Expertise
and insight
for the future

Chanh Nguyen

Diagnosis of Breast Cancer using Deep Learning

Metropolia University of Applied Sciences

Bachelor of Engineering

Name of the Degree Programme

Bachelor's Thesis

12 April 2019

Author	Chanh Nguyen
Title	Diagnosis of Breast Cancer using Deep Learning
Number of Pages	46 pages + 1 appendices
Date	12 April 2019
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Professional Major	Software Engineering
Instructors	Janne Salonen, Supervisor

This project covers the fundamentals knowledge of artificial intelligence in general and deep learning in particular, covers the vision of technologies developed in the future. Artificial intelligent becomes more necessary in human daily life, smartphone, laptop, computer, self-driven cars, smart houses, virtual assistant. On the other hands, they can be quite useful when added to the healthcare system and impacts, managing a large amount of clinical data, diagnosis of certain types of diseases. With the common goal is to make human life better, longer and more joyful.

Cancer is the cause of death of many people around the globe. One of the main reasons for cancer rate increases is simply because of human longevity, climate changes, overpopulation, environment impacts, alcohol consuming, people are getting more and more overweight and obese, changes in lifestyles raises the risk of developing a certain type of cancers. Women are detected positive with breast cancer at a younger age, one of the main reasons is women start to have fewer babies later, breastfeed less.

The motivation of this project is based on my curiosity about learning new technology in the computer field. Artificial intelligence has been and will be a powerful companion of computer scientists in the world. This project was working on the spring of 2019 as the thesis project for my bachelor's degree in engineering. The goal of this project is to bring the newest technologies to improve human healthcare and well-being. Apply deep learning into the healthcare system, help people can diagnose cancer more accurate and at sooner stages to increase the chance of surviving and enjoy the happiness after treatments. The outcome goal will build a convolutional neural network model to predict the presence of a cancer tumor in scanned specimens.

Keywords	breast cancer, invasive ductal carcinoma, deep learning, neural network, artificial intelligence

Contents

List of Abbreviations

1 Introduction 1

2 Overview of Deep learning 2

2.1 Deep Learning 2
2.2 Neural network 2
2.3 Activation functions 3
2.3.1 Sigmoid function 4
2.3.2 TanH function 4
2.3.3 ReLU function 5
2.3.4 Leaky ReLU function 5
2.4 Logistic regression 6
2.5 Backward propagation 7
2.6 Bias-variance tradeoff 7
2.7 Regularization 8
2.7.1 L2 regularization 9
2.7.2 Dropout regularization 9
2.8 Normalization 10
2.9 Extensions and variants 10
2.9.1 Mini-batch gradient descent 10
2.9.2 Momentum 11
2.9.3 RMSprop optimizer 12
2.9.4 Adoptive Moment Estimation (ADAM) 13
2.10 Learning rate decay 14
2.11 Hyperparameter optimization (HPO) 14
2.11.1 Babysitting model 14
2.11.2 Parallel multiple models training 15
2.12 Batch Normalization (BN) 15
2.13 Convolutional Neural Network (CNN) 16

3 Architectures 17

3.1 LeNet-5 17

3.2 AlexNet 18
3.3 GoogleNet 18
3.4 VGGNet (VGG16) 19
3.5 ResNet 19

4 Invasive Ductal Carcinoma (IDC) 20

4.1 Ductal carcinoma 20
4.2 Stages and survival rates 21

5 Implementation and evaluation 21

5.1 Data source 21
5.2 Implementation 22
5.2.1 MNIST CNN model 22
5.2.2 CIFAR10 CNN model 25
5.2.3 Results, evaluation and discussion 29

6 Conclusion 32

References 34

Appendices
Appendix 1. Table of figures

List of Abbreviations

ADAM	Adoptive Moment Estimation.
AI	Artificial Intelligence.
ANN	Artificial Neural Network.
BN	Batch Normalization.
ConvNet	Convolutional Neural Network.
CNN	Convolutional Neural Network.
CPU	Central Processing Unit.
DeconvNet	DeConvolutional Neural Network.
DNN	Deep Neural Network.
DL	Deep Learning.
FC	Fully Connected.
FC-CNN	Fully Convolutional Convolutional Neural Network.
GD	Gradient Descent.
GPU	Graphics Processing Unit.
HPO	Hyperparameter Optimization.
IDC	Invasive Ductal Carcinoma.
ILSVRC	ImageNet Large Scale Visual Recognition Challenge.
ML	Machine Learning.
MGD	Mini-batch Gradient Descent.

ReLU	Rectified Linear Unit.
ResNet	Residual Neural Network.
RNN	Recurrent Neural Network.
SGD	Stochastic Gradient Descent.
VPNN	Vector Product Neural Network.

1 Introduction

Artificial intelligent starts to become more and more powerful, reliable, perform better than human in many tasks. Nowadays, AI spreads fast and wide into many aspects of human life, help human in daily life, such as virtual assistants, self-driven cars, drones, email, YouTube, online advertisements, maps and navigation, smart devices, web searches … And AI is developed with a faster and faster speed every day, become more intelligent and apply into different fields, with the mission to makes human life better. In the healthcare system, AI is applied for managing and recording a mass amount of medical data, making healthcare system analysis, developing pharmaceuticals, helping doctors to make decisions. The machine is better than human at doing repetitive jobs over and over again faster and with stable performance. The machine can also pay more attention in even the smallest detail, a single pixel in an image. By applying these advantages, deep learning was born to solved problems relative to image classification, natural language processing.

Cancer is a group of diseases developing abnormal cell in living creatures' body. These cells grow into tumor and have a potential to invade and spread throughout the whole body, changing the normal functions of cells and organs, could shut down the activities of parts of the body and lead to death. In the past, cancer likes a death sentence to any unlucky person who has it. In the present days, health care has become much better with a variety of new technologies application, better medicines, better diagnosis. People who have cancer can live for a long life with the assistance of medicine and technologies. Furthermore, the stage of when cancer is detected is crucial, the sooner the better, and patients can have a better chance of complete treatment and free of cancer. So, to conclude, we need someone or something to diagnose cancer for a vast amount of medical data to find people with high chances of having cancer, then justify the accuracy and starts the treatment if necessary. By applied deep learning, we could solve this problem, repeatedly analysis a large amount of data with the same accuracy.

The goal of this project is to train a deep learning convolutional neural network to analysis and diagnosis of breast cancer, invasive ductal carcinoma. The program can take input is an image size 50x50 of scanned specimens, and provides an output is that the image received has cancer or not with acceptable accuracy. On the other hands, this thesis also provides principal backgrounds of machine learning in general or deep learning in particular and basic knowledge of invasive ductal carcinoma.

The thesis is divided into 5 sections. The first section is the general background knowledge of deep learning, second is some of the most popular neural network models in the past few years. The third section is the basic medical knowledge for invasive ductal carcinoma, the symptom, treatments, and survival expectation by stages. The fourth section is my implementation, develop the model and evaluate results. The final section is the conclusion, where the project goal is a reevaluation.

2 Overview of Deep learning

2.1 Deep Learning

Deep learning is a subset of machine learning in the artificial intelligence field. Deep learning is developed used neural network architectures, most capable of learning unsupervised from unstructured and labeled data. Deep learning network is inspired by the human brain neural network, takes data input and provided output go through many neuron nodes. [39]

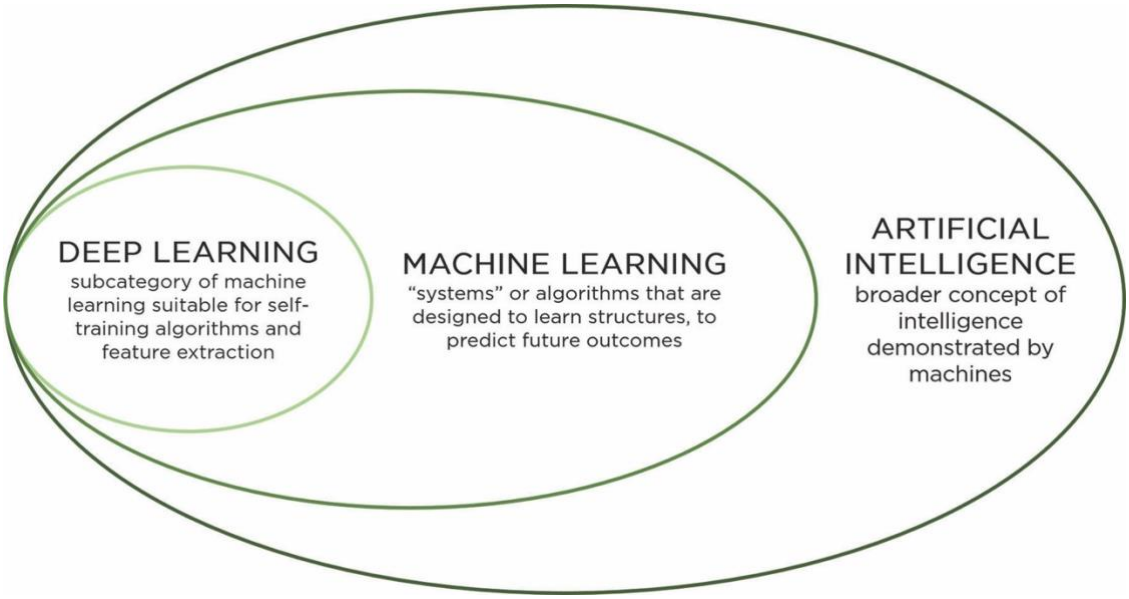


Figure 1. The relation between deep learning, machine learning, and artificial intelligence. Image copied from [40].

2.2 Neural network

A neural network is an interconnected network of neurons or nodes, inspired by a simplification of brain neurons, create to solve AI problems. Each node connects with a network model by weights. A positive weight means an excitatory connection, opposite to a negative weight, which throws backs an inhibitory connection. [4] The components of a standard neural network are layers, input data, targets, loss function, optimizer or learning rate. [3, 58-59]

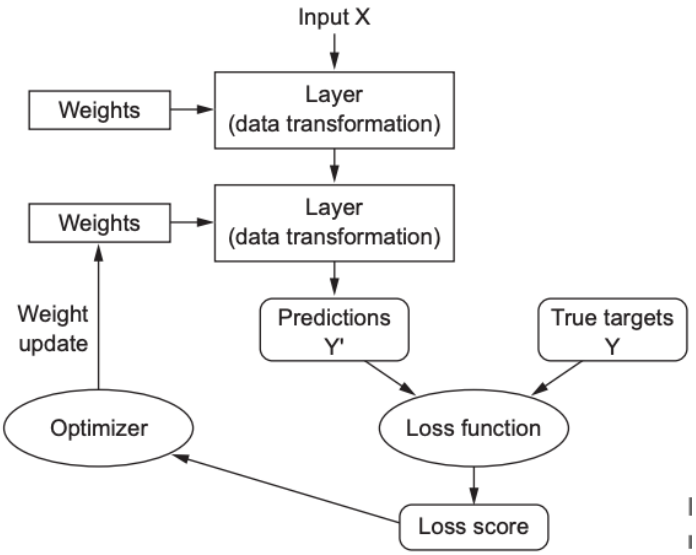


Figure 2. The relationship between components of a neural network. Image copied from [3].

