Fine-grained visual classification of fish[*]

Piotr Żerdziński[1,*,†]

[1]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND

Abstract

Fine-grained visual classification (FGVC) is a concept of classifying images belonging to the same metaclass. This problem is challenging due to the small differences between classes and also the small number of data. In this paper, a fine-grained classification model based on the attention mechanism is proposed. Attention allows the model to focus on small differences that determine class membership. The model used was tested on the Croatian Fish Dataset and achieved an accuracy of 94.375%.

Keywords

cnn, attention, classification, fine-grained visual classification

1. Introduction

Image processing is a basic category of problems that face AI. Image classification is the main subtype of this problem. There is huge potential in diverse approaches to data classification through different methodologies. In the case of artificial neural networks, the most important tool is convolutional networks. However, it is not possible to identify a single architecture that can solve many classification problems. Hence, various solutions are modeled that can focus on the classification of different features. An example is the possibility of using transfer learning, i.e. neural models learned on huge databases. The possibility of their use consists of using weights and training them on new data [1]. The capabilities of neural networks are also supported by additional techniques such as attention modules, as shown in [2], where the attention module was used in recurrent networks.

The problem of fine-grained visual classification, on the other hand, implies the classification of images belonging to the same metaclass. Thus, it is a more challenging task, since the differences between classes are small and may involve a small part of the image. Thus: the classification of bird species occurring in a given area may require perceiving the difference only in the shape and color of the feathers [3, 4], while the classification of aircraft assumes recognizing the difference in, for example, the shape of the wings [5]. In this paper, a fine-grained visual classification of the Croatian Fish Dataset is made. In this case, it is important not only to focus on individual small differences between species, but it is necessary to address the problem arising from the specifics of the dataset, i.e., poor visibility and noise.

Therefore, due to the specificity of the problem, it is necessary to find and focus on the most informative ones that constitute the difference between classes of regions [6, 7, 8, 9, 10, 11].

For this reason, it became necessary to implement an attention mechanism to extract the most important fragments.

In this paper, a Fine-Grained Visual Classification (FGVC) model is presented. Thus, a simple preprocessing coupled with image augmentation is made. In the second place, the CNN architecture implementing the attention mechanism crucial in the analyzed problem is presented, as well as skip connections, which support the extraction of the most relevant image elements. The main contributions of this paper are:

[*] IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania

[1,*] Corresponding author

[†] These author contributed equally.

piotzer046@student.polsl.pl (P. Żerdziński)

- new network architecture based on the attention module and skip connections,
- new CNN-based FGVC model,
- scheme for further expansion with more efficient photo preprocessing.

2. Methodology

Working on images taken underwater, we are forced to solve problems caused by the quality of the analyzed images: noise, discoloration and distortion. Water and particles dispersed in water (pollution, plankton, etc.) absorb, scatter and reflect sunlight. The extreme wavelengths in the visible light range are particularly strongly absorbed. The dominant colors of the images are therefore green and blue, and for this reason, any differences due to the color of the fish are lost and are almost invisible.

In other words, the proposed architectures implemented generative models to improve the quality of the analyzed images [12, 13]. Potentially, denoising and color correction methods also based on generative models could be applied to the fish classification problem [14, 15, 16, 17, 18]. However, the common feature of both solutions is a significant increase in the complexity of the proposed solution. The presented architecture focuses on achieving maximum efficiency with minimized complexity, so the analyzed images were processed only by simple transformations. After preprocessing, the data was divided into two sets: training and test at a ratio of 80% to 20%. Due to the initial imbalance in the number of elements in each class, there was also an imbalance when dividing the training and testing data. The data prepared in this way was then used for training and evaluation of the model.

The proposed model, shown in Figure 3, uses images with a size of 64x64 px. This size was determined based on the minimum size of the photos in the database and represents a compromise between the required size to make the photos informative and the excessive deformation of the images caused by expanding the photos. The higher dimensionality of the images did not affect the efficiency of the model, as the initial images were small, but only increased the computational complexity.

