Pokémon Card Identification Utilizing A Hybrid VGG-16 + SIFT Model

Richard Bridges
Paul Cummins
Marina Sánchez
Mileva Van Tuyl

New College of Florida
Applied Data Science Program
Class of 2023

Executive Summary

This report describes the model we collectively developed to identify queried Pokémon cards from among a reference set numbering 14,396 distinct cards as well as the experiments conducted to optimize that model. The final version features the combined use of a pretrained VGG-16 neural net model to reduce the search space from 14,396 to 50 cards per query image followed by a SIFT (Scale-Invariant Feature Transform) model to generate a ranking of the 50 most similar cards for each query image and perform the final classification. This unified model is able to correctly match a queried image to its reference image in 95.76% of instances and identifies a reference image among the top 5 matches in 100% of instances. The code for this report and the unified model is provided in the associated GitHub repository.

1. Introduction

The automatic identification of Pokémon cards is a problem that has potential business applications, considering that individual cards range in value from mere cents to hundreds of thousands of dollars. With over 14,000 different Pokémon cards, this becomes a non-trivial multiclass image classification problem. Two additional factors complicate the issue further: input noise in the data (glare, shadows, resolution, etc.), and distinguishing between near-identical cards from different release sets.

We utilized a convolutional neural network (VGG-16) and the Scale-Invariant Feature Transform (SIFT) algorithm to address these issues. We performed experiments to compare the performance of multiple models: a baseline VGG-16 model, a VGG-16 model which incorporates transfer learning, and a hybrid VGG-16 + SIFT model. We also examined the effects of augmenting SIFT descriptors in the hybrid model.

The data we used for this project includes a reference set of 14,396 images and a test set of 779 images that was provided by the company ALGRTHM. The images are of Pokémon cards with the backgrounds already removed, and have varying resolutions.

We used 3 main metrics to analyze performance: mean position, top-1% and top-5%. Mean position is the average position that our model correctly identifies the reference card in its ranked list of predictions, and is indexed at 0 (a model that perfectly predicts every test card will have a score of 0). Top-1% is the percentage of cards that are correctly classified, and top-5% is the percentage of cards that are accurately classified within the first 5 predictions. Through our experiments, we found that the hybrid VGG-16 + augmented SIFT model had the highest performance with a mean position of 0.062, and was able to identify 95.76% of test images correctly with 100% of images identified within the top-5.

2. VGG-16

2.1 Base VGG-16

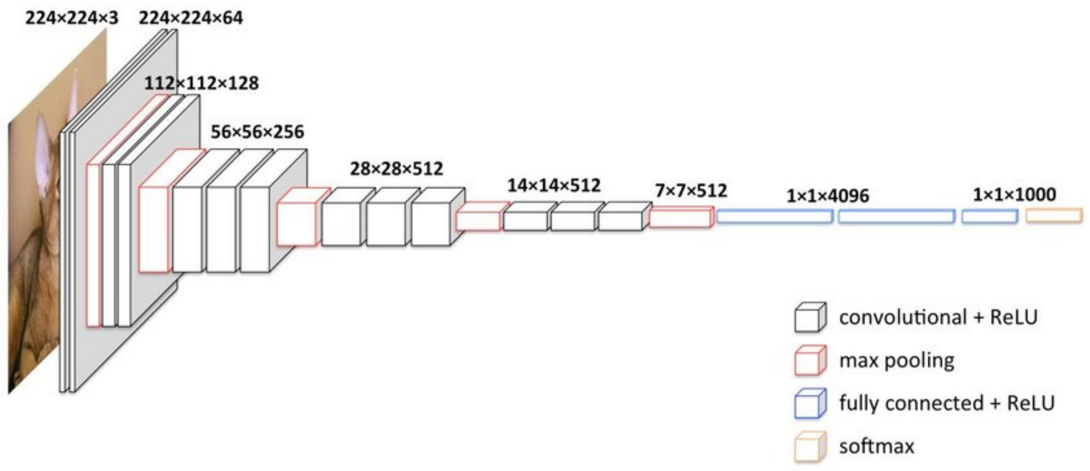

Figure 1: VGG-16 model architecture (Courtesy of Shi, et al., DOI:10.1117/12.2293594)

