=255

# convert labels into one-hot vector

train\_labels\_one\_hot = to\_categorical(train\_labels)

test\_labels\_one\_hot = to\_categorical(test\_labels)

# display the change for category label using one-hot encoding

print('Original label 0: ', train\_labels[0])

print('One-hot label 0: ', train\_labels\_one\_hot[0])

# model class

def createModel():

model = Sequential()

# the first two layers with 32 filters of window size 3x3

model.add(Conv2D(32, (3, 3), padding='same', activation='relu', input\_shape=input\_shape))

model.add(Conv2D(32, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(nClasses, activation='softmax'))

return model

# compile model

model1 = createModel()

batch\_size = 256

epochs = 100

model1.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])

model1.summary()

# train and evaluate the model

history = model1.fit(train\_data, train\_labels\_one\_hot, batch\_size=batch\_size, epochs=epochs, verbose=1, validation\_data=(test\_data, test\_labels\_one\_hot))

model1.evaluate(test\_data, test\_labels\_one\_hot)