Bioinformatics, 2023, 19–4
doi: 10.1093/bioinformatics/xxxxx
Advance Access Publication Date: DD Month YYYY
Manuscript Category

Computer Vision for STEM Education

Cell Discovery with Magnets: Using YOLO for Interactive Learning

John Murphy

[1]Department of Computer Science, Address 4000 Central Florida Blvd, Orlando, FL 32816

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract
Motivation: The motivation is to have kids at an early age to have a fun and interactive time in biology. We know that for children it can be difficult to pay attention when learning about things that they can't even see or interact with. This is why I came up with the idea to get magnetic parts of a cell and allow children to interact and actively learn about the cell and what each part does with the help of the YOLO architecture to detect these pieces in real time. Allowing the students to have a fun and interactive way to learn about something they can't see or touch will help cement the knowledge.
Results: Sampling across children in a 4th grade setting, I was able to determine that the children were able to remember each piece of the cell from the magnetic set they were given to play with. They also seemed to enjoy their time learning about the cell, which I believe will make them more inclined to want to dive into the world of biology and maybe even computer vision. It's a way to help them see their future.
Availability:
Contact: jrm5399@knights.ucf.edu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction
With advancements in the worlds of computers, machine learning and artificial intelligence, the next obvious step is to use aspects of these technologies in the classroom. While some classroom learning is still stuck in centuries-old methods, it's easy to see we can improve upon student outcomes by utilizing the latest advancements in technology. This is a lofty goal, but as my project will demonstrate, it requires effort and diligence to be successful.

When given the option between improving an existing paper or pursuing a novel idea within the bioinformatics world, I wanted to work on a project that I believed would benefit the upcoming generation. As someone who had struggled with biology growing up because of a lack of tangible demonstrations, I wanted the next generation to enjoy themselves while learning about biology in a fun and interactive way. The way I handled this was I developed a custom dataset using a YOLO CNN model to detect different parts of a cell in real time, by creating a customizable synthetic image generation pipeline to create the images. From this I was able to generate a large and diverse dataset that allowed the YOLO model to accurately detect different parts of a cell in real time. Throughout this process, I encountered several challenges and obstacles, including varying object sizes, lighting conditions, and object orientations which led to various trial and error runs. However, by experimenting with different factors and persevering through all these obstacles, I was able to create a robust and accurate dataset that enabled my YOLO model to detect different parts of a cell with high accuracy in real time. It was also able to handle different distances and shadings. In this paper, I share my experiences and insights to provide guidance for others facing similar challenges in creating computer vision datasets.

2 Data Creation Method

Data creation is the heart of any successful machine learning project. Without accurate and relevant data, the results would be meaningless. So I spent a good amount of time to ensure that the data creation step was as reliable as I could make it.

In order to create and build an effective model that is able to handle various real-world scenarios such as distance, lighting, shading, backgrounds, movements, and various cameras, I needed to develop an efficient yet effective approach. The foundation of any computer vision model is the dataset, and when dealing with real-time events, there are many uncontrollable factors that can affect your model when trying to detect your objects. Therefore, I created a synthetic dataset generator to ensure that the YOLO model could handle most reasonable scenarios. This dataset generator takes in one image per class as input and then augments them to create a large and diverse dataset from each object. My dataset consisted of 8 classes which included: Centrioles, Golgi Bodies, Lysosomes, Mitochondria, Nucleus, Rough ER, Smooth ER, and Vesicles. In order to augment these input images to create diversity, I designed a function that takes a number as input and in return, will generate N images for each original image, creating a total of 8xN images where N represents each newly augmented image. The augmentation function rotates, adjusts lighting, and resizes the images randomly, allowing for a diverse dataset. With this approach, I was able to create a strong dataset that can better represent real-world scenarios.

After appending each newly created image onto a randomized background, I had to address a few additional tasks. First, I calculated the intersection over union (IOU) to prevent too much overlap between the object and the background whenever more than one image would be appended to the same background. I also needed to remove the original background from each augmented image, which involved calculating the mask of the object within the image, proceeding to cut it out by using the mask of the image, and then pasting the image onto the new background to create a single image within my dataset. However, I am not done yet, having to follow YOLO's ruleset, I then needed to create a corresponding text file alongside each image, containing five values within each text files. These values corresponded to: the ID of the object class detected, the x-coordinate of the center of the bounding box relative to the width of the image, the y-coordinate of the center of the bounding box relative to the height of the image, the width of the bounding box relative to the width of the image, and the height of the bounding box relative to the height of the image. This process ensured that the YOLO model would be able to accurately detect and localize each object within the image dataset. The interesting part about this was since all background images were converted to 1920x1080, they all had the same calculated height and width of 0.5.

