CONSTELLATION NETS FOR FEW-SHOT LEARNING

Emmanuel Judith et **Wenshan Wu**

Sorbonne Université Sciences

Introduction

Des progrès énormes ont été fait à la fois dans le développement et dans les domaines d'application liés aux CNN. La visualisation des filtres de convolution peut nous permettre de voir avec précision les caractéristiques de classes d'objets pour une tâche de détection par exemple. Cependant ces caractéristiques sont implicites puisqu'elles correspondent à des projection des données en entrée sans connaissance préalable dessus, et apprises durant la phase de backpropagation.
D'un autre côté pour la représentation explicite nous trouvons des modèles en constellation qui ont la particularité de faire de l'apprentissage non supervisé, d'utiliser du clustering et de modéliser la configuration spatiale des parties. Le papier que nous présentons [1] utilise les deux représentations en utilisant des modèles en constellation et des CNN pour de la classification en few-shot. Ce type d'apprentissage attire de plus en plus l'attention car il permet d'apprendre une representation des données et de la généraliser.

Jeux de données

3 jeux de données sont utilisés pour les expériences : CifarFS, FC100 et MiniImagenet. Ce sont des jeux de données très largement utilisés pour faire du few-shot learning. CIFAR-FS est composée de 100 classes, 64 pour le meta-train, 16 pour le meta-evaluation et 20 pour le meta-test. Chaque classe contient 600 images de dimensions 32 x 32. FC100 est un autre dataset basé sur CIFAR-100 mais pour lequel les classes sont regroupées en 20 superclasses. Enfin MiniImageNet est un dataset qui contient également 100 classes. Les classes sont réparties aléatoirement de la manière suivant : 64 pour le train , 16 pour val et 20 pour test. Il y a 600 images pour chaque classe.

Convolutional Features Map

Il y a d'abord une étape de convolution qui produit une *convolutional features map*, après quoi les données passent dans un module de constellation. Enfin on assemble la sortie du module de constellation avec la *convolutional features map*.
Il y a deux architectures de base qui sont utilisées :

- Conv4 :

  4 blocs successifs de convolution

  - convolution 3x3
  - normalisation
  - Relu
  - maxpool 2x2

  64 filtres pour chaque bloc de convolution

- ResNet-12 :

  4 blocs résiduels composé de 3 couches de convolutions

  - convolution 3x3
  - normalisation
  - Relu
  - maxpool 2x2

  64,128,256,512 filtres respectivement pour les blocs

Module de Constellation

A partir de la *feature map* $\in \mathbb{R}^{BxHxWxK}$ on extrait les *cell features* $U = u_1, u_2, ..., u_n$ avec $n = BHW$ et $u_i \in \mathbb{R}^C$. On affecte ces *cell features* à des clusters à l'aide d'une méthode de *cell feature clustering*. Ensuite on performe une cell relation modeling dan le but de constuire des relations spatiales.

Cell Feature Clustering
On pratique un soft k-means.

- Initialisation : Initialiser les centres des clusters $V = v_1, v_2, ..., v_K$
- Affectation : Etant données des *cell features* $U = u_1, u_2, ..., u_n$ on calcule la distance entre u_i et les centres. Ensuite affectation soft :

$$d_{ik} = ||u_i - v_k||_2^2, \qquad m_{ik} = \frac{e^{-\beta d_{ik}}}{\sum_j e^{-\beta d_{ij}}}, \qquad v'_k = \frac{\sum_i m_{ik} u_i}{\sum_i m_{ik}}$$

- Mouvement des centroids :

$$v_k < -(1-\eta)v_k + \eta v'_k, \qquad n = \frac{\lambda}{s_k + \Delta s_k}$$

avec $\Delta s = \sum_i m_i$ un compteur

Mise à jour du compteur :

$$s < -s + \Delta s$$

A la fin on obtient la distance map D $\in \mathbb{R}^{BxHxWxK}$

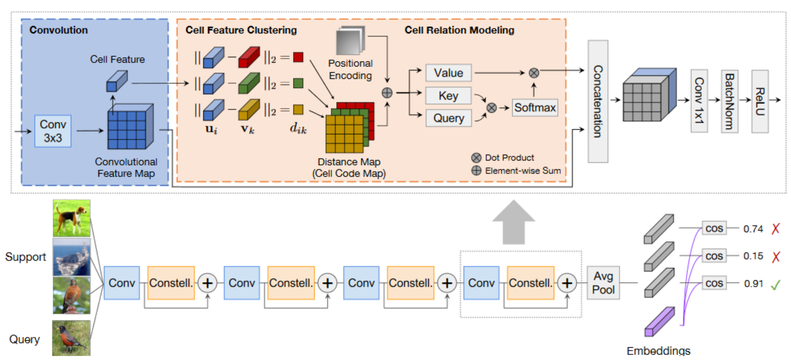
Cell Relation and Spatial Configuration Modeling
Positional encoding P $\in \mathbb{R}^{BxHxWxK}$

$$F_1 = SpatialFlatten(D + P) \in \mathbb{R}^{BxWHxK}, \qquad F'_1 = SpatialFlatten(D) \in \mathbb{R}^{BxWHxK}$$

$$[F^q, F^k, F^v] = [F_1 W^q, F_1 W^k, F'_1 W^v]$$

$$F_A = Att(F^q, F^k, F^v) = softmax(\frac{F^q, (F^k)^T}{\sqrt{K}})F^v$$

Schéma du modèle



Apprentissage

Algorithm 1 Training episode loss computation for prototypical networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. RANDOMSAMPLE(S, N) denotes a set of N elements chosen uniformly at random from set S, without replacement.

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.
Output: The loss J for a randomly generated training episode.

$V \leftarrow$ RANDOMSAMPLE$(\{1, \ldots, K\}, N_C)$ ▷ Select class indices for episode
for k in $\{1, \ldots, N_C\}$ **do**
  $S_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k}, N_S)$ ▷ Select support examples
  $Q_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k} \setminus S_k, N_Q)$ ▷ Select query examples
  $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$ ▷ Compute prototype from support examples
end for
$J \leftarrow 0$ ▷ Initialize loss
for k in $\{1, \ldots, N_C\}$ **do**
  for $(\mathbf{x}, y)$ in Q_k **do**
    $J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_\phi(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_\phi(\mathbf{x}), \mathbf{c}_k)) \right]$ ▷ Update loss
  end for
end for

Fig. 2: ProtoNet pour few-shot learning[2]

Résultats

	CIFAR-FS	FC100
Conv4	0.44	0.50
ResNet12	0.47	0.41

Conclusion

Article intéressant qui mélange deux représentations (implicit avec CNN, explicit avec cell clustering et cell modeling)
Originalité du module de constellation

Références

[1] "Constellation Nets for Few-Shot learning". In: (2020).
[2] R. Zemel J. Snell K. Swersky. "Prototypical Networks for Few-shot Learning". In: (2017).