ISYE 6740 - Spring 2023
Final Project Report

Team Member Name:	Oanh Doan
Project Title:	The Mario Kart Image Classifier: A Race to Accuracy

1. Problem Statement

Mario Kart is a wildly popular racing game on Nintendo Switch, featuring a diverse cast of 42 characters, 8046 vehicle combinations, and 48 racing circuits, each with unique attributes. With such a wide array of character and vehicle design options, the game offers endless possibilities that are visually captivating. The objective of this project is to develop image classification models for Mario Kart characters, which will be riding different vehicles in distinctive background settings. The project will evaluate and compare the performance of k-nearest neighbor classifier - a representative of traditional machine learning methods, and that of a state-of-the-art deep learning algorithm, namely Convolution Neural Network (CNN) [1].

2. Data Source

2.1. Data Collection

To collect data for this project, we developed a Python script to download videos of game playthrough from YouTube and extract screenshots of the characters featured in these videos. For each character, we collected training images from one video and testing images from a different video. This separation is so that the sets of training and testing images are fully disjoint.

Next, we cropped each image to the center, reducing its dimensions from 720×1280 pixels to 360×427 pixels. The goal was to focus solely on the characters in each image, excluding as much of the background as possible, thereby reducing noise and distractions that could interfere with model performance. It is reasonable to perform this operation because in Mario Kart, all characters are depicted from a third-person perspective and maintain a relatively fixed position within the image frame throughout the game. Figure 1 provides an illustration of this cropping.

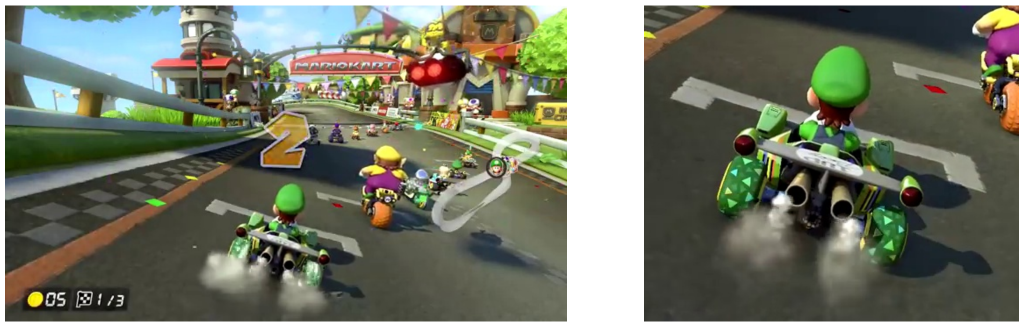
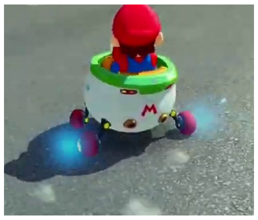


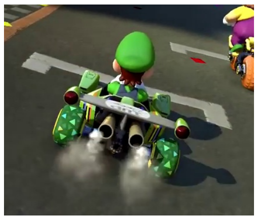
Figure 1: A full-size image and a cropped image of Luigi, respectively

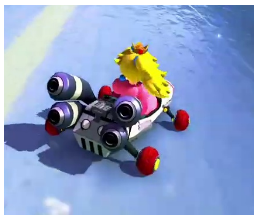
Finally, we manually reviewed the results to exclude any irrelevant images such as transition screens and scoreboards. Removing such irrelevant images helped reduce noise and enhance the models' accuracy.

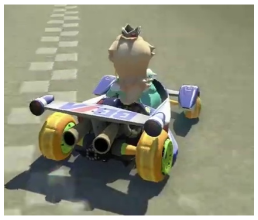
2.2. Data Description

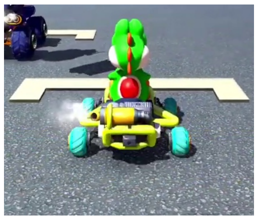
The dataset contains 8 classes corresponding to 8 characters in the game. Each class has about 350 training images and 150 testing images, with each image of dimension 360×427. The eight classes formed four pairs of characters, each having a high degree of similarity. These four pairs are Mario vs. Luigi, Peach vs. Rosalina, Yoshi vs. Birdo, and Bowser vs. Junior Bowser (Figure 2).

