Kaggle project report

by Felix Foucher de Brandois

Formation ModIA - INSA, 4^{e} année
2023-2024

Contents

List of Figures

1 Introduction

Deep learning models are widely used for image classification. The goal of this project is to compare state-of-the-art models on a subset of ImageNet dataset. Then, we will evaluate the impact of bias in the dataset on the models' predictions. The link of the project is : `https://www.kaggle.com/competitions/modia-ml-2024/overview`.

2 Train state-of-the-art CNN models

The goal is to achieve a good classification accuracy on the test dataset, using Pytorch.

2.1 Data preprocessing and augmentation

Using the given dataset (4000 images), I started by defining basic data transformations that will be applied to the images :

- `Grayscale` : convert the image to grayscale
- `CenterCrop` : crop the image to the center so the images have the same size
- `RandomHorizontalFlip` : randomly flip the image horizontally
- `RandomVerticalFlip` : randomly flip the image vertically
- `RandomRotation` : randomly rotate the image

2.2 Model architecture

I used two different models in this project to compare their performances : LeNet and ResNet.

2.2.1 LeNet

The LeNet architecture is as follows :

- Input layer : 1 channel, 256x256 pixels
- Convolutional layer 1 : 6 filters, kernel size 5x5, stride 1, padding 0
- Max pooling layer 1 : kernel size 2x2, stride 2
- Convolutional layer 2 : 16 filters, kernel size 5x5, stride 1, padding 0
- Max pooling layer 2 : kernel size 2x2, stride 2
- Fully connected layer 1 : 120 neurons
- Fully connected layer 2 : 84 neurons
- Output layer : 4 neurons (number of classes)

The activation function used is the ReLU function.

```
class LeNet(nn.Module):
    def __init__(self, num_channels=1, num_classes=4):
        super(LeNet, self).__init__()
        self.conv1 = nn.Conv2d(num_channels, 6, kernel_size=5, padding=2)
        self.conv2 = nn.Conv2d(6, 16, kernel_size=5)
        self.fc1 = nn.Linear(62*62*16, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, num_classes)

    def forward(self, x):
        x = F.max_pool2d(F.relu(self.conv1(x)), kernel_size=(2, 2), stride=2)
        x = F.max_pool2d(F.relu(self.conv2(x)), kernel_size=(2, 2), stride=2)
        x = nn.Flatten()(x)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

2.2.2 ResNet

The ResNet architecture is as follows :

- Input layer : 1 channel, 256x256 pixels
- Convolutional layer 1 : 64 filters, kernel size 7x7, stride 2, padding 3
- Max pooling layer 1 : kernel size 3x3, stride 2
- Residual block 1 : 3 blocks, 64 filters
- Residual block 2 : 4 blocks, 128 filters
- Residual block 3 : 6 blocks, 256 filters
- Residual block 4 : 3 blocks, 512 filters
- Average pooling layer : kernel size 8x8, stride 1
- Fully connected layer : 4 neurons (number of classes)

To implement the ResNet model, I used the Pytorch library :

```
class ResNet(nn.Module):
    def __init__(self, num_channels=1, num_classes=4):
        super(ResNet, self).__init__()
        self.resnet = torchvision.models.resnet18(pretrained=False)
        self.resnet.conv1 = nn.Conv2d(num_channels, 64, kernel_size=7,
stride=2, padding=3, bias=False)
        self.resnet.fc = nn.Linear(512, num_classes)

