Complete Cooperative Education Report

Car Classification Model

PHETPHUM RUNGNAPAVICHET

Department of Electronic Engineering

Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Year Study 2022

K. Sooksood

Cooperative Title: Car Classification Model

Student intern name: PHETPHUM RUNGNAPAVICHET

Faculty: Engineering **Department:** Electronics

Advisor name: Asst. Prof. Kriangkrai Sooksood

Mentor name: Mr. Pravich Jintanakorn

Company: CDG System Ltd.

ABSTRACT

In this project, I use machine learning knowledge applied to image classification to create a car classification model in which the primary data type that can classify the brand, model name, and year of production, including supplementary data that I have tried to use APIs to detect License plate and instant models from Open Source such as color classification. Furthermore, to be used for demonstration of the completion of the UI window.

Keyword: Classification

Acknowledgments

I would like to express my sincere thanks to my project mentor, Mr. Pravich Jintanakorn for his invaluable help and constant encouragement throughout the course of this research. I am most grateful for his teaching and advice, not only the research methodologies but also many other methodologies in life. I would not have achieved this far, and this thesis would not have been completed without all the support that I have always received from him.

In addition, I am grateful for the teachers of electronic engineering: Asst. Prof. Kriangkrai Sooksood. Thank you for giving advice on the project and giving me the opportunity to participate in cooperative education.

Finally, I most gratefully acknowledge my parents and my friends for all their support throughout the period of this research.

Phetphum Rungnapavichet

Contents

Table of Contents Page

Pictures of Contents

Name of Picture **Page**

Introduction

Background and significance

Machine learning (ML) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, agriculture, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks. A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers, but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory, and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. Some implementations of machine learning use data and neural networks in a way that mimics the working of a biological brain. In its application across business problems, machine learning is also referred to as predictive analytics.

Artificial intelligence

As a scientific endeavor, machine learning grew out of the quest for artificial intelligence. In the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what was then termed "neural networks"; these were mostly perceptron and other models that were later found to be reinventions of the generalized linear models of statistics. Probabilistic reasoning was also employed, especially in automated medical diagnosis.

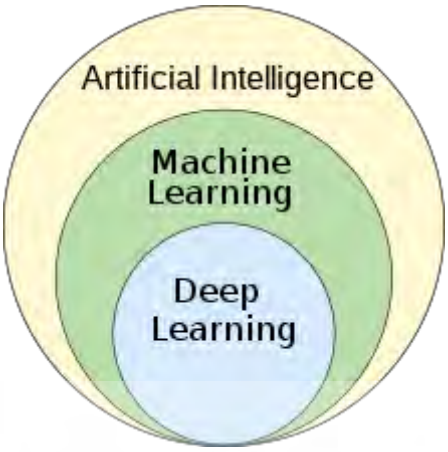


Figure 1 Machine learning subfield

However, an increasing emphasis on the logical, knowledge-based approach caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation. By 1980, expert systems had come to dominate AI, and statistics was out of favor. Work on symbolic/knowledge-based learning did continue within AI, leading to inductive logic programming, but the more statistical line of research was now outside the field of AI proper, in pattern recognition and information retrieval.[26]: 708–710, 755 Neural networks research had been abandoned by AI and computer science around the same time. This line, too, was continued outside the AI/CS field, as "connectionism", by researchers from other disciplines including Hopfield, Rumelhart, and Hinton. Their main success came in the mid-1980s with the reinvention of backpropagation.

Machine learning (ML), reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the symbolic approaches it had inherited from AI, and toward methods and models borrowed from statistics, fuzzy logic, and probability theory.

The difference between ML and AI is frequently misunderstood. ML learns and predicts based on passive observations, whereas AI implies an agent interacting with the

environment to learn and take actions that maximize its chance of successfully achieving its goals.

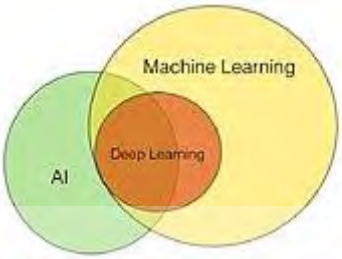


Figure 2 Part of machine learning as subfield of AI or part of AI as subfield of machine learning[

As of 2020, many sources continue to assert that ML remains a subfield of AI. Others have the view that not all ML is part of AI, but only an 'intelligent subset' of ML should be considered AI

Software

Software suites containing a variety of machine learning algorithms include the following:

Free and open-source software

Free and open-source software [edit]

- Caffe
- Deeplearning4j
- DeepSpeed
- ELKI
- Google JAX
- Infer.NET
- Keras
- Kubeflow
- LightGBM
- Mahout
- Mallet
- Microsoft Cognitive Toolkit
- ML.NET
- mlpack
- MLFlow
- MXNet
- Neural Lab
- OpenNN
- Orange
- pandas (software)
- ROOT (TMVA with ROOT)
- scikit-learn
- Shogun
- Spark MLlib
- SystemML
- TensorFlow
- Torch / PyTorch
- Weka / MOA
- XGBoost
- Yooreeka

Proprietary software

- Amazon Machine Learning
- Angoss KnowledgeSTUDIO
- Azure Machine Learning
- Ayasdi
- IBM Watson Studio
- Google Cloud Vertex AI
- Google Prediction API
- IBM SPSS Modeler
- KXEN Modeler
- LIONsolver
- Mathematica
- MATLAB
- Neural Designer
- NeuroSolutions
- Oracle Data Mining
- Oracle AI Platform Cloud Service
- PolyAnalyst
- RCASE
- SAS Enterprise Miner
- SequenceL
- Splunk
- STATISTICA Data Miner

Research objectives

This project aims to study the application of machine learning knowledge to work. In this case, recognizing and classifying images will be mainly applied to classify car models, brands, and years of production.

In addition to the main tasks mentioned above, there are studies and simulations of making UI windows, including fetching. Ready-made models and APIs from web applications and various open sources to add to it to make the project complete.

Scope of research

The scope of this project will be working from gathering datasets of the car brand in each model is then used to classify data types and create a model by making a feature that employs images in the model to classify the name of the car model can come out. Selecting the model type must be an empty model from open source and choose the most suitable one.

Figure 3 OYOTA YARIS 2022 Urban White Platinum

Methods of conducting research

Machine Learning Lifecycle.

Machine learning has given the computer systems the abilities to automatically learn without being explicitly programmed. But how does a machine learning system work? So, it can be described using the life cycle of machine learning. Machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

Machine learning life cycle involves seven major steps, which are given below:

- Gathering Data
- Data preparation
- Data Wrangling
- Analyze Data
- Train the model
- Test the model
- Deployment

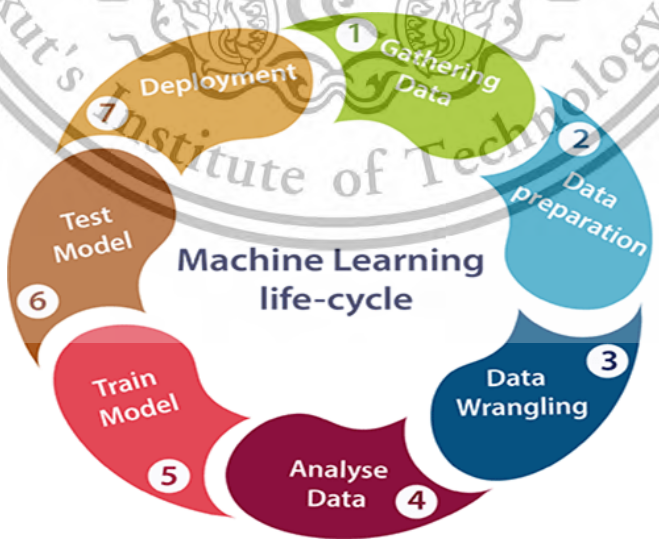


Figure 4 Machine Learning Life-cycle

The most important thing in the complete process is to understand the problem and to know the purpose of the problem. Therefore, before starting the life cycle, we need to understand the problem because the good result depends on the better understanding of the problem.

