Fish Classification: Approaches based on ResNet18, ResNet50 and Autoencoder

Alberto Provenzano PROVENZANO.ALBERTO21@GMAIL.COM
Giuseppe Leocata PEPPELEOCATA@GMAIL.COM
Master's Degree course in Computer Engineering (LM32)

1. Introduction

Our goal was to define a deep learning model capable of classifying 15 species of fish using a small annotated dataset and a large unannotated dataset. The main challenge is the small size of the annotated dataset, which makes it more difficult to train the model for the classification task. To achieve our goal, we used three different approaches:

- **Model 1**: We used a pre-trained ResNet18, fine-tuning the classifier on the small annotated dataset. The ResNet18 was pre-trained on a large dataset, such as ImageNet, which contains millions of images classified into thousands of categories. Through fine-tuning the classifier, we adapted the network to the specific needs of our small annotated dataset.

- **Model 2**: We defined our custom autoencoder with a classifier, utilizing both the small annotated dataset and the large unannotated dataset for training. This approach overcomes the problem of the small annotated dataset because the autoencoder is trained on the large unannotated dataset for the reconstruction task: the encoder learns to generate an optimized latent space for the classification task while the classifier is trained on the small annotated dataset, leveraging the meaningful representations learned by the encoder. This approach allows us to achieve better classification performance despite the scarcity of annotated data by combining unsupervised learning for feature generation with supervised learning for classification.

- **Model 3**: Using the autoencoder with a classifier we previously trained, we create pseudo-labels from the large unannotated dataset. We then combined this dataset with part of the annotated one and used this combined dataset to train a pre-trained ResNet50. Finally, we fine-tuned the classifier with the small annotated dataset. This approach combines the advantages of the first two, as the autoencoder with the classifier was trained on the unannotated dataset and thus can effectively utilize the information contained in that dataset, allowing us to increase the data used for fine-tuning the ResNet50.

Additionally, we compared the approach adopted for model 3 with the ResNet18, and referred to this experiment as **model 4**.

2. Model Description

2.1. Model 1

For the first model, we used a ResNet18, which is a deep network belonging to the family of residual networks. The main characteristic of these networks is the use of Residual Blocks, which allow the passage of information between layers, reducing the problems of vanishing gradient and performance degradation that typically afflict very deep neural networks. The architecture of ResNet18 begins with an Input Layer, which processes the input image through a Convolutional Layer applying a 7x7 convolution, followed by Batch Normalization, ReLU activation function, and finally, Max Pooling, which reduces the spatial dimension of the feature maps. The network continues with four Residual Blocks, each of which consists of two BasicBlocks. Each BasicBlock contains two 3x3 convolutional layers followed by Batch Normalization and ReLU. Within each Residual Block, an addition operation combines the input of the first convolutional layer of the first BasicBlock with the output of the second convolutional layer of the second BasicBlock (before ReLU). At the end of the Residual Blocks, ResNet18 concludes with an Output Layer, consisting of a Global Average Pooling Layer, followed by a Fully Connected Layer with 1000 output units as ResNet18 is trained for classification on 1000 classes. Additionally, we replaced the classifier of ResNet18 with our classifier: a linear layer with 15 output classes.
As optimization algorithm, we have utilized Adam, which has become popular in the field of deep learning because it allows achieving good results quickly. The name "Adam" comes from the term "adaptive moment estimation." Adam differs from classical stochastic gradient descent (SGD) because, while SGD maintains a single learning rate for all weight updates, Adam computes individual adaptive learning rates for different parameters based on estimates of the first and second moments of the gradients. Specifically, in addition to calculating the direction where the loss function is minimized, Adam also evaluates the rate at which it is "descending" (or "ascending"), thus preventing from taking steps too large by continuously adjusting the step size. As the loss function, we used the cross-entropy loss since for this first model we only used the small annotated dataset. The cross-entropy loss in PyTorch is defined analytically as:

$$\mathbf{L}(x, y) = \mathbf{L} = \left\{l_1, \ldots, l_N\right\}^\top, \quad l_n = -w_{y_n} \log \left(\frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \right) \cdot 1\{y_n \neq \text{ignore\_index}\}$$

2.2. Model 2

The second model is an autoencoder with a classifier. This model was defined by us and its structure is very simple, consisting of 3 parts which are:

- **encoder:** consists of 3 convolutional layers. For each convolutional layer, we used ReLU as the activation function, and after each ReLU, we used a MaxPool with a kernel size of 2 and a stride of 2 to halve the size of the image. Therefore, at the output of the encoder, the size of the image has been reduced by a factor of 8.

