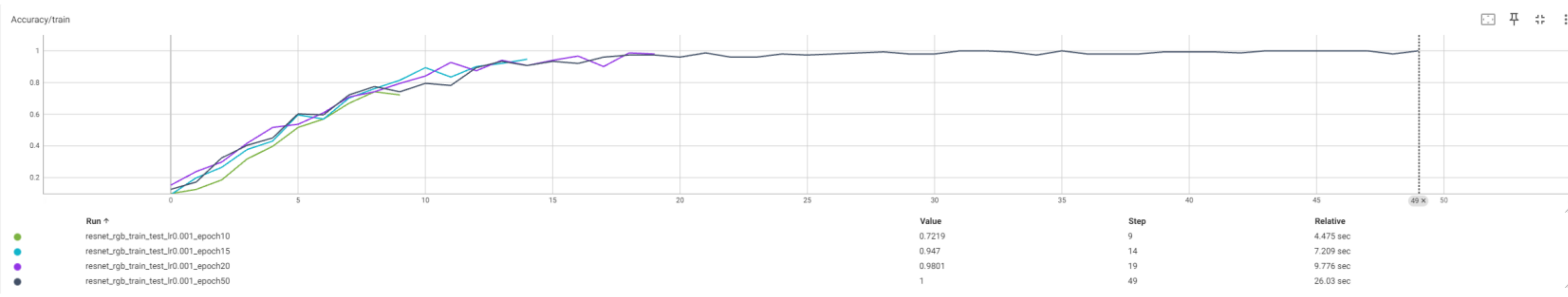
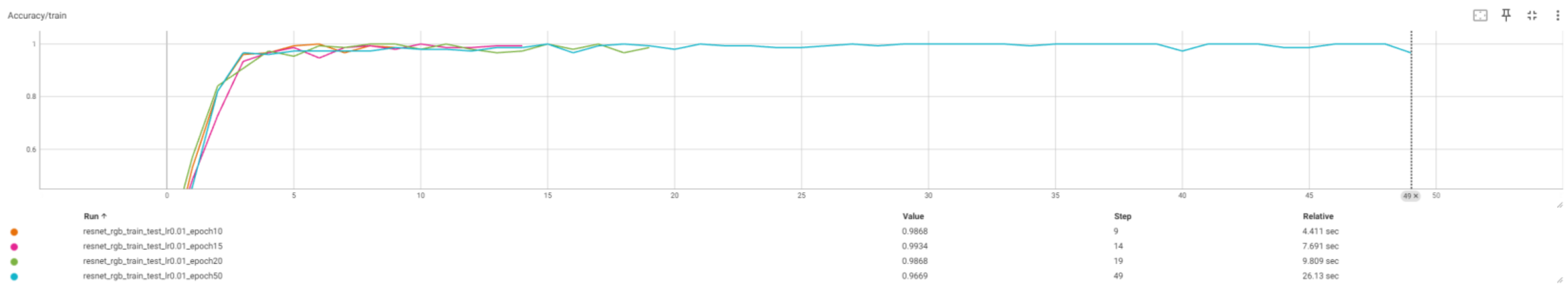
AI capstone project 1

1. Research target: try to use some ML/DL method to classify the dataset containing the cards in BangDream! Game.
2. Dataset:
    a. The dataset containing 189 RGB images (thumbnail of cards) with shape (180 * 180).
    b. Data source: Bestdori!
    c. Labels: the dataset containing 10 different characters in the game, with 10~20 images collected for each character.
3. Methods:
    a. For supervised learning: I use RESNET-18 and HOG+SVM to classify the images.
    b. For unsupervised learning: I convert the images to grayscale and use k-means to cluster them.
4. References:
    a. PyTorch as the main framework.
    b. Sklearn, skimage for k-means, SVM, and HOG.
    c. Some code is slightly modified from the help of ChatGPT.
5. Parameters:
    a. SVM: Linear kernel
    b. RESNET: combination of learning rate = [0.1, 0.01, 0.001] and epochs = [10, 15, 20, 50].
6. Results:
    a. RESNET with 80% train 20% test split
        i. Training accuracy with learning rate = 0.001 and different epochs.