In the complete life cycle process, to solve a problem, we create a machine learning system called "model", and this model is created by providing "training". But to train a model, we need data, hence, life cycle starts by collecting data.

1.) Gathering Data:

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from various sources such as **files, database, internet,** or **mobile devices**. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction.

Figure 5 Dataset Honda Civic 2023

Figure 7 Dataset Honda Civic 2023

Figure 6 Dataset Honda Civic 2023

Example: Dataset of Honda Civic 2023

This step includes the below tasks:

- Identify various data sources
- Collect data
- Integrate the data obtained from different sources

By performing the above task, we get a coherent set of data, also called as a dataset. It will be used in further steps.

2.) Data preparation

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.

In this step, first, we put all data together, and then randomize the ordering of data.

This step can be further divided into two processes:

Data exploration:

It is used to understand the nature of data that we must work with. We need to understand the characteristics, format, and quality of data. A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.

Data pre-processing:

Now the next step is preprocessing of data for its analysis.

In this project, we will focus on collecting image data of the car in the vehicle's front grille that is suitable for deployment in the position where the camera faces the front of the car, for example, at the entrance of a department store, condo entrances, and entrances to various places.

3.) Data Wrangling

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

- Missing Values
- Duplicate data
- Invalid data
- Noise

So, we use various filtering techniques to clean the data.

It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

Figure 9 Lebel and crop to clean a data

Figure 8 Cleaned Data

4.) Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- Selection of analytical techniques
- Building models
- Review the result

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as **Classification, Regression, Cluster analysis, Association**, etc. then build the model using prepared data, and evaluate the model.

Hence, in this step, we take the data and use machine learning algorithms to build the model.

To test the model's algorithms, I have divided the data types into three types used to test the performance between trains. First, we can check the model's accuracy during training by using the unaware data of model training, including the data loss during the procession, such as the Entropy loss.

Three types of datasets:

- Training Set is used for input to use training
- Validation Set is used to test for Metrics after training is completed on how well the model works. And after each tuning, which model works better?
- Test Set is used for testing after getting the best model. We can see how well the model performs with never-before-seen data

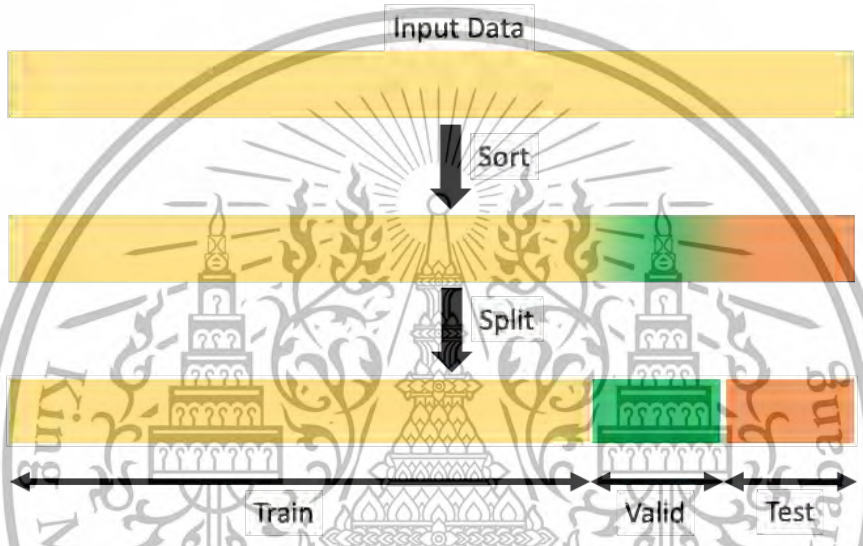


Figure 11 Three types of datasets

1 dataset is including with:

80 % for training set

10 % for validation set

10 % for testing set

5.) Train Model

Now this step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and features.

Cross Entropy Loss

Cross entropy loss is a metric used to measure how well a classification model in machine learning performs. The loss (or error) is measured as a number between 0 and 1, with 0 being a perfect model. The goal is generally to get your model as close to 0 as possible.

Cross entropy loss is often considered interchangeable with logistic loss (or log loss, and sometimes referred to as binary cross entropy loss) but this isn't always correct.

Cross entropy loss measures the difference between the discovered probability distribution of a machine learning classification model and the predicted distribution. All possible values for the prediction are stored so, for example, if you were looking for the odds in a coin toss it would store that information at 0.5 and 0.5 (heads and tails).

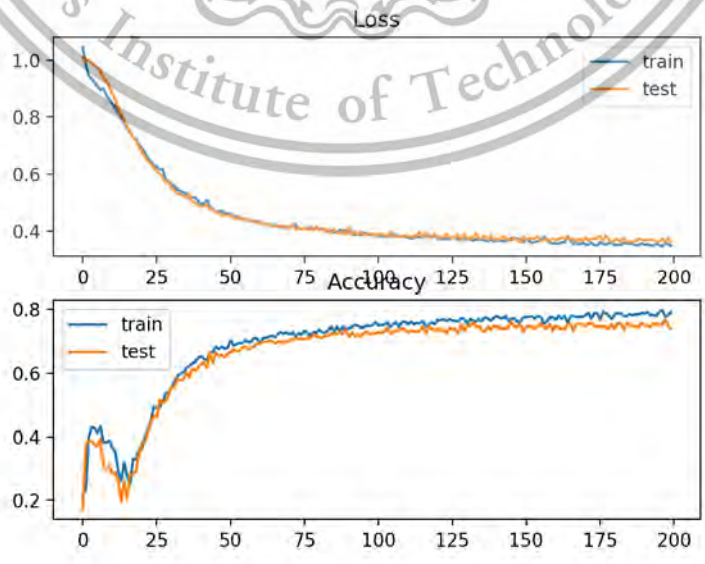


Figure 12 Entropy loss and accuracy

Binary cross entropy loss, on the other hand, store only one value. That means it would store only 0.5, with the other 0.5 assumed in a different problem, if the first probability was 0.7 it would assume the other was 0.3). It also uses a logarithm (thus "log loss").

It is for this reason that binary cross entropy loss (or log loss) is used in scenarios where there are only two possible outcomes, though it's easy see where it would fail immediately if there were three or more. That's where cross entropy loss is often used: in models where there are three or more classification possibilities.

The Theory Behind Cross Entropy Loss

Let's start from the basics. In deep learning, we typically use gradient-based optimization strategy to train a model (say f(x)) using some loss function l (f (x_i), y_i) where (x_i, y_i) are some input-output pair. A loss function is used to help the model determine how "wrong" it is and, based on that "wrongness," improve itself. It's a measure of error. Our goal throughout training is to minimize this error/loss.

The role of a loss function is an important one. If doesn't penalize wrong output appropriately to its magnitude it can delay convergence and affect learning.

There's a learning paradigm called Maximum Likelihood Estimation (MLE) which trains the model to estimate its parameters to learn the underlying data distribution. Thus, we use a loss function to evaluate the how well the model fits the data distribution.

Using cross entropy, we can measure the error (or difference) between two probability distributions.

For example, in the case of Binary Classification, cross-entropy is given by:

$$l = -(y\,log(p) + (1-y)\,log(1-p))$$

where:

- p is the predicted probability, and
- y is the indicator (0 or 1 in the case of binary classification)

Let's walk through what happens for a particular data point. Let's say the correct indicator is i.e., y = 1. In this case,

$$l = -(\ 1 \times log(p) + (1 - 1)\ log(1 - p)\ )$$

$$l = -(\ 1 \times log(p)\ )$$

The value of loss l thus depends on the probability p. Therefore, our loss function will reward the model for giving a correct prediction (high value of p) with a low loss. However, if the probability is lower, the value of the error will be high (bigger negative value) and therefore it penalizes the model for a wrong outcome.

