**VIETNAM GENERAL CONFEDERATION OF LABOR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL REPORT**

**Machine Learning**

*Instructor*: Le Anh Cuong

*Student*:Hoang Dinh Quy Vu - **521H0517**

Nguyen Hoang Phuc - **521H0511**

Tran Nguyen Duy Bao - **521H0493**

Class: **21H50302**

**HO CHI MINH CITY, 2023**

**VIETNAM GENERAL CONFEDERATION OF LABOR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**FINAL REPORT**

**Machine Learning**

*Instructor*: Le Anh Cuong

*Student*:Hoang Dinh Quy Vu - **521H0517**

Nguyen Hoang Phuc - **521H0511**

Tran Nguyen Duy Bao - **521H0493**

Class: **21H50302**

**HO CHI MINH CITY, 2023**

# ACKNOWLEDGEMENT

I would like to thank the teachers of Ton Duc Thang University in Ho Chi Minh City, as well as the teachers of the Department of discrete structure for IT for their enthusiastic guidance during my time studying at the school; Special thanks to Mr.Cuong who directly guided me to complete my graduation thesis.

I sincerely!

**PROJECT COMPLETED**

**AT TON DUC THONG UNIVERSITY**

I hereby certify that this is my thesis project and was conducted under the guidance of Mr.Cuong The research content and results in this topic are honest and have not been previously published in any form. The data in the tables are collected by the author from various sources clearly stated in the reference section for analysis, comments, and evaluation.

Furthermore, this thesis includes comments, evaluations, and data from other authors and organizations, all of which are cited and annotated with their sources.

If any fraud is detected, I will fully take responsibility for the content of my thesis. Ton Duc Thang University is not liable for any copyright infringement or violation caused by me during the implementation process (if any).

*Ho Chi Minh city, 30th April, 2023*

*Author*

*(Sign and write your full name)*

# TEACHER'S CONFIRMATION AND ASSESSMENT SECTION

**The confirmation part of the instructor**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh city, 30th April, 2023*

**The evaluation part of the teacher marks the test**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Ho Chi Minh city, 30th April, 2023*

# TABLE OF CONTENTS

[ACKNOWLEDGEMENT i](#_Toc14208)

[TEACHER'S CONFIRMATION AND ASSESSMENT SECTION iii](#_Toc3111)

[TABLE OF CONTENTS 1](#_Toc30665)

[LIST OF SYMBOLS AND ABBREVIATIONS 3](#_Toc19855)

[CHAPTER 1: OVERFITTING 4](#_Toc16872)

[1.1 Introduction 4](#_Toc1227)

[1.2 Cause of overfitting 4](#_Toc6540)

[1.3 Solutions 5](#_Toc19330)

[1.3.1 Regularization 5](#_Toc2976)

*[1.3.1.1 L1 Regularization](#_Toc5424)* [6](#_Toc5424)

*[1.3.1.2 L2 Regularization](#_Toc30454)* [7](#_Toc30454)

*[1.3.1.3 Dropout](#_Toc440)* [7](#_Toc440)

[1.3.2 Cross-validation 8](#_Toc3813)

[1.3.3 Early Stopping 9](#_Toc26094)

[1.3.4  Noise Reduction 9](#_Toc4177)

[1.3.5 Augment the training data 10](#_Toc16674)

[1.4 Comparation result 10](#_Toc9076)

[1.4.1 Simple model 10](#_Toc20514)

[1.4.2 Model applied Dropout method 11](#_Toc31065)

[1.4.3 Model applied K-fold cross validation 11](#_Toc12640)

[CHAPTER 2: RECOMMENDATION SYSTEMS 13](#_Toc27129)

[2.1 Content-Based Recommendations 13](#_Toc27409)

[2.1.1 User Profile 13](#_Toc16459)

[2.1.2 Item Profile 14](#_Toc27526)

[2.2 Utility Matrix 14](#_Toc3018)

[2.3 Recommending Items to User 15](#_Toc7439)

[2.3.1 Method 1 15](#_Toc19264)

[2.3.2 Method 2 16](#_Toc12833)

[2.4 Cosine similarity 17](#_Toc8917)

[2.4.1 Introduction 17](#_Toc13216)

[2.4.2 Formula 18](#_Toc8121)

[CHAPTER 3: CNN – Convolutional Neural Network 19](#_Toc27253)

[3.1 Abstract 19](#_Toc24024)

[3.2 Theory 19](#_Toc18140)

[3.3 CNN architecture 19](#_Toc18555)

[3.3.1 Convolutional layers 20](#_Toc15113)

[3.3.2 Polling layers 21](#_Toc13520)

[3.3.3 Fully connected layer 23](#_Toc27672)

*[3.3.3.1 Forward propagation](#_Toc19869)* [23](#_Toc19869)

*[3.3.3.2 Loss function](#_Toc29542)* [24](#_Toc29542)

*[3.3.3.3 Backward propagation](#_Toc16819)* [25](#_Toc16819)

[CHAPTER 4: DEEP LEARNING - MACHINE LEARNING 27](#_Toc31682)

[4.1 Introduction: 27](#_Toc29407)

[4.2 Comparation 27](#_Toc8475)

# LIST OF SYMBOLS AND ABBREVIATIONS

**SYMBOLS**

**ABBREVIATIONS**

CHAPTER 1: OVERFITTING

1.1 Introduction

There is an un-detouring issue in supervised machine learning when the model does not generalize correctly from observed data to unseen data. That is called overfitting, a scenario when a model performs perfectly on a training set but fits poorly on a testing set. This is due to the fact that an over-fitted model has difficulty coping with pieces of the information in the testing set, which may be different from those in the training set. On the other hand, over-fitted models tend to memorize all the data, including unavoidable noise on the training set, instead of learning the discipline hidden behind it.

Overfitting occurs when a machine learning model tries to cover all the data points or more than the required data points present in the given dataset.

1.2 Cause of overfitting

- Lack of data: The data requirement for training a machine learning model is usually large. Because when a model tries to learn from a small dataset, it will tend to have greater control over the dataset and will make sure to satisfy all the data points exactly.

- Noisy data: Noise is the measure of irrelevant information or randomness in a dataset. If a dataset contains too much noise, the model tends to memorize the noise instead of learning the underlying patterns. Noisy data also causes an overly complex model.

- Model is too complex: A model is too complex and becomes overfitting when it is not learned from the sufficient data required.

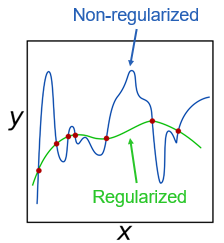
- Unbalance data: An unbalanced data is also a cause to overfitting, some features may dominate the others

- Overlearning: Overlearning is when a model has learned for many epochs, it may start to memorize the training data, rather than learning the underlying patterns.

1.3 Solutions

1.3.1 Regularization

The output of a model is generally affected by multiple features. The larger the number of features, the more complicated the model is. An overfitting model tends to take all features into consideration, even though some of them have very limited effect on the final output. Or even worse, some of them are noises which are meaningless to the output.



