**HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY OF MECHANICAL ENGINEERING**

🙦🕮🙤



**REPORT**

**Project:**

**HAND-DRAWN PREDICTION APPLICATION**

**Lecturer: Nguyen Truong Thinh**

**Student: Nguyen Dang Duy Tan…….20134007**

**Subject: Artificial Intelligence**

**Ho Chi Minh City, January 2023**

# ACKNOWLEDGEMENT

I am almost halfway through the school's training program after two years of hard effort and acquiring specialized knowledge*.*Throughout the journey, I have received dedicated instruction from the teachers of the University of Technology and Education of Ho Chi Minh City, as well as the teachers of the Faculty of Mechanical Engineering. That useful knowledge was essential in completing this project. As a result, I'd like to start by thanking all the teachers in the Ho Chi Minh City University of Technology and Education in general, and the Mechanical Engineering faculty in particular.

I would like to express my gratitude to teacher Nguyen Truong Thinh. The teacher has dedicatedly guided me to complete the project in the best way possible; without his guidance, this project may not be completed.

However, mistakes cannot be prevented when doing so. To make this report more thorough, I welcomes your comments and suggestions.

Finally, I would like to express my thankful to the professors once again.

*Sincerely,*

Nguyen Dang Duy Tan

Table of contents

[ACKNOWLEDGEMENT 2](#_Toc124537283)

[ABTRACT 4](#_Toc124537284)

[CHAPTER 1 : OVERVIEW 5](#_Toc124537285)

[1.1 Rational and reason for choosing the topic 5](#_Toc124537286)

[1.2. Aim of the study 5](#_Toc124537287)

[1.3.Limitation 6](#_Toc124537288)

[1.4. Relative study 6](#_Toc124537289)

[CHAPTER 2: THEORETICAL FOUNDATION 8](#_Toc124537290)

[2.1. Artificial Intelligence 8](#_Toc124537291)

[2.1.1. What is artificial intelligence? 8](#_Toc124537292)

[2.1.2. Application of AI 8](#_Toc124537293)

[2.2. Machine learning 9](#_Toc124537294)

[2.2.1. What is machine learning? 9](#_Toc124537295)

[2.2.2. Key ML terminology 9](#_Toc124537296)

[2.2.3. ML classification criteria 10](#_Toc124537297)

[2.3. Implementing a Machine Learning Project 10](#_Toc124537298)

[2.3.1. Look at the big picture 11](#_Toc124537299)

[2.3.2.Get the data 11](#_Toc124537300)

[2.3.3. Discover and visualize the data to gain insights 11](#_Toc124537301)

[2.3.4. Prepare the data for machine learning algorithms 11](#_Toc124537302)

[2.3.5. Train and evaluate models 11](#_Toc124537303)

[2.3.6. Fine-tune your model 12](#_Toc124537304)

[2.3.7.Analyze and test your solution 12](#_Toc124537305)

[2.3.8.Launch, monitor and maintain your system 12](#_Toc124537306)

[CHAPTER 3: METHODOLOGY 13](#_Toc124537307)

[3.1. Dataset 13](#_Toc124537308)

[3.2. Model 14](#_Toc124537309)

[3.2.1.MLP (Multilayer Perceptron) 14](#_Toc124537310)

[3.2. Convolutional Neuron Network (CNN) 18](#_Toc124537311)

[CHAPTER 5: TESTING AND CONCLUSION 23](#_Toc124537312)

[5.1. Testing with application 23](#_Toc124537313)

[5.2. Result 24](#_Toc124537314)

[5.3. Conclusion 24](#_Toc124537315)

[REFERENCE 26](#_Toc124537316)

# ABTRACT

I demonstrate a hand-drawn prediction application based on a neural network model trained on the Quick Draw dataset. The model achieves excellent levels of accuracy on the Quick Draw dataset and has strong generalization to other datasets. In addition, we analyze the model's strengths and weaknesses, as well as prospective enhancements. Overall, the results show that the neural network model trained on the Quick Draw dataset is an effective tool for hand-drawn picture identification tasks.

# CHAPTER 1 : OVERVIEW

## 1.1 Rational and reason for choosing the topic

The 4.0 revolution, also known as Industry 4.0, is ushering in a new era of technological advancements, one of which is the integration of artificial intelligence (AI) into various aspects of life. AI has the ability to perform tasks that ordinarily require human intellect, making it a powerful tool for automating and improving many processes. The ability of computers to "self-learn" and interpret and analyze new data without human intervention is a defining feature of AI technology. Additionally, the ability to process data in very large amounts and at high speed is another important characteristic of AI, making it a highly efficient tool for various applications.

Keeping in mind the current trend of the 4.0 revolution, I decided to choose the topic of developing a hand-drawn image recognition application based on a neural network model. This topic not only allows me to better understand the capabilities of AI, but also serves as my final project in the subject. Hand-drawn image recognition is a field that has various practical applications in areas such as education, entertainment, technology and accessibility for people with disabilities. The Quick Draw dataset, with its extensive collection of labeled hand-drawn images, provides a valuable resource for training machine learning models in this field. By developing a hand-drawn image recognition application based on a neural network model, I aim to contribute to the advancement of this field and the development of more accurate and efficient hand-drawn image recognition systems.

## 1.2. Aim of the study

Determine the most suitable model for the Quick Draw dataset, giving out the best accuracy.

- Understand the basic components of machine learning models.