A simple extension to a multi-Classification (say N classes) problem exists as follows:

$$\sum_{c=1}^{N} y_c log(p_c)$$

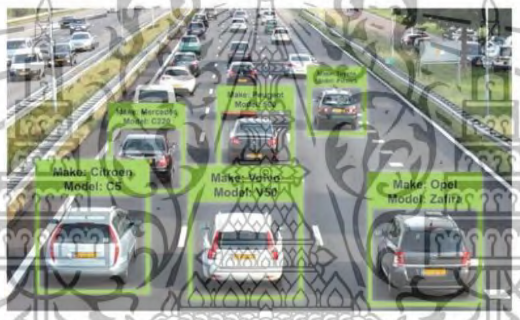
6.) Test Model

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

7.) Deployment

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.



If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is like making the final report for a project.

Expected Benefits

- The Model's accuracy is greater than 90%
- High performance in real time field work
- Save energy and be stable
- Hastiness support in training part

Concepts, Theories and Related Research

Related theories

Data mining

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on known properties learned from the training data, data mining focuses on the discovery of (previously) unknown properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

Optimization

Machine learning also has intimate ties to optimization: many learning problems are formulated as minimization of some loss function on a training set of examples. Loss functions express the discrepancy between the predictions of the model being trained and the actual problem instances (for example, in classification, one wants to assign a label to instances, and models are trained to correctly predict the pre-assigned labels of a set of examples).

Computational learning theory and Statistical learning theory

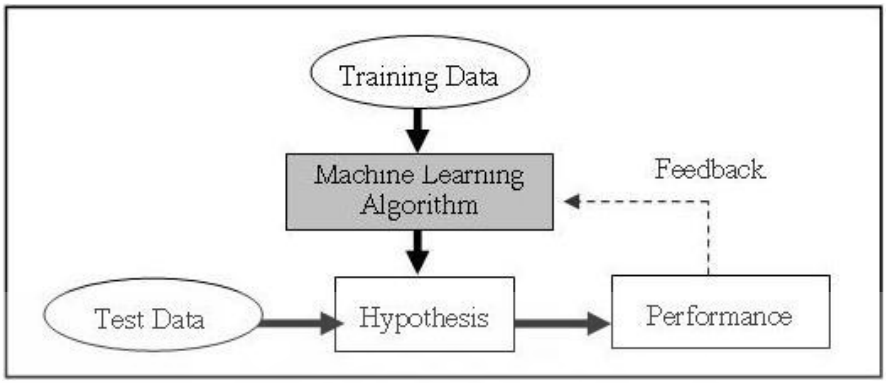


Figure 13 The one part of the machine learning life cycle

A core objective of a learner is to generalize from its experience. Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner must build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory via the Probably Approximately Correct Learning (PAC) model. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has under fitted the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to overfitting and generalization will be poorer.

In addition to performance bounds, learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results: Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

Models

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

Artificial neural networks

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different

kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

The original goal of the ANN approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Artificial neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

Deep learning consists of multiple hidden layers in an artificial neural network. This

Figure 14 An artificial neural network is an interconnected group of nodes.

approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech recognition.

Decision trees learning

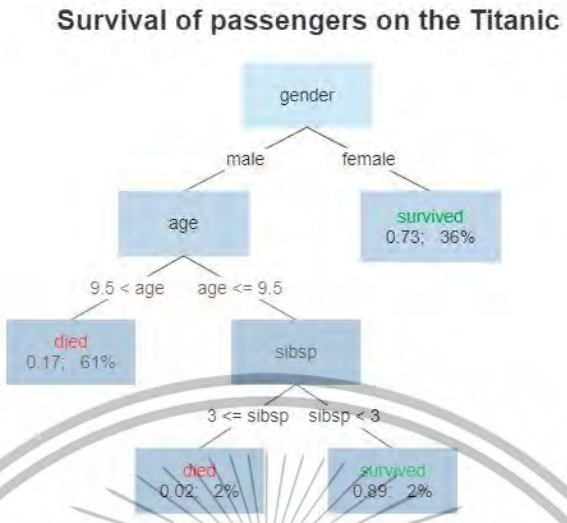


Figure 15 A decision tree showing survival probability of passengers on the Titanic

Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining, and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels, and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to represent decisions and decision making visually and explicitly. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision-making.

Support-vector machines

Support-vector machines (SVMs), also known as support-vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one

category. An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Regression analysis

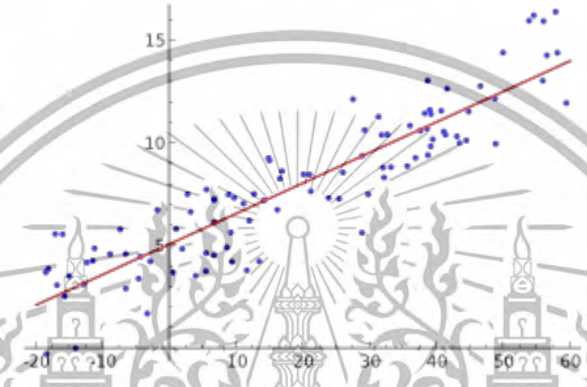


Figure 16 Illustration of linear regression on a data set

Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trendline fitting in Microsoft Excel), logistic regression (often used in statistical classification) or even kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higher-dimensional space.

Gaussian processes

A Gaussian process is a stochastic process in which every finite collection of the random variables in the process has a multivariate normal distribution, and it relies on a pre-defined covariance function, or kernel, that models how pairs of points relate to each other depending on their locations.

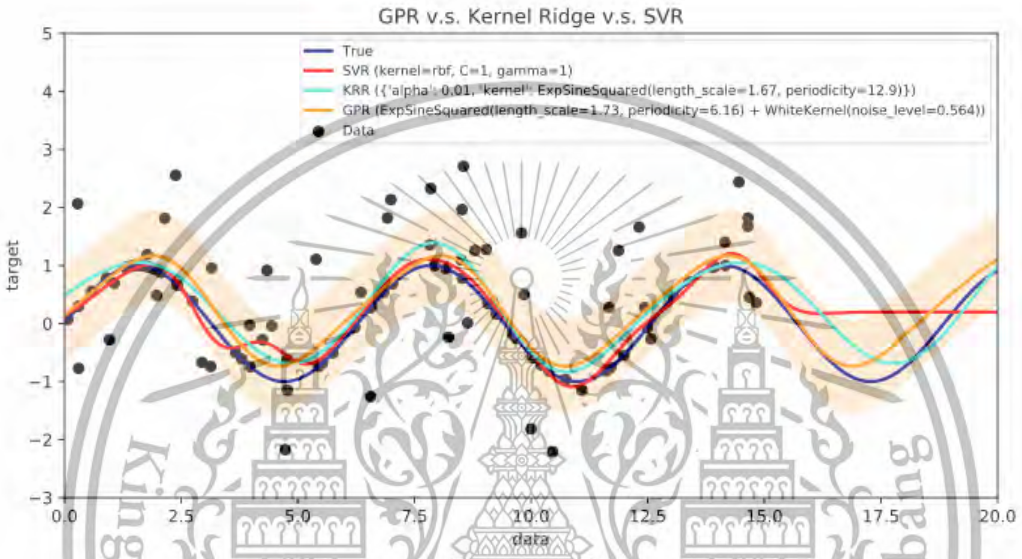


Figure 17 An example of Gaussian Process Regression (prediction) compared with other regression models.

Given a set of observed points, or input–output examples, the distribution of the (unobserved) output of a new point as function of its input data, can be directly computed by looking like the observed points and the covariances between those points and the new, unobserved point.

Gaussian processes are popular surrogate models in Bayesian optimization used to do hyperparameter optimization.