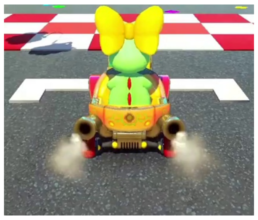

(a) Mario

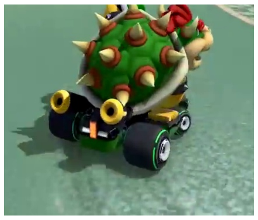

(b) Luigi


(c) Peach


(d) Rosalina


(e) Yoshi


(f) Birdo


(g) Bowser

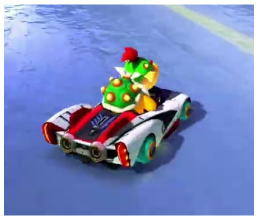
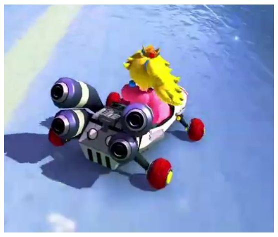

(h) Jr.Bowser

Figure 2: Eight classes

Furthermore, for each character, we made the vehicles in training and testing data different (Figure 3). To make it more interesting, for animal-class characters such as Yoshi and Birdo, we made the training and test characters of different skin colors. For example, Yoshi in training data is green whereas Yoshi in testing data is orange (Figure 4). This choice makes classification more challenging as the models cannot rely on just colors.


(a) Training image

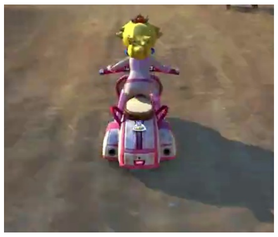
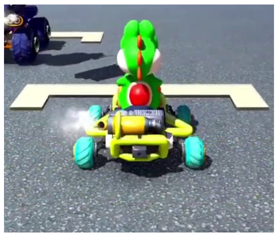

(b) Testing image

Figure 3: Peach riding different vehicles


(a) Training image

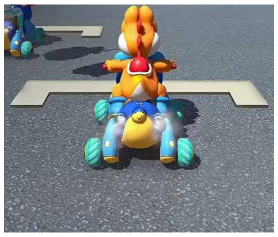

(b) Testing image

Figure 4: Yoshi with different skin colors

3. Methodology

3.1. Methods

1. Traditional machine learning

   We picked K-nearest neighbor to represent the class of traditional machine learning algorithms as KNN allows for a non-linear, flexible decision boundary. The input data to the model are image vectors, with each image corresponding to a vector in $\mathbb{R}^{461160}$ where $461160 = 360 \text{ pixels} \times 427 \text{ pixels} \times 3 \text{ RGB}$.

2. Deep learning: Convolutional Neural Networks using transfer learning [1]

   Convolutional Neural Networks (CNNs) are a type of deep learning model that has revolutionized the field of Computer Vision. Unlike traditional neural networks, CNNs are designed to effectively process and analyze visual data, such as images and videos, by leveraging a large amount of training data. Compared to the neural networks we have seen in the lecture notes so far, state-of-the-art CNNs use special layers and operations such as convolutional layers and pooling layers to extract high-level features from raw input pixels. CNNs have been widely used in various applications such as image classification and object recognition, and have also been shown to outperform traditional techniques in several benchmarks.

   In this project, our data sample is rather small to train the network from scratch. Fortunately, we can leverage the learning result of a convolutional neural network pre-trained on a large image dataset of 1.2 million images and 1000 classes. Then, we can "finetune" the model using our own images to learn additional Mario Kart-specific features. This allows us to conduct our image classification task with a much smaller set of data, in the order of a few hundreds images. This idea was inspired by the ant-vs-bee image classification project [1], which achieved an accuracy of over 90% using a dataset of only 200 images.

