An explainable Convolutional Neural Network approach to fossil shark tooth identification

Andrea BARUCCI, Giulia CIACCI, Pietro LIÒ, Tiago AZEVEDO, Andrea DI CENCIO, Marco MERELLA,

Giovanni BIANUCCI, Giulia BOSIO*, Simone CASATI and Alberto COLLARETA

Supplementary Online Material
BSPI 63 (3) - 2024

Dataset

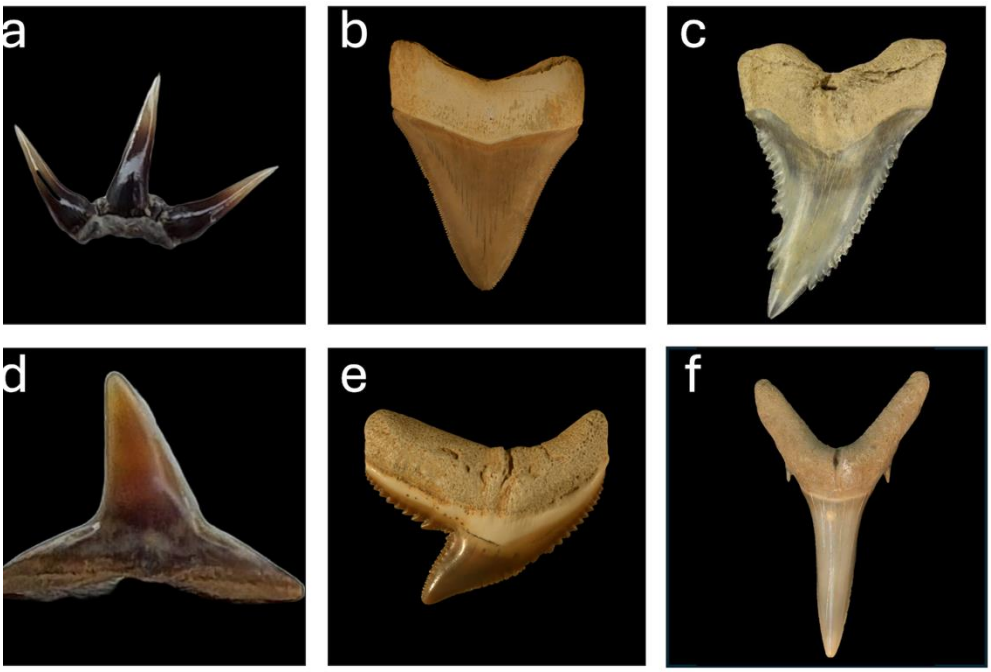


Fig. S1 - Examples of images for the training/validation/test set: a) *Chlamydoselachus* b) *Carcharocles*; c) *Hemipristis*; d) *Squatina*; e) *Galeocerdo*; f) *Carcharias.*

The images of teeth we showed in Figure 2 of the main manuscript are all of the same dimension. We did this just for the aesthetic of the figure, but actually the dataset accommodates images with different tooth sizes inside, many of them with the tooth fitting the dimension of the figure. For clarity, we report here in Fig. S1 and Fig. S2 two panels with some examples.

Fig. S2 - Examples of images for the training/validation/test set in different rotation positions.

In Fig. S3 a histogram of the data distribution is reported showing the number of images for each class.