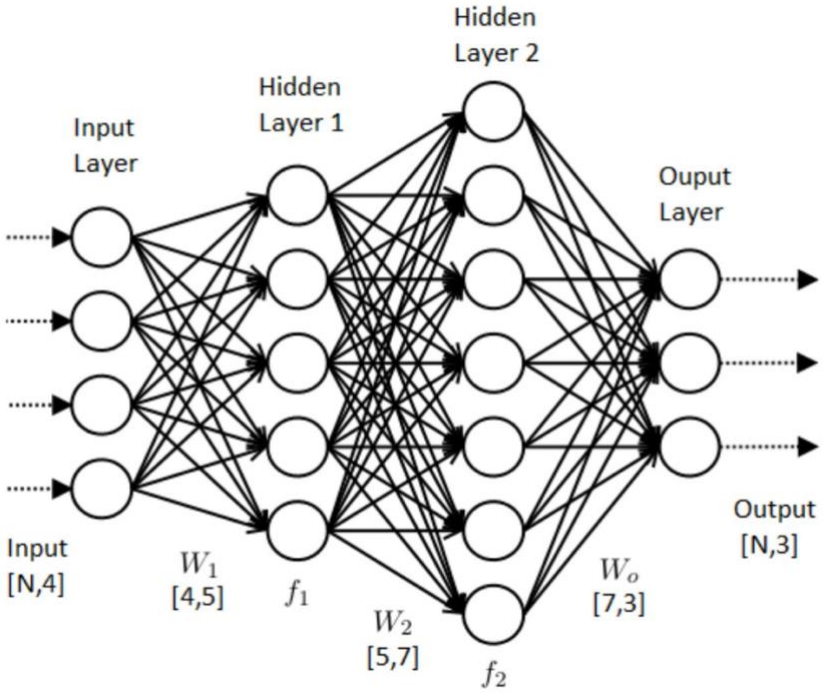


Figure 3. A simple neural network with two hidden layers. Image copied from [6].

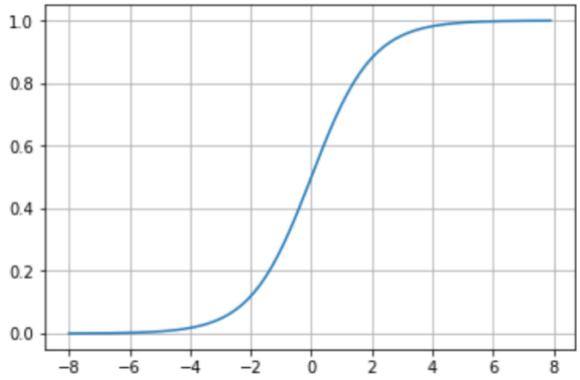
2.3 Activation functions

A neural network model is required to have a non-linear activation functions. If linear activation functions are used, then the neural network will output a linear function of the input. In deep learning, a model network may have many layers, if non-linear activation functions are not used, then the neural network is just computing a linear function for

the output, no matter how many layers the neural network has. [8, 7-17] If a model network has linear activation functions for its hidden layers and a sigmoid function for the output layer, then this neural network model is no more expensive than a logistic regression with no hidden layers. The output will be used for supervised learning, in range (0, 1), so use a sigmoid function for the output layer is sufficient. The hidden layers could use other functions, such as TanH or ReLU. [8]

2.3.1 Sigmoid function

Sigmoid function or logistic function is a monotonic function and has a bell-shaped derivative. This function takes an input and produces an output in range (0, 1).

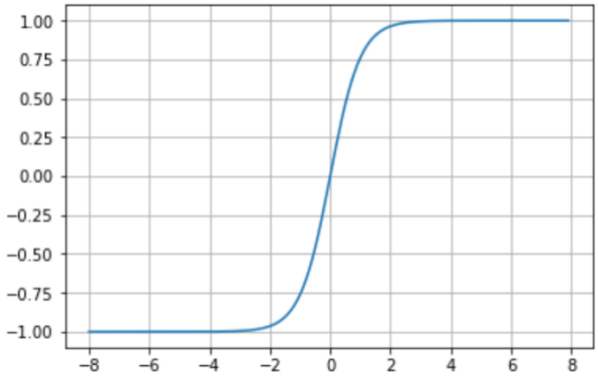


$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

$$f'(x) = f(x)(1 - f(x))$$

2.3.2 TanH function

TanH function is a monotonic hyperbolic function. This function is a rescaled version of the Sigmoid function and takes an input and produces an output in range (-1, 1)

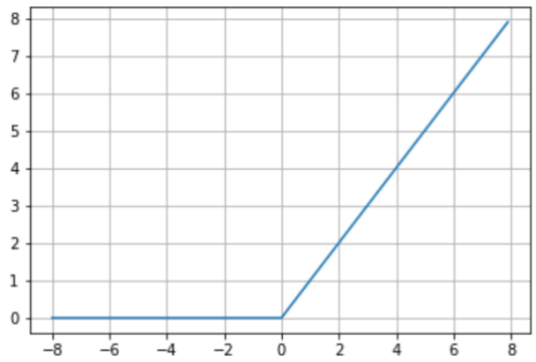


$$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

$$f'(x) = 1 - f(x)^2$$

2.3.3 ReLU function

ReLU or Rectified linear unit function is a monotonic function and also derivative monotonic takes any input and produces a non-negative output.

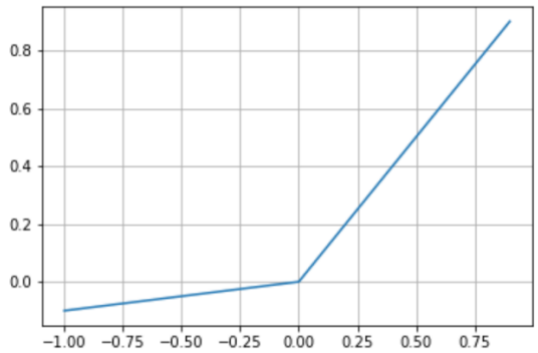


$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

2.3.4 Leaky ReLU function

Leaky ReLU or Leaky rectified linear unit function is a monotonic function and also derivative monotonic takes an input and produces an output in range R.



$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

$$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

2.4 Logistic regression

Logistic regression is a learning algorithm that is used for supervised learning or binary classification problems. With the classification problems, the output label is either zero or one (true or false). Given an input feature vector X, for example, an image that needed to clarify that it is either a cat or non-cat image. Logistic regression model can be used for the prediction Y, the true result of the image. [1] Logistic regression model can have the input of one or many variables, among a response variable and one or many analytical variables. Logistic regression is built by Neural Network mindset, and it is a very simple neural network with only one layer.

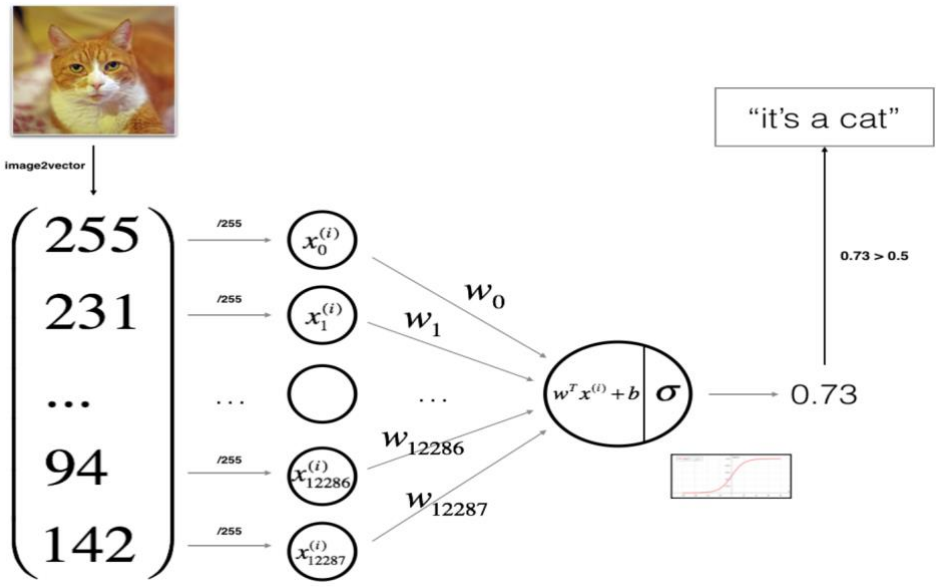


Figure 4. The general architecture of learning algorithm. Image copied from [2].

Equations for one node:

$$z^{(i)} = w^T x^{(i)} + b$$
$$\hat{y}^{(i)} = a^{(i)} = sigmoid(z^{(i)})$$
$$\mathcal{L}(a^{(i)}, y^{(i)}) = -y^{(i)} \log(a^{(i)}) - (1 - y^{(i)}) \log(1 - a^{(i)})$$

The cost function is computed for all training examples:

$$J = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(a^{(i)}, y^{(i)})$$

2.5 Backward propagation

Backward propagation or backpropagation is often the most mathematical challenging part in deep learning. This step is the study part when the parameters modify themselves to fit more the dataset and improve better in the cost function. The formulas of this step are as the figure below. Nowadays, many frameworks and libraries are available for deep learning developers. These frameworks often provide automate calculation for backward propagation.

$dz^{[2]} = a^{[2]} - y$	$dZ^{[2]} = A^{[2]} - Y$
$dW^{[2]} = dz^{[2]}a^{[1]^T}$	$dW^{[2]} = \frac{1}{m} dZ^{[2]}A^{[1]^T}$
$db^{[2]} = dz^{[2]}$	$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$
$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]\prime}(z^{[1]})$	$dZ^{[1]} = W^{[2]T}dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$
$dW^{[1]} = dz^{[1]}x^T$	$dW^{[1]} = \frac{1}{m} dZ^{[1]}X^T$
$db^{[1]} = dz^{[1]}$	$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$

Figure 5. Summary of gradient descent. Source [8].

2.6 Bias-variance tradeoff

Bias is the difference between the error of the neural network model compares with Bayes error. A high bias model is usually simple and underfitting data, it pays very few attentions to the training and test data. [12]

On the other hand, a high variance model used a complex classifier, for example, a deep neural network with many layers, or neural network with many hidden units, the data can be fitted perfectly. This model pays a lot of attention to the training data but has poor performance with the data which it has not seen before. That neural network is overfitting the training data and it is not a great fit either. [12]

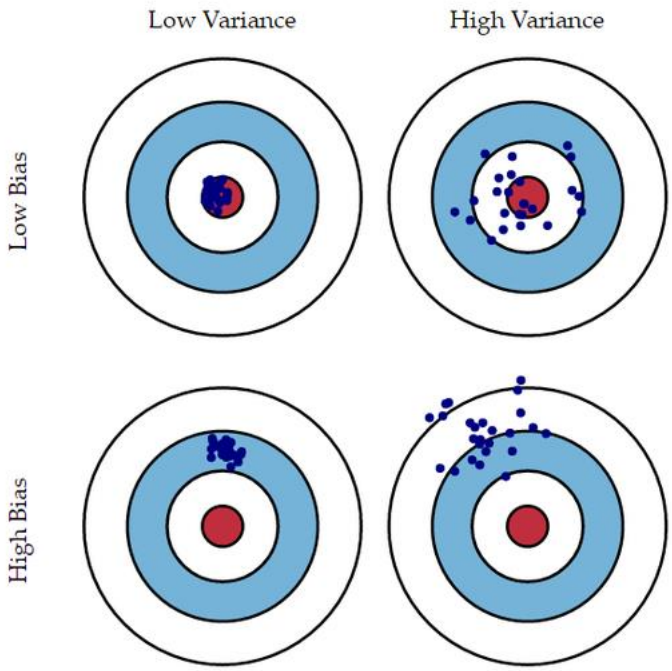


Figure 6. Bias and Variance Trade-off. Image copied from [13].

In the figure above, the red circle is the target which predicts accurately the correct values. The figure displays four different scenarios representing combinations of both high and low bias and variance. In the top left, the model has low bias and variance, most of the predictions are inside the center red circle. This is the model that the neural network should aim to. On the top right, the model has low bias and high variance, the dots are separated from each other's, but the center of the whole group is still in the center. On the bottom left, all the dots are a shank, but the center is laying down somewhere far away from the red circle center. On the bottom right, this is the worst model of all, the dots are separated, and the center is off.

2.7 Regularization

In deep learning, regularization is the process in which the developers regulate models in other to prevent overfitting or solve ill-posted problems. Prevent overfitting data is one of the major aspects of training in deep learning. [14] A symptom of an overfitting model is that it does great in the training set, has good accuracy, but performs badly in the test set, high variance. One of the first things the developer team could try is regularization technique.