	Run ↑	Value	Step	Relative
🟢	resnet_rgb_train_test_lr0.001_epoch10	0.7219	9	4.475 sec
🔵	resnet_rgb_train_test_lr0.001_epoch15	0.947	14	7.209 sec
🟣	resnet_rgb_train_test_lr0.001_epoch20	0.9801	19	9.776 sec
●	resnet_rgb_train_test_lr0.001_epoch50	1	49	26.03 sec

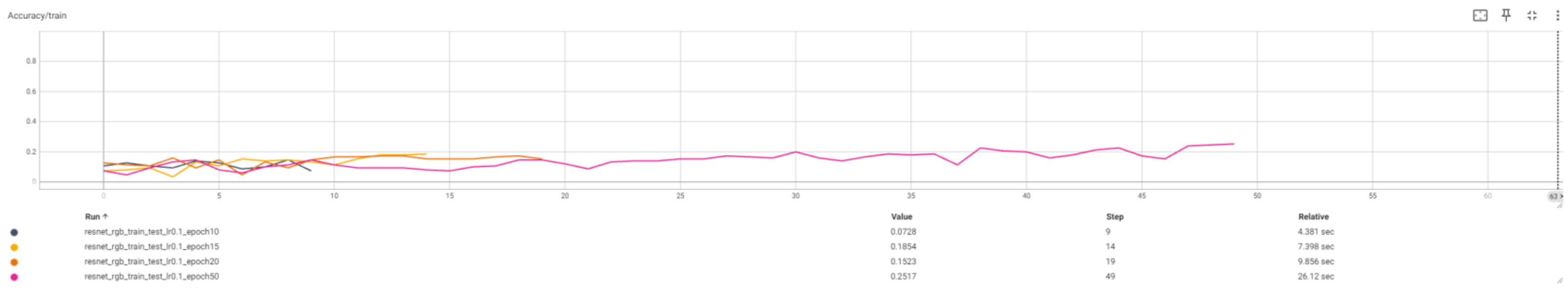
        ii. Training accuracy with learning rate = 0.01 and different epochs.



	Run ↑	Value	Step	Relative
🟠	resnet_rgb_train_test_lr0.01_epoch10	0.9868	9	4.411 sec
🔴	resnet_rgb_train_test_lr0.01_epoch15	0.9934	14	7.691 sec
🟢	resnet_rgb_train_test_lr0.01_epoch20	0.9868	19	9.809 sec
🔵	resnet_rgb_train_test_lr0.01_epoch50	0.9669	49	26.13 sec

iii. Training accuracy with learning rate = 0.1 and different epochs.

