A.A. 2022/2023
Deep Learning and Generative Models
Project assignment #11

Project objective:
- One shot learning with Cifar100 dataset

Dataset:
- cifar100 dataset (https://pytorch.org/vision/stable/generated/torchvision.datasets.CIFAR100.html)

Network model:
- A CNN for image classification can be used for this task. Experiment with a custom one or finetuning a pretrained model (like resnet18)

Detailed information:
- train a classification model on a subset of classes (e.g. 90). Instead of classifying the different classes, train the model to determine if two images have the same class.
- Training such a model can be seen as a binary classification problem (0: different class, 1: same class)
- Then select the remaining classes and take only 1 element for each classes (support set) and the rest as queries. The objective is to classify correctly the queries comparing them with the support set.

Additional notes:
- Learn more about the task here: https://www.analyticsvidhya.com/blog/2021/05/an-introduction-to-few-shot-learning/
- Experiment with splitting the dataset in different ways (90-10, 80-20, 50-50 ecc)

Project specifications

The parallel project was developed using a Siamese network with triplet loss, the reference network is a ResNet:
https://github.com/akamaster/pytorch_resnet_cifar10
The model was trained using a metalearning algorithm called Reptile:
https://openai.com/research/reptile

Metalearning

Metalearning is an approach that aims to develop models capable of learning how to learn. It focuses on the ability to adapt and generalize learning from various experiences.

Reptile is an optimization algorithm used in the context of metalearning. Essentially, Reptile aims to train a model so that it can quickly adapt to new tasks with few gradient updates, leveraging knowledge gained from previous tasks.

Reptile

Reptile is a well-known meta-learning optimization algorithm for its simplicity. Both rely on meta-optimization through gradient descent and are model-agnostic. Reptile seeks an initialization for the parameters of a neural network so that it can be fine-tuned using a small amount of data from a new task.

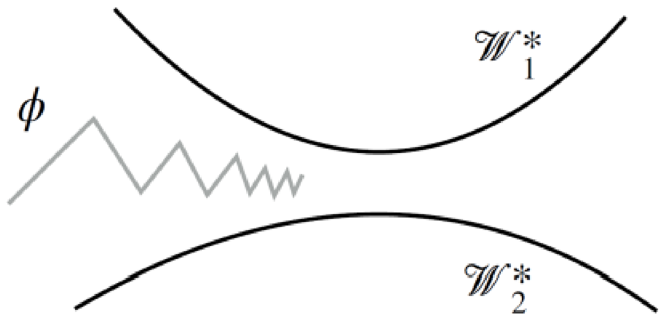
Algorithm 1 Reptile (serial version)

```
Initialize $\phi$, the vector of initial parameters
for iteration $= 1, 2, \ldots$ do
    Sample task $\tau$, corresponding to loss $L_\tau$ on weight vectors $\widetilde{\phi}$
    Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting $k$ steps of SGD or Adam
    Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for
```

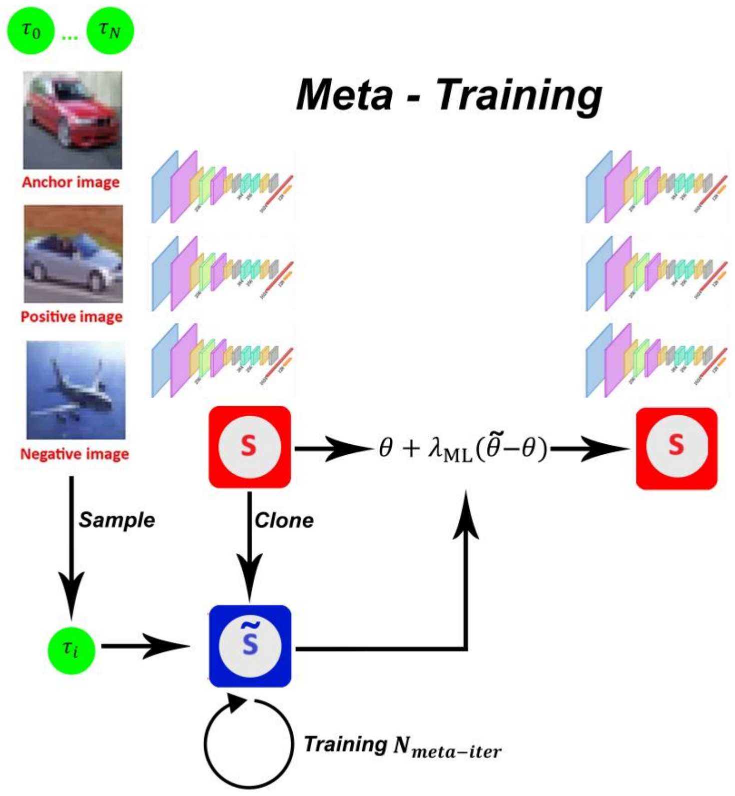
Meta-step size:

$$\begin{cases} \varepsilon_0 & = & 0.01 \\ \varepsilon & = & \varepsilon_0\left(1 - \frac{i}{N}\right) \end{cases}$$

The operation of Reptile can be intuitively explained: each task contains a set of optimal parameters, and for each task, it is reasonable to assume the existence of a set of parameters for which the distance from at least one optimal set is minimal. This particular set is the point at which we want to initialize our network, as it represents the least costly value in terms of effort to reach the optimal set for each task. The purpose of Reptile is precisely to find this set.