2.7.1 L2 regularization

$$J = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right) + (1-y^{(i)})\log\left(1-a^{[L](i)}\right)\right)$$

$$J_{regularized} = \underbrace{-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right) + (1-y^{(i)})\log\left(1-a^{[L](i)}\right)\right)}_{\text{cross-entropy cost}} + \underbrace{\frac{1}{m}\frac{\lambda}{2}\sum_{l}\sum_{k}\sum_{j}W_{k,j}^{[l]2}}_{\text{L2 regularization cost}}$$

In L2 regularization, the regularization cost is the sum of the square of all the weight Wkj as solved in the formula. L2 regularization balances the weights makes the weight of each node smaller when they are large and large when they are really small. [15]

2.7.2 Dropout regularization

Dropout in deep learning is a technique that the developers set some probability to random shut down some neurons in each iteration. The probability is between zero and one, and any node can be eliminated at any given time.

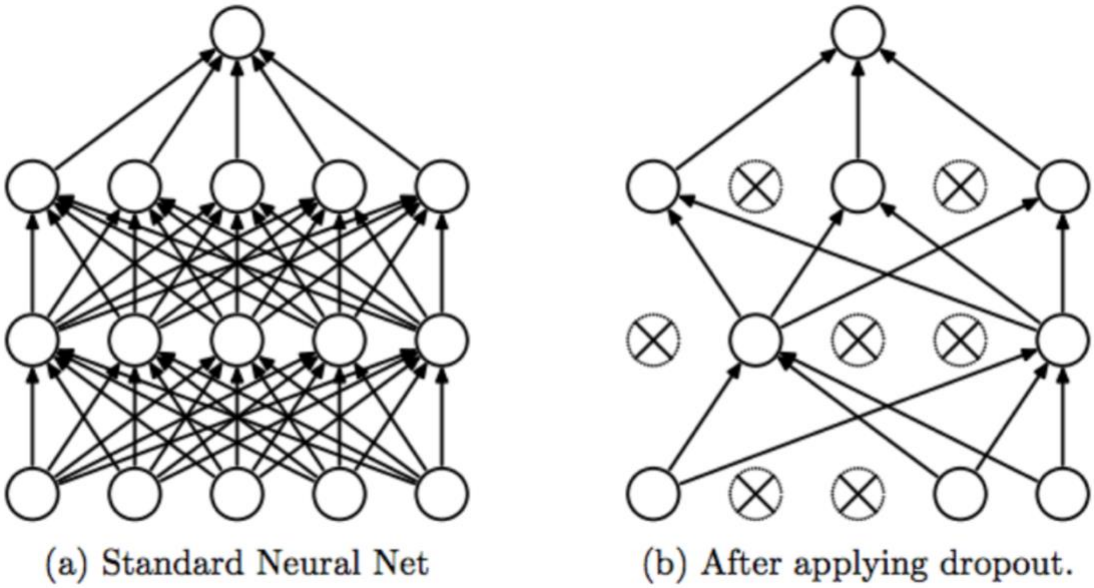


Figure 7. Dropout regularization. Image copied from [17].

With dropout, after each iteration, a new model is trained with only uses a subset of the original network. Neurons less sensitive to the activation of one other specific neuron, because that other neuron could be shut down anytime during an iteration.

2.8 Normalization

Normalization is a technique that could help increase speed when training a neural network.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 8. Equations for batch normalization. Image copied from [30].

2.9 Extensions and variants

2.9.1 Mini-batch gradient descent

Mini-batch gradient descent is one of the most common implementations of gradient descent in deep learning. This is an algorithm that divides the training set of data into smaller batches. These batches are then used to evaluate model error and update model coefficients. Developers might choose to implement the batch gradient descent over mini-batch gradient descent if the training dataset is not too big. [19]

The pros of mini-batch gradient descent are that the model update more frequently, which eventually increase convergence and avoid local minima. Each iteration step of update requires less computational power which allows this method has more efficient in memory and algorithm implementations. On the other side, developers need to configure an additional hyperparameter, the size of the mini-batch, for the learning method. The cost function needs to be accumulated across all mini-batch, similar to the normal batch gradient descent. [19]

Stochastic gradient descent (SGD) is mini-batch gradient descent when the size is set to one. With this implementation, the model uses only one training example before calculating the cost and updating the gradients. SGD could run really fast, but the

parameter might oscillate around the global optimal point rather than converge smoothly.

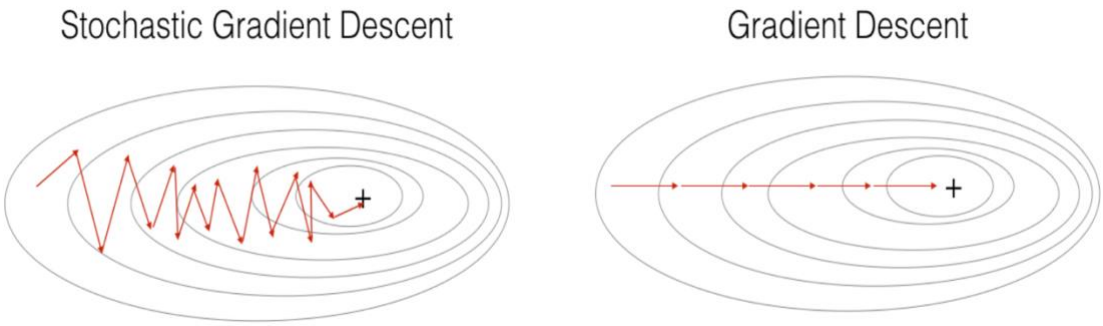


Figure 9. Stochastic gradient descent vs Gradient descent. Image copied from [21].

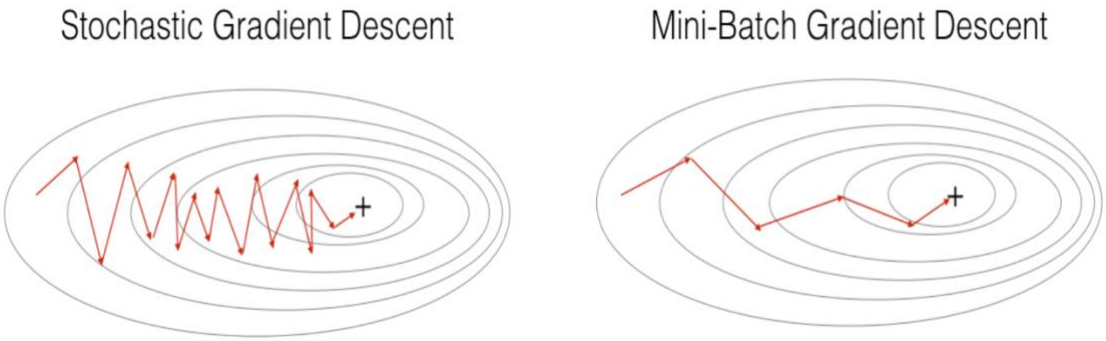


Figure 10. Stochastic gradient descent vs Mini-batch gradient descent. Image copied from [21].

In these figures above, "+" sign indicates the minimum cost. SGD seems to be quite noisy and has many oscillations before reach convergence. But each step is much faster than normal gradient descent. Typically, the get the best out of both, mini-batch gradient descent (MGD) has both the speed of SGD and the stability of GD.

2.9.2 Momentum

Momentum is a popular technique used together with mini-batch gradient descent to reduces the oscillations and converge faster to the optimal point. Momentum records the past gradients and stores the direction in a variable v. These gradients are later on used to smooth out the update by using an exponentially weighted average.

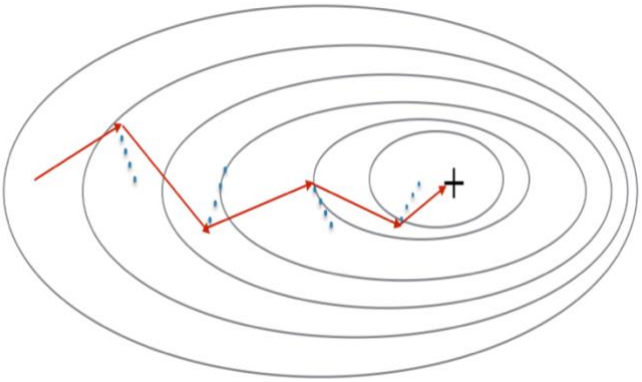


Figure 11. The blue dots are the direction of the current gradient. The red arrows show the direction after taking momentum, make them move toward the optimal point faster and more stable. Images copied from [21].

$$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw$$

$$v_{db} = \beta \cdot v_{dw} + (1 - \beta) \cdot db$$

$$W = W - \alpha \cdot v_{dw}$$

$$b = b - \alpha \cdot v_{db}$$

Figure 12. The equations for gradient descent with momentum. Source [23].

2.9.3 RMSprop optimizer

RMSprop optimizer is an optimization implementation for the neural network. This method limits the vibrations in the vertical direction. Therefore, the learning rate could take considerable steps and converge faster to the optimal point, where the cost is minimum.

$$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw^2$$

$$v_{db} = \beta \cdot v_{dw} + (1 - \beta) \cdot db^2$$

$$W = W - \alpha \cdot \frac{dw}{\sqrt{v_{dw}} + \epsilon}$$

$$b = b - \alpha \cdot \frac{db}{\sqrt{v_{db}} + \epsilon}$$

Figure 13. The equations for RMSprop optimizer. Image copied from [23].

2.9.4 Adoptive Moment Estimation (ADAM)

Adam is currently one of the most effective optimization algorithms for neural network training. It is the combination of gradient descent with momentum and RMSprop optimizer.

$$\begin{cases} v_{dW^{[l]}} = \beta_1 v_{dW^{[l]}} + (1 - \beta_1) \frac{\partial \mathcal{J}}{\partial W^{[l]}} \\ v_{dW^{[l]}}^{corrected} = \frac{v_{dW^{[l]}}}{1 - (\beta_1)^t} \\ s_{dW^{[l]}} = \beta_2 s_{dW^{[l]}} + (1 - \beta_2)(\frac{\partial \mathcal{J}}{\partial W^{[l]}})^2 \\ s_{dW^{[l]}}^{corrected} = \frac{s_{dW^{[l]}}}{1 - (\beta_1)^t} \\ W^{[l]} = W^{[l]} - \alpha \frac{v_{dW^{[l]}}^{corrected}}{\sqrt{s_{dW^{[l]}}^{corrected}} + \varepsilon} \end{cases}$$

Where:

- t: the number of steps taken of Adam.
- L: the number of layers.
- β1 and β2: hyperparameters for exponentially weighted averages.
- α: the learning rate.
- ε: a very small number, used to avoid divide by zero.

Similar to momentum, Adam calculates the exponential weigh an average of past gradients and store them into variable v, then calculate v-corrected with bias correction. Then similar with RMSprop, it evaluates the exponential weight average of the squares

of pass gradients, and store them into variable s, then s-corrected with bias correction. Finally, the function updates coefficients with the calculated v and s correction.

2.10 Learning rate decay

Learning rate decay is a method in which the learning rate will be slowly reduced over time. During the initial phases, when the model has a large learning rate, it can still have fast learned over each iteration to reach the optimal point much faster. When the model approaches the minimum cost, learning rate gets decay allowing the model to takes smaller steps and eventually end up in the tighter region around the optimal minimum cost point.

$$\alpha = \frac{1}{1 + decayRate * epochNum} * \alpha 0$$

The formula for learning rate alpha:

- α0: initial learning rate.
- decay rate: the rate of decay for learning rate, usually use 1.
- epoch num: the number of iterations passes through all the training data.