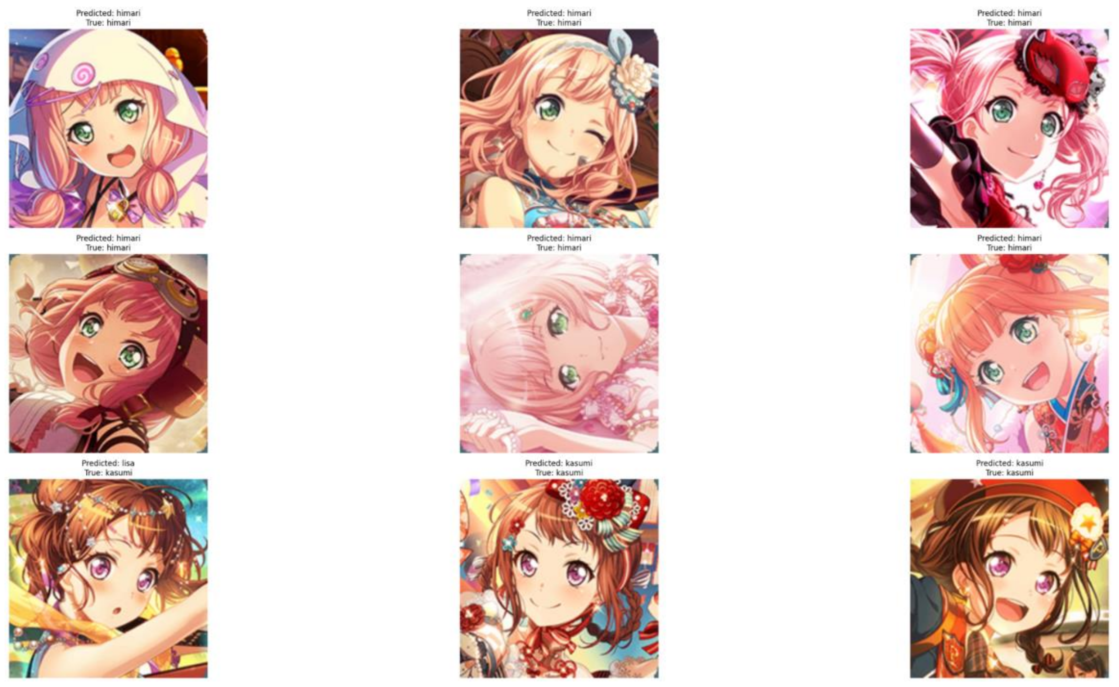
iv. Test Accuracy Result Chart



	Run ↑	Value	Step	Relative
⚫	resnet_rgb_train_test_lr0.1_epoch10	0.0728	9	4.381 sec
🟡	resnet_rgb_train_test_lr0.1_epoch15	0.1854	14	7.398 sec
🟠	resnet_rgb_train_test_lr0.1_epoch20	0.1523	19	9.856 sec
🔴	resnet_rgb_train_test_lr0.1_epoch50	0.2517	49	26.12 sec

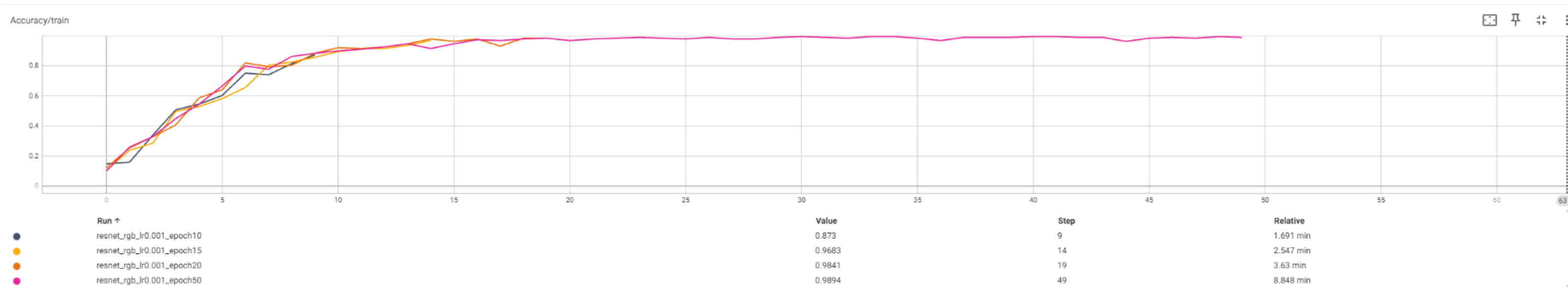
	Lr=0.001	0.01	0.1
Epoch=10	34.2%	60.5%	18.4%
15	47.3%	71.5%	21.5%
20	44.7%	65.7%	7.8%
50	57.8%	63.1%	28.9%

v. Example Result (whole image on GitHub)



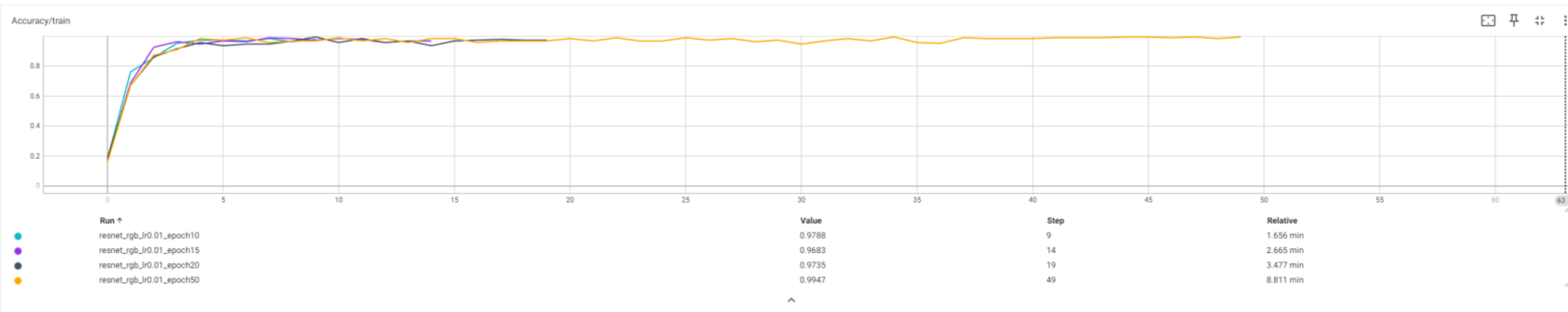
b. RESNET without train test split

i. Training accuracy with learning rate = 0.001 and different epochs.



