UNIVERSITÉ LAVAL

GLO-4030/7030 APPRENTISSAGE PAR RÉSEAUX DE NEURONES PROFONDS

Spatial Transformer Networks (2015)

Max Jaderberg **Karen Simonyan** **Andrew Zisserman** **Koray Kavukcuoglu**

Google DeepMind, London, UK
{jaderberg,simonyan,zisserman,korayk}@google.com

Hiver 2020

Philippe Giguère

Description générique

- **But** : améliorer l'invariance spatiale à
  - translation, rotation, échelle, déformation élastique, etc.
- **Comment** : ajouter un module ajustable (*learnable*) qui manipule explicitement les données dans le domaine spatial
- Pleinement différentiable
- Peut être insérée partout dans une architecture préexistante
- Vérité-terrain sur la meilleure transformation non-nécessaire
- Très populaire : 2700+ citations

Prédiction de la transformation

Nombreuses applications

- Classification d'image (en présence de fortes distorsions)
- Attention spatiale
  - Focaliser sur de petites parties de l'image, ce qui augmente l'efficacité computationnelle
- Co-localisation (lorsque plusieurs instances du même objet sont présents)

Spatial transformer (ST)

- Applique une **transformation spatiale** sur les feature maps, lors du *forward pass*
- Va manipuler directement les *feature maps* (données), pas les filtres
  - Distorsion appliqués sur tous les canaux
- Peut s'insérer à plusieurs endroits (profondeur ***ou*** en parallèle)
- **Pleinement différentiable** : compatible avec entraînement end-to-end via backprop
- S'insère dans n'importe quelle architecture

Paramètres de transformation θ

La transformation θ sera donc conditionnelle à l'entrée

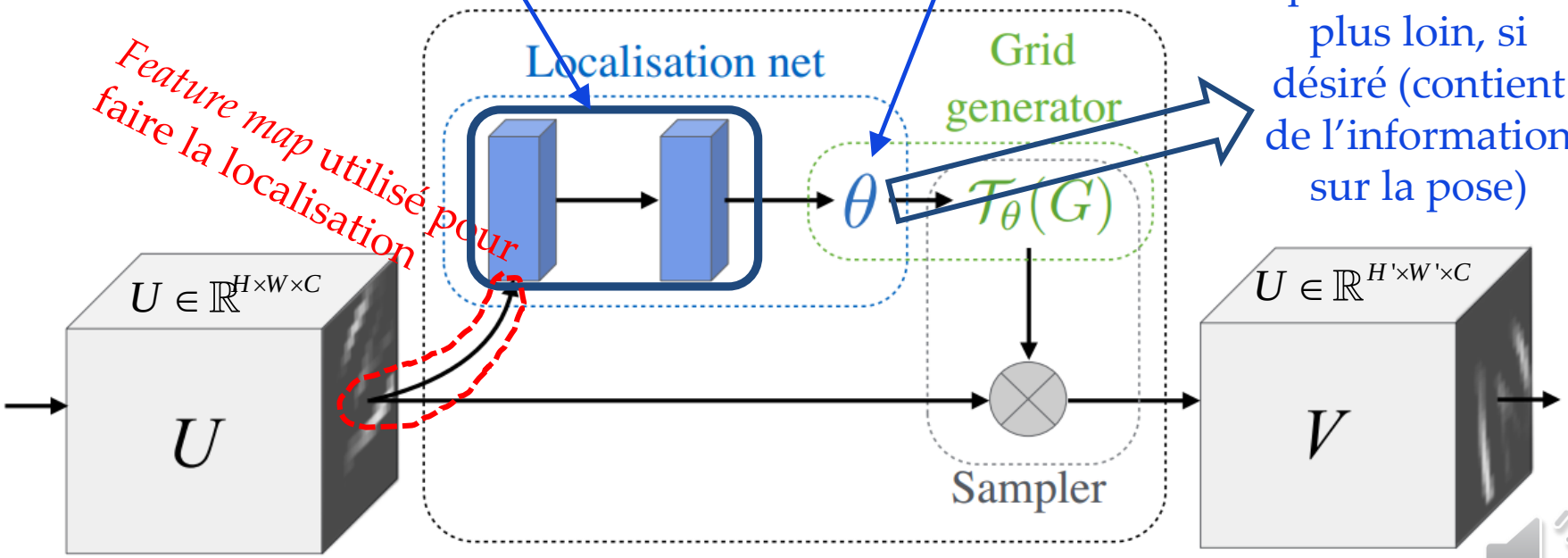
p.e. θ est 6-dim pour affine

Paramètres de transformation spatiale

CNN ou *fully-connected*, avec **couche finale de régression**

θ peut être utilisé plus loin, si désiré (contient de l'information sur la pose)

