TransBoost Project

Luc Blassel, Romain Gautron, Xue Bai, Yifei Fan, Xuefan Xu

20 novembre 2018

Table des matières

1 Context

Classical supervised learning requires a large amount of labeled data and, in certain cases, a very long amount of time to ba able to train reliable models. This is not always possible in a 'real world' setting, it could be that it is not possible to obtain sufficient data to train models or that the required time for training is so long that it becomes practically unfeasible with conventional resources. Transfer learning can help us solve this type of problem.

Transfer learning allows for a transfer of acquired knowledge from a source domain, ideally with a large amount of well labeled data, towards a target domain. With this approach, portions of a pre-trained model can be reused in a new model. The advantage is two-fold : training times are shortened and the amount of training data needed is smaller. This method is considered by some as the next motor of progress in automatic learning after supervised learning.

The TransBoost method [2], proposed by Antoine Cornuéjols et al., outlines a different approach than "classical" transfer learning. Where the latter adapts the hypothesis learned on the source space, to the target space, TransBoost does the opposite. The hypothesis is learned on the source space and the data points from the target space are projected in the source space to make use of the source hypothesis without the need to relearn it.

This way no new frontiers between points are learned, the target points are correctly projected at the right location in the source domain. This projection is done within a boosting algorithm, allowing for the building of a strong projector by combining several weak projectors. This difference in approaching the issue can be seen in Figure 1.

The TransBoost method has been tested on classification of incomplete time series data with real success, outperforming other methods used for this problem. However, image classification being the standard measure right now, this project intends to adapt the TransBoost method toimage classification unsing deep convolutionnal neural networks (CNN). In this project only binary classification is tried.

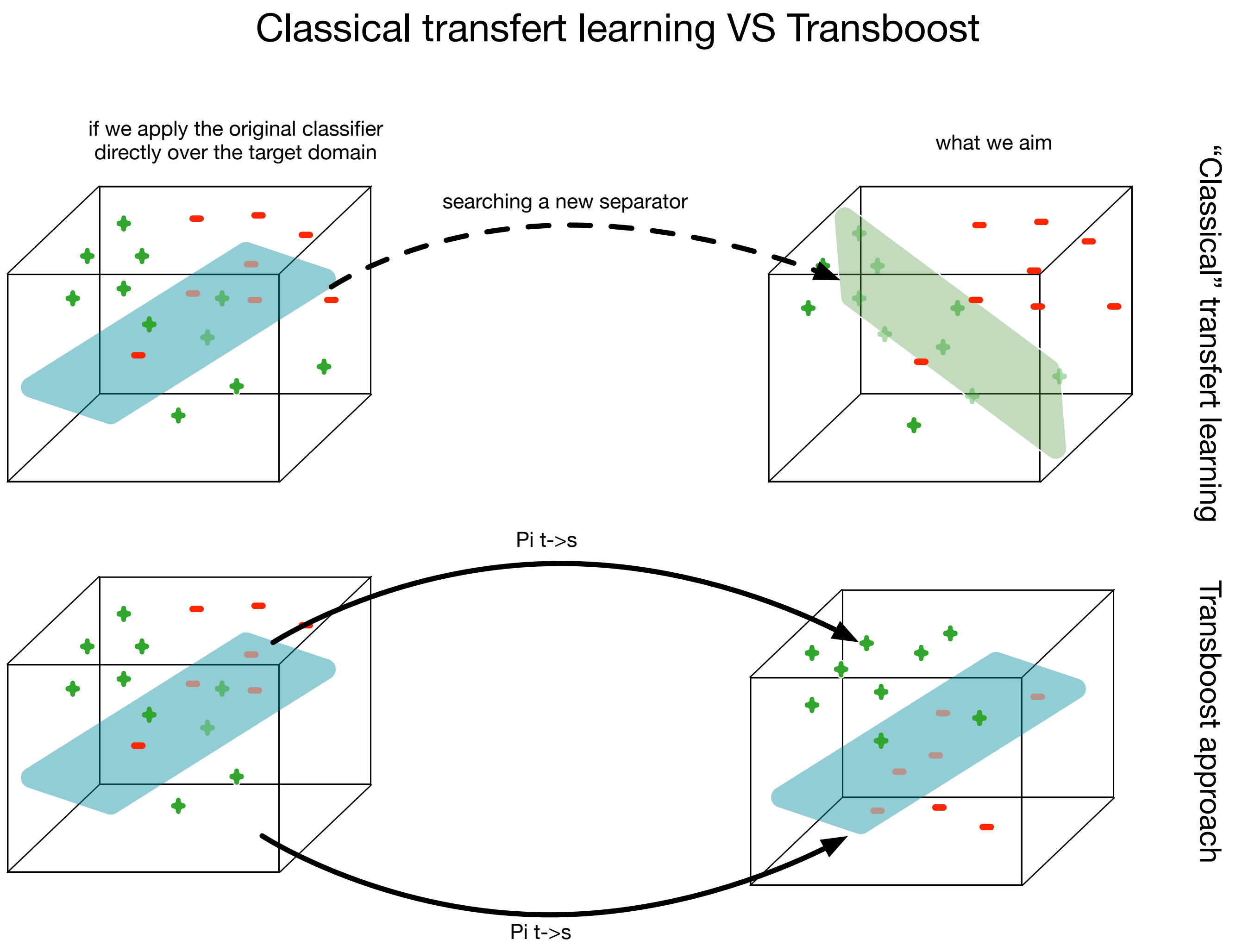
Classical transfert learning VS Transboost
if we apply the original classifier directly over the target domain
searching a new separator
what we aim
"Classical" transfert learning
Pi t->s
Pi t->s
Transboost approach