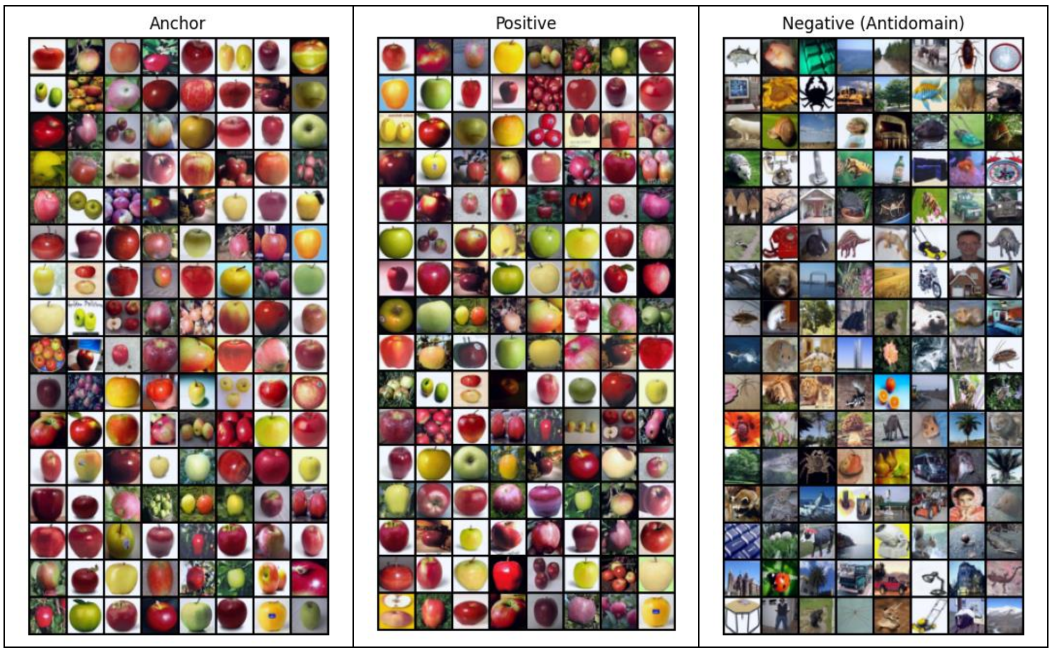
Siamese Network with triplet loss trained with metalearning



The CIFAR100 dataset was divided into 100 tasks, the network was therefore trained on 100 tasks, i.e. 100 domains. As shown in the image below, the network is cloned, trained on a random task and then Reptile's formula is applied on the original network.

The Anchor in this case is the current sampled domain, the Positive is a positive batch of the Anchor and the Negative corresponds to the antidomain, i.e. a batch that contains the other domains.

Example:



Experimental results:

The Siamese network is already natively predisposed to the generalization of similarity and diversity, this project was born from the curiosity to experiment with alternative training with Reptile.

For reasons of time, a single training session was produced in which 90 tasks are shown to the network, and 10 only in the testing phase.

Below is a comparison of the 90-10 meta approach with the classic 90-10.

Hyperparameters

Iterations	10000
Inner-epochs	30
Learning Rate	0.001
Meta-step size	0.01
Batch Size	128

Complete results:

- ***Distance Test Accuracy (triplet):*** Given an anchor, positive and negative, this test is based on how many times the distance between anchor and positive is less than the distance between anchor and negative.
- ***Binary Test Accuracy:*** Given a pair of images, this test measures how many times it was correctly predicted whether the images belong to the same class or not.
- ***Mean Few Shot Accuracy:*** Given a pair of images from a subset excluded during the training phase, this test measures how many times it was correctly predicted whether the images belong to the same class or not.

Training	Test Accuracy (triplet)	Binary Test Accuracy	Mean Few Shot Accuracy
90-10	88.34%	80.95%	83.06%
90-10 meta	73.44%	67.60%	68.80%

Although the model trained with metalearning is less performing, there are some considerations to be made in this regard:

1. Metaiterations may not be enough, the model may need to be trained further.
2. The **threshold** value of binary tests may not be optimal.

Although the overall average is less performing, some unseen tasks were very accurate:

Distance Test Accuracy [TASK: 90]:82.00%	Binary Test Accuracy [TASK: 90]: 68.00%
Distance Test Accuracy [TASK: 91]:68.00%	Binary Test Accuracy [TASK: 91]: 60.00%
Distance Test Accuracy [TASK: 92]:86.00%	**Binary Test Accuracy [TASK: 92]: 80.00%**
Distance Test Accuracy [TASK: 93]:76.00%	Binary Test Accuracy [TASK: 93]: 54.00%
Distance Test Accuracy [TASK: 94]:88.00%	Binary Test Accuracy [TASK: 94]: 64.00%
Distance Test Accuracy [TASK: 95]:96.00%	**Binary Test Accuracy [TASK: 95]: 90.00%**
Distance Test Accuracy [TASK: 96]:76.00%	**Binary Test Accuracy [TASK: 96]: 84.00%**
Distance Test Accuracy [TASK: 97]:72.00%	Binary Test Accuracy [TASK: 97]: 76.00%
Distance Test Accuracy [TASK: 98]:78.00%	Binary Test Accuracy [TASK: 98]: 62.00%
Distance Test Accuracy [TASK: 99]:78.00%	Binary Test Accuracy [TASK: 99]: 50.00%
- Distance accuracy Few Shot: 80.0	- Few Shot Binary accuracy: 68.8

Conclusion

During the experiments, the Siamese Network with triplet loss trained with the classical training paradigm already proved to be a sufficient approach to generalize the knowledge acquired on tasks excluded from the training phase.
Regarding the version trained with Reptile, one thing should be underlined:
the value of Distance accuracy Few Shot: (80.0) and the Few Shot Binary accuracy: (68.8) are closely related values, a better threshold can certainly make the Binary Accuracy tend towards 80, which would make the performance drop not so clear.
In conclusion, more thorough training, the calculation of an optimal threshold (or optimal thresholds) can certainly improve the data.