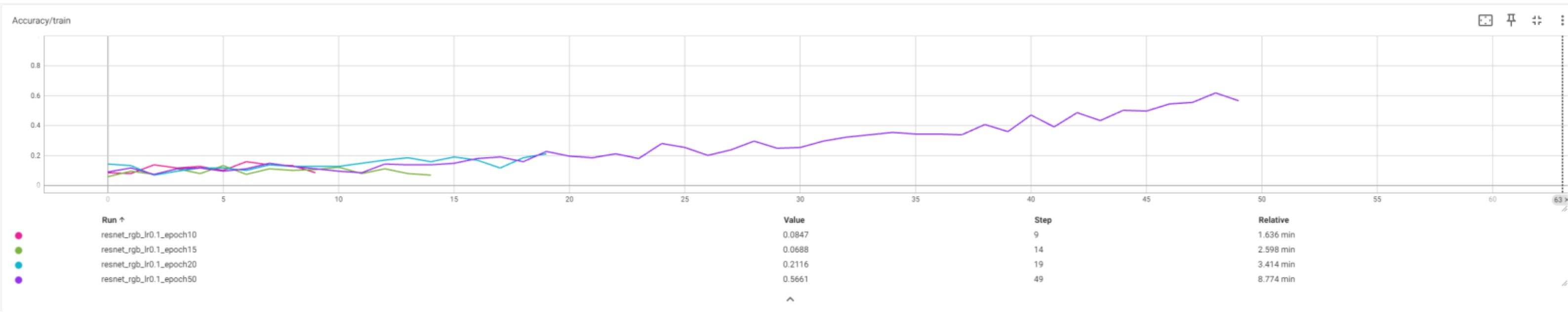
	Run ↑	Value	Step	Relative
●	resnet_rgb_lr0.001_epoch10	0.873	9	1.691 min
●	resnet_rgb_lr0.001_epoch15	0.9683	14	2.547 min
●	resnet_rgb_lr0.001_epoch20	0.9841	19	3.63 min
●	resnet_rgb_lr0.001_epoch50	0.9894	49	8.848 min

ii. Training accuracy with learning rate = 0.01 and different epochs.



	Run ↑	Value	Step	Relative
●	resnet_rgb_lr0.01_epoch10	0.9788	9	1.656 min
●	resnet_rgb_lr0.01_epoch15	0.9683	14	2.665 min
●	resnet_rgb_lr0.01_epoch20	0.9735	19	3.477 min
●	resnet_rgb_lr0.01_epoch50	0.9947	49	8.811 min

iii. Training accuracy with learning rate = 0.1 and different epochs.

