Phytoplankton classification

1 Introduction

The goal of this project was to build an image classification model on different species of Phytoplankton. Phytoplanktons are microorganisms that lives in the waterbody. We had an image set of 63000, comprised of 50 different species of Phytoplanktons that were sampled from Baltic seas.

2.1 Literature Review (Academic papers)

1. CNN trained on Scripps Plankton Camera images identifies 9 phytoplankton species, achieving 92% accuracy, aiding in situ monitoring of ocean health and Harmful Algal Blooms.(1)
2. Contextual data significantly boosts plankton image classification accuracy, with CNNs and methods like Random Forest demonstrating improved outcomes. (2)
3. A study used a CNN for phytoplankton classification in the Palmer Antarctica project, highlighting challenges in natural environments like class imbalance and species similarity. (3)

2.2 Literature Review (Models)

1. **Transfer Learning:** Shallow, pre-trained CNNs fine-tuned with phytoplankton images using transfer learning achieved a 0.95 F1-score, even with many unclassifiable images. (1)
2. **VGG-16 Model**: Applied in phytoplankton recognition, VGG-16 achieved a 94.3% accuracy and F1-score, notable for its depth and stacked 3x3 convolutional layers. (2) (3)
3. **InceptionV3 Model**: Achieved 95.20% accuracy in phytoplankton recognition, featuring Label Smoothing, Factorized 7x7 convolutions, and an auxiliary classifier. (4) (5)

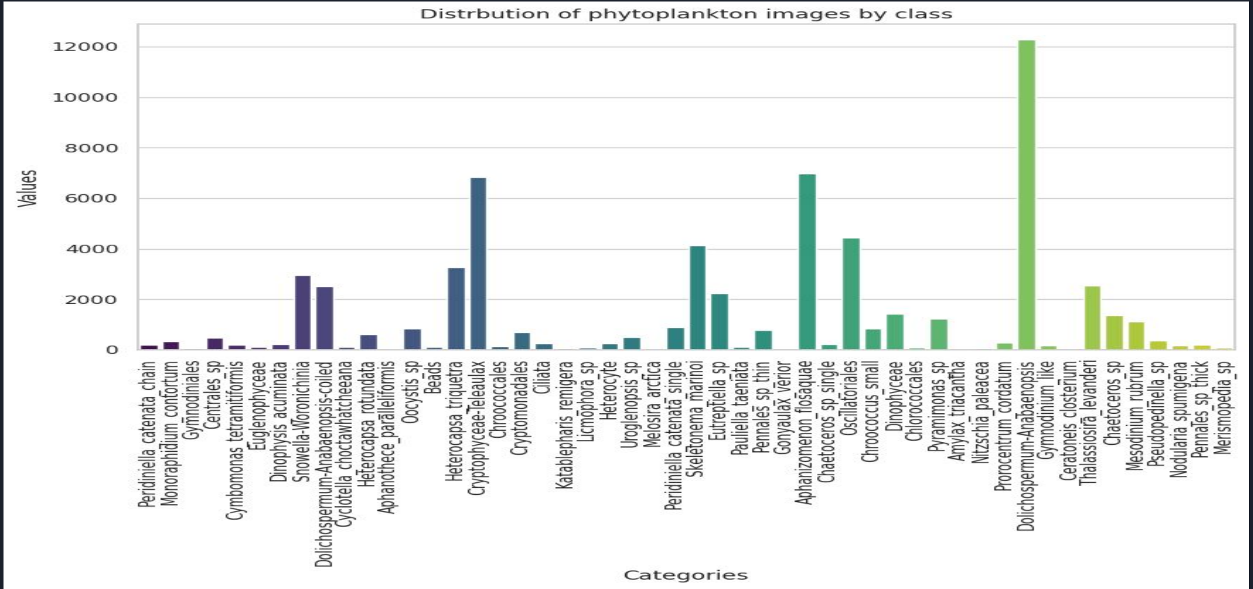
Aided by GPT-4. Complete conversation here.

3 Dataset Overview

- Images collected with an Imaging FlowCytobot (IFCB), a submersible in-situ automated flow cytometer. Particle size range is <10 to 150 µm
- Dataset contains >63,000 annotated images with 50 classes, with varying number of images in each class
- Already used for image classification with PyTorch
- The images were collected from Baltic ocean.
- The dimension of images are not uniform.

3.1 Dataset Overview

The class distribution of the Phytoplanktons in this dataset seems highly imbalanced which is one of the biggest challenges for building an image classification model. Also there is a location bias(since all the samples came from only baltic ocean).