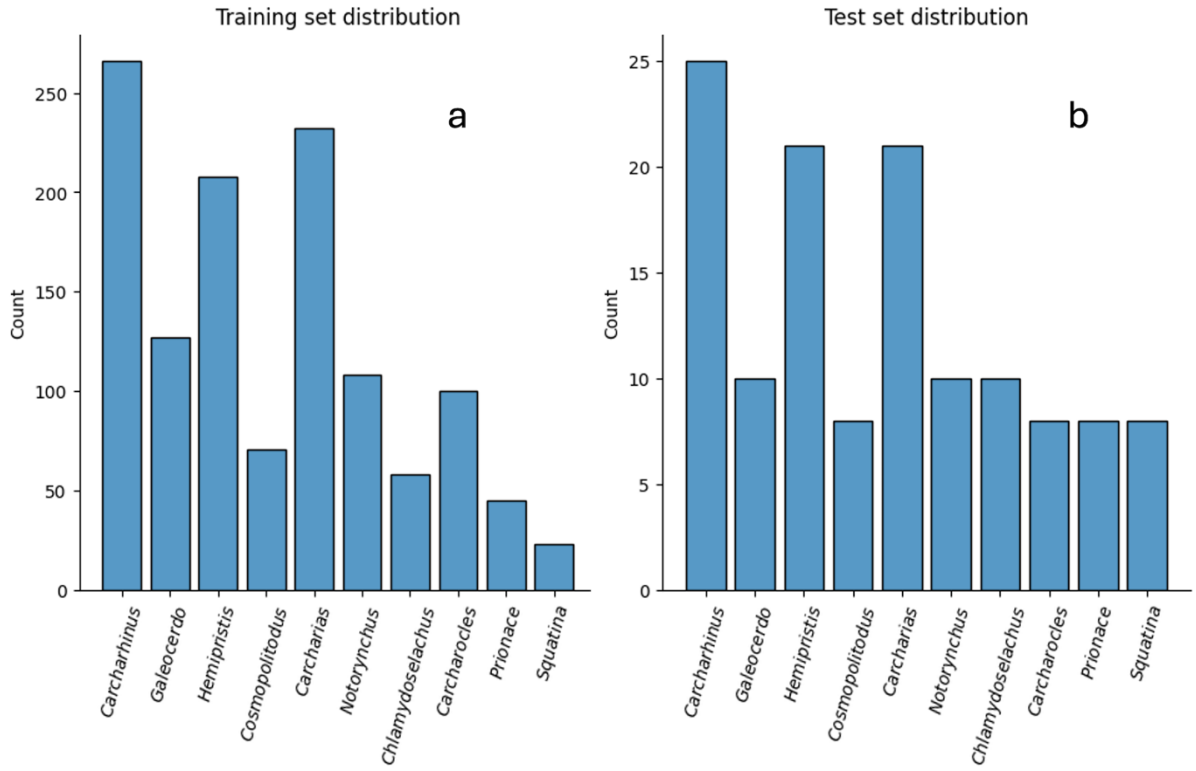


Fig. S3 - Dataset images distribution between the different classes. a) training set; b) the test set.

Classification metrics

Here we report some definitions and comments regarding classifications metrics.

Accuracy measures the proportion of correctly predicted instances out of the total number of instances in a dataset, while balanced accuracy adjusts the score to account for imbalanced datasets, where the number of instances in different classes is not evenly distributed. In order to do so, it calculates accuracy for each class individually and then averages these accuracies across all classes, providing a more reliable measure of performance on imbalanced datasets. In Neural Network training, validation, and testing, balanced accuracy helps ensure that the model performs well across all classes, not just the most populated class.

Using the following definitions:

- **True Positives (TP)**: The number of correctly predicted positive instances.
- **True Negatives (TN)**: The number of correctly predicted negative instances.
- **False Positives (FP)**: The number of incorrectly predicted positive instances.
- **False Negatives (FN)**: The number of incorrectly predicted negative instances.

we can define:

Accuracy = [True Positives (TP) + True Negatives (TN)]/Total Predictions

Balanced Accuracy = ½[TP/(TP+FN) + TN/(TN+FP)]

While accuracy is useful, it can be misleading in cases where the class distribution is imbalanced (e.g., a dataset with 95% negatives and 5% positives).

Precision (also known as Positive Predictive Value) measures the proportion of positive predictions that are actually correct. It is a key metric when the cost of false positives is high. High precision indicates that the model has a low rate of false positives.

Precision = True Positives (TP)/[True Positives (TP) + False Positives (FP)]

Recall (also known as Sensitivity or True Positive Rate) measures the proportion of actual positives that are correctly identified by the model. It is crucial when the cost of false negatives is high. High recall indicates that the model has a low rate of false negatives.

Recall = True Positives (TP)/[True Positives (TP) + False Negatives (FN)]

To illustrate these concepts in the context of fossil image classification, here we report an example: Accuracy would tell you the overall percentage of correctly classified fossil and non-fossil images out of all the images. Precision would tell you the percentage of images classified as fossils that are actually fossils. This is important if you want to minimize the number of non-fossil images mistakenly classified as fossils. Recall would tell you the percentage of actual fossil images that are correctly identified as fossils. This is important if you want to ensure that most, if not all, fossil images are correctly detected.

