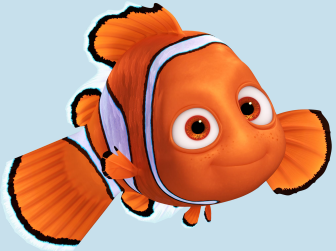
MSE 546 - ML Project

Finding Nemo



Thomas, Elize, Matt, Ria, Mateo

TABLE OF CONTENTS

TASK 01

DATA 02

METRICS 03

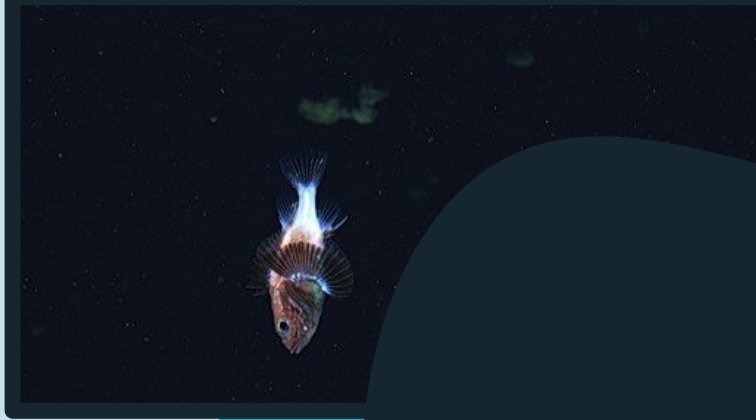
MODELS 04

RESULTS 05

KEY TAKEAWAYS 06

GOAL & DATA

- **Objective:** Multi-label classification, identify multiple marine organisms in an image
- **Input Data:** raw image data (*5950 images*), annotations for 290 fine-grained categories, mapping of fine grained categories to one of 20 super-categories, and species metadata
- **What are the Images of?:** Pictures of marine organisms from the upper ocean <800m.
- **Output:** A vector of 290 values, each representing the probability of a corresponding species being present
- **Evaluation Metric:** MAP@20, which measures how well the top 20 predicted species match the actual species present
- **Challenges:** Class imbalances, computationally heavy, low resolution images
- **Required compute:** Ideally GPUs, with adequate storage to support the difficulty of the task



