AI, GO DOODLE: Web Game Service with AI

Dongwoo Kim[1], Donghan Kim[2], Yeonghyeon Kim[1], and Hyeonggyu Choi[3]

[1] Sungkyunkwan University, Mathematics
[2] Sungkyunkwan University, French Language and Literature
[3] Sungkyunkwan University, Integrative Biotechnology

Abstract. AI, GO DOODLE is a web game in which at least two people participate, draw and enjoy with given random keywords. In addition, artificial intelligence intervenes to score sketches drawn by players or to guess the label of the sketches. For this role, image classification or object detection technology will be applied with AI. according to our resources. We planned to train AI. with a subset of a "Quick, Draw! The data" open-source dataset which has 345 categories of 500,000 sketch data. We also planned to use socket.io to build a system so that multiple players can participate and chat at the same time.

Keywords: AI, GO DOODLE · Web Game Service · Artificial Intelligence · Drawing

1 Introduction

'Covid-19' will be the keyword that has dominated social issues in recent years. This virus changed our lives so much. As we can see in the phrase 'stay home, save lives, the time spent at home has increased and communication with people has decreased significantly. The web game "A.I, GO DOODLE" is designed for people who lack such entertainment and are bored with quarantine. We developed a game for many people under the motto of easy, simple, and enjoyable. In addition, we decided to introduce AI that will play the role of a scorer or examiner of the game. With the development of artificial intelligence technology, people's interest has increased. By using this technology in our games, we hope players feel that AI is not a difficult thing, but is familiar and fun.

2 Objective

Our project aims to create a novel web-based game service with the following features. First, a deep learning model trained with the doodle dataset is applied to the game to help users perceive AI as familiar and interesting. Second, it enables an immediate reaction from the user through AI's fast and accurate picture recognition. Third, it stimulates users' interest through a simple UI and a fun Game Mode. Fourth, multiple users can chat and draw pictures at the same time to communicate with each other and have fun.

3 Related Work

3.1 Related games

3.1.1 Catch mind 'Catch Mind'[1] is a computer game serviced by Netmarble. In a game of 'Catch Mind', Each player takes turns and draws a picture that matches the problem. And if someone else gets the correct answer on the picture, both the person who drew it and the person who guessed it get experience points and credits. The game is played for 20 rounds, and the team or user with the most total points wins.

3.1.2 Quick, Draw! 'Quick, Draw!'[2] is an online game developed by Google that challenges the players to draw a picture of an object or idea for a given keyword and then uses artificial intelligence to guess what the drawings represent. The game is similar to 'Pictionary' in that the player only has a limited time to draw. In a game of 'Quick, Draw!', there are six rounds. Before each round, the player is given 20 seconds to draw a randomized object selected from the game's database. As they begin each round, they have 20 seconds to draw the given prompt, whilst the artificial intelligence attempts to guess the drawing. A round ends either when the artificial intelligence successfully guesses the drawing or the player runs out of time. At the end of a 'Quick, Draw!' game, the player is given their drawing and results of each round. The player can also check the artificial intelligence's comparison and guesses with other player-given objects, before either quitting or replaying.

3.2 Socket.io

Since our game operates in real-time on the Web, we decided to use the socket.io library. Socket.IO is a library that enables real-time, bidirectional, and event-based communication between the browser and the server. It is built on top of the WebSocket protocol and provides additional guarantees like a fallback to HTTP long-polling or automatic reconnection.

3.3 Image Classification

Image Classification is one of the core problems in Computer Vision, which aims to predict the class of the object in the image. Until 2011, the feature descriptor-based methods, which analyze the feature of the given image and predict the class, were superior to any other methods. However, in 2012, the AlexNet[3], which took first place in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), showed outstanding performance of the deep convolutional neural network on the image classification task. This triggered various research on the architecture of the model, appropriate hyperparameters, activation functions, etc. VggNet[4] used a smaller size of the filter to reduce the overall computation and applied deeper layers for the model architecture. They implemented the model with 16 layers or 19 layers while AlexNet has only 6 layers. Moreover, ResNet[5] solved the vanishing gradient problem by skip connection and the bottleneck architecture, which led to deeper architecture. MobileNet[6] focused on the effective usage of CNN-based models on a light device. They applied depthwise separable convolution to reduce the total number of weights to boost the overall computing speed of the model.