- Data selection and preprocessing

- Trainning the model and make application to apply

## 1.3.Limitation

- Take a lot of time to train

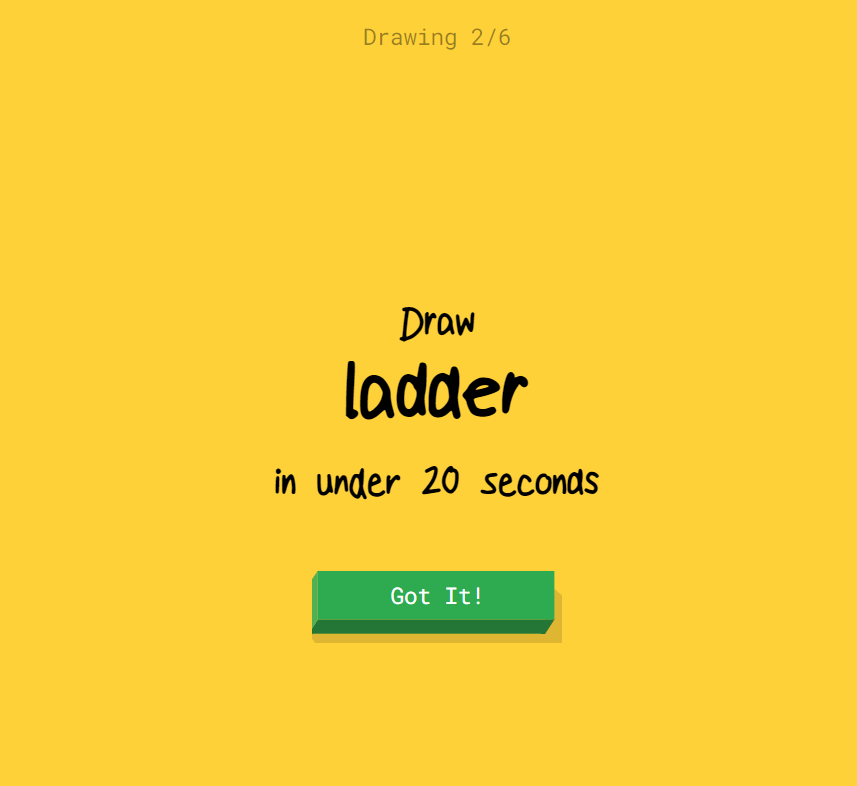
- They take a lot of memory to execute.

- No theoretical foundation for neural network.

- The accuracy might high but when testing can guess wrong a lot

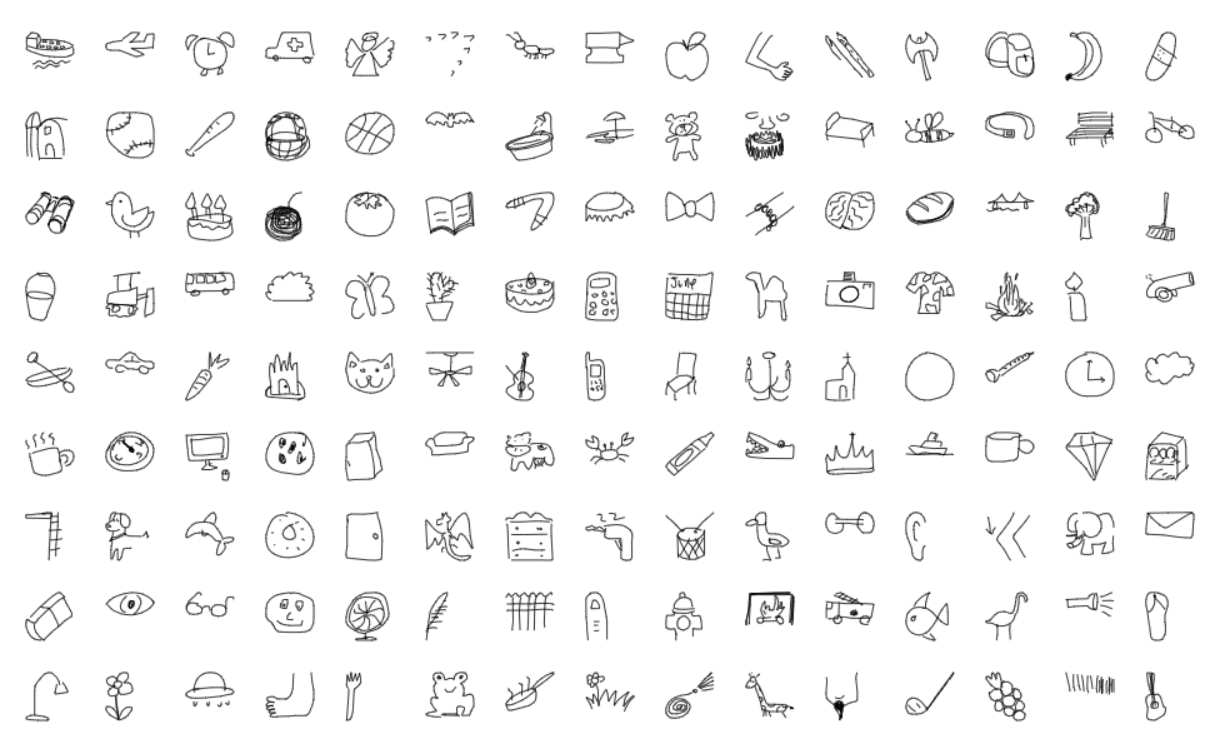
## 1.4. Relative study

In recent years, the Google Quick Draw website has attracted a great deal of attention from researchers in various fields. This web-based application allows users to draw images of various objects and have those images classified by a neural network based on the Quick, Draw dataset which is a collection of over 50 million hand-drawn images of over 345 different object categories.

**Fig1.1.** *The Quick Draw game*

Some researchers have used the Quick Draw dataset to train models for various applications such as image classification, data visualization, and understanding the bias in the dataset. These studies have shown that the dataset has a high potential for various tasks and applications.



**Fig1.2.** Quick,Draw dataset

# CHAPTER 2: THEORETICAL FOUNDATION

The theoretical foundation of my project is rooted in the field of artificial intelligence (AI) and machine learning (ML).

## 2.1. Artificial Intelligence

### 2.1.1. What is artificial intelligence?

Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence. These processes include learning, reasoning, and self-correction

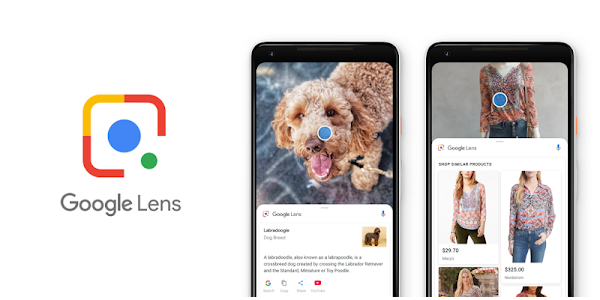
### 2.1.2. Application of AI

* Translator in Skype



**Fig2.1***. Translator in Skype*

* Google lens, Google translate



**Fig2.2** *Google lens*

* Sketch-to-Image Translation



**Fig2.3.***Generate image from sketch*

* ….