Feature map utilisé pour faire la localisation



Spatial Transformer

Détail de l'architecture (PyTorch)

```
# Spatial transformer localization-network
self.localization = nn.Sequential(
    nn.Conv2d(1, 8, kernel_size=7),
    nn.MaxPool2d(2, stride=2),
    nn.ReLU(True),
    nn.Conv2d(8, 10, kernel_size=5),
    nn.MaxPool2d(2, stride=2),
    nn.ReLU(True)
)

# Regressor for the 3 * 2 affine matrix
self.fc_loc = nn.Sequential(
    nn.Linear(10 * 3 * 3, 32),
    nn.ReLU(True),
    nn.Linear(32, 3 * 2)
)
```

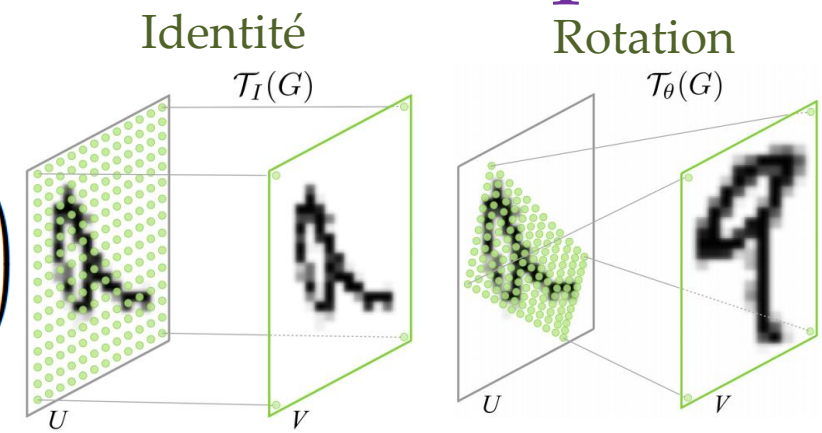
http://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial.html

Déformation des feature maps

6 paramètres θ_{ij} permettant le cropping, translation, rotation, échelle et cisaillement

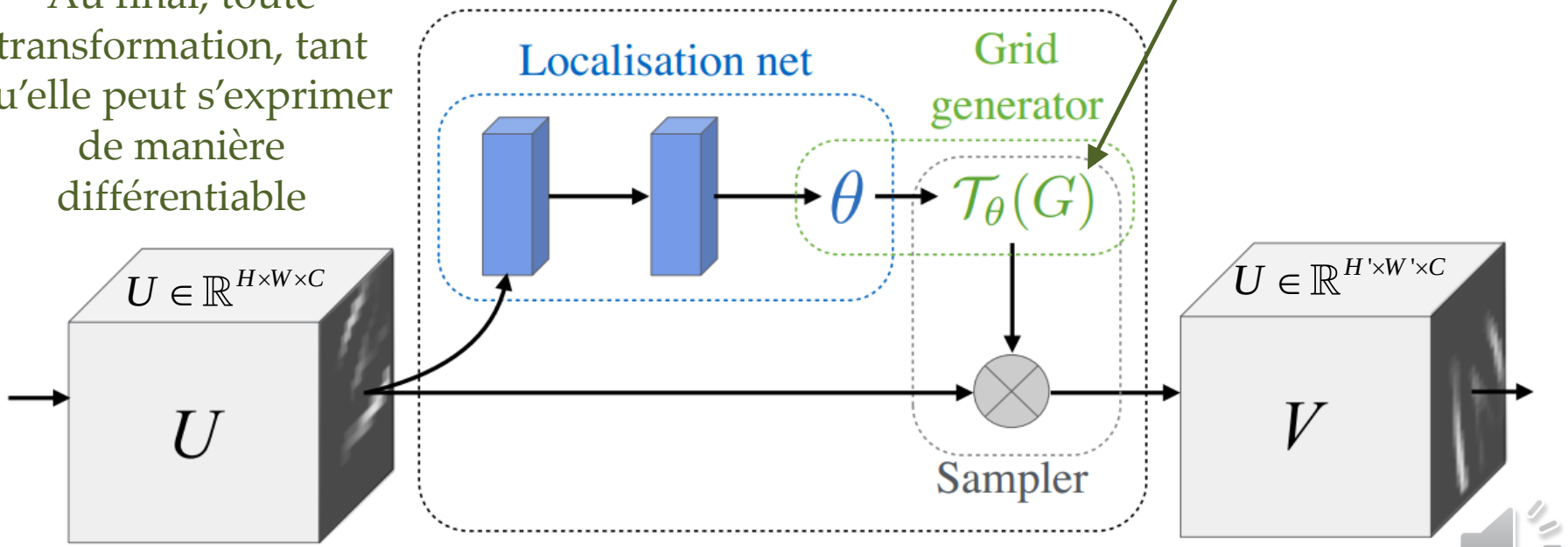
$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