4 Design

4.1 Overall Architecture

The web page for the game will be created with the pug template engine and node express web server. We have two main servers for this project, a Node.js server for the game engine and a simple Python server for AI works and HTTP response. Node.js server will render the game and connects users in real-time using a Web socket. Python server will receive an image file that is drawn in real-time by users as JSON file format and then return the accuracy of images as result to the game server. In this project, these servers communicate with each other using REST API.
AI Server first receives each user's drawing file in the game part as a string in base64 format in JSON through HTTP communication. After decoding the received base64 format and saving it as an image, the bounding box work and preprocessing work are started. After that, it is put into the learned model, and when the accuracy according to the keywords is obtained, the accuracy of the input keyword is output. When the accuracy of each user comes out, post-processing such as sorting and ranking according to the accuracy is carried out, and the result is put back in JSON and sent to the game part through HTTP communication. The **Fig. 1** below shows the overall project structure.

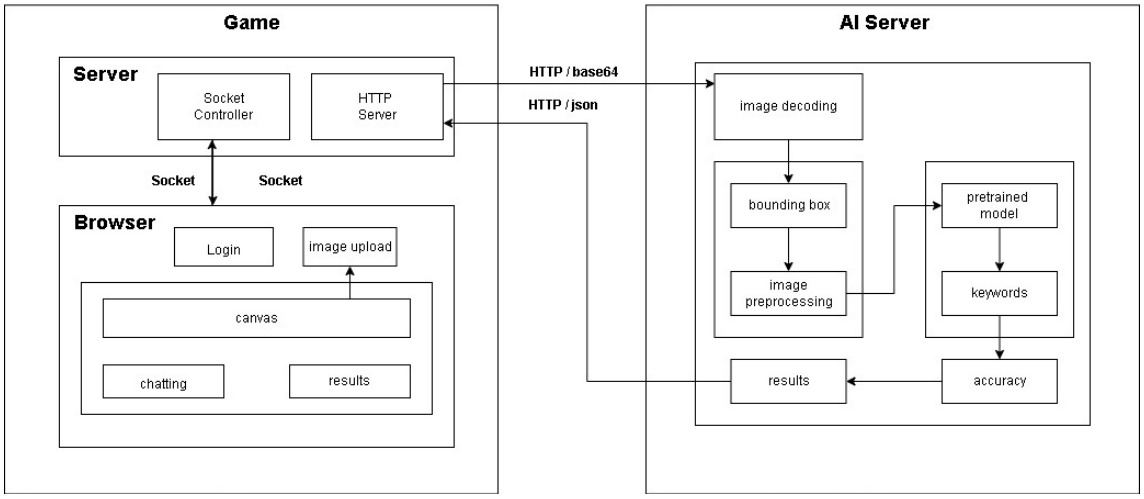

Fig. 1. Overall Architecture shows overall architecture of project. The game server receives an image from a browser connected by a web socket, transmits it to the AI server, and sends a response to it to the browser.