The base VGG-16 model (implemented in TensorFlow) was adapted for use in this project as a preliminary measure to reduce the search space of queried Pokémon cards fed into the SIFT algorithm. Its architecture comprises 21 total layers (figure 1), with 13 of those being trainable convolutional layers, each consisting of 3 x 3 filters taking 1 pixel strides. Zero-padding is enabled for each of the convolutional layers. Additionally, the VGG-16 architecture features 5 max pooling layers with each layer using a 2 x 2 pixel window taking a 2 pixel stride. The layers composing the image convolution chain begin with the two 224 x 224 x 64 convolution layers and end with the 7 x 7 x 512 max pooling layer. Following the convolutional chain, the model terminates with three fully connected layers. The first two fully connected layers have dimensions 1 x 1 x 4096 while the last has dimension 1 x 1 x 1000. All hidden layers are configured with a ReLU activation function while the output layer is configured with a softmax activation function. For additional details on VGG-16 architecture, training, and performance evaluation, refer to the original publication [VGG-16].

It was found in preliminary experiments that VGG-16 was remarkably adept at identifying features in Pokémon trading cards despite being trained on random images taken from flickr and various search engines (see the description of dataset on the Imagenet 2014 page for more details: [ImageNet 2014]. For our image matching task, VGG-16 was converted from a classifier to a feature vector extraction tool. This was done by removing the final softmax layer so that the model instead outputs a vector array descriptor of length 4096 for each input image. Briefly, the strategy employed to reduce SIFT's search space proceeded as follows. For each query image in the test set, the euclidean distance between its extracted vector and every reference image's extracted vector was calculated. The fifty reference images with the lowest distance values were then identified. The output of the VGG-16 model to be used by the SIFT algorithm is a dictionary containing the test image file names as the keys and a list of the top fifty reference image labels as each keys' values.

2.2 Transfer Learning

In order to retrain the base VGG-16 classifier to better identify Pokémon cards, we attempted to adjust the weights of the first and second fully connected layers using transfer learning. Two retraining strategies were employed. The first entailed creating a retraining set of a thousand images consisting only of the reference images corresponding to test images incorrectly identified by base VGG-16 plus randomly selected images from the reference set. The second entailed using the entire reference set as the retraining set. The first retraining strategy allowed the VGG-16 architecture to remain unchanged since the 1 x 1 x 1000 dimension output layer could be used to classify the retraining set consisting of 1000 image classes. The second retraining set, however, required that the output layer be expanded to 14396 nodes so that all image classes in the reference set could be accommodated.

Both retraining sets were extremely shallow, however, since each image class was only represented by one instance. In order to generate a more robust training set, image augmentation was employed using keras's built-in ImageDataGenerator class. Rather than create augmented image instances requiring hard drive storage, ImageDataGenerator allows random image transformations to be performed on the fly, thus allowing augmented images to be fed into the retraining model without taking up storage resources. ImageDataGenerator features a large assortment of transformation processes, from which we chose to employ random rotations between -5 and 5 degrees, translational shifts along the x and y axes between -5 and 5 percent, and brightness reduction up to 50 percent.

While the base VGG-16 model was trained using an image rescaling process that included first isotropically rescaling the images so that the shorter side measured 224 pixels in length, then cropping out any part of the image exceeding 224 pixels on the longer side, the images fed into the retrained model were non-isotropically rescaled to the required 224 x 224 dimension. This resulted in compression of the image along the longer dimension. Because the same resizing strategy was employed in the vector extraction function, the decision was made to pursue this strategy in the transfer learning experiments.