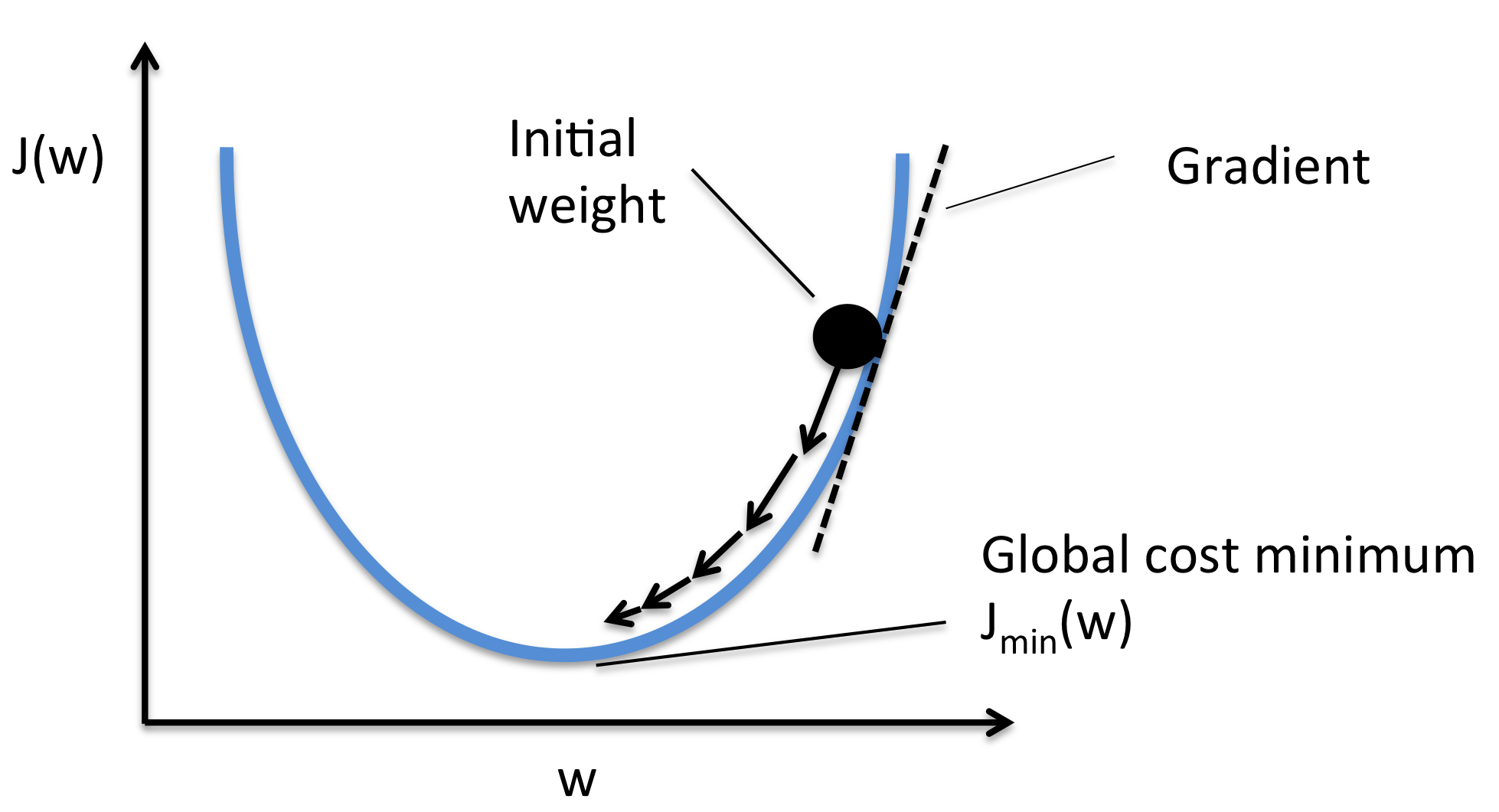
Picture 1.1 Regularization and Non-regularization

There are two kinds of solutions to limit these cases

* Select only the useful features and remove the useless features from the model.
* Minimize the weights of the features which have little influence on the final classification.

General idea of these solutions above is to limit the effect of useless features. However, it is hard to consider which features are useless, so trying to minimize the cost function of the model is usually applied in order to limit them. To do this, a regularization term or a regularizer is added to the cost function.

* ω is weight. Gradient Descent can be used to determine the set of weights.
* X is training set.
* y is the labeled value (true value).
* is regularization coefficient.
* is the penalty term.



#### *1.3.1.1 L1 Regularization*

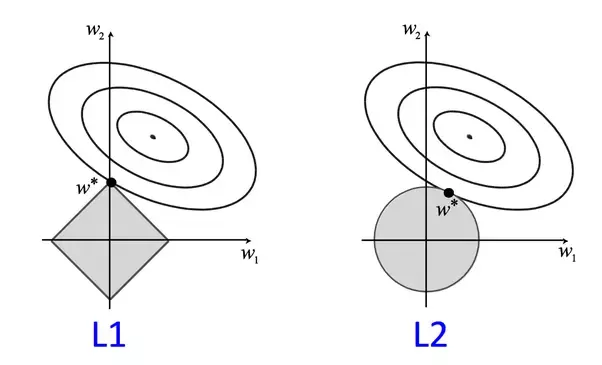
L1 regularization or L1 norm or Lasso Regression, prevent overfitting by shrinking the parameters towards 0. This makes some parameter is ignored during training process.

To minimize the cost function, the weight of some features may be set to be zero in order to remove it out of the model. This approach made the model simpler. However, some useful features with lower influence on the output may also be removed, which led to underfit.

#### *1.3.1.2 L2 Regularization*

L2 regularization or L2 norm or Ridge Regression, prevent overfitting by forcing weights to be small, but not making them exactly 0.

This approach makes the networks prefer to learn features with small weights. Instead of rejecting those less valuable features like L1 norm, L2 norm gives them lower weights. So that the model can get as much information as possible. For those features that have considerable influence on the initial cost function, large weights would be given to them.

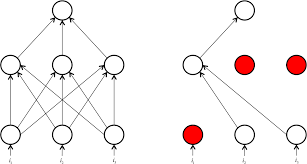


Picture 1.2 L1 regularization vs L2 regularization

#### *1.3.1.3 Dropout*

Dropout is a technique where randomly selected neurons are ignored during training. It is an effective way to prevent overfitting and provides a way to approximately combining exponentially many different neural network architectures.

The idea is this method is temporarily removing a unit out of the network, along with all its incoming and outgoing connections. The choice of which units to drop is random.