4.2 Game Design

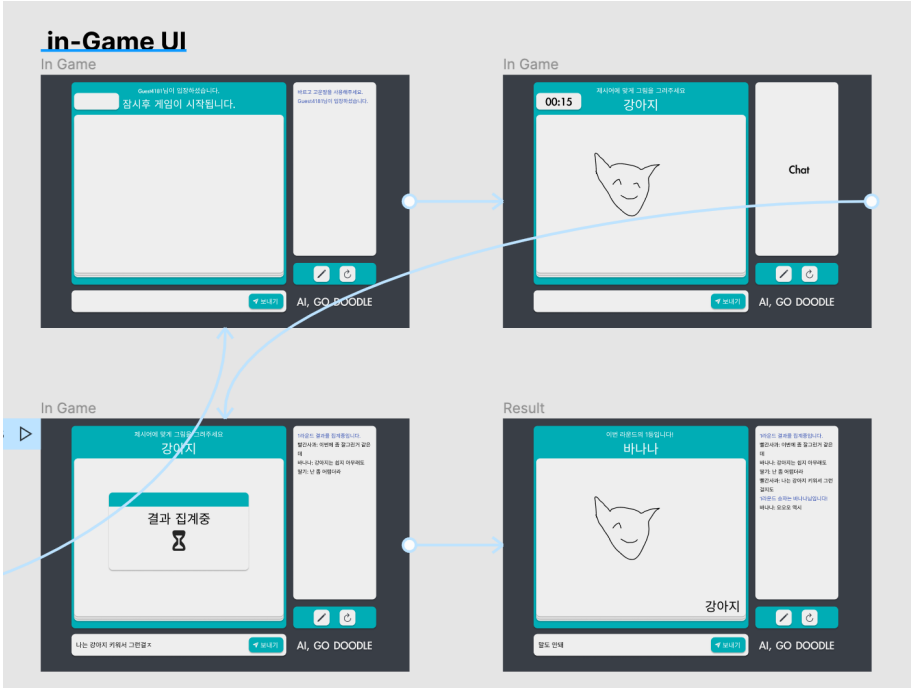

Fig. 2. Figma Example. This is the image of designing the prototype of the project using Figma. The overall composition of each screen was devised, and the organic connection between each screen and the UI was also designed.

4.2.1 Prototyping For UI/UX initial work and prototyping, we used Figma, a prototyping tool. First of all, since we pursued the direction of our game to be highly accessible, we set the

direction with a design that was easy to see and intuitive. The reason for using Figma as a design tool is that it is easy to place UIs on the screen and provides a function to set the flow. **Fig. 2** is the result of our prototype design. The screen was composed by limiting the colors to three, the game was designed to be played on one page, and the arrows on the screen indicate the flow between the UIs.

4.2.2 Front-end Since AI, GO DOODLE is a web-based game, it was implemented through HTML/CSS/Javascript, and the server was also implemented using Node.js.

Fig. 3. PUG templete engine makes HTML code more concise and easy to understand. Both codes produce the same result but can see that the PUG code is much more concise.

We applied PUG and SCSS, which are template engines for HTML and CSS preprocessors, for convenient work. Looking at **Fig. 3**, the HTML code on the left and the PUG code on the right shows the same style. The result is the same but can see that the code on the PUG side is much more concise and easy to understand. SCSS works on the same principle.

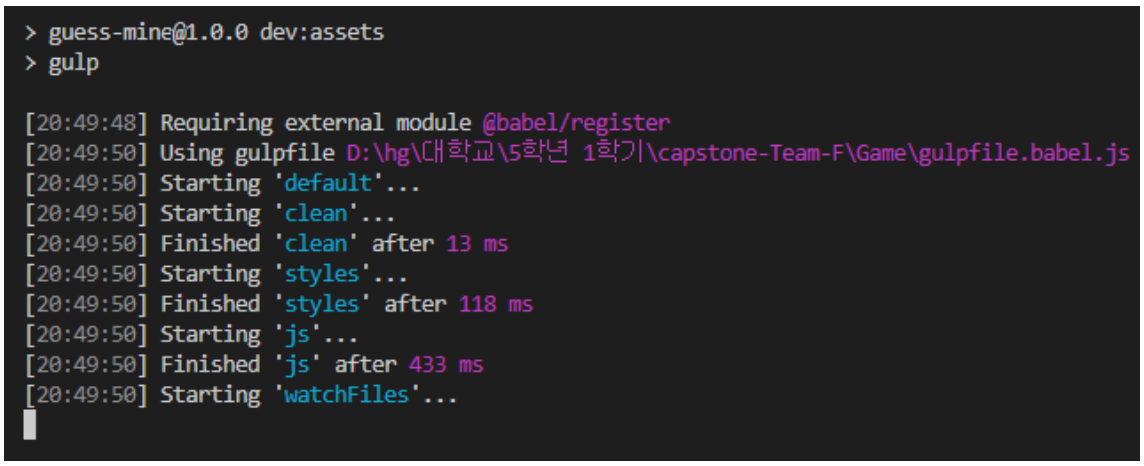

Fig. 4. Gulp is a tool that performs repetitive build tasks instead. This fig shows Gulp running and working in the terminal.

Gulp was also used. Gulp is a very useful tool that automates repetitive and frequently used tasks in JavaScript and is called a build system. It can save time and is a great help in improving productivity, so it is often used during project progress. This tool collects JavaScript libraries and third-party apps, performs minification and compression, performs unit tests, compiles HTML/CSS, and helps browser refresh. **Fig. 4** shows Gulp in action. In this state, the programmer just edits and saves the PUG/SCSS code, and the tool automatically performs compression, build, and refresh.