2.11 Hyperparameter optimization (HPO)

Hyperparameters are the essential parts in every neural network model. These parameters play important roles in the mission to increase the performance of the model. One of the most basic tasks when developing deep learning is to automatically set these hyperparameters to the optimize performance, especially the modern deep learning network depends on a wide range of parameters choices for the architectures, regularization, and optimization. [24, 3-4] One of the well-known methods for hyperparameter optimization is Bayesian. Bayesian optimization is a framework for the global optimization using black-box functions which builds mapping function to evaluate the values of the hyperparameter on the validation sets. [25] This method receives great results for tuning deep neural network model for image classification, natural language model, speech recognition, and proves a wide capability to different problems settings. [24, 9-11]

2.11.1 Babysitting model

Babysitting model, or Pandas strategy, is another tuning method for parameters when the developer team does not have a large available computational power to train multiple models at once. They patiently assess the achievement of the model over time and attuning the hyperparameters of the model based on previous days. [21] This

approach requires lots of time and effort, used for huge neural network model that takes a long time to progress a training period.

2.11.2 Parallel multiple models training

Parallel multiple models training or Caviar strategy, is the method in which the developer team trains many models at once in parallel, but with slightly different hyperparameters settings. Eventually, these models will generate different results and learning curves. The curve with the best result will be picked and saved all the hyperparameters data. Then the team might continue the progress with different hyperparameters and choose the best one again. [21] This approach is much faster than the babysitting model, but requires heavily computational power, or uses for a simple model that does not take a long time to progress a training period.

2.12 Batch Normalization (BN)

Batch normalization is a technique to reduce the covariate between internal nodes, prevents small changes to amplifying the parameters and allows larger learning rate, adjust the speed of the model. [27]

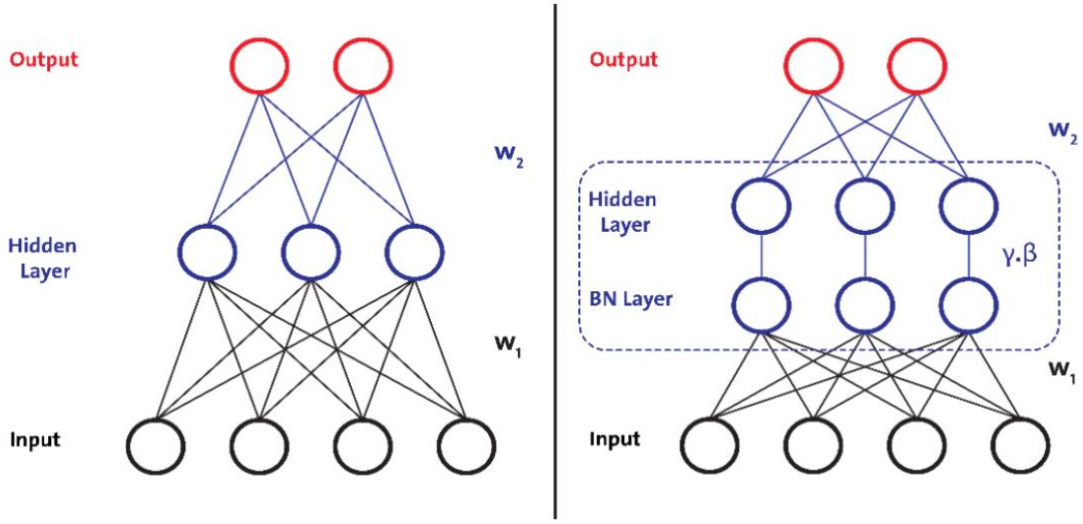


Figure 14. Batch normalization. Images copied from [27].

Batch normalization places an additional layer for each hidden layer. BN normalize the data, makes them centralized, and equalized the variances before using them as an input for the next layer. In the figure above, BN activates before any processes of this layer network apply and can take part in the forward and backward propagation. [27] The advantages of batch normalization are that it reduces the problem of input values changing, and even has a slight regularization effect. Batch norm can be learned to

apply Adam, Gradient descent with momentum, or RMSprop optimizer, not just with gradient descent.

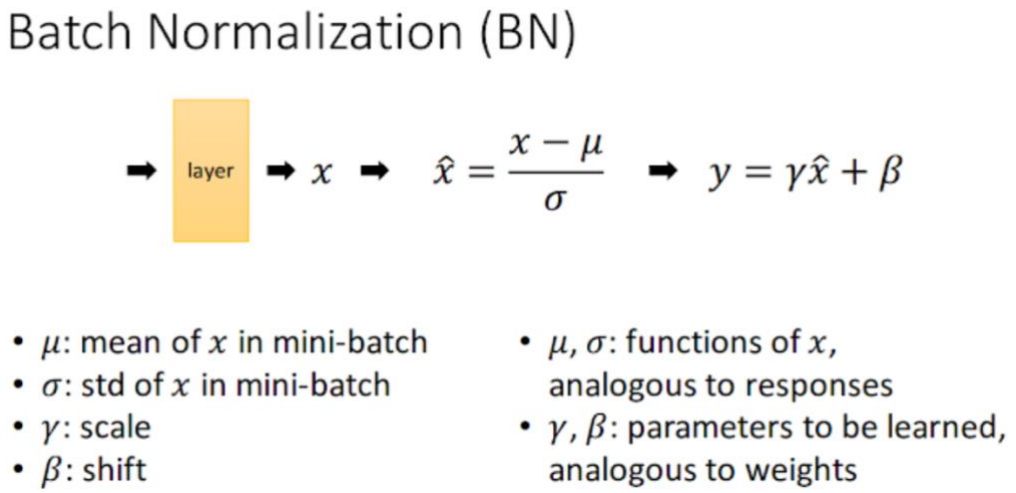


Figure 15. The equations of Batch Normalization. Images copied from [28].

When using batch normalization, during test time, to evaluate the neural network on new samples, an additional normalization step much to be taken into account. Using μ and σ2 estimated with an exponentially weighted average across mini-batches during training to normalize the test data.

2.13 Convolutional Neural Network (CNN)

A Convolutional Neural Network is an algorithm in which some layers of neural network contain filters. These filters are used mostly for image edges detection and reduce a large number of parameters in this model. For modern images, the size of the images could be really huge for normal network model to works on, CNN adds filter layers to reduces the definition of images, in support for the later layers. [30]

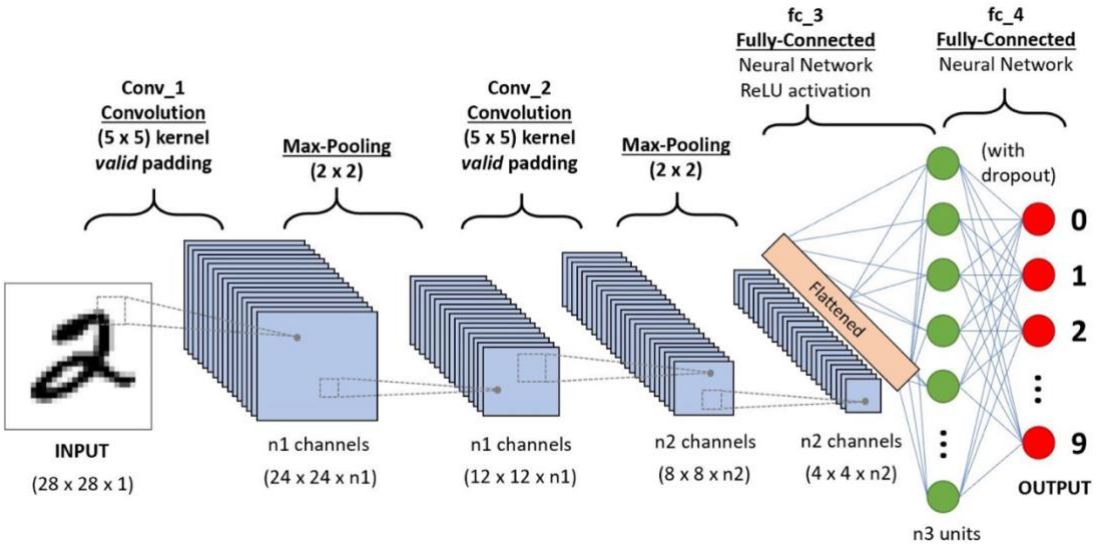


Figure 16. CNN sequence with two convolution layers. An example of a simple model of CNN, there are two convolution layers and one fully connected layer in the end to justify the outputs. Images copied from [30].

3 Architectures

3.1 LeNet-5

LeNet-5 is one of the pioneering convolutional neural networks, developed by LeCun in 1998. This network is very simple with only seven layers and takes the input of 32x32 pixels no color images. The model is applied by several banks to justify the hand-written letters and numbers on cheques. [33]

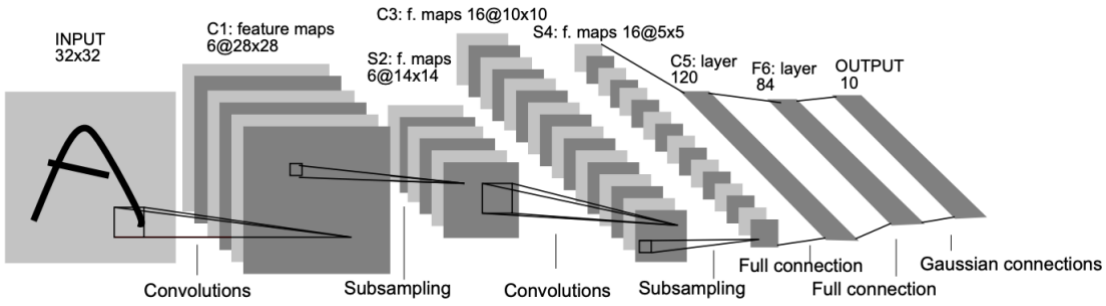


Figure 17. The architecture of LeNet-5. Each layer contains a feature map of filters handling the input image. Image copied from [33].

3.2 AlexNet

First appears in 2012, AlexNet performance is outstanding, superior to most of the competitors and won first place in a challenge by reduces the top 5-error from 26.2% to 15.2%. The network is inspired by LeNet of Yann LeCun but has more layers and more filters for each layer. AlexNet contains eight layers with the first five are convolutional layers and the remainders are fully connected layers. The architecture includes filters of 11x11, 5x5, 3x3, drop out, ReLU activation function and max-pooling layers. The output of AlexNet is a SoftMax activation function with 1000 outcomes. [31]

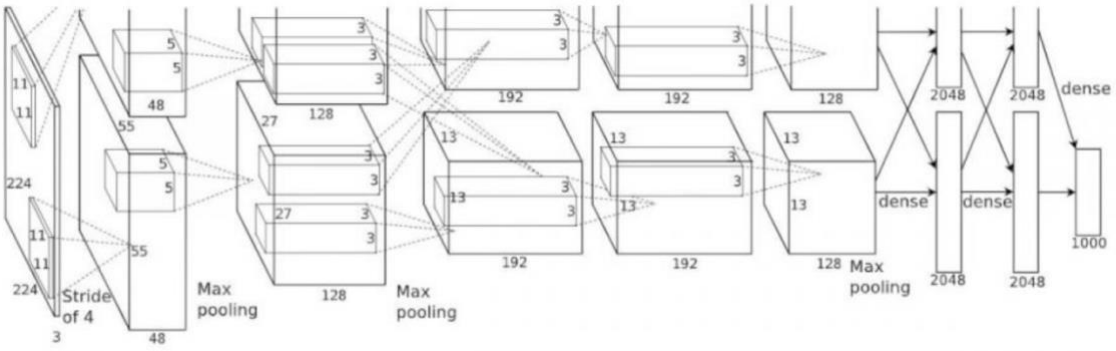


Figure 18. The architecture of AlexNet. Images copied from [32].