Picture 1.3

1.3.2 Cross-validation

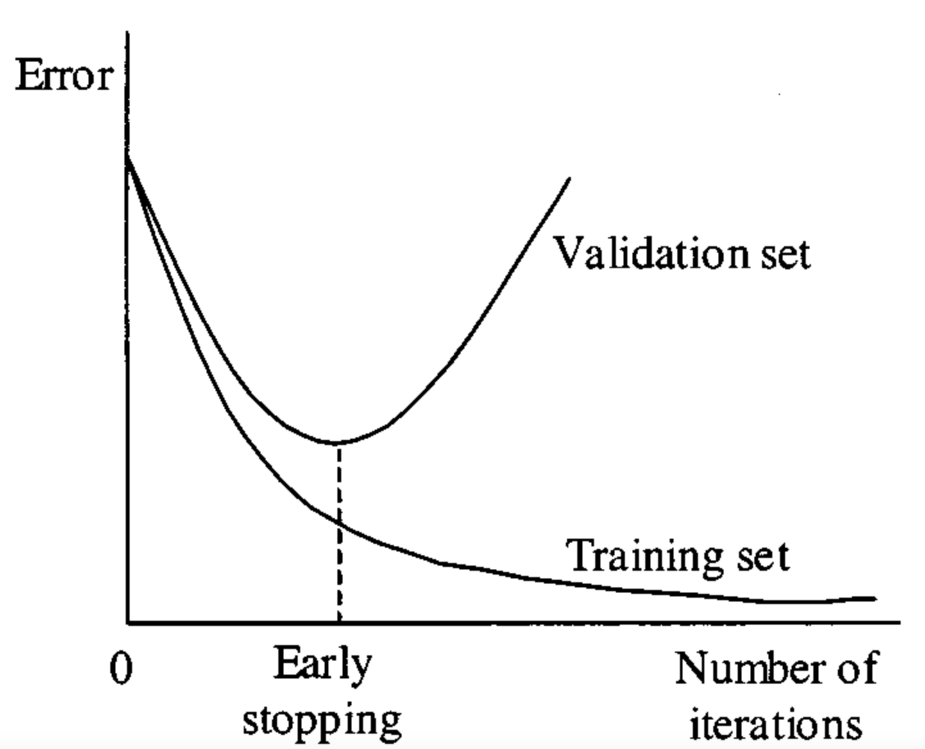
This is a technique used to check the generalization performance of a model on a new dataset. In this method, the dataset is divided into smaller folds, where the model is trained on the majority of the dataset and evaluated on the remaining data. This process is repeated for different combinations of the folds, allowing the model to be tested over all the data.



Picture 1.4 K-Fold cross validation

1.3.3 Early Stopping

This is a method used to prevent overfitting by stopping the training process before the model reaches its optimal accuracy on the training data. This method works by evaluating the model on a validation set, and when the accuracy on the validation set starts to drop or stops improving, the training is stopped.



Picture 1.5 Early stopping

1.3.4  Noise Reduction

Noisy data is one of the reasons led to overfitting. So logically, noise reduction becomes one researching direction for overfitting inhibition.

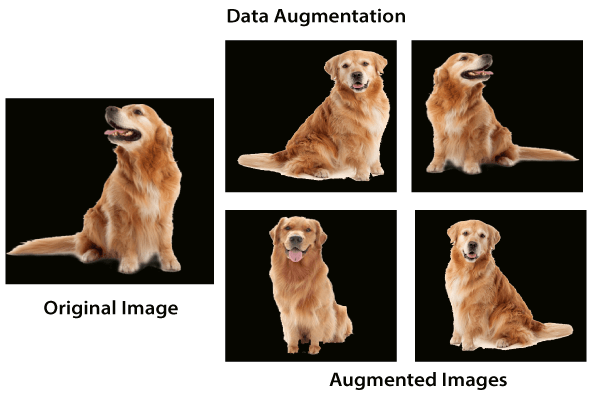
Pruning is a significant theory used to reduce classification complexity by eliminating less meaningful, or irrelevant data, and finally to prevent overfitting and to improve the classification accuracy.

* Pre-pruning algorithms function during the learning process. The tree growth is stopped based on a predefined condition.
* Post-pruning is simpler and faster than pre-pruning, but it may not be as effective as pre-pruning in reducing the model complexity. It also requires a larger validation set and costs more computational power.

1.3.5 Augment the training data

In some cases, overfitting is caused by a lack of training data. Datasets play a crucial role in affecting the performance of a model.

 Model training is a process of tuning hyper-parameters. Well-tuned parameters make a good balance between accuracy and regularity, and then inhibit the effect of overfitting, as well as that of underfitting.

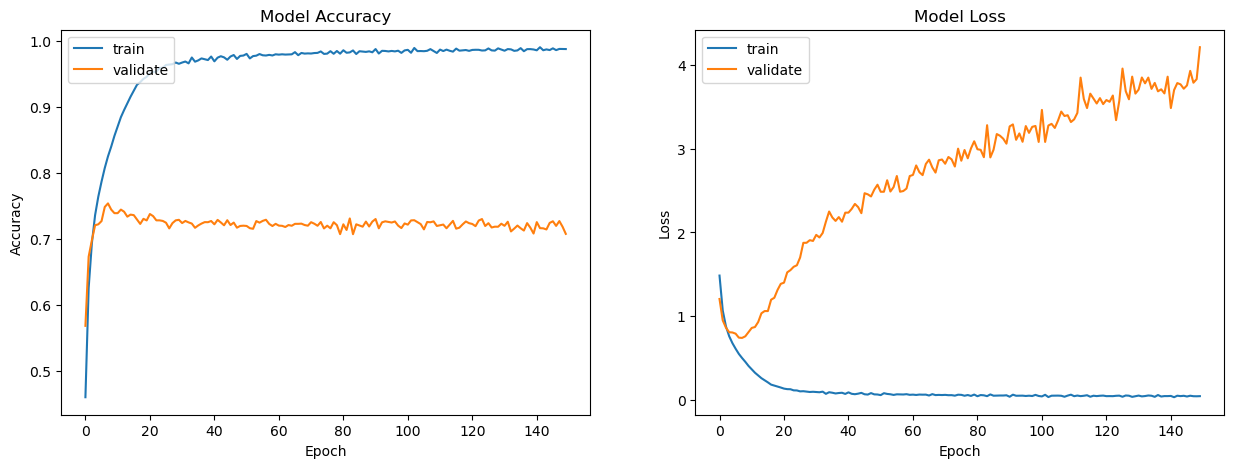


Picture 1.6 Data augmentation

1.4 Comparation result

The results below are all tested on cifar100 dataset without any preprocessing method.