F1-score: Balancing Precision and Recall

Often, there is a trade-off between precision and recall. Improving one can sometimes decrease the other. The F1 score, which is the harmonic mean of precision and recall, is commonly used to balance the two:

F1 Score = 2 x [Precision x Recall] / [Precision + Recall]

Loss function

Another important parameter to look at when training a Neural Network is the Loss function, which quantifies how well the model's predictions match the true values of the target variable. Loss is typically calculated using a function, such as categorical cross-entropy for classification tasks or mean squared error for regression tasks. The goal during training is to minimise the loss function, which involves adjusting the model parameters (e.g., weights and biases) through optimization algorithms like gradient descent. Lower loss values indicate

better agreement between the model's predictions and the true values, reflecting improved performance.

Training, validation and test sets

It is important to outline this concept of training, validation and test sets, because it is at the root of all Machine Learning models developing. During the training of a Neural Network, the dataset is typically divided into two main subsets: the training set and the validation set. Within the context of Supervised Learning (Bishop 2006), the training set is a subset consisting of input data along with corresponding target labels (or values in the case of regression tasks). The model learns from the training set by adjusting its parameters (e.g., weights and biases) through optimization algorithms like gradient descent to minimise the loss function. The training process involves iterating through the training set multiple times (epochs), updating the model parameters to improve its performance in making predictions.

The validation set is a separate subset of the dataset used to evaluate the performance of the model during training. It serves as an independent sample to assess how well the model generalises to unseen data. The validation set helps monitor the model performance, detect overfitting (where the model learns to capture noise or irrelevant patterns from the training data rather than generalise, resulting in poor performance on unseen data), and tune hyperparameters. The model performance on the validation set is evaluated periodically during training, typically after each epoch, to determine if further training is improving or worsening the model ability to generalise (Bishop & Bishop, 2023).

It is important to stress that this concept of separating the dataset into these two subsets helps ensure that the model learns effectively and generalises well to new, unseen data. Ensuring this ability is a key aspect when developing a Machine Learning model.

5-fold Cross-Validation

Cross-Validation is a well-established technique used in Machine Learning to evaluate the performance of a model by partitioning the dataset into subsets, training the model on some of the subsets, and testing it on the remaining subset. This process is repeated multiple times, with different partitions used for training and testing each time. Cross-validation helps assess the generalisation ability of the model and reduces the risk of overfitting.

SharkNet-X architecture

SharkNet-X was built using Keras and TensorFlow. In particular we used a sequential model composed by the following blocks:

- First block:
    - Conv2D (32,3,activation = ‘relu’);
    - MaxPooling2D();
- 2nd block:
    - Conv2D (32,3,activation = ‘relu’);
    - MaxPooling2D();
- 3rd block:
    - Conv2D (32,3,activation = ‘relu’);
    - MaxPooling2D();
- Flatten();
- Dense (128,activation = ‘relu’);
- Dropout (0.5);
- Output Layer: Dense (10,activation = ‘softmax’);

Here we report the network's architecture, layer by layer, showing the layer type, the output shape and the number of parameters for SharkNet-X (Tab. S1).

Layer (type)	**Output Shape**	**Number of Parameters**
conv2d (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 32)	9.248
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 32)	0
conv2d_2 (Conv2D)	(None, 52, 52, 64)	18.496
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 64)	0
flatten (Flatten)	(None, 43264)	0
dense (Dense)	(None, 128)	5.537.920
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1.290
Total parameters: 5.567.850 Trainable parameters: 5.567.850 Non-Trainable parameters: 0		

Tab. S1 - SharkNet-X Convolutional Neural Network architecture. For each layer composing the CNN, it shows the type, output shape and number of parameters. At the bottom of the table, a summary of the total number of parameters is provided. 'Trainable' and 'Non-Trainable' refer to the possibility of adjusting the network parameters during the training process.

Classification results

Here we report the confusion matrices for the test we performed using the two CNNs and the different combinations of data augmentation.