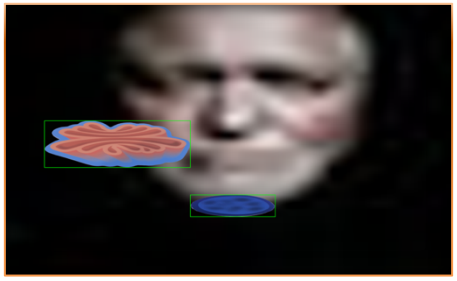

Fig. 1. **End Result of a Single Image in my Dataset created from Synthetic Image Generation**

After creating my dataset with randomly generated backgrounds from web scraping and applying each of my newly augmented objects onto these backgrounds, I had a total of 2,180 images, each with a corresponding text file which shows the object's class and bounding box coordinates. To train my YOLO model on this dataset, I had my dataset generator produce 545 images at a time. That allowed me to adjust the data augmentation numbers and also allow for varied background images since it would scrape for different images each time. After running my dataset generator 4 times, I was then able to come up with 2,180 uniquely generated images for my model to be trained on. Once I created my final dataset, the next step for me was to export this dataset and feed it into my YOLO model. Generally, when training any sort of CNN to detect objects, especially when you want the objects to be detected in real time, we must feed our CNN an abundance of data, but this alone won't be sufficient. I am passing through 2,180 uniquely created images but in order to have a robust model to work in real time, we must utilize something called "transfer learning" from the coco dataset. With the help of darknet it will not only create a solid model but it will also help to speed up the training process. Transfer learning involves utilizing pre-trained weights from other classification problems and applying them to my own dataset. To accomplish this, I set up a new Google Colab environment specifically for training my weights, which allowed me to quickly generate accurate weights for my dataset. By using transfer learning and a well-designed dataset, I was able to produce a highly accurate YOLO model capable of detecting objects in a wide range of real-world scenarios.

With the completion of this step, I felt ready to move on to the next critically important part of my project, which was my computer set-up.

2.1 Computer Setup

After creating my dataset and getting my resultant weights for my YOLO model which were generated with the help of transfer learning on the coco dataset and darknet, I then needed to set up my windows computer to run YOLO in real-time with a reasonable framerate. Running the model only on the CPU gave me a framerate of around 6 FPS, which was not ideal for interactive feedback, especially with children. In order to get my model to run efficiently, I would need it to run on my GPU. First, I had to determine how to approach this. After several hours of research, I had stumbled upon people with similar issues to mine and all their results were the similar. I discovered I needed to set up cuDNN and CUDA on my computer to effectively utilize my GPU. Setting up CUDA was not an easy task. I had to obtain the binaries from CUDA and then set them up in my system environment path. After hours of trial and error and consulting help forums, I was finally able to successfully get CUDA up and running alongside the SDK toolkit. Next, I had to activate parallel processing with cuDNN to further enhance the experience. This was also a similar process. After getting my trained weights and being able to run the YOLO model on my GPU, the final step was to get the model to run on my local computer with the trained weights which I had created from my synthetically generated dataset.

Once I had the model running on my GPU, I encountered a new challenge. The default settings for YOLO on my computer were not optimized for real-time inference, resulting in slow framerates and high CPU usage. To overcome this, I experimented with different settings, such as adjusting the batch size and image resolution, to find the optimal configuration for real-time object detection on my system. I also explored other optimizations, such as using the OpenCV DNN module to accelerate the model inference and reducing the number of layers in the YOLO architecture to improve the model's speed. Through trial and error, I was able to obtain significant improvement in model performance, achieving framerates of around 30 FPS while maintaining a high accuracy in object detection. These optimizations allowed me to run the YOLO model in real-time during my teaching sessions with the magnetic cell kit, providing instant feedback to the students and enhancing their learning experience. Overall, the process of optimizing the YOLO model for real-time inference was a challenging yet rewarding experience. It was satisfying to see the fruits of my labor in action during the teaching sessions.

