Assignement 3 - Object Recognition and Computer Vision

Elyas Benyamina
ENS Paris-Saclay
elyas.benyamina@ens-paris-saclay.fr

1. Introduction

We are working on the classifysketch dataset. This dataset contains sketches of 250 different objects. We will use 12.000 pictures for training (48 images for each class), and then test the models we'll train on the validation set which contains 2.250 images (9 for each class).
The size of the images is 1.111 pixels per 1.111 pixels.

2. Dataset transformations

First, we have to work on smaller images, as we have limited computational resources and limited training set. Indeed, we have worked with different sizes between 64 per 64 pixels and 256 per 256. During experiments, a good balance between accuracy and training time seemed to be working with 128 per 128 pixels.
We notice, looking at the images, that the objects are depicted in different orientations. We want to take this into account during training as we can discover new orientations in the validation set.
Further more, a lot of images are oriented to one side (right or left), but we could face a similar image oriented in the opposite direction. That's why, it seems to be accurate to give an image and its horizontal mirror during training.
Here are other transformations we made.

1. **Resize**
2. **Mirror:** It seems to be an important factor, we used two approaches, one is to randomly mirror the images, the other is to add all the mirror images in the train_images directory with XnView software. This second approach makes the learning longer and also may lead to overfitting in some situations, but gave better results depending on the model.
3. **Rotation:** It seems to be an important factor
4. **Affine Transform:** It seems to be an important factor
5. **Random Erase:** Poor results, as we are working with sketches, and most of the information in the images (which is not in the background) is important
6. **Blur:** Same comment

Except from the second mirroring approach, all these operations have been made with PyTorch transforms.compose function.

3. Models used

The basic model implemented in the template gave poor results. We noticed that working with pre-trained model gave far better results. We worked with $ResNet50$ pretrained model first, and we obtained an accuracy in the validation set of 0.65.
We then moved to EfficientNet models on as it seemed to be accurate for image classification tasks.
Let's notice that we used a batch_size of 16 as we doesn't have enough GPU memory to work with larger batch size.

1. **EfficientNet-b0:** We first used this pretained model, and succeed to reach an accuracy of 0.74. To do so, we played with optimizers and fine-tuned the parameters. We'll discuss it after.
2. **EfficientNet-b7:** We also wanted to test a newer version of EfficientNet which is more complex but is known to achive better results.
   During our experiments, we were unable to obtain better results than with EfficientNet-b0.

We have worked especially with two optimizers (we tried other also but we couldn't obtain good performance with them):

1. **SGD:** We used the one given in the template and reduced the momentum to obtain better results.
2. **AdamW:** We used this optimizer with learning rate $2 \cdot 10^{-4}$

4. Improvement suggestions

We can work with bigger images, although training will be longer. We can also manually add images to the training set, adding orientations of the sketches that are missing.
Moreover, we can finetune different parameters, such as weight decay, and work with learning rate (by using schedulers to adjust it over the iterations for example).
We can try to combine different models and choose a prediction (by choosing the the majority class or by summing the outputs)