4.2.3 Back-end As mentioned above, a game server is created with Node.js, for HTTP requests to the AI server and web socket communication with browsers. The HTTP communication is used to send an image file to an AI server and receive a processed result value.
Web socket communication is used by servers and browsers to exchange data. To facilitate the implementation of web socket communication in the project, we utilized the socket.io library. The server provides a user's entry and exit notification to the browser through web socket communication and enables real-time chat. In addition, Sending the browser's picture to the server was also implemented through socket communication.

4.3 AI Design

4.3.1 AI Server AI Server was written using the Python language and was created using Flask, a Python web framework, and PyTorch, an open-source machine learning library. After that, it was deployed by uploading it to an instance of AWS EC2.

4.3.2 Judging system When the user's drawing is passed to the AI server, the model consumes the image and predicts the score of the drawing with the given keyword. Our model showed low accuracy when trained with more than 50 keywords. Since our project aimed to deal with 100 keywords, we split the keywords into 4 groups and trained each group of keywords with 4 independent models. Also, we considered the combination of the keywords that shows reasonable performance by using the confusion matrix.

4.3.3 Bounding Box Since our model is trained with the image of size 32x32 pixel, the user's drawing is resized into the same size in order to get the accuracy for the given keyword. However, resizing the image into a small size has one inheriting problem, the loss of information. **Fig.5.** shows the situation when a 500 x 500 image is resized into a size of 32 x 32. This problem caused the inaccurate prediction of the model, so it was a big challenge for the project. We applied a bounding box to handle the problem. Applying the bounding box makes the actual resizing region gets smaller. Also, if the thickness of the line is adjusted, it is able to resize the image into the required size for the model, preserving the information of the user's drawing as seen in **Fig. 6.**. We applied a square bounding box to preserve the shape of the drawing.

Fig. 5. Login screen is the screen that appears when a user accesses a web address for the first time. Users can choose a nickname and enter the game here.

Fig. 6. Resizing user's image The left image shows the resized image of the user's image without the bounding box and the right image shows the resized image of the user's image with the bounding box applied.

5 Implementation

5.1 Servers Workflow

A game server created with Node.js and express frameworks delivers HTML files that implement the game UI to users who access the game. And then, when the game starts, users in the same

room are tied up through socket communication using the socket.io package to maintain sessions and enable real-time chat and game operation. When users finish sketching each game, the game server sends a post request with an image file attached in JSON file format to the flask server for AI operation. The flask server for AI work interprets the JSON file contained in the request sent from the game server as an image file and evaluates the accuracy value of each image after deep learning using PyTorch. These evaluated values are sent back to the game server in the form of strings. The game server then processes the responses received from the Flask server and outputs the results to the users.

5.2 Viewpoints of UX/UI

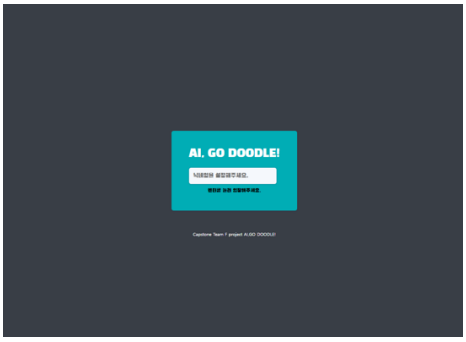

Fig. 7. Login screen is the screen that appears when a user accesses a web address for the first time. Users can choose a nickname and enter the game here.

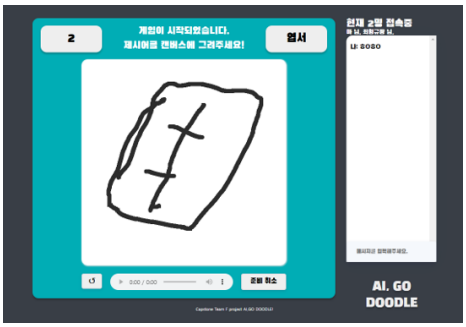

Fig. 8. Game screen is the screen that appears after the user enters a nickname in the login window. UIs are arranged around the canvas located in the center of the screen.

Fig. 9. Result screen When the game ends and the AI finishes calculating the results, the result screen appears to all players.