… ou à avoir un modèle d'attention pur (cropping via échelle s + translation)



Similaire à *texture mapping*

Au final, toute transformation, tant qu'elle peut s'exprimer de manière différentiable

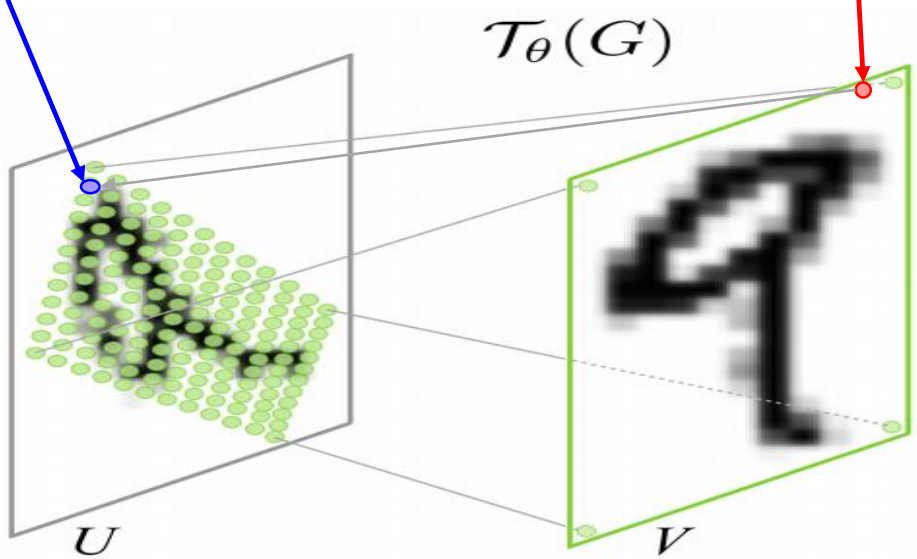


Spatial Transformer

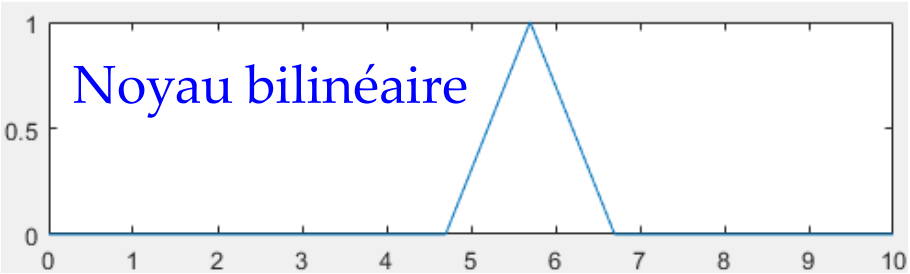
Détail de l'échantillonnage

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

Noyau pour l'échantillonnage



(grille fixe et régulière)