## 2.2. Machine learning

### 2.2.1. What is machine learning?

ML, on the other hand, is a subset of AI that involves the development of algorithms and statistical models that enable machines to improve their performance on a specific task without being explicitly programmed.(Samuel, 1959)

### 2.2.2. Key ML terminology

* Dataset: a dataset in machine learning is, quite simply, a collection of data pieces that can be treated by a computer as a single unit for analytic and prediction purposes.
* Sample/example/instance/data point:Is an independent unit of information in your data set. A set of data points is called a dataset.
* Feature: a feature is an individual measurable property or characteristic of a sample. Choosing informative, discriminating and independent features is a crucial element of effective algorithms in pattern recognition, classification and regression
* Training data: is the data you use to train an algorithm or machine learning model to predict the outcome you design your model to predict.
* Testing data: is used to measure the performance, such as accuracy or efficiency, of the algorithm you are using to train the machine.
* Model: is a file that has been trained to recognize certain types of sample. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data
* Performance: Performance of an ML model is just "how good" it does at a particular task. Ex: Accuracy, precision, recall…

### 2.2.3. ML classification criteria

**2.2.3.1. With human supervision or not?**

* Supervised learning: use labeled data. Classification and regression (k-Nearest Neighbors, Linear Regression, Logistic Regression, Support Vector Machines (SVMs), Decision Trees and Random Forests, Neural networks…)
* Unsupervised learning: use unlabeled data.

+ Clustering:K-Means, DBSCAN, Hierarchical Cluster Analysis (HCA)…

+ Anomaly detection:One-class SVM, Isolation Forest

+ Dimensionality reduction:Principal Component Analysis (PCA), Locally-Linear Embedding (LLE), t-distributed Stochastic Neighbor Embedding (t-SNE)

* Semi-supervised learning: combination of Supervised and Unsupervised
* Reinforcement learning: Agents are taught using rewards

**2.2.3.2. Learn online or not?**

* Batch learning (offline learning): the system is incapable of learning incrementally: It must be trained using all the available data. This will generally take a lot of time and computing resources, so it is typically done offline, first the system is trained and then it’s launched into production and runs without learning anymore; it just applied what it has learned.
* Online learning: we train the system incrementally by feeding it data instances sequentially, either individually or by small groups called mini-batches. Each learning step is cheap and fast, so the system can learn about new data.

**2.2.3.3. Use model or not?**

* Instance-based learning (kNN):compares new data to the learned examples using a similarity measure.
* Model-based learning: build a modelof the learned examples

## 2.3. Implementing a Machine Learning Project

*Main step:*

* Look at the big picture.
* Get the data.
* Discover and visualize the data to gain insights.
* Prepare the data for machine learning algorithms.
* Train and evaluate models.
* Fine-tune your models.
* Test and analyze your solution.
* Launch, monitor, and maintain your system.

### 2.3.1. Look at the big picture

Take an overview and determine the goals, ideas, and directions that the project needs to achieve. Set out the problems to be solved, compare with current capabilities, determine the approach to the problems posed. Identify situations, possible risks, develop coping methods. Decide how the problem will be solved, evaluate the effectiveness of the proposed solutions based on what criteria.

### 2.3.2.Get the data

Determine the data to be used in the project: define the types of data (images, sounds, data, reports, ...); the number of instances as well as the features of the data; format of the data as well as finding the data collection source and its reliability

### 2.3.3. Discover and visualize the data to gain insights

Explore, analyze, observe, visualize data to get the most intuitive view before training the model. This discovery process (EAD) will help to identify data elements such as data mismatches, missing instances or features, relationships between features. From there, orient to use the appropriate model and algorithm

### 2.3.4. Prepare the data for machine learning algorithms

Process and transform data to match the overview obtained from previous exploration. Including processing missing data, redundant or containing unnecessary and inappropriate features; define the label; handle relationships and extend features; Split data into train and test sets.

### 2.3.5. Train and evaluate models

Train model is based on algorithms and predefined data sets. Compare and evaluate models for fine-tuning. Can use libraries like TensorFlow, Keras,... and use algorithms like Linear Regression, Random Forest, SVM, ...

Calculate and compare the efficiency and accuracy of the models.

Train model involves predicting the label of the model based on the features, changing the appropriate parameters to minimize errors and false predictions.

There are 3 main parameters during model training:

* Accuracy: evaluate the effectiveness of the model.
* Loss function: calculate the loss value of the training process, the lower the loss value, the better.

Optimizer: changes based on loss value

### 2.3.6. Fine-tune your model

Fine-tune the model on the set of hyperparameters, select the best model for the hyperparameter. Cross-validation can be used with random search or grid search

### 2.3.7.Analyze and test your solution

Analyze and evaluate the obtained model against the goals, criteria, as well as the requirements set out by the project and of the Look at the big picture.

Identify the limitations and gains of the selected solution.

Test with actual data

### 2.3.8.Launch, monitor and maintain your system

Deploy solutions into products, combined with other systems, applications, APIs. Regularly evaluate, re-train the modem with the collected data to evaluate performance and make appropriate changes. If the data changes, it is necessary to retrain the model.

