– COS429 Final Report, Spring 2023 –

Deep Gambit Edge Net: a Poker Chip Counting Model

Ketya Nop (cnop), Bomb Vongthongsri (kritinv), Harvey Wang (harveyw)

1. Motivation and Goal

Miscounting poker chips and bad dollars can have serious consequences in the world of gambling. In poker games, chips are the equivalent of currency, and any discrepancy in their count can lead to confusion, mistrust, and even accusations of cheating. Similarly, bad dollars or counterfeit money can create chaos and disrupt the integrity of the game.

Miscounting poker chips can occur in several ways. For example, a player may lose track of the number of chips they have, or they may mistakenly believe that they are owed more chips than they actually have. This can be particularly problematic in games with high stakes, where even small discrepancies can result in significant losses. In addition, miscounting can be intentional, as some players may try to cheat by adding chips to their stack or taking chips away from others.

To prevent miscounting, it is essential for players and dealers to keep careful track of the number of chips in play. This often involves using chip racks or trays, which can be stacked and counted easily. In addition, it is crucial to establish clear rules for how chips are exchanged and handled during the game. This can include protocols for buying in, cashing out, and distributing pots.

The primary goal of this project is to develop a poker chip counting model that can accurately and quickly count poker chips in real-time. The model will utilize computer vision and machine learning techniques to analyze images of poker chips and determine their quantity.

The model's success will be evaluated based on its accuracy, speed, and ability to handle different chip configurations and lighting conditions. If successful, this model can be a valuable tool for both professional and casual poker players, dealers, and casino staff, improving the overall gameplay experience and reducing errors.

2. Background and Related Work

There is a remarkably limited amount of existing work done in the application of computer vision to poker chip counting. Most of the existing literature focuses on developing automated poker-playing systems using camera input to detect card and chip values. In 2011, Martins et al. proposed a novel chip counting system in “Poker Vision: Playing Cards and Chips Identification based on Image Processing.”

The system used color segmentation with Hough circle transform to isolate the poker chip stacks from a table and compute the number of chips in the stacks. However, the resulting accuracy was extremely low.

In 2020, Syed et al. presented "DeepGamble: Towards unlocking real-time player intelligence using multi-layer instance segmentation and attribute detection," which proposed another system for automated playing card and poker chip identification. DeepGamble achieved high accuracy rates for chip counting at over 95% by using instance segmentation to separate the chip stacks and XGBoost to compute the number of chips. However, DeepGamble relied on a fixed dual camera setup and a designated spot on the poker table for where chips were placed. Our goal is to provide a more flexible and accessible system that recreational poker players can use with their own mobile devices at any point.

Since there is a standard convention for poker chip dimensions, we also considered the idea of estimating the poker chip stacks as cylinders in three-dimensional space and computing the height of the cylinder. Some existing literature tackles the problem of robust cylinder detection, such as "A Robust and Efficient Framework for Fast Cylinder Detection" by Figueiredo et al. in 2019, which used a generalized Hough transform to detect cylinders using point clouds. However, cylinder detection almost always requires depth information; Figueiredo et al. used an RGB-D camera while other similar projects employ stereo camera setups. We instead aimed to produce chip counts efficiently using only a typical RGB photo captured on mobile devices. Another important limitation of cylinder detection and similar techniques is the irregularity of poker chip stack shapes; singular chips or short chip stacks would also likely fail to be recognized as cylinders.

Therefore, there does not exist a poker chip counting model accessible to casual or recreational players looking to automate chip counting in their poker games. This project explores several modern deep and non-deep learning algorithms to tackle this task.

3. Methods

3.1 Deep Learning Approaches

Due to the limited related work, we opted to survey a variety of models and algorithms to identify the best performer. We considered three different types of computer vision techniques for this problem: object detection, instance segmentation, and classification. Apart from classification, each of these techniques were performed at the stack level and chip level.

There are several popular algorithms used for object detection in computer vision, including: Faster R-CNN[1], YOLO (You Only Look Once)[2], SSD (Single Shot Detector)[3], Mask R-CNN[4], and RetinaNet[5].

---

[1] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. ArXiv.org. https://arxiv.org/abs/1506.01497

[2] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. ArXiv.org. https://arxiv.org/abs/1506.02640

[3] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. Computer Vision – ECCV 2016, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

[4] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. ArXiv.org. https://arxiv.org/abs/1703.06870

[5] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. ArXiv.org. https://arxiv.org/abs/1708.02002

To optimize user experience, speed is an important factor alongside accuracy for our system. YOLO is a real-time object detection algorithm that can detect objects in images or videos with high efficiency (it processes images in a single forward pass through a neural network, which makes it faster than many other object detection algorithms), and is also known to be extremely accurate. Since YOLO also supports instance segmentation, we have decided to implement each of the subsequent models with YOLOv8, the latest version of YOLO developed by *Ultralytics*, using pretrained weights as a starting point for transfer learning to prevent overfitting due to our small dataset.

3.1.1 DGEN-ObjDet-Stack

Since an image may contain multiple chip stacks, object detection may quickly identify each stack and classify them based on stack size. The images were labeled with bounding boxes around each stack and a class name corresponding to the number of chips in the stack. This would serve as our baseline because it is straightforward and relatively simple to implement.

3.1.2 DGEN-ObjDet-Chip

Similarly, each stack contains multiple poker chips; we can also perform object detection on the individual poker chips and count the total number of chips to determine the stack size. The images were labeled with a smart polygon around each individual poker chip in every stack. This model provides a more intuitive approach to the counting problem: a typical procedure for determining stack size is to count each individual chip.

3.1.3 DGEN-Segment-Stack

Whereas object detection draws bounding boxes as a coarse estimate of where an object is predicted to be, instance segmentation creates masks at the pixel level outlining different types of objects in an image. This may allow the model to better predict the number of chips in a stack since we are providing more fine-grained descriptions of the stacks and their shapes. The images were labeled such that each stack was outlined and classified based on the number of chips in the stack.

