A new way to explore history and culture: Developing an image recognition web application for the Yup’ik

Seth E. Spickard ’26[1] and Sean P. Gleason[2]

[1]*Department of Mathematics and Computer Science,* [2]*Rhetoric Program, Hampden-Sydney College, Hampden-Sydney, VA 23943*

Abstract

By packaging the YOLO computer vision model in a web-based application, we can develop software for museums that recognizes their unique collection of artifacts. This application creates a nonlinear hands-on experience while ensuring the artifacts stay preserved by keeping them out of the hands of museum guests. With the new Starlink internet, we can help to connect the Yup’ik youth with their culture through an additional phone-based method.

Keywords— Neural networks, Computer vision, YOLO, Nanalleq, Yup’ik

Background Information

One does not simply jump into computer science research, especially when delving into neural networks and artificial intelligence. From the preliminary work and educational steppingstones I completed as an unspoken research prerequisite to the weekly and final goals we as a research team had to make, I will explore the steps I have taken to get to this point and courses of action Dr. Gleason and I will make this summer to complete our computer vision web application.

Preliminary knowledge must be built up to allow any aspiring computer science researcher to develop software and train networks. I have gathered this information foundation from my time spent in RHET 285 and with Dr. Gleason during individual sessions.

In my pursuit of computer science at Hampden-Sydney, I was introduced to Dr. Gleason who, while not a computer scientist by trade, has a prolific knowledge of computer vision through his work with drone and GIS research with the Yup’ik tribe in Alaska. Dr. Gleason led a class during the 2022-2023 spring semester for those interested in joining his research team in Alaska. The class touched on everyone’s individual interest (photography, computer vision, GIS mapping) with a broad focus on the coproduction of knowledge—how we should interact and research with the native population as to respect and utilize their cosmology and science rather than encroaching or promoting useless validation. The class served to develop some of the foundation for research but put it through our specific lens of communicating and working with the Yup’ik tribe.

In tandem with the class, Gleason coached me on learning Python as part of the preparation for summer research. I needed to walk before I could run, so before diving into computer vision, I needed to gather a grasp of Python before the summer. Our individual sessions focused on reviewing my work over the past week through my explanation, his critique, and our collaborative goal setting for the following week. Not only did these sessions provide me with the skills necessary to explore computer vision this summer but they allowed me to see how Dr. Gleason and I best work together and what it is like to work collaboratively on programming projects more broadly.

Materials and Methods

Overall Project Description.

Our project is the development of a web application that users can use to identify artifacts in the Yup’ik museum collection. The web app utilizes the You Only Look Once (YOLO) computer vision model to identify submitted images from the user’s device [1]. The YOLO model allows the neural network to be small enough to work on LAN servers if needed and is much faster than other comparable computer vision neural networks (such as the R-CNN family of models or Deformable Part Models) [2] [3]. Once the desired artifact is recognized, the user would be given information about the given artifact, including a video from an elder explaining its use, 3-D models of the artifact, and some text detailing any additional information.

Development Cycle.

To create the training set, the Nunalleq Culture and Archaeology Center—the museum in Yup’ik village of Quinhagak, Alaska—photographed multiple angles of each item in the collection on a white background with custom white-balance settings [4]. An example of one such picture is shown in Figure 1.

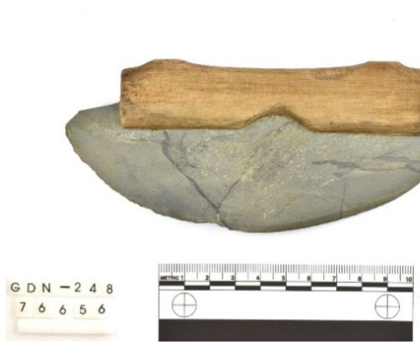

Fig. 1. Picture of ulu from the Cultural Center collection.

Next, we removed the image backgrounds through one of two methods. The first saw the images imported into Photoshop. Then, the select and mask tool was used to generate a new document containing the artifact. The second involved writing a small Python program to remove a certain pixel color range which matches the background color (in this case, white) in each image [5] [6]. The program also generated a new document containing the artifact. The new documents from both method were exported as a clipped .jpg file with the snakecase syntax number_item.jpg and a corresponding mask of the same name. The Python program was much less tedious but was not as precise if the colors of the background and the item were too similar. See Figures 2 and 3 for a cropped item and its mask. Noise images—some photographed by us and some gotten from the HanCo project—and their masks were created using the same tools [7] [8]. The generation and exportation of the artifacts and the noise images took four weeks.