The basic elements of the presented architecture are two convolutional blocks, with a fixed composition visible in Figure 2. The first contains a convolution layer, followed by a batch normalization and a GELU activation function, and a pooling layer, whose output is subjected to a spatial dropout with a probability equal to 20%. The second, on the other hand, lacks only a pooling layer. Another important component that follows each of the blocks except the last is CBAM - Channel Attention Module Block, which implements the attention mechanism to the model. The skip connection mechanism, which passes the information after the first block deep into the model, requires a convolution layer with a kernel size of 1x1 - in this way, it is possible to change the dimension of the layer, which allows the outputs of the two blocks to be combined. At the very end, two fully connected layers are implemented. The first one is preceded by a dropout with a probability equal to 20%, and after it the GELU function is used. The output of the second uses the LogSoftmax function, which returns the probability for each class.

2.1. Preprocessing images

In the proposed architecture, the image preprocessing phase has been reduced to a minimum. As the dataset is loaded, each image is brightened, by increasing the value of each pixel by 50%. For very blurry images, the brightening allows the fish to be significantly separated from the background, while for the more accurate ones, the fish's features become more visible. An example application of this transformation is in Figure 1.

The next step is the process of augmenting the training data. It is necessary due to the small size of the initial set (see Table 1). The orientation in the case of the analyzed dataset is irrelevant, so two transformations were applied without fear of losing informativeness: vertical reflection and horizontal reflection. Both transformations are applied with a probability of 90%.

Below is the initial photo and its modification:

(a) Original photo.

(b) Photo brightened by 50%.

Figure 1: Image before and after preprocessing.

2.2. CNN model

To analyze the images and classify them, convolutional neural networks (CNNs), which are dedicated solutions to computer vision problems, were used.

CNN's operating principle is based on analyzing an image, represented as a matrix, through a series of layers and functions aimed at classifying or segmenting images. The most important of these is the convolution layer, which detects the basic features of an image through a process of convolution, that is, element-wise multiplication and summation between an image and a set of filters. The filters are also represented as matrices, are initialized randomly, and are corrected in the learning process. The problem of fine-grained visual classification requires a detailed analysis of the image due to the small differences between the classes, so the size of the filters remained small in the proposed solution.

Pooling layers reduce the size of spatial dimensions of input feature maps. The proposed model uses average pooling with a filter size equal to 2. Importantly, the pooling layer, despite the reduction of dimensionality, does not cause a significant loss of informativeness of the data. This operation is described by the formula:

,

$$AvgPool(F^c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} F_{cij},$$

where F^c is the input feature map with dimensions described as C is a number of channels, W is width, and H is height.

The other elements present in each block forming the presented model (see Figure 2) are batch normalization, GELU activation function and dropout. Due to the small size of the analyzed dataset, it became necessary to prevent overfitting. For this reason, spatial dropout is used, which removes entire feature maps, while classical dropout disables neurons. The use of the GELU function is aimed at introducing nonlinearity, which leads to the model learning more complex relationships. Moreover, it is more efficient than ReLU and ELU [19]. On the other hand, batch normalization stabilizes and speeds up the learning process, reduces internal covariance shift, improves convergence and presets slight regularization [20]. This process can be represented by the formula: (2)

$$x' = \frac{x - \hat{m}}{\hat{d}} \cdot \alpha + \beta,$$

is learnable parameter.

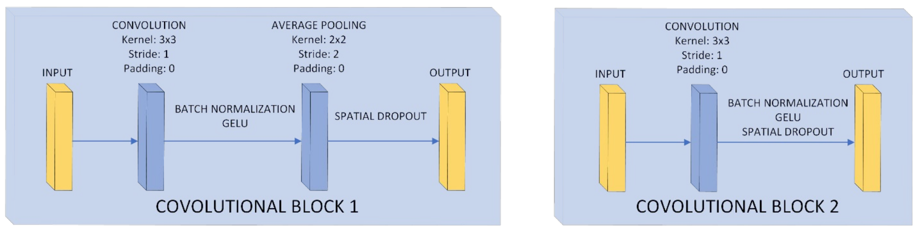


Figure 2: Two convolution blocks forming the presented model.