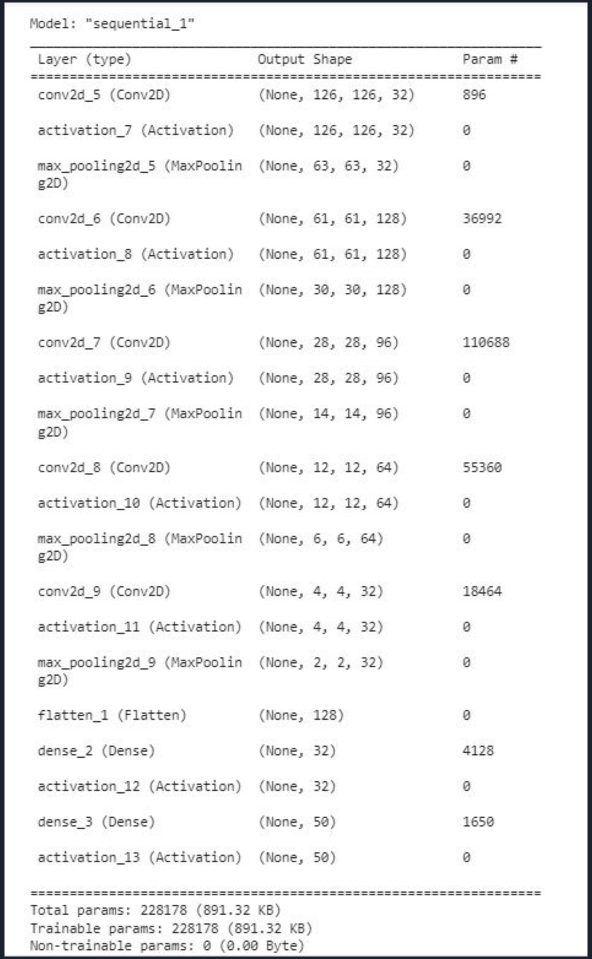


4 Baseline Model

Our baseline model was a self made CNN

with the summary provided in the picture.

It had a very bad accuracy of 9%.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 126, 126, 32)      896

 activation_7 (Activation)   (None, 126, 126, 32)      0

 max_pooling2d_5 (MaxPoolin  (None, 63, 63, 32)        0
 g2D)

 conv2d_6 (Conv2D)           (None, 61, 61, 128)       36992

 activation_8 (Activation)   (None, 61, 61, 128)       0

 max_pooling2d_6 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 conv2d_7 (Conv2D)           (None, 28, 28, 96)        110688

 activation_9 (Activation)   (None, 28, 28, 96)        0

 max_pooling2d_7 (MaxPoolin  (None, 14, 14, 96)        0
 g2D)

 conv2d_8 (Conv2D)           (None, 12, 12, 64)        55360

 activation_10 (Activation)  (None, 12, 12, 64)        0

 max_pooling2d_8 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_9 (Conv2D)           (None, 4, 4, 32)          18464

 activation_11 (Activation)  (None, 4, 4, 32)          0

 max_pooling2d_9 (MaxPoolin  (None, 2, 2, 32)          0
 g2D)