3.1.4 DGEN-Segment-Chip

Instance segmentation may also benefit for identifying individual chips, which often appear in irregular forms within the stack depending on position and occlusion. Each stack is constructed from several instances of these chips, so the model may learn to identify when a tiny sliver at the bottom of a stack may also represent another chip. The images were labeled such that each poker chip was outlined and classified as a chip.

3.1.5 DGEN-Crop-Classification

Rather than attempting to count the total number of chips in an image of multiple stacks at once, we can also isolate each individual stack first, then count the number of chips in every stack. To do so, we can use a two-step model which first uses object detection to identify a poker chip stack, then crops the stack and performs a classification task to predict the number of chips in the cropped image. This requires an additional step but also is an intuitive way to simplify the larger problem into several similar subproblems. Note that we opted to instead use GoogleNet for this

classification task due to its high accuracy, especially with smaller objects like a stack of poker chips. This model architecture is shown in **Figure 1** below.

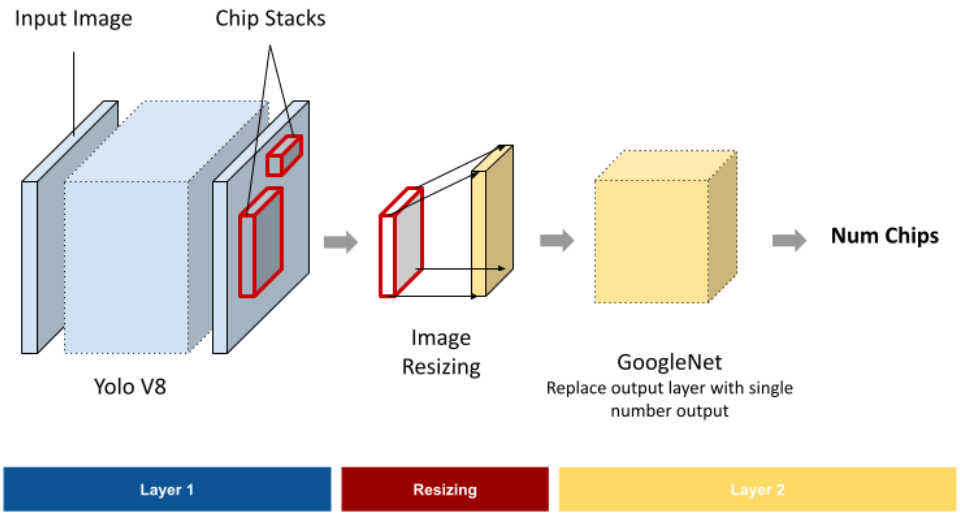

Figure 1. *DGEN-Crop-Classification Architecture.*

3.2 Non-Deep Learning Approaches

Although the vast majority of modern computer vision models apply deep learning, simpler non-deep learning approaches can sometimes be equally effective, especially when working with a smaller dataset. One technique we also considered was applying a horizontal Sobel edge filter and thresholding to highlight the edges of the chips in the stack, then counting the number of chips using a simple regression model. However, we soon discovered that due to inconsistent lighting, image quality, and picture angle, the edges of the chips were extremely difficult to define. Moreover, patterns on the poker chips created further edges which also made it difficult to distinguish the chips from each other after applying the filters. As seen in **Figure 2**, the chips are clearly distinguishable in some images (left) and difficult to count in others (right). Therefore, we decided to forego this approach because of the variability in the dataset and the inconsistent outputs from the edge detection algorithm.

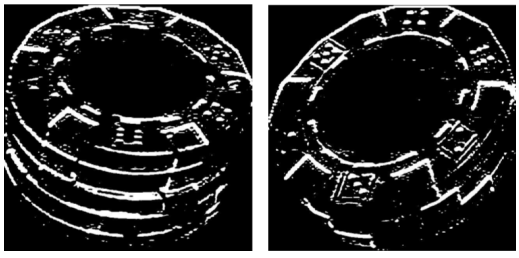

Figure 2. *Horizontal edge detection outputs on a 5-chip stack (left) and 3-chip stack (right)*

3.3 Implementation

Our models were trained on Google Colab using Tesla T4 GPUs. We used the Ultralytics (v=8.0.28) Python package[6] to import the YOLOv8 models. For object detection, we started with the pretrained weights of the YOLOv8s model. For instance segmentation, we used the pretrained weights of the YOLOv8s-seg model. Finally, for classification we used the GoogleNet model provided by the PyTorch[7] package. These models were imported into our Google Colab notebook and then retrained using a custom dataset. We then performed our own evaluations on a holdout test set and computed the corresponding accuracy and error rates.

4. Data

4.1 Data Acquisition

As there is no publicly available dataset of poker chip images, we created our own dataset by taking pictures of poker chip stacks using our smartphones (iPhone SE, iPhone 8, Samsung S21). Our dataset consists of images of chip stacks of different colors, representing varying value denominations, taken from various distances, angles, and with different backgrounds (such as carpet and table) and lighting conditions. Additionally, we varied the number of stacks and the number of chips in each stack in each photo. Our original plan was to generate a dataset of 600 images, consisting of 300 random online images and 300 images from our smartphones, according to the distributions presented below.

