Projet Transboost

Luc Blassel, Romain Gautron

22 Mars 2018

Le projet Transboost

- Transfer learning $\rightarrow$ réutiliser des modèles entraînés
- boosting $\rightarrow$ faire des prédictions fortes a partir de plusieurs prédicteurs faibles
- associer les deux

Succès dans la classification de séries temporelles incomplètes.
Est-ce aussi efficace pour de la classification d'images avec des CNN ?

Rappel sur le boosting

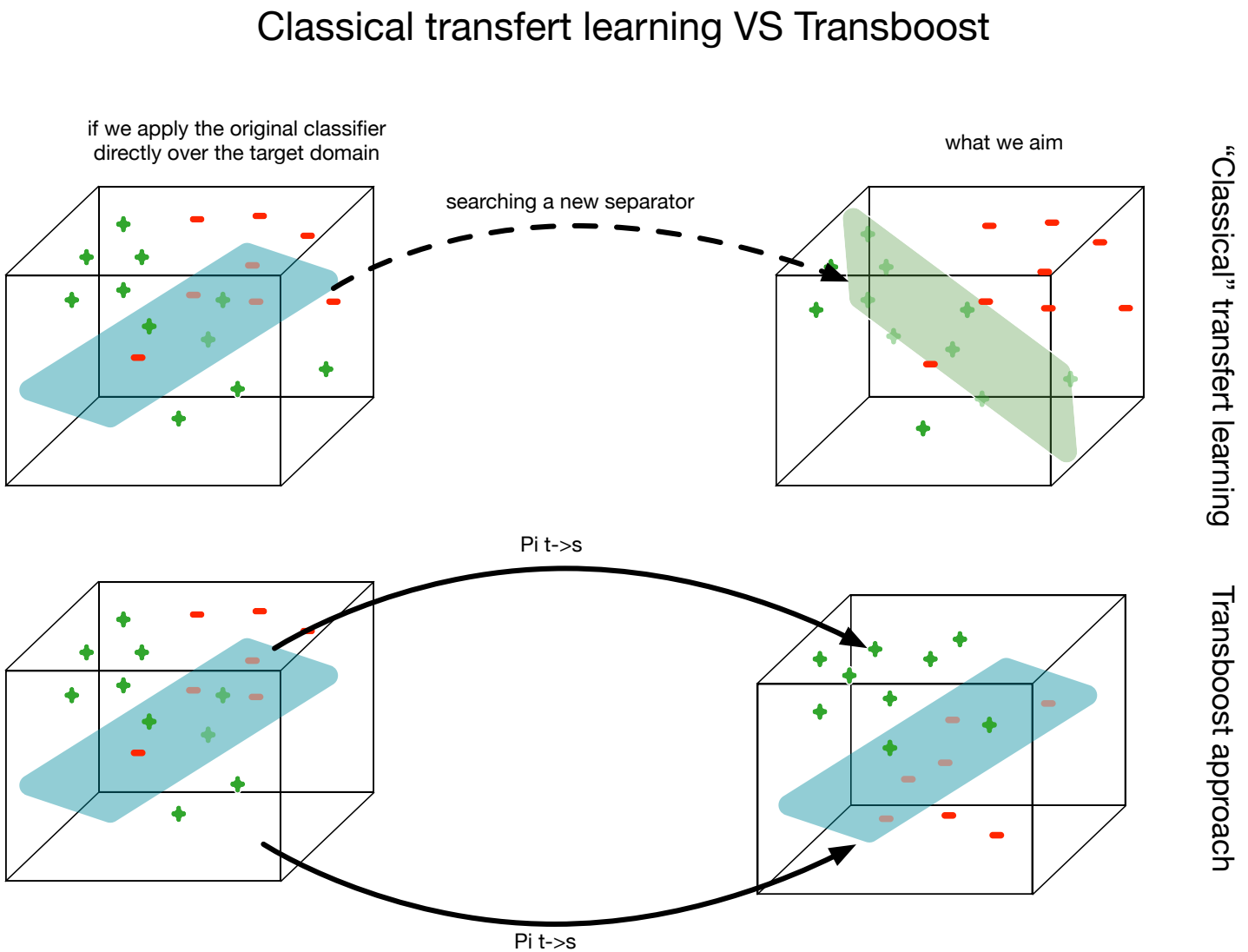
On part d'un ensemble de points pondérés :

1. on entraîne un classifieur faible
2. on sur-pondère les points mal classés
3. on sous-pondère les points bien classés
4. on calcul le poids α_i du classifieur a partir de son erreur.