The fully connected layer, also known as the Dense layer implements the classic linear approach, in which each neuron in a given layer is connected to each neuron in the previous layer, and each neuron in a given layer passes its activation to each neuron in the next layer. The proposed model uses two dense layers, the first of which uses dropout functions in the form described above before linear transformation, while it passes the result of its action to the GELU activation function. The last layer, with an output size corresponding to the number of classes, passes its result through LogSoftmax functions. The task of this function is to normalize the results of the model to the distribution of the logarithm of probability. It is expressed by the formula:

$$LogSoftmax(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right). \tag{3}$$

The presented model also implements an attention mechanism that selectively focuses on parts of the analyzed image, assigning different weights to different areas. The implemented attention mechanism is based on attention block (CBAM - Convolutional Block Attention Module [21]), consisting of Channel Attention and Spatial Attention.

Due to the depth of the presented model and the small number of data in the analyzed set, the proposed model uses a skip-connections mechanism to help fight with degradation problem. The most important feature of the skip-connections mechanism is the ability to transfer low-level features captured at the initial layers of the network, to deeper layers, where they are mixed with high-level features. In the proposed model, skip connections are arranged according to the architecture of DenseNets [22], i.e. the result of the first block is passed to each subsequent layer. However, the results of subsequent layers are no longer combined. At each stage, convolutional blocks analyze current feature maps combined with low-level features from the first convolutional block.

During training, the model tries to match the real data as closely as possible, for this reason, it is necessary to use a loss function. This function, which accounts for how much the model's predictions, deviate from the actual data. Minimizing this function is therefore the main goal of training. The proposed solution uses a Cross-entropy Loss function, expressed by the formula:

$$\mathcal{L} = -\sum_{N}^{j=1} t_j \log(p_j), \tag{4}$$

where N is number of classes, t is true distribution, p is predicted distribution. Complementary to the task of minimizing the loss function is the selection of new values of model parameters, based on the value of this function. This role is assumed by the optimization algorithm, which in the proposed solution is ADAM (Adaptive Moment Estimation).

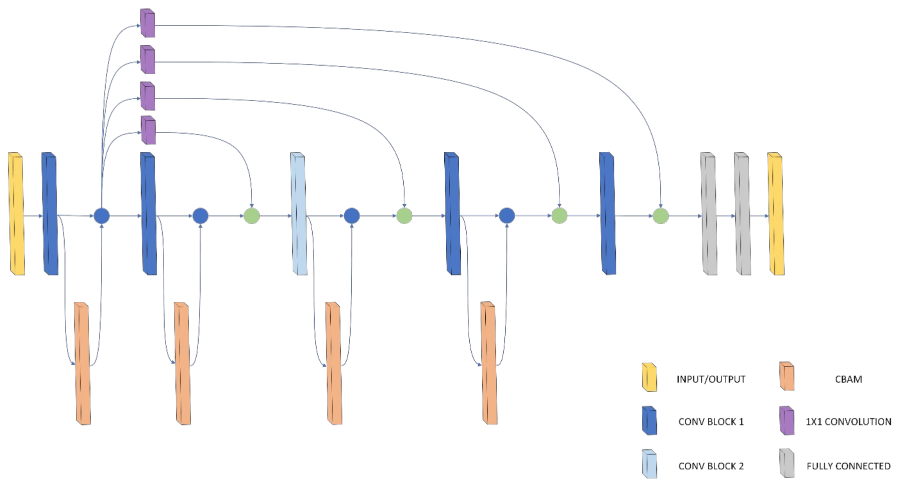


Figure 3: CNN architecture.

3. Experiments

This section is devoted to analyzing the results obtained for the Croatian Fish Dataset. The results were obtained for two approaches: in the first, the images were not preprocessed at all and were not augmented, while in the second, the full preprocessing described in Section 2.1 was implemented. The results obtained were compared to the results of the authors of the dataset.