Chip Stack Size					
Stack Size	1-5	5-10	10-15	15-20	20+
Distribution	15%	20%	30%	20%	15%

Chip Color					
Stack Size	Red	Green	Blue	Black	White
Distribution	20%	20%	20%	20%	20%

Number of Chip Stacks					
No. Stack	1-2	2-4	4-6	6-8	8+
Distribution	25%	25%	20%	10%	10%

[6] https://github.com/ultralytics/ultralytics
[7] https://pytorch.org/hub/pytorch_vision_googlenet/

Photo Angle					
Angle	Eye Level	Right Pan	Left Pan	Bottom Pan	Top Pan
Distribution	20%	20%	20%	20%	20%

Background			
Background	Carpet	Table	Solid Color
Distribution	40%	40%	20%

Table 1. *Image category distributions*

However, we encountered several issues with the online images we acquired, including unrealistic lighting conditions and blurring of the background chip stacks, which made them unsuitable for training. As a result, we made the decision to solely utilize images (476) captured from our smartphones for the construction of our dataset. Additionally, we found that the majority of our images consisted of chip stack sizes in multiples of 5 after training. Therefore, while our models were able to accurately detect stack sizes in these instances, they proved to be less effective when detecting stacks that were not in multiples of 5. To address this issue, we took additional images of poker chips to increase the dataset, ensuring that we had a minimum of 300 samples for each stack size ranging from 1 to 10, increasing our total image count to 718. Due to time constraints, we were unable to do the same for stack sizes greater than 10.

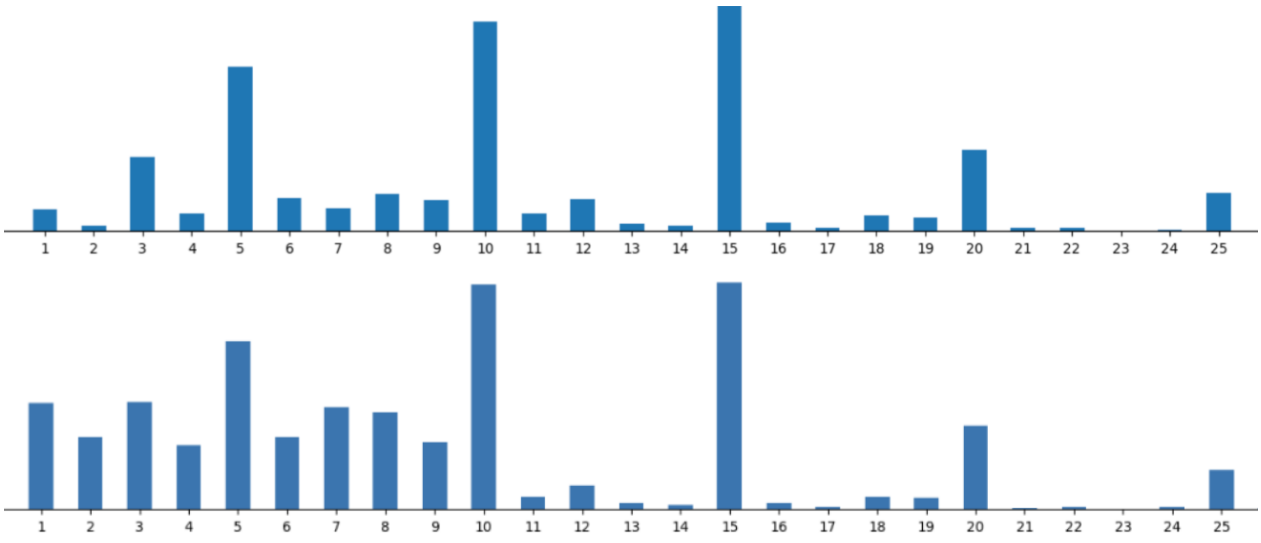

Figure 3. *Old vs new image distributions.*

Figure 4. *Sample unprocessed images from dataset and initial biased stack size distribution.*

4.2 Data Labels

To label our dataset, we uploaded the images to Roboflow, an image labeling tool, and labeled them accordingly. We used different types of labels such as bounding boxes and segmentation masks for the different models we trained.

Model	**Label**
DGEN-ObjDet-Stack	Bounding Box (Rectangle)
DGEN-ObjDet-Chip	Bounding Box (Smart Polygon)
DGEN-Segment-Stack	Instance Segmentation Mask
DGEN-Segment-Chip	Instance Segmentation Mask
DGEN-Crop-Classification	Bounding Box (Rectangle), Single-Class Label

Table 2. *Label types for each model.*

To label our dataset, we used different types of bounding boxes, including rectangles and smart polygons. For rectangular bounding boxes, we manually drew them around each chip or chip stack in the image and assigned the appropriate class label (chip type or count) to each box. To convert rectangular bounding boxes to smart polygons, we used Roboflow's label assisting tool. For instance segmentation tasks, we utilized Roboflow's label assisting tools to select specific points around the borders of the target object until the desired section was properly segmented. Finally, for class labeling, we simply assigned each image a specific class.

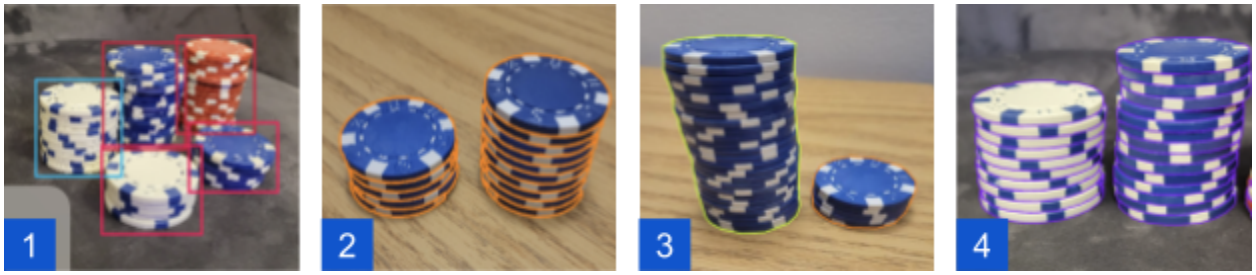

Figure 5. *(1) DGEN-ObjDet-Stack - bounding boxes for chip stacks (2) DGEN-ObjDet-Chip - bounding boxes (smart polygon) for individual chips (3) DGEN-Segment-Chip - instance segmentation for individual chips (4) DGEN-Segment-Stack -instance segmentation for chip stacks.*

4.3 Data Preprocessing and Augmentation

To increase the size of our dataset and introduce variation to the existing images, we implemented a series of preprocessing and data augmentation techniques. Our images were resized to 640x640 dimensions, auto-oriented, and horizontally flipped. We also randomly adjusted the brightness and rotation between -15% to 15% and introduced a random noise of 2%. For the second layer of DGEN-Crop-Classification, we cropped the images according to the bounding box output of the first layer, resized the data to 640x640, and fed it to the second classification layer. Our dataset size was increased by three times to a total of 2514 images after preprocessing and augmentation. As there were multiple stacks present in each image, our cropped chip image dataset comprised 8256 images.