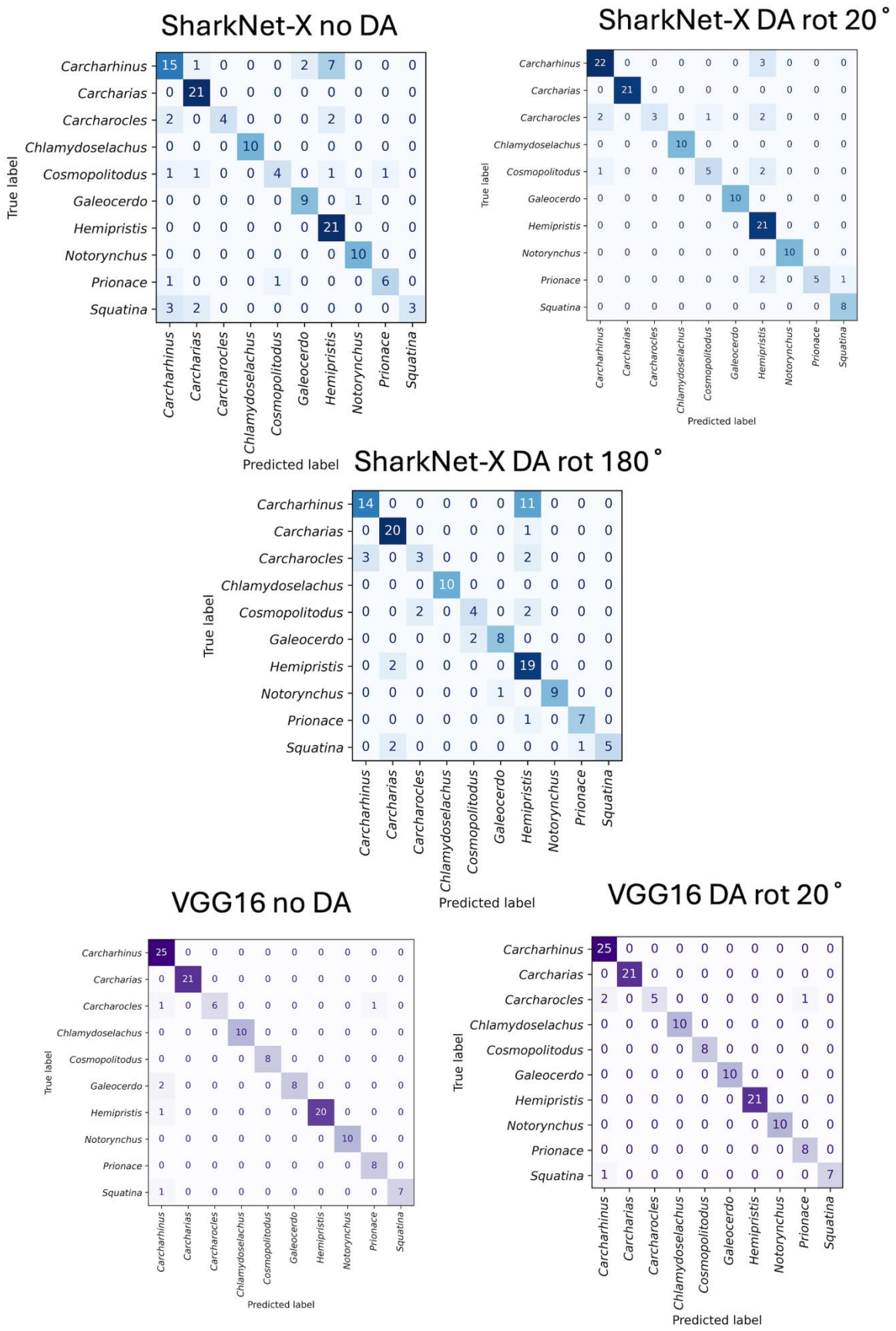
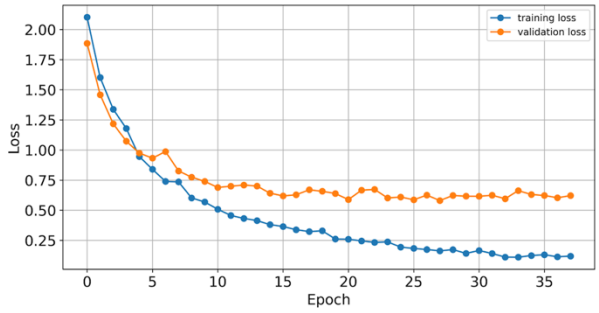


