Rapport de TP3-CNN1

Jiaxin XUE

1) **Essayer de faire varier le nombre d'epochs pour améliorer la capacité du réseau à discriminer les différentes catégories d'images. Quel est le nombre d'epochs optimal ?**

   To do the experiment, I create de class Trainer to perform training and validation conveniently.

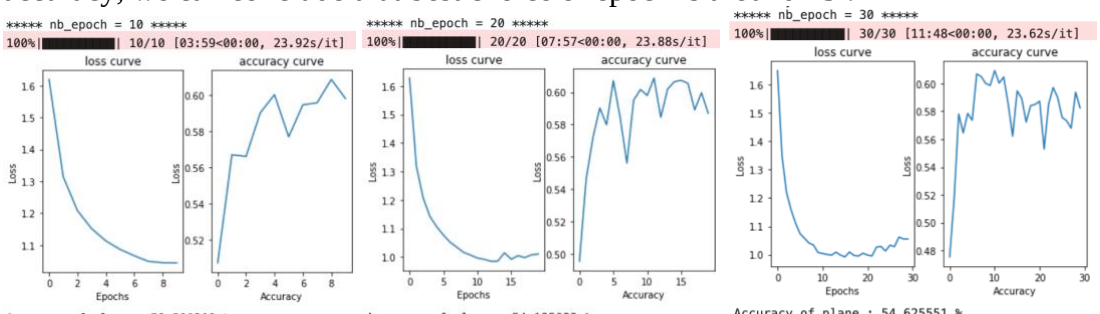
```
class Trainer:
    def __init__(self, nb_epoch, model, criterion, optimizer, device = device
                 ):

    def train_epoch(self, ):

    def valid_epoch(self, ):

    def fit(self, ):
```

   By varying the number of training epoch and drawing training loss and validation accuracy, we can conclude that best choice of epoch is around 15 :



2) **Pour ne pas tomber dans un phénomène de sur-apprentissage, modifier le code donné ci-dessus pour intégrer un ensemble de validation qui permet de déterminer les hyper-paramètres du réseau et notamment un nombre d'epochs adapté.**

   I set up a simple **early stopping mechanism** on validation to control training epoch. (rewrite fit() function of Trainer)

```
***** nb_epoch = 40 *****
 32%|███       | 13/40 [06:15<12:59, 28.86s/it]
Early stopping at the 13th epoch, best epoach = 10
```

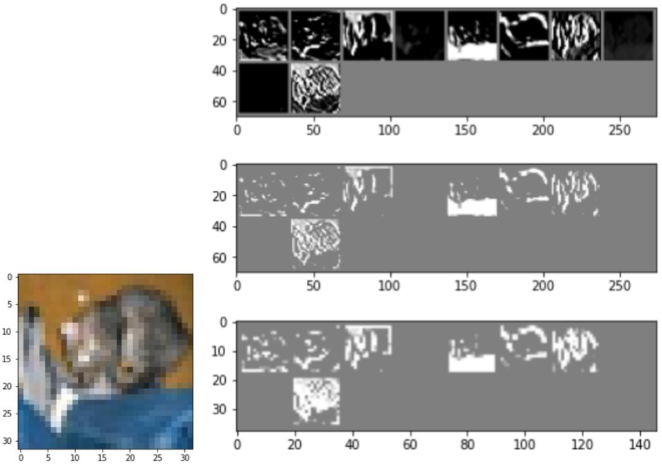
3) **Essayer de modifier l'architecture du réseau pour améliorer le taux de reconnaissance de ce dernier. Vous devez faire attention au nombre de filtres utilisés et à la dimension des données de sortie.**

   I added one more conv layer, resetted channel of other layes, the adapted AdaptiveAvgPool2d at the last conv layer.

4) Proposer une interface permettant de visualiser les sorties des filtres de la première couche.

I used hook on some shallow layers and make grid in PyTorch to achieve. As we can see the conv layer did return some basic forms, relu changed lightness, and pool maked it more blurred.

```
conv_layer_1.0 Conv2d(3, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
conv_layer_1.1 ReLU()
conv_layer_1.2 MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
```



5) Refaire cet exercice en utilisant un autre réseau comme SqueezeNet déjà pré-entrainé sur ImageNet que vous affiner (fine-tuning) sur les classes de CIFAR10.

```python
import torchvision.models as models

class squeeze_pretrained(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = models.squeezenet1_1(pretrained = True)
        self.net.classifier = nn.Sequential(
            nn.Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1)),
            nn.AdaptiveAvgPool2d(output_size=(1, 1)),
            nn.Flatten(),
            nn.Linear(in_features=256, out_features=64),
            nn.ReLU(),
            nn.Linear(in_features=64, out_features=10),
        )

    def forward(self, x):
        out = self.net(x)
        return out
```

```
***** nb_epoch = 20 *****
100%|██████████| 20/20 [19:11<00:00, 57.56s/it]
```