Figure 6. *Sample cropped images from DGEN-Crop-Classification.*

4.4 Data Collection Limitations

Although we were able to draw rectangle and smart polygon bounding boxes around chip stacks for all 2514 images, it was not feasible to do the same for individual chips due to the large number of chips in each image. Labeling instance segments of all the chips or drawing smart polygons for each chip in every image took approximately 5-15 minutes per image, depending on the number of chips present. This

required drawing detailed borders around each chip, taking special care to navigate around occlusion and neighboring chips. As a result, we could only label 70 images for individual chip detection tasks in the timeframe of this project.

5. Evaluation

5.1 Qualitative Evaluation Metrics

We visually inspected the model's output to assess its performance. This involved comparing the bounding boxes and segmentation masks generated by the model and comparing them with the ground truth labels. We qualitatively assessed the model's ability to accurately identify the objects in the images by assessing the models' abilities to separate overlapping objects, handle occlusions and partial objects, and accurately delineate object boundaries. By performing a qualitative evaluation, we gained a deeper understanding of the strengths and limitations of the models' capabilities that may not be apparent within the quantitative results, which can inform future improvements to the model's design and training process.

We also visually inspected the model output of our classification models, specifically focusing on images with incorrect labels. Through this inspection, we identified common denominators in images that produce incorrect output.

5.2 Quantitative Evaluation Metrics

To quantitatively evaluate the performance of our poker chip counting model, we will measure its accuracy, adjusted accuracy, precision, mean absolute error (MAE), standard deviation, and inference time. These metrics provide a comprehensive evaluation of the model's accuracy, robustness, and ability to handle real-world scenarios.

Accuracy measures the proportion of correct predictions out of the total predictions. It indicates how well the model is able to correctly identify the number of chips in a given stack. The formula for accuracy is:

$$Accuracy = CP/(TP)$$

CP is the number of total correct predictions and TP is the number of total predictions.

The adjusted accuracy metric measures the overall accuracy of the model in identifying the number of chips over a set of images by taking into account the average chip count error (MAE) and the total number of chips in each image. This provides a way to compare the performance of chip stack detection algorithms and individual chip detection methods and provides a universal metric for evaluating how close the model is to predicting the true chip count.

$$Adjusted\ Accuracy = MAE * N/(TC)$$

MAE is the mean absolute error, N is the total number of images, and TC is the total number of chips in all images.

MAE (Mean Absolute Error) measures the average absolute difference between the predicted number of chips and the ground truth number of chips in an image. It indicates the average magnitude of error the model makes when predicting the number of chips.

$$MAE = 1/N * \sum|Pred - Label|$$

N is the total number of images in the test set, Pred is the model's prediction of the chip count in an image, and Label is the ground truth for the number of chips in the image.

Standard deviation is a statistical measure that calculates the amount of variation or dispersion in a set of data. It is used to measure the consistency of a model's predictions in relation to its actual performance. The formula for standard deviation is:

$$Standard\ Deviation = \sqrt{(1/N * \sum(Pred - Label - Mean)^2)}$$

N is the total number of images in the test set, Pred is the model's prediction of the image, Label is the ground truth label of the number of chips in the image, and Mean is the average difference between the predicted number of chips and the ground truth number of chips.

Inference time measures the amount of time it takes for the model to make a prediction. It indicates how fast the model is able to count the number of chips in an image.

5.3 Comparisons

In order to identify the best models for the chip counting task, we compared the different models and algorithms described in the Methods section. We evaluated these models using the metrics outlined in the Evaluation section, providing us with a comprehensive understanding of each model's strengths and weaknesses. This allowed us to determine which model was best suited for accurately counting the number of chips, while also providing insight into the trade-off between accuracy and speed. By performing this comparison, we were able to identify the model that provided the best balance of accuracy and efficiency for the chip counting task.

6. Results and Analysis

6.1 Qualitative results

We visually inspected the deep learning models' outputs to assess their performance in the counting tasks. For the object detection models, we visualized the bounding box predictions for the testing sets as seen in **Figure 7.** We examined whether the boxes accurately encompassed the chips at every part of the stack (top, middle, bottom), and analyzed which scenarios led to the poorest performance.