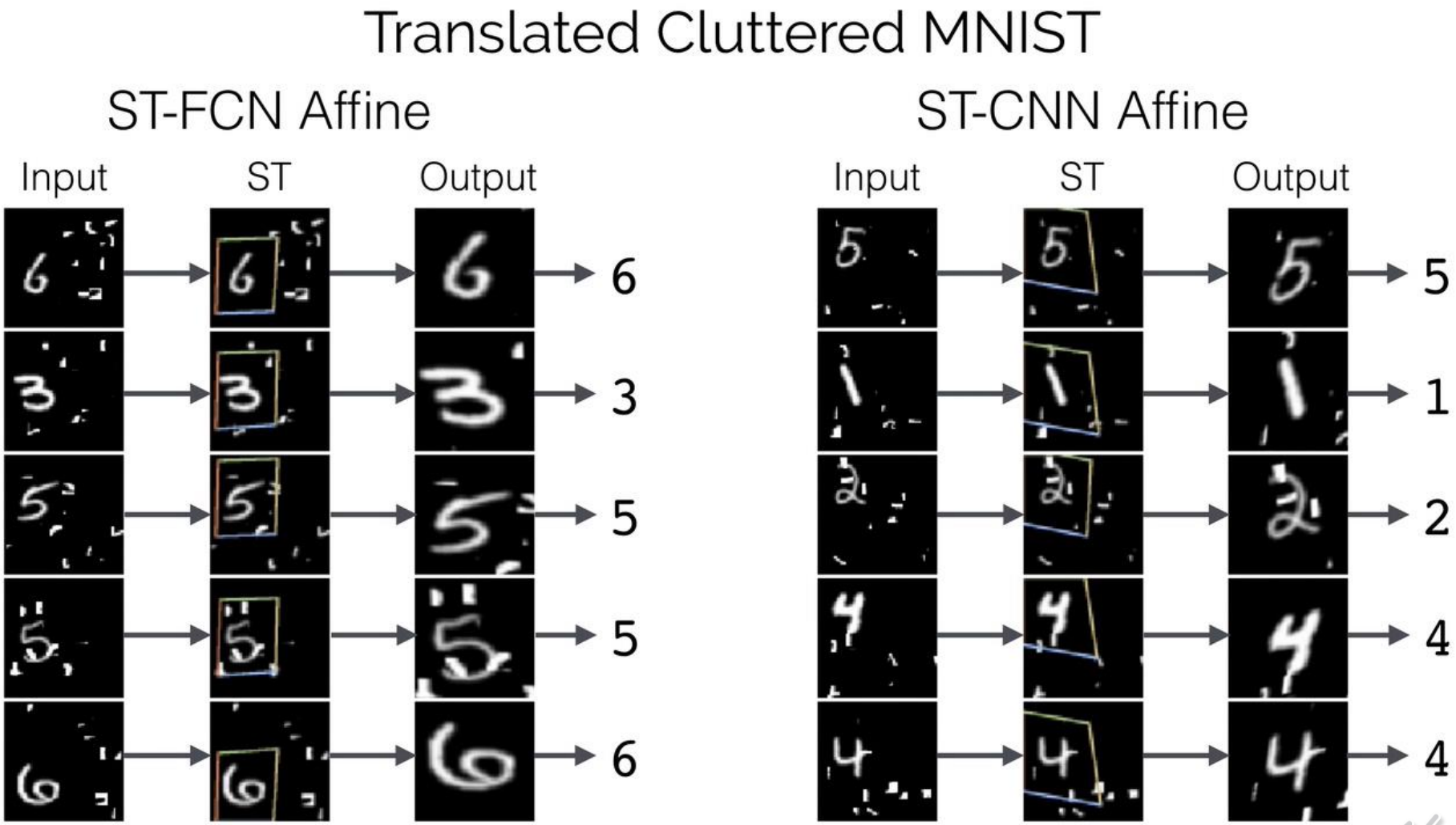


Pleinement différentiable par rapport à U, x^s_i et y^s_i

Spatial transform : notes

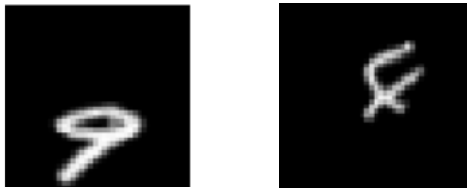
- Peut les avoir à différentes profondeurs dans un CNN
  - Transformations de plus en plus arbitraires et complexes, via composition
- Ou *n* en parallèle
  - Pour focaliser sur <u>exactement</u> *n* objets/parties dans une image