2.2 Training YOLO Architecture

Diving into how all of this was possible, I briefly had mentioned earlier that I had set up training my weights using transfer learning from the coco dataset and had to import my dataset in order to get the custom trained weights, but I wanted to discuss in detail how I was able to accomplish this using google colab. The first step was getting my data usable within the colab environment. This was a simple step involving uploading my data to my google drive and then mounting my account to my colab environment. The next step was to get my environment to run within a GPU setting. Because this is colab, that was a simple switch of changing from CPU to GPU. Then, a few essential steps were necessary to train the model properly. The first part was to get the darknet repository. This was used to help with transfer learning. My YOLO model had learned through the darknet model, which has many pre-trained YOLO models that my own model could train off of. This allowed my model to be exposed to even a greater sized dataset than just the one I created. After getting darknet all set up and getting its make file working, the next step was to insert object names into my dataset for it to change the numerical value that is present within the YOLO text file into the actual name of the object we are wanting to detect. Afterwards, I needed to calculate the steps, batch size, and filters which are all a part of the training configuration file. Fortunately there was an optimal way to calculate all of this easily and efficiently. Then I needed to write all my image files into the training data folder for the model to then get started to train. I allowed my model to train for up to 30 hours where it would train on over 30,000 images. From this I was able to graph out the class loss over my 8 objects over time! As seen below in figure 2, this graph plots out all the class loss over time from all 8 of my objects over a 1000 epoch period.

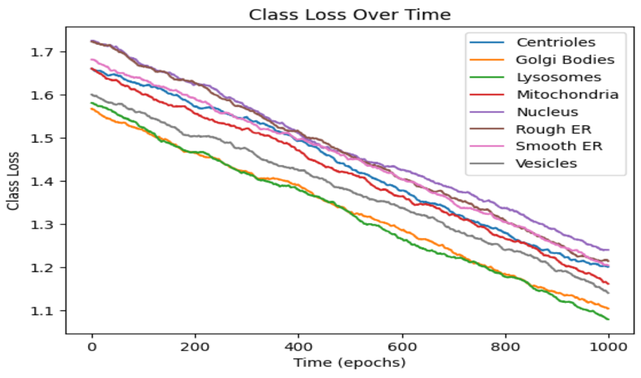

Fig. 2. **Relation between Loss and** ***Time for Class Loss.***

3 Results

Overall, it was an extremely rewarding experience with the children at Roosevelt Elementary school. With the help and supervision of Annie Holtrup, I was able to allow the students to play with the YOLO model in a fun and interactive manner, and they seemed fascinated with the results! Being able to have participants enjoy my hard work really paid off and allowed the students to have a good time in class. I'm optimistic they will have much better odds of retaining what they learned due to this highly visual and interactive experience. Afterwards, I had given them a quiz to see if they'd remember the names and shapes of the cell pieces we just played around with, and astonishingly the average was a 6.8 out of 8. With 22 participants within the class, this I believe was a great success. Essentially the vast majority of students retained almost all the information they were exposed to. Below in figure 3, are all the quiz results from the students represented in a bar graph. How we structured the quiz was we allowed the students to experience the YOLO model, giving each student their own turn to play around with it. Afterwards we administered the quiz and we allowed students to collaborate with one another to try out the quiz. Even though we allowed the students to work together in pairs, this still shows that the students were still able to recollect a majority of the information within a 30-minute time frame, since it was 30 minutes total for the students to play around in groups up at the white board while interacting with the model. Most students missed the similar looking objects such as the Lysosome for the Vesicle, or the Smooth ER to the Golgi Body. In spite of this I believe their retention was excellent. I gave each student their own quiz because I wanted it to be their choice to choose which they thought was the correct answer but allow for collaboration with one partner.

It would be interesting to see how this type of experience would translate for even older students such as middle schoolers or even high school age students. There may be differing problems that would result when trying to work with more intricate information and more advanced students. However I feel the results seen at the 4th grade level are encouraging.