- **decoder:** characterized by 3 transpose convolutional layers with a stride of 2 so that each layer doubles the size of the input image in order to obtain the reconstructed

image at the output from the latent space produced by the encoder. In this case, ReLU was used as the activation function for the first two layers, while a sigmoid was used for the last one to ensure that the output values are normalized in the desired range [0, 1], ensuring that the reconstructed image maintains the same pixel intensity scale as the input.

- **classifier:** simply consists of 2 linear layers where ReLU is applied to the first linear layer.

The advantage of the autoencoder with classifier lies in its ability to be used on both annotated and unannotated data simultaneously in a single training phase. This is because there are two branches at the output of the encoder, one towards the decoder for the reconstruction task and one towards the classifier for the classification task. Again, Adam optimizer was used in this case, while two different loss functions were used, one for each task. For classification, cross-entropy loss was used, while for reconstruction, we used MSE, which in PyTorch is defined analytically as:

$$\ell(x, y) = L = \left[\ell_1, \ldots, \ell_N\right]^\top, \quad \ell_n = (x_n - y_n)^2$$

2.3. Model 3

For model 3, we chose the ResNet50: another CNN distinguished by its depth and complexity. Unlike ResNet18, this one contains 32 additional convolutional layers, but still relies on the concept of residual blocks (i.e. skip connections). With such depth, ResNet50 is capable of capturing more complex information present in images. The architecture here also consists of Input Layer – Residual Blocks – Output Layers. Specifically, there are 4 residual blocks, each composed of a different number of bottlenecks, which are sub-blocks characterized by 3 convolutional layers of size 1x1 – 3x3 – 1x1, where ReLU is used as the activation function and Batch Normalization as the standardization technique between each layer.
Once again, we decided to replicate the choices made for model 1, using Adam as the optimization algorithm and Cross Entropy as the loss function. Obviously, in this case too, we replaced the classifier of ResNet50 with our own classifier: a linear layer with 15 output classes.

3. Dataset

The data used to achieve the objective are:

- **labeled train set:** 150 images of 15 fish species (10 images per species), each with its corresponding label. The labels consist of integers ranging from 0 to 14.
- **unlabeled set:** 43156 unlabeled fish images.
- **pseudo-labeled set:** 21109 labeled images (also of fish) extracted from the previous dataset using the second model discussed in the report. Since these are pseudo-labels, we deemed it appropriate to discard all classifications with a probability below the 80% confidence threshold (this justifies the downsizing). This dataset was then concatenated with a subset of the labeled train set, consisting of 120 images (following the split between the validation set and the train set), resulting in what we called *combined set*.
- **test set:** 148 unlabeled images intended for evaluating the classification capabilities of all models.

We applied both resizing the images to 224x224 (ResNets were trained on images of this size) and normalization of the pixel values to all datasets; in particular, each pixel x was replaced with: $x_{\text{norm}} = \frac{x-0.5}{0.5}$
Regarding Data Augmentation techniques, we applied horizontal flipping of the image to the labeled train set with a 50% probability along with a random rotation chosen within the range of [-30°, 30°]. The same transformations mentioned above were also applied to the unlabeled set for training model 2. For training models 3 and 4, no data augmentation was used for the unlabeled set, so we used the same transformations as the test set. Finally, for the pseudo-labeled set, as data augmentation, we maintained the same horizontal flipping as the labeled train set but extended the angle of random rotation to [-45°, 45°].

4. Training details and procedure

1. **Evaluation Metrics**
   After several attempts where initially we chose to use loss as the evaluation metric, we later opted for validation accuracy as it led to slightly better results. For this reason, we selected the strategy of model selection to store the configuration of the model weights corresponding to the epoch where a higher validation accuracy was achieved.

