CS 510 - Wild Wild Animals
Matthew Ernst & James Yost

Overview and Setup:

The goal of this assignment was to further our understanding of convolutional neural networks and how they classify. Specifically, for this assignment, we looked at a dataset that was collected by Barry M. Goldwater Range (BGMR). A sample of this dataset can be seen in Figure 1. From this dataset, we were given the freedom to apply a convolutional network of our choosing. For this task, we utilized the open-source platform Tensorflow 2 to build our models. This paper explains the results of our network as well as pretrained networks with this dataset.

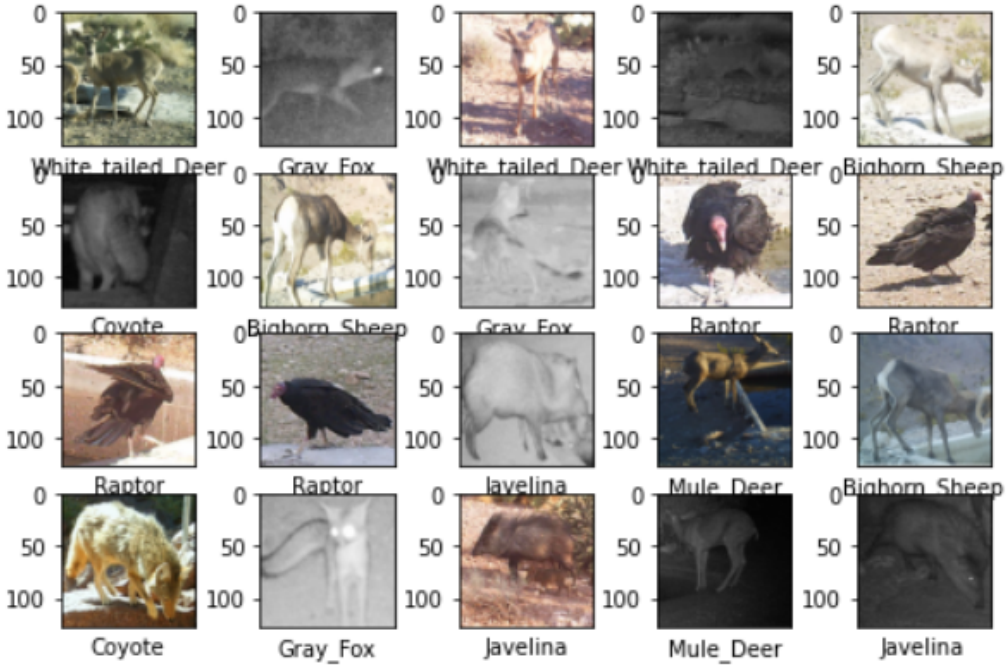

Figure 1

BMGR Data Chips:

The first part of this assignment was looking at a smaller sample of the BMGR dataset. This dataset contained 1791 image chips that were hand-labeled by students. A distribution of the data is as follows, Bighorn Sheep: 200, Bobcat:200, Coyote: 200, Gray Fox: 210, Javelina: 200, Mule Deer: 179, Raptor: 402, and White-Tailed Deer: 200. With our dataset understood, we began the construction of our own defined neural networks utilizing Tensorflow 2.

Our Networks:

To begin building our networks we first had to develop a few functions to help facilitate building the networks. First, we defined labels for the dataset to be used further for training. Second, to improve the speed of our training we had our code run a GPU (Nvidia 2080). In order for this to work, however, we had to set the memory growth for the GPU to true. This was due to the memory constraints on the GPU. Having this GPU to increase or speed was vital for our networks in this assignment.

With our GPU set and the classes labeled, we were able to begin tailoring the dataset. We first partitioned the data into sections for training, testing, and validation. Due to this specific dataset being somewhat small, we decided to partition the data with k-fold cross-validation. This allowed us to expand on the data. For this assignment, we went ahead and did 3-fold cross-validation. With our data split, we were now able to decide on the architectures of the networks.

We decided on the architectures of our convolutional neural networks based on previous experiments in other classes such as CS440 and CS445. Our first network was a simple 3 hidden layer with 15 nodes for each. Second, was another 3 hidden layer networks but with 128 nodes. We chose 128 nodes due to the size of the dataset. Our third network was a small bow-tie shape of 128 for the first hidden layer, 64 for the middle, and 128 for the last. This one was chosen to look at this widely used network shape and see its reliability for this type of problem. The last network had an increase in hidden layers to 5 with 10 nodes in each layer. This architecture was chosen to see the differences in nodes as well as layers. Each of these models used relu for their activation functions.

