Snakes' species classification

Grazia Noemi Compagnino NOEMICOMPAGNINO97@LIVE.IT
Maqbool Thoufeeq Tharayil THRMBL95B15Z222Z@STUDIUM.UNICT.IT

1. Snakes' species model description

We describe here **SnakeImages**, a dataset that consists in RGB images of snakes, divided in 5 classes on the basis of their corresponding species and geographic location (continent, country). The goal is to create a system capable to automatically categorize snakes on the species class (see Fig.1). This analysis has the scope to explore how can be reduced erroneous and delayed healthcare actions and improved snakebite eco-epidemiological data thanks to the help of Machine Learning's snake identification. Initially, input data flows trough a 2D encoder, to exploit inter slice volumetric information. Then the extracted features are provided to a capsules-based encoder, which predicts treatment outcome as positive or negative. It presents an initial 2D encoder of five convolutional layers, to pro-

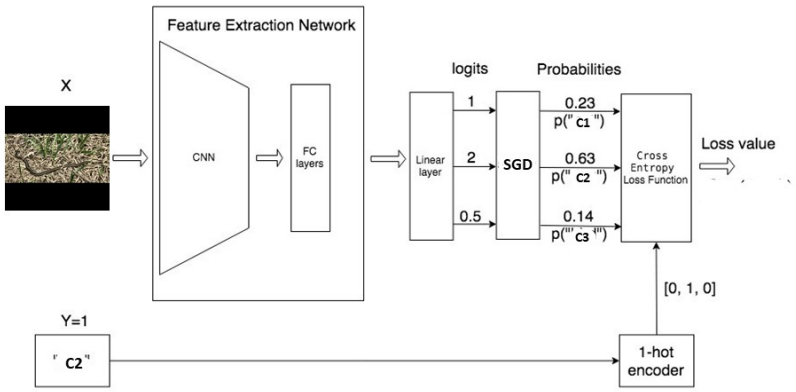

Figure 1: A simple representation of the proposed model.

cess the input volume. The five 2D convolution layers have respectively 3, 64, 128, 256 and 512 channels, 3x3x3x3x1 kernels applied with a stride of 1 in each dimension and followed by a ReLU activation function. The features produced by the 2D encoder is then reshaped, concatenating volumetric features over the channels, and provided as an input to the *primary* capsules. (This layer has 512 primary capsules, taking as input the features

learned through by the 2D convolutional encoder and producing their combinations. The 32 "primary capsules" are similar to convolutional layer in their nature. Each capsule applies 128 9×9 convolutional kernels (with stride 1) to input volume and, therefore, produces 128×3×3 output tensor. Since there are 32 such capsules, the output volume has a shape of 32×128×3×3. The output of primary capsules is passed to the output capsules layer, that, given the binary classification nature of our problem, consists of two capsules, one for each class. Each capsule takes as input a 32×128×3×3 tensor, which can be seen as 1152 input vector. Each of these input vectors has its own weight matrix that maps 128-dimensional input space to the 32-dimensional capsule output space. Each output capsule outputs two 32-dimensional vectors v_i, one for each output class. We applies a 2D convolution over an input signal composed of several input planes. The output value of the layer with input size.

2. Snakes' species dataset

The majority of the data were gathered from online biodiversity platforms (i.e.,iNaturalist, HerpMapper). The **SnakeImages** dataset consists of 17389 snakes images, 384*384 pixels (a commonly used size for CNN inputs), containing the representation of a species of snake each. Each training class contains more than 3000 samples. We procedurally divided the dataset in training set, test set and validation set, for 0.8, 0.1 and 0.1 fraction respectively. So, we will have 13913 images in training sample, and 1738 in validation and test samples. The snake species we can see are:

- **Class 1:** Nerodia Sipedon - Northern Watersnake
- **Class 2:** Thamnophis Sirtalis - Common Garter snake
- **Class 3:** Storeria Dekayi - DeKay's Brown snake
- **Class 4:** Patherophis Obsoletus - Black Rat snake
- **Class 5:** Crotalus Atrox - Western Diamondback rattlesnake

So, when we import the dataset, we transform each image samples into 64x64 size RGB initially for sample representation. After that, when we add it to the train set, we transform the image sample to 28x28 to Tensor format which is normalized in 0.5x0.5.

