## **Artificial Intelligence**

## **::Challenge 1 (10%)**

### **Due: 11 Nov 2023, 11:59 PM**

Email to: [vu.tran@vnuk.edu.vn](mailto:vu.tran@vnuk.edu.vn)

According to the VNUK *Academic integrity policy*, plagiarism is:

*"Claiming and using the thoughts or writings or creative works of others without appropriate acknowledgement or attribution.  It includes:  
(a)  copying part or all of another student's assignment;  
(b)  allowing another person to write some or all of an assignment;  
(c)  copying paragraphs, sentences or parts of sentences directly from texts or the internet without enclosing them in quotation marks or otherwise showing them to be copied - even if the source is acknowledged, this is still plagiarism;  
(d)  using concepts or developed ideas, even if paraphrased or summarised, from another person, from texts or the internet without acknowledging the source;  
(e)  copying graphics, architectural plans, multimedia works or other forms of intellectual property without appropriate acknowledgment."*

The consequences of plagiarism (depending on the seriousness of the case) range from reducing your mark or failing the assignment up to a formal reference to a summary inquiry:

|  |
| --- |
| *By signing below I certify that the attached assignment is my own work.*  Student ID: 20020023                         Student Name: Huynh Ngoc Thai Khang                 Signature: |
|  |

**Grade:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Question** | **Grade** |
| 1 | Question 1 |  |
| 2 | Question 2 |  |
| 3 | Question 3 |  |
| 4 | Question 4 |  |
| 5 | Question 5 |  |
| Total gold coins | |  |

This problem set will introduce you to using control flow in Python and formulating a computational solution to a problem.

## Data

<https://github.com/googlecreativelab/quickdraw-dataset>

The Quick Draw Dataset is a collection of 50 million drawings across [345 categories](https://github.com/googlecreativelab/quickdraw-dataset/blob/master/categories.txt), contributed by players of the game [Quick, Draw!](https://quickdraw.withgoogle.com). The drawings were captured as timestamped vectors, tagged with metadata including what the player was asked to draw and in which country the player was located. You can browse the recognized drawings on [quickdraw.withgoogle.com/data](https://quickdraw.withgoogle.com/data).



## Requirements:

|  |  |  |
| --- | --- | --- |
| **No.** | **Criteria** | **Weight (%)** |
| 1 | Train the model | 20% |
| 2 | Deploy the model | 30% |
| 3 | Explain the math/model | 15% |
| 4 | Complete app | 15% |
| 5 | Git usage | 10% |

## 

## Questions

1. **Train the model**

Choose only 2 or 4 items to train your model

Getting Data:

* creating a new CNN class for implementing a Convolutional Neural Network model
* loading three datasets (for example: car, fish and snowman)
* splitting datasets into training and test data shuffling data

Building the Model:

* creating a sequential CNN model
* adding layers to the model
* compiling the model

Training the Model

* fetching batches of data
* training, testing and evaluating the model
* plotting graphs of the model loss and accuracy during training

Reference:

<https://github.com/zaidalyafeai/Notebooks/blob/master/Sketcher.ipynb>

Explain about CNN layers:

<https://youtu.be/NL6eCtMjikQ>

1. **Deploy the model**

Use Web technology (for example Python Flask/NodeJS/PHP/Laverel) to code function same as <https://quickdraw.withgoogle.com/>

Predicting Samples

* fetching batches of samples
* predicting fetched samples

Drawing Doodles

* creating a new Painter class to allow users to draw their own doodles with the mouse
* defining painting objects: drawing area, bitmaps, pencil
* adding a function for drawing a smooth line between two points using quadratic curves

Recognizing Doodles

* resizing doodle drawing to the required size of 28x28
* normalizing array of pixels before passing it as the input of the CNN model predicting doodle

**Debriefing Report :: Part 1**

**Part 1. Report on the challenge.**

# Report on Handwritten Character Recognition Challenge

## Introduction

The goal of this challenge is to build a model to recognize handwritten characters using a convolutional neural network (CNN) and deploy it as a web application.

## Data Collection and Preprocessing

- The Quick Draw dataset from Google was used which contains 50 million drawings across 345 categories.

- 2 classes were chosen: cat and dog. The image data for these classes was extracted from the full dataset.

- Each image is 28x28 pixels in size and contains a single drawing.

- The data was split 80/20 into training and validation sets. The training set had 10000 images per class.

- Data was normalized by converting pixel values to range 0-1 before feeding into the model.

## CNN Model Architecture

- A sequential CNN model was built with the following layers:

- 2D Convolutional layers with ReLU activation and max pooling

- Flattening layer

- Fully connected layers

- Output layer with softmax activation

- The convolutional layers extract features from the input images while the dense layers classify the features into output classes.

- Categorical crossentropy loss and adam optimizer were used during training.

## Model Training

- Model was trained for 10 epochs with a batch size of 32.

- Training and validation accuracy reached >95% showing the model was learning the patterns well.

- Loss decreased consistently during training.

## Model Deployment

- Flask webapp was built to draw characters and recognize them.

- Drawing coordinates were captured on mousedown, mousemove etc.

- Drawn image was resized, normalized and fed to the CNN.

- Model predicted the character class and displayed it to the user.

**Explain some functions in the code:**

1. init() function (Javascript)

* Initializes drawing context on the canvas
* Attaches event handlers to canvas:
  + mousedown: Starts drawing when mouse is pressed down
  + mousemove: Draws line when mouse is moved
  + mouseup: Stops drawing when mouse is released

1. draw() function

* Draws a straight line between two points (x1, y1) and (x2, y2)
* Uses ctx.moveTo(), ctx.lineTo() and ctx.stroke()
* Updates last coordinates to continue drawing

1. postImage() function

* Converts canvas to base64 string using toDataURL()
* Sends POST request with image string to '/recognize'
* Processes response to get prediction result

1. recognize() function in app.py

* Receives and decodes base64 string into image
* Resizes and normalizes the image
* Makes prediction using trained CNN model
* Returns prediction, accuracy as JSON