Résultat :

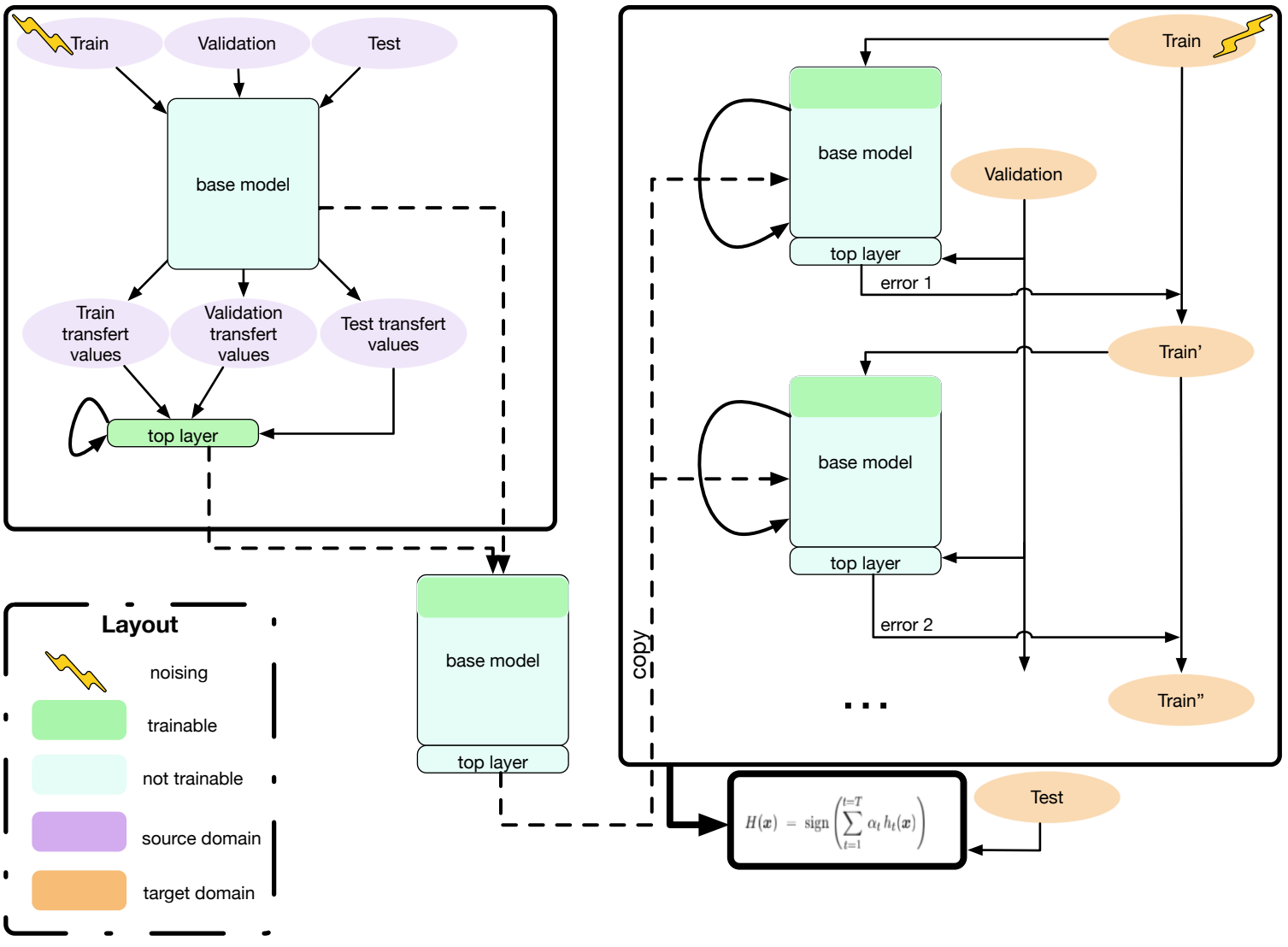
On fait la combinaison linéaire des n classifieurs faibles pondérés avec les α_i
$\rightarrow$ on obtient un classifieur fort

