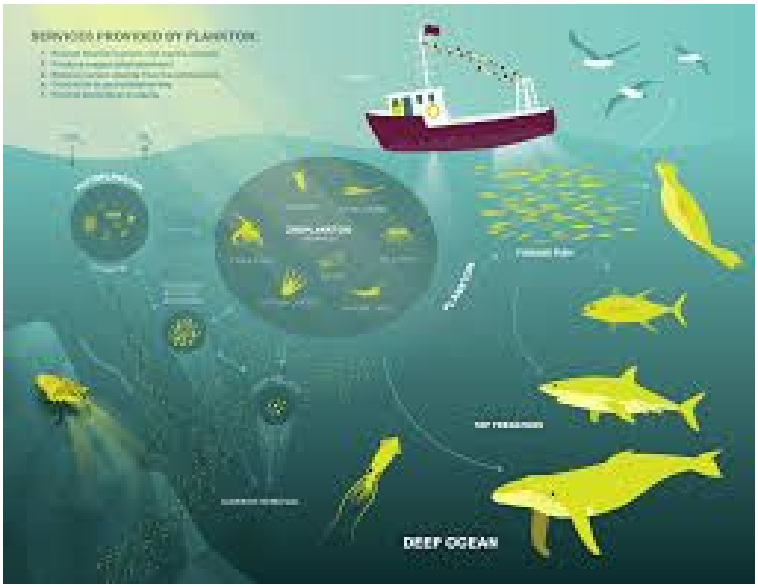
Plankton Classification using Inception-v3 and Keras



COGS 181

Brian Henriquez, Simon Fong, Wilson Tran

Abstract:

Plankton are critically important to our ecosystem, accounting for more than half the primary productivity on earth. Forming the foundation of aquatic food webs including those of large, important fisheries, loss of plankton populations could result in ecological upheaval as well as negative societal impacts. Plankton's global significance makes their population levels an ideal measure of the health of the world's oceans and ecosystems. The traditional methods for measuring and monitoring plankton populations are time consuming and cannot scale to the scope necessary for large-scale studies, requiring new and improved approaches. One approach is through the use of an underwater imagery sensor. This towed, underwater camera system captures microscopic, high-resolution images over large study areas, creating the dataset for our project.

Manual analysis of this dataset would take a year or more to manually analyze the imagery volume captured in a single day. However, with machine learning tools we can analyze the data at speeds and scales previously thought impossible. Thus, using Inception-v3 and Keras, we automated the image identification process of plankton to allow analyses to be made faster, allowing quicker assessment of ocean and ecosystem health. With Inception-v3 as the base, we vary the Inception layers and split of training-validation-testing data to compare the effects on the testing accuracy and loss of the models.

Introduction:

Based on the National Data Science Bowl competition hosted by Kaggle, we wanted to create a image classification neural network that classified grayscale images of plankton into one of 121 classes. These images were created using an underwater camera to determine which species occur in an area and how common they are. The images obtained were already processed by a segmentation algorithm to identify and isolate individual organisms and cropped accordingly. As a result the size of an organism in the resulting image is proportional to its actual size and does not depend on the distance to the lens of the camera.

The approach for our solution was based on Google's Inception-v3 model, a convolutional neural network trained for the ImageNet Visual Recognition Competition. From there, we varied the amount of Inception layers, the data split between training, validation, and testing, and added additional layers to see if we can improve the results from the base Inception-v3 model. We did not expect that adding layers would improve the results from the base Inception-v3 model due to it losing data that was already optimized to a certain extent. However, adding additional Inception layers might be able to optimize the Inception-v3 model for the plankton dataset opposed to the ImageNet dataset.

Dataset + Experiments (Data and Problem Description):

We worked with Oregon State University's Hatfield Marine Science Center's plankton dataset which is composed of approximately 30,000 labeled images of plankton from 121 categories. Each image was processed so that they contain only a single organism/entity. Some difficulties of classifying this dataset include high variation in species, organisms having any orientation in 3D space, existence of non-living debris, noise in the ground truth due to human error, and even having classes of unidentifiable objects.