3.1. Database

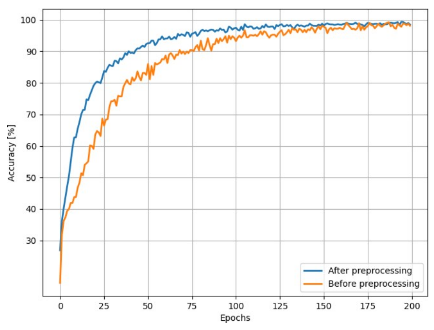
The analyzed dataset was prepared by researchers from Fulda University of Applied Sciences, Friedrich Schiller University Jena and the University of Zadar [23]. The Croatian Fish Dataset contains 764 photos of 12 species of fish found in the Adriatic Sea in Croatia (see Table 1). The images are a subset of the main dataset, which includes 1280x960 px and 1920x1080 px resolution videos. Each detected fish in the output set was marked with a bounding box and extracted as a separate photo. For this reason, the sizes of the images in the analyzed database vary from over 500 × 200 px to 19 × 23 px. Also because these are photos cut from a larger image the position of the fish and their visibility, as well as the type of background and its lighting, varies.

Species	Number of images
Chromis chromis	106
Coris julis female	57
Coris julis male	57
Diplodus annularis	94
Diplodus vulgaris	111
Oblada melanura	57
Serranus scriba	56
Spondyliosoma cantharus	51
Spicara maena	49
Symphodus melanocercus	105
Symphodus tinca	34
Sarpa salpa	17
Total	794

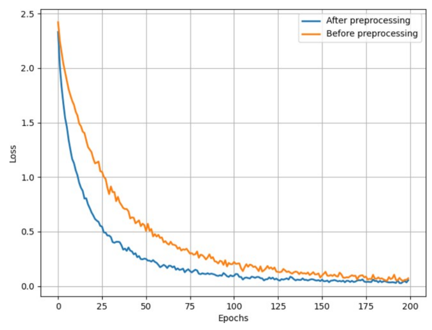
Table 1
Number of images per species.

Table 2
The results of the proposed method compared to other algorithms.

Method	Accuracy (%)
Jäger et al. (2015) [23]	66.75
Qiu et al. (2018) [24]	83.92
Sudhakara et al. (2022) [25]	95.64
Proposed architecture without preprocessing	**91.41**
Proposed architecture with preprocessing	**96.88**

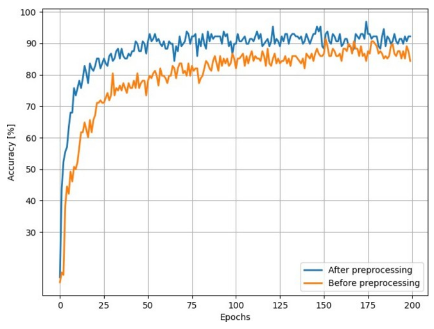


(a) Accuracy before and after preprocessing

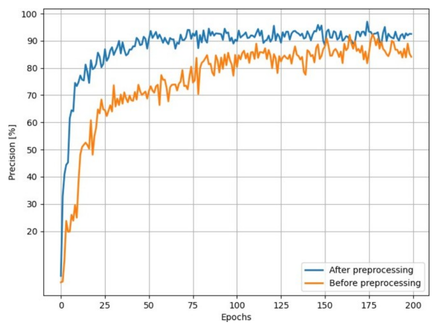


(b) Loss before and after preprocessing

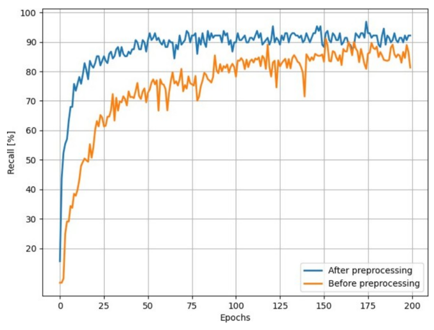
Figure 4: Accuracy and loss function during training process before and after preprocessing.



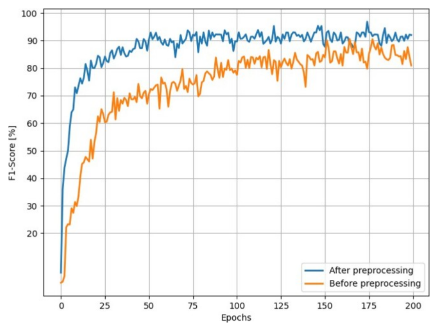
(a) Accuracy before and after preprocessing.