3.3 GoogleNet

GoogleNet or Inception network is developed by Google and achieved the highest prize in the ImageNet Large Scale Visual Recognition Competition 2014. This neural network has a percentage of 6.67% for the top-5 error test set. This is really high and very close to the human-level performance. This network is an upgrade from LeNet and AlexNet, with the same basic structures but with more layers, better filters, and implements a new technique which is called inception module. The model also uses other normalization techniques such as batch normalization, RMSProp. The model is very deep with 22 convolutional layers, but Google developers reduce to the number of parameters from 60 million of AlexNet to only 4 million. [44]

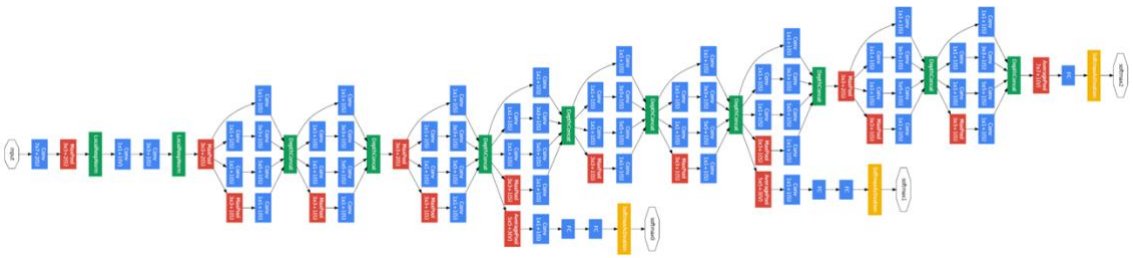


Figure 19. The architecture of GooleNet. Image copied from [44].

3.4 VGGNet (VGG16)

VGG16 is a convolutional neural network developed by K. Simonyan and A. Zisserman from University of Oxford won the runner-up at the ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2014. This model passes the test for top-5 error ImageNet with the accuracy of 92.7%. Implemented based on the model of AlexNet with only 3x3 convolution layers, VGG16 makes the improvement by replacing bigger kernel filters. VGG16 could take a larger image and reduces the dimensions after each convolution layers. It is one of the most popular models for extracting small features from images. One drawback of VGG16 is that it required over 100 million parameters, quite heavily to handle for normal CPU. [42]

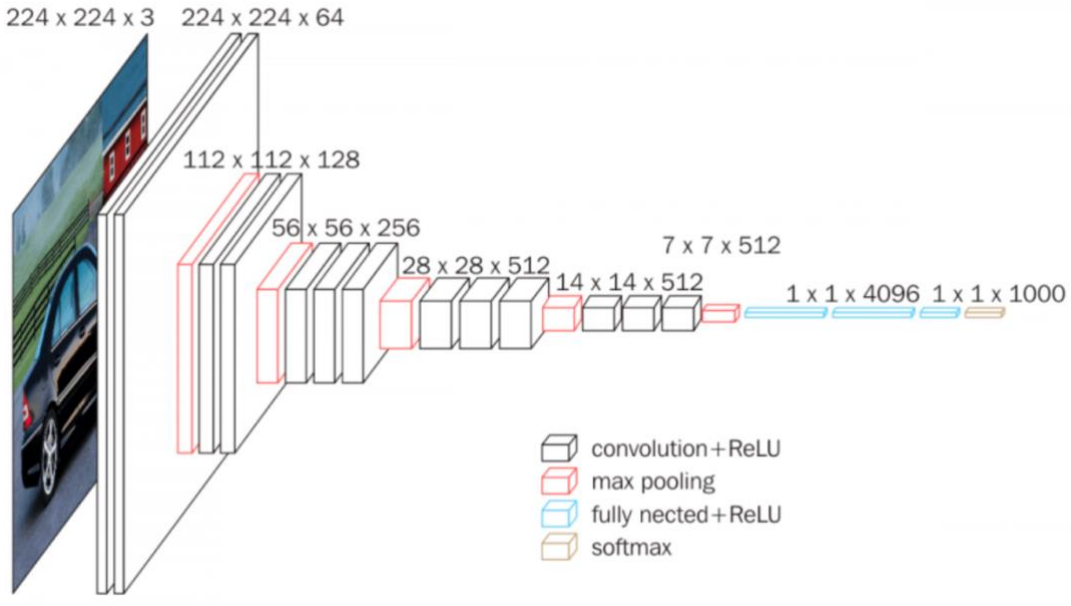


Figure 20. VGG16 Architecture. Image copied from [42].

3.5 ResNet

Residual Neural Network (ResNet) is developed by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun at the ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2015. This network is built out of residual blocks which allow users to train very deep networks, more than 100 layers. The team inventors introduce a new architecture called “skip connections”, so the output of the far previous layer could affect the output of current layers. Due to this technique, the modern neural network could have much more layers and still maintains a low complexity. The network achieves rate 3.6% in top 5-error and goes beyond human-level performance on a specific dataset. [32]

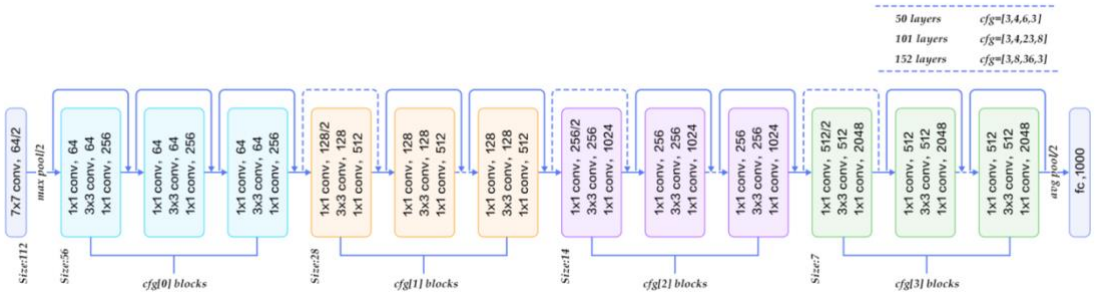


Figure 21. The architecture of ResNet. Image copied from [32].

4 Invasive Ductal Carcinoma (IDC)

4.1 Ductal carcinoma

Ductal carcinoma is one of the most general types of breast cancer. More than 50% of breast cancer patients is related to Ductal carcinoma. There are two types of DC: Invasive ductal carcinoma (IDC) and Ductal carcinoma in situ (DCIS). IDC takes responsible for more than 50% of invasive breast cancer patients [36]. IDC begins in the cells of milk ducts, the pipes that carry milk from lobules to the nipple and starts to invade or grows to the surrounding breast tissues, and it can spread to other parts of the patient body. [34]

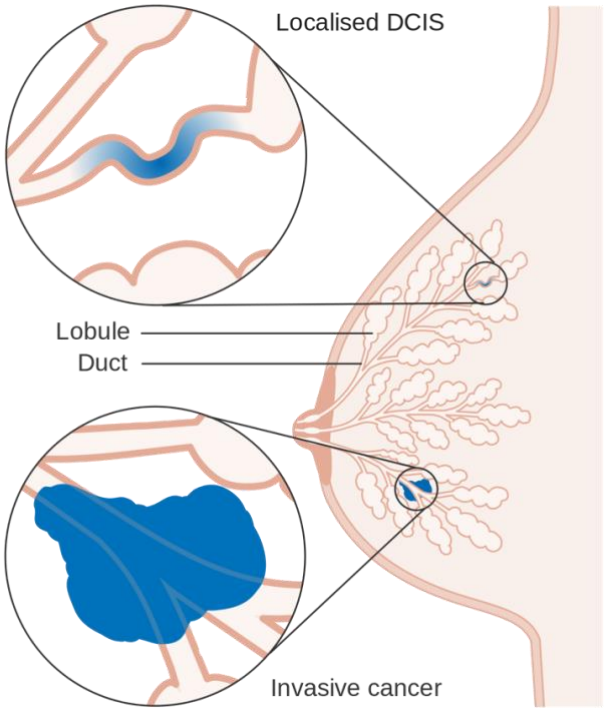


Figure 22. Ductal carcinoma. Image copied from [35].

4.2 Stages and survival rates

Based on the Medical News Cancer Society, 89 percent of the patients live longer than 5 years after diagnosed with breast cancer. [38]

- Stage 0-1: The tumor is smaller than two centimeters in diameter and the cancer is limited inside breasts. The survival rate is close to 100 percent, with approximately more than 60 percent of patients are diagnosed at this stage.
- Stage 2: The tumor is between 2 to 4 centimeters in diameter and cancerous cells have spread to the lymph nodes. The survival rate is around 93 percent.
- Stage 3: Cancer develops and starts to spread, more extensive cancer tumor is found, surrounding tissues and lymph nodes in the underarm area, but the cancer is confined to the breast. The survival rate is around 72 percent. Many women are treated successfully at this stage.
- Stage 4: The cancer tumor has metastasized to lymph nodes beyond the underarm area, could expand to other organs, such as lungs, liver, bones or brain. The survival percentage is around 22 percent. [38]

5 Implementation and evaluation

5.1 Data source

The dataset contains 162 whole-mount slide images of breast cancer scanned specimens. From this, about 277524 patches of size 50 x 50 are exacted, with around 30% are positive and 70% negative. The data was collected from 279 patients and each file name with the standard format for example 16085_idx5_x251_y851_class1. This file is collected from a patient with ID 16085, the patch was cropped from the coordination (x, y) is (251, 851), and class 1 means positive with IDC, and 0 is negative. This data source is from [41]. The data description is as below:

```
Number of examples: 277524
Number of negative IDC samples: 198738
Number of positive IDC samples: 78786
Percentage of IDC(+): 28.39%
Percentage of IDC(-): 71.61%
Sample shape (Width, Height, Channels): (50, 50, 3)
```

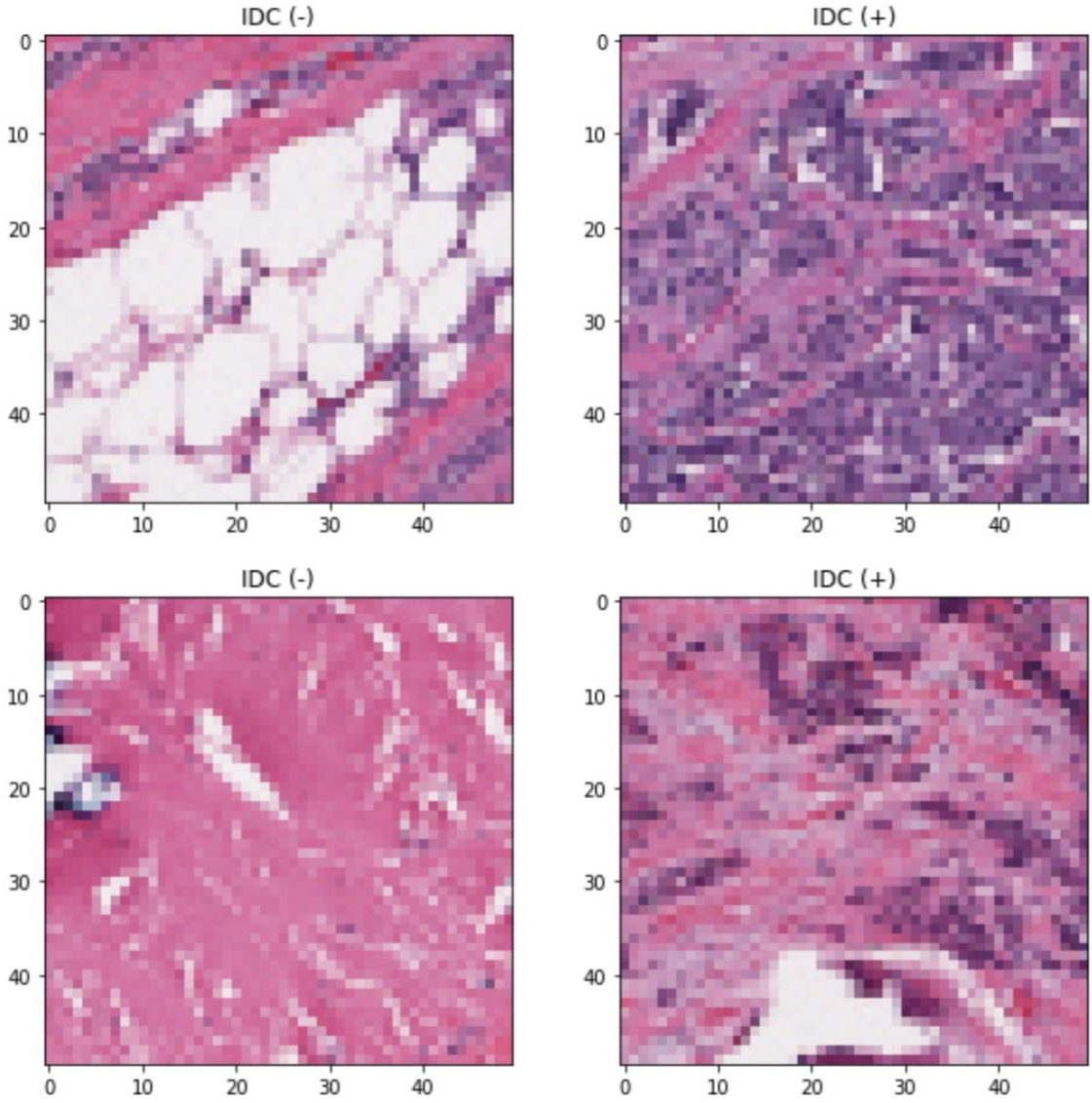


Figure 23. Samples with IDC (-) and IDC (+).