The screen of the game can be divided into two main parts. The screen of the game can be divided into two main parts. First, **Fig. 7** is a login screen. It is the first thing the user sees when they enter the URL of the game in the Internet window. There is a window where you can enter a nickname. Enter a nickname within 6 characters and press Enter to move to the next screen. **Fig. 8** is the main screen of the game. In the center of the screen is a canvas on which to draw a picture. A timer, a guide message, and a keyword to draw are displayed at the top centering on this canvas.

The information of currently connected users is displayed on the right side of the canvas, and below that is a chat window where you can communicate in real-time. At the bottom of the canvas, there is a remote control to play or stop bgm, a canvas reset button, and a ready button.
After the 10-second timer expires, there is a short waiting time while the AI scores the drawings drawn by users. When the result is displayed, the result window as shown in **Fig. 9** is displayed on all screens of the players. The resulting window informs the winner's nickname and score, and if one person presses the redo button below it, the game returns to the initial screen and prepares for the next game.

5.3 Socket communication

Web socket communication was the most focused part of the game implementation. The server and each of the browsers can generate a socket event or process the generated socket event. For example, when a user sends a chat message from a browser, the browser generates a predefined socket event called "sending", and the server processes the event to receive the transmitted chat content. Thereafter, the server generates another event called "newMsg", containing the received chat message, and sends it to all browsers except the browser that generated the "sendMsg" event. Finally, browsers that receive a "newMsg" event including a chat message from the server adds it to the chat window. Through this flow, users can chat in real-time.
User notifications, game timers, and getting pictures of each browser to the server at once were also implemented through socket communication between the server and the browser.

5.4 Dataset

We adopted the QuickDraw dataset as our dataset. The QuickDraw dataset has 345 classes of doodling, each with more than 100,000 image data. We used 100 classes, each class with 5,000 images. The classes are selected by considering the combination that makes the model show as high accuracy as possible. Since the image of the dataset is too small, the model showed low performance if trained with more than 50 keywords. The QuickDraw dataset provides image data of size 32 x 32 in a npy format. **Fig. 10** shows the example data of the QuickDraw dataset.

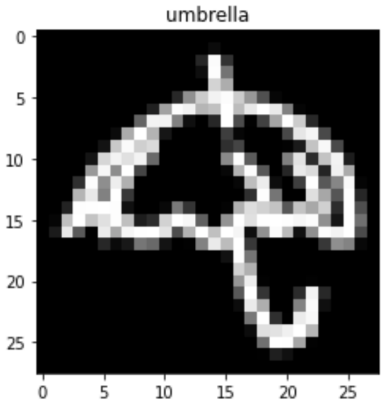

Fig. 10. QuickDraw data example The data is preprocessed into 32 x 32 pixel size.

5.5 Model

We used ResNet34 model since it showed fast convergence during the training and good performance on our dataset. We trained each of the 4 independent models for 80 epochs with 25 classes.

We used a learning rate of 1e-4, batch size of 32, categorical cross-entropy for the loss function, and Adam optimizer. **Fig. 11** shows the confusion matrix of each model after completion of the training. Also, **Fig. 12** shows the training curve of each model.

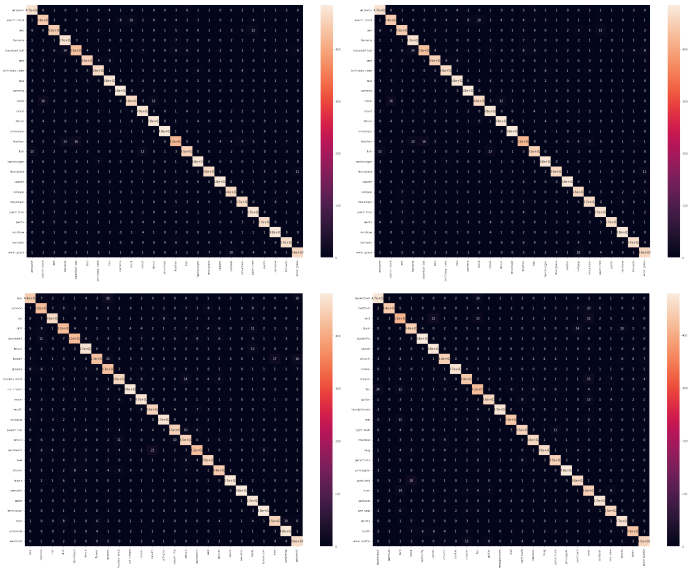

Fig. 11. Confusion matrix of each model The confusion matrix shows the correlation of each keyword. The figure shows the confusion matrix of each model, sequentially