Le principe derrière Transboost



projection hamster $\rightarrow$ lion

Transboost et réseaux de neurones



CIFAR 10

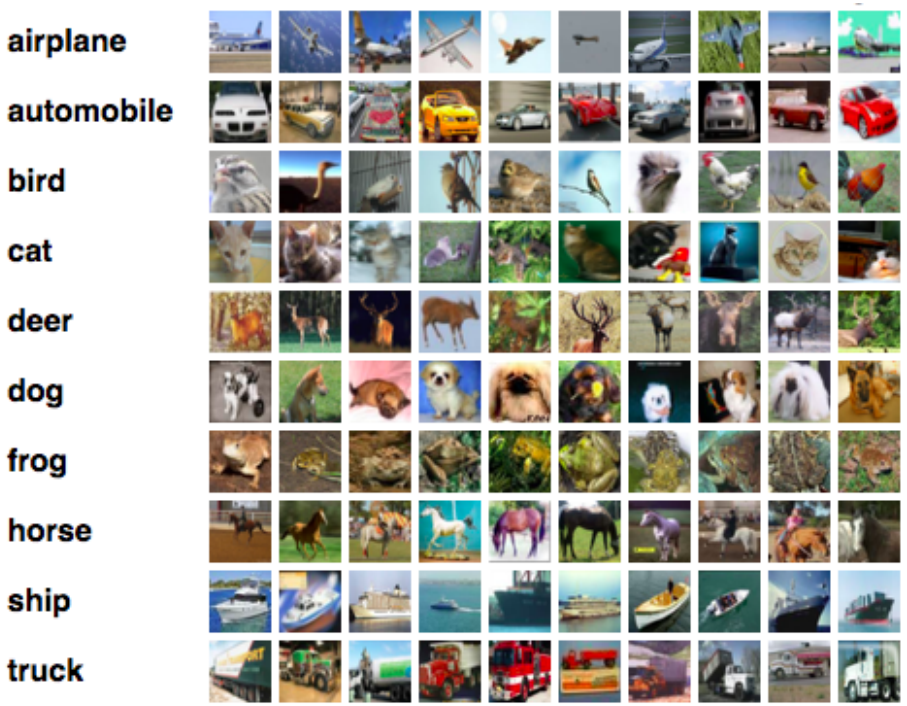


Figure 1 – CIFAR 10 dataset

- 60 000 images : 50 000 train + 10 000 test

Xception

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

Figure 2 – Comparaison des modèles de bases sur ImageNet

Sans fine tuning

- 98.9 % de précision sur dog/truck
- 92.2 % sur deer/horse

La machine

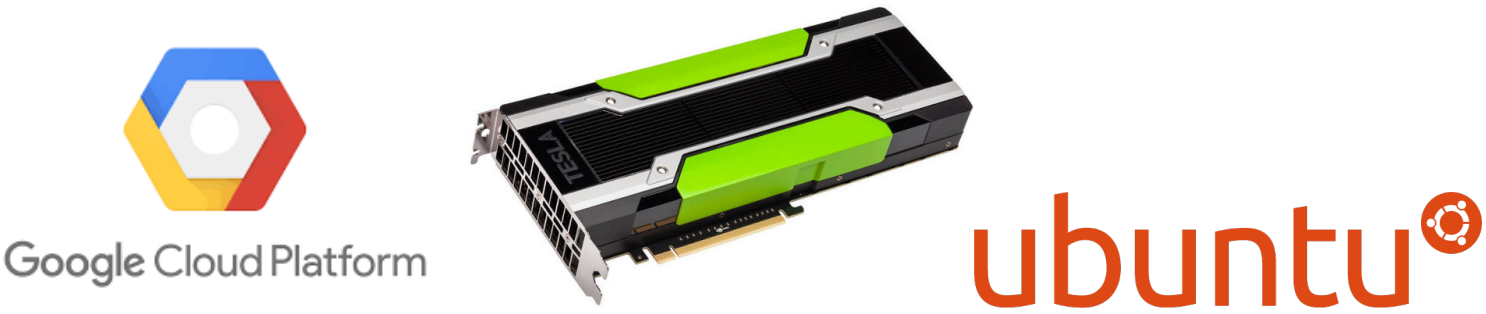


Figure 3 – Setup

- 8 coeurs
- 30 Go de RAM
- 12 Go de GPU

Fonctionnement du programme (1)