Limitations

Although machine learning has been transformative in some fields, machine-learning programs often fail to deliver the expected results. Reasons for this are numerous: lack of (suitable) data, lack of access to the data, data bias, privacy problems, badly chosen tasks, and algorithms, wrong tools and people, lack of resources, and evaluation problems. In 2018, a self-driving car from Uber failed to detect a pedestrian who was killed after a collision. Attempts to use machine learning in healthcare with the IBM Watson system failed to deliver even after years of time and billions of dollars invested. Machine learning has been used as a strategy to update the evidence related to a systematic review and increase the reviewer burden related to the growth of biomedical literature. While it has improved with training sets, it has not yet developed sufficiently to reduce the workload burden without limiting the necessary sensitivity for the findings research themselves.

Overfitting

Settling on a bad, overly complex theory gerrymandered to fit all the past training data is known as overfitting. Many systems attempt to reduce overfitting by rewarding a

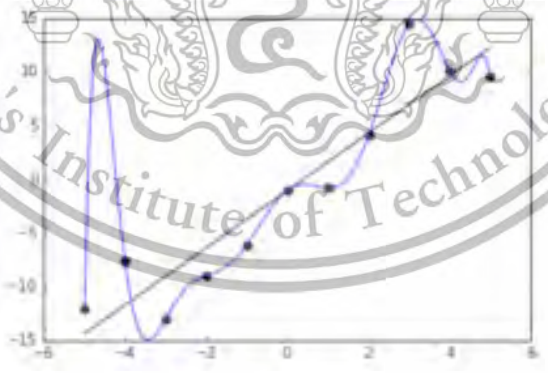


Figure 18 The blue line could be an example of overfitting a linear function due to random noise.

theory in accordance with how well it fits the data but penalizing the theory in accordance with how complex the theory is.

Model assessments

Classification of machine learning models can be validated by accuracy estimation techniques like the holdout method, which splits the data in a training and test set (conventionally 2/3 training set and 1/3 test set designation) and evaluates the performance of the training model on the test set. In comparison, the K-fold-cross-validation method randomly partitions the data into K subsets and then K experiments are performed each respectively considering 1 subset for evaluation and the remaining K-1 subsets for training the model. In addition to the holdout and cross-validation methods, bootstrap, which samples n instances with replacement from the dataset, can be used to assess model accuracy.

In addition to overall accuracy, investigators frequently report sensitivity and specificity meaning True Positive Rate (TPR) and True Negative Rate (TNR) respectively. Similarly, investigators sometimes report the false positive rate (FPR) as well as the false negative rate (FNR). However, these rates are ratios that fail to reveal their numerators and denominators. The total operating characteristic (TOC) is an effective method to express a model's diagnostic ability. TOC shows the numerators and denominators of the previously mentioned rates; thus, TOC provides more information than the commonly used receiver operating characteristic (ROC) and ROC's associated area under the curve (AUC).

Hardware

Since the 2010s, advances in both machine learning algorithms and computer hardware have led to more efficient methods for training deep neural networks (a particular narrow subdomain of machine learning) that contain many layers of non-linear hidden units. By 2019, graphic processing units (GPUs), often with AI-specific enhancements, had displaced CPUs as the dominant method of training large-scale commercial cloud AI. OpenAI estimated the hardware computing used in the largest deep

learning projects from AlexNet (2012) to AlphaZero (2017) and found a 300,000-fold increase in the amount of compute required, with a doubling-time trendline of 3.4 months.

Embedded Machine Learning

Embedded Machine Learning is a sub-field of machine learning, where the machine learning model is run on embedded systems with limited computing resources such as wearable computers, edge devices and microcontrollers. Running machine learning model in embedded devices removes the need for transferring and storing data on cloud servers for further processing, henceforth, reducing data breaches and privacy leaks happening because of transferring data, and minimizes theft of intellectual properties, personal data, and business secrets. Embedded Machine Learning could be applied through several techniques including hardware acceleration, using approximate computing, optimization of machine learning models and many more.

Related research

Machine Learning on Facial Recognition

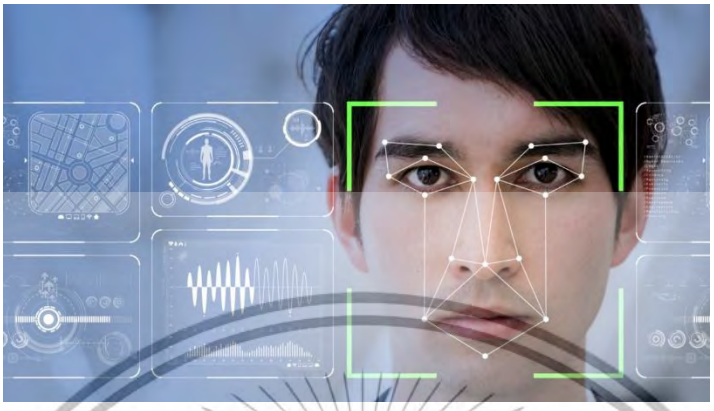


Figure 19 Facial recognition

In machine learning, a Convolutional Neural Network (CNN or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. They have applications in image recognition (facial recognition) and video analysis, recommender systems and natural language processing. Here, facial recognition would be analyzed.

Facial Recognition

Facial recognition is a biometric software application capable of uniquely identifying or verifying a person by comparing and analyzing patterns based on the person's facial contours. Facial recognition is mostly used for security purposes, though there is increasing interest in other areas of use. In fact, facial recognition technology has received significant attention as it has potential for a wide range of application related to law enforcement as well as other enterprises.

There are different facial recognition techniques in use, such as the generalized matching face detection method and the adaptive regional blend matching method. Most facial recognition systems function based on the different nodal points on a human face. The values measured against the variable associated with points of a person's face help in uniquely identifying or verifying the person. With this technique, applications can use

data captured from faces and can accurately and quickly identify target individuals. Facial recognition techniques are quickly evolving with new approaches such as 3-D modeling, helping to overcome issues with existing techniques.

There are many advantages associated with facial recognition. Compared to other biometric techniques, facial recognition is of a non-contact nature. Face images can be captured from a distance and can be analyzed without ever requiring any interaction with the user/person. As a result, no user can successfully imitate another person. Facial recognition can serve as an excellent security measure for time tracking and attendance. Facial recognition is also cheap technology as there is less processing involved, like in other biometric techniques.

How to Use Machine Learning on Facial Recognition

The approach we are going to use for facial recognition is very straight forward, but you could check this out in the case of a very complicated problem. Let's learn how modern face recognition works!

The goal here is to get deep neural network to output a person's face with identification. This means that the neural network needs to be trained to automatically identify different features of a face and calculate numbers based on that. The output of the neural network can be thought of as an identifier for a particular person's face — if you pass in different images of the same person, the output of the neural network will be very similar or close, whereas if you pass in images of a different person, the output will be very different.

Machine learning has solved many problems by choosing one machine learning algorithm, feeding in data, and getting the result. We do not need to build our own neural network. We have access to a trained model dlib that can be used. It does exactly what we need it to do — outputs a bunch of numbers (face encodings) when we pass in the image of someone's face; comparing face encodings of faces from different images will tell us if someone's face matches with anyone we have images of.

However, a facial recognition library called face recognition is much easier to install and use. This would be use at some point here.

These are the steps we will be taking:

Detect: Find faces in pictures

Landmark: Find and manipulate facial features in pictures

Compare: Identify faces in pictures

Steps on Facial Recognition

1. Preparing images

Figure 20 Preparing image to use in face recognition

2. Landmark

Manipulations are done in the image by getting the locations and outlines of each person's eyes, nose, mouth, and chin.

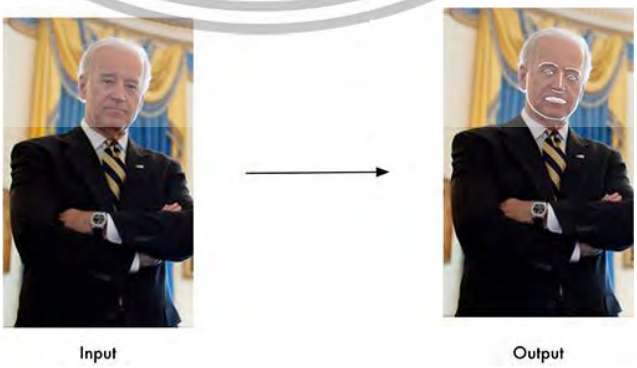


Figure 21 loacte and outline face

```
import face_recognition

image = face_recognition.load_image_file("your_file.jpg")

face_landmarks_list =
face_recognition.face_landmarks(image)
```

Getting output from facial recognition is a little bit tedious and facial recognition features should be considered important to analyze.