# CHAPTER 3: METHODOLOGY

## 3.1. Dataset

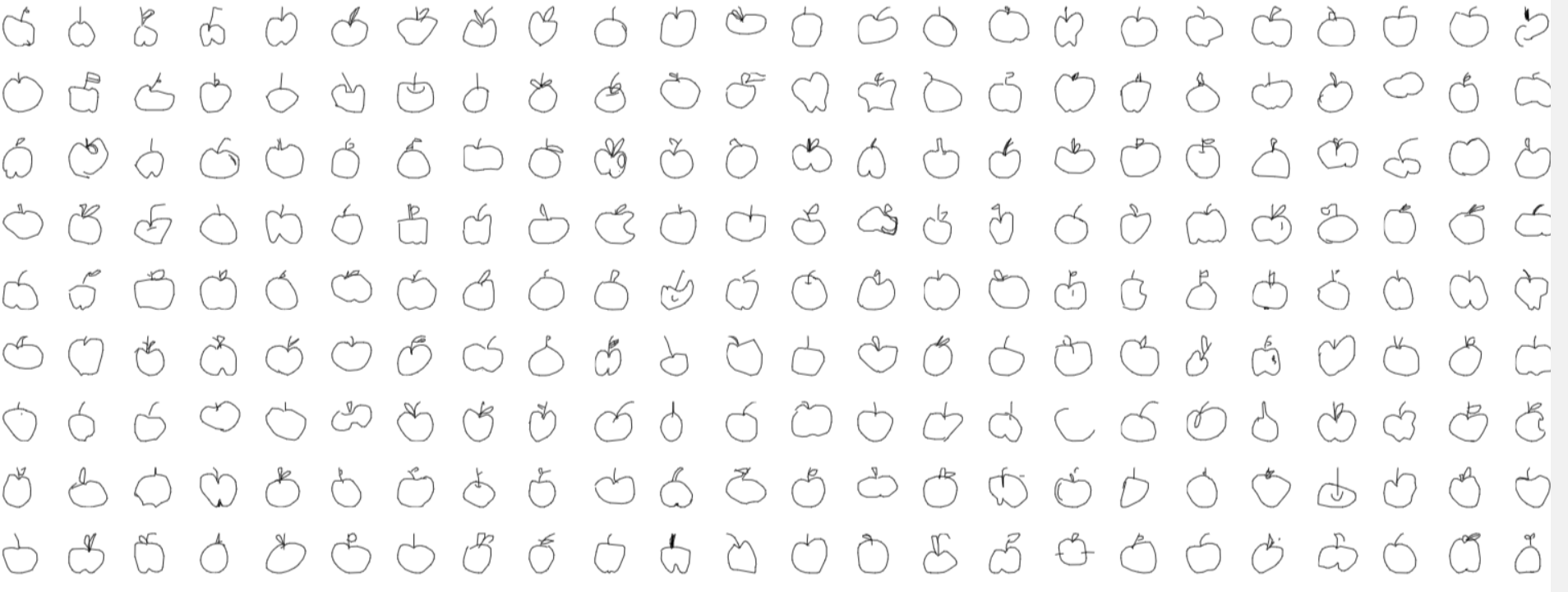
The Quick Draw dataset is a collection of over 50 million hand-drawn images of over 345 different object categories. The dataset was created by Google, and it is publicly available for research purposes. The images in the dataset were drawn by users of the Quick Draw website, which is a web-based application that allows users to draw images of various objects and have those images classified by a neural network.



**Fig3.1.** *Quick, Draw website*

The dataset is divided into several categories, such as animals, transportation, and household items. Each category contains a large number of images, with some categories having tens of thousands of images and others having just a few hundred. The number of images per class is not balanced, some classes have a lot of images while others have a few. This can affect the performance of models trained on the dataset.

The images in the dataset are in PNG format and have a resolution of 28x28 pixels. The images are also normalized to have a black background and white lines. This makes the dataset well suited for image classification tasks as it contains a large number of images and a wide variety of classes. However, it's important to note that the diversity of drawings within each class can vary significantly. This means that some classes might have very similar drawings while others might have very different drawings.



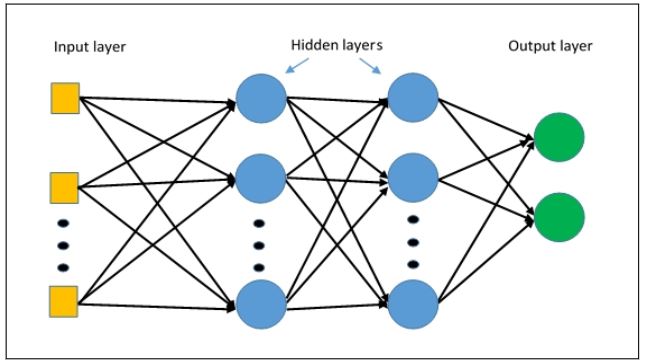
**Fig3.2.** *In the apple category, there are many variations of apples, however some of them are imperceptible noise*

## 3.2. Model

In this project, I will use the model to categorize 80 different types of hand-drawn images. This training will use 6000 photos from each class.

### 3.2.1.MLP (Multilayer Perceptron)

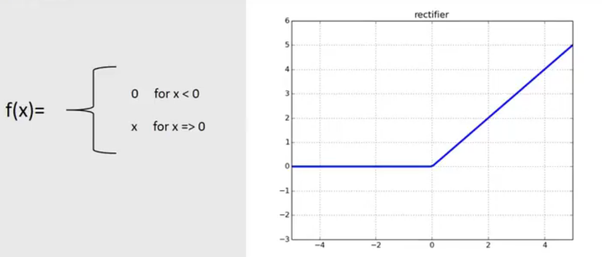
A Multi-Layer Perceptron (MLP) is a type of Artificial Neural Network (ANN) that uses supervised learning technique, backpropagation, to train a neural network. An MLP is composed of multiple layers of artificial neurons, also known as perceptrons, that are connected in a directed acyclic graph, with no cycles. It usually includes an input layer, one or more hidden layers, and an output layer. MLPs are good at learning non-linear function and can be used to solve a wide range of problems such as image recognition, natural language processing, and supervised learning problems such as classification and regression. They are relatively simple to understand and easy to implement but their performance depends on the data quality and problem complexity.



**Fig3.3** Basic structure of MLP

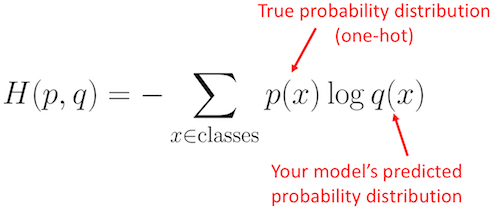
*Model deployment*

In this model, I will be utilizing the ReLU (Rectified Linear Unit) activation function. This function is widely used in deep learning models because of its ability to filter data effectively. It only allows values greater than 0 to pass through, effectively removing any negative values, which can help to improve the overall performance of the model.