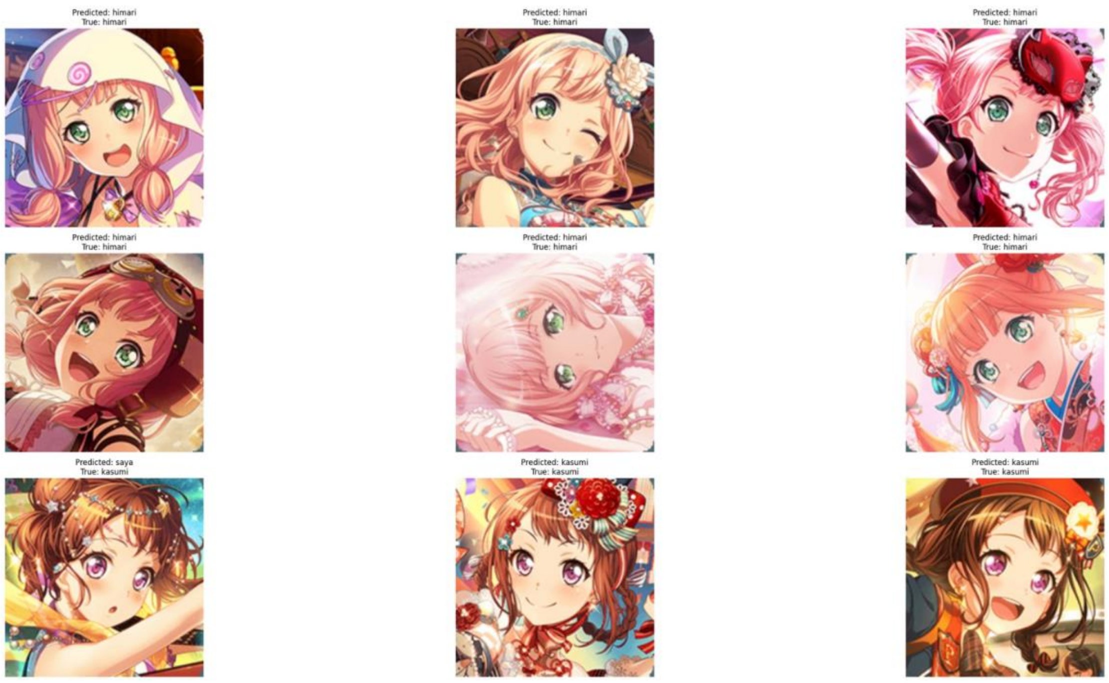


	Run ↑	Value	Step	Relative
●	resnet_rgb_lr0.1_epoch10	0.0847	9	1.636 min
●	resnet_rgb_lr0.1_epoch15	0.0688	14	2.598 min
●	resnet_rgb_lr0.1_epoch20	0.2116	19	3.414 min
●	resnet_rgb_lr0.1_epoch50	0.5661	49	8.774 min

iv. Ending Accuracy Result Chart

	Lr=0.001	0.01	0.1
Epoch=10	87.3%	97.8%	8.4%
15	96.8%	96.8%	6.8%
20	98.4%	97.3%	21.1%
50	98.9%	99.4%	56.6%

v. Example Result (whole image on GitHub)