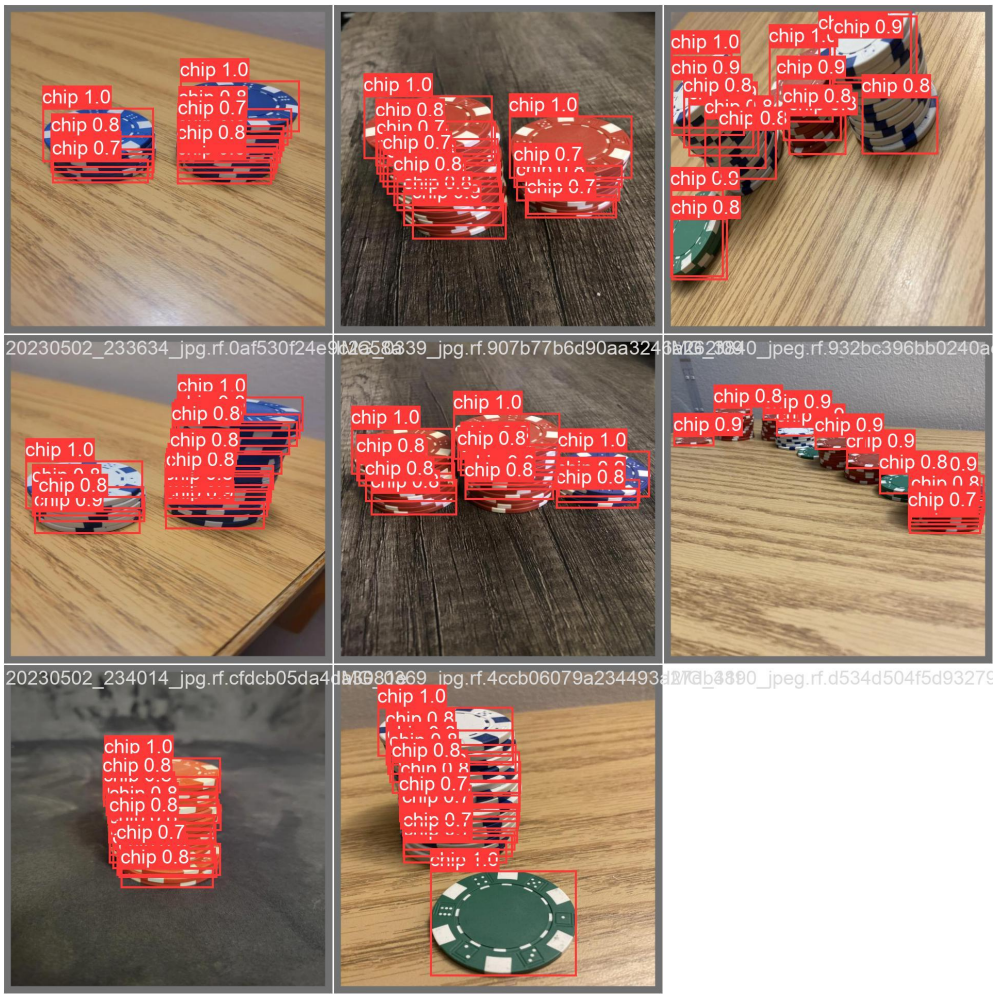

Figure 7. *Example bounding box predictions with confidence levels for DGEN-ObjDet-Chip.*

These visualizations allowed us to examine the cases where the models were performing the best and worst and determine why such a contrast was occurring. For instance, we inspected the predictions for DGEN-ObjDet-Chip and found that in cases of rotation of over 15 degrees and when the picture is taken from a farther distance, the model tended to struggle to identify the chips. In contrast, close-up and well-aligned images generally resulted in high accuracy rates, as seen by accurate bounding box placement around the chips. The top chip was also almost always correctly predicted with an extremely high confidence level, likely due to its consistent visibility and similar shape. These findings are exemplified in **Figure 8.**

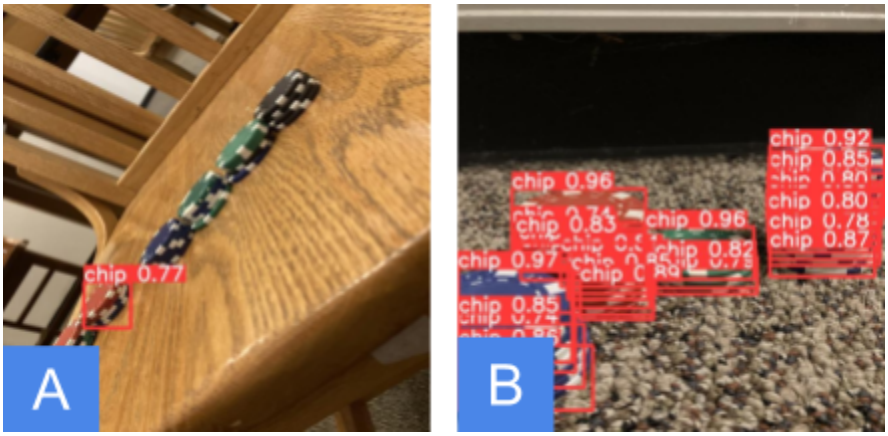

Figure 8. *Example images where DGEN-ObjDet-Chip performs poorly (left) and well (right).*

We also examined the generated segmentation masks (**Figure 9)** for the instance segmentation models and evaluated their ability to accurately segment objects in the image. During this evaluation, we assessed the model's ability to separate overlapping objects, handle occlusions and partial objects, and accurately delineate object boundaries.

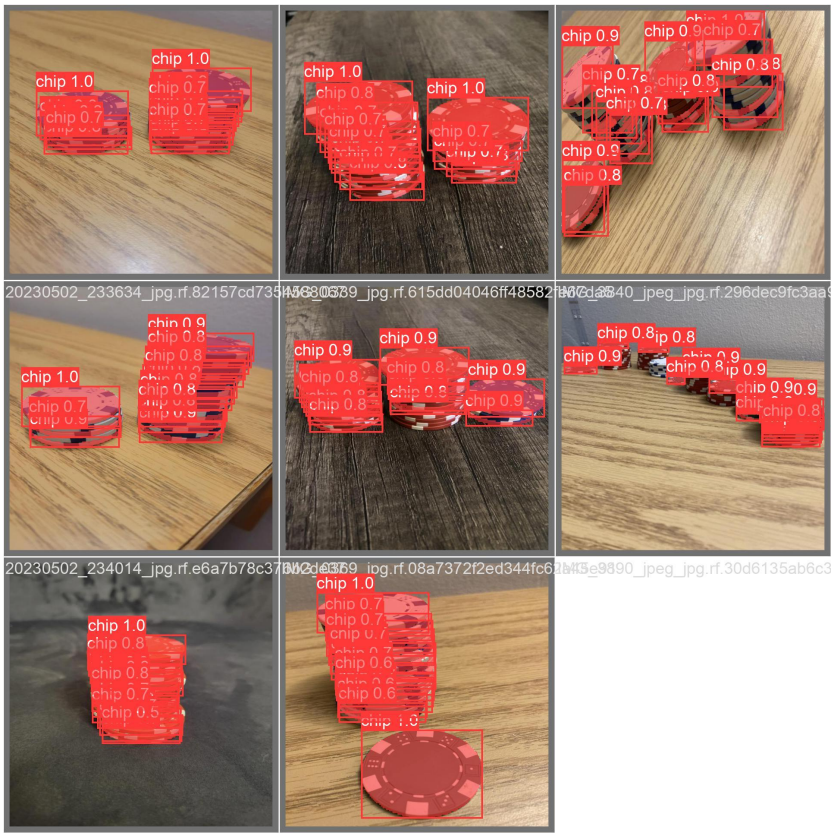

Figure 9. *Example segmentation mask predictions (highlighted in red) for DGEN-Segment-Chip.*