**Fig3.4.** *ReLU activation function*

Additionally, I will be using the Cross-Entropy Loss function for loss calculation. This function is commonly used in classification problems, such as image recognition, as it measures the dissimilarity between the predicted and true probability distributions of the classes. This allows for accurate and efficient loss calculation, which is crucial for training and optimizing a deep learning model.



**Fig3.5.** Cross-entropy function

Overall, the combination of the ReLU activation function and Cross-Entropy Loss function will provide optimal performance for this model.

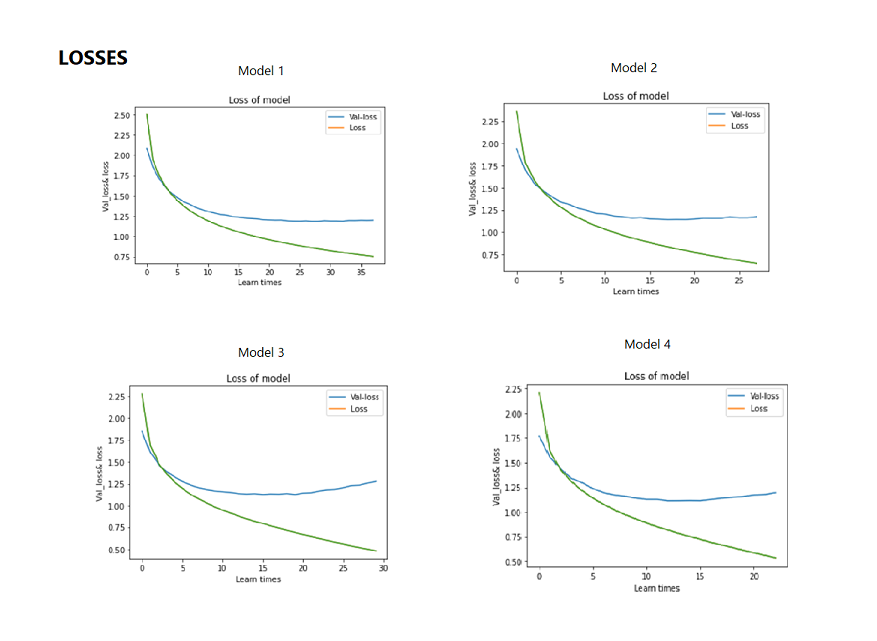
Next, use some MLP architectures to evaluated , from that find out the best performance.

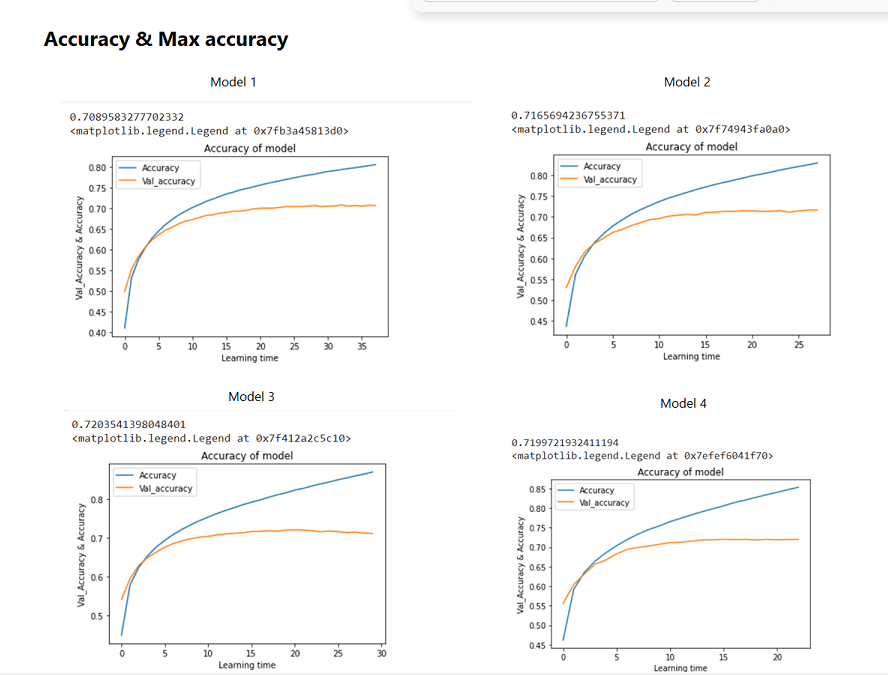
First, I create a basic model with only three layers, and evaluate the dataset. Later on, I increase the complexity of the network trying to obtain a better performance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **MLP ARCHITECTURE** |  |  |
| **Model** | Model 1 | Model 2 | Model 3 | Model4 |
| **Learning rate** | 0,0001 | 0,0001 | 0,0001 | 0,0001 |
| **Optimizer** | Adam | Adam | Adam | Adam |
| **n layers** | 3 | 4 | 5 | 6 |
| **n epochs** | 100 | 100 | 100 | 100 |
| **Layer 1** | (784, 500) | (784,500) | (784,500) | (784,500) |
| **Layer 2** | (500, 256) | (500,500) | (500,500) | (500,500) |
| **Layer 3** | (256, 80) | (500,256) | (500,500) | (500,500) |
| **Layer 4** |  | (256,80) | (500,256) | (500,500) |
| **Layer 5** |  |  | (256,80) | (500,256) |
| **Layer 6** |  |  |  | (256,80) |

**Table 1**

Collect all result for better comparision:





The highest level of accuracy achieved using these architectural designs was 72.03% in Model 3.

## 3.2. Convolutional Neuron Network (CNN)

In order to improve the accuracy of the model for training on the Quick Draw dataset, I decided to used Convolutional Neural Network because of it advantages.