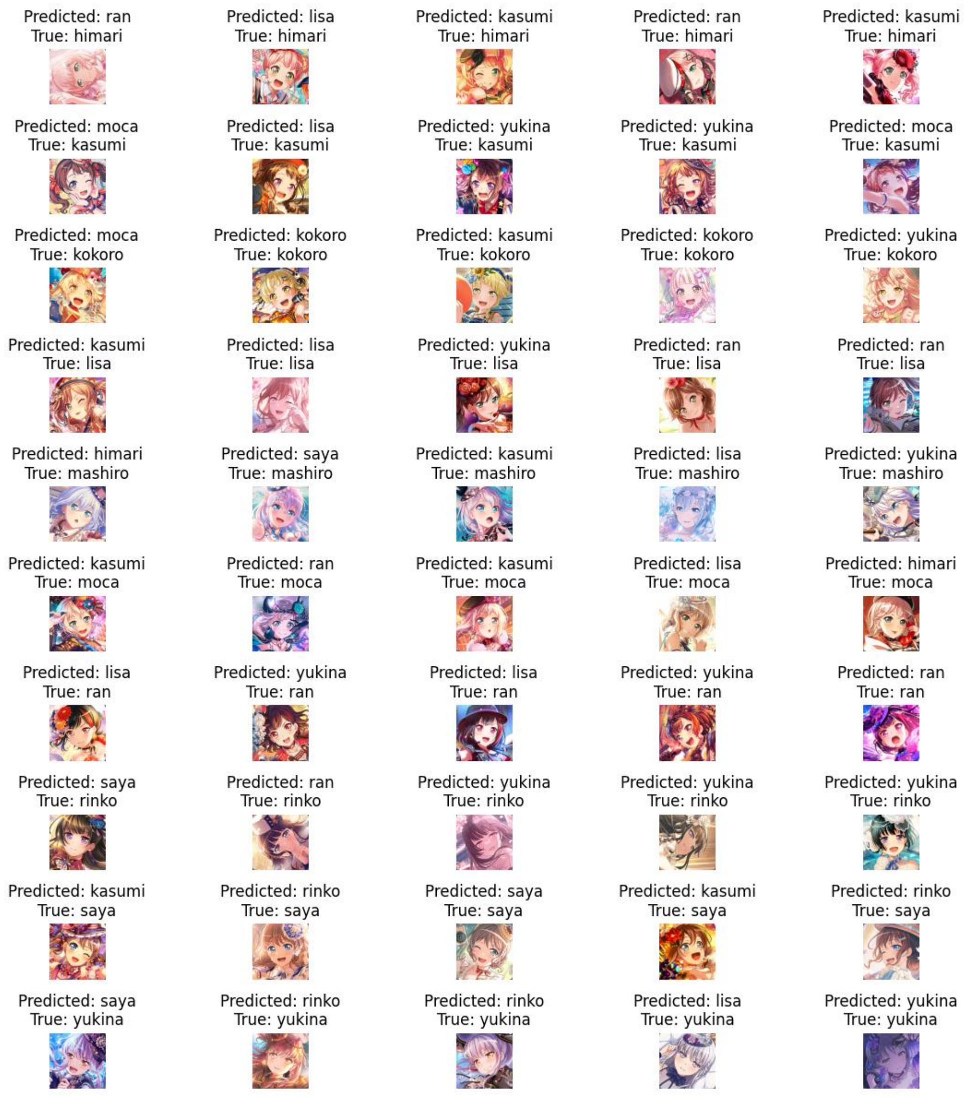


c. HOG+SVM with cross validation

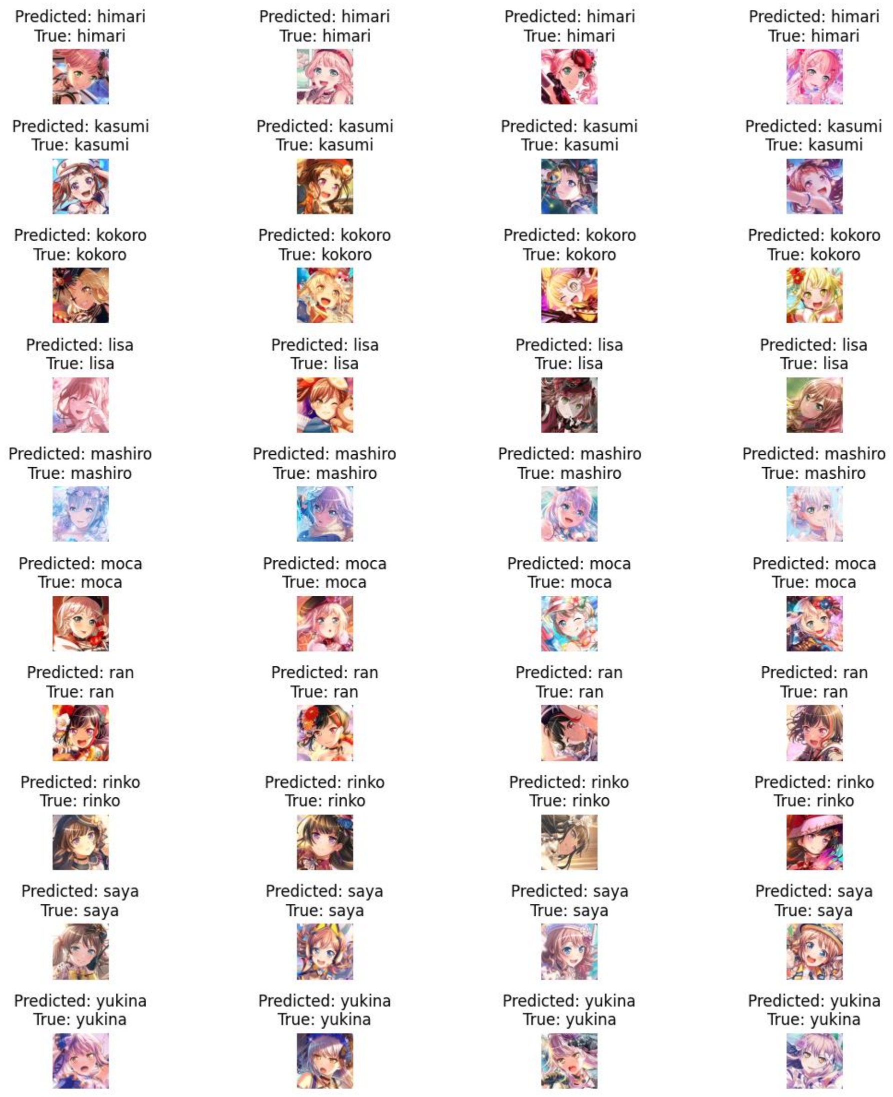
i. 5-fold cross validation accuracy:

23.6%	21.0%	5.2%	34.2%	18.9%
Mean:	20.6%			

ii. Example Result:

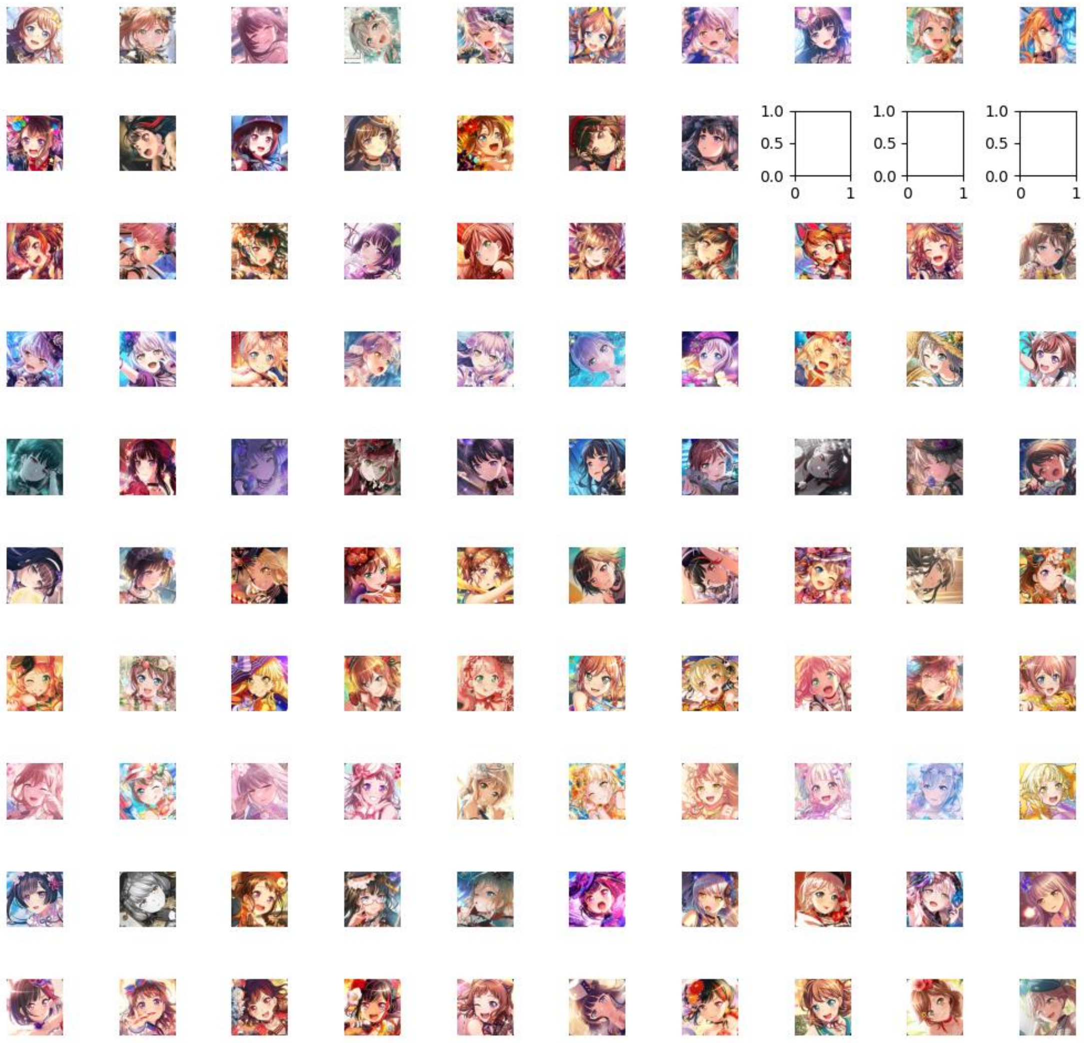

Predicted: ran True: himari
Predicted: lisa True: himari
Predicted: kasumi True: himari
Predicted: ran True: himari
Predicted: kasumi True: himari
Predicted: moca True: kasumi
Predicted: lisa True: kasumi
Predicted: yukina True: kasumi
Predicted: yukina True: kasumi
Predicted: moca True: kasumi
Predicted: moca True: kokoro
Predicted: kokoro True: kokoro
Predicted: kasumi True: kokoro
Predicted: kokoro True: kokoro
Predicted: yukina True: kokoro
Predicted: kasumi True: lisa
Predicted: lisa True: lisa
Predicted: yukina True: lisa
Predicted: ran True: lisa
Predicted: ran True: lisa
Predicted: himari True: mashiro
Predicted: saya True: mashiro
Predicted: kasumi True: mashiro
Predicted: lisa True: mashiro
Predicted: yukina True: mashiro
Predicted: kasumi True: moca
Predicted: ran True: moca
Predicted: kasumi True: moca
Predicted: lisa True: moca
Predicted: himari True: moca
Predicted: lisa True: ran
Predicted: yukina True: ran
Predicted: lisa True: ran
Predicted: yukina True: ran
Predicted: ran True: ran
Predicted: saya True: rinko
Predicted: ran True: rinko
Predicted: yukina True: rinko
Predicted: yukina True: rinko
Predicted: yukina True: rinko
Predicted: kasumi True: saya
Predicted: rinko True: saya
Predicted: saya True: saya
Predicted: kasumi True: saya
Predicted: rinko True: saya
Predicted: saya True: yukina
Predicted: rinko True: yukina
Predicted: rinko True: yukina
Predicted: lisa True: yukina
Predicted: yukina True: yukina

d. HOG+SVM without split
   i. Accuracy: 100%
   ii. Example Result:


Predicted: himari True: himari
Predicted: himari True: himari
Predicted: himari True: himari
Predicted: himari True: himari
Predicted: kasumi True: kasumi
Predicted: kasumi True: kasumi
Predicted: kasumi True: kasumi
Predicted: kasumi True: kasumi
Predicted: kokoro True: kokoro
Predicted: kokoro True: kokoro
Predicted: kokoro True: kokoro
Predicted: kokoro True: kokoro
Predicted: lisa True: lisa
Predicted: lisa True: lisa
Predicted: lisa True: lisa
Predicted: lisa True: lisa
Predicted: mashiro True: mashiro
Predicted: mashiro True: mashiro
Predicted: mashiro True: mashiro
Predicted: mashiro True: mashiro
Predicted: moca True: moca
Predicted: moca True: moca
Predicted: moca True: moca
Predicted: moca True: moca
Predicted: ran True: ran
Predicted: ran True: ran
Predicted: ran True: ran
Predicted: ran True: ran
Predicted: rinko True: rinko
Predicted: rinko True: rinko