 flatten_1 (Flatten)         (None, 128)               0

 dense_2 (Dense)             (None, 32)                4128

 activation_12 (Activation)  (None, 32)                0

 dense_3 (Dense)             (None, 50)                1650

 activation_13 (Activation)  (None, 50)                0

=================================================================
Total params: 228178 (891.32 KB)
Trainable params: 228178 (891.32 KB)
Non-trainable params: 0 (0.00 Byte)
```

5. Model definitions

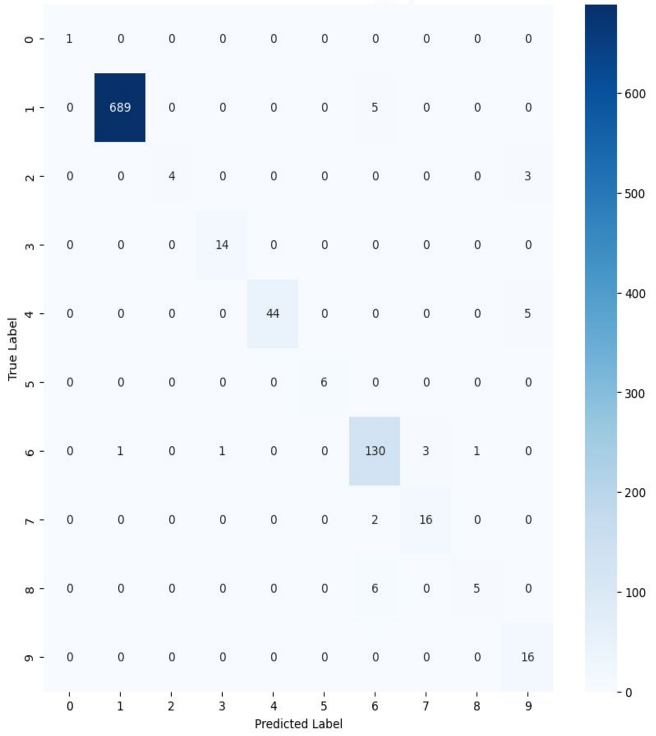
1. Self-trained model with 4 layers of Conv2D and Pooling (Baseline Model)
2. Transfer-learning using the MobileNetV2 trained on the Imagenet Dataset (Freezing layers, adding custom output layer)
3. Split the dataset into 5 groups of 10 classes and stuck to the MNV2 model adjusting the Output layer to softmax and 10 classes
    a. Trained 5 models for 10 classes each with a test accuracy of ~95%
4. Trained “parent model” to classify image into one of 5 groups

5.1 Model definitions
Tricks and methods

- Splitting the 50 classes into 5 groups of 10 classes
- Verifying that on average 10 classes are a good enough measure to predict accurately
- Undersampling of overrepresented classes to ~200 samples per class
- Writing a custom ImageDataGenerator to especially augment classes with underrepresented sample numbers to ~200 samples per class
- Use Transfer-Learning based off of the MobileNetV2 trained on the imagenet dataset. Define custom output layer and train.

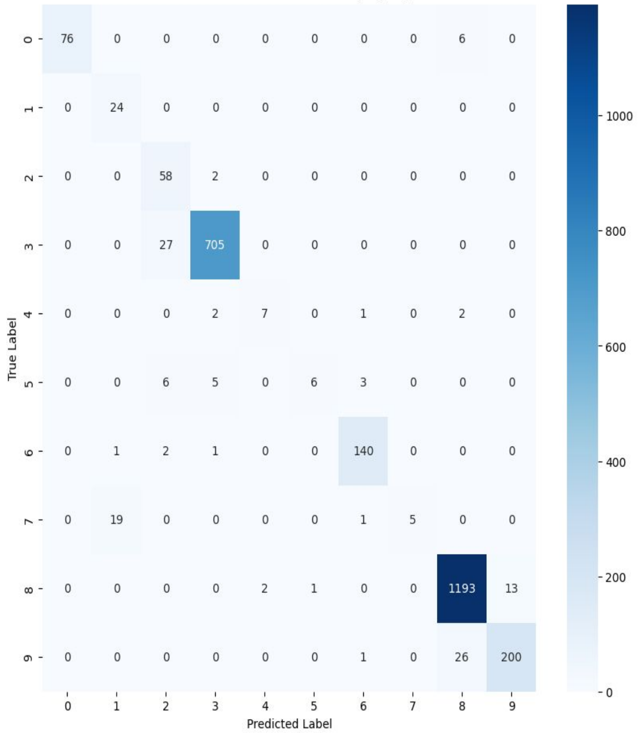
5.2 Model evaluations

