Laboratory of Machine Learning - Pet Classification

Francesca Visini (502176), Alice Menna (502172)

This project is about a binary classification task using Convolutional Neural Networks, which is the optimal model to deal with image classification. We leveraged the Oxford Pet Dataset containing 6349 images of cats and dogs, resized to 160x160 pixels and divided in train and test folders. We used the PyTorch library to build our model with the sublibrary torchvision, with the goal to recognize whether an image represents a cat or a dog.

1. Setup and Data

The beginning of this project consists in the import of all the useful libraries that we will leverage and to mount the Drive in which we have uploaded the dataset. Afterwards we created data loaders (one for train and one for test) that access the datasets and extract data efficiently, with the addition of some transformations. Regarding transformations, we converted images into tensors, we randomly flip the images horizontally during training and we normalize the colors around mean and standard deviation. In particular, the tensors will have dimension (3, 160, 160) in which 3 represents the number of channels RGB, 160 the height of the images and 160 the width of the images. We create a visualization representation of the first batch in the train loader, composed by 64 images.

2. Definition of CNN

We split the network into the feature extractor and the classifier. The feature extractor is formed by 5 convolutional blocks and each block resThere are two convolutional layers that apply convolution operations to input data and learn spatial hierarchies of features. The dimension of the convolution filter is 3x3, the padding equal to 1 means that the output layer will keep the same dimension of the input and the stride is 1 for the first convolutional layer and 2 for the second. The stride indicates the movement of the filter, so in the second layer the spatial dimension is reduced.
pectively is formed by 6 layers.

- There is a BatchNormalization layer that improves training stability and convergence by normalizing the inputs of each layer in a mini-batch. It helps to mitigate issues like vanishing/exploding gradients and accelerates the training process
- There are 2 ReLu layers, which represents the activation function for the output layer

The classifier is a sequential model too, whose goal is to assign an image to the right label.

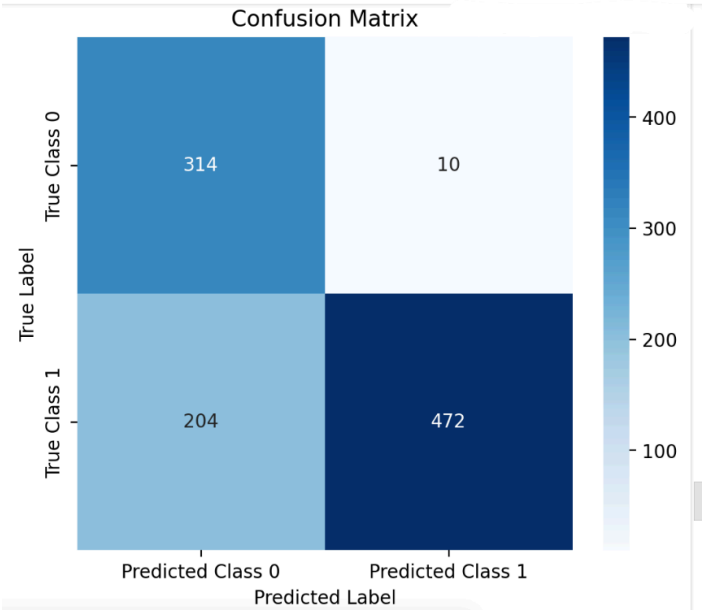
- there is an adaptive average pool layer that reduces the spatial dimensions of the input to 1x1. In particular, as the filter moves, it calculates the average value within the receptive field
- there is a fully connected layer with 256 input features and 1 output feature
- there is a sigmoid layer that computes the probability of an image to belong in the classes

3. Training loop

The loss function for binary classification is the Binary Cross Entropy and our goal is to minimize it. We will use an optimizer which is an algorithm that finds the set of parameters values that results in the best possible fit of the model to the training data. In this case, we have utilized the ADAM algorithm, a variant of the bani mini-batch SGD algorithm.
Afterwards, we manually defined the training loop, noting that we switched to T4 GPU. In each epoch, the model will scan the train data loader, compute the probability of estimates that establishes the output of the classifier, compute the loss function defined before, compute the gradient and then apply ADAM algorithm. We determined also the accuracy at each step,the percentage of correct predictions in the batch. Our results are satisfactory because the loss function decreases and the accuracy increase, with the advancement of the epochs.
For security reasons, we saved the parameters of the classifier and optimizer, at the end of each epoch.

4. Evaluation

First of all we put the classifier in the evaluation mode to oprimize the results.
We scan all the images and labels in the test data loader and computed the total number if correct predictions and used that value to compute accuracy of our model. The accuracy of the model is around 88% and it's acceptable. After representing graphically the results of one batch to see whether the predictions are true or false, we plot the confusion matrix to better understand the misclassification of the classes.
A confusion matrix is a performance evaluation tool used in machine learning that summarizes the performance of classification comparing the predicted labels to the true labels. Each row of the matrix represents the instances of the actual class, while each column represents the instances of the predicted class. The diagonal elements of the matrix show the number of instances that are correctly classified, while the off-diagonal elements represent the misclassified instances.
The results show that dogs are better classified compared to cats.



