# Image Classification of Marine Animals using CNN

Student: Jingjing Yang 2104210

Option: number 2

Title of the work: Image Classification of Marine Animals

Used algorithm: Convolutional Neural Networks (CNNs)

# Application Overview

The application of Convolutional Neural Networks (CNNs) in this project is aimed at image classification tasks within the marine biology field. The main objective is to develop a model that can identify and differentiate between images of different sea creatures, specifically 'Dolphin', 'Eel', 'Penguin', 'Seal', and 'Sharks'.

This will be achieved by using TensorFlow and Keras to create training and testing datasets, train the CNN model, evaluate it, and ultimately achieve high classification accuracy.

# Dataset

## Data Description

The dataset, sourced from Kaggle, comprises a collection of images representing 23 different classes of marine animals, with a total of 12,608 images. The classes include 'Clams', 'Corals', 'Crabs', 'Dolphin', 'Eel', 'Fish', 'Jelly Fish', 'Lobster', 'Nudibranchs', 'Octopus', 'Otter', 'Penguin', 'Puffers', 'Sea Rays', 'Sea Urchins', 'Seahorse', 'Seal', 'Sharks', 'Shrimp', 'Squid', 'Starfish', 'Turtle_Tortoise', and 'Whale'. The images are resized to either (300px, n) or (n,300px), where n is a pixel size less than 300px.

## Data Preparation

To simplify training, I selected five classes: 'Dolphin', 'Eel', 'Penguin', 'Seal', 'Sharks'. I then used a script to divide these classes' images into training and testing sets, with an 80:20 split ratio.

The original dataset is not balanced. To handle this, I've chosen to limit the number of images in classes 'Dolphin' and 'Sharks' to a maximum of 500.

Labels: ['Dolphin', 'Eel', 'Penguin', 'Seal', 'Sharks']
Number of images in each training class:
Dolphin : 400
Eel : 397
Penguin : 385
Seal : 331
Sharks : 400

## Examples of Training Data


## Data Preprocessing and Augmentation

- The data consists of images stored in separate training and testing directories.
- Image augmentation techniques like rotation, width/height shift, shear, zoom, and horizontal flip are applied to the training data.
- Both training and test datasets are rescaled.

# Model Architecture

- The model is a Convolutional Neural Network (CNN) comprising of Conv2D, MaxPooling2D, Dropout, Flatten, Dense, and BatchNormalization layers:

  The CNN model consists of 3 convolutional layers, each followed by a batch normalization layer, an activation layer (ReLU), a max pooling layer, and a dropout layer. After the convolutional layers, there is a flatten layer, followed by a dense layer, another batch normalization layer, another activation layer (ReLU), another dropout layer, and finally a dense layer with a softmax activation function.
- Activation function used is 'relu', and the final layer has a 'softmax' activation for multi-class classification.
- The loss function is Sparse Categorical Crossentropy, and the optimizer is Adam.
- The model includes callbacks for reducing learning rate on plateau and early stopping to prevent overfitting.
- Training is conducted for up to 50 epochs.

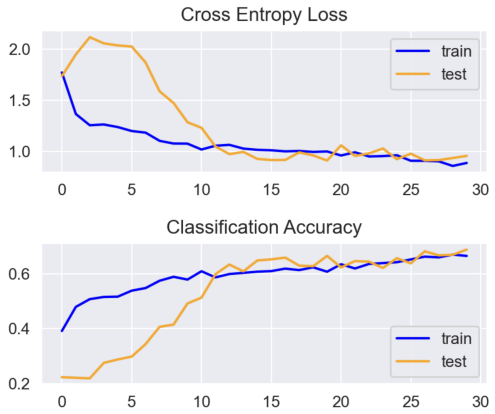
# Training of the Neural Network

## curves

The training of the neural network is visualized with curves. The accuracy and loss of both the training and validation sets are plotted over the number of epochs.

### metrics

- **Accuracy**: The ratio of correct predictions to the total number of predictions made. It measures how many predictions the model got right, regardless of the predicted class.
- **Loss**: The degree of error in predictions. A lower loss indicates that the model is performing better.
- **Classification Report**: A report that includes precision, recall, F1-score, and support for each class. This helps to understand the performance of the model on each individual class.




```
> Accuracy on test data: 66.46
> Loss on test data: 0.91
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Dolphin | 0.23 | 0.27 | 0.25 | 100 |
| Eel | 0.21 | 0.21 | 0.21 | 100 |
| Penguin | 0.12 | 0.08 | 0.10 | 97 |
| Seal | 0.14 | 0.13 | 0.14 | 83 |
| Sharks | 0.20 | 0.23 | 0.21 | 100 |
| accuracy | | | 0.19 | 480 |
| macro avg | 0.18 | 0.19 | 0.18 | 480 |
| weighted avg | 0.18 | 0.19 | 0.18 | 480 |

# Conclusions

## Model Performance and Limitations

The model's training accuracy gradually increases to around 66.46%. However, validation accuracy fluctuates and is generally lower, indicating some overfitting. The loss decreases over time for both training and validation data, yet validation loss remains higher, further indicating overfitting. Precision, recall, and F1-scores are relatively low (approximately 0.18 to 0.25), suggesting the model struggles with accurate, consistent image classification.

```
Predicting image: images_to_predict/eel_1.jpg
1/1 [==============================] - 0s 256ms/step
Image: eel_1.jpg, Predicted Class: Eel, Confidence: 39.98%
Predicting image: images_to_predict/shark_1.jpeg
1/1 [==============================] - 0s 62ms/step
Image: shark_1.jpeg, Predicted Class: Dolphin, Confidence: 24.13%
Predicting image: images_to_predict/seal_1.jpeg
1/1 [==============================] - 0s 62ms/step
Image: seal_1.jpeg, Predicted Class: Seal, Confidence: 38.70%
Predicting image: images_to_predict/penguin_1.jpeg
1/1 [==============================] - 0s 63ms/step
Image: penguin_1.jpeg, Predicted Class: Dolphin, Confidence: 22.47%
Predicting image: images_to_predict/dolphin_1.jpeg
1/1 [==============================] - 0s 62ms/step
Image: dolphin_1.jpeg, Predicted Class: Dolphin, Confidence: 38.40%
```

## Use Cases and Applicability

For critical applications requiring high precision and recall, the model may not be suitable. However, it could still be useful for less critical applications or as a preliminary classification tool.

## Potential Improvements

Improvements could be made by addressing overfitting through data augmentation or regularization techniques. Hyperparameter tuning and adjustments to the model architecture could potentially enhance performance. Addressing class imbalance, if present, could also lead to improvements.

## Exploring Advanced Methods

Considering more advanced models or pre-trained networks via transfer learning could be beneficial, especially if the dataset is not very large. Fine-tuning early stopping and learning rate adjustment parameters, along with implementing k-fold cross-validation, might yield better results.

## Summary and Next Steps

In summary, while the model shows some ability to classify images, it's not yet optimal. Further refinements are necessary for higher accuracy and reliability.