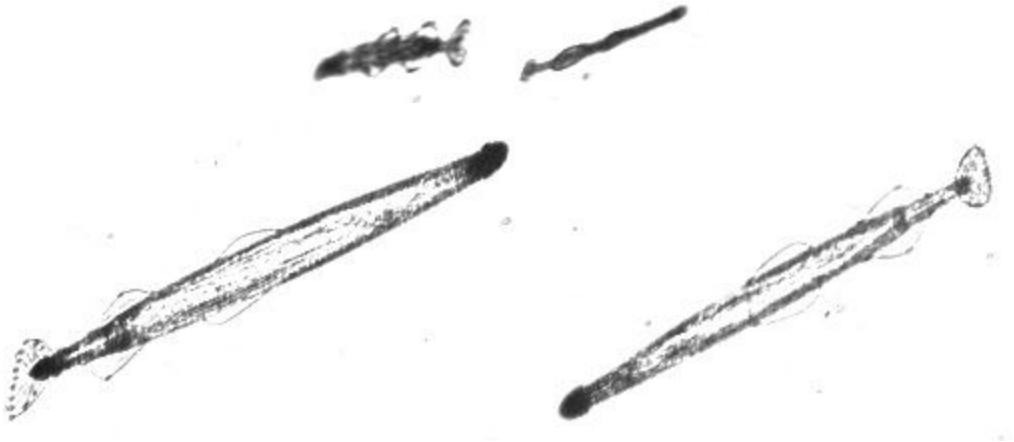


Figure 1: Chaetognath Sagitta

In order to use these images in training our model, we needed to resize all of them to 299x299, the shape that the model expected. At first, we resized the images during runtime every time we trained the model. However, that was a naive idea as resizing 30,000 images every time would take approximately 50 minutes. After several runs, we realized we should resize them all once and save them as a new dataset. This reduced our dataset loading time to 2 minutes, a significant decrease in total amount of time spent computing which allowed us to train models much faster.

Figure 2: Resized Chaetognath Sagitta

An interesting note about this dataset is that many of the different classes fall into subcategories of Plankton such as Fish and Crustaceans. Figure 3 shows how all the different classes relate to each other.

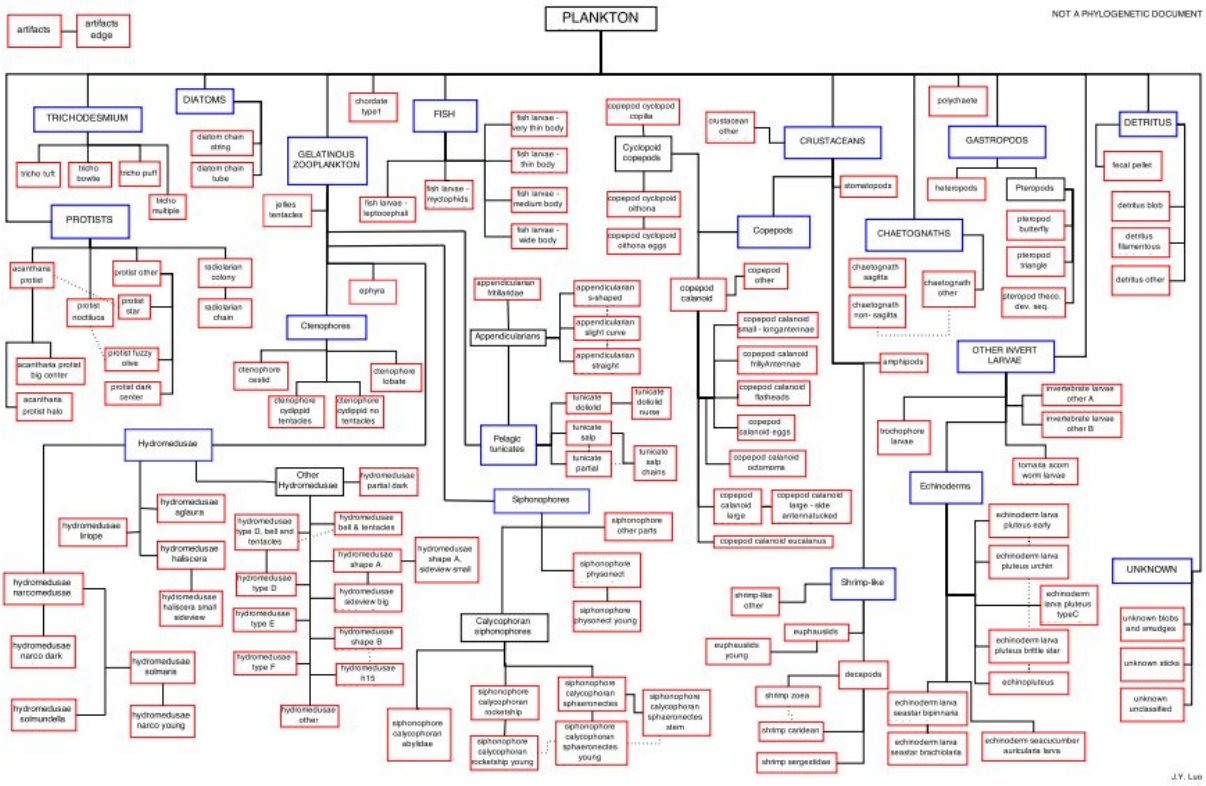


Figure 3: Species Relationships

Method/Architecture/Algorithm development/Hyperparameters:

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Figure 4: Inception V3 Architecture

The default architecture used for all of the following subsections is shown above. This is the vanilla Inception v3 module with two fully connected layers. For all of the following tests, the following hyperparameters were held constant: learning rate = 10^{-4} , train/validation/test split = 10/5/85, batch size = 50, epochs = 100, SGD optimizer with momentum, and weights initialized to the pre-trained tiny-image set. The reason for choosing to begin with the weights

initialized to the tiny-image dataset was to allow for faster learning since some features may have been shared between the two-datasets. Finally, we decided to initialize the network without the fully connected layer in the end in order to be able to design our own fully connected output (which was varied in the proceeding section). In general, the default network will be that with the classification tail as shown above.