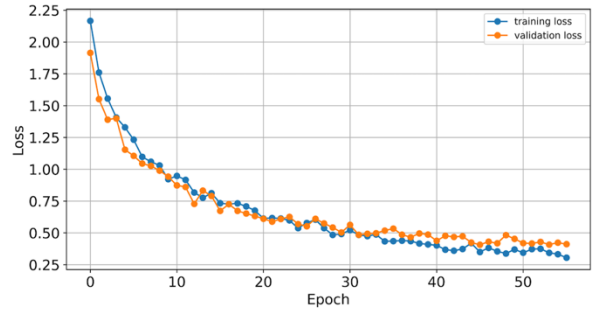
Fig. S4 - Confusion matrices for SharkNet-X and VGG16 for different possible cases: no data augmentation (label “no DA”), data augmentation with rotation angle in range up to ±20° and ±180°.

Learning curves

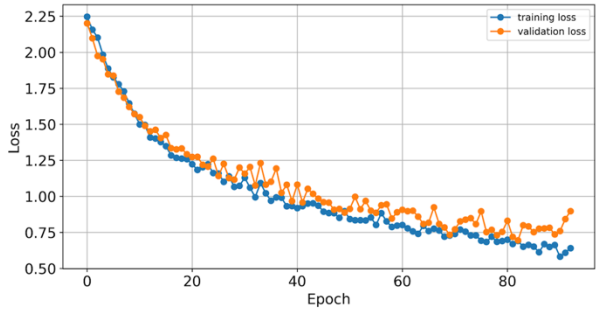
Here we report the plot of the learning curves (for training and validation dataset) relatives to the two CNNs model, for classification loss (Fig. S5) and accuracy (Fig. S6).