Exemple


Translated Cluttered MNIST
ST-FCN Affine
Input
ST
Output
6
3
5
5
6
ST-CNN Affine
Input
ST
Output
5
1
2
4
4

Expérimentations

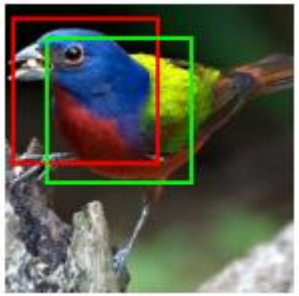
- MNIST déformés



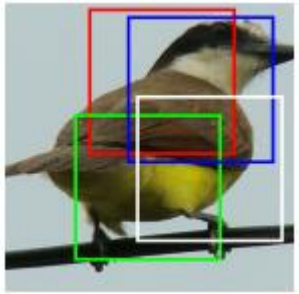
- Street View House Numbers (SVHN)

- Bird Classification dataset CUB-200-2011

2 ST en parallèle



4 ST en parallèle



MNIST

- Rotation (R)
- Rotation, échelle et translation (RTS)
- Déformation élastique (E)

Réseaux

Baseline : CNN Fully-Connected (FCN)

Nouveau : ST-CNN ST-FCN

(entraînement standard : backprop, SGD, horaire d'entraînement pour learning rate, x-entropy multinomiale)

MNIST

Deux couches de maxpool (pour un peu d'invariance spatiale)

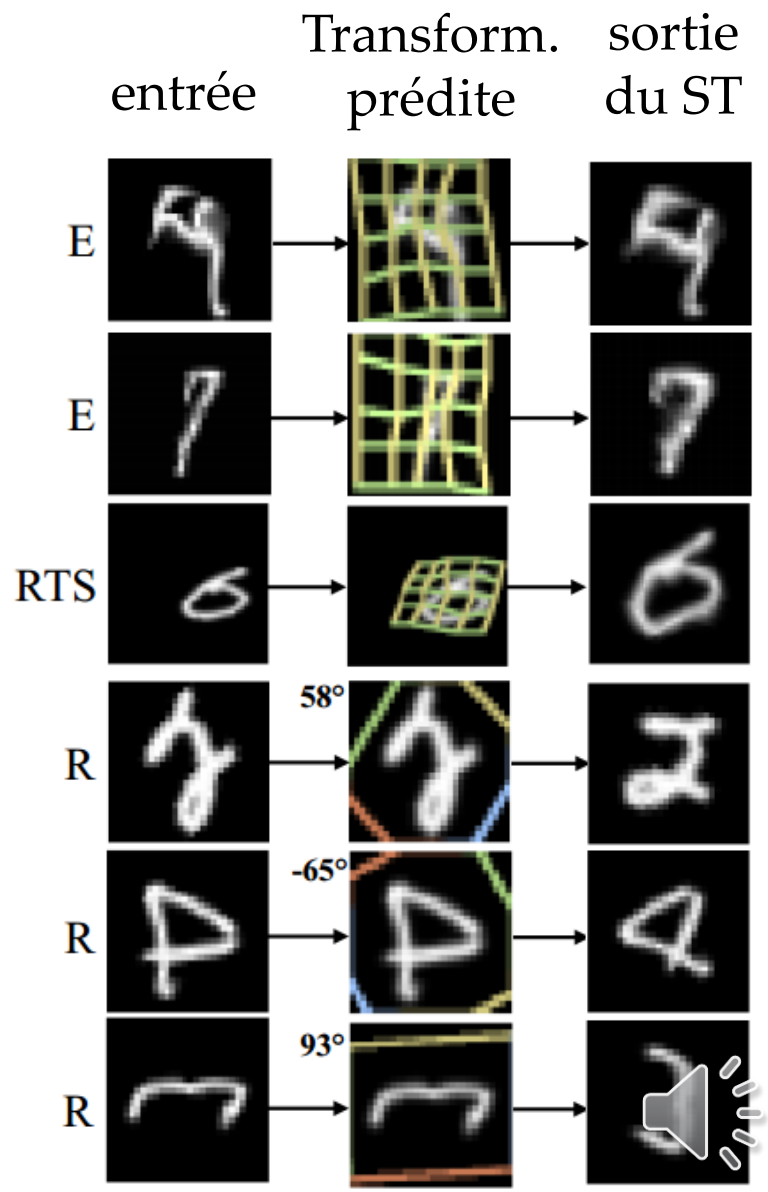
Taux d'erreur (%)