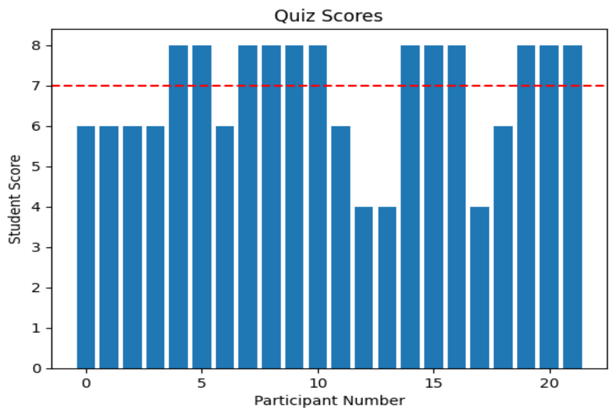

Fig. 3. **Participants Quiz Scores.**

3.1 Why YOLO?

Diving into the architecture of the YOLO model, you may ask why I decided to go with the YOLO model when there are much stronger models out there such as ResNet or RetinaNet. The reason is I needed to balance strength vs efficiency for the model. When working with real-time data, each frame is getting fed through my model and needing to get processed and handled in real time in order to show to the user the video feed with little to no latency of what it is trying to detect. Yes there are stronger models that have higher accuracy when trying to detect objects within images, but the goal of this project is to detect these frames in real-time and allow the user to enjoy and get nearly instant feedback from the model. This allows them to see if they were correct when playing around with the model. From figure 4, you can see how well YOLO can run in real-time against other models within real-time.

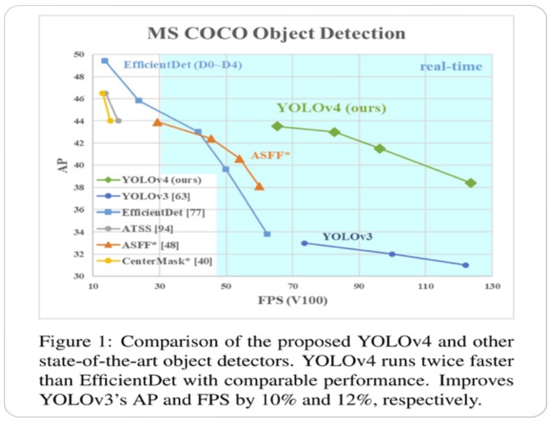

Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

Fig. 4. CNN comparisons for real-time usage.

From figure 4, on the X axis it shows how fast and how many FPS YOLO can handle and compares them to other state-of-the-art models. There's really no comparison as to which model I should choose for real time detection. It's YOLO hands down.

Reflecting on this experience, I believe that this project has shown me the importance of bringing education to life through technology. It was amazing to see how much more engaged the students were when they were able to interact with a computer vision model in real-time. As technology continues to advance, I believe that we should take advantage of these advancements to make education more interactive and engaging for students. That's the best way to increase their retention and understanding, and hopefully get them interested in future careers. In the future, I hope to continue working on projects that bridge the gap between technology and education, and I believe that this project was just the beginning of what's to come. Overall, this experience taught me a lot about the importance of education and making learning interactive and engaging for students. By using technology like the YOLO model, we were able to bring abstract concepts to life in a way that was fun and memorable for the students. I hope to have more opportunities to work with students in the future and continue to explore the ways in which technology can be used to enhance education. I think teachers will welcome the assistance of this type of technology in the classroom.

3.2 Improvements on Experiment

Looking back at my trials, there were several areas where improvements could have been made. Firstly, I should have obtained permission from parents to record or film in the classroom. Although I was able to conduct the experiment, privacy concerns meant that I couldn't record any of it. Secondly, we were limited to a 45-minute time frame, which made it challenging to strike a balance between giving the students enough time to take the exam and allowing them to play with the model. While the model performed well in varying lighting conditions, I only tested it during the day, and this could have biased the results. Although the model worked well in a well-lit room or a even a more dimly-lit classroom, I cannot confirm if it performs better under certain lighting conditions.

Another issue I had encountered was the tripod being in the middle of the room which was being used to hold the camera. Due to its positioning the students would sometimes accidently bump into the tripod or move it. Even though I tried to keep it unobtrusive, it was still within reach of the students, and they often moved it or played with it. This could have affected the accuracy of the results, as the camera was supposed to be stationary while being in front of the whiteboard. Additionally, I did not test the model at different distances from the whiteboard while at the school, even though I attempted this at home. This could have provided valuable insights into how the model would perform under varying conditions. However, I think that overall, the model performed well under the given circumstances.