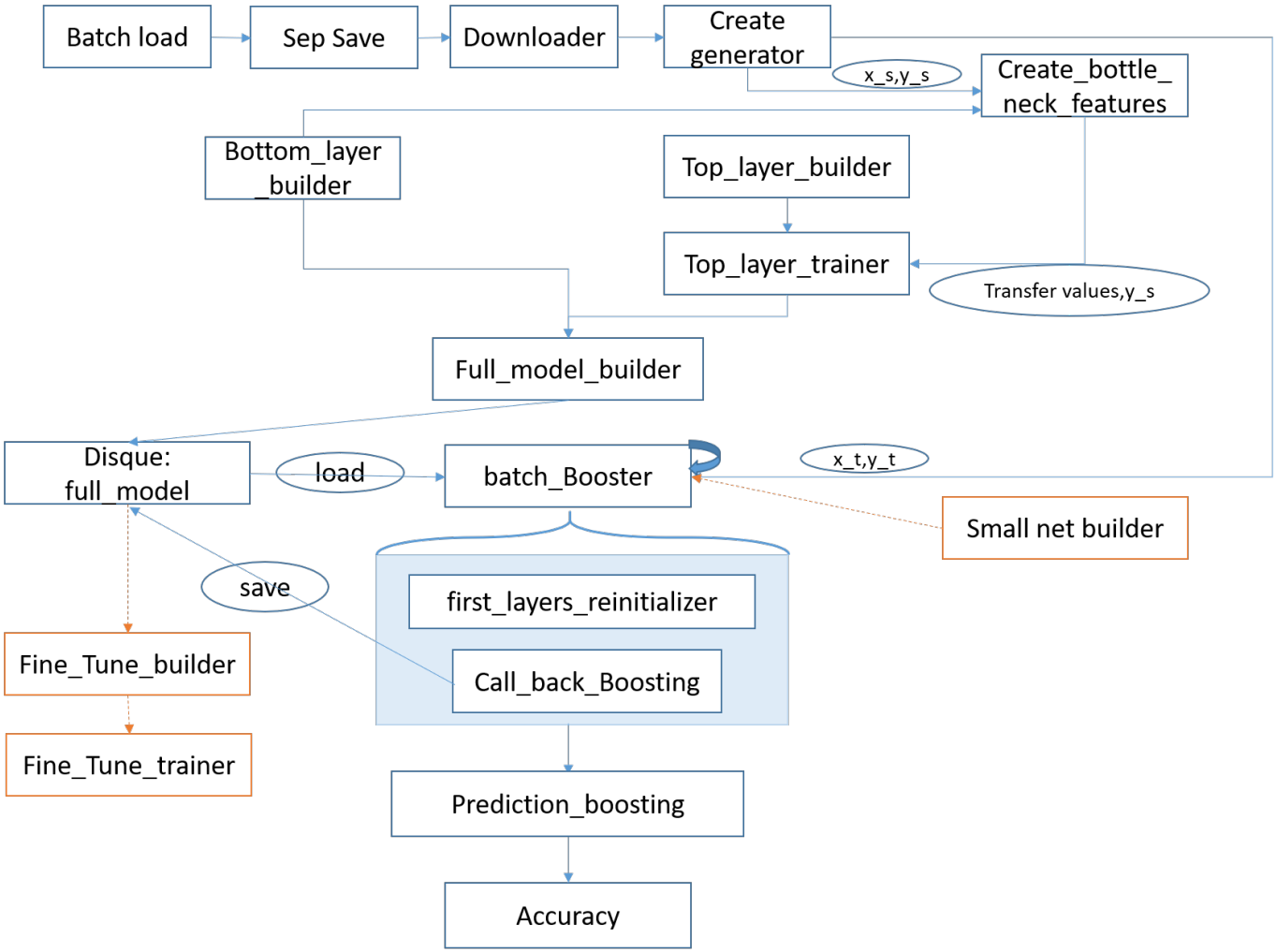


Figure 4 – Programme

Fonctionnement du programme (2)