FIGURE 1 – Differences between "classical" transfer learning and the TransBoost method

2 Transboost for image classification : working principles

Implementation of the TransBoost method for image classification requires consideration of several issues. First, the high dimensionality of image data induces the use of very heavy methods sur as CNNs. Second, how to project points of source to target domains in the case of images ?

The choice has been made, to modify the lower layers of the source network to classify target images. Therefore, the lower layers of the existing network will find the right low-level descriptors so that the new task can be done. However hyper-parameter optimization will have to be done.

We could also have implemented a separate "projector" network that would take the source domain images and transform them in projected images. Howver, even though potentially visually interesting, it would have required larger computing power (more back-propagation) and was not implemented.

The construction of a projector is as follows :

— A pre-trained CNN is obtained. It is very performant for one source domain binary classification task and random for the target domain task. Since pre-trained models are available with a large number of outputs (general image classifiers can recognise hundreds of classes), it is necessary to modify the last layer and retrain it to have a binary output.
— The higher layers of the modified CNN are frozen, leaving only the lower layers trainable
— This partially frozen CNN is retrained to obtain a weak projector, using the precision metric as a stopping criterion. Precision has been chosen because it is meaningful and pertinent for well-balanced data-sets.

A set of weak projectors, specialised on the error of the previous projector, is therefore iteratively built using the AdaBoost algorithm [3]. Our final hypothesis on the target domain in a linear combination of these weak projectors.

3 Application

In this section, Figure 2 wil be used as reference.