Varying Inception Layer Count

Google has prepared several versions of the Inception module that can recognize either smaller or larger features as they are applied on the network. While it was also possible to modify the initial Inception layers on the Inception v3 network, we believe that doing so would heavily diminish the performance of the network (although this is mostly due to the fact that we were unsure if the changes that were to be made would cause an informational bottleneck as indicated on the Inception v3 paper). Therefore, changes were only performed on the Inception layers that were intended to recognize larger features (which we will call Inception B from now on):

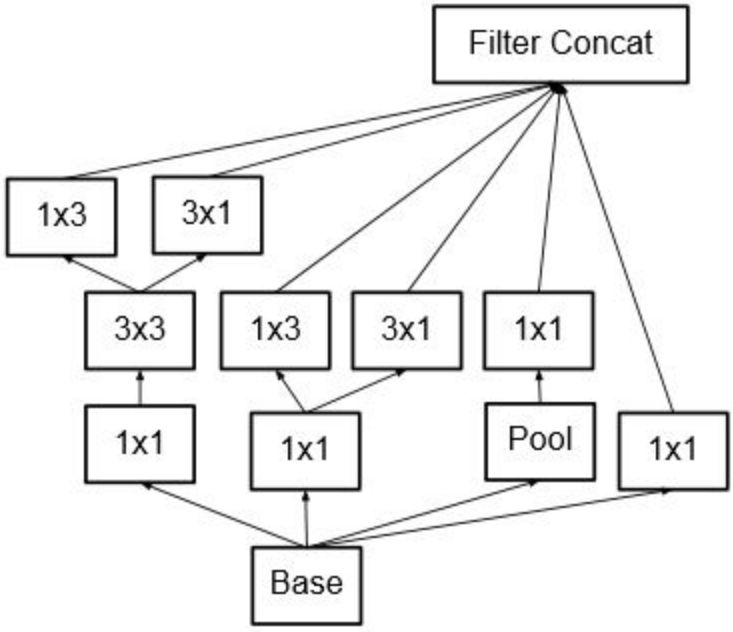


Figure 5: Inception B Module

The purpose of this particular Inception B module was to recognize larger features without the use of a larger convolution and keep up with the spirit of reducing the dimensionality of the inputs for each layer slowly to promote faster training. According to the Inception v3 paper, splitting up the original 3x3 convolutions into individual 1x3 and 3x1 convolutions would promote the learning of higher dimensional representations. This can be somewhat reasoned by seeing how splitting up this convolution results in more information available for the next layer.

By varying the number of these Inception layers, we believe that we would be able to achieve higher accuracy by promoting the learning of much more higher dimensional features.

In order to be able to make comparisons between each of the different models, the hyperparameters were kept the same as indicated above. The only major change was that for our initial testing, only 10 epochs were used (the reason for this being that if 10 epochs were not used, then all training would take several days to complete!). Furthermore, we decided to test the change along with a slight change of the number of neurons in the first fully-connected layer. There was no reason behind this; however, we felt that perhaps some interesting results can come from this variation. As we will soon see, the Inception B module is quite important for the recognition of features in this dataset (although initialization may have been an issue for the models with removed Inception B modules).

Varying Fully-Connected Layers

In our next set of trials, we wanted to see what effect varying the amount and type of fully-connected layers would have on the model's accuracy and loss. We kept all layers of Inception V3 constant except for the last three layers where start modifying them to the varying amounts of dense and types of dense layers. Additionally the added layers we kept constant were flattening the neurons without pooling and the final classification layer of 121 fully-connected neurons with softmax activation. For this experiment, we did three trials: one with a fully-connected layer of 256 neurons and relu activation, and a dropout layer with probability 0.5, another trial with fully-connected layer of 1024 neurons and relu activation, and a dropout layer with probability 0.5, and the final trial with a fully-connected layer of 256 neurons and relu activation, a dropout layer with probability 0.5, 5 more fully-connected layers of 256 neurons and relu activation.

Varying Training/Validation/Test Dataset Splitting

In order to better understand the results, we also trained the model using a larger dataset. To control the variation, hyperparameters such as learning rate, batch size, number of epochs, and weights initialized were kept the same. The new train/validation/test split that was used for comparison is 25/5/70. The reason for this split was to train with more data while not having to wait extensive periods of time for the results of a variation. By running these comparisons, we can further extrapolate whether or not additional data would be statistically significant when varying the amount of layers of the base Inception-v3 model.