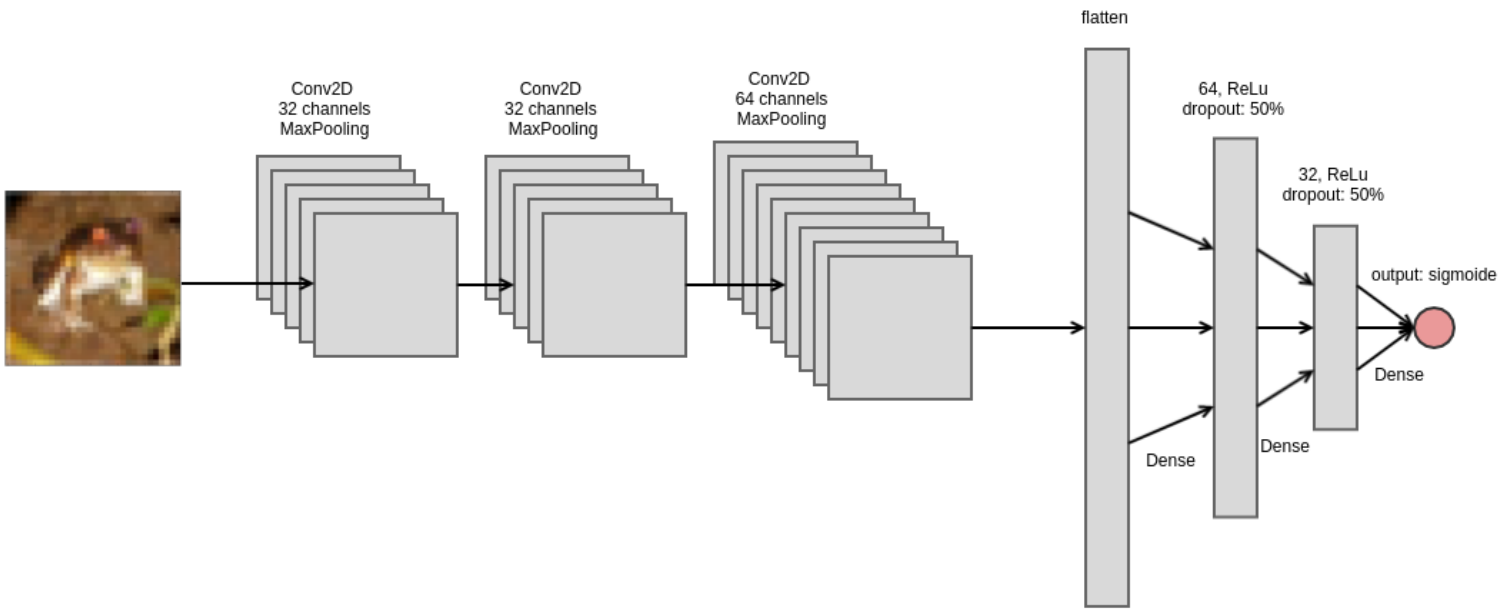
Listing 1 – Exemple de configuration

```
{
  "models_path" : "models",
  "models_weights_path" : "models_weights",
  "path_to_best_model" : "best_top_model.hdf5",
  "threshold" : 0.65,
  "proba_threshold" : 0.5,
  "transformation_ratio" : 0.05,
  "originalSize" : 32,
  "resizeFactor" : 5,
  "batch_size_source" : 10,
  "batch_size_target" : 10,
  "epochs_source" : 1000,
  "epochs_target" : 1000,
  "classes_source" : ["dog","truck"],
  "classes_target" : ["deer","horse"],
  "layerLimit" : 15,
  "times" : 1,
  "lr_source" : 0.0001,
  "lr_target" : 0.0001,
  "step" : 3,
  "recompute_transfer_values" : false,
  "train_top_model" : false,
  "reinitialize_bottom_layers" : false,
  "bigNet" : true,
  "verbose" : true
}
```

Figure 5 – Exemple de fichier de configuration

Petit CNN

Apprentissage à partir de 0 directement sur le domaine cible dans le boosting $\rightarrow$ moins de back-propagation.
Nécessité d'utiliser un gros réseau ?