Confusion Matrix for classes (0, 9)



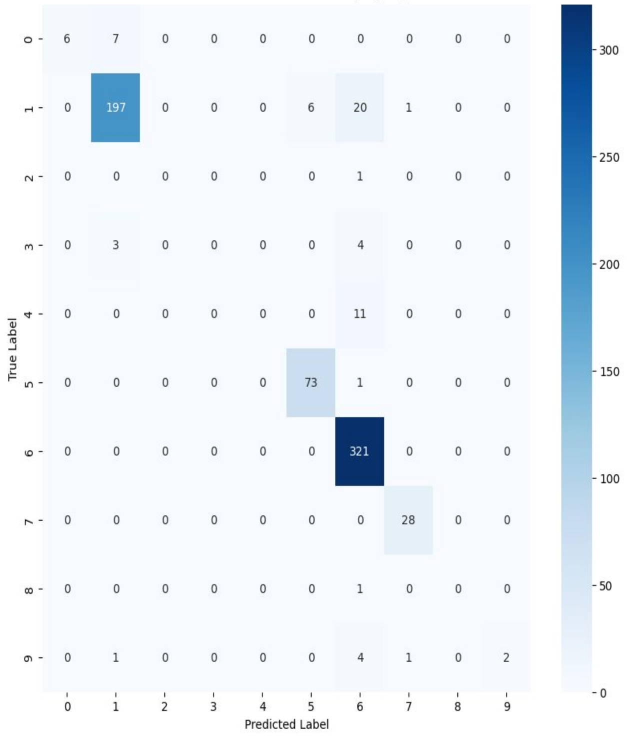
True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	1	0	0	0	0	0	0	0	0	0
1	0	689	0	0	0	0	5	0	0	0
2	0	0	4	0	0	0	0	0	0	3
3	0	0	0	14	0	0	0	0	0	0
4	0	0	0	0	44	0	0	0	0	5
5	0	0	0	0	0	6	0	0	0	0
6	0	1	0	1	0	0	130	3	1	0
7	0	0	0	0	0	0	2	16	0	0
8	0	0	0	0	0	0	6	0	5	0
9	0	0	0	0	0	0	0	0	0	16

Confusion Matrix for classes (10, 19)



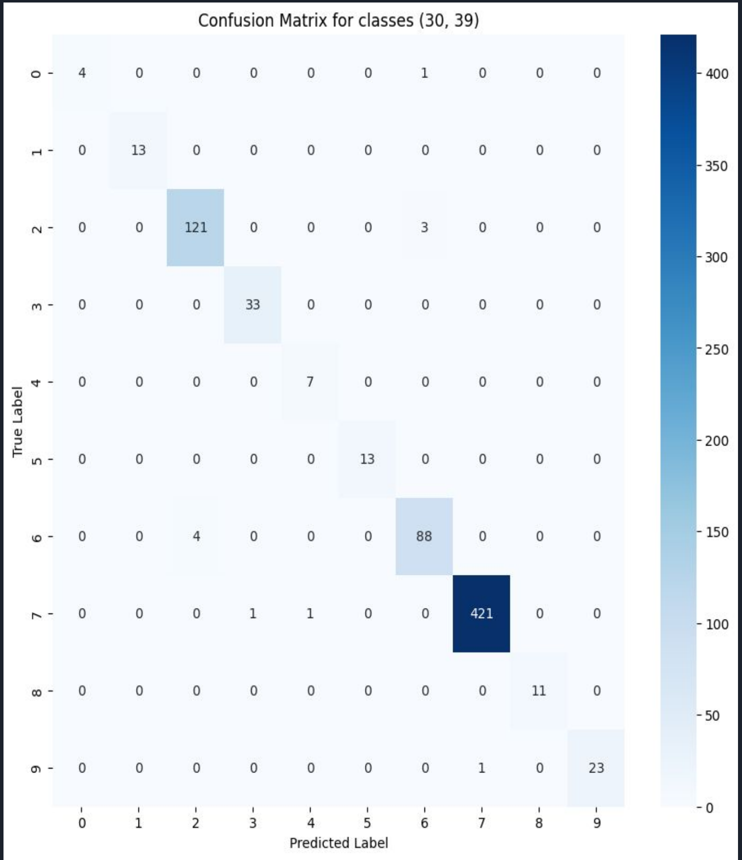
True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	76	0	0	0	0	0	0	0	6	0
1	0	24	0	0	0	0	0	0	0	0
2	0	0	58	2	0	0	0	0	0	0
3	0	0	27	705	0	0	0	0	0	0
4	0	0	0	2	7	0	1	0	2	0
5	0	0	6	5	0	6	3	0	0	0
6	0	1	2	1	0	0	140	0	0	0
7	0	19	0	0	0	0	1	5	0	0
8	0	0	0	0	2	1	0	0	1193	13
9	0	0	0	0	0	0	1	0	26	200

Confusion Matrix for classes (20, 29)

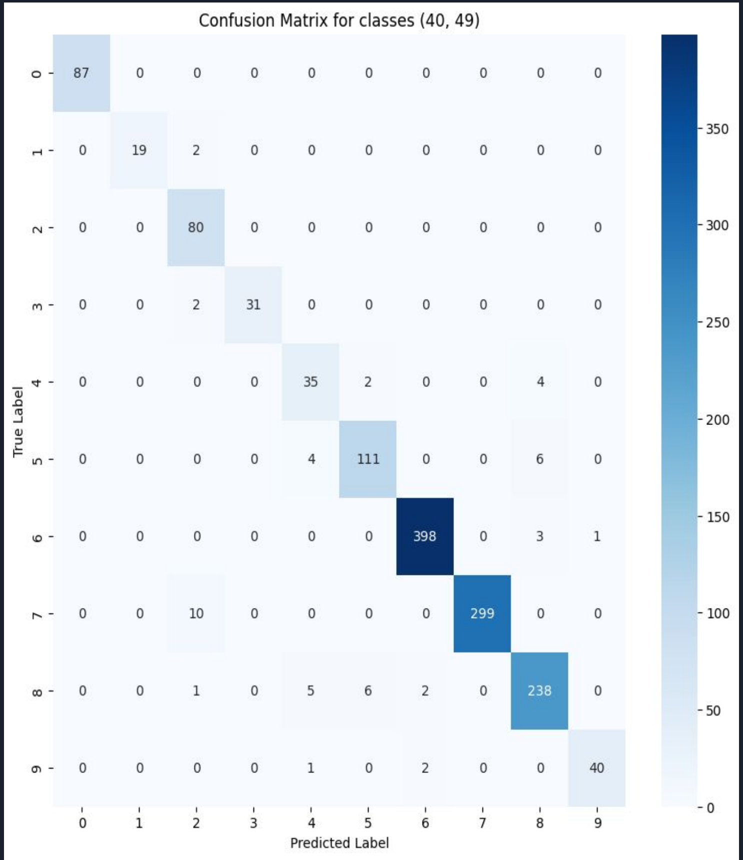


True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	6	7	0	0	0	0	0	0	0	0
1	0	197	0	0	0	6	20	1	0	0
2	0	0	0	0	0	0	1	0	0	0
3	0	3	0	0	0	0	4	0	0	0
4	0	0	0	0	0	0	11	0	0	0
5	0	0	0	0	0	73	1	0	0	0
6	0	0	0	0	0	0	321	0	0	0
7	0	0	0	0	0	0	0	28	0	0
8	0	0	0	0	0	0	1	0	0	0
9	0	1	0	0	0	0	4	1	0	2

5.3 Model evaluations



True Label \ Predicted Label	0	1	2	3	4	5	6	7	8	9
0	4	0	0	0	0	0	1	0	0	0
1	0	13	0	0	0	0	0	0	0	0
2	0	0	121	0	0	0	3	0	0	0
3	0	0	0	33	0	0	0	0	0	0
4	0	0	0	0	7	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0
6	0	0	4	0	0	0	88	0	0	0
7	0	0	0	1	1	0	0	421	0	0
8	0	0	0	0	0	0	0	0	11	0
9	0	0	0	0	0	0	0	1	0	23



True Label \ Predicted Label	0	1	2	3	4	5	6	7	8	9
0	87	0	0	0	0	0	0	0	0	0
1	0	19	2	0	0	0	0	0	0	0
2	0	0	80	0	0	0	0	0	0	0
3	0	0	2	31	0	0	0	0	0	0
4	0	0	0	0	35	2	0	0	4	0
5	0	0	0	0	4	111	0	0	6	0