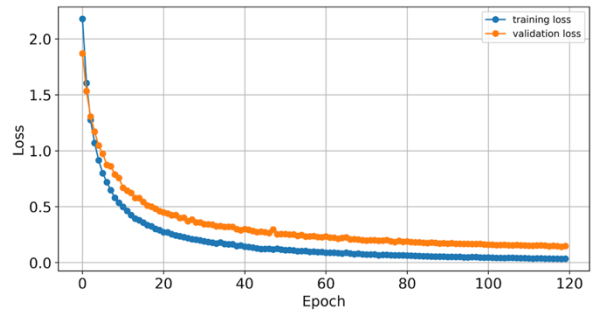
SharkNet-X no DA



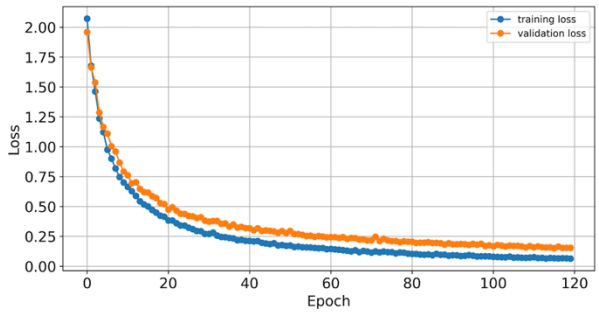
SharkNet-X DA rot 20°



SharkNet-X DA rot 180°

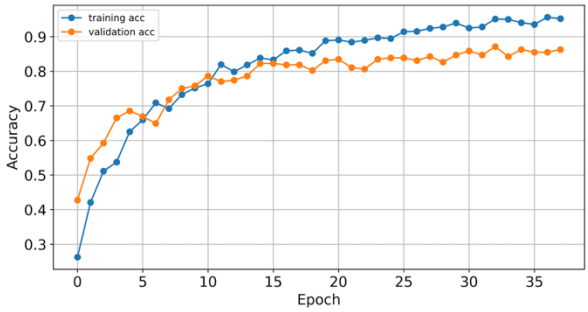


VGG16 no DA

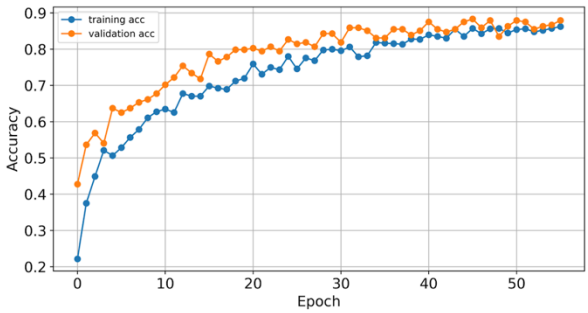


VGG16 DA rot 20°

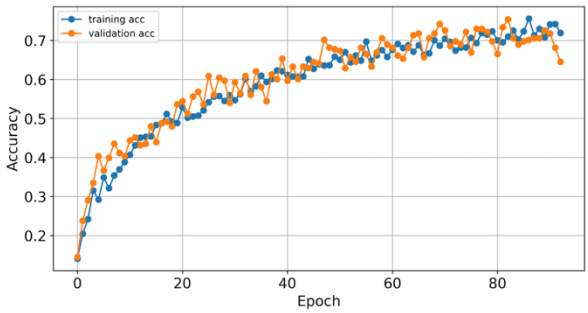
Fig. S5 - Classification loss earning curves (for training and validation dataset) relatives to the two CNNs model, for different possible cases: no data augmentation (label “no DA”), data augmentation with rotation angle in range up to ±20° and ±180°.



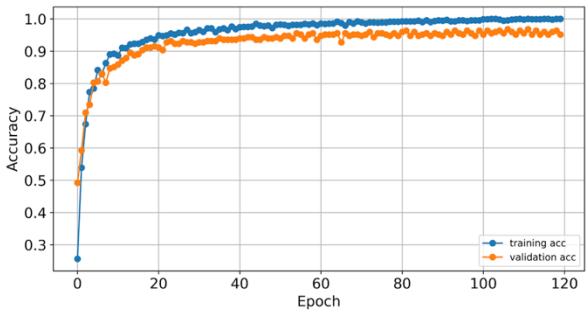
SharkNet-X no DA



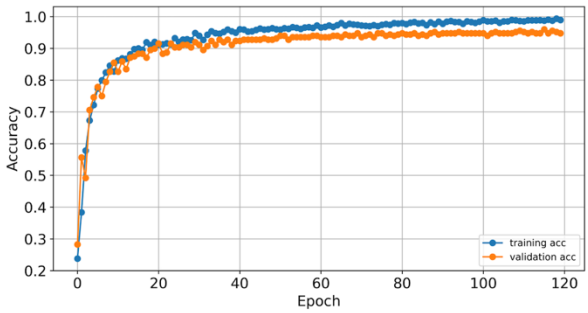
SharkNet-X DA rot 20°



SharkNet-X DA rot 180°



VGG16 no DA



VGG16 DA rot 20°

Fig. S6 - Classification accuracy earning curves (for training and validation dataset) relative to the two CNNs model, for different possible cases: no data augmentation (label “no DA”), data augmentation with rotation angle in range up to ±20° and ±180°.

Cross-Validation

The results for the selected as best model - VGG16 with data augmentation using a rotation angle up to ±180°- are reported here below in Table S2.

	CV fold 1	**CV fold 2**	**CV fold 3**	**CV fold 4**	**CV fold 5**
Accuracy (%)	88	88	91	93	85
Balanced Accuracy (%)	87	87	89	96	89
Precision (%)	90	88	93	94	88
Recall (%)	88	88	91	93	87
F1 (%)	89	88	91	93	87

Tab. S2 - Five fold Cross-Validation scores for VGG16 model with Data Augmentation (Rotation = ± 180°).

Final results:

- Average Accuracy: 89% ± 2% (std)
- Average Balanced Accuracy: 89% ± 3% (std)
- Average Precision: 90% ± 2% (std)
- Average Recall: 89% ± 2% (std)
- Average F1: 90% ± 2% (std)

t-SNE results

t-SNE results for the different CNN models are reported here below.