Finally having the network structure defined, we began training. We ultimately decided on running these networks for 7 epochs. This was mainly chosen as a time constraint for training multiple networks, even with utilizing the GPU. Figure 2 shows a summary of the models and their accuracy for train, test, and validation.

	architecture	Epochs	Train Accuracy	Validation Accuracy	Test Accuracy
0	[15, 15, 15]	7	0.816936	0.620971	0.602818
1	[128, 128, 128]	7	0.780147	0.611771	0.614760
2	[128, 64, 128]	7	0.762656	0.605110	0.603718
3	[10, 10, 10, 10, 10]	7	0.553351	0.521161	0.523003

Figure 2

From this figure, we can see some interesting results. The first being that all of the networks overfit the data. This can be seen in the train and validation accuracy. All of the training accuracies for these models were higher than the validation accuracies. The worst model was the last one with 5 hidden layers with 10 nodes at each layer. This model had a test accuracy of roughly 52%. The best model was the 3 hidden layers with 128 nodes at each layer with a test accuracy of 61.4%. However, all of the models besides the last one were all within the same test accuracy. Figure 3 shows the confusion matrix for the best model from these results.

	Bighorn_Sheep	Bobcat	Coyote	Gray_Fox	Javelina	Mule_Deer	Raptor	White_tailed_Deer
Bighorn_Sheep	95.4 %	2.3 %	0.8 %	0.0 %	0.8 %	0.8 %	0.0 %	0.0 %
Bobcat	8.5 %	37.3 %	11.0 %	23.7 %	2.5 %	12.7 %	4.2 %	0.0 %
Coyote	22.0 %	8.5 %	28.8 %	6.8 %	0.0 %	27.1 %	5.1 %	1.7 %
Gray_Fox	1.4 %	13.0 %	4.3 %	71.0 %	0.0 %	10.1 %	0.0 %	0.0 %
Javelina	15.5 %	28.2 %	12.7 %	8.5 %	16.9 %	5.6 %	11.3 %	1.4 %
Mule_Deer	1.3 %	12.0 %	14.7 %	1.3 %	1.3 %	64.0 %	0.0 %	5.3 %
Raptor	15.4 %	0.0 %	0.7 %	0.0 %	0.7 %	0.0 %	81.9 %	1.3 %
White_tailed_Deer	13.2 %	11.3 %	11.3 %	1.9 %	0.0 %	13.2 %	1.9 %	47.2 %

Figure 3

From this figure, we have quite a distribution of classification for each animal. First, the bighorn sheep had the best classification at 95.4%. This made sense to us as this animal has fairly predominant horns that distinguish it from the other animals. From here, the second-highest classified animal was the raptor. Similar to the bighorn sheep, this made sense to us as it is unique in its appearance compared to the other animals in the dataset. What we found interesting was that the bobcat, coyote, and javelina were all classified the worst. And in some cases, they were classified incorrectly into each other's other categories. We believe this to be due to the dataset. These animals are a bit smaller and closer to the ground. Given that some of these photos were taken at night and in varying positions. The network could be confused by these factors and classify these animals incorrectly.

Pretrained Networks:

Once we were convolutional neural networks were finished training, we looked pretrained models to compare accuracy for this dataset. For us, we wanted to look at convolutional neural networks that are used frequently in deep learning communities. Ultimately, we ended up picking 4 models; VGG19, Inception V3, ResNetV2, and ResNet50V2. Luckily since we were using Tensorflow 2, loading these models in and training them was quite simple. In order to train these, however, we first had to cut the top off of these networks. Again, Tensorflow 2 made this task simple and allowed us to train quickly with the added GPU support.

As stated earlier, we had the networks run for 7 epochs for time and to effectively compare our models versus the pretrained models. Figure 4 shows the results of these pretrained networks.

	architecture	Epochs	Train Accuracy	Validation Accuracy	Test Accuracy
0	<function VGG19 at 0x7f3e712bb4c0>	7	0.956764	0.852356	0.850286
1	<function InceptionV3 at 0x7f3e7130fe50>	7	0.739650	0.667199	0.663976
2	<function InceptionResNetV2 at 0x7f3e7130f9d0>	7	0.654321	0.673883	0.660319
3	<function ResNet50V2 at 0x7f3e712adf70>	7	0.985742	0.809112	0.804967

Figure 4

This figure holds some interesting results. First, all the models increased in accuracy compared to the models that we developed. This is no surprise to us as these models have been fine-tuned to be the best in class for solving problems such as these. However, all these models overfit the data just like our models. However, the overall test accuracy had increased. The worst model was the ResNetV2 at 66% testing accuracy and the best being the VGG19 at 85%. Figure 5 shows the confusion matrix of the best model, the VGG19.