With the images processed, we moved onto generating a dataset [9]. Instead of testing the process manually, we assembled a Python script to generate images for our training, testing, and validating datasets. This script pulls a random background, a small random assortment of noise images, and a small random assortment of artifacts out of the appropriate folder in the file directory seen in our GitHub's read me text file [10]. Each object—be it noise or artifact—is placed in the image, scaled, and slightly colored at random. The coordinates of the artifact (written as the X and Y values of where they are placed in relation to the top-left corner of the image) along with the object class or name of the artifact is generated in a text document alongside each dataset image [11] [12]. This "scrambling" of objects ensures a variety in dataset images and removes weeks of tedious labor [13] [14]. The NumPy random number generator generates all random numbers with certain specified limits for each parameter.

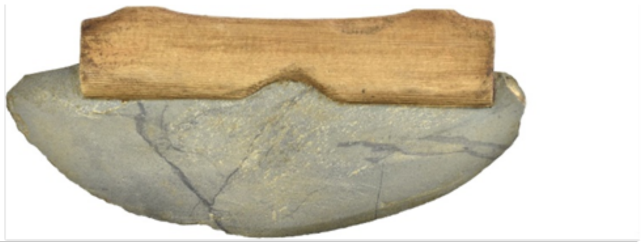

Fig. 2. Ulu cropped to remove the background.

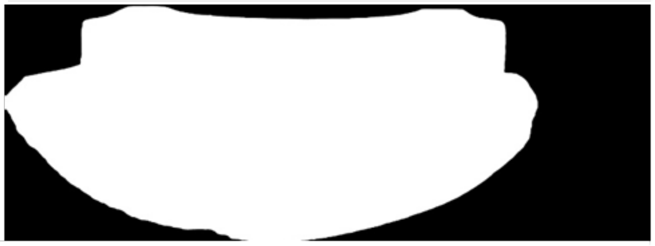

Fig. 3. Mask of ulu.

The programming and troubleshooting took roughly two weeks, and the image generation time depends on the processing power of the computer and the number of images one wishes to generate.

Unfortunately, we ran into certain issues in image generation when moving from a Linux Ubuntu machine to a Windows machine. To remedy this and any other operating system issues, we are packaging our application in a Docker image. Docker sets up an operating system kernel on any machine to run applications while minimizing OS differences [15]. So far, our addition of Docker has taken two weeks.

In the future, we hope to finish the generation of the synthetic dataset and begin training the YOLO model. After training the model on the artifacts at different angles, we will move onto validating the model—that is, feeding the model images and seeing how much the model has "learned." Finally, we test it in real-life scenarios [16]. Once the model has been properly trained, validated, and tested with all the artifacts, we will package the program into a Flask web application. Flask allows our program to access a website and fetch the information paired with each detected image. Training, validating, and testing is expected to take roughly two months collectively with the web development lasting for about half a year. During web development, we hope to compare our YOLO model with other YOLO and computer vision models to improve our product.

Additionally, this project has been worked on remotely. As such, we have relied on GitHub to share code and manage project updates and Google Colab to coordinate work plans weekly and test sections of code remotely. The Nalaquq GitHub will house the programs we created to mask artifacts, generate datasets, and train the YOLO model for any future developmental work. The web application will be implemented in the Nunalleq Culture and Archeology Center.

Discussion

As touched on in the previous section, the application would be customized to suit the Nunalleq Culture and Archaeology Center. The application would be used as a learning aide in the cultural center by combining technology with preexisting teaching methods. We hope to create a nonlinear experience that allows users to examine the exhibits at their own pace. Additionally, because most, if not all, of the artifacts have emerged from the permafrost, they may be made of wood or bear other composable materials. To prevent the deterioration of the collection, the application protects the artifacts while still providing any museum goers with a hands-on experience. And it is not just limited to the museum. New archeologists can potentially use the tool to identify artifacts that are already in the collection and gain knowledge of the items they are handling.

Like any type of software development, the ultimate goal is a working model of the application that can be easily distributed, scaled, and used in whatever or any environment necessary. However, a polished and market-ready product developed by a two-person team in the span of eight weeks is unrealistic. We do hope to get most of the back-end work of the application finished. Realistically, we should have a network that can recognize a handful of artifacts from the museum's collection. Currently, we are starting on preparing images and masks to train the YOLO model to recognize endblades (similar to arrowheads) and seal masks. Over the following two years, we hope to incorporate the rest of the collection while developing the front-end design work to transform the program into a proper Flask web application.