Predicted: rinko True: rinko
Predicted: rinko True: rinko
Predicted: saya True: saya
Predicted: saya True: saya
Predicted: saya True: saya
Predicted: saya True: saya
Predicted: yukina True: yukina
Predicted: yukina True: yukina
Predicted: yukina True: yukina
Predicted: yukina True: yukina

e. K-means with 10 clusters
   i. Accuracy: 2.6%
   ii. Example Result:

Clustered Images



7. Discussion
   a. Because I know classifying these images is quite a difficult job, especially for the models that are not CNN-based. So, I don’t expect those models perform good.
   b. For RESNET, it could extract the features like the color accent, the eyes, the hairstyle, to classify the images. As you can see, if we use same dataset for training and testing, it can nearly reach 100% accuracy and this is the job what we need to do in reality (because we can get all available cards’ thumbnail and train on it). Although I’ve trained it on train test split, it still performs well (around 60%).

   c. For SVM+HOG, it can classify the train set well, but since the HOG feature extractor is trained on gray scale image, so it can’t extract those features I’ve mentioned above well. Thus, it performs badly when encountering the images other than training images. As for the case when all images are known, it can perform perfectly to classify all the images, so this could be the best to conduct the game-card classification job in reality.
   d. For k-means, it could cluster the images based on the color accent, but not learning the features of the characters, so its performance is surely bad.
   e. If I could have more time, I want to experience more unsupervised learning to classify these images. Since I’ve searched and found some deep learning based unsupervised image clustering model, but I don’t have time to dig into and understand the background, so I haven’t taken those methods into experience.
   f. From this project, I’ve learned about how to use the pretrained feature extractor (RESNET) and train on another dataset, and this is also first time for me to use HOG to extract features.
   g. Remaining problem could be how to use the model to classify the unseen images, because training could be expensive and time-consuming, so maybe I need more hyperparameter tuning, or even try another Deep Learning feature extractor to train on this dataset.

8. References:

   a. Bestdori: https://bestdori.com
   b. ChatGPT: https://chat.openai.com

9.