Evaluation Metrics:

We used Testing Accuracy and Testing Loss as metrics to quantize the effectiveness of our new model when comparing to the base Inception-v3 model. The Testing Accuracy is given as:

$$Testing\ Accuracy\ = \frac{\#TestingImagesCorrectlyLabeled}{\#TestingImages}$$

and Testing Loss as:

$$Testing\ Loss\ =\ -\sum_{k}(\ d_k log(y_k)\ +\ (1-d_k)\ log(1-y_k)\ ),\qquad y_k = \frac{exp(\sum_j w_{kj}^{(2)} h_j^{(1)})}{\sum_k exp(\sum_j w_{kj}^{(2)} h_j^{(1)})}$$

By using testing accuracy and testing loss as evaluation metrics, we can observe the current state of the model and better understand how changing parameters will modify our convolution neural network. With this newfound knowledge we have a more grounded reasoning on how to implement new modifications to improve our model.

Discussion:

Varying Inception Layer Count

Inception-B Count	**-2 (0 Total)**	**-1 (1 Total)**	**0 (2 Total)**	**+1 (3 Total)**	**+2 (4 Total)**	**+3 (5 Total)**
Accuracy (1024)	12.6%	11.3%	28.5%	23.9%	27.1%	22.9%
Accuracy (2048)	10.8%	10.8%	30.4%	20.9%	26.2%	19.5%

As seen above, the first set of tests were to see whether varying the Inception-B layer count would improve or worsen the accuracy on the plankton dataset. For this particular series of tests, 10 epochs were used (mostly to see whether it was worth training the model for a larger amount of time). Out of interest, we also decided to vary the fully-connected layer slightly by varying the number of neurons slightly. As expected, the models using two or more Inception layers seemed to generalize to the testing set much better than those using less layers. However, one interesting feature to notice is that using an even number of layers seemed to produce higher than those with odd layers around them (i.e. two Inception B layers performed much better than using one or three of the same layer). Unfortunately, the reason for this cannot be explained with our current knowledge, but it would certainly make for an interesting discussion later on. Given the small number of training epochs, the number of neurons used in the first fully-connected layer did not seem to make too much of a difference (that would be considering the fact that the higher neuron count would likely need more training to adjust to some stable weights).

In the above, we noticed that the best performing models were those with two and four Inception B layers. Therefore, we decided to conduct some longer tests using these two models. The following shows the result of training these models for 100 epochs with the same dataset:

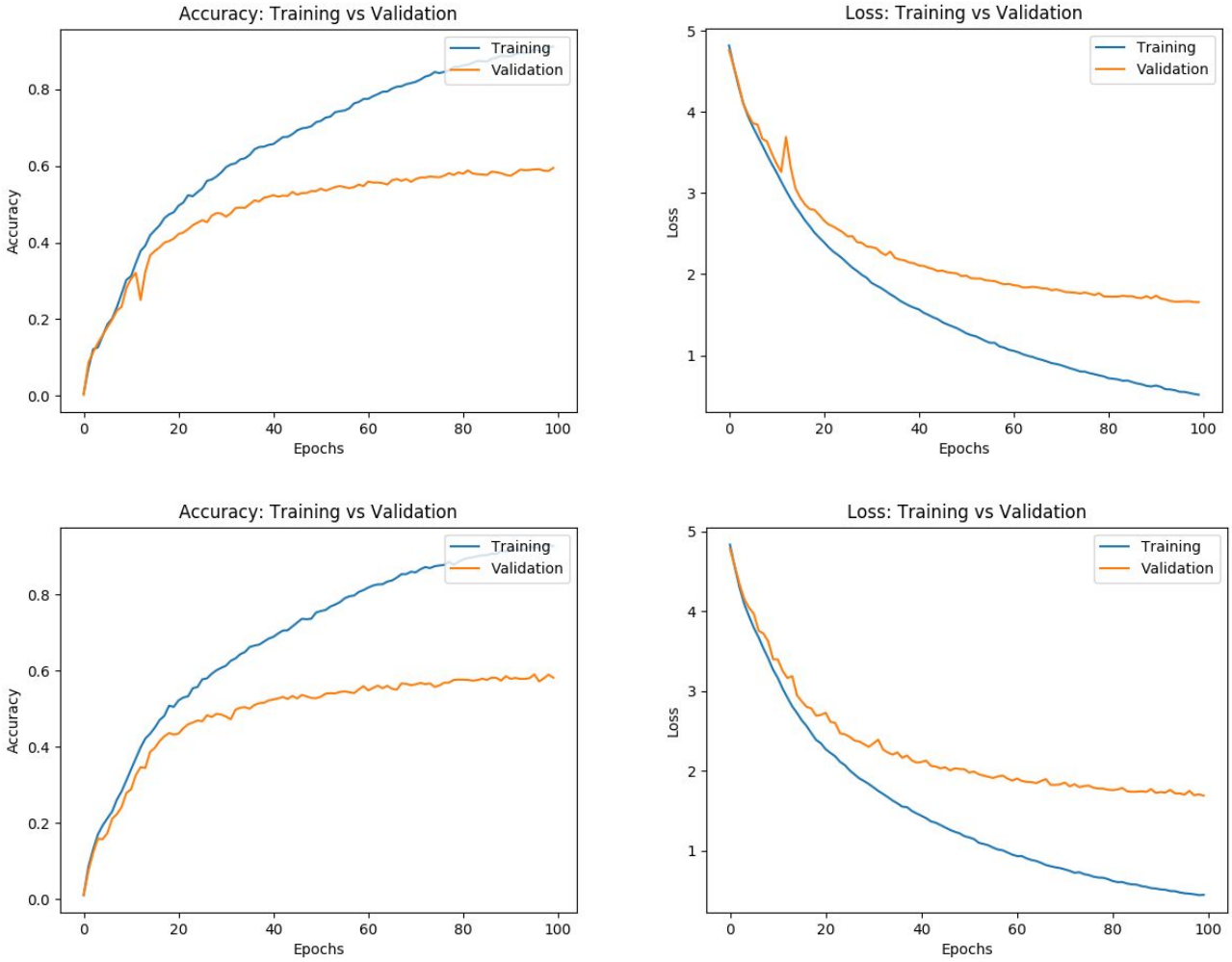


Figure 6: Training Accuracy and Loss with 2048 Neuron Hidden Layer (above) and 1024 Neuron Hidden Layer (bottom) [Default Inception v3]

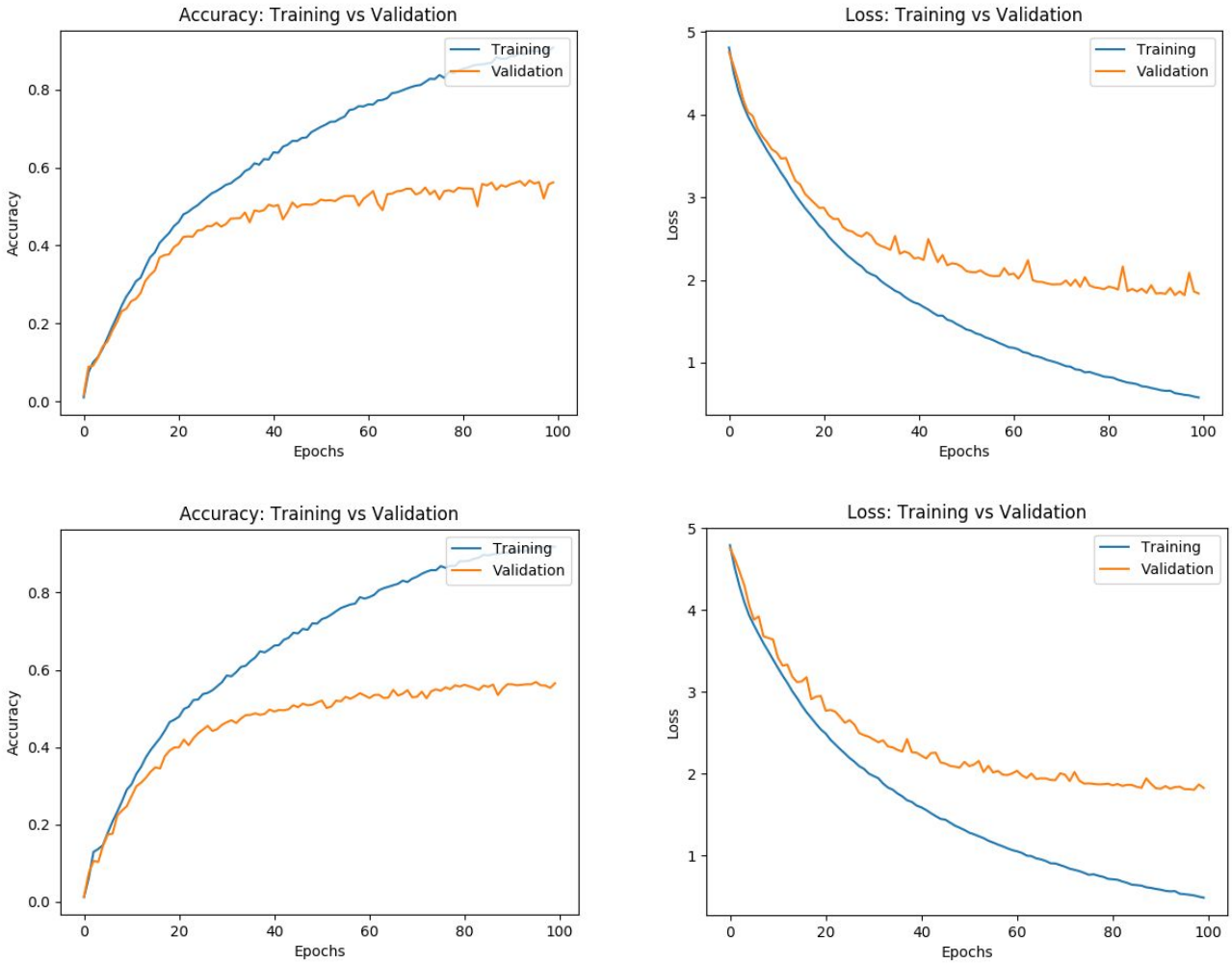


Figure 7: Training Accuracy and Loss with 2048 Neuron Hidden Layer (above) and 1024 Neuron Hidden Layer (bottom) [4 Inception B Layers]