Conclusion

Overall, I am very blessed to have the opportunity to develop software as a freshman in college. The development of software, especially involving machine learning and computer vision, at this level is almost unheard of at other universities. And this application is not just a feather in my cap or an obscure piece of technology with a buried article to accompany it. I am using the skills I have garnered to build tools that will immediately be put into place to assist others in their pursuit of education. Additionally, this process is opening my eyes to the world of research and development in the STEM fields, preparing me for a future in computer science.

Acknowledgements

The authors would like to thank the Nunalleq Culture and Archaeology Center in Quinhagak, AK for their support and access to the photographs of their artifact collection and the Hampden-Sydney Office of Undergraduate Research for their funding of the project.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 9 May 2016.

[2] H. Zanini, "Custom YOLOv7 Object Detection with TensorFlow.js," 28 March 2023. [Online]. Available: https://towardsdatascience.com/training-a-custom-yolov7-in-pytorch-and-running-it-directly-in-the-browser-with-tensorflow-js-96a5ecd7a530.

[3] J. Li, J. Zhang, S. J. Maybank and D. Tao, "Bridging Composite and Real: Towards End-to-end Deep Image Matting," 27 October 2021.

[4] S. Kench, "What is White Balance — How it Works and Why It Matters," 5 June 2022. [Online]. Available: https://www.studiobinder.com/blog/what-is-white-balance-definition/#:~:text=White%20balance%20is%20a%20camera,so%20this%20helps%20correct%20it..

[5] T. Kummarikuntla, "Blue or Green Screen Effect with OpenCV [Chroma keying]," 1 March 2019. [Online]. Available: https://medium.com/fnplus/blue-or-green-screen-effect-with-open-cv-chroma-keying-94d4a6ab2743.

[6] L. Block, A. Raiser, L. Schon, F. Braun and O. Riedel, "Image-Bot: Generating Synthetic Object Detection Datasets for Small and Medium-Sized Manufacturing Companies," Procedia CIRP, vol. 107, pp. 434-439, 2022.

[7] T. Vorenkamp, "Understanding Exposure, Part 1: The Exposure Triangle," 26 January 2022. [Online]. Available: https://www.bhphotovideo.com/explora/photography/tips-and-solutions/understanding-exposure-part-1-the-exposure-triangle#:~:text=%E2%80%9CExposure%20Triangle.%E2%80%9D-,Exposure%20Triangle,(film%20or%20digital%20ISO)..

[8] C. Zimmermann, M. Argus, and T. Brox, “Contrastive Representation Learning for Hand Shape Estimation,” in *Pattern Recognition*, C. Bauckhage, J. Gall, and A. Schwing, Eds., Cham:

Springer International Publishing, 2021, pp. 250–264.

[9] J. Brownlee, "A Gentle Introduction to Object Recognition With Deep Learning," 27 January 2019. [Online]. Available: https://machinelearningmastery.com/object-recognition-with-deep-learning/.

[10] S. Gleason, S. Spickard, “Nalaquq/nanalleq_cv,” [Online]. Available: https://github.com/Nalaquq/nunalleq_cv.

[11] A. P, "How to Create Synthetic Dataset for Computer Vision (Object Detection)," 10 January 2022. [Online]. Available: https://medium.com/@alexppppp/how-to-create-synthetic-dataset-for-computer-vision-object-detection-fd8ab2fa5249.

[12] S. Borkman, A. Crespi, S. Dhakad, S. Ganguly, J. Hogins, Y.-C. Jhang, M. Kamalzadeh, B. Li, S. Leal, P. Parisi, C. Romero, W. Smith, A. Thaman, S. Warren and N. Yadav, "Unity Perception: Generate Synthetic Data for Computer Vision," 19 July 2021.

[13] M. Walia, "Overfitting in Machine Learning and Computer Vision," 31 October 2022. [Online]. Available: https://blog.roboflow.com/overfitting-machine-learning-computer-vision/.

[14] dewangNautiyal, "ML | Underfitting and Overfitting," 5 June 2023. [Online]. Available: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/.

[15] Docker Docs, "Docker Overview," [Online]. Available: https://docs.docker.com/get-started/overview/.

[16] E. d'Archimbaud, "Training, Validation and Test Sets: How To Split Machine Learning Data," 2023. [Online]. Available: https://kili-technology.com/training-data/training-validation-and-test-sets-how-to-split-machine-learning-data.