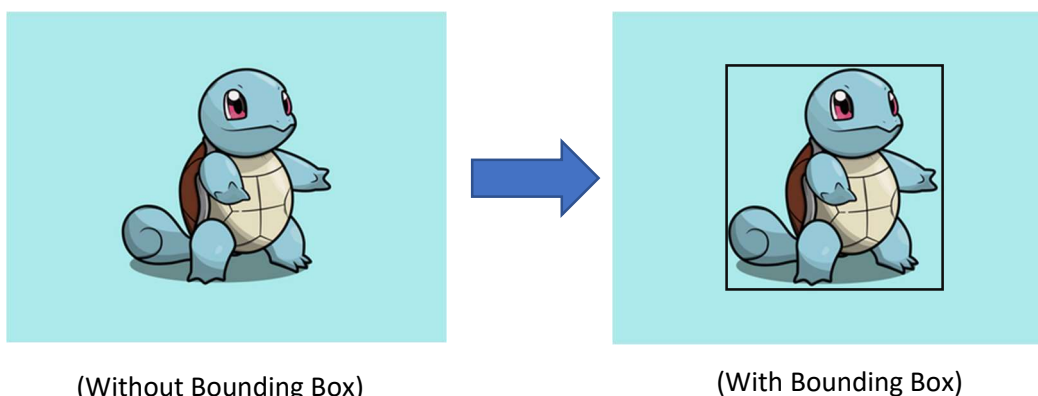





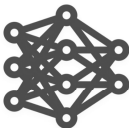
Training CNN Networks on YOLOv5 to Identify Pokémon:

I knew going into this project that the main obstacle was finding a way to correctly identify objects. Fortunately for me YOLOv5 turned out to be the perfect solution. YOLOv5 is essentially a collection of object detection models and datasets that when given image training data will correctly train and create custom models to identify said object [1]. I decided to use YOLOv5 not only because it's primary focus is image detection, but also because it supports using custom data to train on. Because Pokémon wasn't in any default database, I needed the ability to train on custom data.

To start, I needed to find a large amount imagery for each of the four Pokémon that I chose. I was able to find a Pokémon database that supported what I was looking for [2]. Unfortunately, it was limiting in my options because it only had 151 Pokémon to select from. I decided on four of the most popular ones, Pikachu, Bulbasaur, Charmander, and Squirtle, because their datasets had high file counts and because if I needed to find more data, I had a higher chance of finding unique images. With the data in hand, I needed to create labels for them. In YOLOv5 labels are basically text files that give additional information to the network training it. In my particular case I made my labels bounding boxes. The bounding box information that I provided were rectangles that surrounded the Pokémon, telling the system what's contained within the box is what we are going to be identifying. An example of a bounding box that I provided looked something like this:



Secondly, I needed a network structure to train my data on. Instead of creating my own network structure I decided to use the pre-built ones that YOLOv5 provides. I did this for a couple of reasons. First, unlike project one, where we trained a network to detect a single object, I am going to be training a network to recognize 4 separate objects. I don't have the time to figure out what kind of network I would need to structure to work for every single one. Secondly, the networks and weights that YOLOv5 provides have already been tested on over 80 images in the default coco dataset. For me this was enough to just use the pre-made structures and weights provided. I ended up using *YOLOv5s* and *YOLOv5m*. Details about these networks can be found in the image provided below and here [3]:

			
Small YOLOv5s	Medium YOLOv5m	Large YOLOv5l	XLarge YOLOv5x
14 MB _{FP16} 2.0 ms _{V100} 37.2 mAP _{COCO}	41 MB _{FP16} 2.7 ms _{V100} 44.5 mAP _{COCO}	90 MB _{FP16} 3.8 ms _{V100} 48.2 mAP _{COCO}	168 MB _{FP16} 6.1 ms _{V100} 50.4 mAP _{COCO}

Training images for Pikachu:

Starting off with Pikachu should have been a good indicator that this project wasn't going to be a simple one. Since Pikachu has large ears and a tail that sticks out much further than his body, it was quite difficult to put bounding boxes around him. When it came to training Pikachu was on the lower end in terms of accuracy. I personally believe that this was due to having to put the bounding boxes around his ears and tail which consequently picked up additional background nonsense. In total, I ran 8 simulations on Pikachu, switching the structure, image size, and batch size. Here were my results:

Run #	CNN Structure	Image Size	Batch Size	Epochs	Accuracy
1	yolov5s.pt	416	16	200	78%
2	yolov5m.pt	416	16	200	83.5%
3	yolov5s.pt	468	16	200	78.8%
4	yolov5m.pt	468	14	200	81.5%
5	yolov5s.pt	520	14	200	82.4%
6	yolov5m.pt	520	10	200	85.2%
7	yolov5s.pt	624	14	200	78.2%
8	yolov5m.pt	624	8	200	83.3%



Looking at my results and my test data, it was apparent to me that YOLOv5m was more accurately identifying Pikachu. However, when looking at the runs using YOLOv5s I noticed that it was also very good at identifying Pikachu. But the issue was that it was identifying several other Pokémon as Pikachu. To me this is a major issue because even though it identifies Pikachu with a high accuracy, it has the chance to declare the other 3 Pokémon as Pikachu as well. Overall run #4 and #6 were the best for Pikachu.

Training images for Bulbasaur:

Second, was Bulbasaur. Bulbasaur doesn't have the long ears or tail that Pikachu has. Bulbasaur is much more compact and concise. The largest thing on him is the bulb on the back which actually fits quite nicely with the bounding boxes. Bulbasaur had the best results out of all the Pokémon by a longshot. Like Pikachu, I ran 8 simulations for Bulbasaur, switching the structure, image size, and batch size. Here were my results:

Run #	CNN Structure	Image Size	Batch Size	Epochs	Accuracy
1	yolov5s.pt	416	16	200	95.2%
2	yolov5m.pt	416	16	200	96.6%
3	yolov5s.pt	468	16	200	95.5%
4	yolov5m.pt	468	14	200	97.4%
5	yolov5s.pt	520	14	200	92.7%
6	yolov5m.pt	520	10	200	96.3%
7	yolov5s.pt	624	14	200	92.4%
8	yolov5m.pt	624	8	200	97.7%



Unlike Pikachu, the results for Bulbasaur were pretty consistent. Using YOLOv5m provided slightly better accuracy results, but when looking through the test images they were all pretty identical. This is very fortunate because I will be needing to pick a CNN and image size that will work for all of them. Since Bulbasaur had high accuracy in all sections I won't have to worry about accounting for the best possible CNN for him.

Training images for Charmander:

Charmander was by far the most difficult Pokémon to train. Just like Pikachu, Charmander also has a very long tail that extends the bounding box. However, unlike Pikachu, Charmander doesn't have any defining physical characteristics besides his flame on the tail. I initially thought that the flame on the tail would help distinguish Charmander from other Pokémon but I found out all the images have the flame depicted differently. The flame on the tail actually became a hinderance and worked against identifying him. This resulted in me having to scrap my first dataset and create a new dataset that had different images and new labels that didn't include the flame. Here were my results:

Run #	CNN Structure	Image Size	Batch Size	Epochs	Accuracy
1	yolov5s.pt	416	16	200	90.1%
2	yolov5m.pt	416	16	200	93.4%
3	yolov5s.pt	468	16	200	90%
4	yolov5m.pt	468	14	200	89.7%
5	yolov5s.pt	520	14	200	91.7%
6	yolov5m.pt	520	10	200	92.4%
7	yolov5s.pt	624	12	200	91.6%
8	yolov5m.pt	624	8	200	90.2%



Looking at my results for the second Charmander dataset was shocking. The accuracy went from averaging 70% accuracy on the first dataset to averaging 90% accuracy on the second dataset. Additionally, the test results ended up looking much better. Other Pokémon weren't identified as Charmander as often anymore. Finally, just like Bulbasaur, all the resulting accuracies were high, and with that I won't have to take into account which trained CNN is the best.