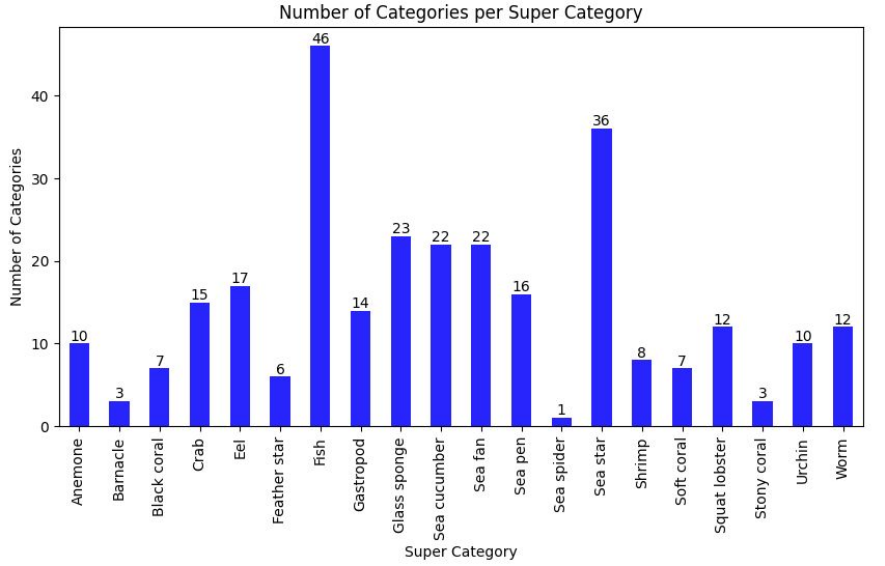
5950 Images

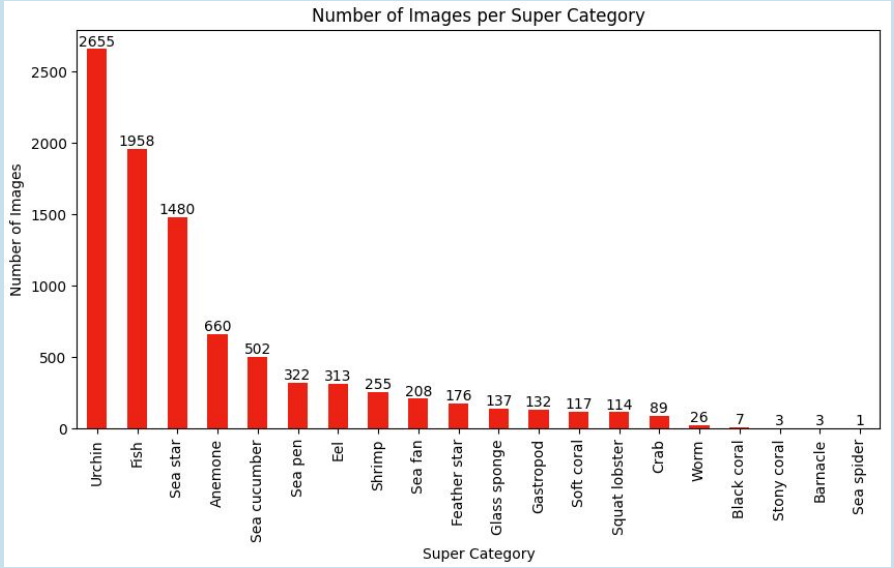
Including bounding boxes, and species classification

290 Species

20 Super-categories

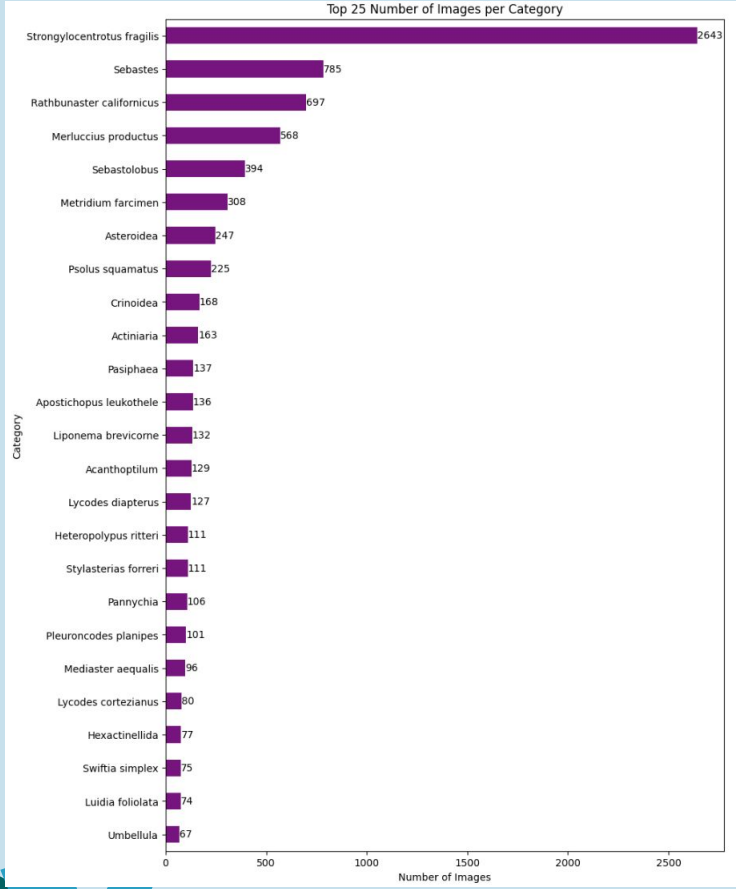
Super-Category EDA





This analysis presents the number of categories for each super category. The distribution is noticeably unbalanced. For example, the super category "Sea Spider" has only one animal category, whereas the super category "Fish" contains 46 distinct categories.

The most prevalent super categories in the image dataset are "Urchin", "Fish", and "Sea Star". This chart reveals that the dataset is quite imbalanced, as these top three super categories appear in significantly more images than the less common ones. For instance, only one image contains "Sea Spider".



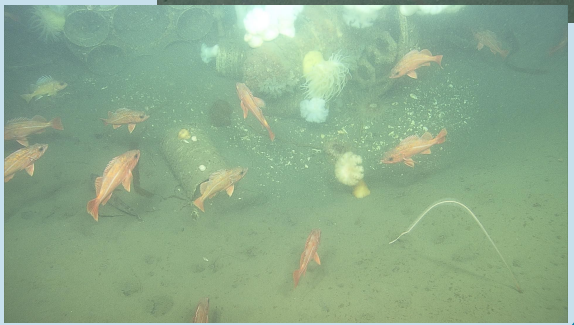
Data Points Per Category

Given that there are 290 categories in total, we focus on the top 25 most frequent categories. The analysis reveals that the dataset is highly imbalanced. In the training images, the animal category "Strongylocentrotus fragilis" appears in an impressive 2643 images, while the number of images per category drops steeply. For instance, "Umbellula," which ranks 25th, appears in only 67 images.