3. Identify faces in the pictures

This is done by recognizing the faces of who appear in a photo.

Figure 22 face recognition

```
biden_encoding =
face_recognition.face_encodings(known_image)[0]

unknown_encoding =
face_recognition.face_encodings(unknown_image)[0]

results = face_recognition.compare_faces([biden_encoding],
unknown_encoding)
```

This outputs “Joe Biden” as the person in the photo. The approach could also be used for several images in one photo, and this is done by picking out faces appearing in the photo. Then the steps listed above are used to analyze the input of (n) to give outputs of (n).

We have been able to detect who the image represents in the photo using a simple approach by detecting, manipulating, and identifying the contours of the face.

This clearly shows how machine learning has rapidly taken charge in the world of artificial intelligence. A day-to-day sample is Facebook, which automatically tags people found in a photo without tagging them manually. This was not achieved sometimes ago rather people in photos were tagged with suggestions.

Research Methods

1.) Create Model

PyTorch Model

Figure 23 PyTorch

PyTorch is a machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, originally developed by Meta AI and now part of the Linux Foundation umbrella. It is free and open-source software released under the modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface.

PyTorch Tensor

PyTorch defines a class called Tensor (torch.Tensor) to store and operate on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are like NumPy Arrays, but can also be operated on a CUDA-capable NVIDIA GPU. PyTorch has also been developing support for other GPU platforms, for example, AMD's ROCm and Apple's Metal Framework.

Note that the term "tensor" here does not carry the same meaning as tensor in mathematics or physics. The meaning of the word in those areas, that is, a certain kind of object in linear algebra, is only tangentially related to the one in Machine Learning.

Building Resnet-34 model using PyTorch

Deep learning has evolved a lot in recent years, and we all are excited to build deeper architecture networks to gain more accuracies for our models. These techniques

are widely tried for Image related works like classification, clustering, or synthesis. Going deep may look cool but won't help as neural networks face an issue called degradation.

The accuracy is affected to a great extent here. This also leads to a problem called Vanishing gradient descend. This won't allow us to properly update the weights during the backpropagation step. During the backpropagating step, we use the chain rule, the derivatives of each layer as we go down the network get multiplied.

If we use deeper layers and have hidden layers like sigmoid, the derivatives are scaled down below 0.25 within each layer. So, when a lot of layers derivatives get multiplied the gradient decreases exponentially and we get a very small value which is useless for the gradient calculation. This led to the making of Resnet by Microsoft Research which used skipped connections to avoid degradation. In this article, we will discuss an implementation of 34 layered ResNet architecture using the Pytorch framework in Python.

Architecture of Resnet-34

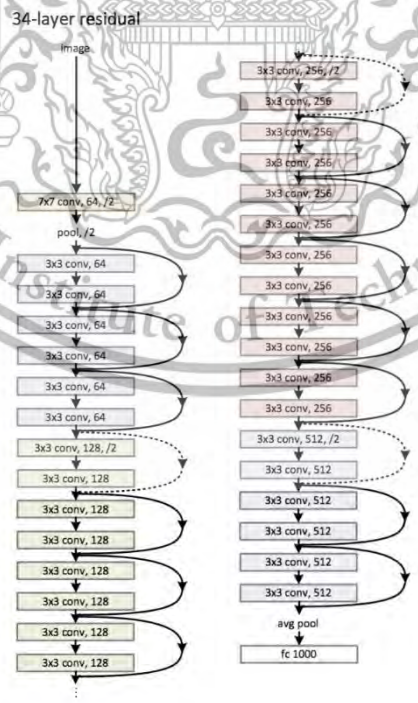


Figure 24 Architecture of Resnet-34

Initially, we have a convolutional layer that has 64 filters with a kernel size of 7×7 this is the first convolution, then followed by a max-pooling layer. We have the stride specified as 2 in both cases. Next, in conv2_x we have the pooling layer and the following convolution layers. These layers are normally grouped in pairs because of the way the residuals are connected (the arrows show that are jumping every two layers).

Here, we have the 2 layers of which have a kernel_sizeof 3×3, num_filters of 64, and all these are repeated x3, which corresponds to the layers between the pool,/2 and the filter 128 ones, 6 layers in total (one pair times 3). These 2 layers are kernel_size of 3×3, num_filters are 128, and these are also got repeated but on these times 4. This continues until the avg_pooling and the softmax function. Each time the number of filters gets doubled we can see the first layer specifies num_filters/2.