3.2. Experiment Design

In this project, we conducted three experiments that were progressively more challenging. For each experiment, we adjusted the parameters of the two models mentioned earlier and then compared their performance. The parameters subject to change were the number of nearest neighbors in KNN, and the number of epochs and learning rates in CNN. Our goal is to understand how the optimal parameters and model performance would change as the classification problem became increasingly difficult. Below is an outline of the three experiments.

3.2.1. Experiment 1: Simple binary classification.

To simplify the classification task in this experiment, we chose to focus on Luigi (4a) and Bowser (2g) as they possessed distinct physical attributes that made them easily distinguishable. For instance, one characteristic that sets Bowser apart from Luigi is Bowser's abundance of thorns on his back. Below are the specifics regarding parameter optimization.

- For KNN: Search for the optimal number of nearest neighbors k for $k \in [1, 30]$.
- For CNN: Search for the number of epochs e at which the model starts to converge ($e \in [1, 25]$).
- For CNN: Using the number of epochs above, evaluate the model performance at different values of learning rates λ for $\lambda \in \{0.0001, 0.001, 0.01\}$.

3.2.2. Relatively more challenging binary classification.

In the second experiment, we would apply KNN and CNN models to four relatively more challenging binary classification problems, each corresponding to one pair of characters introduced in Figure 2. The parameter optimization details are the same as Experiment 1.

3.2.3. Experiment 3: Multi-class classification

Finally, we would use KNN and CNN models to classify all eight classes at once. However, due to the problem's increased complexity, we would have to downsize the experiment by reducing the search range for k in KNN and the number of options for λ in CNN. The parameter optimization details are as follows.

- For KNN: Search for the optimal number of nearest neighbors k for $k \in [1, 20]$.
- For CNN: Evaluate the model performance after 25 epochs at two different values of $\lambda \in \{0.001, 0.0001\}$. We skip the learning rate $\lambda = 0.01$ as we believe a more complex problem generally requires a smaller step size for the model.

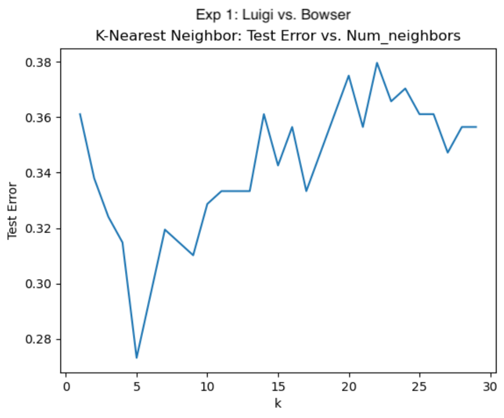
4. Results and Discussion

We would use test errors as the main metric to answer the following questions.

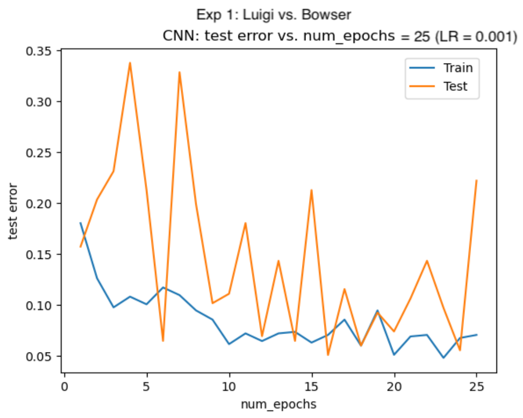
- For each experiment, identify the optimal parameters (if possible) and associated model performance.
- As the classification problem becomes progressively challenging, how do the optimal parameters and model performance change?