:

	Bighorn_Sheep	Bobcat	Coyote	Gray_Fox	Javelina	Mule_Deer	Raptor	White_tailed_Deer
Bighorn_Sheep	100.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %
Bobcat	0.8 %	84.7 %	7.6 %	1.7 %	1.7 %	0.0 %	3.4 %	0.0 %
Coyote	7.7 %	12.3 %	61.5 %	3.1 %	6.2 %	4.6 %	1.5 %	3.1 %
Gray_Fox	0.0 %	6.4 %	0.0 %	91.0 %	2.6 %	0.0 %	0.0 %	0.0 %
Javelina	0.0 %	4.2 %	5.6 %	0.0 %	90.3 %	0.0 %	0.0 %	0.0 %
Mule_Deer	3.1 %	0.0 %	10.8 %	0.0 %	1.5 %	56.9 %	1.5 %	26.2 %
Raptor	0.0 %	0.0 %	0.0 %	0.0 %	0.0 %	0.8 %	98.4 %	0.8 %
White_tailed_Deer	3.2 %	0.0 %	4.8 %	0.0 %	4.8 %	3.2 %	0.0 %	83.9 %

Figure 5

Compared to our developed model's confusion matrix, we had a dramatic increase in classification accuracy. First, the bighorn sheep are classified correctly every time. Again as stated earlier, with the unique horns of this animal we believe it to be the easiest to classify. We see an increase in classification for the bobcat and coyote but an extreme increase for the Javelina to 90.3%. However, the most interesting result we found was the mule deer classification went down. Upon further investigation, it can be seen that the second closest it was classified as was a white-tailed deer which makes sense as these are both deer with small differences between each other.

Ensemble:

For the last part of this portion of the assignment, we wanted to explore a different way of finding higher accuracy with models. We ended up looking into ensemble methods. This stems from both of us taking CS540 along with this course and wanting to mix the classes together and test areas of artificial intelligence and machine learning. We do not have a visual for this however we can explain the process. We had 3 voters for the models with 3 hidden layers and 15 nodes each. 3 voters were chosen to make sure there would not be any ties. We let these models train and come back with their accuracy. From here we checked their prediction against the testing to see the percentage of correctness and allow them to vote. Due to the data being split we had variations of accuracy but we average 70% with our best being 80%. This didn't really change our previous results too drastically but it did help. A benefit to this learning style is that it allows voting for increasing accuracy and it is fairly quick to implement and execute.

Larger BMGR Data:

For the second part of the assignment, we were tasked with the same as part 1. However, this time our dataset was increased. This dataset contained 29385 images compared to the 1791 images from the original. Due to this increase image size, we had some changes to our developed networks. Also, this dataset did not have the animals cropped out the image. Due to this, we explored ways of using pretrained networks to try and identify animals in the original photos and crop them. However, as shown in Figure 6, the network was able to recognize some animals but would also classify random parts of images as an animal. Due to this, we relied on our ensemble technique to help increase accuracy which is explained further on.

Figure 6

Our Networks:

To begin our convolutional neural network build for this portion of the assignment, we followed the same process as before. However, there were some tweaks to the architectures chosen. This mainly due to fact that there was a lot more data being pushed through the networks. Figure 7 shows a summary of the networks.

	architecture	Epochs	Train Accuracy	Validation Accuracy	Test Accuracy
0	[15, 15, 15]	7	0.888229	0.782991	0.783519
1	[256, 256, 256]	7	0.914313	0.788726	0.788760
2	[256, 128, 256]	7	0.919434	0.785306	0.786055
3	[64, 64, 64, 64, 64]	7	0.816477	0.780695	0.779385

Figure 7

As stated, we followed the same process as before but increased nodes at certain levels of the architectures. This can be seen in the last three models. Overall, we found very similar results to that of the original dataset. Namely, all of the models overfit again. However, the overall testing accuracy for all the models did increase. We believe this was due to the increase in data to be used for training. The model with 3 hidden layers and 256 nodes at each layer did the best with 78.8% testing accuracy. Figure 8 shows this model's confusion matrix.