Calling Model

```
model = models.resnet34(pretrained=True)
```

Figure 25 Calling Model Code

Training Setup

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
lrscheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='max', patience=3, threshold = 0.9)
```

Figure 26 Set parameter to set up training

This criterion computes the cross-entropy loss between input and target. This is particularly useful when you have an unbalanced training set.

To construct an Optimizer, we must give it an inerrable containing the parameters (all should be Variable s) to optimize. Then, we can specify optimizer-specific options such as the learning rate, weight decay, etc.

Allows reducing the learning rate based on specific validation metrics dynamically.

2.) Collect Dataset

This step will collect data for each car brand and model name. The data type that we will use to train the model is a .jpg file.

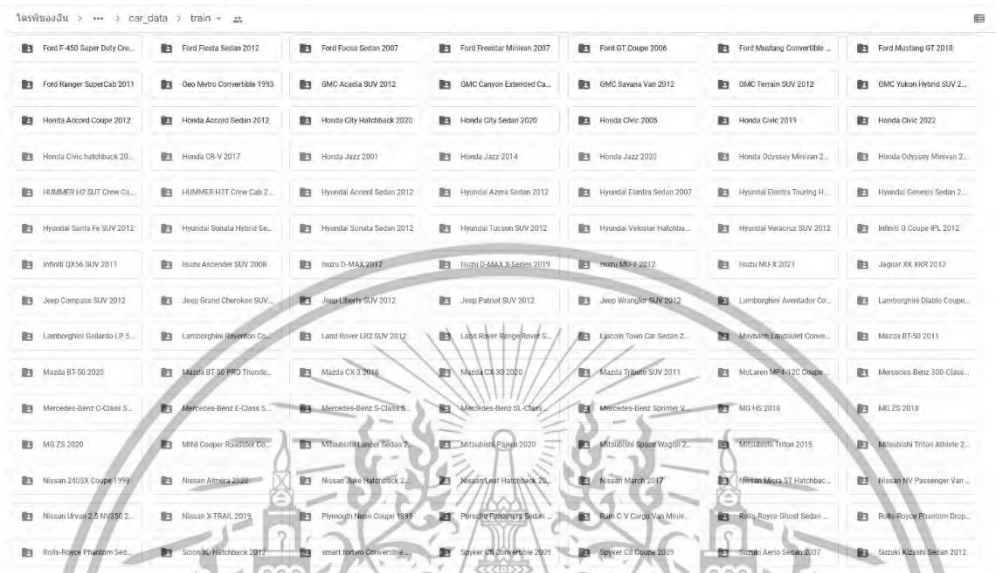
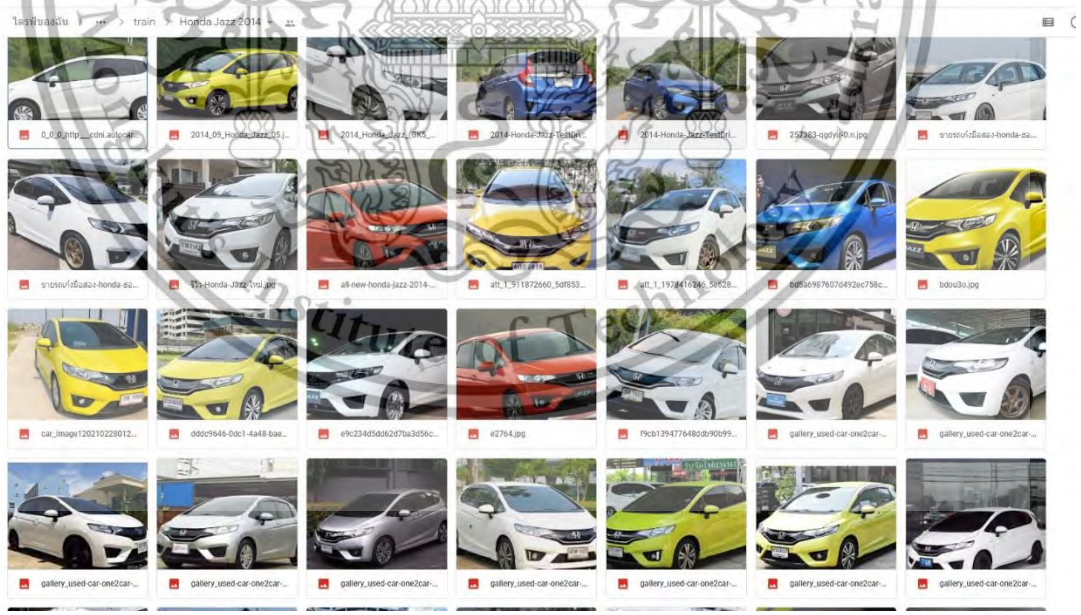


Figure 27 List of Car Model Name folder

Each folder of car model is including with datasets pictures (.jpg file)



3.) Categorize Dataset

This step divides the data into three categories: valid, train and test.

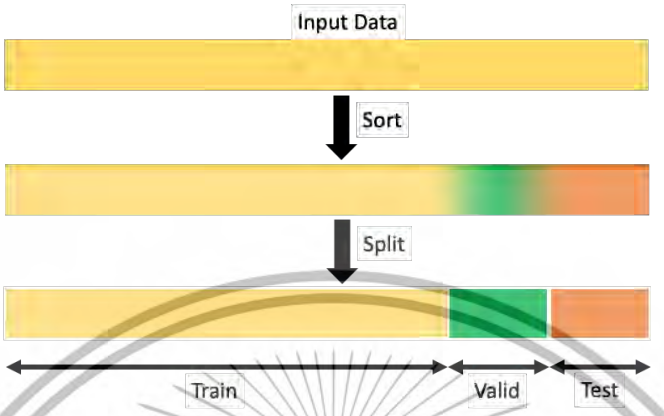


Figure 28 Catagorize Dataset

1.) Training Set is used for input to use training

2.) Validation Set is used to test for Metrics after training is completed on how well the model works. And after each tuning, which model works better?

3.) Test Set is used for testing after getting the best model. We can see how well the model performs with never-before-seen data

Figure 29 3 types of datasets

We use this 3-section data to Train, Validate, Tune Hyperparameter, Train, Validate, Tune Hyperparameter, Train, Validate, Tune Hyperparameter to choose the most accurate model with the Validation Set. Finally, we will test it with a test set representing real-world data to see how well it performs when we release the system.

4.) Transform Dataset

This step involves modifying random rotation and flipping data to build a more robust model and resizing the dataset to the typical values used to train the model.

```
[ ]  # Training transform includes random rotation and flip to build a more robust model
train_transforms = transforms.Compose([transforms.Resize((244,244)),
                                       transforms.RandomRotation(30),
                                       transforms.RandomHorizontalFlip(),
                                       transforms.ToTensor(),
                                       transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
```

Figure 30 Modify Dataset

Deep Neural Networks are based mainly on Mini-Batch Stochastic Gradient Optimization algorithms. The number of data samples we feed the model at one time, or 1 Mini-Batch is Hyperparameter. One important thing we must adjust in different libraries is Batch Size in cases with significant data, such as high-resolution images. Due to the limitation of GPU memory size, we must reduce Batch Size to train the model without Out of Memory Error.