    def forward(self, x):
        return self.resnet(x)
```

2.3 Training

I trained the models using the Stochastic Gradient Descent (SGD) optimizer.
I splitted the dataset into a training set (80%) and a validation set (20%).

2.4 Results

2.4.1 LeNet

I didn't explored the hyperparameters of the LeNet model, since the best accuracy I got on the validation set around 50%. However, here are the results :

Train acc	Val acc	Nb of epochs	Batch size	Learning rate	Training time
53.91%	46.29%	10	32	0.01	1m46
42.87%	39.79%	10	32	0.001	1m15
53.77%	42.05%	50	32	0.01	5m23

The following figure shows the evolution of the training loss and validation accuracy over the epochs :

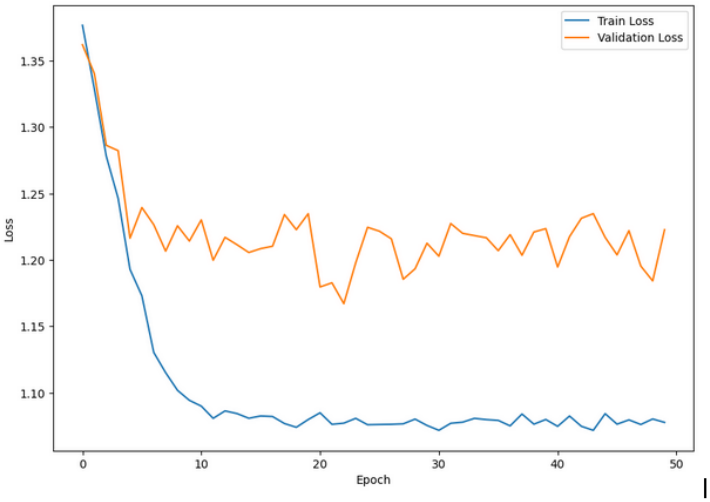

Figure 1: LeNet train and validation loss

We can see that the model is too simple to capture the underlying patterns in the data. The training loss and validation loss are decreasing over the epochs, but the validation accuracy is not increasing. We will therefore use a more complex model, such as ResNet, to improve the accuracy.

2.4.2 ResNet

I explored the hyperparameters of the ResNet model, and I got the best accuracy on the validation set around 80%. Here are the results :

Train acc	Val acc	Nb of epochs	Batch size	Learning rate	Training time
63.64%	54.86%	10	32	0.01	1m31
51.97%	50.89%	10	32	0.001	1m11
63.34%	62.40%	10	64	0.01	53s
72.06%	69.13%	20	32	0.01	8m12

The following figure shows the evolution of the training loss and validation accuracy over the epochs for the best model (80% accuracy) :

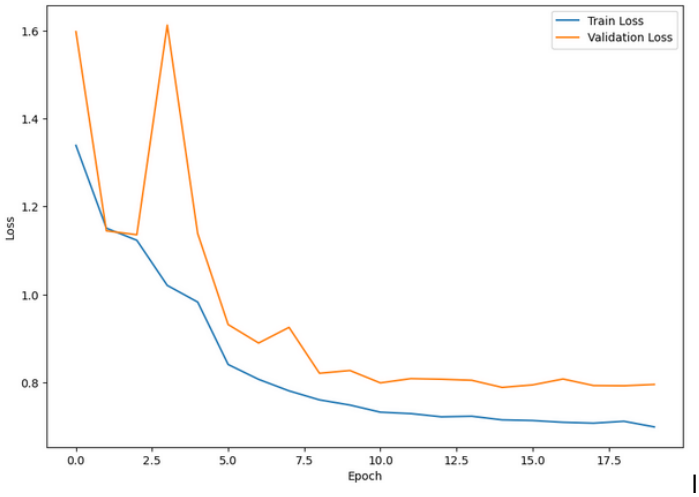

Figure 2: ResNet train and validation loss

We can see that the model is not overfitting, since the training loss and validation loss are decreasing over the epochs.

3 Machine learning and AI safety

3.1 Bias introduction in ImageNet

We try to understand how an image classification model (such as the ones trained in section 2) can be biased towards a specific outcome. To make things easier we consider the binary classification problem (2 classes).
We propose to implement the following setup :

3.1.1 Biased Dataset Construction

We decide to concatenate to each of our images x from ImageNet a new set of variables ϵ. As we shall specify, the second variable ϵ is a noise-like image, which is not directly computed from x. Yet, it introduces some bias as we choose the value of ϵ to be strongly correlated to the label (0 or 1) of the images.

More precisely, we assume $(\tilde{x}, y) = ([x, \epsilon], y)$ is a random sample of the modified dataset. We shall specify the value of ϵ using y. Let $p_0 \in [0, 1], p_1 \in [0, 1]$ be two probabilities. Given $(\tilde{x}, y)$, the bias variable S is defined as

$$S \sim Bernoulli(p_k), \quad \text{if } y = k, k \in \{0, 1\}.$$

Then, we choose ϵ according to S as below :

– $\epsilon = 0$ if $S = 0$.

– $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ if $S = 1$.

In the extreme case where $p_0 = 0$ and $p_1 = 1$, one can ignore the original image x and only use ϵ to predict y. In that case, our predictions are never using the image x and we are only relying on the noise-like ϵ that we introduced to the dataset.

3.1.2 Model Bias Evaluation

We compare two different settings :

– $model_1$ is trained on the unaltered ImageNet dataset

– $model_2$ is trained on the biased version of ImageNet (where we append the biased variables ϵ to the images in a separate channel)

We finally compare the actual bias in both $model_1$ and $model_2$.

Assume $\hat{y}$ is the prediction of a model based on $\tilde{x}$. We shall separate the dataset set into 2 groups, one with $S = 0$, the other with $S = 1$. The bias of this model can be computed from a ratio of these two groups, using the DI metric defined as :

$$DI = \frac{P(\hat{y} = 1|S = 0)}{P(\hat{y} = 1|S = 1)}.$$

In practice, a model is considered unbiased as long as the DI metric is close to 1.

- I compared the bias of the two models using the DI metric.
  - For the unbiased model, I created a dataset with $p_0 = 0.5$ and $p_1 = 0.5$. I got a DI metric of 0.87 which is close to 1, so the model is unbiased.
  - For the biased model, I created a dataset with $p_0 = 0.1$ and $p_1 = 0.9$. I got a DI metric of 0.23 which is far from 1, so the model is biased.

- I also compared the accuracy scores between the two models.
  The accuracy of the biased model is higher than the accuracy of the unbiased model. This is due to the fact that the biased model is using the noise-like ϵ to predict the label, which is strongly correlated to the label.

4 Conclusion

In this project, I trained two different models (LeNet and ResNet) on a subset of ImageNet dataset. I compared their performances and evaluated the impact of bias in the dataset on the models' predictions. The results show that the ResNet model has a better accuracy than the LeNet model.
I also worked on the impact of bias in the dataset on the models' predictions, using the DI metric.