METRICS

Mean Average Precision at 20 (MAP@20 Score)

- Evaluates the model's ability to rank the correct categories within the top 20 predictions
- A higher MAP@20 score means the model is correctly ranking species in the top 20 per image
- MAP@20 was chosen as the evaluation metric because precision alone doesn't consider ranking, only the fraction of correct predictions. MAP@20 is also better suited for imbalanced datasets, like this one.
- Partially handles our dataset's imbalanced number of images per category
    - Helps asses if species are ranked well, rather than being overshadowed by common ones



1. Baseline: Logistic Regression

Overview

The concatenated RGB tensor was passed into a linear layer from these values to 290 classes. Binary-cross entropy with logits loss in PyTorch was used to convert raw output to probabilities with a sigmoid function, before then computing the binary cross-entropy loss.

Hyperparameters

- batch_size = 32
- channels, height, width = 3, 224, 224
- lr = 1e-3
- num_epochs = 3
- threshold = 0.5

Limitations

Ignores Spatial Structure – Unlike CNNs, logistic regression, built off linear relationships to all pixel values, does not capture spatial relationships

Strengths

Very straightforward to implement, easily interpretable, and low computational power requirements.

2. Classical Neural Network

Overview

The vectorized input was passed into a linear layer. Additional ReLU layers were added and connected, while the hidden layer size was reduced for optimization. MSE Loss in PyTorch was applied to measure prediction errors.

Hyperparameters

- batch_size = 32
- channels, height, width = 3, 224, 224
- lr = 1e-3
- num_epochs = 3
- threshold = 0.5

Limitations

Struggles with detecting objects with complex spatial relationships, and may not generalize the best in our context.

Strengths

Suits our task as it efficiently captures spatial hierarchies and local patterns, recognizing features like edges, textures, and shapes

3. ResNet18 (Residual CNN)

Overview

ResNet is a deep CNN that enhances feature learning through residual (skip) connections, allowing layers to learn modifications on top of their inputs. This enables training of much deeper networks, yielding robust feature representations ideal for complex tasks such as multi-label classification.

Hyper Parameters

- batch_size = 32
- lr = 1e-3
- num_epochs = 15

Limitations

ResNet models are more computationally and memory intensive than classical neural networks, but also can be less parameter-efficient compared to newer architectures.

Strengths

ResNet uses residual connections that ease training of very deep networks, leading to robust feature extraction and high accuracy. It is less computational expensive then larger models and can be better suited for less complex issues.

4. EfficientNetB0

Overview

EfficientNet is a deep CNN that improves accuracy while using fewer parameters by scaling depth, width, and resolution efficiently. It uses MBConv layers and SE blocks to enhance feature extraction while keeping computations low. This makes it ideal for real-world AI applications requiring high performance with lower resource costs.

Limitations

Predefined scaling coefficients may not generalize well across all datasets and hardware. Its complex architecture makes interpretability challenging, and it requires large-scale pre-training for best performance.

Hyper Parameters

- EfficientNet B0 Architecture
- Pre-trained model weights
- batch_size = 32
- lr = 1e-3
- num_epochs = 80
- threshold = 0.5

Strengths

EfficientNet outperforms larger models while using fewer resources, enabling faster training and inference. It generalizes well across various computer vision tasks, making it suitable for enterprises needing scalable AI solutions.

5. DenseNet121 (Densely Connected CNN)

OVERVIEW

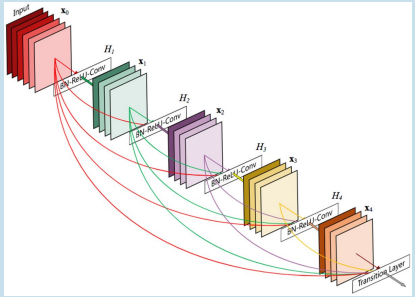
DenseNet is a deep CNN that improves feature learning by densely connecting layers, meaning all layers can access all previous layers. It's efficient feature reuse strengthens representation learning which is useful for multi-label classification tasks.