```
# Using the image datasets and the trainforms, define the dataloaders
# The trainloader will have shuffle=True so that the order of the images do not affect the model
trainloader = torch.utils.data.DataLoader(train_data, batch_size=128, shuffle=True)
testloader = torch.utils.data.DataLoader(test_data, batch_size=32, shuffle=True)
validloader = torch.utils.data.DataLoader(valid_data, batch_size=32, shuffle=True)
```

Figure 31 Prepare data and set batch size (Standard Value)

5.) Train Model

- Implement a function for the validation pass
- Change model to work with cuda (Runtime type: GPU)
- Iterate over data from validloader
- Change images and labels to work with cuda
- Forward pass image though model for prediction
- Calculate loss
- Calculate probability
- Calculate accuracy

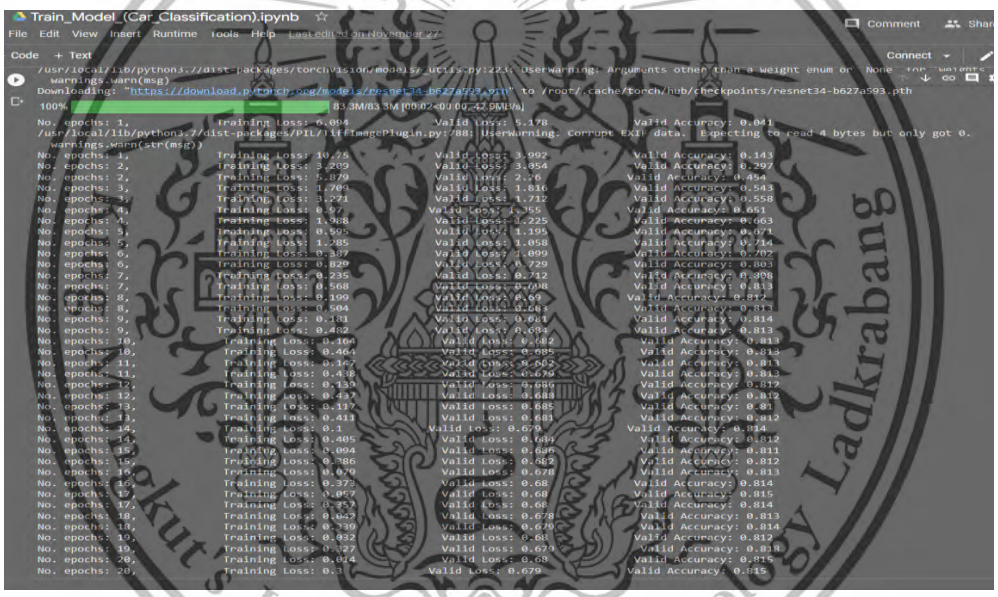


Figure 32 Result of each training epochs. (Model: ResNet34)

6.) Test Model

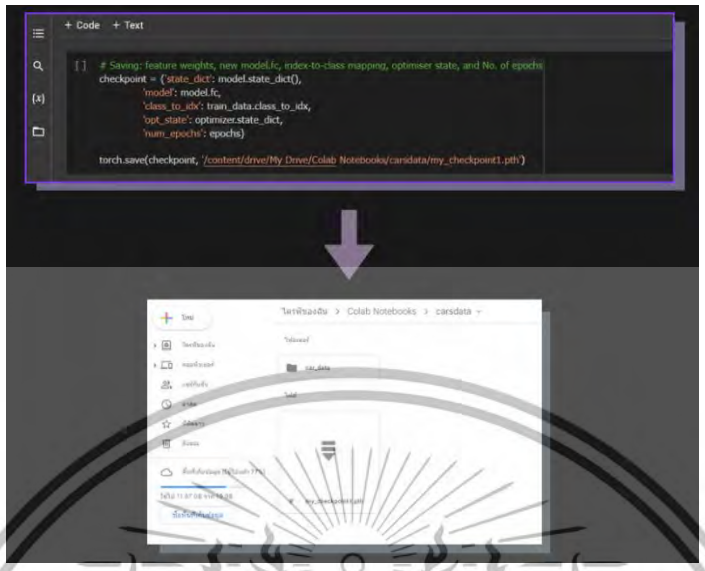


Figure 33 Save Model Method

To test the model's accuracy, I will use the sample image in the test image folder to let the model predict the sample image. Which model does it match?

In this method, I use matplotlib to show that the model can predict the top 5 car model name to check which cars are similar; then, we can add the more characteristic datasets to those similar cars. This method can reduce the error in predicting.

Testing Result:

The y-axis shows the car model name that the model can predict the closest model's name from the test image, and the value on the x-axis is the ratio that compares only the five cars, which is the most comparable prediction.

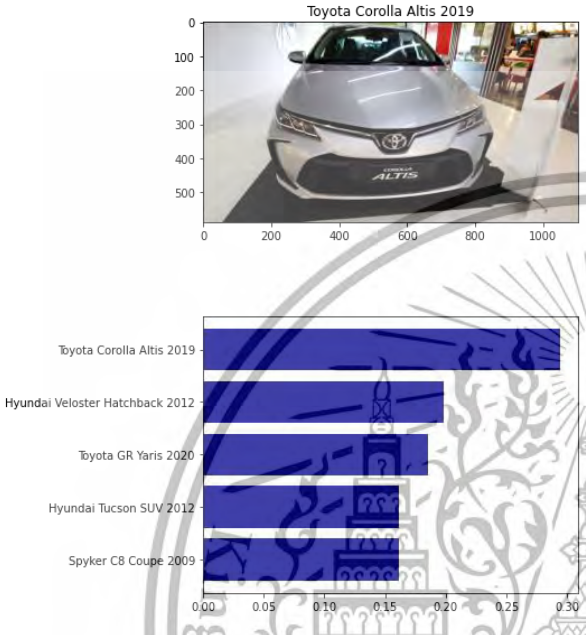


Figure 35 Testing Image1

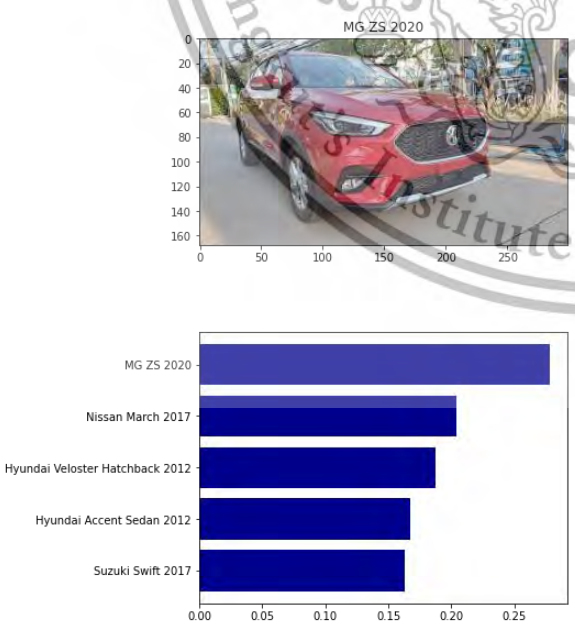


Figure 34 Testing Image 3

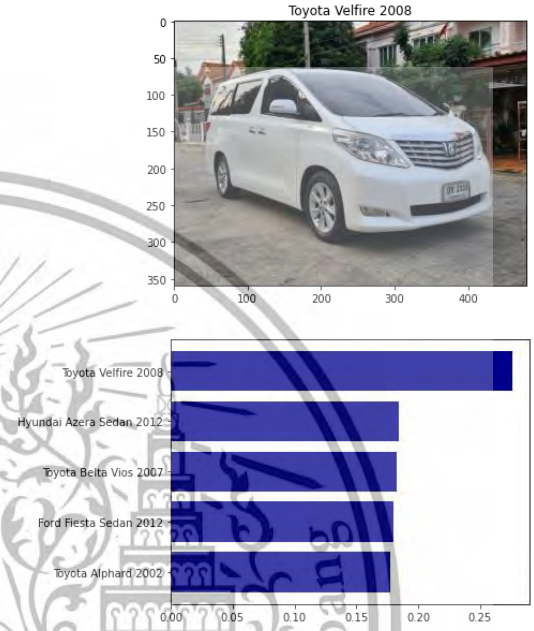


Figure 37 Testing Image 2

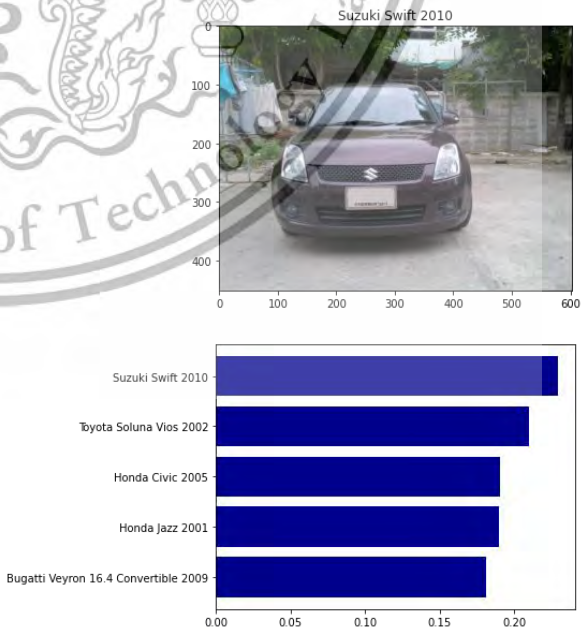


Figure 36 Testing Image 4

7.) Deployment

At this stage, I have deployed it with UI window to the video file and the result of the car classification model, as well as using it with other APIs and instant models such as color classification, License Plate Detection and time recording to create a model to complete the simulation program.

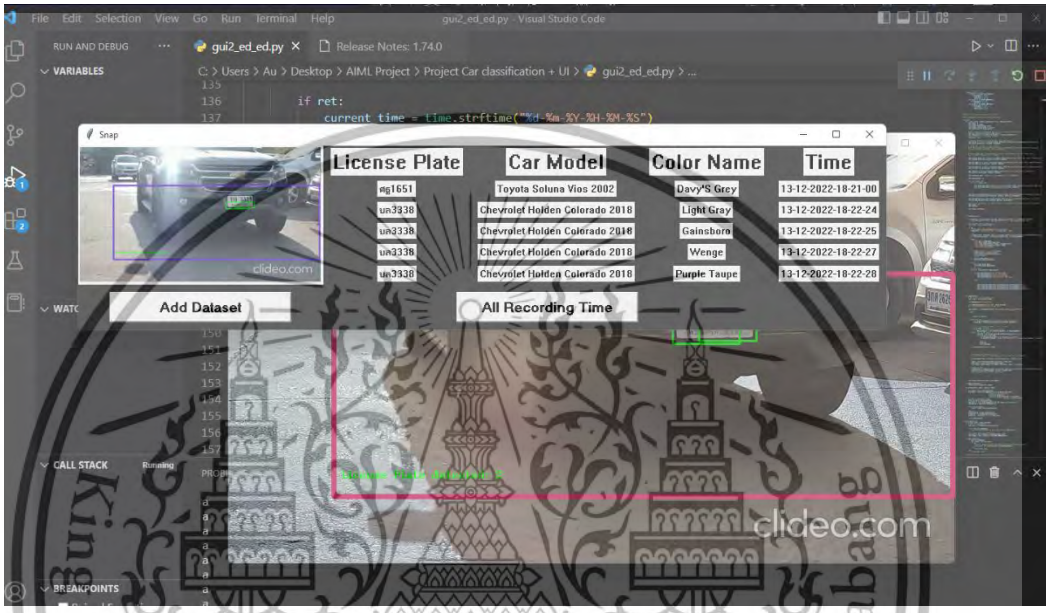


Figure 38 Deployment

Research Results and Recommendations

The latest version of this model has test accuracy 75.965%