1.4.1 Simple model

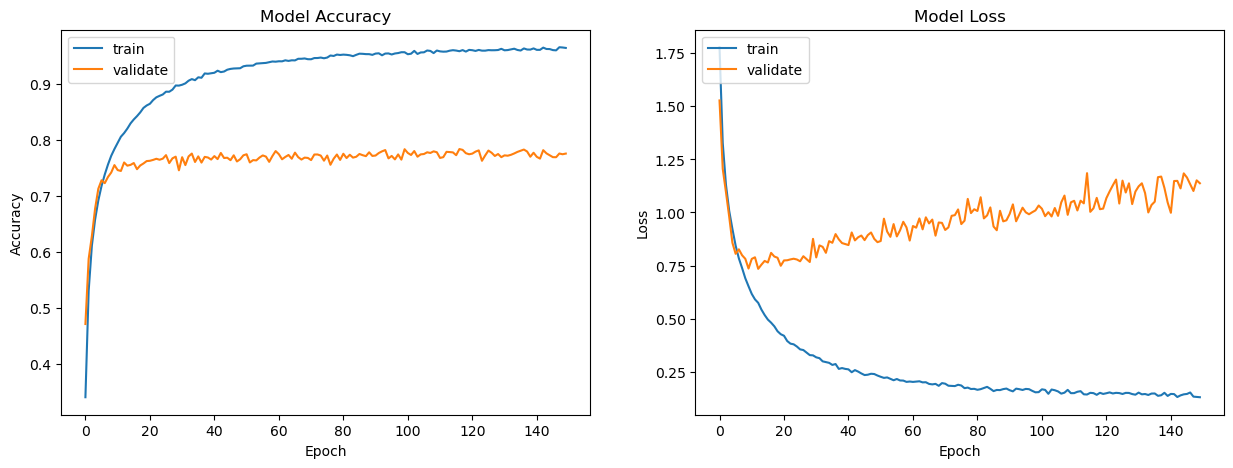


Picture 1.7 Simple model’s accuracy and loss

A model that not apply any method to prevent overfitting is easy to overfitting.

The difference between train lines and validation lines are clearly visible. Accuracy lines tend to stay stable while loss line on validation set tends to increase quickly.

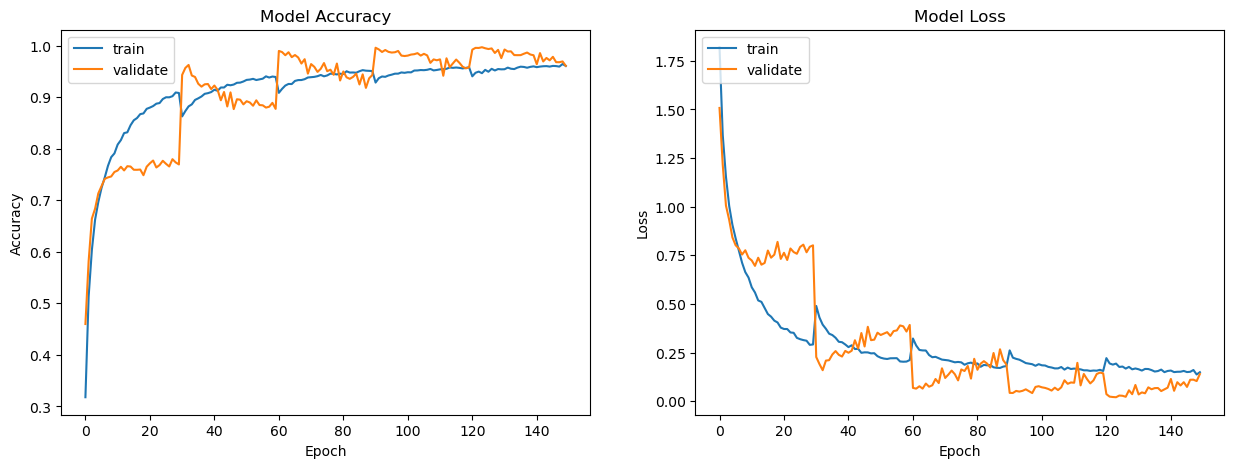
1.4.2 Model applied Dropout method



Picture 1.8 Model applied Dropout’s accuracy and loss

After applied Dropout method, the model still overfitted. However, model have better accuracy on validation set and loss value is more stable.

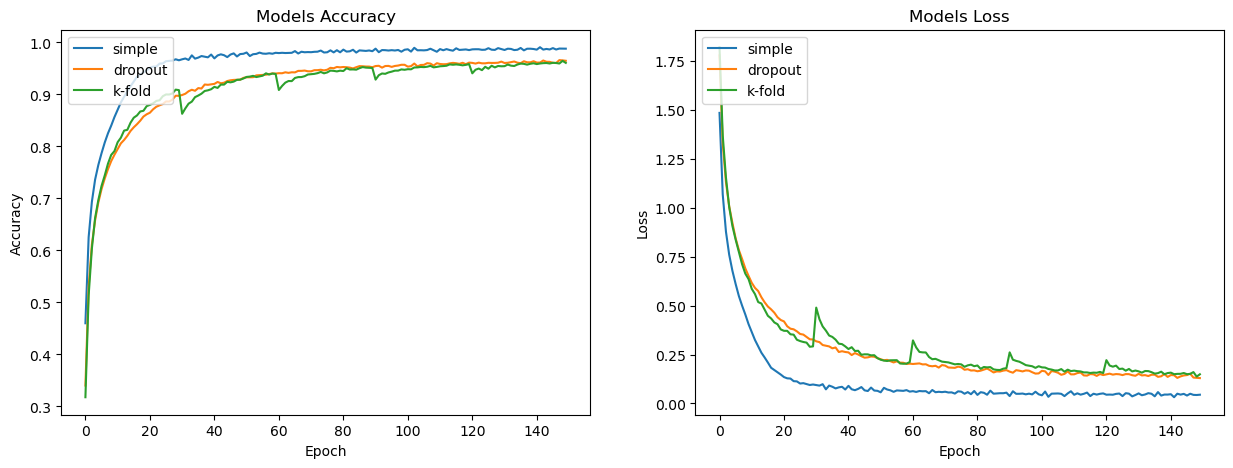
1.4.3 Model applied K-fold cross validation



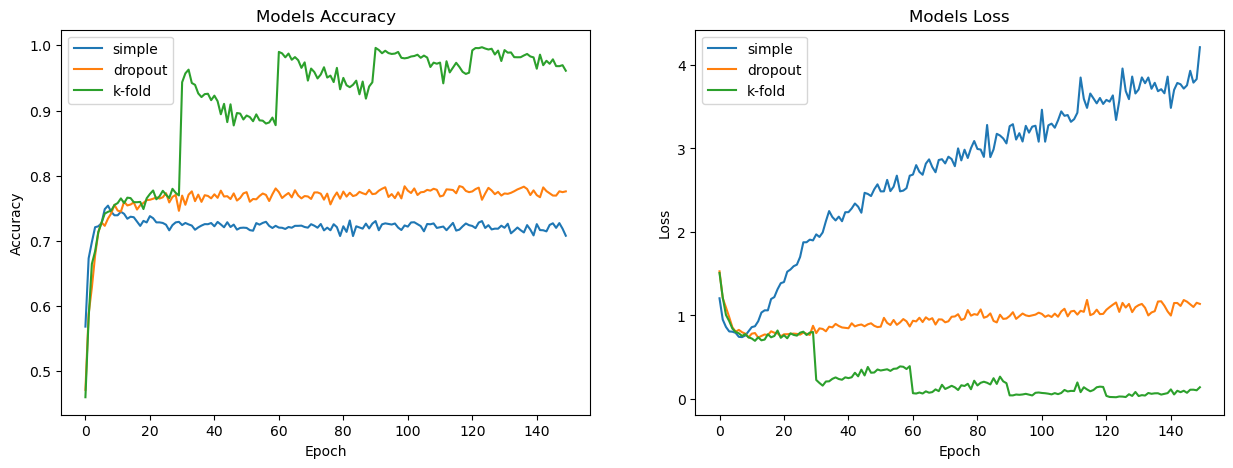
Picture 1.9 Model applied K-fold cross validation’s accuracy and loss

Applied K-fold cross validation method helped the model not to be overfitted. Accuracy and loss value have good trend on both train set and validation set.