Model		MNIST Distortion R	RTS	P	E
FCN		2.1	5.2	3.1	3.2
CNN		1.2	0.8	1.5	1.4
ST-FCN	Aff	1.2	0.8	1.5	2.7
	Proj	1.3	0.9	1.4	2.6
	TPS	1.1	0.8	1.4	2.4
ST-CNN	Aff	0.7	0.5	0.8	1.2
	Proj	0.8	0.6	0.8	1.3
	TPS	0.7	0.5	0.8	1.1

Type de ST
(spatial transform)

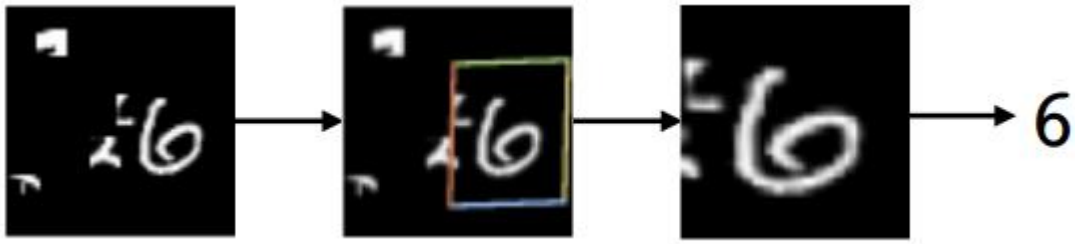
Thin plate spline transform

TPS est le mieux
(ne semble pas overfitter sur R)



MNIST

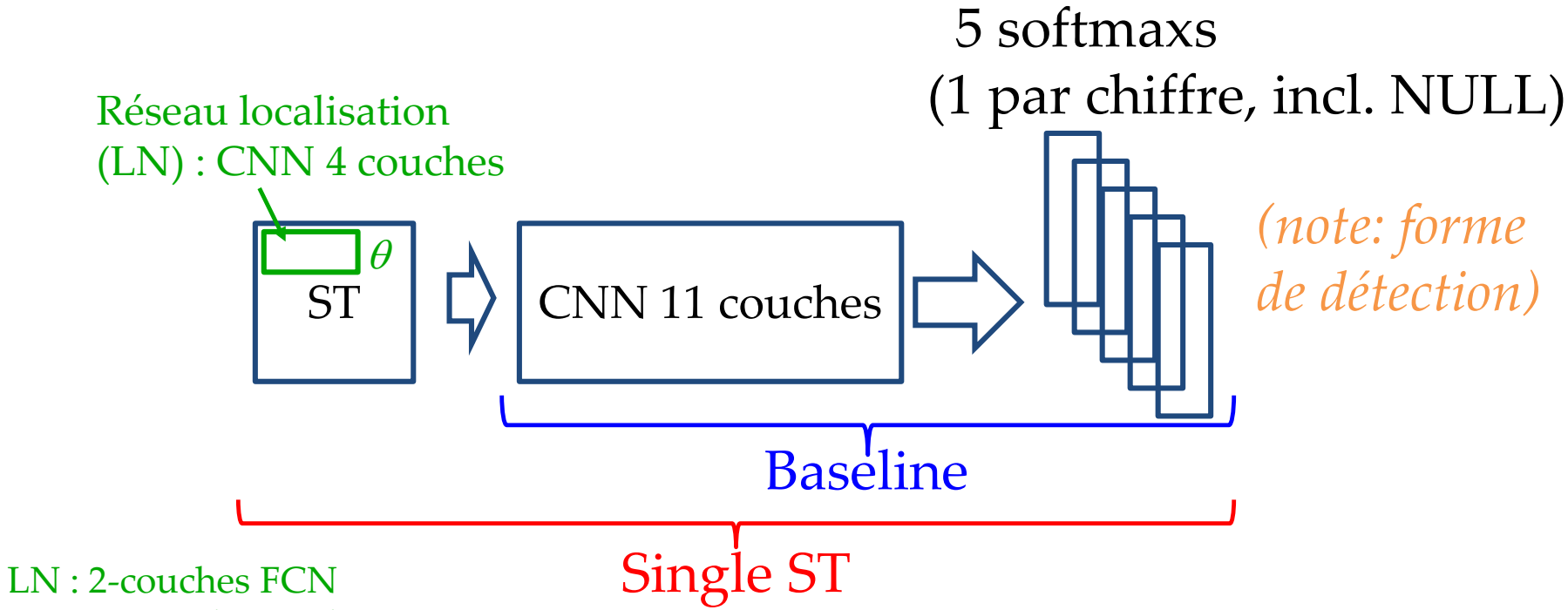
- Images 60x60
- Grandes translation + rotation + clutter



