Image Intel Classification Model

Fabiana Monaco FABIANAMONACO96@GMAIL.COM
Federico Puglisi FEDERICO.PUGLISI10@GMAIL.COM

1. Model Description

We describe here a deep model that consists of 4 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 that, generally increases the channels and reduce the image size, to exploit image information. Then the extracted features are provided to the fully connected layers and to the classifier that, combine the features extracted before and predicts the class outcome among one of the 6 possible classes.

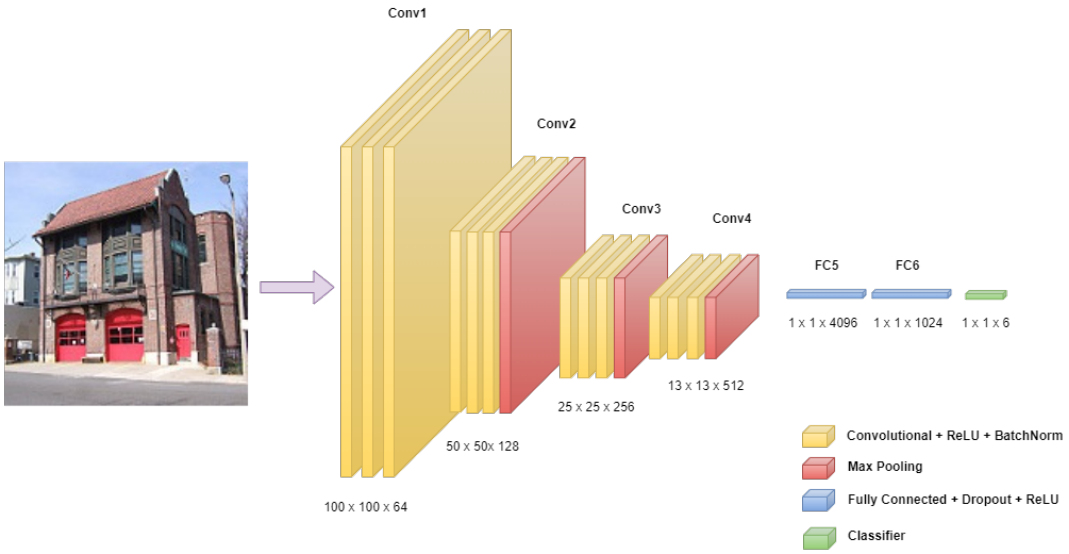

Figure 1: A simple representation of the proposed model.

Initially, it presents four convolutional layers, to process the input volume of the images. The four convolutional layers have respectively 64, 128, 256 and 512 channels. The first layer has a kernel size of 3, a zero padding, followed by a ReLU activation function and by a layer of Batch Normalization.
The second layer has a kernel size of 3, a zero padding with a stride of 1 in each dimension, followed by a ReLU activation function, by a layer of Batch Normalization and by a Maxpool with a kernel size of 2 and a stride of 2. We decided to use the Maxpool because we want to aggregate multiple values into a single value since our object is to reduce the features

map size for faster computations and keeping most important information.
The third layer has a kernel size of 3, a zero padding with a stride of 1 in each dimension, followed by a ReLU activation function, by a layer of Batch Normalization and by a Maxpool with a kernel size of 2 and a stride of 2. The reason why we applied the Maxpool is the same as before.
The forth layer has a kernel size of 3, a zero padding with a stride of 1 in each dimension, followed by a ReLU activation function, by a layer of Batch Normalization and by a Maxpool with a kernel size of 2 and a stride of 2. Also in this case the reason why we applied the Maxpool is the same as before.
The features produced by these convolutional layers, that in our case are indicated by the output that we computed in the code(output of the convolution layers), equal to 18432, are provided as input to the Fully Connected layers. The first fully connected layer has 4096 output features. 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 6, that are the classes of the dataset.

2. Dataset

The dataset is a collection of image data of Natural Scenes around the world. It consists of 17034 images of size 150x150 pixels, distributed under 6 categories. The number of images for each category is different. In the train set we have:

- buildings: 2191
- forest: 2271
- glacier: 2404
- mountain: 2512
- sea: 2274
- street: 2382

All the object in the image acquisition are positioned in various displacements and angles and with different light that does not change significantly the conditions of the images. Example of images from the collected dataset are illustrated in Figure 2.
We decide to resize all images to 100 x 100 pixels because otherwise the Colab program give us some problem of memory.
Then, we passed our images through an augmentation algorithm that consist in a rotation with non-repeated random angles.
The dataset was splitted in train and test but, unfortunally, it does not contain the validation set. Therefore, we decide to build a validation set by taking 10 percent of the images from the train set.
After this procedure, the train set consists of 12034 images, validation consist of 1403 images and the test set consists of 3000 images. All the images, in each class, are named as number.jpg, an example can be "392.jpg" that belongs to the buildings class.
The labels of each images are taken using the ImageFolder function of Pytorch. It takes as

label the name of the folder containing the images.
Since we know that CNNs are not rotation invariant, as we said before, we choose to apply an augmentation that help us to generalize as much as possible our model. Thanks to this technique, the DataLoader does not show the exact same items at every epoch but showing a variant of that.