1.4.4 Comparation between models above



Picture 1.10 Comparation of models on train set



Picture 1.11 Comparation of models on validation set

On train set, simple model has the best result of accuracy and loss. However, it is the worst result on validation set. Dropout and K-fold method provide good trend on accuracy and loss lines on train set but K-fold has better result on validation set. The result of Dropout method may be not the most optimal one, because the model has not been optimally tuned.

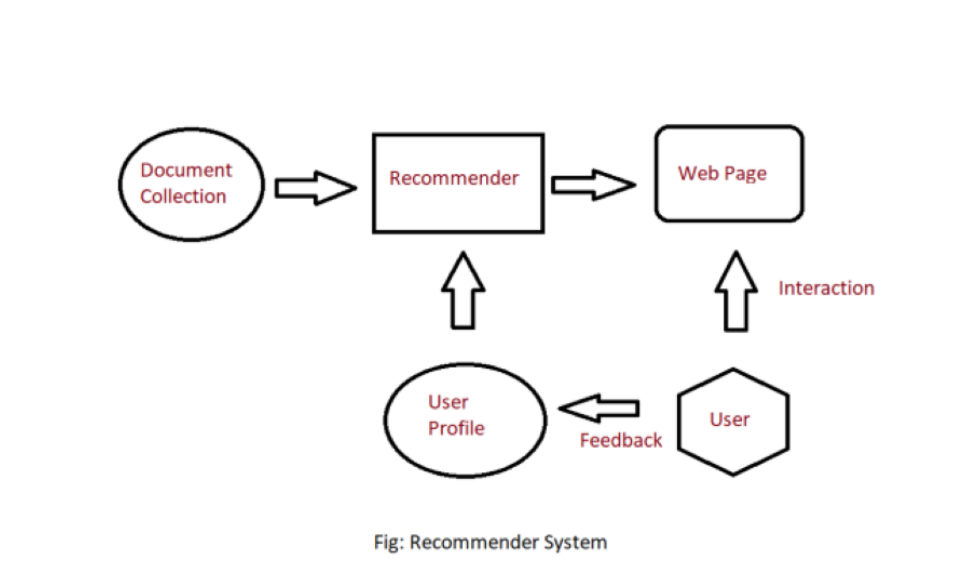
CHAPTER 2: RECOMMENDATION SYSTEMS

2.1 Content-Based Recommendations

Content-based recommendations are a type of recommendation system that makes suggestions to users based on the properties or attributes of the items they have interacted with or liked in the past. This approach relies on analyzing the content of the items, such as their genre, keywords, descriptions, and other features, to identify similarities and differences between them. Based on this analysis, the system generates recommendations for other items that have similar content to the items the user has already enjoyed.

Advantages: Effective in suggesting items in the same category, capable of customizing suggestions based on the individual preferences of each user.

Disadvantages: Limited by specifying the properties of each item, can not make suggestions beyond the known properties of that item.



Picture 2.1 Recommendation system model

2.1.1 User Profile

In the User Profile, we create vectors that describe the user’s preference. In the creation of a user profile, we use the utility matrix which describes the relationship between user and item. With this information, the best estimate we can make regarding which item a user likes, is some aggregation of the profiles of those items.

2.1.2 Item Profile

In Content-Based Recommender, we must build a profile for each item, which will represent the important characteristics of that item. For example, if we make a movie as an item then its actors, director, release year and genre are the most significant features of the movie. We can also add its rating from the IMDB (Internet Movie Database) in the Item Profile.

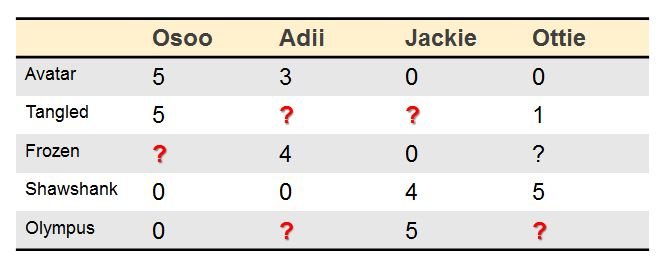
2.2 Utility Matrix

Utility Matrix signifies the user’s preference with certain items. In the data gathered from the user, we have to find some relation between the items which are liked by the user and those which are disliked, for this purpose we use the utility matrix. In it we assign a particular value to each user-item pair, this value is known as the degree of preference. Then we draw a matrix of a user with the respective items to identify their preference relationship.

Build Utility Matrix:

Ask users to rate the product

The second approach is based on user behavior: clip, newspaper, liked



Picture 2.2

2.3 Recommending Items to User

2.3.1 Method 1

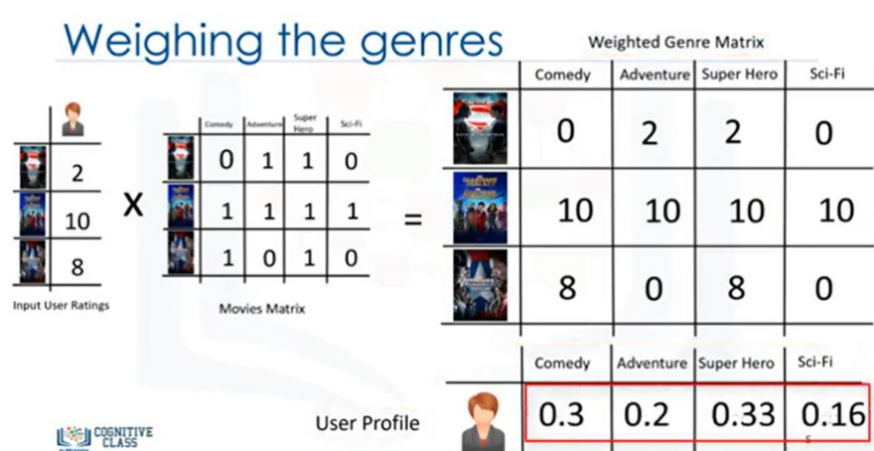
Using the cosine distance between the vectors of the item and the user to determine its preference to the user.