To better understand the results, we identify the badly classified test images using the following function:

```
import numpy as np

net.eval()
misclassified_examples = []
for images, labels in test_loader:
    images = images.to(DEVICE)
    labels = labels.to(DEVICE)
    with torch.no_grad():
        probs = net(images)
    predictions = (probs > 0.5).long()
    correct += (predictions.squeeze(1) == labels).long().sum().item()
    tot += labels.shape[0]

    misclassified_mask = predictions.squeeze(1) != labels
    misclassified_images = images[misclassified_mask].cpu()
    misclassified_labels = labels[misclassified_mask].cpu()
    misclassified_predictions = predictions.squeeze(1)[misclassified_mask].cpu()

    misclassified_examples.append((misclassified_images, misclassified_labels, misclassified_predictions))

accuracy = 100 * correct / tot
print(f"Accuracy: {accuracy:.1f}%")

# Stampa solo gli esempi classificati erroneamente
for misclassified_batch in misclassified_examples:
    show_batch(*misclassified_batch)
```

Looking at the badly classified images, we understand that that the causes for the errors are:

- Cropped Images where it is not possible to entirely see the animal
- Images where there isn't a close up of the animal and the background is dominant

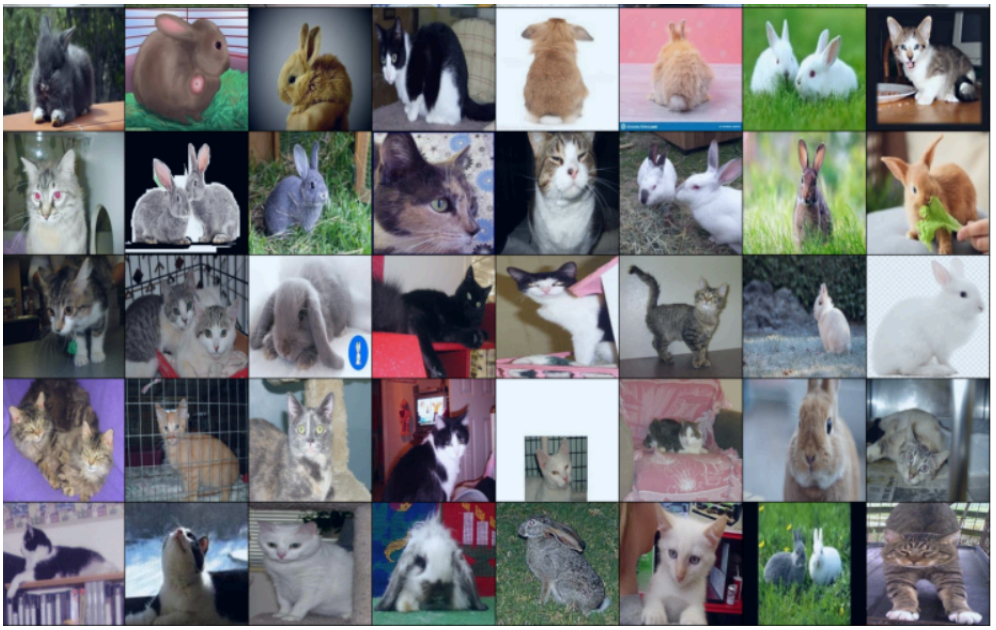
Here below some examples of wrong classification

cropped image

background dominant

Pet Classification - Rabbit vs Cats

In the last part of the code we experimented with another dataset containing images of rabbits and cats.



Our model is able to classify correctly the images with an accuracy of 82% following the same architecture of the cat-dog classification explained before.

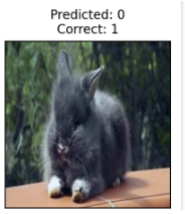
One of the first step taken is the resize of the dataset to a dimension of 160x160 to reduce the computational complexity, with the following code:

```
def resize_images_in_zip(input_zip, new_size):
    with zipfile.ZipFile(input_zip, 'r') as zip_ref:
        temp_dir = "temp_extracted_images"
        zip_ref.extractall(temp_dir)

        for filename in os.listdir(temp_dir):
            if filename.endswith(('.jpg', '.jpeg', '.png', '.gif')):
                img_path = os.path.join(temp_dir, filename)
                img = Image.open(img_path)
                resized_img = img.resize(new_size, Image.ANTIALIAS)
                resized_img.save(img_path)

        new_zip_data = io.BytesIO()
        with zipfile.ZipFile(new_zip_data, 'w') as new_zipfile:
            for filename in os.listdir(temp_dir):
                img_path = os.path.join(temp_dir, filename)
                new_zipfile.write(img_path, arcname=filename)

        with open(input_zip, 'wb') as original_zipfile:
            original_zipfile.write(new_zip_data.getvalue())

        shutil.rmtree(temp_dir)

# Example usage:
input_zipfile = "/content/drive/MyDrive/Colab Notebooks/class.zip"
new_size = (160, 160)  # Set the new width and height here

resize_images_in_zip(input_zipfile, new_size)
```

We decided to experiment with another dataset to understand if the causes of errors are shared among different dataset. In fact, results shows that the wrongly classified images are linked to the same issues identified before:

- Cropped Images where it is not possible to entirely see the animal

- Images where there isn’t a close up of the animal and the background is dominant



5. Conclusions

With this project, we demonstrated that one of the most important step in a classification problem is the correct choice of the dataset.
Providing 2 different dataset and performing the same classification task, the results show that the badly classified images are those where it is difficult to visualize the object to classify.
To avoid this problem, it is necessary to execute a screening of the data and consider only those which are suitable for the classification task.