3.3 Accuracy

Now that we have investigated the overall experiment, and discussed how I created my own dataset and how the overall event went, I'd like to dive into how my model performed looking at visualizations with various factors. Figure 5 from below shows me trying out the model after I had just finished training the model. I trained the model in various intervals, testing it out every 5-10 hours and checking on its results. In figure 5, you can see me trying out my model's final weights to detect the nucleus. I did various things such as walk back and forth until the camera couldn't detect the object, or I'd have someone dim the lights and try various altering factors while doing this over a 60-second interval to get my averages. The 0's show when the model couldn't detect the nucleus anymore.

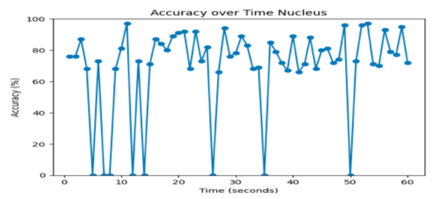

Fig. 5. Accuracy over a minute timespan with varying conditions

To ensure accuracy in varying environments it boiled down to a few core components which was the dataset that I generated and through the help of transfer learning, which allowed my dataset to work well with only around 30 hours of training.

3.3.1 Data Improvements

I believe there are some improvements I could make for next time. I would try doing a more extreme data augmentation route. What I mean by this is that currently my data does flips, dimming, movement, and resizing, but I limited that to a certain degree. In the future I might choose to use an unrestricted augmentation function so that my dataset can come into contact with a lot more extreme outcomes. Not only would this expose the YOLO model to a higher variation to each object, it would also let me create a more diverse and larger dataset with even more unique images for my model to train from.

In hindsight, there were a few areas where I could've improved my approach to the project. Even though my synthetic image generator did a excellent job of creating a diverse set of images for my dataset, I feel that I could have done more to try and introduce additional variability into the data. This could have been done through means of trying more extensive data augmentation, such as adding more types of transformations and introducing more extreme changes to the images. Examples of this would be heavy distortion, random occlusion, or even bigger changes to the lighting conditions. These additional changes I believe would have enabled the YOLO model to better generalize in different scenarios and would have made it more robust to noise and outliers in the input data.

Another area where I could have tried and made improvements is in the choice of the dataset itself. While I was able to generate a large number of synthetic images, it is possible that the dataset could not fully capture the range of possible realistic scenarios that the YOLO model would need to handle within the real world. In particular, the model might struggle with certain types of objects or in certain lighting conditions that were not well-represented in the dataset. To address this, I could have either collected additional data from real-world sources or used a more diverse set of 3D models to generate synthetic images that better cover the full range of object types and scenarios. Additionally, I could have taken steps to ensure that the dataset was balanced across different object types, to ensure that the model did not become overly biased towards certain classes of objects.

3.4 Comparison to Other Approaches

In order to evaluate the effectiveness of our custom-trained YOLO model on the magnetic cell kit, we compared it to another, and more common approach used within a school setting. The approach we compared it to was having the students examine images of the actual parts of the cell that you'd generally see within a textbook. This method is a lot less engaging and less likely to grab the students' attention.

Furthermore, we compared our custom-trained YOLO model to a pre-trained YOLO model from the coco dataset weights. Essentially I wanted to show that I needed to train my own model because the typical coco weights have so many identification features for generic objects such as cars and planes, but it couldn't detect the cell magnets I had purchased from the internet. I knew that this would be the case but I wanted to test it just to be sure it was necessary to have custom trained weights or not. As seen in figure 6, however, when I got the graph, I only left the nucleus object in frame and never moved it.

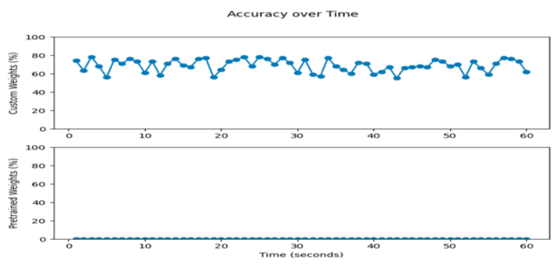

Fig. 6. Accuracy over a minute timespan custom vs pretrained

Overall, our custom-trained YOLO model provided a more efficient and accurate approach compared to traditional manual identification and other deep learning models commonly used in the field.