Figure 2: A random sample of snake.

3. Training procedure

We employ several data augmentation mechanisms, namely, random input rotation, translation and mirroring, to reduce overfitting issues. We use 2D Maxpooling technique for down sampling the input along its spatial dimensions of Height and Width by taking the maximum value over each input window for each channel of input.

$$out(N_i, C_out_j) = bias(C_out_j) + \sum_{j=0}^{(C_in-1)} Weight(C_out_j, k) * input(N_i, k) \tag{1}$$

where N is the batch size an output of class C denotes a number of channels, H is the height of input planes in pixels and W is width in pixels.
Stochastic Gradient Descent (SGD) is chosen as optimizer algorithm using a learning rate of 0.01. For the baseline model we use standard binary Cross-Entropy Loss. Then we compute the Cross-Entropy Loss between input and target using nn.CrossEntropyLoss() method, useful to train a classification problem with C classes.

Also we use Adaptive Average Maxpooling which is an average pooling operation that, given an input and output dimensionality, calculates the correct kernel size necessary to produce an output of the given dimensionality from the given input. We create a fully connected layer by applying linear transformation with 10 input features and 1024 output features and vice-versa with a ReLu activation function. We Implements stochastic gradient descent as the optimizer with the default learning rate of 0.01.

Then we compute the cross entropy loss between input and target using CrossEntropyLoss() method. It is useful when training a classification problem with C classes.

All the models are trained from scratch for 10 epochs on a Nvidia GeForce RTX 2060 with 32GB of VRAM. Stochastic Gradient Descent(SGD) is chosen as optimizer algorithm using a learning rate of 0.01.

4. Experimental Results

Several experiments conducted by us to test the effectiveness of the proposed **SnakeImages** model in comparison to traditional approaches. The baseline network and the **SnakeImages** were trained and tested individually. Models were trained as described in the previous section.

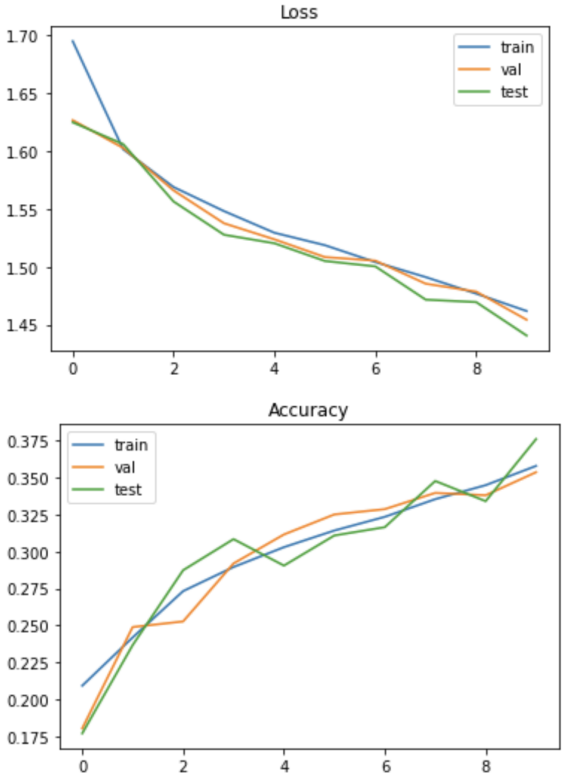

Figure 3: Best model.

We choose as the best model the one with 3 layers, a Fully-Connected layer (FC) and the classifier. By the plot we can see that our model is predicted with an accuracy of 37.57% and that the accuracy increase in accordance to the number of epochs. Table 1 shows test results for the proposed model as well as of ablation studies (i.e., different variants of the final model when adding or removing layers).

Model	Accuracy
Baseline Net	28.31%
– + Layer 1 + FC + classifier	34.36%
– + Layer 2 + FC + classifier	37.57%
– + Layer 3 + FC + classifier	33.35%
– + Layer 4 + FC + classifier	29.04%
– + Layer 5 + FC + classifier	28.31%
Your final model	37.57%

Table 1: Test performance of models.