The outcomes of the retraining experiments, along with the parameters adjusted in each of the experiments are presented in Table 1:

Table 1: VGG-16 Retraining Experimental Results

Training version	Training Set (# Images)	Layers Unfrozen	Learning Rate	Optimizer	Notes	Performance (Avg ID Ranking)
V0.0 Baseline	N/A	N/A	N/A	N/A		0.348
V1.0	1000	Top two	1e-5	SGD	Augmentations: random rotations, X-Y shifts, 40 epochs	0.348
V2.0	14396	Top two	1e-5	SGD	Augmentations: random rotations, X-Y shifts, 40 epochs	0.346
V1.1	1000	Top two	1e-5	SGD	Augmentations: random rotations, X-Y shifts, brightness reduction, 40 epochs	0.348
V1.2	1000	Top one	1e-5	SGD	Augmentations: random rotations, X-Y shifts, brightness reduction, 40 epochs	0.348
V1.3	1000	Top two	1e-6	SGD	Augmentations: random rotations, X-Y shifts, brightness reduction, 40 epochs	0.348
V1.4	1000	Top one	1e-6	SGD	Augmentations: random rotations, X-Y shifts, brightness reduction, 40 epochs	0.348
V1.5	1000	Top two	1e-6	Adam	Augmentations: random rotations, X-Y shifts, brightness reduction, 40 epochs	0.348
V1.6	1000	Top two	1e-4	Adam	Augmentations: random rotations, X-Y shifts, brightness reduction, 40 epochs	0.348
V1.7	**1000**	**Top one**	**1e-4**	**Adam**	**Augmentations: random rotations, X-Y shifts, brightness reduction, 40 epochs**	**0.348**
V1.8	1000	Top one	1e-3	Adam	Augmentations: random rotations, X-Y	0.348

					shifts, brightness reduction, 40 epochs	
V1.9	**1000**	**Top one**	**1e-4**	**Adam**	**Augmentations: random rotations, X-Y shifts, brightness reduction, 100 epochs**	**0.348**

Retraining on the entire reference set took approximately 3 hours on a laptop computer equipped with an NVIDIA RTX3050Ti GPU, compared to 10 minutes on the thousand image set. Because retraining on the entire set (model version V2.0) barely improved the performance score of the image matching function (specifically it reranked the correct reference image from fifth to fourth for only one test image), subsequent experiments used the thousand image set for purposes of expediency. Ultimately, none of the retrained models were able to improve on the performance of the base VGG-16 model, and thus the decision was made to employ base VGG-16 in the combined image identification model. However, the models using the settings that induced the highest degree of convergence are indicated with bolded text.

With additional time, additional transformations could, perhaps, have been instituted to improve VGG-16's robustness against the Pokémon training set. As stated in the original publication, the base VGG-16 model was trained using horizontal flipping and random RGB color shifts, both augmentations that could be utilized in future retraining experiments. Furthermore, pytorch's counterpart to ImageDataGenerator, Torchvision Transform, encompasses a much larger assortment of options for on the fly transformations, including Sci-kit image's gaussian blur and adjust gamma functions. These features could also be tested in future retraining experiments.

3. SIFT

3.1 Introduction

The VGG-16 model narrowed the search space from 14,396 reference images to 50 reference images per query image. This enabled us to apply SIFT (Scale-Invariant Feature Transform) to optimize the results obtained from the base VGG-16 model.

The SIFT algorithm, developed by David G. Lowe extracts distinct feature descriptors that are scale and rotation invariant. Additionally, the descriptors are developed to be robust to noise and difference in viewpoint and lighting. These features make SIFT an ideal approach to address the two primary challenges: input noise associated with glare and lighting differences and nearly identical cards.

We developed a two-phase SIFT approach consisting of 1) a "training" step where we extract and store the SIFT keypoints and features for the 14,396 reference images, and 2) a “prediction” step that uses a matching process to provide a ranking of the fifty most similar reference images per query image. The training step or SIFT feature extraction phase consists of four main steps summarized below:

1. Scale-space extrema detection scans the image at different scales to find locations that are scale and rotation invariant and, thus, potential keypoints.
2. Keypoint localization selects a subset of those keypoints identified in step 1 based on their stability.
3. Orientation assignment assigns an orientation to each keypoint, which will allow future steps of the algorithm to ensure rotation invariance.
4. Keypoint descriptors are distinct features developed to represent the region surrounding each keypoint.

For additional details, refer to Lowe’s original publication, “Distinctive Image Features from Scale-Invariant Keypoints” [Lowe (2004)].