We noticed the segmentation mask predictions for DGEN-Segment-Stack performed poorly when the chips were located far away and did not have one consistent stack color. In these cases, the model often failed to recognize entire stacks of chips or labeled the stacks with the incorrect chip counts. In contrast, close-up and separated stacks were usually highly accurate with high confidence intervals. These findings are reflected in **Figure 10.**

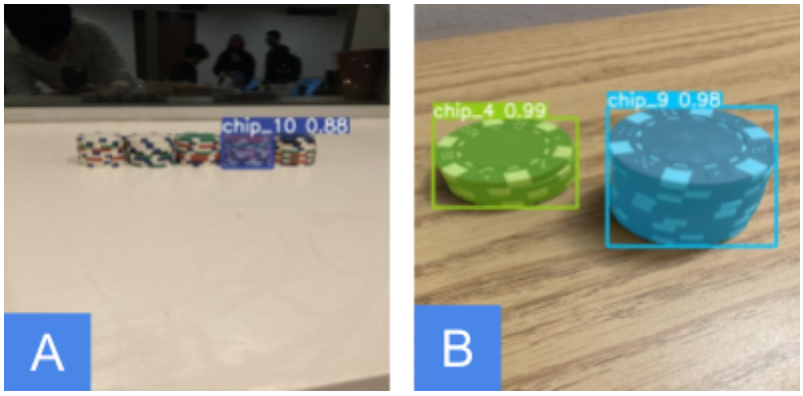

Figure 10. *Example images where DGEN-Segment-Stack performs poorly (left) and well (right).*

Figure 11 demonstrates the kinds of images that performed the best (left) and the worst (right) for the DGEN-Segment-Chip model. We found that the model performed well on pictures taken from a side profile, but its performance deteriorated as the image deviated from this orientation, such as in cases of rotation and occlusion. Another interesting trend we observed for the incorrect predictions was that sometimes neighboring chips would fail to be labeled, as seen in the left photo of **Figure 11**. This may be due to the model attempting to avoid overlap between chip instances or difficulty distinguishing edges between certain chips. Similar to DGEN-ObjDet-Chip, close-up and non-rotated images typically resulted in highly accurate masks of chips, even when only parts of the bottom chips were visible.

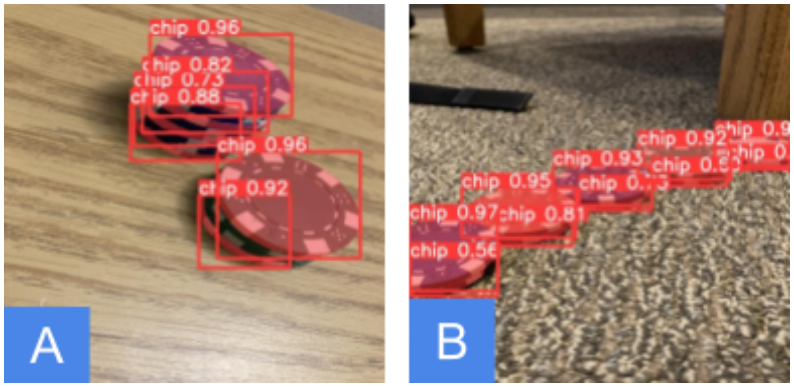

Figure 11. *Example images where DGEN-Segment-Chip performs poorly (left) and well (right).*

Next, we visually inspected the output from our DGEN-Crop-Classification model and observed that it tends to produce incorrect labels on images of chip stacks with mixed colored chips. An example of this is shown in **Figure 12**. We suspect that this model inaccuracy may be due to most of the training images being composed of single colored chip stacks. Additionally, it is unclear why the model tends to under predict the value of the chip count by 1. One possible explanation could be that the different colored chips at the bottom cause the model to confuse the chips with the background as the color of the chips changes.

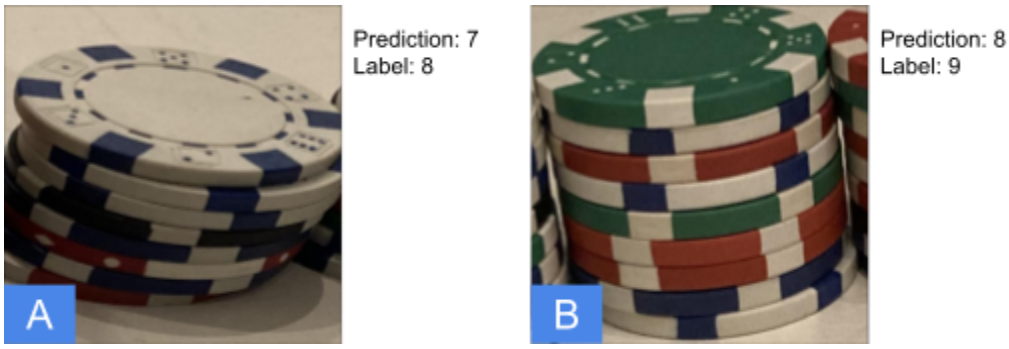

Figure 12. *Sample chip stacks with mixed colored chips.*

Another noteworthy observation is that the model tends to over-predict the chip count when the cropped image is occluded. This may be due to the chip stack that obstructs the main stack in focus, causing confusion in the model about which chips to count. Similarly, the presence of other chips in the image could also lead to confusion, causing the model to count foreground chips in addition to the chips in focus. **Figure 13** illustrates two examples of occluded chip stacks, one with a blue chip stack being occluded by a stack of red chips (left), and another with a mixed-colored chip stack being occluded by a stack of blue chips (right).