6	0	0	0	0	0	0	398	0	3	1
7	0	0	10	0	0	0	0	299	0	0
8	0	0	1	0	5	6	2	0	238	0
9	0	0	0	0	1	0	2	0	0	40

6.1 Model results: Individual models

Class range	Macro Average F1 Score	Weighted Average F1 Score
0 - 9	91%	98%
10 - 19	77%	95%
20 - 29	48%	89%
30 - 39	97%	99%
40 - 49	94%	96%
Parent model	85%	86%

6.2 Model results:
Complete model accuracy

First predicting the group and then calling the corresponding child model is not working well. We have reached an accuracy score on the test set of ~10,80% which is much worse than the individual models.

7 Challenges and Errors

- Initially we had some setup issues, where to store the dataset, how to load it in etc.
- High number of samples makes it hard to train a model on all the images, as the training is taking ages.
- Dataset imbalance was giving us some headache - over /undersampling was solving aspects of it.
- High similarities in some classes
- The inaccuracies of the parent model completely water down the high accuracy scores of the child models. The “errors” of the parent model get passed onto the child models.

8 Discussion & Conclusion & Prospective Work

- A hierarchical approach is difficult, as the errors in the first layers are passed on to the later layers.
- The child models can be very potent on each of the subsets of the dataset. On that front we have achieved quite a good F1 Score

- As the child models are working very very well, we can build a parent model, that only differentiates between 2 of the child models.
- Find improvements on the global parent model

9 Q & A

Questions, Suggestions, Ideas?