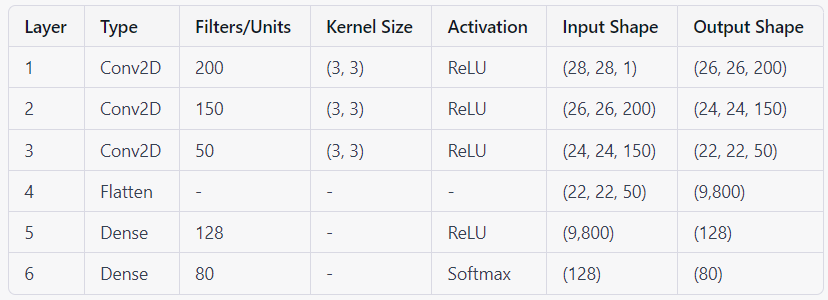
Specifically, CNNs are particularly well-suited for image classification tasks due to their ability to automatically and adaptively learn spatial hierarchies of features through backpropagation. This allows CNNs to effectively identify and extract features such as edges, textures, and shapes from images, which are crucial for recognizing and classifying the different drawings in the Quick Draw dataset. Additionally, CNNs can also take into account the spatial relationships between features, allowing them to effectively classify images even when the objects in the images are translated or rotated.



**Fig3.6.** Basic structure of CNN

After implementing the CNN architecture, compared the results with previous used MLP model and found that CNNs significantly improved the accuracy.

First I use a simple CNN architecture

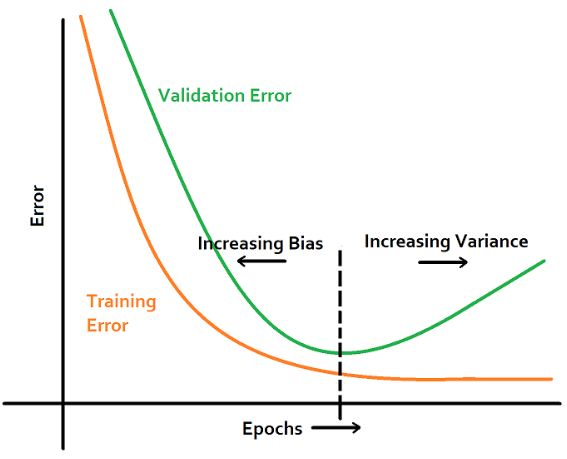


In addition to the architecture of the model, I also implemented several techniques to optimize the training process. One of the most important factors in training a deep learning model is the learning rate. The learning rate controls the step size at which the optimizer makes updates to the model's weights. A high learning rate can cause the model to converge quickly, but it may result in suboptimal performance. On the other hand, a low learning rate can cause the model to converge slowly, but it may result in better performance. I carefully selected an appropriate learning rate for the model to achieve optimal performance.



**Fig3.7.** *How to choose learning rate*

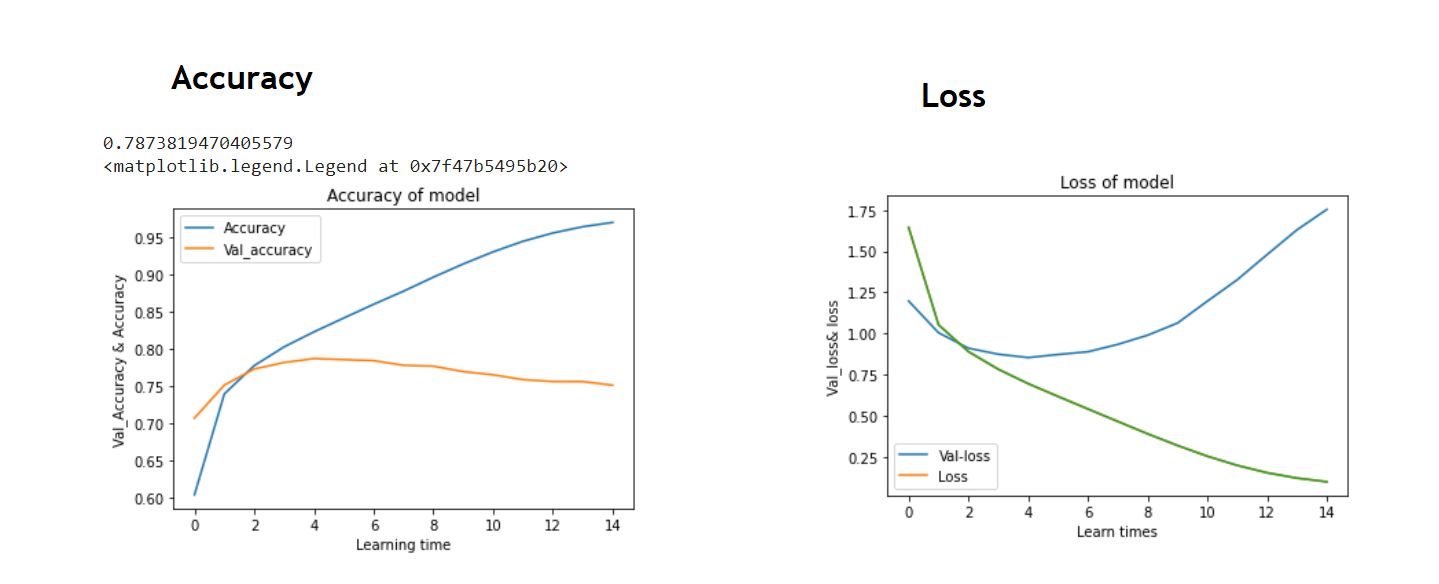
Another technique that I used to optimize the training process is the use of callbacks. Callbacks are functions that can be applied at certain stages of the training process, such as at the end of each epoch. One of the callbacks that I implemented is the Early Stopping callback. This callback monitors the performance of the model on a validation set and stops the training when the performance stops improving. This can help to prevent overfitting and save computational resources.