4.1. Experiment 1

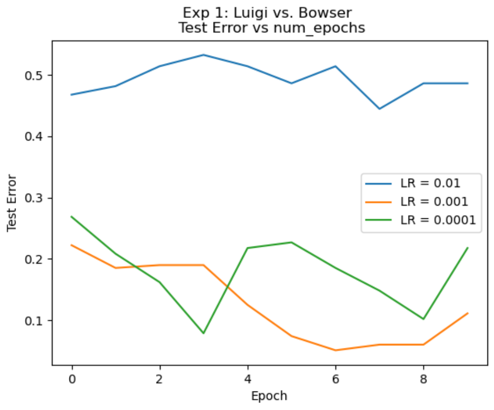
Figure 5a displayed the test error rates for KNN when parameter k was adjusted between 1 and 30. The lowest error rate of 28% occurred when $k = 5$. Figure 5b showed the model performance when the number of epochs was increased from 1 to 25 while fixing $\lambda = 0.001$. Although some fluctuation in test errors persisted until the last epoch, the lowest error rate was achieved after 6 epochs and the fluctuation in error rates decreased after 10 epochs. Therefore, we used 10 epochs to tune learning rates as shown in Figure 5c. At $\lambda = 0.01$, we observed that the step size was too large for the model to converge, causing accuracy to fluctuate around 50%. Similarly, at $\lambda = 0.0001$, the step size was so small that the model might have become trapped in a local minimum or experienced slow convergence. Thus, the optimal learning rate was determined to be 0.001, which allowed the model to converge after just 6 epochs with an error rate of 10%. At the optimal parameters, the test error rate was 28% for KNN ($k = 5$) and 10% for CNN (epoch = 6, $\lambda = 0.001$). Thus, the CNN model outperformed KNN for this set of images.



(a) KNN - Tune k



(b) CNN - Tune epoch



(c) CNN - Tune LR

Figure 5: Experiment 1 - Results for image classification with Luigi and Bowser

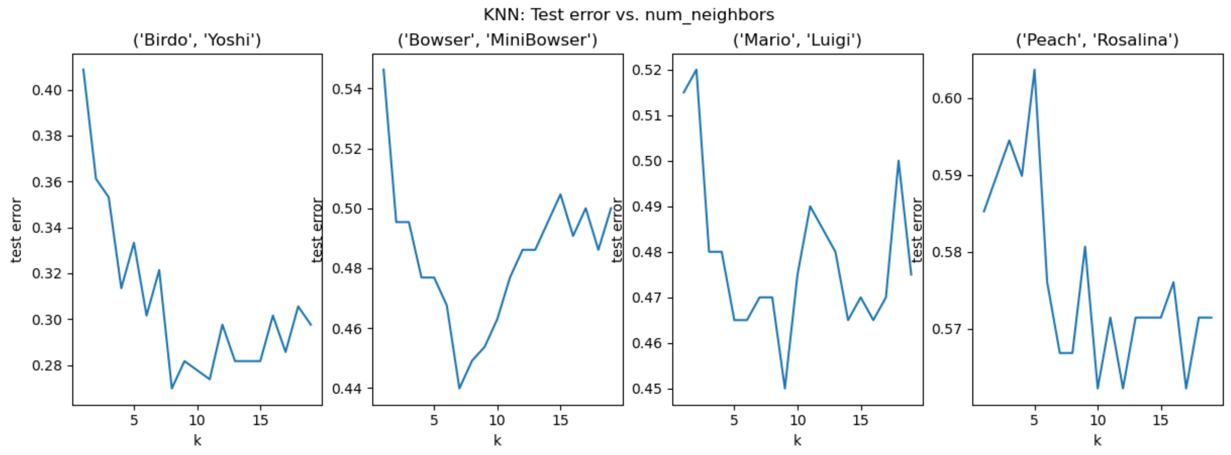


Figure 6: Experiment 2 - Test errors of KNN with different class pairs and k values

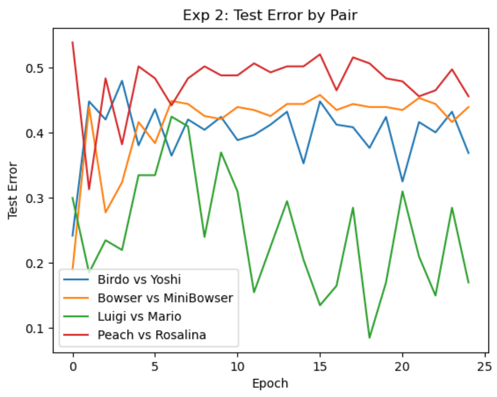
4.2. Experiment 2

In this experiment, we aimed to classify four sets of images that are more challenging to distinguish than the ones in Experiment 1. Table 1 summarizes the experiment's results, indicating the lowest test errors and associated parameters for each model and each pair of characters.

