

2 National Data Science Bowl Challenge

The second challenge we chose is the National Data Science Bowl 2015⁵. This challenge is about classifying plankton' images into the species the decrypted plankton belongs to. The evaluation metric for this challenge and thus the one used throughout this Section is the Multi-class Logarithmic Loss, Equation 9.

$$\text{LogLoss} = -\frac{\sum_{i=1}^N \sum_{j=1}^M y_{ij} \log(p_{ij})}{N} \quad (9)$$

where N is the number of products in the set, M is the number of plankton species, $y_{ij} = 1$ if observation i is in class j and 0 otherwise and p_{ij} is the probability that observation i came from class j .

2.1 Data

The dataset consists of approximately 30,000 greyscale images of planktonic species. The raw data was preprocessed by Oregon State University's Hatfield Marine Science Center to extract regions of interest, in order to have the images containing a single organism/entity. Due to the varying nature of the sizes of these organisms, the images vary in size, with image dimensions from as small as 21 pixels to as large as 428 pixels. Our task is to automate the image identification process of plankton⁶. As mentioned in the referenced link, it is possible for different classes to contain the same underlying organism, in which case the classifier needs to separate them based on other factors of interest (e.g. one may represent an organism in motion vs one that is still). There are 121 classes in total in this dataset.

2.2 Methods

The most commonly used models in computer vision are Convolutional Neural Networks. This is because they utilise convolutional filters that are able to extract position-invariant spatial hierarchies of features from the input image, in order to detect and classify the object in the image.

Another approach that has been very popular in recent years is transfer learning. Transfer learning is the idea of using knowledge acquired in one task, to solve other related ones. It

⁵<https://www.kaggle.com/c/datasciencebowl>

⁶<https://www.kaggle.com/c/datasciencebowl/data>

has become very popular due to it making the learning process faster, and requiring less data for the model to generalise well on the task we would like to apply the model to. In our context, transfer learning entails using pre-trained deep CNNs to solve this classification challenge. There are several models that have been pre-trained on the famous ImageNet dataset, a collection over 14 million images belonging to 1000 classes, that we looked into using for this task. In some cases, it is suitable to freeze the model's feature extraction layers, if the images are similar to those in ImageNet. However, given that plankton images were not included in Imagenet, we decided to allow our training to modify these parameters in the models.

We used a shuffled stratified split to divide the data into 90% training and 10% validation subsets. The validation set was used during training to find the best hyperparameters and prevent overfitting of our model by tracking the validation loss. The best combination of hyperparameters was the one that produced the lowest validation loss.

2.2.1 Preprocessing

Augmentation:

This dataset has a severe class imbalance problem, as shown in Figure 13.

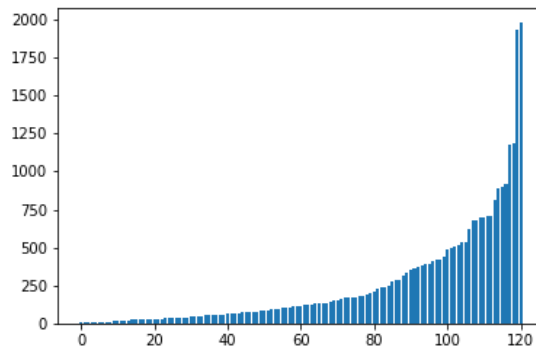


Figure 13: Class distribution of the plankton training data

The size of the classes in the training data range from as small as 9 to as large as 1,979. Given that deep learning models require lots of data to generalise well, we looked into data augmentation methods to artificially increase the size of the smaller classes.

One of the methods to do this was simply to create copies of the images belonging to the smaller classes in the dataset, and then augmenting them, for example using rotation, flips,

and Gaussian blurring. This image manipulation was done using the cv2 library, a popular computer vision library available in Python⁷. Credits for the specific implementation of data augmentation using this library go to Shijia Shu⁸.

An example of our image augmentation is shown in Figure 14.

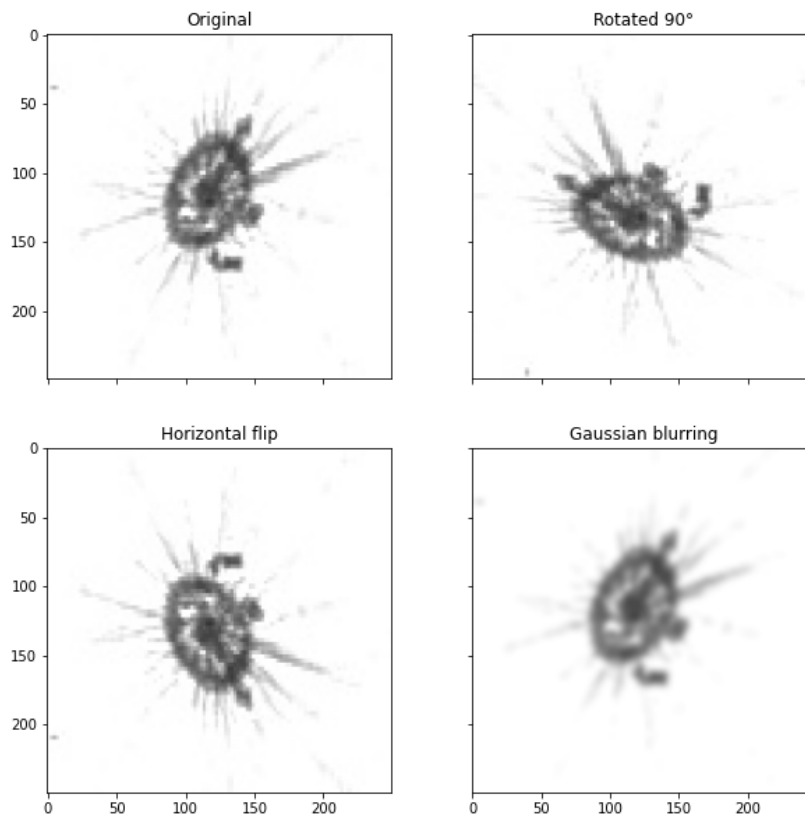


Figure 14: Example of augmentation

This data augmentation was only applied to classes whose number of original data samples was below a certain threshold. We experimented with different threshold values for different runs and models, for example 100, 250 etc.