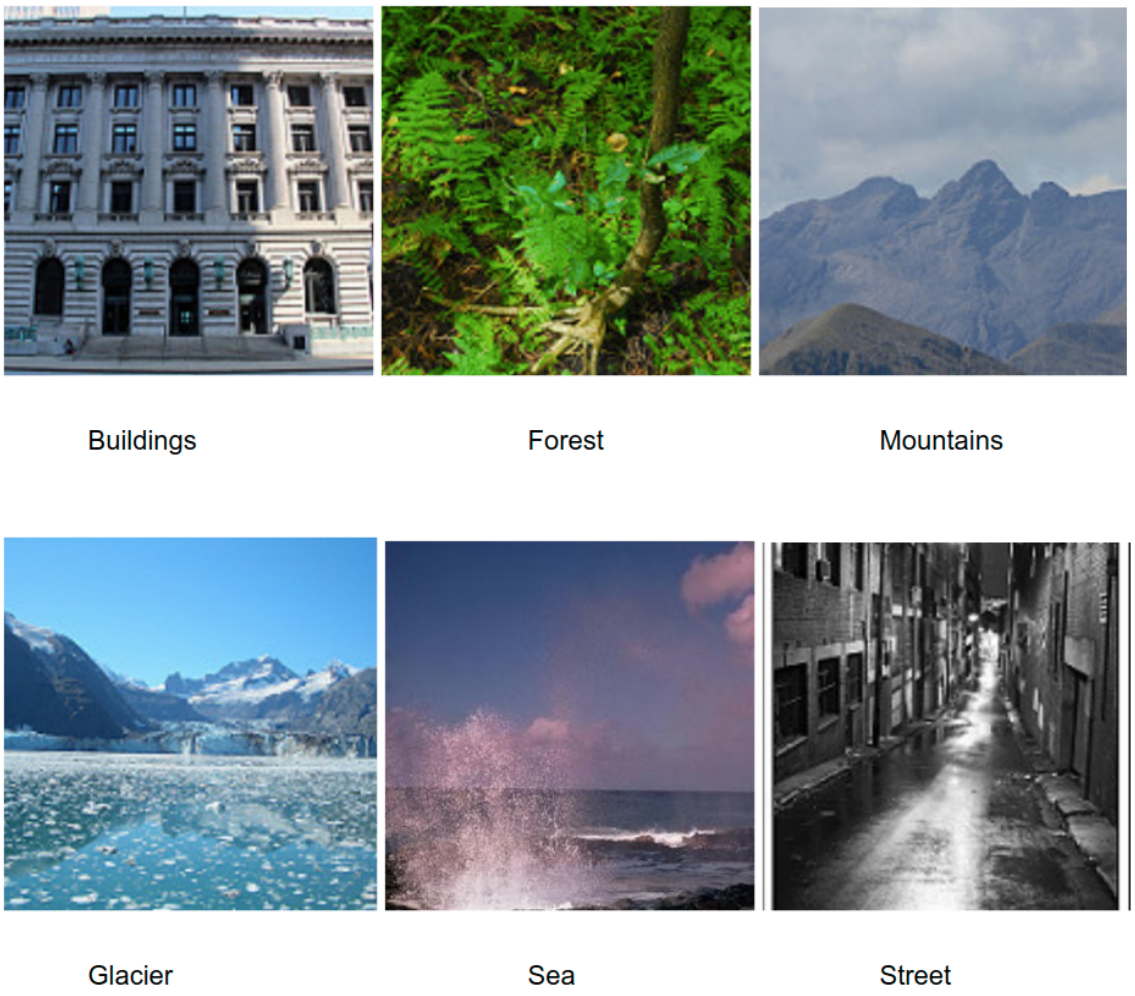

Figure 2: Example images from the collected data.

3. Training procedure

As we said before, we employed data augmentation mechanisms, like rotation, to reduce overfitting issues. Beside the Intel Image Classification model, we test also other 4 models, one baseline model with only one convolution layer and the others three model which were respectively based on a sequence of 2 convolution layers, 3 convolution layers and 5 convolution layers. As training loss, we employed the Cross Entropy Loss, that combines the

Negative Log Likelihood Loss and the LogSoftmax in a single class, therefore the output will be a probability vector which represents predicted probabilities of all classes, whose sum will be always 1. All the models are trained from scratch for 10 epochs. 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. In the baseline model we use a kernel size of 3 applied with a zero padding, followed by a ReLU activation function, by a layer of Batch Normalization and a Maxpool with a kernel size of 2 and a stride of 2. In the model with 2 CNN we use just the first two convolution layer of our Intel Image Classification Model. The same we did for the 3 CNN but this time with the first 3 convolution layers of the Intel Image Classification Model. Instead, in the 5 CNN we adding the fifth convolution layer with a kernel size of 3 a zero padding with a stride of 1 in each dimension, followed by a ReLU activation function, by a layer of Batch Normalization and by a Maxpool with a kernel size of 2 and a stride of 2.

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 Accurancy
Baseline Net	76.55%	75.85%
– + Layer 2	79.54%	79.07%
– + Layer 3	81.31%	81.18%
– + Layer 4	83.81%	83.50 %
– + Layer 5	82.59%	81.97 %
Best model(Intel Image Classification)	83.81%	83.50 %

Table 1: Test performance of the model.

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