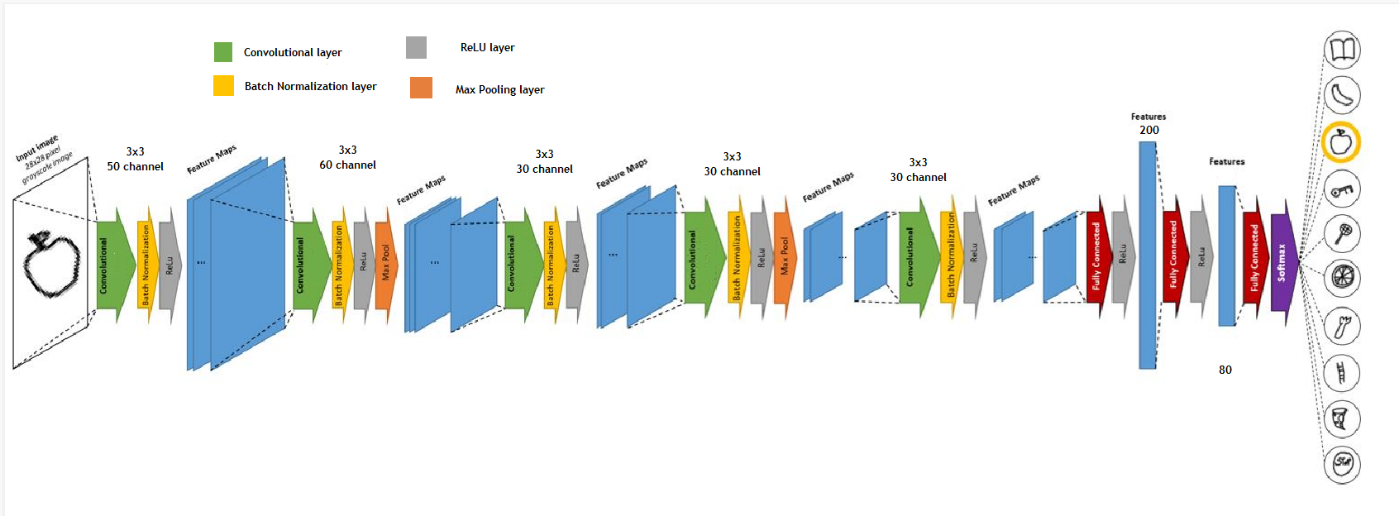
**Fig3.8.** *Using Early Stopping to prevent early overfitting*

Another callback that I used is the Model Checkpoint callback. This callback saves the model's weights after each epoch if the performance on the validation set improves. This allows me to save the best performing version of the model, and use it for further evaluation or deployment.

With the model above, I got the result as below:

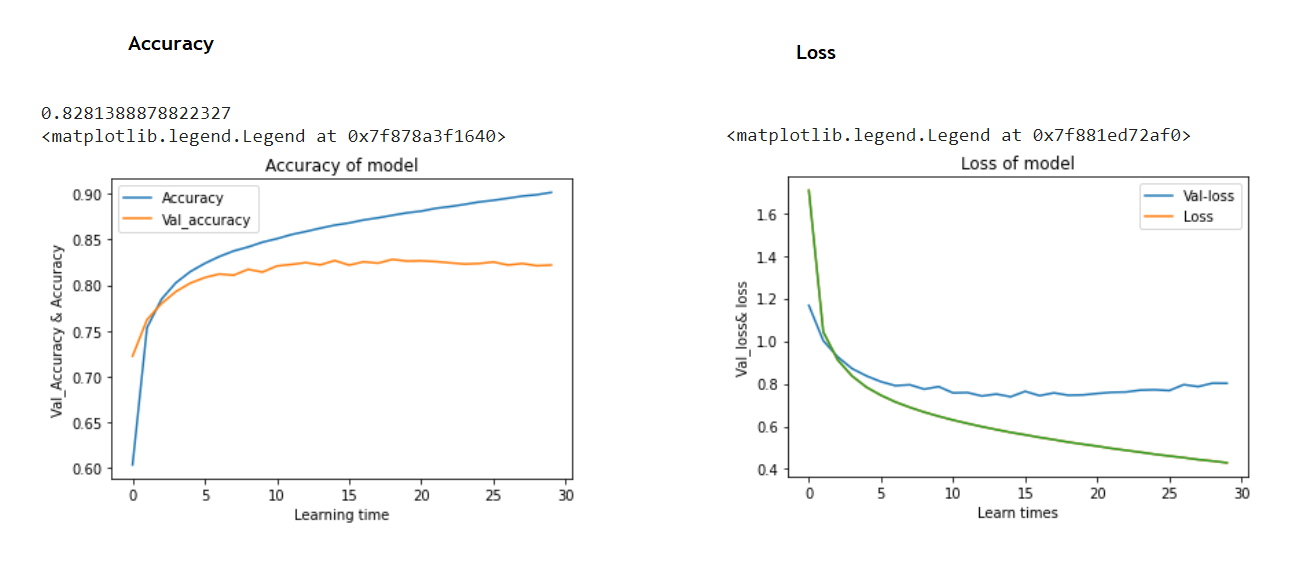


As can be seen from the graph, it is very shallow and gave poor result. In fact, most of the times it got overshoot very quickly after a few epochs. In fact, most of the times it got stuck very soon in a local minimum, so the results were awful.

So to improve the performance of the model, I expand the model more and combine it with new component. After a lot of testting I come up with the following structure that resulted to be excellent in terms of performance. This final architecture, which will be followingly explained, consists basically on alternating 5 convolutional layers (followed by a non-linearity) with 2 max-pooling layers and, ending with 3 fully connected layers also followed by non-linearity.