5.2 Implementation

5.2.1 MNIST CNN model

```
from keras.layers import Dense as KerasDens, Dropout as KerasDrop, Activation
as KerasActi, Flatten as KerasFlat, Conv2D as KerasConv, MaxPooling2D as
KerasMaxP

def IDCModel_mnist_cnn(input_shape):
    """
    IDCModel MNIST CNN Implementation.
```

```
    Arguments:
    input_shape -- dataset images shape

    Returns:
    model -- an instance model of framework Keras
    """

    # Initial input using tensor
    X_input = Input(input_shape)

    # Model implementation.
    X = KerasConv(32, kernel_size=(3, 3), padding='same',
                activation='relu', input_shape=input_shape)(X_input)
    X = KerasConv(64, (3, 3), activation='relu')(X)
    X = KerasMaxP(pool_size=(2, 2))(X)
    X = KerasDrop(0.25)(X)
    X = KerasFlat()(X)
    X = KerasDens(128, activation='relu')(X)
    X = KerasDrop(0.5)(X)
    X = KerasDens(1, activation='sigmoid', name='fc')(X)

    # Create Keras model instance.
    model = Model(inputs = X_input, outputs = X, name='IDCModel')

    return model
```

Listing 1. Convolutional neural network MNIST model to predict the presence of IDC on images.

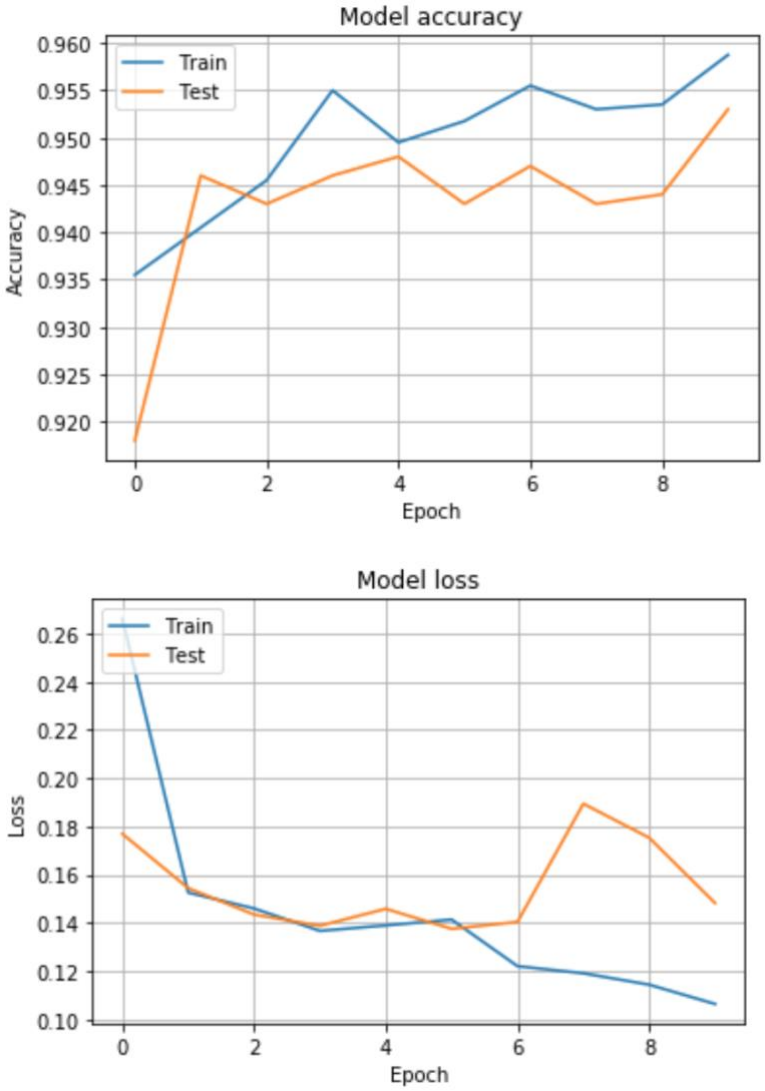


Figure 24. The prediction accuracy and model loss of MNIST CNN.

The model is trained with 10 epochs and batch_size is 64. The training time is around 30 minutes, the trained model performs good with both the training and test set, with the accuracy of the training set is 95.8% and the accuracy of the test set is 95.3%. The loss of the training set is 0.11 and for the test set is 0.15. It is not much difference between the performance of the model on the training and test set. A next step to improve this model is to expand more layers and train with a longer time to fix avoidable bias, extends the accuracy of the model. The model is stretched as the figure below.

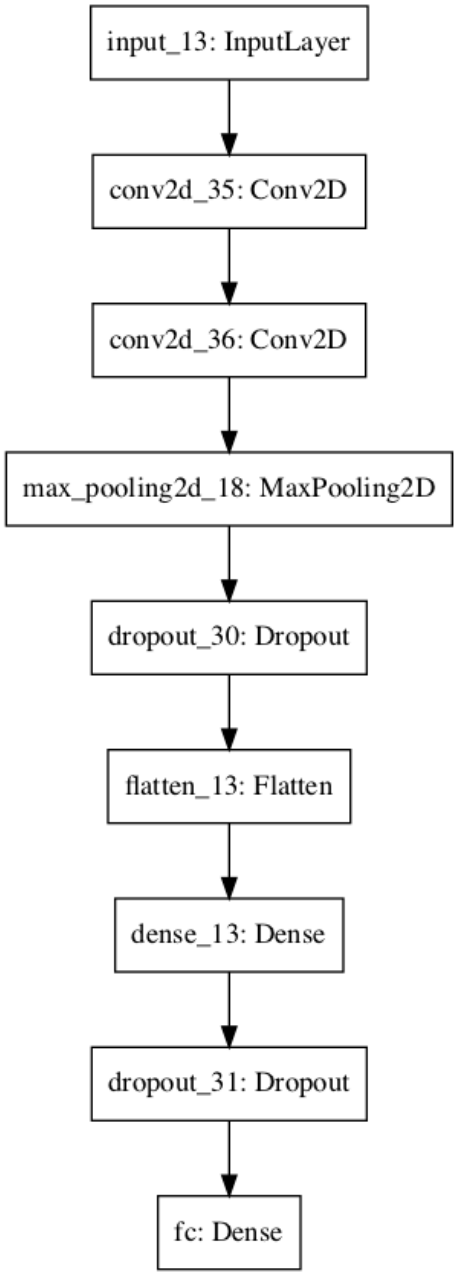


Figure 25. Outline of the IDC prediction CNN MNIST model.

5.2.2 CIFAR10 CNN model

```
from keras.layers import Dense as KerasDens, Dropout as KerasDrop, Activation
as KerasActi, Flatten as KerasFlat, Conv2D as KerasConv, MaxPooling2D as
KerasMaxP

def IDCModel_cifar10_cnn(input_shape):
    """
    The IDCModel based on CIFAR10 CNN implementation.

    Arguments:
    input_shape - dataset images shape
```

```
    Returns:
    model -- an instance model of framework Keras
    """

    # Initial input using tensor
    X_input = Input(input_shape)

    # Model implementation.
    X = KerasConv(32, kernel_size=(3, 3), padding='same',
input_shape=input_shape)(X_input)
    X = KerasActi('relu')(X)
    X = KerasConv(64, (3, 3))(X)
    X = KerasActi('relu')(X)
    X = KerasMaxP(pool_size=(2, 2))(X)
    X = KerasDrop(0.25)(X)

    X = KerasConv(64, (3, 3), padding='same')(X)
    X = KerasActi('relu')(X)
    X = KerasConv(64, (3, 3))(X)
    X = KerasActi('relu')(X)
    X = KerasMaxP(pool_size=(2, 2))(X)
    X = KerasDrop(0.25)(X)

    X = KerasFlat()(X)
    X = KerasDens(512)(X)
    X = KerasActi('relu')(X)
    X = KerasDrop(0.5)(X)

    X = KerasDens(1, activation='sigmoid', name='fc')(X)

    # Create Keras model instance.
    model = Model(inputs = X_input, outputs = X, name='IDCModel')

    return model
```

Listing 2. Convolutional neural network CIFAR10 model to predict the presence of IDC on images.

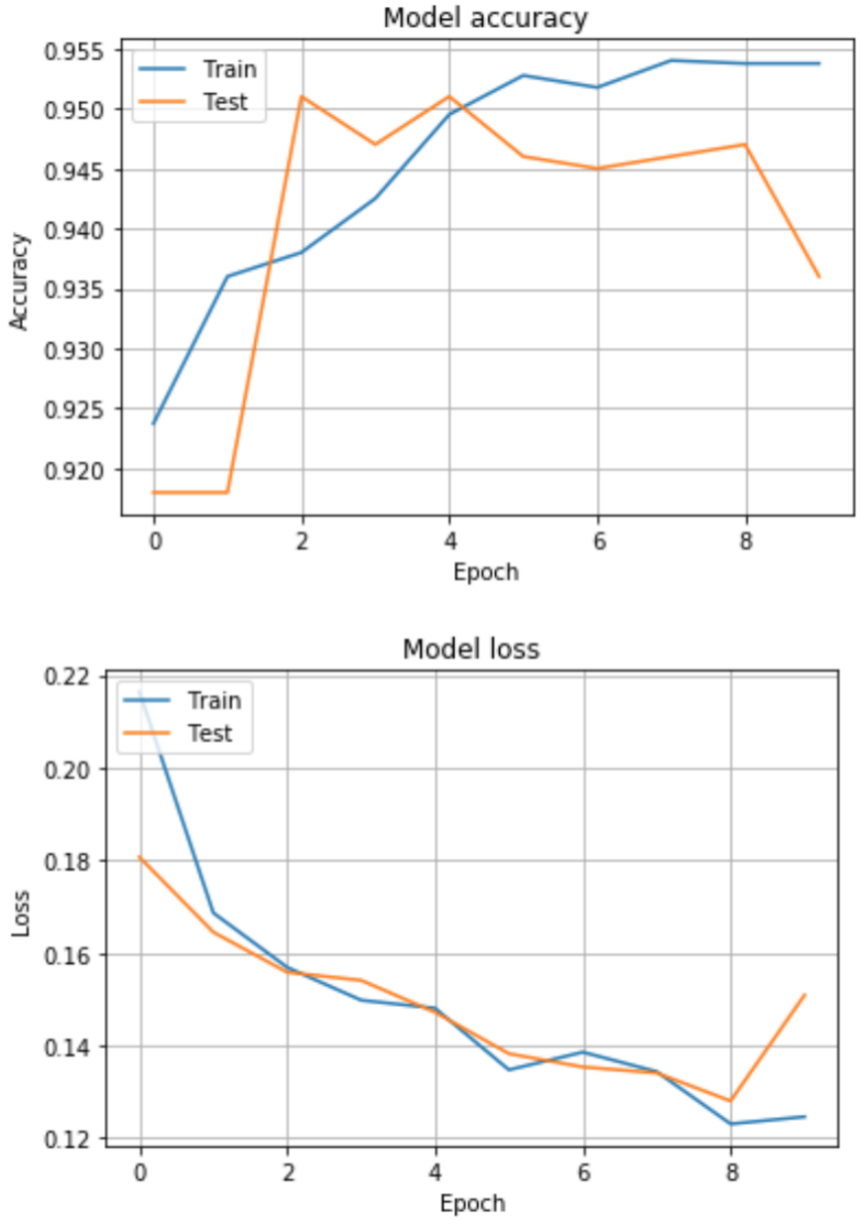


Figure 26. The prediction accuracy and model loss of CIFAR10 CNN.