2. **Training procedure**
   For all models, the batch size used for the test set loader was 32.

   - **Model 1:** Regarding the validation strategy, we split the labeled train set into two subsets, one for training and the other for validation, respectively comprising 80% and 20% of the original set. The samples in the validation set were chosen systematically; specifically, the first two samples, in order, were taken for every 10 samples from the original dataset, ensuring 2 samples for each class. Additionally, we used a batch size of 32 for the training subset loader and 15 for the validation loader. We initialized the ResNet18 with default weights obtained by training the network on ImageNet. For the optimizer, we used Adam with a learning rate of 0.01 and conducted a single training cycle of 30 epochs.
   - **Model 2:** For training the classifier, we used the annotated training set, and the same validation strategy as in model 1 was maintained, including the batch size. For training the autoencoder, the unlabeled set was used, and for the validation strategy, the dataset was split into two subsets with an 80-20 ratio, but randomly. However, in this case, both the validation and training subset loaders used a batch size of 128. We trained the model from scratch using an Adam optimizer, this time with a learning rate of 0.0001, and conducted a single training cycle of 40 epochs.
   - **Model 3/4:** For creating the pseudo-labeled set, we used a batch size of 64 for the loader of the unlabeled set. For the combined set, we followed the validation strategy of randomly splitting the dataset into 2 subsets, with 75% chosen for the training part and 25% for the validation part. A batch size of 64 was used for the loaders of both subsets. For both models, we initialized their weights to default values, i.e. those obtained after training on ImageNet. In both cases, two training cycles were performed:
     - **first training cycle:** It concerns the fine-tuning of the entire network on the combined set, for which we used the Adam optimizer with a learning rate of 0.00001, and conducted 25 training epochs.
     - **second training cycle:** The network was initialized with the checkpoint from the first training cycle, corresponding to the model weights recorded at the epoch where the best validation accuracy was achieved. We then fine-tuned only the classifier on the small labeled set, maintaining the same validation strategy applied to this dataset as in models 1 and 2. In this case, we used the Adam optimizer with a learning rate of 0.01, and trained the network for 30 epochs.

5. Experimental Results

- **Model 1**: Based on the training details outlined in the previous paragraph, the following learning curves were obtained:

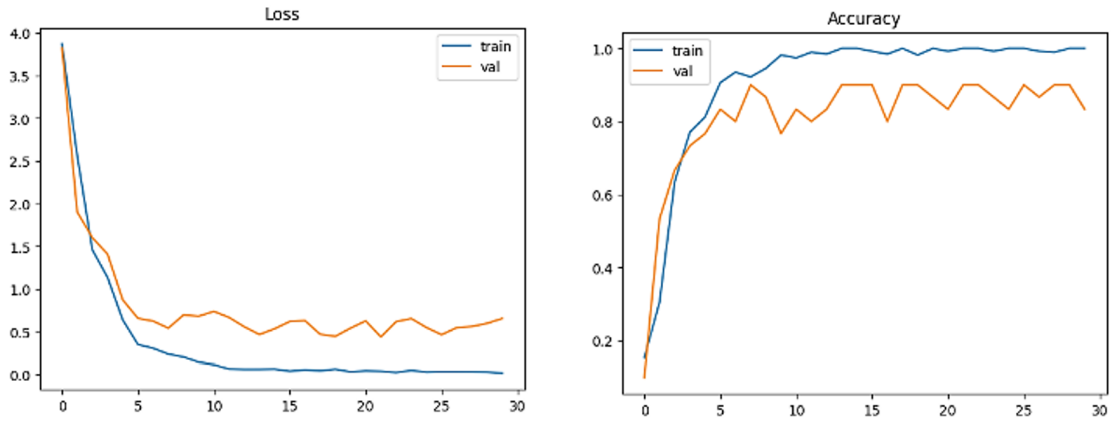

Figure 1: Trend of loss and accuracy for model 1

After training, we tested the model for classification on 50% of the test set, obtaining a score of 0.810.