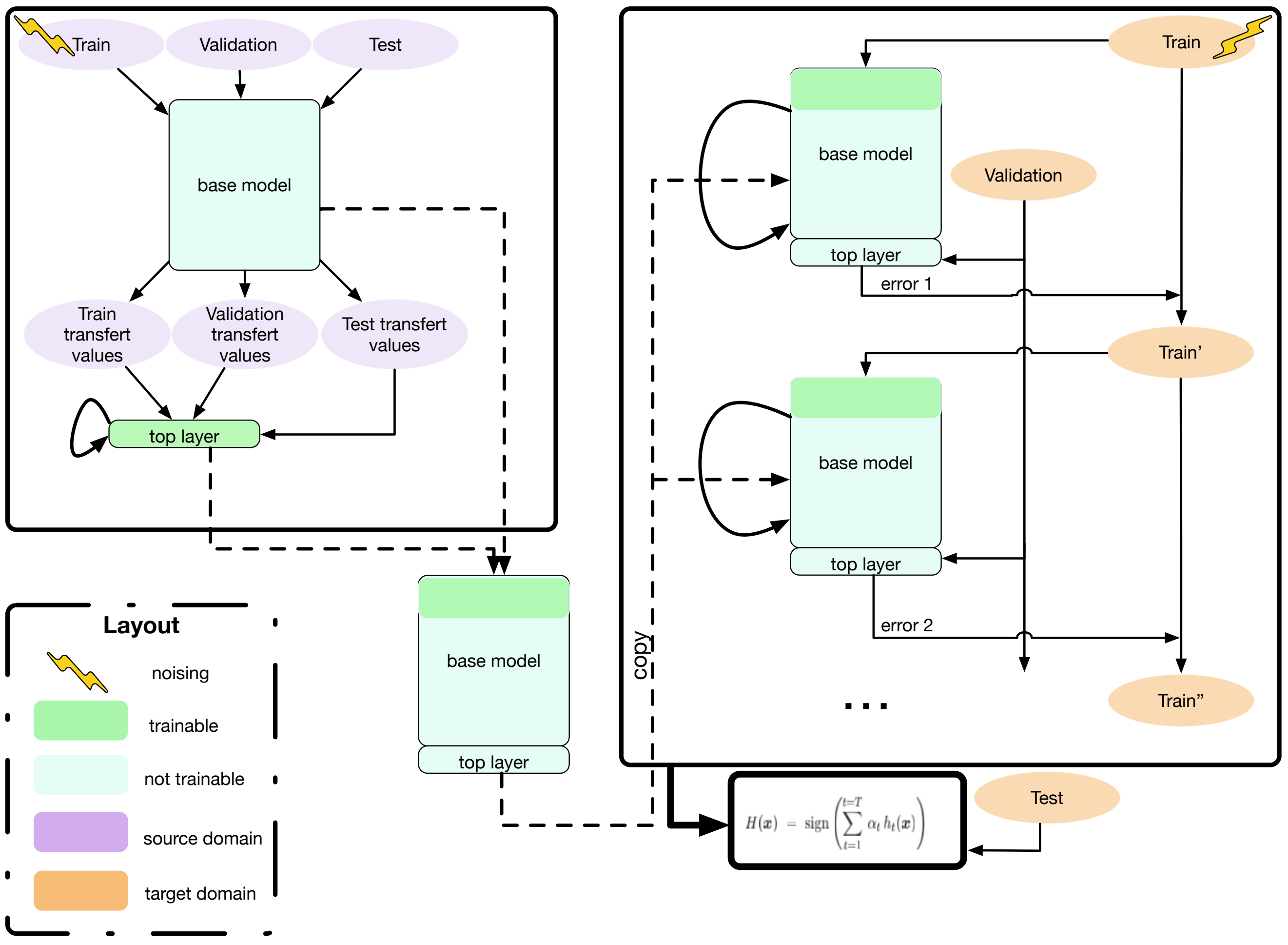

FIGURE 2 – Execution steps of the TransBoost method, within this project

3.1 Our Data

We have chosen the CIFAR-10 image dataset. It comtains 60000 RGB images 32 by 32 pixles. Thes images are separated in 10 classes (airplane, car, bird, cat, deer, dog, frog, horse, ship and truck). The reasons for this choice are as follows :

- It is a reference dataset in image classification.
- The small size of the images limits the size of data to manipulate
- in spite of th esmall number of classes it is possible to make pairs of varying differentiability (plane/cat vs deer/horse)

Data is separated into 3 sets : training, validation and test. To counteract the low number of images in the training set, images are noised to avoid over-fitting. This noise consists of rotation, zooming and random deformations. No noisingis done for the test and validation sets. We want images in these sets to be as reprensatative as possible of "real world" images to test the model.

The target and source domains are just two different binary image sets *(eg. source = ['dog', 'truck'] and target = ['deer', 'horse'])*. This can be easily controlled by input parameters. It is important to note that noising of training data is applied to both domains.

3.2 Elaboration of strong binary classifier on the source domain

To be able to apply TransBoost, it is necessary to have a strong binary classifier for one task that is very weak for another task. To obtain sych a model a base model is chosen, ie. a pre-trained model without the last layers (Dense layer and softmax layer). The `Keras` package offers a large choice of pre-trained CNNs (see this page). The choice has been made to use the Xception model. On the one hand it produces good transfer values and on the other hand the time necesary to generate these transfer values is low. Transfer values are the values of the activation function for all the neurons of the last layer of the considered network for a given set of images.