The model is trained with 10 epochs and batch_size is 64. The training time is around 40 minutes. This model has more layers and more complexity than MNIST, thus the training time is longer. This model achieves the accuracy of the training set is 95.4% and the accuracy of the test set is 93.6%. The model loss for training set is 0.125 and loss for test set is 0.15. There is a gap between the performance of the training set and test set, so this model still has variance. A way to improve this model is to add regularization techniques to avoid overfitting and help the model perform better in the test sets. But overall, the result of 93.6% accuracy is pretty good for a simple model.

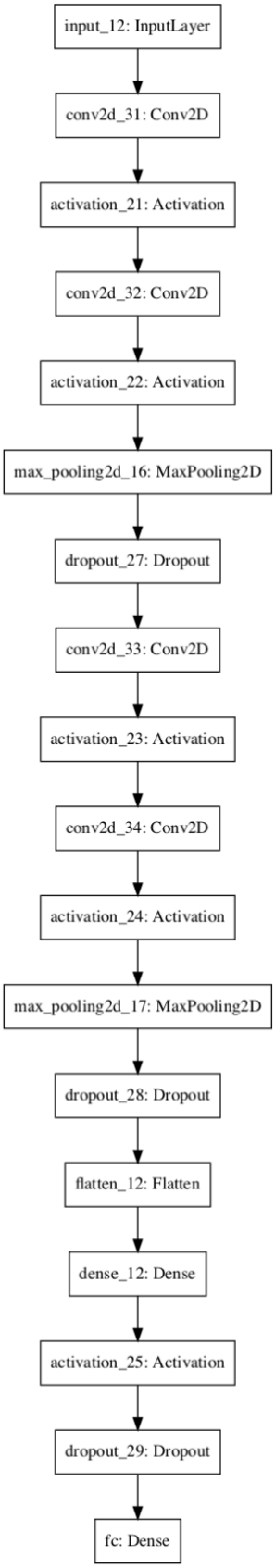


Figure 27. Outline of the IDC prediction CNN CIFAR10 model.

5.2.3 Results, evaluation and discussion

This chapter covers the observations of the results using two convolutional network models. Many different ways can be applied to measure the performance of these two models. Confusion matrix and summary tables are two of them. The summary table introduces the layers structures of the models, one of the primary figures is the number of parameters. The confusion matrix is a table that evaluates the performance of a classification model.

Layer (type)	Output Shape	Param #
input_15 (InputLayer)	(None, 50, 50, 3)	0
conv2d_39 (Conv2D)	(None, 50, 50, 32)	896
conv2d_40 (Conv2D)	(None, 48, 48, 64)	18496
max_pooling2d_20 (MaxPooling	(None, 24, 24, 64)	0
dropout_34 (Dropout)	(None, 24, 24, 64)	0
flatten_15 (Flatten)	(None, 36864)	0
dense_15 (Dense)	(None, 128)	4718720
dropout_35 (Dropout)	(None, 128)	0
fc (Dense)	(None, 1)	129

Total params: 4,738,241
Trainable params: 4,738,241
Non-trainable params: 0

Figure 28: The summary table of MNIST CNN model.

Layer (type)	Output Shape	Param #
input_16 (InputLayer)	(None, 50, 50, 3)	0
conv2d_41 (Conv2D)	(None, 50, 50, 32)	896
activation_26 (Activation)	(None, 50, 50, 32)	0
conv2d_42 (Conv2D)	(None, 48, 48, 64)	18496
activation_27 (Activation)	(None, 48, 48, 64)	0
max_pooling2d_21 (MaxPooling	(None, 24, 24, 64)	0
dropout_36 (Dropout)	(None, 24, 24, 64)	0
conv2d_43 (Conv2D)	(None, 24, 24, 64)	36928
activation_28 (Activation)	(None, 24, 24, 64)	0
conv2d_44 (Conv2D)	(None, 22, 22, 64)	36928
activation_29 (Activation)	(None, 22, 22, 64)	0
max_pooling2d_22 (MaxPooling	(None, 11, 11, 64)	0
dropout_37 (Dropout)	(None, 11, 11, 64)	0
flatten_16 (Flatten)	(None, 7744)	0
dense_16 (Dense)	(None, 512)	3965440
activation_30 (Activation)	(None, 512)	0
dropout_38 (Dropout)	(None, 512)	0
fc (Dense)	(None, 1)	513

Total params: 4,059,201
Trainable params: 4,059,201
Non-trainable params: 0

Figure 29. The summary table of CIFAR10 CNN model.

The two figures above are the summary tables of the MNIST and CIFAR10 models. MNIST model has more parameters, 4.7 million compared to 4 million of CIFAR10 model and fewer layers, so it is more sensitive with changes and takes more time to study, but very flexible and easier to adopt, even though it has fewer layers. CIFAR10 model has more convolutional layers, with 32 or 64 filters, and there are activation functions, max pooling, and dropout between them. These layers help to detect edges efficiently, so this model is more sensitive with small details of images. In the other

hand, CIFAR10 model also has fewer parameters, so the study speed is slightly better, but the model may fit less to the data set.

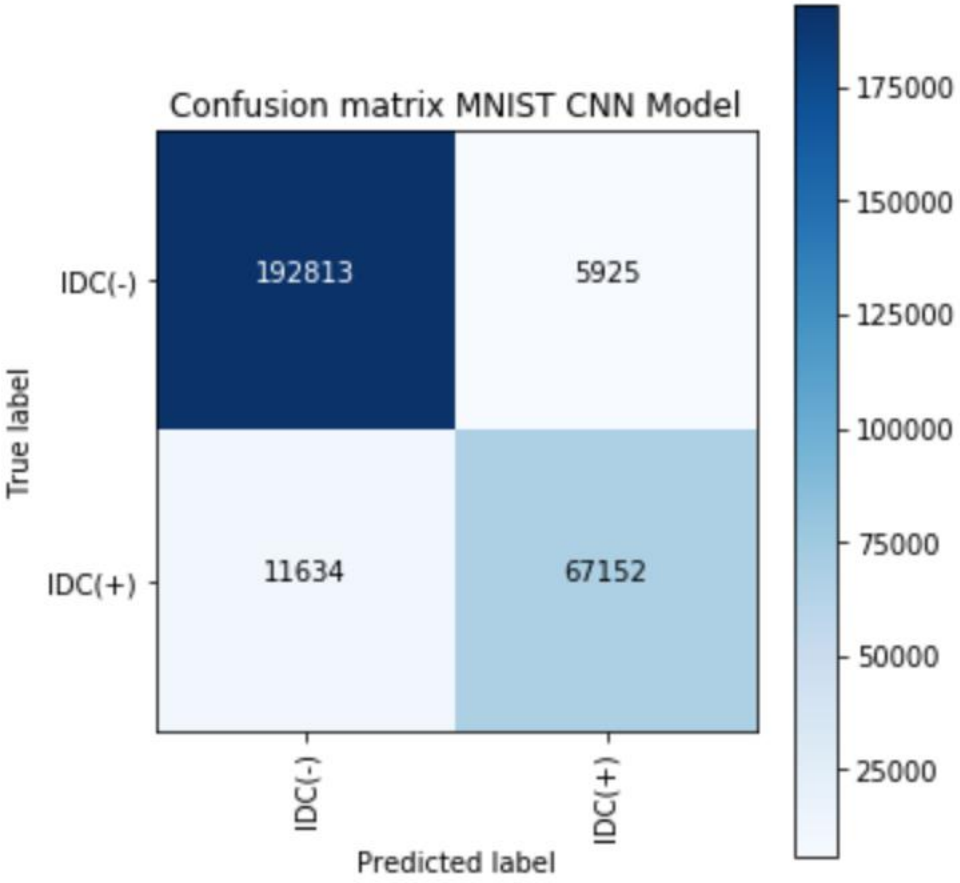


Figure 30. Confusion matrix of MNIST CNN model.

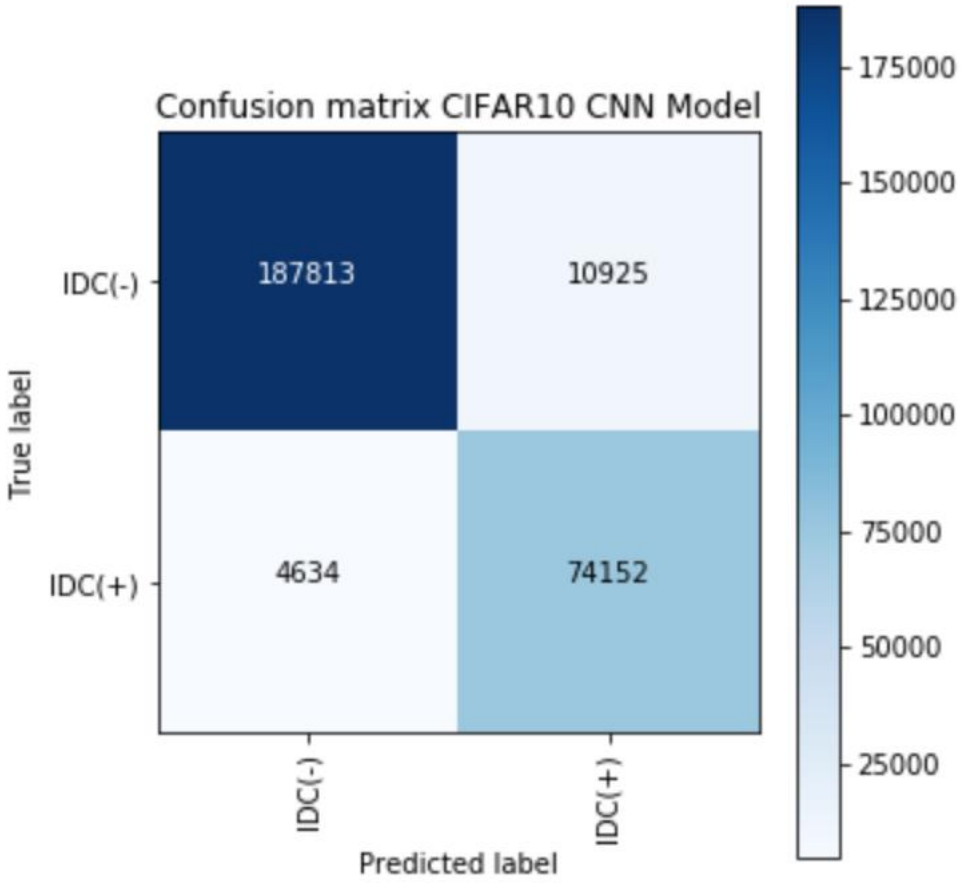


Figure 31. Confusion matrix of CIFAR10 CNN model.

