CentraleSupélec

université PARIS-SACLAY

Machine Learning - Project

Skin lesion image classification

Mr : Stephane HERBIN

M2 ATSI

Chaimae HADROUCH
Kaoutar AIT Ahmad

6 mars 2022

1 Introduction

Our task will be to train a convolutional neural network (CNN) that classify images into 8 classes. Then, instead of building and training a CNN from scratch, we'll use a pre-built and pre-trained model applying transfer learning, finetuning and using Resnet-50 .

2 CNN

The objective of this first method is to build a convolutional neural network (CNN) using PyToch framework.

To make the optimization efficient, we will work with the GPU acceleration provided by Colab. To set the GPU acceleration :

— Edit
— Notebook settings
— Hardware accelerator : GPU

Data :
We will use the *Skin lesion *dataset for image classification. It is composed of 25331 images .

2.1 Create network :

We create a network with 2 convolutions and 3 linear layers.

In the __init__ function, you need to declare the layers with parameters (convolutional and linear layers). In the forward function, you describe the information flow from the input (x) to the final output.

The object with parameters are :

— nn.Conv2d(a,b,c) where a is the input channel number, b the output channel.
— number and c the kernel size.
— nn.Linear(a,b) where a is the input size, b the output size.

Here are some useful functions :

— F.relu(a) : apply a relu on a.
— F.max_pool2d(a,2) : apply a max pooling of size 2 on a.
— b = a.view(a.size(0), -1) : flattens a to be usable with linear layer.

The network will be :

conv (3 -> 16, kernel 5x5)

relu

max_pooling

conv (16 -> 32, kernel 5x5)

relu

max_pooling

Linear (89888 (32x53x53) -> 1200)

relu

Linear (1200 -> 840)

relu

Linear (840 -> 8 (8classes))

Code python :

```
class SimpleCNN(nn.Module):

  def __init__(self):
    super(SimpleCNN, self).__init__()

    # define here the convolutions and linear layers
    self.conv1 = nn.Conv2d(3, 16, 5)
    self.pool = nn.MaxPool2d(2, 2)
    self.conv2 = nn.Conv2d(16, 32, 5)
    self.fc1 = nn.Linear(32*53*53, 1200)
    self.fc2 = nn.Linear(1200, 840)
    self.fc3 = nn.Linear(840, 8)

  def forward(self, x):

    # define here the forward pass
    x = self.pool(F.relu(self.conv1(x)))
    x = self.pool(F.relu(self.conv2(x)))
    x = x.view(-1, 32*53*53)
    x = F.relu(self.fc1(x))
    x = F.relu(self.fc2(x))
    x = self.fc3(x)

    return x
```

2.2 Training and Test :

The algorithm that we followed for training :

— We set the gradients to zero.
— We compute the outputs.
— We compute the cross entropy loss.
— We call backward on the loss.
— We call step on the optimizer.
— We compute the predictions on the outputs (in numpy format), it is the argmax of the prediction vector.
— We compute the predictions on the outputs (in numpy format), it is the argmax of the prediction vector.
— We update the confusion matrix.

The algorithm that we followed for test :

— We compute the outputs.
— We compute the predictions on the outputs (in numpy format), it is the argmax of the prediction vector.
— We update the confusion matrix.

2.3 Displaying accuracies :

We display the training and testing curves :

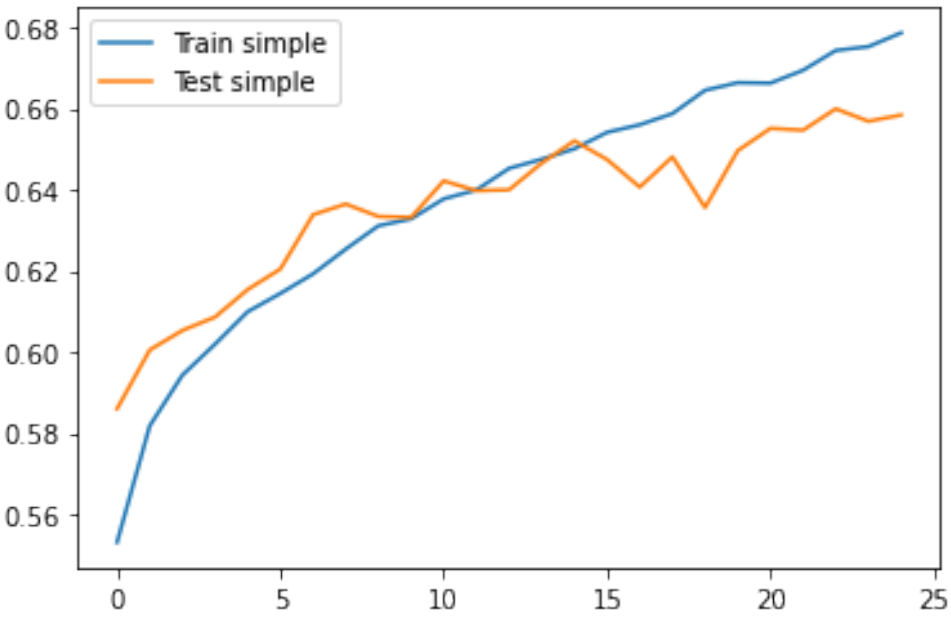

FIGURE 1 – Training evolution CNN

This figure shows that the training and test accuricies are close, in other words, that thay have close values of accuracy at each epoch, and that they are increasing as we increase the number of epochs so, our trainig model is good and it reaches the accuracy of 0.678 for traning and 0.658 for the test .