Hyper Parameters

- Batch Size = 32
- Learning Rate = 1e-3
- Epochs = 10

LIMITATIONS

DenseNet has high computational and memory requirements due its dense connectivity. DenseNet is also sensitive to hyperparameter tuning, which can cause difficulties when balancing performance and efficiency.



STRENGTHS

DenseNet reuses features effectively, reducing parameter redundancy while improving performance. It is also less prone to overfitting due to strong feature propagation.

6. ResNet18 with Cutout Regularization

Overview

In many image classification tasks with large neural networks, overfitting can occur. Cutout regularization is a commonly used image augmentation technique which removes square(s) from the images at random.

Hyperparameters

- # of cutouts/image = 1
- Size of cutout = 16 x 16
- batch_size = 32
- lr = 1e-3
- num_epochs = 3

Limitations

Due to our computational resource constraints, and complex dataset, it is unlikely that overfitting is occurring in training. Therefore, this regularization may cause further underfitting and accuracy issues.

Strengths

Helps us assess if model benefits from regularization. Encourages the model to learn more robust and generalized features instead of relying on specific image regions.

RESULTS

	Initial logistic regression	Basic NN	ResNet18	EfficentNet B0	DenseNet121	ResNet18 w/ Cutout
Epochs	3	3	15	80	10	12
MAP@20	0.4767	0.5203	0.7074	0.2483	0.7532	0.6877

IMPROVEMENTS OVER BASELINE

BASELINE TO NEURAL NETWORK

Transitioning from simple logistics regression to a classical neural network showed improvements (MAP@20 from 0.4767 -> 0.5203). This helped but fully flattening the image data limits the model's understanding of spatial patterns limiting its performance.

INTRODUCTION OF CNNs

ResNet improved the performance significantly, increasing the MAP@20 score to 0.7. This showed that retaining spatial info. In the convolutional layers was critical to capture meaningful features.

DEEP CNNs

The dense connectivity of DenseNet and its efficient feature reuse outperformed the baseline significantly, achieving a MAP@20 score of 0.75. It's architecture allowed for efficient learning despite challenges with class imbalances and low resolution images.

BASELINE MAP@20

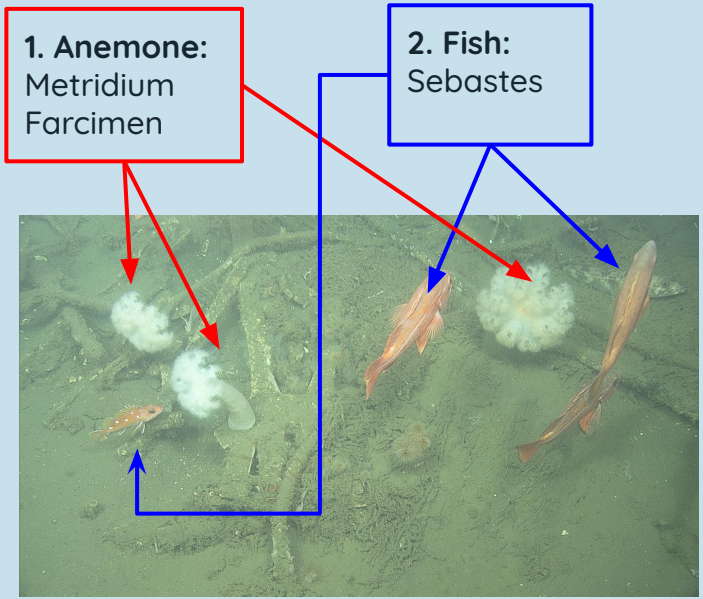
0.4767

DENSENET MAP@20

0.7532

Classification Output Example

Ground-Truth Categories:



Logistic Regression Baseline:

1. **Urchin:** Strongylocentrotus fragilis
   Probability: 0.4127
2. **Fish:** Sebastes
   Probability: 0.0441
3. **Sea star:** Rathbunaster californicus
   Probability: 0.0433
4. **Fish:** Merluccius productus
   Probability: 0.0165
5. **Fish:** Sebastolobus
   Probability: 0.0162

DenseNet:

1. **Anemone:** Metridium farcimen
   Probability: 0.3318
2. **Fish:** Sebastes
   Probability: 0.2149
3. **Sea cucumber:** Pannychia
   Probability: 0.1445
4. **Eel:** Eptatretus
   Probability: 0.0526
5. **Crab:** Chionoecetes Tanneri
   Probability: 0.0404

PROPOSED SOLUTION

Best model

DenseNet produced the best output on the test data with a MAP@20 score of 0.752 after 10 epochs.