	Bighorn_Sheep	Bobcat	Coyote	Gray_Fox	Javelina	Mule_Deer	Raptor	White_tailed_Deer
Bighorn_Sheep	89.4 %	0.0 %	4.2 %	0.0 %	0.0 %	3.6 %	2.8 %	0.1 %
Bobcat	5.2 %	15.9 %	27.7 %	35.4 %	1.5 %	9.2 %	2.6 %	2.6 %
Coyote	2.4 %	0.9 %	77.3 %	1.6 %	0.3 %	11.4 %	5.8 %	0.5 %
Gray_Fox	0.4 %	5.9 %	6.5 %	79.8 %	0.6 %	5.1 %	1.4 %	0.2 %
Javelina	0.0 %	2.5 %	5.8 %	0.8 %	52.9 %	27.3 %	5.0 %	5.8 %
Mule_Deer	1.4 %	0.3 %	9.8 %	0.6 %	0.6 %	85.4 %	0.6 %	1.3 %
Raptor	5.5 %	0.2 %	13.1 %	1.3 %	0.6 %	1.8 %	74.9 %	2.6 %
White_tailed_Deer	0.7 %	0.4 %	2.8 %	0.7 %	3.2 %	8.0 %	8.5 %	75.7 %

Figure 8

Unlike the previous part of this assignment, we have a slightly different confusion matrix. First, we see that the bighorn sheep classification went down. We believe this may be due to the cropping of photos. However, the trend of bobcats, coyotes, and javelinas being classified incorrectly into each other is still similar. Overall, the classification was a bit more sparse.

Pretrained Networks:

Just like our developed networks, we followed the same process for the pretrained networks as before. Figure 9 shows the results of these networks.

	architecture	Epochs	Train Accuracy	Validation Accuracy	Test Accuracy
0	<function VGG19 at 0x7f3e712bb4c0>	7	0.888739	0.830872	0.830344
1	<function InceptionV3 at 0x7f3e7130fe50>	7	0.847512	0.788232	0.789219
2	<function InceptionResNetV2 at 0x7f3e7130f9d0>	7	0.832948	0.775846	0.773311
3	<function ResNet50V2 at 0x7f3e712adf70>	7	0.958960	0.810692	0.809909

Figure 9

These pretrained networks compared to the original dataset are a bit different. Overall, the accuracy went up but specifically for the VGG19 model, the accuracy went down. We this particular as the other model increased in accuracy more than likely to access of more data. However, even with this decrease, it still was the best model at 83% testing accuracy. Figure 10 shows this model's confusion matrix.

	Bighorn_Sheep	Bobcat	Coyote	Gray_Fox	Javelina	Mule_Deer	Raptor	White_tailed_Deer
Bighorn_Sheep	92.5 %	0.1 %	2.8 %	0.0 %	0.0 %	2.8 %	1.8 %	0.1 %
Bobcat	1.8 %	13.9 %	28.1 %	32.7 %	1.8 %	15.3 %	1.8 %	4.6 %
Coyote	2.6 %	0.9 %	73.9 %	1.5 %	0.3 %	14.6 %	6.1 %	0.2 %
Gray_Fox	0.7 %	2.4 %	10.4 %	79.1 %	0.2 %	5.9 %	0.2 %	1.1 %
Javelina	0.0 %	5.7 %	4.3 %	2.8 %	44.7 %	26.2 %	2.1 %	14.2 %
Mule_Deer	1.1 %	0.0 %	3.9 %	0.2 %	0.1 %	93.7 %	0.2 %	0.9 %
Raptor	1.7 %	0.2 %	10.6 %	1.3 %	0.2 %	1.2 %	84.3 %	0.5 %
White_tailed_Deer	0.4 %	0.2 %	4.2 %	0.4 %	1.3 %	18.1 %	3.8 %	71.7 %

Figure 10

From this confusion matrix we again similar trends to the part one pretrained matrix. First, the bighorn sheep's classification increased but not to 100%. The most interesting we found was that the bobcat classification went down by 2%. We found this odd as the pretrained networks typically do much better but in this case, it seemed to fail for this specific case. We believe again a lot of this incorrect classification could be because of the cropping of animals. Seeing as bobcats tend to be smaller, it may have been harder for the networks to identify them in images.

Ensemble:

To combat the image cropping issue, we again applied ensemble to try and vote on models to increase accuracy. For this dataset, we used 9 voters at 3 hidden layers with 15 nodes each. After running through and voting we averaged accuracy of 77% and the best at 84%. Similar to the part results, we don't see a big increase in accuracy but it is interesting to see how this type of learning can change the accuracy.

Conclusion:

Ultimately this assignment helped us have a firm grasp of convolutional neural networks. Namely, that differing dataset sizes can change models quite drastically. As well as, having fuller images and non-cropped ones can decrease accuracy. We had a huge appreciation for people that go through and hand label images for datasets. We also were able to have CS540 and CS510 worlds collide with ensemble learning and see the impact that this style can bring.