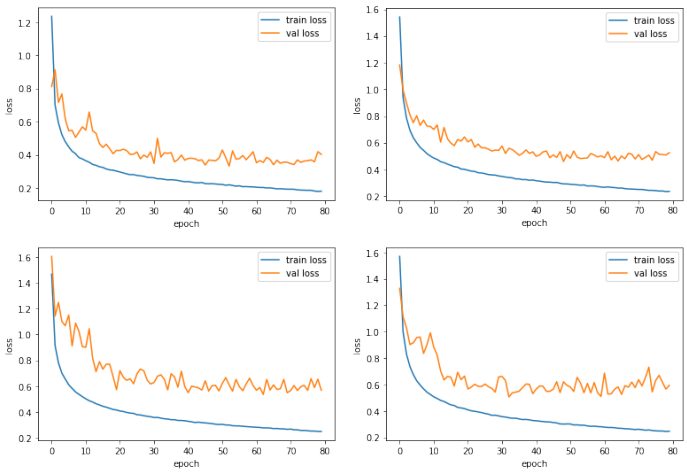

Fig. 12. Training curve of each model The training curve of each model with different keywords. Most of them show rapid convergence during the first 20 epochs and then converge around 60 epochs.

6 Evaluation

6.1 Model Accuracy

For each of the 4 models, we used 100,000 data for training, 13,000 for validation, and 12,000 for the evaluation. We used accuracy as our evaluation metric, which is the percentage of True Positive and True Negative over every prediction of the model. The **Table 1** shows the accuracy of each model.

	Model 1	Model 2	Model 3	Model 4
Acc	0.9334	0.9059	0.9080	0.9067

Table 1. The accuracy of each model. The model 1 achieved highest accuracy with 93.34% while model 2 achieved the lowest accuracy with 90.59%.

7 Limitations

7.1 Prepare for User Scenarios

This game was prepared for various user scenarios such as entering during the game, leaving during the game, leaving during the result counting, and restarting while counting the result. However, as more and more user scenarios were created, it was difficult to prepare for all situations. We think that we have prepared for situations that can be considered big, but there is a limit to preparing for all situations.

7.2 Implementing Responsive Design

Our project is a web game, so it can be played directly in the browser without specification restrictions or separate installation. However, on mobile, there are limitations such as not being able to see properly due to the aspect ratio of the screen being incorrect, or being unable to draw a picture with a touch pad.

7.3 Polarization of Scores

The image to be searched for by increasing the accuracy of the model was found with high accuracy, but if it has the characteristics of the dataset of the image to be searched, the accuracy is displayed as almost 99%, otherwise the accuracy is close to 0%. So there is a limit in expressing this as a game score there was. We tried to express the user's score so that it feels as if it was actually scoring rather than simply with accuracy, but there was a limit to scoring in other ways, so in the end, the score was displayed with accuracy.

7.4 Dataset Limitations

The quick draw dataset has more than 50 million drawings across 345 categories. However, the dataset we used for our model has 100 categories and an image size of 32 by 32. We wanted to use more categories, more drawings, and better quality datasets for rich content and high-quality games. However, due to the cost aspect, time limit, lack of data, etc., it was produced using the current dataset.

8 Conclusion

Our project utilizes the ResNet34 model learned from the quick draw dataset to provide a web game where two or more people can participate and communicate and enjoy in real-time. Among the existing web games, there were hardly any games made using AI. Also, even if it was created using AI, there was no web game that multiple people could enjoy together. Therefore, AI, GO DOODLE helps users to enjoy the game for free, easy and convenient, and to feel more familiar with AI technology. We've also created it so that many people can log on at the same time and have fun chatting with each other, so you can enjoy our game whether it's for a quick break or a light bet. It has been designed with an intuitive design so that there is no difficulty in playing, and login is also made simple so that you can access it immediately by entering your ID. To make the game more exciting while playing, we have added a fun BGM that is repeated over and over again, and each game is played quickly so that you can enjoy many games in a short amount of time.

References

1. google. Quick, draw! the data. https://quickdraw.withgoogle.com/data/. [Online; accessed 2-Jun-2022].
2. netmarble. Catch mind. https://cmind.netmarble.net/main.asp. [Online; accessed 2-Jun-2022].
3. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
4. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
5. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
6. Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.