Once the keypoints and descriptors for each reference image have been extracted, we must rank the most similar reference images for each query image. First, we extract the keypoints and descriptors for a given query image. After, we match the descriptors from the query image with the collection of those from the 50 similar reference images (as provided by the VGG-16 model). We do this by independently matching each keypoint in the query image to the closest keypoints in the reference collection. We then apply Lowe’s ratio test to remove false matches and retain a final set of correct matches [Lowe (2004)]. In our case, we define the most similar reference image as that which has the largest number of correct matches with the query image. Using this approach, we re-ranked the 50 reference images provided by the VGG-16 model for each query image in order to obtain the results from the hybrid VGG-16 + SIFT model.

3.2 Baseline SIFT

The baseline SIFT approach follows the training and prediction steps detailed above with no additional changes to the SIFT keypoints or descriptors. However, we conducted three experiments in relation to baseline SIFT as shown in Table 2. These experiments aimed to answer the following questions:

1. Should scaling (true / false) be applied to the reference and query images before conducting the SIFT training and prediction steps?
2. How many reference images, *N* (50 / 15), should be compared with a single query image at a given time during the matching process?

At the core of both of these questions is the trade-off between optimizing the model’s performance and the model’s runtime.

Scaling and the value of *N* influence the number of SIFT keypoints and descriptors we extract and match throughout the training and prediction phases. In the case of scaling, applying scaling significantly reduces the number of SIFT keypoints and descriptors in both phases of the SIFT algorithm. (When scaling was applied, we extracted 663 keypoints and descriptors on average per reference image. When scaling was *not* applied, that number increased to 3,650 per reference image.) In the case of *N*, a larger *N* means we must compare a larger number of images and associated keypoints and descriptors when generating predictions for each query image. A small number of keypoints and descriptors in the matching process may result in poorer model performance, but improves running times. Likewise, too large a number of keypoints and descriptors can also decrease model performance [Lowe (2004)].

We noticed these trends in the results of our experiments (Table 2). In summary, experiment 1 prompted us to always apply scaling because the performance and runtime were heavily compromised using the original, unscaled images. Comparing experiments 2 and 3, we observed that too low a value of *N*, as shown in experiment 3, would compromise results but improve runtime. We prioritized model performance, thus, we selected to use the parameters defined in experiment 2 (scale = true, *N* = 50) for all SIFT experiments going forward.

Table 2: Experiments for baseline SIFT run on hardware with an Apple M1 chip and 8GB RAM

	Experiment 1
	Scale = FALSE
	N = 50
		**Experiment 2
		Scale = TRUE
		N = 50**
			Experiment 3
			Scale = TRUE
			N = 15

Mean Position	0.186	**0.064**	0.095
Top-1 (%)	88.56	**95.50**	96.02
Top-5 (%)	**100.00**	99.87	99.74
Top-10 (%)	100.00	**100.00**	99.87
SIFT Training (seconds per reference image)	0.088	**0.032**	0.032
SIFT Prediction (seconds per test image)	16.10	0.723	**0.235**

With additional time, we would conduct additional experiments to find the optimal value of *N*. Additionally, it would be beneficial to explore approaches by which we could cut down the number of SIFT keypoints and descriptors without losing crucial information. One potential idea would be to try and identify a smaller subset of SIFT keypoints and descriptors that are unique to each specific reference image. This would reduce the number of SIFT descriptors while retaining important information.

3.3 Augmented SIFT

We also looked into the process of augmenting SIFT descriptors by adding additional keypoint features to the descriptors. The keypoint features added were location, rotation, hue, and size. Location, rotation, and hue each had an (x,y) component. Rotation and hue were both converted to points around a circle, to capture the fact that 1 and 360 (rotation) or 255 (hue) are adjacent to each other and the euclidean distance of the vectors should reflect that. Thus, we added 7 total features to the descriptors, bringing their size from 128 to 135. We also looked into the keypoint features response and octave, however we found they did not improve the performance of the model so they were excluded.

The motivation for this approach stems from the fact that SIFT is scale and rotation invariant, and by default operates on grayscale images. This allows SIFT to match features from a different perspective and scale in different parts of the image, and is color independent, but sensitive to brightness. Because of the nature of our data (backgrounds removed, all images scaled to same size), we want SIFT to add additional information to the descriptors to emphasize matches that are in the same location and have similar keypoint features.