The two best performing models look quite similar in their testing and validation results, so the following tabulates their performance with respect to the testing dataset:

Inception-B Count	**0 (2 Total)**	**+2 (4 Total)**
Accuracy (1024)	61.3%	58.8%
Accuracy (2048)	60.8%	58.0%

Unfortunately, it seems as though our hypothesis was not entirely correct. The original Inception v3 model seems to perform better without the two extra Inception B layers. Even more

interestingly, the model with 1024 hidden neurons in a single hidden layer seemed to perform better consistently (with about the same margin or difference between the layer difference). For the most part, the hidden neuron difference will be ignored, but it could use some future investigation. Regarding the differences in accuracy depending on the Inception B layer count, there are a few possibilities - either the model with more Inception B layers needs more data in order to generalize better (in which case the fact that the other inception layers had pretrained weights contributed to their better values) or the model itself is inherently worse at classifying (but that cannot be concluded without running the same tests without the pre-trained weights).

Varying Fully-Connected Layers

Final Set of Layers	**Original Inception V3**	**Flatten Fully-Connected 256 Dropout 0.5**	**Flatten Fully-Connected 1024 Dropout 0.5**	**Flatten Fully-Connected 256 Dropout 0.5 5(Fully-Connected 256)**
Accuracy	60.8%	58.9%	59.4%	55.5%
Loss	1.62	2.01	1.83	2.86

In the above table, we see the effect of what varying the fully-connected layers has on the testing accuracy and loss. We see that in both respects the unchanged Inception V3 model performed better than when we meddled with its architecture by adding our own layers. This was not surprising since, we believe that Inception V3 was very well thought out and fine tuned compared to our decision to just try these new changes. However, the accuracy differences of the original with the single 256 and single 1024 are 1.9% and 1.4% respectively, which is not a significant difference could change with respect to each other easily if we were to use different hyperparameters. Interestingly though, the model with an additional 5 fully-connected layers performed significantly worse at with a 5.3% difference in accuracy. We think this may be due to the additional layers making the model overfit to the training data, however, we are not quite sure. The losses for each model seem to fit with how the accuracies performed in that the original model had the lowest loss and the one with an additional 5 fully-connected layers had the highest loss.