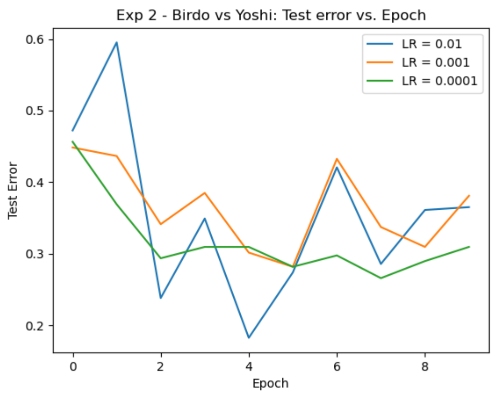
As expected, the error scores worsened across all models due to the images' higher similarity. For KNN, the test score rose from 28% to 50%, and the optimal k value increased from 5 to 10 (figure 6). For CNN, interpretation of the results is more nuanced. Figure 7a showed that the models converged after 10 epochs at a fixed learning rate of $\lambda = 0.001$. At convergence, the test scores ranged from 20% for Luigi vs. Mario to $40 - 50\%$ for the other three pairs. These scores were generally worse than the performance achieved in Experiment 1 (figure 5b). Even after testing various learning rates and handpicking the lowest test scores, as shown in Table 1, the results were still relatively worse. Furthermore, it isn't obvious what the optimal parameters for the CNN models in this experiment are because the models have not reached convergence in some cases. To make a definite claim, we would probably need to increase the number of epochs for smaller learning rates such as $\lambda = 0.0001$. However, even at a possibly sub-optimal choice of parameters for CNN, we can still claim that CNN continues to outperform KNN in this experiment.

	KNN		CNN	
Pair	Lowest error	Parameters	Lowest error	Parameters
Birdo vs. Yoshi	26%	$k = 8$	20%	epoch = 4, $\lambda = 0.0001$
Bowser vs. MiniBowser	44%	$k = 7$	25%	epoch = 3, $\lambda = 0.001$
Mario vs. Luigi	45%	$k = 9$	14%	epoch = 3, $\lambda = 0.001$
Peach vs. Rosalina	50%	$k = 10$	32%	epoch = 5, $\lambda = 0.001$

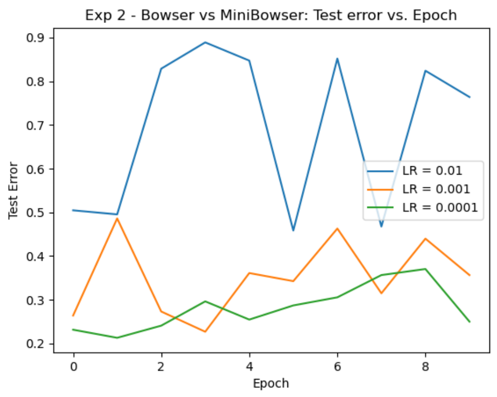
Table 1: Experiment 2 - Summary of KNN and CNN results across different class pairs