IMPLEMENTATION DETAILS

- Pretrained DenseNet121 was used to improve the starting point of the model and leverage transfer learning
- The final classification layer was adjusted to 290 output classes (species labels)
- The model was trained for 10 epochs with BCE and Logits Loss, optimizing with Adam
- Using a batch size of 32, and a learning rate of 1e-3

RESULTS

0.752	10	0.0078
MAP@20	Epochs	Training Loss

IMPROVEMENT 1

Longer training with a learning rate decay could allow the model to fine-tune

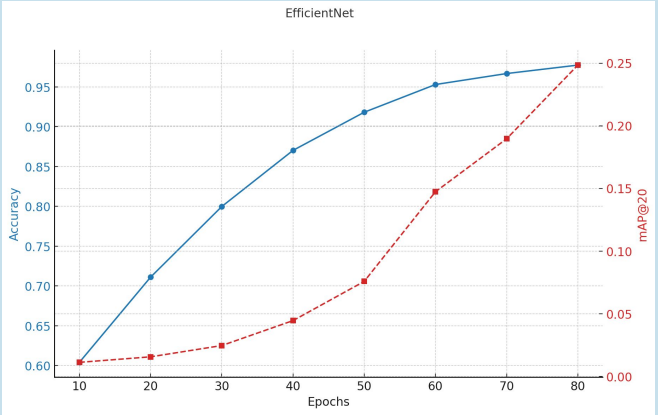
IMPROVEMENT 2

Using weighted loss to help handle the class imbalance, this would help improve MAP@20 for rate species.

INSIGHTS

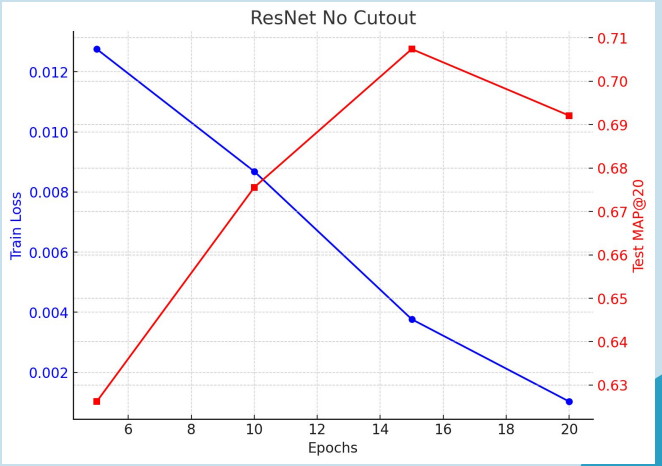
INTERPRETATION & TRENDS

Visualizing training loss and MAP@20 across epochs highlighted how models learn at different rates. Both ResNet12 & EfficientNet showed strong initial learning but required more tuning to have further increases. It is also observed that EfficientNet has a high accuracy despite its low MAP@20 score. DenseNet on the other hand consistently improved due to feature reuse



CUTOUT REGULARIZATION

Adding cutout regularization to ResNet slightly decreased the models performance. Typically deep models are prone to overfitting, making regularization helpful. In this case, due to the resource constraints, and complex images, the model did not exhibit typical overfitting issues. This resulted in cutout regularization leading to underfitting rather than improvement.



KEY TAKEAWAYS
for Future Iterations

LARGER MODELS

Would be able to capture more complex features using models with more layers (e.g. ResNet18 vs ResNet 50)

DATA AUGMENTATION

To compensate for the minority classes, we would leverage synthetic data in the training data set to enhance the model

COMPUTE POWER

Would reduce the time needed for model convergence and would definitely allow us to run more epochs to support deeper training

TRANSFORMER

Vision transformers can learn spatial hierarchies directly from data and scales well with large datasets. ViTs process entire images at once, allowing them to understand more global relationships

REFERENCES

- *FathomNet 2023*. Kaggle. (n.d.). https://www.kaggle.com/competitions/fathomnet-out-of-sample-detection/discussion?sort=hotness
- *Densenet*. PyTorch. (n.d.). https://pytorch.org/hub/pytorch_vision_densenet/

GITHUB

- *https://github.com/thomask902/MSE_546_Project*