```
No. epochs: 19,          Training Loss: 0.032          Valid Loss: 0.68          Valid Accuracy: 0.812
No. epochs: 19,          Training Loss: 0.327          Valid Loss: 0.679          Valid Accuracy: 0.818
No. epochs: 20,          Training Loss: 0.014          Valid Loss: 0.68          Valid Accuracy: 0.815
No. epochs: 20,          Training Loss: 0.3          Valid Loss: 0.679          Valid Accuracy: 0.815
No. epochs: 20,          Training Loss: 0.605          Valid Loss: 0.684          Valid Accuracy: 0.813
Test accuracy of model: 75.965%
```

After the accuracy, the result is out. Therefore, additional hardware factor needs to be studied that will help Higher accuracy improve the model's efficiency.

Hardware

When dealing with machine learning, and especially when dealing with deep learning and neural networks, it is preferable to use a graphics card to handle the processing, rather than the CPU. Even a very basic GPU is going to outperform a CPU when it comes to neural networks.

A CPU (Central Processing Unit) is the workhorse of your computer, and importantly is very flexible. It can deal with instructions from a wide range of programs and hardware, and it can process them very quickly. To excel in this multitasking environment a CPU has a small number of flexible and fast processing units (also called cores).

A GPU (Graphics Processing Unit) is a little bit more specialized, and not as flexible when it comes to multitasking. It is designed to perform lots of complex mathematical calculations in parallel, which increases throughput. This is achieved by having a higher number of simpler cores, sometimes thousands, so that many calculations can be processed all at once.

This requirement of multiple calculations being carried out in parallel is a perfect fit for:

- graphics rendering — moving graphical objects need their trajectories calculated constantly, and this requires a large amount of constant repeat parallel mathematical calculations.
- machine and deep learning — large amounts of matrix/tensor calculations, which with a GPU can be processed in parallel.
- any type of mathematical calculation that can be split to run in parallel.

Nvidia's own blog:

CPU	GPU
Central Processing Unit	Graphics Processing Unit
Several cores	Many cores
Low latency	High throughput
Good for serial processing	Good for parallel processing
Can do a handful of operations at once	Can do thousands of operations at once

GPU Features

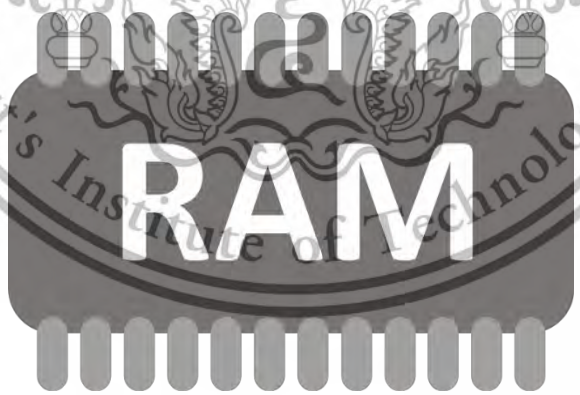
Picking out a GPU that will fit your budget, and is also capable of completing the machine learning tasks you want, basically comes down to a balance of four main factors:

- How much RAM does the GPU have?
- How many CUDA and/or Tensor cores does the GPU have?
- What chip architecture does the card use?
- What are your power draw requirements (if any)?

GPU RAM

It really comes down to what are modelling, and how big those models are. For example, if it is dealing with images, videos, or audio, then we are going to be dealing with quite a large amount of data, and GPU RAM will be an extremely important consideration.

There are always ways to get around running out of memory (e.g., reducing the batch size). However, it wants to limit the amount of time we have to spend messing about with code just to get around memory requirements, so a good balance for requirements is essential.



4GB — The absolute minimum would consider, this will work well in most cases if it is not dealing with overly complicated models, or large amounts of images, videos, or audio. Great if you are just starting out and want to experiment without breaking the bank. The improvements over a CPU will still be night and day.

8GB — this is a good middle ground. We can get most tasks done without hitting RAM limits, but you will have problems with more complicated models with images, video, or audio.

12GB — This would describe as optimal without being ridiculous. You can deal with most larger models, even those that deal with images, video, or audio.

12GB+ — The more the better, we will be able to handle larger datasets and bigger batch sizes. However, beyond 12GB is where the prices really start to ramp up.

According to the latest Accuracy Test result, the image data train needs to increase the batch size and the number of times to train, including other factors such as the specification of the leading hardware, such as the graphic card. For example, a graphic card with more memory and cores can make parameter and batch size adjustments higher in accuracy and shorten the train time.

Summary of Research

Machine learning has many applications. Depending on the type and size of the work, we will decide which model is suitable for use. We must compare the details from deep learning used in the calculations to the deployment of each situation so that the work is appropriate and stable. It also needs to look at the hardware factors used to be aware of the time it takes to train data into the model and can predict the time. All of which will make it work more efficiently, faster, and more suitable.

Reference

1. Saurav Maheshkar. (15 Mar 2022). What Is Cross Entropy Loss? A Tutorial with Code. Retrieved 4 Dec 2022. From https://wandb.ai/sauravmaheshkar/cross-entropy/reports/What-Is-Cross-Entropy-Loss-A-Tutorial-With-Code--VmlldzoxMDA5NTMx
2. Mike Clayton. (20 Sep 2022). How to Pick the Best Graphics Card for Machine Learning. Retrieved 4 Dec 2022. From https://towardsdatascience.com/how-to-pick-the-best-graphics-card-for-machine-learning-32ce9679e23b
3. JavaPoint Team. (17 May 2022). Machine learning Life cycle. Retrieved 6 Dec 2022. From https://www.javatpoint.com/machine-learning-life-cycle
4. Sumit Saha. (16 Dec 2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Retrieved 6 Dec 2022. From https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
5. Damilola Omoyiwola. (26 Oct 2018). Machine Learning on Facial Recognition. Retrieved 6 Dec 2022. From https://medium.datadriveninvestor.com/machine-learning-on-facial-recognition-b3dfba5625a7
6. From Wikipedia, the free encyclopedia. (4 Dec 2022). Machine learning. Retrieved 7 Dec 2022. From https://en.wikipedia.org/wiki/Machine_learning
7. Surapong Kanoktipsatharporn. (27 July 2019). Training Set, Validation Set and Test Set in Machine Learning. Retrieved 7 Dec 2022. From https://www.bualabs.com/archives/532/what-is-training-set-why-train-test-split-training-set-validation-set-test-set/
8. Andrew Murphy. (02 May 2019). Batch size (machine learning). Retrieved 9 Dec 2022. From https://radiopaedia.org/articles/batch-size-machine-learning
9. From Wikipedia, the free encyclopedia. (18 Oct 2022). PyTorch. Retrieved 9 Dec 2022. From https://en.wikipedia.org/wiki/PyTorch
10. Siddharth M. (14 Sep 2021). Building Resnet-34 model using Pytorch – A Guide for Beginners. Retrieved 10 Dec 2022. From

https://www.analyticsvidhya.com/blog/2021/09/building-resnet-34-model-using-pytorch-a-guide-for-beginners/

11. Surapong Kanoktipsatharporn. (25 July 2019). Artificial Intelligence, Computer Vision, Deep Learning, Knowledge, Machine Learning, Python, Software Engineering. Retrieved 11 Dec 2022. From https://www.bualabs.com/archives/482/resnet34-vs-resnet50-deep-learning-pets-cats-dogs-image-classification-with-fastai-v1-ep-2/
12. Pragati Baheti. (3 Oct 2022). A Comprehensive Guide to Convolutional Neural Networks. Retrieved 11 Dec 2022. From https://www.v7labs.com/blog/convolutional-neural-networks-guide