To keep computing times low, the transfer values, for all the images of our datasets, are computed only once and used to train the last layer.

The architecture of the last layers are :

- Dense layer, of size 1024, "relu" activation function
- 50% dropout
- Dense layer, of size 512, "relu" activation function
- 50% dropout
- Dense layer, of size 1, "sigmoïd" activation function

This topology was chosen empirically.

Just like the noising of input data, the dropout layers are there to prevent overfitting that could occur due to the low number of training inputs. After the last layer is trained to our desired output, the base model is reassembled from the frozen layer and the last layers to create a complete network.

3.3 Application of TransBoost

Once our strong classifier is trained, the next step is just the application of the AdaBoost algorithm. The slight nuance is that instead of building a new classifier "from scratch" we have a partially trained classifier as our weak projector.

The do not want to relearn a separator function by retraining the last layer(s), as would be done in classical transfer learning with CNNs. Instead we seek to project the input points on the correct "side" of the separator function by retraining the first layers of the deep neural network.

To do so we unfreeze the first layers and freeze the weight of all the other layers, so they become untrainable. At this point two separate approaches are possible : either reinitialize the weights of the unfrozen layers or keep these weights. It is possible that by reinitializing the weights will stop the model from converging with so little training data. The choice will be made on experimental results. In both cases, the resulting model will be used as the weak projector / classifier in the boosting steps.

There are three important parameters to optimize while trying to keep a good baance between performance and computing cost :

— The power of the weak classifiers (precision threshold)

— The number of convolution blocs to train

— The number of weak models to train

The next steps are just the AdaBoost algorithm *(cf. Algorithm 1)* At each step, training of a projector stops when the precision threshold is reached. A limit on training epochs is set in case of non convergence.

```
Algorithm 1: TransBoost Algorithm
  input  : $\mathcal{X}_\mathcal{S} \to \mathcal{Y}_\mathcal{S}$ : source hypothesis
           $\mathcal{S}_\mathcal{T} = \{(\mathcal{X}_\mathcal{I}^\mathcal{T}, \mathcal{Y}_\mathcal{I}^\mathcal{T})\}_{1 \leq i \leq m}$ : target training set
  Initialisation : distribution of training set :$D_1(i) = 1/m \ for \ i = 1, \ldots, m$ ;
  for $n = 1, \ldots, N$ do
      Find a projection $\pi_i : \mathcal{X}_\mathcal{T} \to \mathcal{X}_\mathcal{S}$ tq. $h_S(\pi_i(.))$ That is better than random on $D_n(\mathcal{S}_\mathcal{T})$ ;
      Let $\epsilon_n$ be the error rate $h_S(\pi_i(.))$ on $D_n(\mathcal{S}_\mathcal{T})$ : $\epsilon_n = P_{i \ D_n}[h_S(\pi_n(x_i)) \neq y_i]$(with $\epsilon_n < 0.5$);
      Compute $\alpha_i = \frac{1}{2}\log_2(\frac{1-\epsilon_i}{\epsilon_i})$ ;
      Update : for $i = 1, \ldots, m$ do
          $$D_{n+1}(i) = \frac{D_n(i)}{Z_n} \times \begin{cases} e^{-\alpha_n} \ si \ h_\mathcal{S}(\pi_n(x_i^\mathcal{T})) = y_i^\mathcal{T} \\ e^{\alpha_n} \ si \ h_\mathcal{S}(\pi_n(x_i^\mathcal{T})) \neq y_i^\mathcal{T} \end{cases} = \frac{D_n(i)\exp(-\alpha_n y_i^{(\mathcal{T})} h_\mathcal{S}(\pi_n(x^{(\mathcal{T})})))}{Z_n}$$
          Where $Z_n$ is a normalization factor such as $D_{n+1}$ is a distribution of $\mathcal{S}_\mathcal{T}$ ;
      end
  end
  output: The final hypothesis : $H_\mathcal{T} : \mathcal{X}_\mathcal{T} \to \mathcal{Y}_\mathcal{T}$ :
           $H_\mathcal{T}(x_\mathcal{T}) = sign\{\sum_{n=1}^{N} \alpha_n h_\mathcal{S}(\pi_n(x^\mathcal{T}))\}$
```