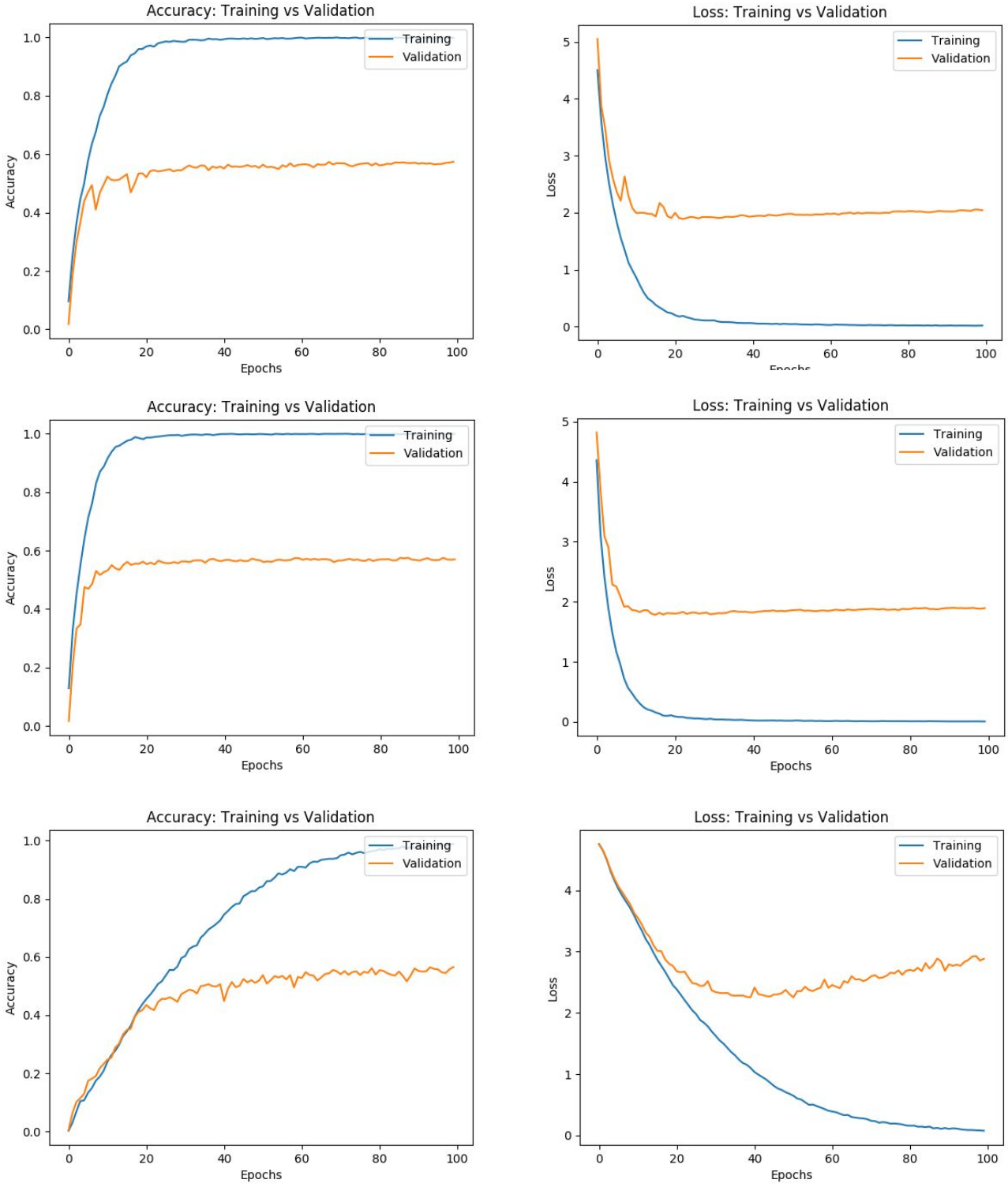


Figure 8: Training Accuracy and Loss with 1 Fully-Connected 256 Neurons (top)
Training Accuracy and Loss with 1 Fully-Connected 1024 Neurons (middle)
Training Accuracy and Loss with 5 Additional Fully-Connected 256 Neurons (bottom)

In Figure 8, we can see that in all the models, the training and validation accuracies climb together and at a certain point, the validation accuracy starts plateauing. The plots for the two models with only 256 neurons in their fully-connected layers seem to have been much more noisy than the model with 1024 neurons in its fully-connected layer. The single 256 fully-connected model was very noisy while the one with 5 more of those same layers was still noisy, but in smaller amounts rather than large spikes.

In terms of loss, one interesting note is that in all the loss plots, the validation losses predictably rapidly dropped in the beginning, however, they did not stay at the low loss. Near the end, their validation losses started to creep up. In the model with an additionally 5 fully-connected layers, its validation loss curved back up significantly from a low of approximately 2.5 back to approximately 2.9. We do not know if this is an effect from having too many fully-connected layers back to back or some other cause, but it is odd.

Varying Training/Validation/Test Dataset Splitting

	5Dense256relu 10/5/85	**5Dense256relu 25/5/70**	**Conv+MaxPool 10/5/85**	**Conv+MaxPool 25/5/70**
Accuracy	55.5%	62.6%	56.3%	65.0%
Loss	2.86	2.75	2.20	1.91

When comparing the data, it is clear that having more data to train with improves the testing accuracy, as shown in the table above. This experiment compares varying data splitting when attaching the following: Flatten layer, Dense layer with 256 units and a relu activation, Dropout with a rate of 0.5, followed by 5 Dense layers with 256 units and a relu activation. The second variation consists of: 2D-Convolution with 32 filters and a kernel size of 3x3, followed by a 2D-Max pooling with a pool size of 3x3 and a 2x2 stride, a Flatten layer, a Dense layer with 256 units and a relu activation, and finally a Dropout layer with a rate of 0.5.

The outcome is expected with the larger training dataset being more accurate. It was surprising to see that a convolution layer without batch normalization is more accurate than 5 dense layers which greatly affected the base model. However, the elapsed time to run the Convolution layer and Max Pooling is longer than the 5 Dense layers, though providing about 1~2% increase in accuracy. The loss is also significantly lower compared to the 5 Dense layers model. This is especially shown when comparing the larger training dataset size. This proposes that continuously adding Dense layers to the model can be detrimental to the model and perhaps other methods to compress the model should be used instead.

Comparing the time elapsed, the model with 5 Dense layers is marginally faster than the Convolution and Max Pooling model. For the 25/5/70 split, the 5 Dense layer model took 2 hours and 34 minutes. On the other hand, the Convolution and Max Pooling model took 2 hours and 43

minutes. When comparing the time difference between 10/5/85 split and 25/5/70 split, the 10/5/85 split for the Convolution and Max Pooling model took 1 hour and 13 minutes while, which is a little less than half the time of the 25/5/70 split. Comparing the accuracy, this shows that the results received from the 10/5/85 split are statistically significant because a larger training dataset size does not significantly improve the results. In another experiment, a 45/10/45 split resulted with a validation accuracy of 68% by the 56 epoch. We were not able to complete the training however due to the pod crashing. From our results, we can extrapolate that the great increase in training data would result to a testing accuracy of around 80%. We can visually confirm that the increase in training data size does not significantly affect the shape of the data as shown in the plots below in Figure 9.