(b) Precision before and after preprocessing.



(c) Recall before and after preprocessing.



(d) F1-Score before and after preprocessing.

Figure 5: Performence metrics before and after preprocessing.

3.2. Results and discussion

The model was trained for 200 epochs at a batch size of 64. The graphs corresponding to the accuracy value and the loss function value for both models are in Figures 4a and 4b. The graphs for the model trained on images with and without preprocessing in the final stage of training converge to identical values - accuracy reaches nearly 100% with a loss function value of about 0.1. However, the accuracy graph for the model with full preprocessing converges faster to the maximum value, and the difference in the loss function value during training is also evident. Moreover, the value of the loss function for the case without preprocessing is more chaotic even in the final stage, which is translated into fluctuations in accuracy values.

After each epoch, the models were evaluated on a test set to check their accuracy, precision, recall, and F1-Score. Corresponding graphs can be seen in Fig 5a, 5b, 5c, and 5d. The model based on the preprocessed data reaches higher values for all metrics in the vast majority of epochs - the exception being around epochs 150 and 165. The regularity from the model training stage is observed in the evaluation stage: the model trained on the preprocessed data converges to maximum values faster reaching 80% in epoch 14, 90% in epoch 43, and ultimately reaching its maximum value of 96.88% in epoch 174. Meanwhile, the model trained on the set not preprocessed reaches the accuracy value of 80% in the 30th epoch, and surpassing 90% only 4 times, with its maximum value of 91.41% achieved in the 151st epoch. The values in the other metrics present similar patterns: the values for the preprocessed data converge more quickly to maximum values, reaching values equal to or higher than 90%, while the model for data without preprocessing does not reach this value except for the same 4 epochs where accuracy also reached that level.

Significantly, there was much more fluctuation in the value of each metric for both models. During the training stage, such behavior was evident only for the model based on data not preprocessed, so the training set in this case was only 635 elements, while the more stable training stage for the second model had 1778 images at its disposal. Thus, it can be assumed that the stability of the results obtained is a product of, among other things, the size of the test set. In the case of the test data, it was equal to 159 elements in both models, which, together with the imbalance of class sizes, leads to visible fluctuations close to the maximum value for each metric.

A summary of the obtained results can be found in Table 2, where the maximum results obtained by the presented architecture, divided into models based on data with and without preprocessing, can be seen. Also included is the accuracy value obtained by the authors of the analyzed database, which equals 66.75% and that was obtained by using pre-trained CNN with SVM for the classification part [23]. Another well-known solution in the literature is learning transfer [24], where the authors achieved accuracy on a level of 83.92%. A similar approach was shown by [25], where deep learning CNN was described. The reached accuracy was 95.64%. Compared to those works known from the literature, the proposed solution achieves a higher accuracy value, which was 96.88%. This is due to the deep network, which was extended with an attention module. This solution allowed the classifier to focus on the important features of the classified objects.

4. Conclusion

This paper proposes an attention-based CNN model supported by a simple preprocessing process. The architecture used was tested on the Croatian Fish Dataset twice, once subjecting the data to preprocessing and the second time not. The results achieved are the highest available. In the future, emphasis should be placed on:

- achieving a better method of image preprocessing and more efficient data augmentation based on generative models,
- to achieve a more efficient and accurate attention mechanism that would more accurately select key elements of the image.

References

[1] A. Jaszcz, Vgg16-based approach for side-scan sonar image analysis, IVUS 2022: 27th International Conference on Information Technology (2022).

[2] D. Połap, G. Srivastava, A. Jaszcz, Energy consumption prediction model for smart homes via decentralized federated learning with lstm, IEEE Transactions on Consumer Electronics (2023).

[3] H. Zheng, J. Fu, Z.-J. Zha, J. Luo, Looking for the devil in the details: Learning trilinear at- tention sampling network for fine-grained image recognition, 2019. arXiv:1903.06150.

[4] T. Do, H. Tran, E. Tjiputra, Q. D. Tran, A. Nguyen, Fine-grained visual classification using self assessment classifier, 2022. arXiv:2205.10529.