Conclusion

After doing this experiment, I really enjoyed the process of being able to develop my own custom-trained weights on a magnetic cell kit that I had gotten from Amazon to allow students to interact and have fun in real-time. Combining machine learning with biology in a novel way was a difficult yet exciting task that I was certainly willing to take on. During my time throughout this entire process, I tried and failed many times until I had a good YOLO model that was able to handle most real-world situations. From adjusting my dataset more than two or three times, to learning about how to create a robust dataset there were many challenges to overcome in the process. I was able to not only create a reliable synthetic image generator that would also give me the YOLO text files alongside each image, I also was able to then get the YOLO model to run on my GPU hardware on my windows computer which was very difficult to set up. Then I got permission to come in for a class period over in Tampa to teach the students about the parts of the cell and help them identify these parts with my YOLO model. Overall I believe this was a great learning experience and I really enjoyed my time doing this project.

Insights

Some insights I had gathered from this project would include:

1. Combining machine learning with biology is a novel and exciting way to approach scientific education.

2. Developing a custom YOLO model requires significant experimentation to achieve reliable results.

3. Creating a robust dataset is crucial for training a YOLO model to accurately detect cell components.

4. GPU hardware is essential for efficient YOLO model training, and setting up the necessary hardware and software can be a challenge.

5. Synthetically generated images can be an effective way to generate training data for a YOLO model.

6. The use of a magnetic cell kit can provide students with an interactive and engaging way to learn about cell components.

7. Teaching students to identify cell components with a YOLO model can be an effective and innovative way to teach biology.

Despite all the challenges I had endured, the process of developing and implementing a YOLO model with a purpose of teaching children scientific education was definitely rewarding and enjoyable. Not only did this provide a unique and innovative approach to teaching biology, but it also now opens up new opportunities for anyone to explore the intersection of technology and science. By teaching students how to identify cell components with a YOLO model, they can develop valuable skills in computer vision and machine learning, which are increasingly important in many fields. Moreover, students can gain a deeper understanding of the biological processes and functions of cells through the interactive and engaging experience provided by the magnetic cell kit and the YOLO model.

Overall, this project demonstrates the potential for technology to enhance scientific education and inspire students to pursue careers in STEM fields. By continuing to explore the possibilities of machine learning and other advanced technologies, we can continue to improve the way we teach and learn about the world around us. I am excited to see what the future holds for the intersection of technology and education, and I hope to be a part of it.

One thing to keep in mind is that if this type of process were to be used commercially, we would need to find a way to package it in an accessible way, and also find a way to deal with the large weight files. If I continued to pursue this in any meaningful way, it would have to be commercialized in such a way that other computers could easily use this technique. In order to go mainstream it obviously would need to be adaptable to varying computer processors. One way to handle that is to use a server, in which I could host the weights in a database, and the computer could fetch the weights from there. That would allow any computer to utilize these files without straining the computer's CPU and GPU resources.

Overall I found the project very rewarding and informative, and I think the future of education will be greatly impacted by advancements in these areas. It's a way of having students' education come alive before their very eyes, and to see that for myself has been an amazing experience.

To sum up, I'd like to quote Andrew Yan-Tak Ng, a machine learning pioneer and founder of Landing A.I., who said, *"Deep-learning will* *transform every single industry. Healthcare and transportation will be transformed by deep-learning. I want to live in an AI-powered society. When anyone goes to see a doctor, I want AI to help that doctor provide higher quality and lower cost medical service. I want every five-year-old to have a personalised tutor."*

References

Meel, V. (2023, January 2). Yolov3: Real-time object detection algorithm (guide). viso.ai. Retrieved April 29, 2023

Redmon, Joseph. YOLO: Real-Time Object Detection. pjreddie.com/darknet/yolo.
COCO - Common Objects in Context. cocodataset.org/#home.

Yin, Tsung Li. "COCO - Common Objects in Context." Coco dataset.org, cocodataset.org/#home.

Lyashenko, Vladimir. "Data Augmentation in Python: Everything You Need to Know." neptune.ai, Apr. 2023, neptune.ai/blog/data-augmentation-in-python.

Miller, Caroline, and Dave Anderson PhD. "Improving Behavior in the Classroom." Child Mind Institute, Aug. 2021, childmind.org/article/improving-behavior-classroom.

Google Colaboratory. colab.research.google.com.

Redmon, Joseph. Darknet: Open Source Neural Networks in C. pjreddie.com/darknet.

Brownlee, Jason. "A Gentle Introduction to Transfer Learning for Deep Learning." MachineLearningMastery.com, Sept. 2019, machinelearningmastery.com/transfer-learning-for-deep-learning.