Training images for Squirtle:

Squirtle was much easier to put bounding boxes around than Charmander. Additionally, Squirtle also has several defining features that Charmander doesn't. Like Bulbasaur's bulb Squirtle's tail doesn't extend far from his body. Additionally, Squirtle has both a brown belly and shell to work with. Having these two defining features and a diverse image pool to select from, Squirtle was one of the simpler Pokémon to train. Here were my results:

Run #	CNN Structure	Image Size	Batch Size	Epochs	Accuracy
1	yolov5s.pt	416	16	200	79.5%
2	yolov5m.pt	416	16	200	83.2%
3	yolov5s.pt	468	16	200	80.2%
4	yolov5m.pt	468	14	200	87.5%
5	yolov5s.pt	520	14	200	84.7%
6	yolov5m.pt	520	10	200	86.3%
7	yolov5s.pt	624	14	200	79.7%
8	yolov5m.pt	624	8	200	82.3%



After having Charmander's first dataset identify Squirtle as a Charmander so often I didn't have high hopes going into Squirtle. But I was genuinely surprised to see how well Squirtle did. When running test data, Squirtle was often correctly identified by 90% or higher. Additionally, Squirtle was often the only one identified over other Pokémon in the images. However, unfortunately he was also classified as a Charmander majority of the time. Charmander and Squirtle both have similar facial structures but I was hoping for a bigger discrepancy when it came to identifying the two.

Final Results:

After running over 32 different simulations for the various Pokémon, I decided I had enough information to start training all at once. From my previous results I knew that YOLOv5m produced the higher accuracies and gave me the best results for majority of the Pokémon. I took that information and decided to run only 4 simulations using YOLOv5m. Here were my results:

Run #	CNN Structure	Image Size	Batch Size	Epochs	Accuracy
1	yolov5m.pt	416	16	300	89.9%
2	yolov5m.pt	468	14	300	89.1%
3	yolov5m.pt	512	10	300	89.3%
4	yolov5m.pt	624	8	300	87.8%



To my surprise all of the networks seemed to have similar accuracies. I didn't expect an even split between all of them, but rather each run improving on the last. With similar accuracies I had to leave it up to which one gave me the best test results. I found the best network to be network #1 which performed well on both test images and a test video.

Conclusions:

I learned a tremendous amount from this project. Before starting, my understanding was that structures and variables played the biggest role in training intelligent networks. My understanding now is that, that isn't the only thing that plays a major role when it comes to training networks. Having to create and modify my datasets several times I know that having a well-constructed dataset plays a huge role when it comes to training networks to be accurate. The other thing that I learned was how to make the most out of having limited recourses. YOLOv5 puts a heavy emphasis on using a GPU, and I don't have the greatest GPU in the world. I had a limitation that I wasn't used to having. I learned that I couldn't train with large images and a high number of epochs. I had to limit myself to what my GPU was capable of which led to me learning how to balance. Throughout this project I trained several networks with several image sizes and epochs that varied drastically. With that I learned that the best results came from balancing my variables.

I had several challenges come up throughout this project. The biggest one that I ran into was underestimating how long things would take. When I first proposed my schedule for this, I planned on sticking to it, but that didn't end up happening. I found out that each simulation would take much longer to run than I initially expected. This was especially true near the end when the four simulations took a combined 20 hours to run. Having these long training times turned out to be a huge setback. Another major challenge that I ran into was with my datasets. I had a sufficient number of images for each Pokémon, however, not all the images in the datasets were valid. Many of the images were not clear and had bizarre backgrounds. I was only able to use about 65% of each dataset and had to find the rest of the images on my own. Doing this took several hours and wasted more time than I would have liked.

Overall, this has to be the best project that I have ever done for school. I not only learned a tremendous amount but I found myself very happy with what I was doing. I was very nervous going into this project because there were so many unknowns but I can safely say that I accomplished what I set out to do. I am very proud of myself for completing this.

References:

- [1]: <https://github.com/ultralytics/yolov5/>
- [2]: <https://www.kaggle.com/thedagger/pokemon-generation-one>
- [3]: https://pytorch.org/hub/ultralytics_yolov5/