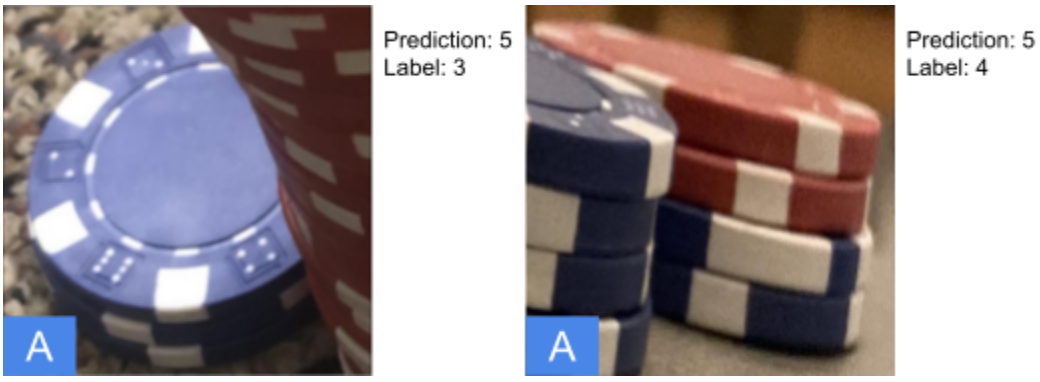

Figure 13. *Sample chip stack with occluded chips.*

6.2 Quantitative results

We evaluated the models for the poker chip counting task and compared their performance using the following metrics: accuracy, adjusted accuracy, mean absolute error (MAE), standard deviation, and inference time.

Model	Accuracy	Accuracy (Adjusted)	MAE	Standard Deviation	Inference Time
DGEN-ObjDet-Stack	60.34%	87.95%	3.621	8.524	12.5ms
DGEN-ObjDet-Chip		88.02%	4.033	5.687	7.4ms
DGEN-Segment-Stack	45.04%	86.02%	4.715	7.675	11.2ms
DGEN-Segment-Chip		87.04%	4.371	6.053	15.3ms
DGEN-Crop-Classification	64.81%	88.34%	0.986	1.937	21.1ms

***Table 3**. Performances of various models.*

The DGEN-Crop-Classification model achieved a remarkable 99.7% accuracy in annotating the poker chip stacks. However, for the purpose of our evaluation, we focused on the model's ability to count chips, which resulted in an accuracy of 64.81%. This accuracy reflected whether or not the number of poker chips our model predicted matched the number of actual chips in each stack. While perfect accuracy would be ideal, the model's MAE of 0.986 chips suggests that it is still highly capable of counting the number of chips. The majority of its incorrect predictions were only off by one or two chips; though imperfect, it shows there is great potential for this approach to produce highly accurate chip counts. Below is a breakdown of the accuracy by stack size for the DGEN-Crop-Classification model:

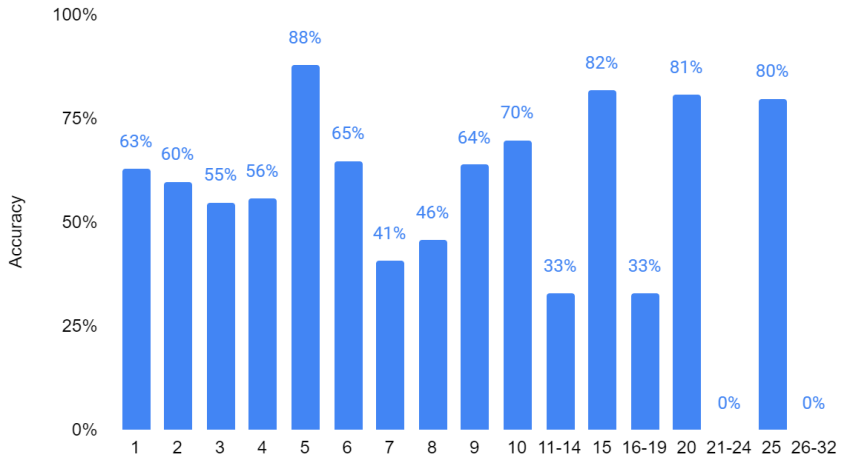

***Figure 14.** DGEN-Crop-Classification accuracy by stack size (class label)*

It's worth noting that the DGEN-Crop-Classification model had the longest inference time compared to the other evaluated models, though not be a noticeable margin.

All of the models were trained using a batch size of 16 for 100 epochs. During the training process of the DGEN-Crop-Classification model, the training loss continued to decrease until the 80th epoch, after which it began to converge. A visual representation of the DGEN-Crop-Classification model training loss can be seen in **Figure 15.**