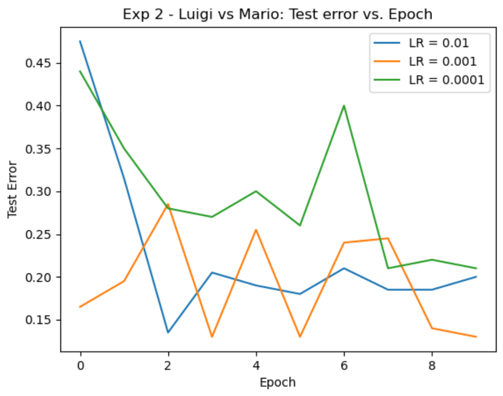
(a) CNN - Tune epoch



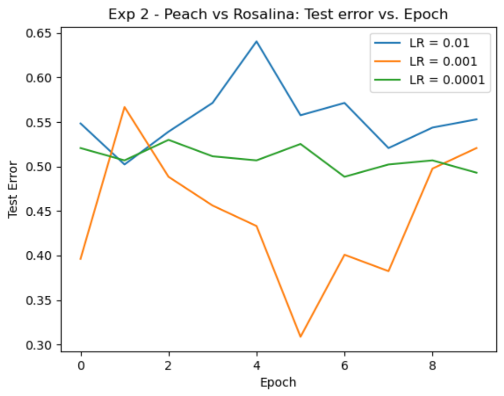
(b) CNN Birdo vs. Yoshi - Tune LR



(c) CNN Bowser vs. Jr Bowser - Tune LR



(d) CNN Luigi vs. Mario - Tune LR



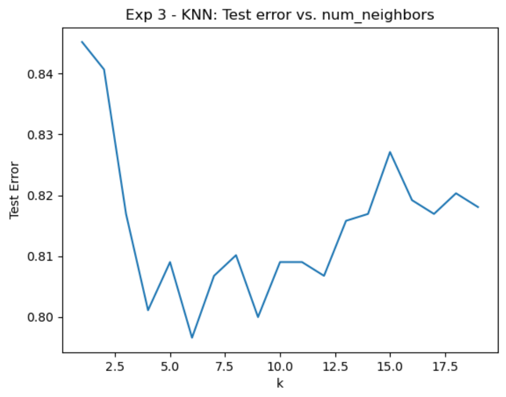
(e) CNN Peach vs. Rosalina - Tune LR

Figure 7: Experiment 2 - Results for CNN across different class pairs

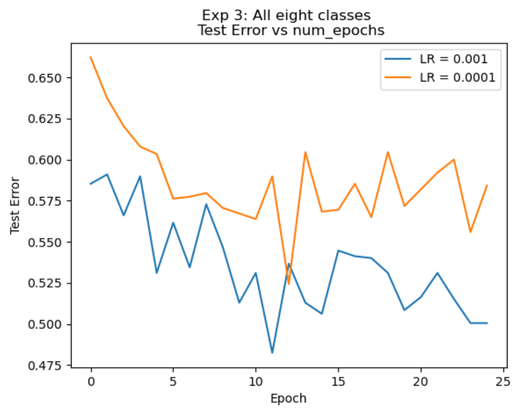
4.3. Experiment 3

In the final experiment, we aimed to classify all eight classes of characters using both KNN and CNN models. However, due to the scale of the problem, we had to downsize the experiment. For KNN, we tested only k for $k \in [1, 20]$, while for CNN, we ran 25 epochs with two learning rates $\lambda \in \{0.001, 0.0001\}$. Given Experiment 2 results, we decided to skip the learning rate $\lambda = 0.01$ because we believed it wouldn't perform well in a problem relatively more challenging than the previous experiments.

Figure 8 showed the experiment results. For KNN, as shown in figure 8a, $k = 6$ still yielded the best performance, but the test score was extremely high (80%). As there were 8 classes in the data, the probability of correctly guessing the true label of a data point was 12.5%. This means that the KNN model performed only slightly better than a random guess. For CNN, figure 8b showed that the model required slightly more than 10 epochs to reach convergence (higher than previous experiments), and that the best test score was 48%, which was achieved after 11 epochs at $\lambda = 0.001$. It is interesting that the model performance did not significantly degrade compared to Experiment 2 even though we added 3 times more the number of classes.



(a) KNN



(b) CNN

Figure 8: Experiment 3 Results

5. Conclusion

In conclusion, after comparing the performance of a convolutional neural network (CNN) model and a k-nearest neighbor (KNN) model, we found that the CNN model outperformed the KNN model. The CNN model was able to achieve a higher accuracy and a lower error rate on the test set. This indicates that the CNN model was able to capture more complex features and patterns in the data than the KNN model.

References

[1] Sasank Chilamkurthy. Transfer learning for computer vision tutorial. `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html#load-data`, 2018. Accessed on March 31, 2023.