Example:

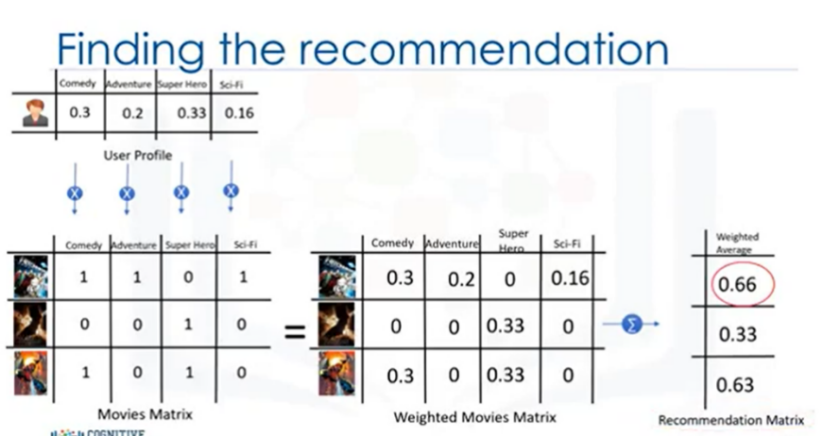
Step 1:



Step 2:



Step 3:

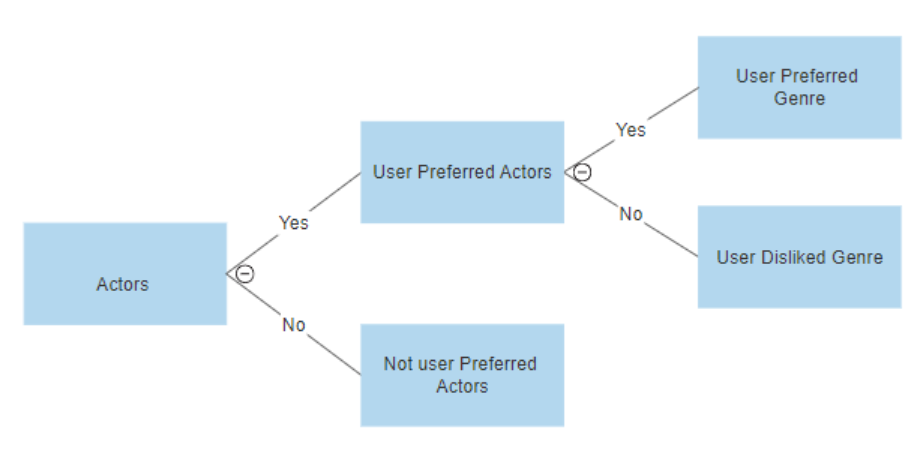


Step 4:



2.3.2 Method 2

Using a classification approach in the recommendation systems too, like we can use the Decision Tree for finding out whether a user wants to watch a movie or not, like at each level we can apply a certain condition to refine our recommendation.



Picture 2.3 Classification approach in recommendation system

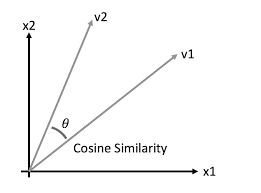
2.4 Cosine similarity

2.4.1 Introduction

Cosine Similarity is a method for calculating the similarity between two vectors in a multi-dimensional space. It is commonly used in natural language processing, data mining, and recommendation systems.

In the case of building a movie recommendation system, Cosine Similarity is used to calculate the similarity between movies. Specifically, each movie is represented as a vector in a multi-dimensional space, where each dimension corresponds to a feature of the movie (e.g. genre, actors, director, runtime, rating, etc.).

Then, the similarity between two movies is calculated by computing the cosine of the angle between the two vectors representing the movies. The result of this calculation indicates the degree of similarity between the two movies, and based on this, the system will suggest movies that are highly similar to the one the user is interested in.



Picture 2.4 Cosine similarity

2.4.2 Formula

The formula for calculating Cosine Similarity between two vectors a and b is:

cosine\_similarity(a, b) = dot(a, b) / (norm(a) \* norm(b))

where:

dot(a, b) is the dot product of vectors a and b.

norm(a) and norm(b) are the Euclidean lengths of vectors a and b.

The value of cosine\_similarity(a, b) always falls within the range of [-1, 1], where a value closer to 1 indicates a higher similarity between vectors a and b, and a value closer to -1 indicates a higher dissimilarity between vectors a and b. When the value is 0, it means that the two vectors are completely different.

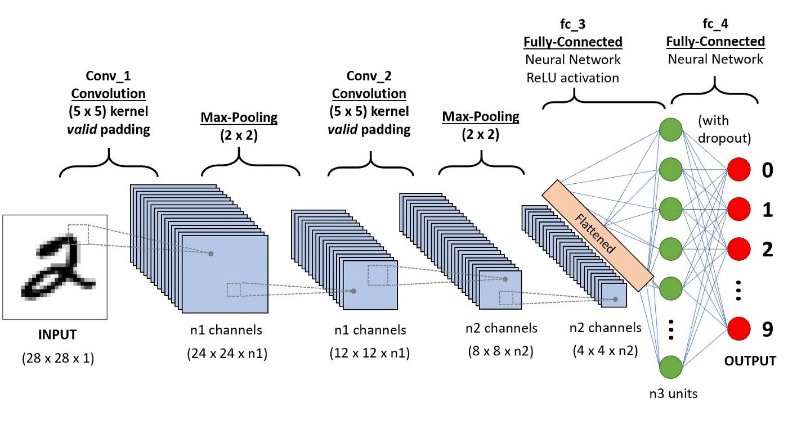
If the value of cosine\_similarity(a, b) is lower than a predetermined threshold, then vectors a and b can be considered significantly different, and not similar. In a movie recommendation system, this threshold can be set based on experience or through testing.

If the value of cosine\_similarity(a, b) is higher than this threshold, it means that vectors a and b are similar to the extent that they can be considered to have common characteristics, and the system will recommend movies with high similarity to the m ovie being evaluated.

CHAPTER 3: CNN – Convolutional Neural Network

3.1 Abstract

Convolutional Neural Networks (CNNs) are a powerful class of deep learning models that have become the state-of-the-art for many computer vision tasks, such as image classification, object detection, and segmentation. In this paper, we provide a comprehensive overview of the theory, architecture, and mathematical formulas behind CNNs, as well as how each layer works in detail. We also discuss recent advancements in CNN research and applications in various fields.



Picture 3.1 CNN model architecture

3.2 Theory

CNNs are composed of several layers, each with a specific purpose. The input layer receives the raw image data, which is then passed through a series of convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply filters to the input data, which helps to extract local features from the image. The pooling layers reduce the size of the output from the convolutional layers, which helps to reduce the number of parameters and **prevent overfitting**. Finally, the fully connected layers use the output from the previous layers to make predictions about the input image.