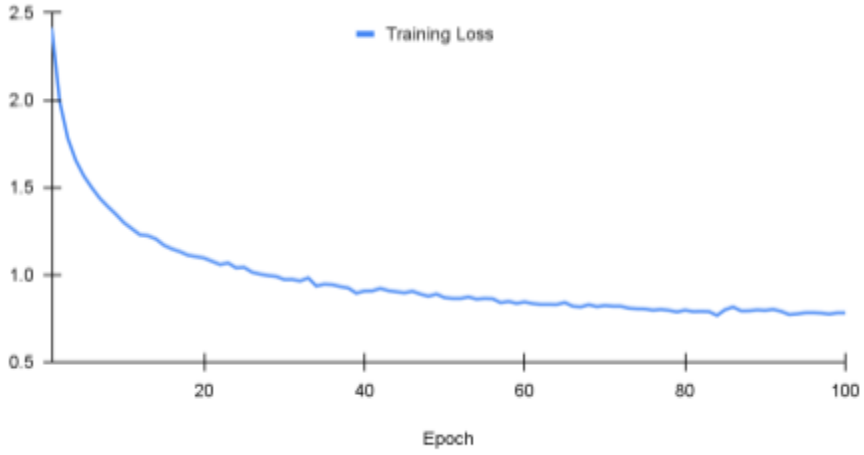

Figure 15. *DGEN-Crop-Classification training loss*

The DGEN-ObjDet-Chip and DGEN-Segment-Chip models achieved comparable performances, with adjusted accuracies of 88% and 87% respectively, and MAEs of 4.033 and 4.371. However, the average error standard deviation is significantly higher than desirable. These models show a greater tendency to miss entire stacks or produce very obvious errors in counting the number of chips. It is also worth noting that although the performances of DGEN-ObjDet-Chip and DGEN-Segment-Chip models are significantly less accurate than DGEN-Crop-Classification, these models were trained on a limited set of only 70 images due to labeling feasibility constraints.

Based on the MAE metric, DGEN-ObjDet-Stack, which was our baseline model, outperformed all the models except for DGEN-Crop-Classification. It achieved an MAE of 3.621 and maintained a decent accuracy of 60.34% with an adjusted accuracy of 87.95%. However, its standard deviation was exceptionally high. On the other hand, DGEN-Segment-Stack was our worst performing model with an MAE of 4.715 and a high standard deviation of 7.675. We hypothesize that this may be due to the variability in perspectives and distances from which the photos were taken, which reduces the usefulness of the additional stack shape information provided by segmentation masks. In contrast, the baseline model uses bounding boxes, which does not include such fine representations of the stack shapes, leaving more opportunities for features like the chip edges to be identified.

7. Conclusions and Future Work

7.1 Conclusion

To address the common problem of miscounting of poker chips during gameplay, we conducted an investigation into multiple deep learning solutions for automating chip detection and counting. Specifically, we explored five different Convolutional Neural Network (CNN) algorithms, namely: DGEN-ObjDet-Stack, DGEN-ObjDet-Chip, DGEN-Segment-Stack, DGEN-Segment-Chip, and DGEN-Crop-Classification. DGEN-ObjDet-Stack and DGEN-Segment-Stack utilize object detection and instance segmentation to identify stacks of poker chips and assign them to their corresponding size class. DGEN-ObjDet-Chip and DGEN-Segment-Chip use the same techniques to detect individual chips instead of stacks, and the number of detected objects corresponds to the total number of chips. Lastly, DGEN-Crop-Classification takes a different approach by detecting poker chip stacks, cropping the images, and then using a single-class classification algorithm to determine stack size.

DGEN-Crop-Classification was the best-performing algorithm, followed by DGEN-ObjDet-Stack, DGEN-ObjDet-Chip, DGEN-Segment-Chip, and DGEN-Segment-Stack, achieving MAEs of 0.986, 3.621, 4.033, 4.371, 4.715, respectively. Although none of these algorithms have currently achieved operational accuracy, we believe that it is very achievable with a larger dataset and more time.

7.2 Future Works

Given the small dataset size, achieving operational accuracy through classification alone was challenging. Therefore, we trained additional models using various methods, including detecting individual chips. Our individual chip detection models performed well on eye-level side profiles but struggled with images taken from wide angles or those that were rotated. However, due to time constraints, we were only able to train these models on a limited number of images. We believe that labeling all our images would likely result in higher accuracy, and we could utilize these individual chip detection models in the second layer of the DGEN-Crop-Classification model (which would become DGEN-Crop-Chip) since images would be cropped to show close-up chip stacks.

To further improve our results, we plan to incorporate a more deterministic approach. The horizontal edge detection method we used lacked the ability to distinguish patterns from edges due to rotations and varying lighting conditions. One possible solution is to train CNNs to detect two key points in a poker stack: the middle of the edge of the top chip and the middle of the edge of the bottom chip. This would enable us to calculate the gradient in the direction of the line connecting the two key points and threshold the edges, instead of just calculating the horizontal edge gradients. We would also include a lower threshold and an upper threshold to distinguish between the patterns in a poker chip and the edges of the chips. We could input these images into a neural network to achieve potentially higher accuracy rates.

In conclusion, while chip detection is an easy task, chip counting proved to be much more challenging than originally thought due to the small dataset size and variation in camera perspectives. Our individual chip detection models show promise, but more labeled images would be necessary to create a sufficiently large training set and achieve more capable models. Finally, incorporating more deterministic methods

would be helpful given the difficulty of achieving highly accurate weights for the neural network in this task.

References

Figueiredo, R., Dehban, A., Moreno, P., Bernardino, A., Santos-Victor, J., & Araújo, H. (2019). A robust and efficient framework for fast cylinder detection. *Robotics and Autonomous Systems*, *117*, 17–28. https://doi.org/10.1016/j.robot.2019.04.002

Martins, P., Reis, L. P., & Teófilo, L. (2011). Poker Vision: Playing Cards and Chips Identification Based on Image Processing. *Pattern Recognition and Image Analysis*, 436–443. https://doi.org/10.1007/978-3-642-21257-4_54

Redmon J, Divvala S, Girshick R, Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. arXiv.org. Published 2015. Accessed May 10, 2023. https://arxiv.org/abs/1506.02640

Syed, D., Gandhi, N., Arora, A., & Kadam, N. (2020). DeepGamble: Towards unlocking real-time player intelligence using multi-layer instance segmentation and attribute detection. *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. https://doi.org/10.1109/icmla51294.2020.00067

Szegedy C, Liu W, Jia Y, et al. Going Deeper with Convolutions. arXiv.org. Published 2014. Accessed May 10, 2023. https://arxiv.org/abs/1409.4842