- **Model 2:** For the autoencoder with classifier, on the other hand, we have these curves:

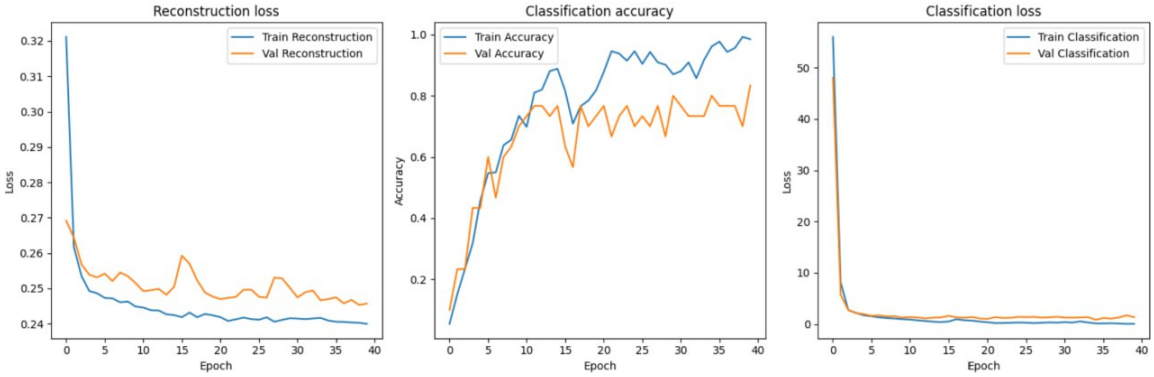

Figure 2: Trend of loss (reconstruction and classification) and accuracy of model 2

Also in this case, after training, we tested the model on half of the test set for the classification task, obtaining a score of 0.837.

- **Model 3/4:** For models 3 and 4 (ResNet50 and ResNet18), we used the same configuration in terms of training procedure and details, with the aim of comparing them.
  - **ResNet50:** For the first training (fine-tuning of the entire network), the loss and accuracy curves are as follows:

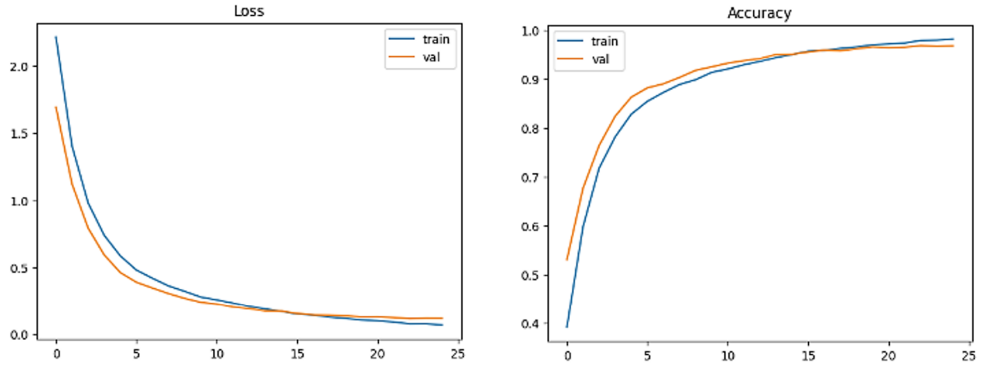

Figure 3: Trend of loss and accuracy for model 3 training 1

Evaluating the model on half of the test set directly after this first training, we obtained a score of 0.783. We then thought that this deterioration in performance could be due to the pseudo-labels, so we conducted further fine-tuning only on the classifier using the small labeled dataset, in order to compensate for any inaccuracies introduced by the pseudo-labels. This fine-tuning resulted in these curves:

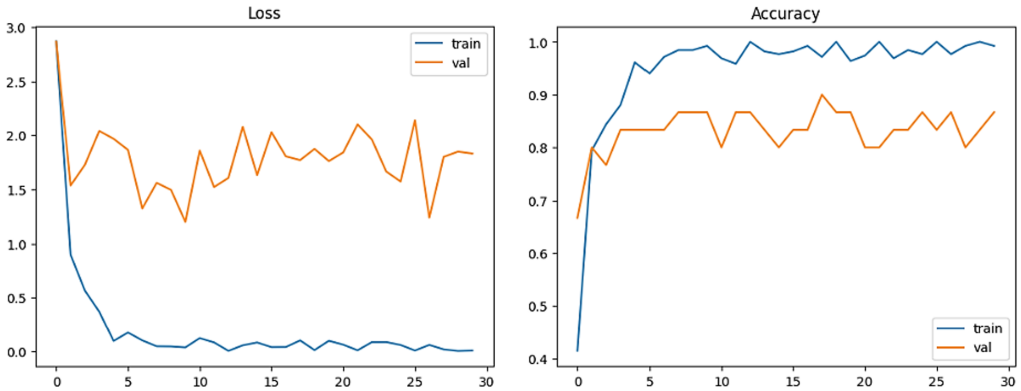

Figure 4: Trend of loss and accuracy for model 3 training 2

Thanks to this second training cycle, the evaluation score on half of the test set has increased to a value of 0.905.

– **ResNet18:** For the first training cycle, we obtained the following graphs:

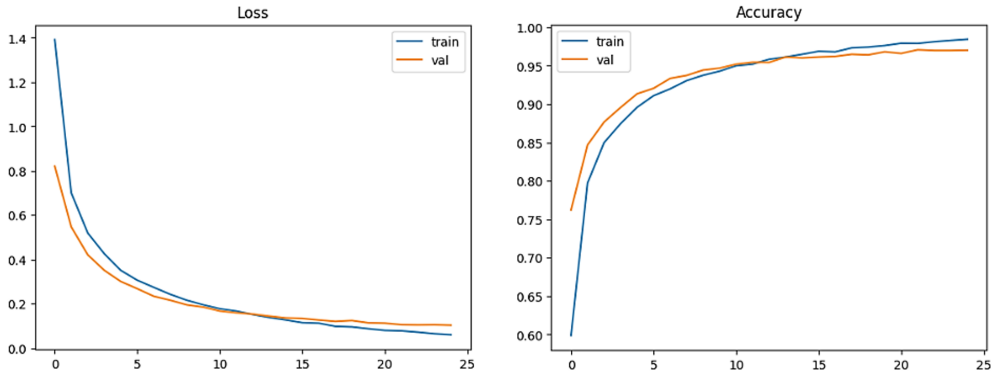

Figure 5: Trend of loss and accuracy for model 4 training 1

In this case, evaluating the model on half of the test set directly after this first training, we obtained a score of 0.743, lower than the result obtained with ResNet50. For the second training cycle, the obtained curves are as follows:

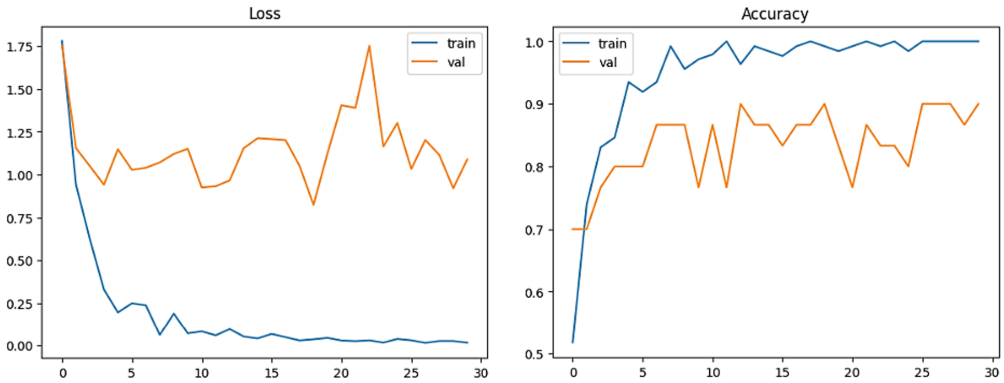

Figure 6: Trend of loss and accuracy for model 4 training 2

In this case, the final score obtained for the classification of 50% of the test set was 0.864, lower than that obtained with ResNet50, as we could have expected based on the score from the first training cycle.