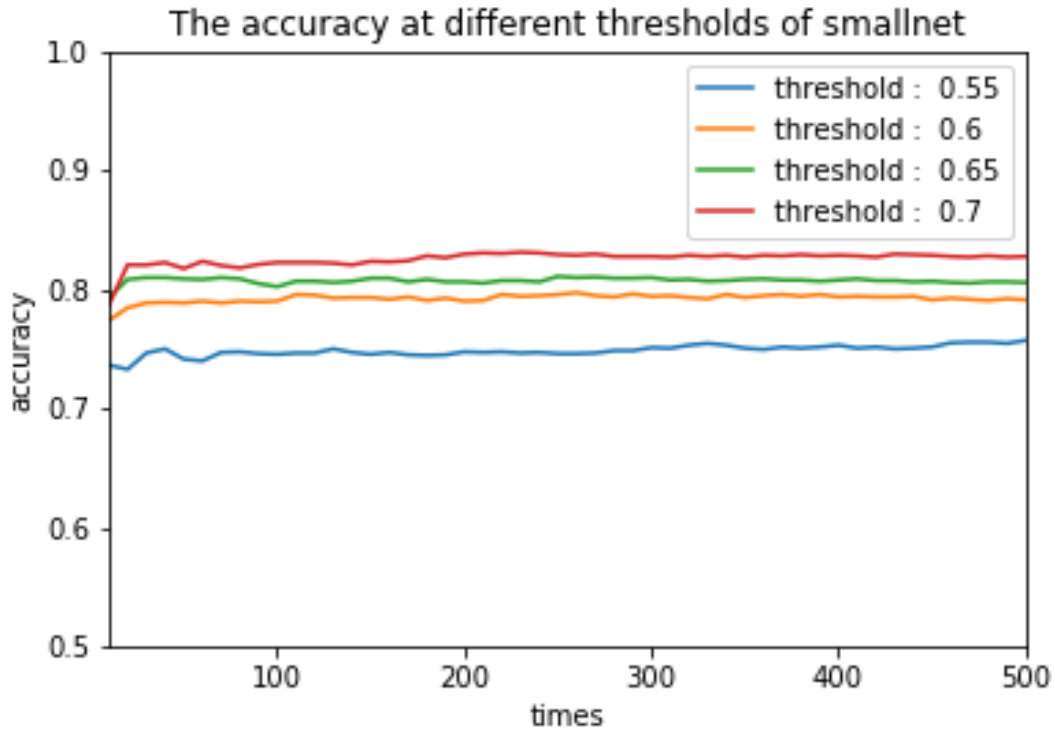


Figure 6 – Évolution de la précision de l'algorithme de Boosting avec des petits réseaux de convolution (sans transfert)

Plus le projecteur est fort, plus la précision augmente.
L'algorithme de boosting a été correctement implémenté.

Résultats de Transboost avec Xception

Précision à chaque étape de boosting .65 à 10 projecteurs → converge mais précision en test 51 % environ

Premier projecteur : 52 % environ sur le set de validation

↪ seuil .70 pour avoir au moins 55 % de précision en validation ?
Ne converge pas en 4h

Divers configurations testées : classes, learning rate, optimiseur ! ! !

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)*

Tensor Flow : compliquée à modifier la structure du modèle ou geler certaines couches.
Keras : surcouche de Tensor Flow. Une syntaxe plus claire, des outils plus simples.

Machines physiques/virtuelles : Temps d'exécution très long (dizaine d'heures et même en jours). Erreurs de mémoire.

problèmes de mémoire

Fonction *del* de python ne garantit pas la suppression immédiate de l'objet.

Même en forçant l'exécution du *"garbage collector"* du langage C (utilisé pour gérer la mémoire en Python), des modèles n'ont pas été entièrement supprimés.

$\Longrightarrow$ Perturbation de l'entraînement de nouveaux modèles.

Solution :

- Enregistrer les modèles (architecture et poids) sur le disque dur de la machine avec des fonctions de Keras.
- Utiliser la fonction $k.clear\_session()$ pour effacer tous les objets de la session.
- Créer un réseau chaque fois à partir de l'architecture sauvée et d'y charger les poids du modèle pour l'entraîner.

Inconvenient

temps d'accès beaucoup plus long du disque dur par rapport a la mémoire vive. *(malgré tout très petit par rapport au temps d'entraînement)*

- La méthode Transboost est séduisante sur les séries temporelles
- Le boosting marche avec le petit CNN (sans transfert)
- Échec avec transfert & gros CNN. Y a-t-il déstabilisation des couches de bas niveau ?
- Challenge : Nécessité de capacité de calcul importante pour parcourir le réseau de convolution de base.
- le boosting pourrait peut-être marcher avec du transfer learning classique.

Bibliographie

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

[2] Murena P.A. Olivier R. Cornuejols A., Akkoyunlu S. 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.