The two figures above are the confusion matrixes of MNIST and CIFAR10 CNN models. The two models work well in the prediction tests, with the accuracies are above 90%. In figure 30, MNIST predicts lean to negative and makes more false prediction when the images are positive, with false positive is 16.4% and false negative is 3%. On the other hands, the CIFAR10 model is more balance, the false positive and false negative are roundly 6%. To conclude, both models have strengths and weaknesses, but CIFAR10 CNN model is more successful. It has fewer parameters, with more convolutional layers, this model is more suitable with this dataset, which required to be sensitive with small details.

6 Conclusion

To conclude, this project emphasizes the indispensable role of artificial intelligence in modern life. Deep learning has been and will improve more and more, to provide great tools, better solutions for the human. And AI can be used in a majority of aspects, especially healthcare and well-being. By using deep learning, this project is able to develop a network that can diagnosis invasive ductal carcinoma in scanned specimens' images. However, due to the lack of computational power, time and a larger amount of

training data, this neural network is still very simple and has a lot of room for improvements. For example, one suggestion is to apply the design of Residual network, add the weights of the previous outcome to improve performance, or add more filter layers, or use more regularization to avoid variance and avoid bias and data augmentation. A modern neural network used by large corporations could contain thousands to ten thousand layers, with millions of parameters. To build a neural network likes that required a strong core team, huge dataset and superpower computers used to train machine for many days and months. Finally, this small project is pretty successful to predict the presence of cancer tumor up to 95% accuracy for any given scanned specimens.

References

1 David W Hosmer and Stanley Lemeshow. Applied logistic regression. Canada, 31 July 1989; 1-16. http://resource.heartonline.cn/20150528/1_3kOQSTg.pdf

2 Hezhiyao. Neural network and Deep learning. November 29, 2017. URL: https://www.cnblogs.com/hezhiyao/p/7922712.html

3 François Chollet. Deep learning with python. Manning Publications Company, 28 October 2017. Chapter 3, 56-66. http://faculty.neu.edu.cn/yury/AAI/Textbook/Deep%20Learning%20with%20Python.pdf

4 Neural network, Wikipedia, 20 March 2019. URL: https://en.wikipedia.org/wiki/Neural_network

5 Artificial Neural network, Wikipedia, 26 March 2019. URL: https://en.wikipedia.org/wiki/Artificial_neural_network

6 Jayesh Bapu Ahire, The artificial neural networks handbook, 24 August 2018. URL: https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4

7 Wikipedia, Sigmoid function, 22 February 2019. URL: https://en.wikipedia.org/wiki/Sigmoid_function

8 Andrew Yan-Tak Ng, Machine Learning Yearning, Technical Strategy for AI Engineers, in the Era of Deep Learning, 2018. https://gallery.mailchimp.com/dc3a7ef4d750c0abfc19202a3/files/4e8b6931-c0c0-4093-8379-0ab43302eedf/MLY_V0.5_Full_Draft.pdf

9 Martin Ford, Architects of Intelligent: The Truth about AI from the People Building it, 16 November 2018.

10 Nikki Castle, Supervised vs Unsupervised Machine Learning, 13 July 2017. URL: https://www.datascience.com/blog/supervised-and-unsupervised-machine-learning-algorithms

11 Diederik P. Kingma and Jimmy Lei Ba, ADAM: A method for stochastic optimization, 30 January 2017. URL: https://arxiv.org/pdf/1412.6980.pdf

12 Seema Singh, Understanding the Bias-Variance Tradeoff, 21 May 2018. URL: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

13 Karthik Ramasubramanian and Abhishek Singh, Machine Learning Using R, Apress, Berkeley, California, 2019.

14 Prashant Gupta, Regularization in Machine Learning, 15 November 2017. URL: https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

15 Renu Khandelwal, L1 and L2 Regularization, 4 November 2018. URL: https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2

16 Amar Budhiraja, Dropout in (Deep) Machine leaning, 16 December 2016. URL: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

17 Alex Krizhevsky, Nitist Srivastava, Ruslan Salakhutdinov, Ilya Sutskever and, Geoffrey Hinton. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014. URL: http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf

18 Keita Kurita, An Overview of Normalization Methods in Deep Learning, 30 November 2018. URL: http://mlexplained.com/2018/11/30/an-overview-of-normalization-methods-in-deep-learning/

19 Jason Brownlee, Basic of Linear Algebra for Machine Learning, Discover the Mathematical Language of Data in Python, 2018. URL: https://www.mobt3ath.com/uplode/book/book-33342.pdf

20 Muhammad Rizwan, Gradient Descent with Momentum, 20 May 2018. URL: https://engmrk.com/gradient-descent-with-momentum/

21 Andrew Ng, Deep Learning Specialization, 2018.

22 Ayoosh Kathuria, Intro to optimization in deep learning: Momentum, RMSProp and Adam, 13 June 2018. URL: https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

23 Rohith Gandhi, A Look at Gradient Descent and RMSprop Optimizers, 19 Jun 2018. URL: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b

24 Matthias Feurer and Frank Hutter , Hyperparameter optimization, 2018. URL: https://www.automl.org/wp-content/uploads/2018/09/chapter1-hpo.pdf

25 Wikipedia, Bayesian optimization, 13 February 2019.
URL: https://en.wikipedia.org/wiki/Bayesian_optimization

26 Wikipedia, Bayesian optimization, 13 February 2019.
URL: https://en.wikipedia.org/wiki/Bayesian_optimization

27 Pradeep Pujari, Md. Rezaul Karim, Mohit Sewak, Practical Convolutional Neural Networks, Packt Publishing, February 2018.
URL: https://www.oreilly.com/library/view/practical-convolutional-neural/9781788392303/928bb5f2-d518-4063-acd2-a87bdd786b6d.xhtml

28 Tzu TaLin, Deep Learning, Batch Normalization, 20 July 2017.
URL: http://tzutalin.blogspot.com/2017/07/deep-learning-batch-normalization-note.html

29 Sergey Ioffe and Christian Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Google, 1600 Amphitheatre Pkwy, Mountain View, CA, 2015.
URL: http://proceedings.mlr.press/v37/ioffe15.pdf

30 Sumit Saha, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way, 15 December 2018.
URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

31 Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, University of Toronto, 2012.
URL: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

32 Siddharth Das, CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more, 16 November 2017.
URL: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

33 Yaan LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, Gradient-Based Learning Applied to Document Recognition, Proc. Of the IEEE, November 1998.
URL: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

34 Australian Institute of Health and Welfare & National Breast Cancer Centre. Breast cancer in Australia: an overview, 2006.
URL: https://www.myvmc.com/diseases/breast-cancer-invasiveinfiltrating-ductal-carcinoma-idc/

35 Wikipedia, Ductal carcinoma in situ, 6 April 2019.
URL: https://en.wikipedia.org/wiki/Ductal_carcinoma_in_situ

36 Laura J. Martin, MD, Invasive Ductal Carcinoma (IDC) & Ductal Carcinoma in Situ (DCIS), WebMD Medical Reference, 27 February 2019.
URL: https://www.webmd.com/breast-cancer/ductal-carcinoma-invasive-in-situ#1-1

37 Laura J. Martin, MD, Invasive Ductal Carcinoma (IDC) & Ductal Carcinoma in Situ (DCIS), WebMD Medical Reference, 27 February 2019.
URL: https://www.webmd.com/breast-cancer/ductal-carcinoma-invasive-in-situ#1-1

38 Christina Chun, MPH, Statistics on breast cancer survival rates by stage, 10 April 2017.
URL: https://www.medicalnewstoday.com/articles/316867.php

39 Marshall Hargrave, Deep Learning, 14 March 2019.
URL: https://www.investopedia.com/terms/d/deep-learning.asp

40 Townend Oliver; Nielsen Jens Brehm, PhD; and Ramsgaard Jesper, MA, Real-life of the Applications for Machine Learning in Hearing Aids, 26 March 2018.
URL: http://www.hearingreview.com/2018/03/real-life-applications-machine-learning-hearing-aids/

41 Paul Mooney, Breast Histopathology Images, 2018.
URL: https://www.kaggle.com/paultimothymooney/breast-histopathology-images

42 Muneeb ul Hassan, VGG16 – Convolutional Network for Classification and Detection, 20 November 2018.
URL: https://neurohive.io/en/popular-networks/vgg16/

43 Xiangyu Zhang, Jianhua Zou, Kaiming He, Jian Sun, Accelerating Very Deep Convolutional Networks for Classification and Detection, 26 May 2015.

44 Sik-Ho Tsang, Review: GoogLeNet (Inception v1)— Winner of ILSVRC 2014 (Image Classification), 24 August 2017.
URL: https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7

45 Christian Szegedy, Wei Liu , Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Going Deeper with Convolutions, Google Inc. , University of North Carolina, Chapel Hill, IEEE Xplore, 2015.
URL: https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

46 Sik-Ho Tsang, Review: LeNet-1, LeNet-4, LeNet-5, Boosted LeNet-4 (Image Classification), 8 August 2018.
URL: https://medium.com/@sh.tsang/paper-brief-review-of-lenet-1-lenet-4-lenet-5-boosted-lenet-4-image-classification-1f5f809dbf17

47 Min Lin, Qiang Chen, Shuicheng Yan, Network In Network, Department of Electronic & Computer Engineering National University of Singapore, Singapore, 4 March 2014.
URL: https://arxiv.org/pdf/1312.4400.pdf

Appendix 1. List of figures

Figure 1. The relation between deep learning, machine learning, and artificial intelligence. Image copied from [40]. 2
Figure 2. The relationship between components of a neural network. Image copied from [3]. 3
Figure 3. A simple neural network with two hidden layers. Image copied from [6]. ..3
Figure 4. The general architecture of learning algorithm. Image copied from [2]. 6
Figure 5. Summary of gradient descent. Source [8]. 7
Figure 6. Bias and Variance Trade-off. Image copied from [13]. 8
Figure 7. Dropout regularization. Image copied from [17]. 9
Figure 8. Equations for batch normalization. Image copied from [30]. 10
Figure 9. Stochastic gradient descent vs Gradient descent. Image copied from [21]. 11
Figure 10. Stochastic gradient descent vs Mini-batch gradient descent. Image copied from [21]. 11
Figure 11. The blue dots are the direction of the current gradient. The red arrows show the direction after taking momentum, make them move toward the optimal point faster and more stable. Images copied from [21]. 12
Figure 12. The equations for gradient descent with momentum. Source [23]. 12
Figure 13. The equations for RMSprop optimizer. Image copied from [23]. 13
Figure 14. Batch normalization. Images copied from [27]. 15
Figure 15. The equations of Batch Normalization. Images copied from [28]. 16
Figure 16. CNN sequence with two convolution layers. An example of a simple model of CNN, there are two convolution layers and one fully connected layer in the end to justify the outputs. Images copied from [30]. 17
Figure 17. The architecture of LeNet-5. Each layer contains a feature map of filters handling the input image. Image copied from [33]. 17
Figure 18. The architecture of AlexNet. Images copied from [32]. 18
Figure 19. The architecture of GooleNet. Image copied from [44]. 18
Figure 20. VGG16 Architecture. Image copied from [42]. 19
Figure 21. The architecture of ResNet. Image copied from [32]. 20
Figure 22. Ductal carcinoma. Image copied from [35]. 21
Figure 23. Samples with IDC (-) and IDC (+). 22

Figure 24. The prediction accuracy and model loss of MNIST CNN. 24
Figure 25. Outline of the IDC prediction CNN MNIST model. 25
Figure 26. The prediction accuracy and model loss of CIFAR10 CNN..... 27
Figure 27. Outline of the IDC prediction CNN CIFAR10 model. 28
Figure 28: The summary table of MNIST CNN model. 29
Figure 29. The summary table of CIFAR10 CNN model. 30
Figure 30. Confusion matrix of MNIST CNN model. 31
Figure 31. Confusion matrix of CIFAR10 CNN model. 32