3.4 Difficulties

We initially made the decisio of working in a pure `Tensor Flow` environnment, since the model we had chosen was available in this package. However when we had to modify the structure of the network (to have a binary output for example), or when certain layers had to be frozen using `Tensor Flow` became quite complicated. The actual model object was not found so we had to directly modify the graph which caused us some problems. For this reason we decided to use the `Keras` package that has a `Tensor Flow` backend, allowing us to use pre-trained models from the `Tensor Flow` library, but with a much clearer syntax and easier to use tools, allowing us to bring down the development times.

We also encountered computing power issues, since the virtual machines we had access to were limited in resources. This meant dozens of hours of computing time for each run, some times days and ending some times in memory errors. It is therefore necessary to have a powerful machine that could allow us to debug and develop the program without long delays.

4 Critical comparison for the TransBoost method

4.1 Weak vs. strong projectors

We want to compare the Transboost approach of boosting weak projectors to the training of one very good projector. If the performance is higher, with shorter training times, in the TransBoost approahch, then we can prove a real advantage of this method.

4.2 Projectors vs. new classifiers

The transboost method requires quite a lot of memory and computing time. Indeed after having trained a big number of classifiers it is necessary to store them so as to be able to reuse them for class prediction of testing points. The space needed can be lowered by storing the architecture of the base model and only the weights of the modified layers at each boosting step, and reconstructing each boosting model when we need to predict a class.

We want to compare the TransBoost method with a classical boosting method.

It is possible to reach a low precision score (0.7) at each step quite easily with a simple CNN (only a few convolution blocks). This allows for shorter computing times.

5 Results

5.1 Building the binary classifier

Dans les faits, pour la base d'apprentissage “chien/camion”, l'algorithme arrive à 98.9% de précision sur le set de test en seulement 2 epochs. D'un autre côté ce même modèle entraîné sur le dataset précédent a un précision de 50% en prédiction sur le dataset “deer/horse”, ce qui répond bien à ce que l'on recherchait.

5.2 Boosting classique sur petit CNN

Avec un petit nombre de projecteurs (12) et un petit CNN (3 blocs de convolution) on arrive à une précision de 76.8% en test sur le dataset “horse/deer”. Expérience à renouveler avec un 100 projecteurs dès qu'on aura la puissance de calcul ad hoc.

6 Experimentations a venir

6.1 Comportement de l'algorithme Transboost

6.1.1 Influence des domaines source et cible

Peut t-on partir d'un domaine source simple (ex : “dog/truck”) pour aller vers un domaine cible plus compliqué (ex :”deer/horse”) ?

Bien sur dans les temps venant il faudra tester les critiques soulevées plus haut, en comparant les performances des projecteurs faibles contre un projecteur fort, et la performance de ces projecteurs par rapport à celle d'un classifieur nouvellement entraîné.

6.1.2 Influence des hyper-paramètres : recherche d'un optimum précision/coût calculatoire

Influence de la force des projecteurs

Influence du nombre de projecteurs

Influence du nombre de blocs entraînés Dans l'idéal on veut en modifier le moins possible pour atteindre au plus vite le seuil de précision

6.2 Comparaisons critiques de l'algorithme Transboost

6.2.1 Projecteurs faibles vs. projecteurs forts

6.2.2 Projecteurs vs. nouveaux classifieurs

Bibliographie

[1] Francois Chollet. Building powerful image classification models using very little data, 2016.

[2] A. Cornuejols, S. Akkoyunlu, P.A. Murena, and R. Olivier. Transfer learning by boosting projections from the target domain to the source domain.

[3] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55 :119–139, 1997.

Appendices

A Code

Code sur gitlab.