[5] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, A. Vedaldi, Fine-grained visual classification of aircraft, 2013. arXiv:1306.5151.

[6] Y. Chen, Y. Bai, W. Zhang, T. Mei, Destruction and construction learning for fine-grained image recognition, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 5152–5161. doi:10.1109/CVPR.2019.00530.

[7] S. Huang, X. Wang, D. Tao, Snapmix: Semantically proportional mixing for augmenting fine-grained data, 2020. arXiv:2012.04846.

[8] H. Zheng, J. Fu, T. Mei, J. Luo, Learning multi-attention convolutional neural network for fine-grained image recognition, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 5219–5227. doi:10.1109/ICCV.2017.557.

[9] H. Zheng, J. Fu, Z.-J. Zha, J. Luo, Looking for the devil in the details: Learning trilinear at- tention sampling network for fine-grained image recognition, 2019. arXiv:1903.06150.

[10] T. Do, H. Tran, E. Tjiputra, Q. D. Tran, A. Nguyen, Fine-grained visual classification using self assessment classifier, 2022. arXiv:2205.10529.

[11] D. Połap, A. Jaszcz, N. Wawrzyniak, G. Zaniewicz, Bilinear pooling with poisoning detection module for automatic side scan sonar data analysis, IEEE Access (2023).

[12] C. Qiu, S. Zhang, C. Wang, Z. Yu, H. Zheng, B. Zheng, Improving transfer learning and squeeze- and-excitation networks for small-scale fine-grained fish image classification, IEEE Access 6 (2018) 78503–78512. doi:10.1109/ACCESS.2018.2885055.

[13] D. Połap, A. Jaszcz, Heuristic feedback for generator support in generative adversarial

network, Proceedings of the 16th International Conference on Agents and Artificial Intelligence 3 (2024) 863–870.

[14] C.-H. Yeh, C.-H. Huang, C.-H. Lin, Deep learning underwater image color correction and contrast enhancement based on hue preservation, in: 2019 IEEE Underwater Technology (UT), 2019, pp. 1–6. doi:10.1109/UT.2019.8734469.

[15] Y. Wang, J. Zhang, Y. Cao, Z. Wang, A deep cnn method for underwater image enhancement, in: 2017 IEEE International Conference on Image Processing (ICIP), 2017, pp. 1382–1386. doi:10.1109/ICIP.2017.8296508.

[16] J. H. Park, D. Han, H. ko, Adaptive weighted multi-discriminator cyclegan for underwater image enhancement, Journal of Marine Science and Engineering 7 (2019) 200. doi:10.3390/jmse7070200.

[17] X. Chen, P. Zhang, L. Quan, C. Yi, C. Lu, Underwater image enhancement based on deep learning and image formation model, 2021. arXiv:2101.00991.

[18] M. J. Islam, Y. Xia, J. Sattar, Fast underwater image enhancement for improved visual perception, 2020. arXiv:1903.09766.

[19] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), 2023. arXiv:1606.08415.

[20] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. arXiv:1502.03167.

[21] S. Woo, J. Park, J.-Y. Lee, I. S. Kweon, Cbam: Convolutional block attention module, 2018. arXiv:1807.06521.

[22] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, Densely connected convolutional networks, 2018. arXiv:1608.06993.

[23] J. Jäger, M. Simon, J. Denzler, V. Wolff, K. Fricke-Neuderth, C. Kruschel, Croatian fish dataset: Fine-grained classification of fish species in their natural habitat, 2015, pp. 6.1–6.7. doi:10.5244/C.29.MVAB.6.

[24] C. Qiu, S. Zhang, C. Wang, Z. Yu, H. Zheng, B. Zheng, Improving transfer learning and squeeze- and-excitation networks for small-scale fine-grained fish image classification, IEEE Access 6 (2018) 78503–78512. doi:10.1109/ACCESS.2018.2885055.

[25] M. Sudhakara, M. J. Meena, K. R. Madhavi, P. Anjaiah, L. P. K, Fish classification using deep learning on small scale and low-quality images, International Journal of Intelligent Systems and Applications in Engineering 10 (2022) 279 –. URL: https://ijisae.org/index.php/IJISAE/article/view/2292.