The hybrid VGG-16 + augmented SIFT model that results from adding the 7 keypoint features as described above had improved results over the hybrid model with baseline SIFT. The mean position of the augmented model is 0.061, with 95.76% of cards identified correctly in the first position, and 100% of cards identified within the top-5 predictions.

Most of the testing of SIFT augmentation was performed with a previous version of the code than the final model. Unfortunately, with updates to the matching process and changes to other aspects of the model that occurred, the final augmented SIFT model does not perform as well as initial testing indicated. After some bug testing, we were able to identify an issue in the scaling of images and were able to improve the results of augmented SIFT to the level we are currently reporting in the final model, which does offer modest improvements over the baseline SIFT model. The results of experiments on the augmentation process which are described in Appendix A should not be seen as indicative of the final model's performance, but rather as a proof-of-concept and hopefully may provide some intuition/motivation for further testing.

Results

We have discussed some experiments and approaches to solve this problem. The selection of the final model was determined based on a compromise between these two aspects:

- Performance
- Efficiency

The results shown in this section correspond to the performance of the models trained on the whole reference set and assessed on the complete test set.

4.1 Performance

The following table shows the performance of each of the models:

Table 3: Comparison of Models performance (*) Run for 8 epochs due to computational limitations

	VGG16 Baseline	**VGG16 Baseline + SIFT**	**VGG16 Baseline + Augmented SIFT**	**Retrained VGG16 (Transfer Learning) (*)**
Mean Position	0.33	0.064	**0.061**	31.34
Top-1 (%)	86.88	95.50	**95.76**	72.87
Top-5 (%)	99.23	99.87	**100.00**	86.76
Top-10 (%)	99.87	100.00	**100.00**	89.58

We can see how VGG-16 baseline definitely sets a very good starting point, being able to correctly match almost 87% of the query images. After applying the baseline SIFT algorithm on top of VGG-16, we can observe that the percentage of correct matches increases up to 95.5%. At this point we do not have a lot of margin for improvement but, surprisingly, if we apply augmented SIFT, the accuracy of the model increases up to 97.76% and is able to retrieve the correct image within the top 5 matches on 100% of the test images.

On the other hand, VGG-16 with Transfer Learning seems to be on track to provide good results. However, those values were obtained after only 8 epochs. We were not able to run the algorithm for a reasonable number of epochs (e.g. 50 or 100) due to computational limitations.

4.2 Efficiency

The following table shows the hardware specifications of the environments used to run the experiments:

Table 4: Hardware Specifications

	OS	**CPU**	**RAM**
Environment 1	macOS Monterey	Apple M1	8 GB
Environment 2	Linux	Intel Xeon Silver 4210	264 GB

The results shown in this section correspond to the performance of the models after refactoring the code. The following table shows the running times of each of the models for training and for making predictions.

Table 5: Running Times (*) Run for 8 epochs due to computational limitations

	Training (14396 images)		**Predictions** (779 images)	
VGG16 Baseline Env 1	1 h 10 min	290 ms/image	8 min	616 ms/image
Retrained VGG16 (*) (Transfer Learning) Env 2	5 h	1250 ms/img	27 min	2080 ms/image
VGG16 Baseline + SIFT Env 1	1 h 26 min	358 ms/img	9 min	693 ms/image
VGG16 Baseline + Augmented SIFT Env 1	1 h 27 min	358 ms/img	11 min	847 ms/image

The fastest algorithm is VGG-16 baseline, running on Environment 1. The training or feature generation for the reference set is completed in 1 hour 10 min, which translates into less than 300 ms per image. We also run in Environment 1 the Hybrid model, based on VGG-16 and the two SIFT variations: baseline SIFT and augmented SIFT. Both versions of the algorithm are computationally similar in the training part though they differ when it comes to predictions. Baseline SIFT is able to process each test image in less than 700 ms while Augmented SIFT takes about 850 ms per image.