3.3 CNN architecture

3.3.1 Convolutional layers

Convolutional layers are a fundamental concept in the field of deep learning, specifically for tasks such as image recognition, object detection, and natural language processing.

Convolutional layers are used for primarily two purposes:

* Convolutional layers apply filters to an input image or feature map to find meaningful patterns, also called features, present in the data.
* By applying filters over the input image, convolutional layers convert the image into a smaller, compressed representation, while preserving the essential features that distinguish between different classes.

The convolutional neuron performs an elementwise dot product with a unique filter and the output of the previous layer's corresponding neuron. The convolutional neuron is the result of all of the intermediate results summed together with the learned bias.

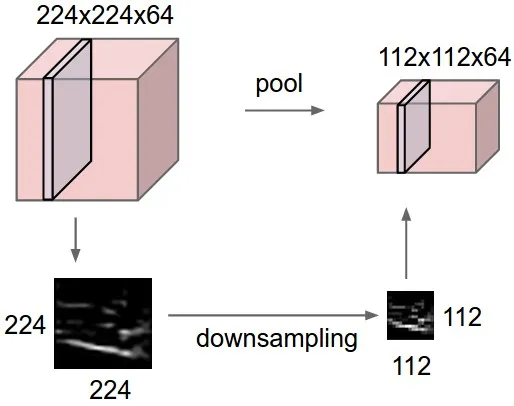
* Each neuron of the previous layer have its own kernel learned to form the output on the next layer

Hyperparameters in convolution function:

* Padding is often necessary when the kernel extends beyond the activation map. Padding conserves data at the border of activation maps, which leads to better performance, and it can help preserve the input's spatial size, which allows an architecture designer to build depper, higher performing networks.
* Kernel size, often also referred to as filter size, refers to the dimensions of the sliding window over the input. Choosing this hyperparameter has a massive impact on the image classification task. A smaller kernel leads to a smaller reduction in layer dimensions,  which allows for a deeper architecture. Conversely, a large kernel size extracts less information, which leads to a faster reduction in layer dimensions, often leading to worse performance. Large kernels are better suited to extract features that are larger.
* Stride indicates how many pixels the kernel should have shifted over at a time. The impact stride has on a CNN is similar to kernel size. As stride decreases, more features are learned because more data is extracted, which also leads to larger output layers. The kernel has to be ensured to slide symmetrically when implementing a CNN.

3.3.2 Polling layers

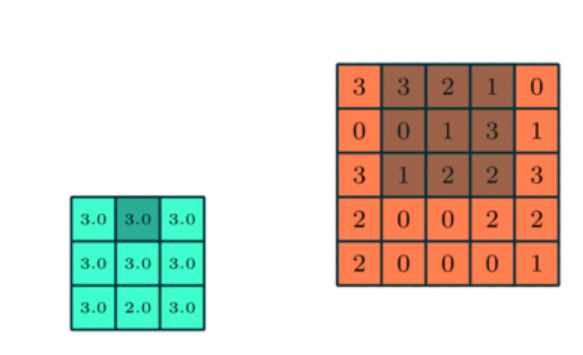
Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down samples of every depth slice in the input by 2 along both width and height, discarding 75% of the activations.



Picture 3.2 Pooling layers

Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged.

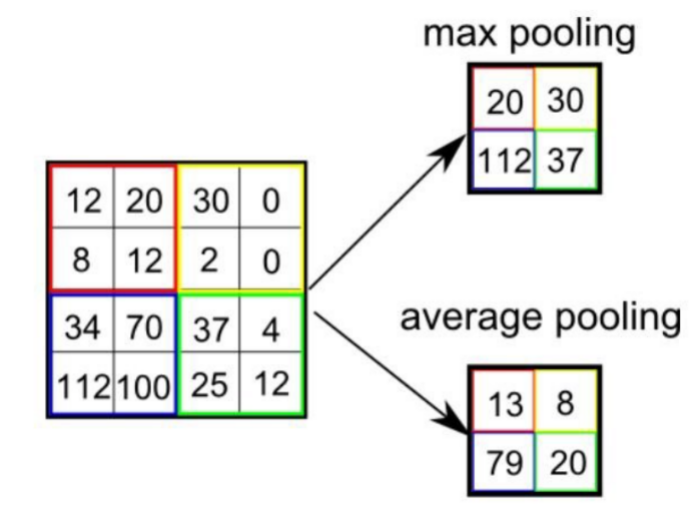
* Call pooling size of size K\*K.
* The input of the pooling layer has size H \* W \* D, we split it into D matrix of size H \* W.
* For each matrix, on the area of size K\*K on the matrix we find the maximum or average of the data and then write it in the resulting matrix. The rules for stride and padding apply like convolution on images.



Picture 3.3 Max-pooling layer

Assuming the filter of size 2x2, the three common ways of creating a pooling layer in a neural network are:

* Max Pooling: It returns the maximum value out of the 4 elements in the receptive field.
* Average Pooling: In this case, the average of the four values is computed to produce the output.



Picture 3.4 Types of pooling layers

3.3.3 Fully connected layer

A fully connected layer is also known as a dense layer, which is one of the most widely used types of layers in ANNs.

The fully connected layer is a type of layer in ANNs where every neuron in the previous layer is connected to every neuron in the next layer. This means that each neuron in the previous layer is connected to every weight in the next layer. The fully connected layer is an essential part of the ANN architecture because it enables the network to learn complex non-linear functions that map input data to output data.



Picture 3.5 Fully connected layers

To perform an ANN, we could use one approach called “Forward propagation”

*3.3.3.1 Forward propagation*

The mathematical method used in the fully connected layer is known as forward propagation. Forward propagation is the process of computing the output of a layer given the input to that layer. In the case of a fully connected layer, forward propagation involves the following steps:

1. Weighted Sum: Each input neuron is multiplied by its corresponding weight, and the products are summed to generate a weighted sum for each output neuron. Mathematically, this can be expressed as:

* z1 = w1 \* x + b1
* z2 = w2 \* x + b2

where x is the input to the layer, w1 and w2 are the weights for the first and second output neurons, respectively, and b1 and b2 are the bias terms for the first and second output neurons,respectively.

1. Activation Function: The weighted sums are then passed through an activation function to introduce non-linearity into the network. The activation function is typically a non-linear function such as the sigmoid, ReLU, or tanh function. Mathematically, this can be expressed as:

* a1 = f(z1)
* a2 = f(z2)

Nowadays, people tends to use ReLU, a simple function but works effectively. It is computationally efficient and helps prevent the vanishing gradient problem, which can occur in deeper networks. The function is defined as f(x) = max(0,x). One disadvantage of ReLU is that it can cause some neurons to "die" if they receive a negative input and never activate again.

*3.3.3.2 Loss function*

In a Convolutional Neural Network (CNN), a loss function is used to measure the difference between the predicted output and the actual output of the network during training. The goal of the network during training is to minimize this difference, or loss, so that it can accurately classify new input data.