⁷<https://opencv.org/>

⁸https://github.com/shu1995shijia/Plankton_Classification_228/blob/master/dataset/data_augmentation.ipynb

2.2.2 Data loading

Data loading was done utilising Pytorch’s Dataset and Dataloader classes. This allowed for automatic batching and iterating over the samples in the dataset, without having to load the entire dataset into memory at once⁹.

We also utilised Pytorch’s Transforms class. This allows for online or on-the-fly image transformations, such as cropping, rotation and flips, on the batch of images currently being executed, to help improve generalisation of the model further.

2.2.3 Pre-trained Models

ResNet:

One of the best performing computer vision models is the ResNet model [5] that makes it possible to train very deep models without vanishing gradients. It does this by introducing ‘skip connections’ between layers that allows some blocks of layers to not be used, if they do not contribute to improving performance. Therefore, adding layers can only improve the model’s performance.

DenseNet:

DenseNet is a variation of the ResNet model, that also utilises skipped connections. However, instead of connecting one layer to a previous layer, as in ResNet, the DenseNet connects all layers with all previous ones. So the input of each layer is the feature maps of all earlier layers, and its output is fed as input to all future layers [6]. Due to the dense connections, this model does not require as many parameters as the ResNet counterpart.

2.3 Experiments and Results

ResNet:

We tried variations of 4 ResNet models. These included a ResNet34 model, 2 ResNet50 models and 1 ResNet152 model. We attempted different layers and hidden units for the classifier portion of these models and employed early stopping at 15 epochs. The best validation losses achieved are presented in Table 16.

DenseNet:

⁹<https://pytorch.org/docs/stable/data.html>

We trained 7 DenseNet models in total, with different variations of image augmentation (both manual preprocessing and online transforms), classifier architecture, learning rates and dropout rates.

From our experiments, we noticed that using Pytorch transforms helped improve the validation loss from 0.81745 to 0.78126. We also noticed that increasing the complexity of the classifier, by adding a hidden layer, and by using regularisation with dropout helped lower the validation loss from 0.78126 to 0.76933 (Table 16).

We attempted to address the class imbalance further using a weighted loss and weighted random sampling. However, both these strategies were yielding much worse results compared to the runs without them, so they were abandoned.

In the end, most of the DenseNet models achieved a similar test loss, and no individual model achieved a test loss lower than 0.80.

Model	Train Loss	Valid Loss
Resnet34	0.48510	0.89116
Resnet50_1	0.51438	0.88061
Resnet50_2	0.47869	0.86121
Resnet152	0.54820	0.86385
Densenet169_1	0.29328	0.78126
Densenet169_2	0.51498	0.76933
Densenet169_3	0.65175	0.78698
Densenet169_4	0.49021	0.81051
Densenet169_5	0.54968	0.81745
Densenet169_6	0.32745*	0.31114*
Densenet169_7	0.33184*	0.33380*

Table 16: Model Losses *These runs were on a modified dataset that included augmented data. Therefore, the training and validation sets included more samples than the other runs, and so the losses should not be compared directly

2.3.1 Ensemble


Even though no individual model yielded a validation loss lower than the baseline, we combined the efforts of all models by ensembling. We took a weighted average of the predictions of the 8 models as shown in Table 17 to achieve a test result below baseline.

Model	Weight
Resnet34	0.02
Resnet50_1	0.02
Resnet50_2	0.05
Resnet152	0.05
Densenet169_1	0.10
Densenet169_2	0.10
Densenet169_3	0.10
Densenet169_4	0.08
Densenet169_5	0.08
Densenet169_6*	0.20
Densenet169_7*	0.20
Final Test Loss	0.62097

Table 17: Plankton Final Ensemble *These runs were on a modified dataset that included augmented data. Therefore, the training and validation sets included more samples than the other runs, and so the train and validation ensemble losses are not included

2.4 Conclusion

The ensemble of these pre-trained Convolutional Neural Networks with offline and online augmentation allowed us to beat the baseline score. Our submission is shown in Figure 15.



National Data Science Bowl

Predict ocean health, one plankton at a time
\$175,000 · 1,049 teams · 5 years ago

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [My Submissions](#) [Late Submission](#)

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
planktonbestsubmission.zip	a few seconds ago	0 seconds	25 seconds	0.61431

Complete

[Jump to your position on the leaderboard ▼](#)

You may select up to 2 submissions to be used to count towards your final leaderboard score. If 2 submissions are not selected, they will be automatically chosen based on your best submission scores on the public leaderboard. In the event that automatic selection is not suitable, manual selection instructions will be provided in the competition rules or by official forum announcement.

Your final score may not be based on the same exact subset of data as the public leaderboard, but rather a different private data subset of your full submission — your public score is only a rough indication of what your final score is.

You should thus choose submissions that will most likely be best overall, and not necessarily on the public subset.

```
>_ kaggle competitions submit -c datasciencebowl -f submission.csv -m "Message"
```

15 submissions for [MyrtoPapa](#) Sort by Most recent ▼

All Successful Selected

Submission and Description	Private Score	Public Score	Use for Final Score
planktonbestsubmission.zip a few seconds ago by MyrtoPapa add submission details	0.62097	0.61431	<input type="checkbox"/>

Figure 15: Plankton Submission