On the other hand, the retrained version of the VGG-16 model was executed in environment 2 due to memory limitations in Environment 1. For only 8 epochs, this algorithm took 5 hours to be trained and needed more than one second to process each test set. This retrained version of VGG-16 is definitely interesting but apart from being computationally very expensive, it doesn't seem to improve by a lot the results provided by baseline VGG-16.

After evaluating the performance and efficiency of each individual model, we selected the hybrid version basen on VGG-16 baseline plus augmented SIFT as our winning algorithm. This model definitely adds a high value when it comes to making predictions (95.76% of accuracy) plus its processing time is quite acceptable (358 ms/img on training and 847 ms/image on predictions).

5. Conclusion and Future Work

In this report, we propose a hybrid VGG-16 + augmented SIFT model to solve the problem of Pokemon Card Identification. The hybrid model correctly matches the queried image with its reference image 95.76% of the time and always identifies the reference image among the top 5 matches. Additionally, the model is robust to input noise (glare, shadows, and resolution) and able to distinguish between similar cards.

With additional time, there are various optimizations that could be performed on the VGG-16 transfer learning and SIFT components. Specifically, we would continue our experiments to incorporate image transformations during the VGG-16 transfer learning stage and to tune the SIFT hyperparameters (e.g. *N* and feature weights). However, we believe the model's performance and runtime could also be improved with a new venue of exploration that focuses on identifying subsets of SIFT descriptors that are unique to each of the reference images.

Along with this report detailing our experiments, hybrid VGG-16 + augmented SIFT model, and future steps, the code deliverables and the final PowerPoint presentation delivered on December 15, 2022 are available in the associated GitHub repository.

Appendix A: Experiments on SIFT descriptor augmentation

Initial testing of the SIFT augmentation process led us to slightly different results than we achieved in our final model. In general, individual experimentation produced slightly higher performance results than are seen in the performance of the final model. These results are not indicative of the performance of the final model, but are provided as a proof-of-concept.

In individual experimentation, we achieved a mean position score of 0.0398 for the augmented SIFT baseline model (all features added have a weight of 1). The model correctly identified 96.9% of images in the top position, and 100% of images in the top-5. At the time, this was being compared to a baseline SIFT model with a mean position score of 0.074, which correctly identified 96% of images in the top position, and 99.7% of images in the top-5.

We tested the individual contributions to the performance gain of individual features. This was performed by only adding individual features at a time and testing their performance. The results are listed below:

	Location	**Rotation**	**Hue**	**Size**
Mean Position	0.068	0.067	0.042	0.064
Top-1%	95.63	95.63	96.79	95.5
Top-5%	100	100	100	100

At the time of testing, the addition of all of these 4 features showed improvements over the baseline SIFT model. As can be seen, hue appears to be the most significant feature. Our guess that location would be the most significant appears to be wrong, as the result of adding location sees the least gain in performance amongst these features.

We also experimented with replacing features in the descriptors with the keypoint features as opposed to augmenting them. This was performed by simply replacing the final 7 features in the descriptor vector. In comparison to the augmentation process, this sees a very modest decrease in performance, but slight increase in runtimes. The replacement process also has a mean position score of 0.398, with 96.8% of images identified correctly in the top position, and 100% of images identified correctly in the top-5. The augmentation process ran for about 47 minutes on an individual machine, but the replacement only ran for 42 minutes. Our final model incorporates the augmentation process, but there seems to be a reasonable tradeoff in performance vs. runtime between the two models.

Finally, we also tried weighting the augmented keypoint features through a random search with 40 iterations. The values that features could be weighted ranged from 0:5 (20 iterations) and 0:2 (20 iterations). We found that the baseline augmentation model (with all features weighted = 1) outperformed over 90% of combinations. Because of this, and with a lack of additional testing data to verify results, we decided to implement this baseline into the final model, however additional parameter tuning of the augmentation process may provide improved results. The baseline model results with the parameter combinations that showed an improvement in score are provided below:

Location	Rotation	Hue	Size	Mean Position
1	1	1	1	0.398
1	1.9	1.1	1.9	0.0373
1.5	0.5	1.1	1.9	0.0373
4	0.8	1.5	3	0.036