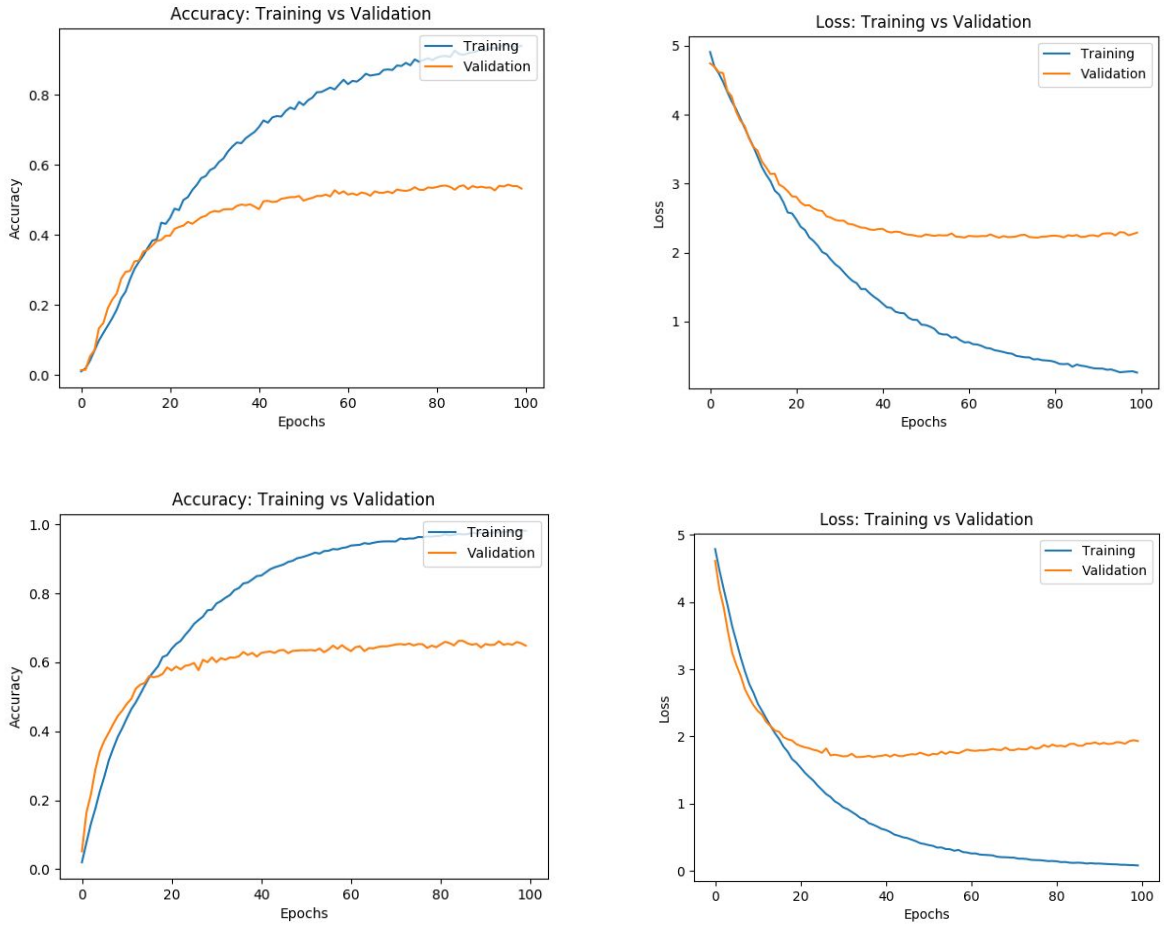


Figure 9: Convolution w/o Batch Normalization and Max Pooling 10/5/85 (Top) 25/5/70 (Bottom)

Conclusion:

The purpose of this report was to attempt to improve the generalization capabilities of the Inception v3 network for the plankton dataset. As was shown in the discussion section of the report, varying the number of Inception B layers does have a large effect on the model itself. By

removing all of them, the model simply could not generalize well and the accuracy suffered as a result. In general, the original network performs consistently better than the modified networks, but that makes sense given that the default network is fine-tuned to speed up the training process. This means that although deeper models presented above may perform better, they would require more time to train, and, thus, they performed worse in the tests above.

Additionally, we see that in replacing the last few layers of the network with many fully-connected layers back to back has a significant negative impact on the accuracy. Possibly if added even more fully-connected layers the accuracy may even drop further than the 5.3% we saw with only 5 additional fully connected layers. We also see that they drastically hurt performance in terms of loss as well with a significant difference. Again this was all expected to perform worse than the original as these trials were kind of a shotgun approach to see if it improved. We can also conclude that simply varying the size of the training data is not statistically significant and the results from the 10/5/85 split, albeit not very promising due to the very small training and validation size, still gives respectful results.

References:

1. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *arXiv preprint* *arXiv:1512.00567v3*, 2015.
2. National Data Science Bowl - Plankton Dataset. *https://www.kaggle.com/c/datasciencebowl*