The most commonly used loss function in CNNs is the cross-entropy loss function. It is a measure of the difference between the predicted probability distribution and the true probability distribution of the output. For example, in a binary classification problem, the cross-entropy loss function can be defined as:

* L = -(y\*log(p) + (1-y)\*log(1-p))
* where y is the true label (either 0 or 1), p is the predicted probability of the positive class (between 0 and 1), and log is the natural logarithm function.

The cross-entropy loss function penalizes the network heavily for incorrect predictions and gives a smaller penalty for correct predictions. During training, the network uses an optimization algorithm such as stochastic gradient descent to adjust its parameters in order to minimize the loss function.

Other types of loss functions can also be used in CNNs, such as mean squared error (MSE) loss and mean absolute error (MAE) loss, depending on the specific problem being solved.

*3.3.3.3 Backward propagation*

Backpropagation is a widely used algorithm for training neural networks, including Convolutional Neural Networks (CNNs). The main goal of backpropagation is to adjust the weights and biases of the network in order to minimize the loss function.

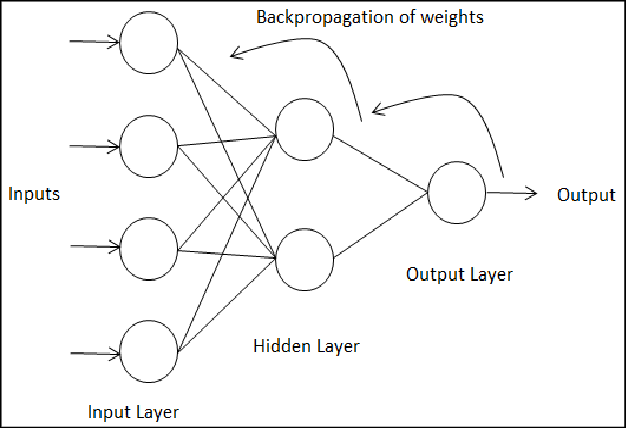
In a CNN, backpropagation works by computing the gradients of the loss function with respect to each parameter in the network, including the weights and biases of the convolutional and fully connected layers. The gradients are then used to update the parameters using an optimization algorithm such as stochastic gradient descent.

The backpropagation algorithm consists of two phases: the forward pass and the backward pass.

During the forward pass, the input data is passed through the layers of the network, and the output is computed. The output is then compared to the true output using the loss function, and the loss is computed.

During the backward pass, the gradients of the loss function with respect to the output of the last layer are computed. These gradients are then backpropagated through the network, and the gradients of the loss function with respect to the parameters of each layer are computed using the chain rule of calculus.

Finally, the parameters are updated using the computed gradients and the optimization algorithm. This process is repeated for multiple epochs until the loss function is minimized, and the network can accurately classify new input data.

****

Picture 3.6 Backpropagation

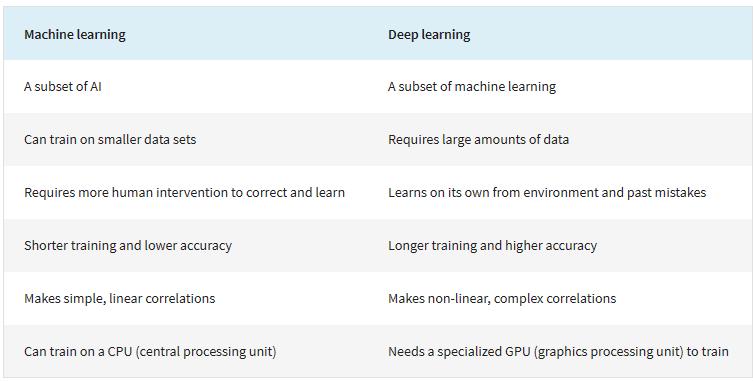
CHAPTER 4: DEEP LEARNING - MACHINE LEARNING

4.1 Introduction:

Machine learning is a subset of artificial intelligence that uses statistical techniques to enable machines to learn from data and make predictions or decisions based on that data. Deep learning is a subset of machine learning that uses neural networks with multiple layers to analyze and learn from data.

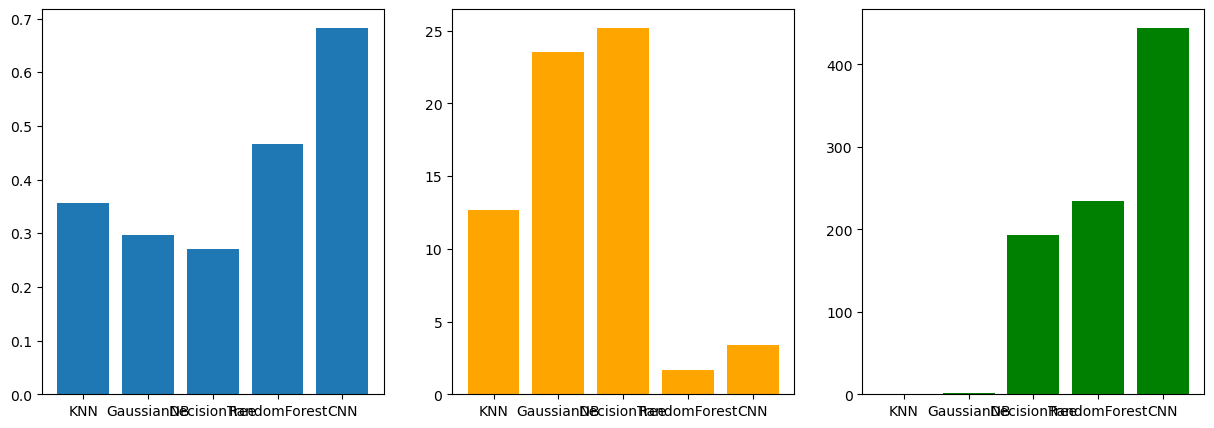
Machine learning models typically require a human expert to manually engineer the features used in the model, whereas deep learning models can learn features directly from raw data.

Deep learning models are typically more complex and computationally expensive than traditional machine learning models, but can often achieve better accuracy on complex tasks.



Picture 4.1 Brief comparison between Machine learning and Deep learning

4.2 Comparation



Picture 4.2 Accuracy - Lost - Train time of the traditional machine learning model vs CNN model

From the bar chart above, we see that the traditional machine learning models although training very fast, the prediction accuracy is low and the loss is quite high. Among the traditional machine learning models, the random forest gives the best results. However, the CNN deep learning model gives a much better result even though the loss is higher than the random forest. The CNN model used has not been tuned to give the best results. In terms of training time, decision tree and random forest are the two models with the highest time. However, the CNN model takes more time than all of them. As far as tracking during training is concerned, machine learning models use only the CPU, while deep learning models require a GPU for training. This indicates that deep learning models use much more computational power than machine learning

REFERENCES