3 Transfer learning and Finetuning

There are several methods either build a CNN from as we have seen above scratch or adopt an encoder like ResNet or VGG which have several types. Concerning this part, we adopted the ResNet-50 which is a convolutional network architecture giving a very good performances and widely used in the field of computer vision and Deep Learning. It has 48 convolution layers with 1 MaxPool layer and 1 Average Pool layer. It has $3.8x10^9$ floating point operations.

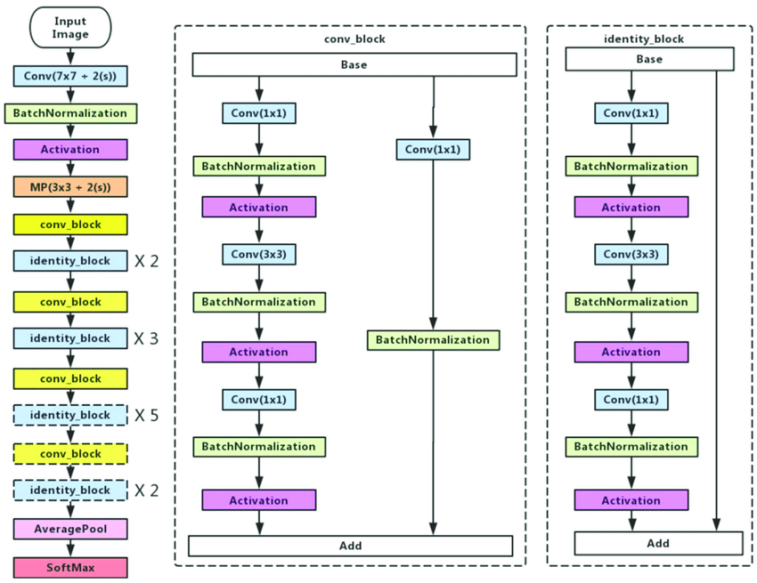

FIGURE 2 – Resnet-50 Architecture

Here we use Resnet-50 for our dataset, which has 8 classes. Like all the models that have been pre-trained on Imagenet, they all have output layers of size 1000. The goal here is to reshape the last layer to have the same number of outputs as the number of classes in our case which is 8 classes :

— Melanoma ("MEL" = 0)
— Melanocytic nevus ("NV" = 1)
— Basal cell carcinoma ("BCC" = 2)
— Actinic keratosis ("AK" = 3)
— Benign keratosis ("BKL" = 4)
— Dermatofibroma ("DF" = 5)
— Vascular lesion ("VASC" = 6)
— Squamous cell carcinoma ("SCC" = 7)

So we have change the output of the last FC layer to 8 using this code :

Code python :

```
import torch
from torchvision import models

model = models.resnet50(pretrained=True)

num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, 8)

model = model.to(device)
```

Then we tune the hyperparameters and optimize them :

Code python :

```
criterion = F.cross_entropy

grad_clip = 0.1
weight_d = 0.00015

num_epochs = 25
lr_m = 0.0007

optimizer = torch.optim.Adam(model.parameters(), lr = lr_m,
weight_decay = weight_d)

lr_scheduler = torch.optim.lr_scheduler.OneCycleLR(optimizer, lr_m,

epochs=num_epochs, steps_per_epoch=len(train_dataloader))
```

Over 25 epochs we obtained :

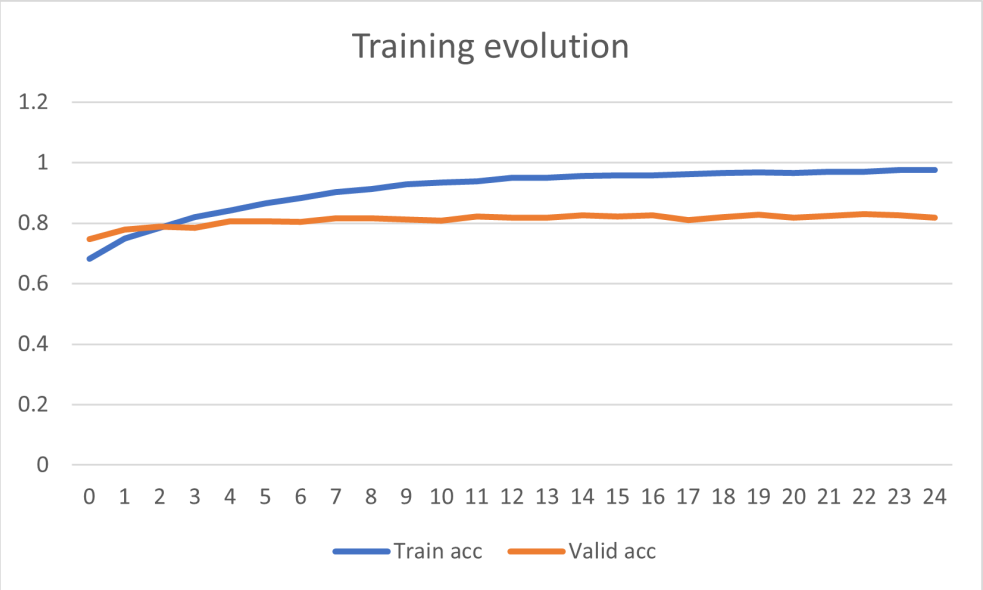

FIGURE 3 – Training evolution Resnet-50

4 Conclusion

We used two types of models : CNN and transfer learning with finetuning. The basic premise of transfer learning is simple : take a model trained on a large dataset and transfer its knowledge to a smaller dataset. For image classification, to make a prediction we calculated the metrics. we couldn't reach a high accuracy (65.8%) with a CNN. However, with Resnet-50 we could reach (83.22%).

Through this project, we were able to see the basics of using PyTorch as well as the concept of transfer learning, an effective method for image classification. Instead of training a model from scratch, we can use existing architectures that have been trained on a large dataset and then tune them for our task. This reduces the time to train and often results in better overall performance.