B Schéma du programme

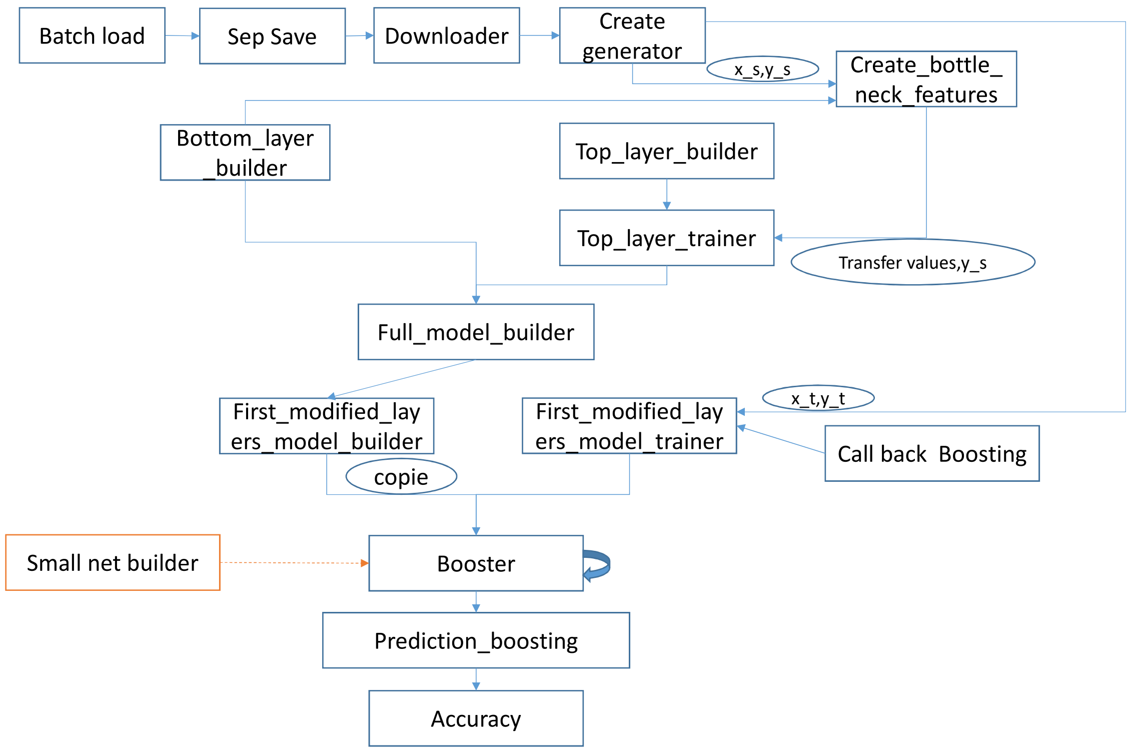

FIGURE 3 – schema du programme

Descriptif des fonctions :

Batch_loader : Prend les images d'un batch de CIFAR-10 et les enregistre dans un dictionnaire avec comme clé la classe de l'image.

Sep_saver : Parcourt le dictionnaire et enregistre les images de chaque classe dans un dossier correspondant.

Downloader : Télécharge les batches de CIFAR-10 et exécute les deux fonctions précédentes pour chaque batch d'entrainement, de test et de validation.

Create_generator :

Bottom_layer_builder : Charge et construit le modèle sans derniere couche (Xception dans ce cas).

Create_bottleneck_features : Calcule les valeurs de transfert pour chaque image des jeux d'entrainement, de test et de validation passant par le modèle issu de la fonction précédente.

Top_layer_builder : Construit la couche de sortie binaire.

Top_layer_trainer : Entraine la dernière couche avec les valeurs de transfert.

Full_model_builder : Assemble le modèle sans derniere couche et la dernière couche nouvellement entraînée.

First_modified_layers_model_builder : Gèle les poids des n dernières couches pour effectuer le boosting.

First_modified_layers_model_trainer : Entraine le modèle partiellement gelé sur le jeu d'entrainement.

Callback_boosting : est appelé à chaque epoch de l'entrainement du modèle et si la précision est supérieure à un seuil α arrête l'entraînement.

Booster : Applique l'algorithme adaboost au réseau. chaque modèle entraîne faisant office de projecteur faible est construit puis entraîne en appelant les deux fonctions précédentes n fois.

Prediction_boosting : Effectue un e prédiction sur la classe de l'image présentée en prenant en compte les modèles entraînés dans la fonction précédente.

Accuracy : Évalue la précision des prédictions.