A Large-Scale Dataset for Fish Classification

Bruno Casella CASELLA0798@GMAIL.COM

1. Model Description

I describe here a deep model that consists of 2 Convolutional Layers, to extract and recognize increasingly complex features, followed by 2 Fully Connected Layers and a Classifier to perform multiclass classification (see Fig. 1). Initially, input data flows trough the convolutional layers, to exploit image information. Then the extracted features are provided to the fully connected layers and to the classifier, which predicts the class outcome among one of the 9 possible classes.

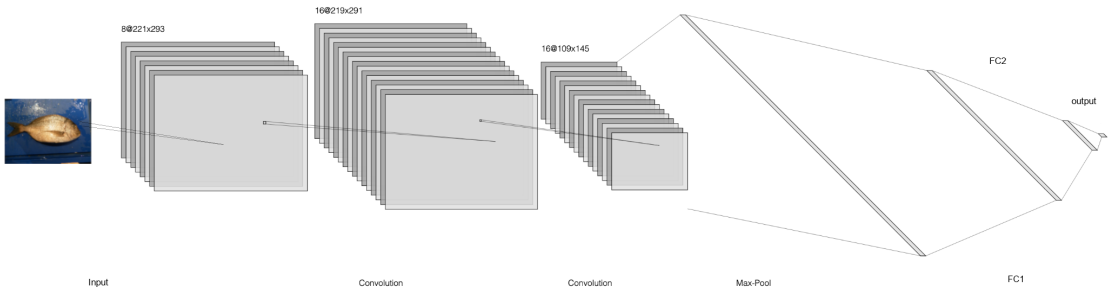

Figure 1: A simple representation of the proposed model.

Initially, It presents two convolutional layers, to process the input volume. The two convolutional layers have respectively 8 and 16 channels. The first layer has a kernel size of 5 applied with a stride of 2 in each dimension, followed by a ReLU activation function and by a layer of Batch Normalization. The second layer has a kernel size of 3 applied with a stride of 1 in each dimension, followed by a ReLU activation function and by a layer of Batch Normalization. Moreover, in this second layer, is applied also Max Pooling to reduce the size of the feature map for faster computations, keeping most important information.

The features produced by these convolutional layers are flattened and provided as an input to the *Fully Connected* layers. The first layer has 4096 output features, taking as input 252880 features (the output of the convolutional layers). The second layer takes as input the output of the previous layer, and it has as output 1024 features. Finally, there is a classifier, with an output size of 9, that are the classes of the dataset. Indeed, I will use the Cross Entropy as Loss Function, so the output will be a probability vector, meaning it represents predicted probabilities of all classes, summing up to 1. This method combines the Softmax activation function principle and the negative log likelihood loss.

2. Dataset

The dataset was provided by the University of Economics of Izmir, Turkey.[1] Images of nine different seafood types are collected from the fish counter of a supermarket. Two cameras are used in the dataset gathering process, a Kodak Easyshare Z650 and a Samsung ST60 with spatial resolutions of 2832 × 2128 and 1024 × 768 pixels, respectively. While 50 distinct fish images are collected per each of seven classes as follows: red mullet, gilt head bream, horse mackerel, sea bass, red sea bream, black sea sprat and striped red mullet, 30 distinct images are captured for trout and shrimp. All fish in the image acquisition process are fresh, and they are positioned in various displacements and angles but lighting conditions do not change significantly. Lastly, instead of a clean white background, a blue and noisy background is preferred in order to make the dataset usable in studies with real-life problems. Example images from the collected dataset are illustrated in Fig. 2.

Figure 2: Example images from the collected dataset.

Furthermore, the sample images of all nine classes are resized to 590×445 pixels by nearly preserving their aspect ratio. Then, these samples are passed through an augmentation algorithm where each image is rotated with non-repeated random angles and they are reflected. For each seafood type, 1000 images are finally obtained for the construction of the dataset, and so, there full dataset consists of 9000 images. I manually split the dataset into training, validation and test, with one of the usual convention, 80%, 10%, 10%. So, at the end, training set consists of 7200 images (800 per class), validation and test consists both of 900 images (100 per class). The labels of each each class are taken using the ImageFolder function of Pytorch. It takes as label the name of the folder containing the images. Indeed, for each folder of train, validation and test set, there are 9 folders titled with the name of the respective class, containing the images of that class. All images for each class are ordered from "00000.png" to "01000.png". CNNs are not rotation invariant, so data augmentation can help. For this reason I applied horizontal and vertical flip, with a chance of 50%. In this way, in any epoch, the DataLoader will apply a fresh set of random operations "on the

fly". So, instead of showing the exact same items at every epoch, we are showing a variant that has been changed.

3. Training procedure

As I said before, I employed data augmentation mechanisms, like mirroring, to reduce overfitting issues. Beside the FishNet model, I tested also other 3 models, one baseline model with only one convolutional layer, and two deeper models, respectively based on a sequence of 3 and 4 convolutional layers. In the baseline model I used a stride size equale to 3 to reduce the number of parameters of the network, avoiding Runtime Errors of CUDA out of memory. As training loss, I employed the Cross Entropy Loss, that combines the Negative Log Likelihood Loss and the LogSoftmax in a single class. A cost function based on multiclass log loss for data set of size N might look like this:

$$J = -\frac{1}{N}(\sum_{i=1}^{N} y_i * log(\hat{y}_i)) \tag{1}$$

where y is the ground truth vector and $\hat{y}$ is the estimate.
All the models are trained from scratch for 10 epochs on a Nvidia Tesla K80. Batch size is set to 64 for all the models. Stochastic Gradient Descent is chosen as optimizer algorithm using a learning rate of 0.01.

4. Experimental Results

Models were trained as described in the previous section. Table 1 shows test results for the proposed architectures as well as of ablation studies (i.e., different variants of the final architecture when adding or removing layers).

Model	Validation Accuracy	Test Accuracy
Baseline Net (1 conv layer)	90.49%	87.92%
+ Layer 2 (FishNet)	97.50%	92.71%
+ Layer 3	92.28%	90.10%
+ Layer 4	93.54%	83.96%
Best model (FishNet)	97.50%	92.71%

Table 1: Test performance of the models.

As shown in the previous table, the model described in this paper performed slightly better than the baseline and the deeper models in validation test, and also in test phase.

References

[1] Oguzhan Ulucan, Diclehan Karakaya, and Mehmet Turkan. "A Large-Scale Dataset for Fish Segmentation and Classification". In: *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE. 2020, pp. 1–5.