**Fig3.9.** *Suitable structure of CNN model*

With the model above, I got the result as below:

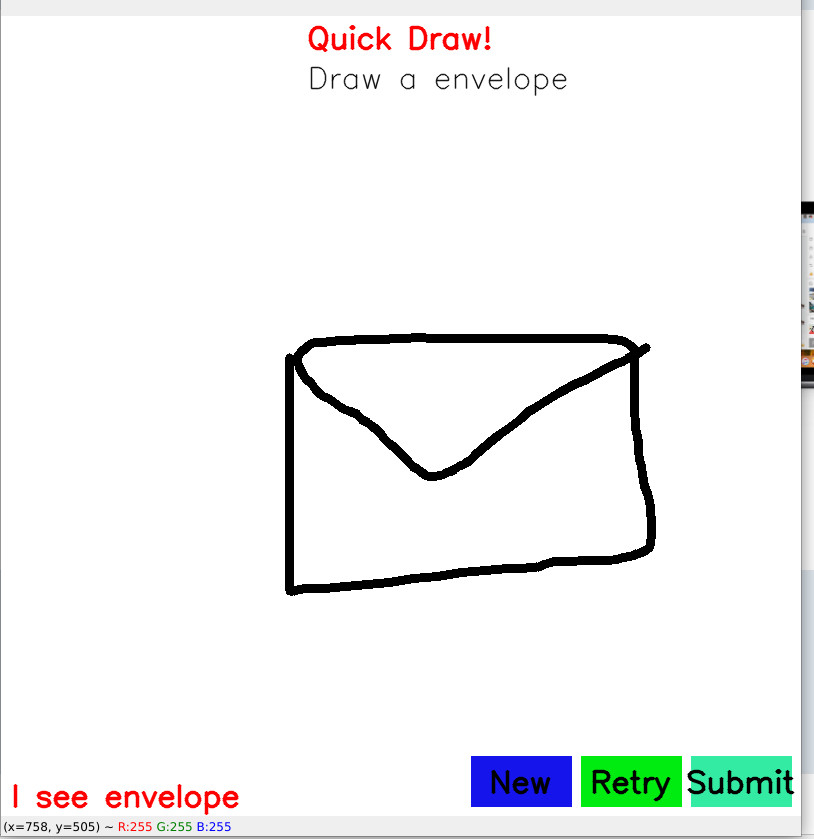
**

I obtained an accuracy of 82.81% on the test set using the above model. Despite some overfitting, the use of weight decay and batch normalization improved regularization and convergence.

# CHAPTER 5: TESTING AND CONCLUSION

## 5.1. Testing with application

I have developed an application that utilizes user input by allowing them to draw according to given prompts. The drawn images are then processed using basic image processing techniques before being passed through the trained model for evaluation. This process enables us to assess the stability and accuracy of the model, ensuring that it is suitable for the task at hand. This application not only allows users to have fun with the drawing task but also provides a way to check the performance of the model and make improvements if necessary.

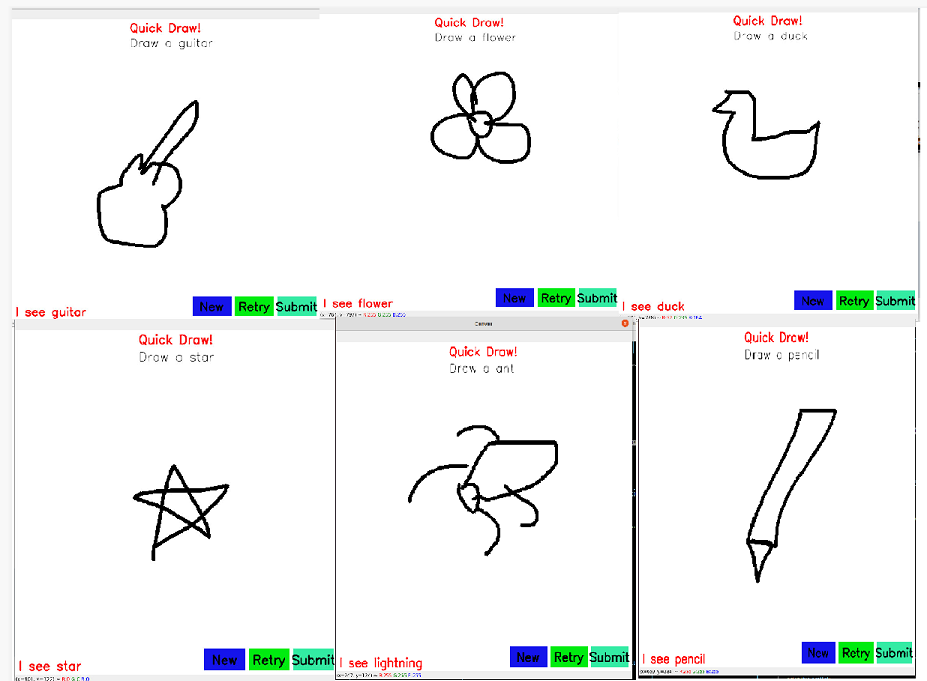


**Fig5.1.** *Application interface*

It features three buttons for the user to interact with, each serving a specific purpose. The first button is for requesting a new drawing assignment, the second button is for retrying a drawing task in case the machine fails to recognize it, and the third button is for submitting the drawing for the machine to evaluate and guess what the user has drawn.

## 5.2. Result

After several tests, the machine has shown to give quite accurate results, however, it still has a certain margin of error. This is due to the limitations of the model and the fact that the Quick Draw dataset used for training the model, while diverse, also contains some errors. These limitations and errors can cause the model to make incorrect guesses. To further improve the model's performance, it would be necessary to clean and pre-process the data more thoroughly, and also apply more complex architectures or techniques, such as transfer learning, to the model. These steps would help to reduce the number of errors and increase the overall accuracy of the model.

****

**Fig5.2.** *Wrong and right guess of the model*

## 5.3. Conclusion

his project marked my first experience with a deep learning problem. I encountered common deep learning challenges such as overfitting and hyperparameter tuning. Through this process, I have found that the CNN model performed the best with an accuracy of 83%.

In conclusion, this project has been an incredibly valuable learning experience. I started with little to no prior knowledge of machine learning, but by the end of the project, I was able to implement my own model. Although the process took longer than I expected, the satisfaction of obtaining successful results made it all worth it. This project also helped me better understand and grasp the key concepts discussed in class.

# REFERENCE

**[1]** game Quick, Draw!; A.I Experiment, made by Google team: https://quickdraw.withgoogle.com/

**[2]** Understanding the effect of the Batch Normalization layers: <https://papers.nips.cc/paper/7996-understanding-batch-normalization.pdf>

[3] A CNN implementation of Google's "Quick, Draw!":

<https://github.com/ck090/Google_Quick_Draw>

[4] Quick, Draw, Keras! CNN model:

https://www.kaggle.com/code/amneves/quick-draw-keras-cnn-model