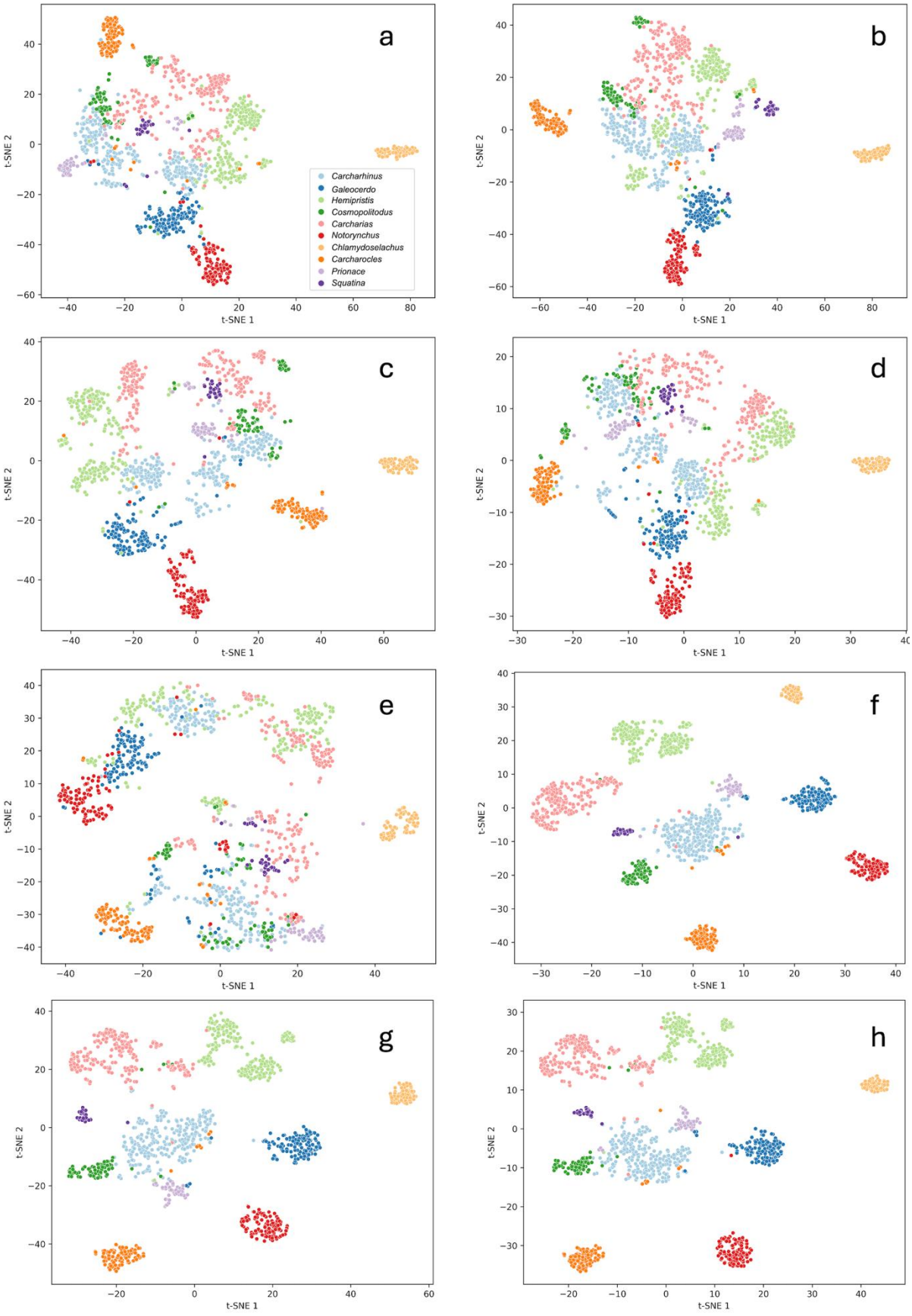


Fig. S7 - t-SNE results for the different CNN models trained. a) SharkNet-X without data augmentation, perplexity 30, exaggeration 12; b) SharkNet-X no data augmentation, perplexity 30, exaggeration 50; c)

SharkNet-X with data augmentation with rotation angle in range up to ±20°, perplexity 30, exaggeration 50; d) SharkNet-X with data augmentation with rotation angle in range up to ±20°, perplexity 80, exaggeration 12; e) SharkNet-X with data augmentation with rotation angle in range up to ±180°, perplexity 30, exaggeration 12; f) VGG16 without data augmentation, perplexity 50, exaggeration 50; g) VGG16 with data augmentation with rotation angle in range up to ±20°, perplexity 30, exaggeration 12; h) VGG16 with data augmentation with rotation angle in range up to ±20°, perplexity 50, exaggeration 50.