	FCN	CNN	ST-FCN	ST-CNN
Erreur (%)	13.2	3.5	2.0	1.7

Street View House Numbers

- SVHN : 200k images, 1 à 5 chiffres à identifier
- Grande variabilité dans l'échelle, disposition spatial



Multi ST: ST→conv→ST→conv→ST→conv→ST→conv

Tous entraînés avec SGD + dropout

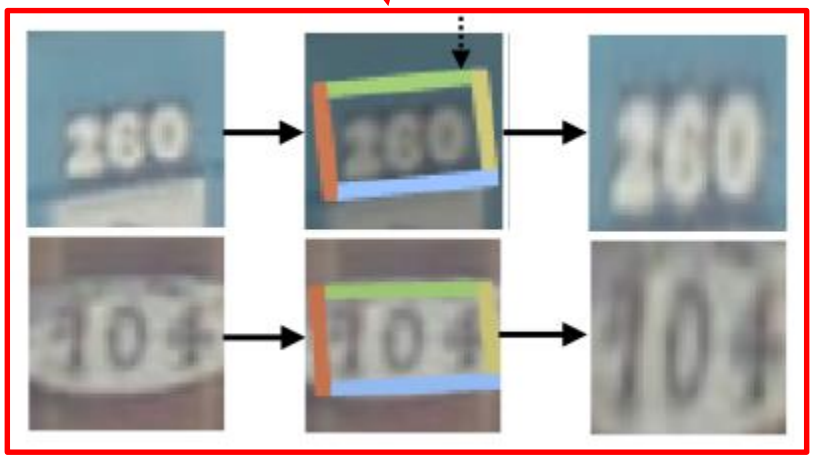
SVHN

Model averaging +
Monte Carlo averaging
(donc plusieurs passes)

Model		Size 64px	Size 128px
Maxout CNN [13]		4.0	-
CNN (baseline)		4.0	5.6
DRAM* [1]		3.9	4.5
ST-CNN	Single	3.7	**3.9**
ST-CNN	Multi	**3.6**	**3.9**

Une seule *forward pass*

ST-CNN est seulement 6% plus lent que CNN



Bird data set : CUB

- Classification fine : 200 espèces
  (6k images entraînement, 5.8k test)

[1] Thanks to the eagle-eyed Hugo Larochelle and Yin Zheng for spotting the birds nested in both the ImageNet training set and CUB test set.

- Plusieurs ST en parallèle (détails plus loin)
- Seule étiquette : classe dans l'image (pas la position de l'oiseau)
- Baseline :
  - Inception + batch normalization,
  - Pré-entraîné sur ImageNet
  - Fine-tuned sur CUB
  - Obtient l'état de l'art (82.3%) (Rappel : 2015)

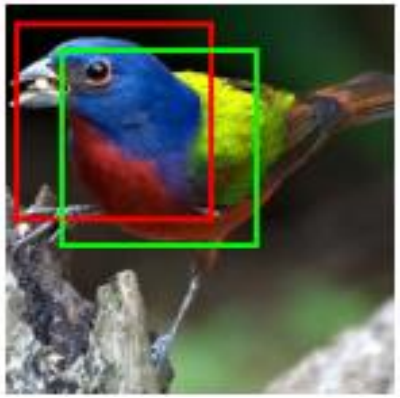
Bird data set : CUB

- Tests avec deux architectures
  - 2 Spatial Transform en parallèle
  - 4 Spatial Transform en parallèle
- Transformation est du type “attention”

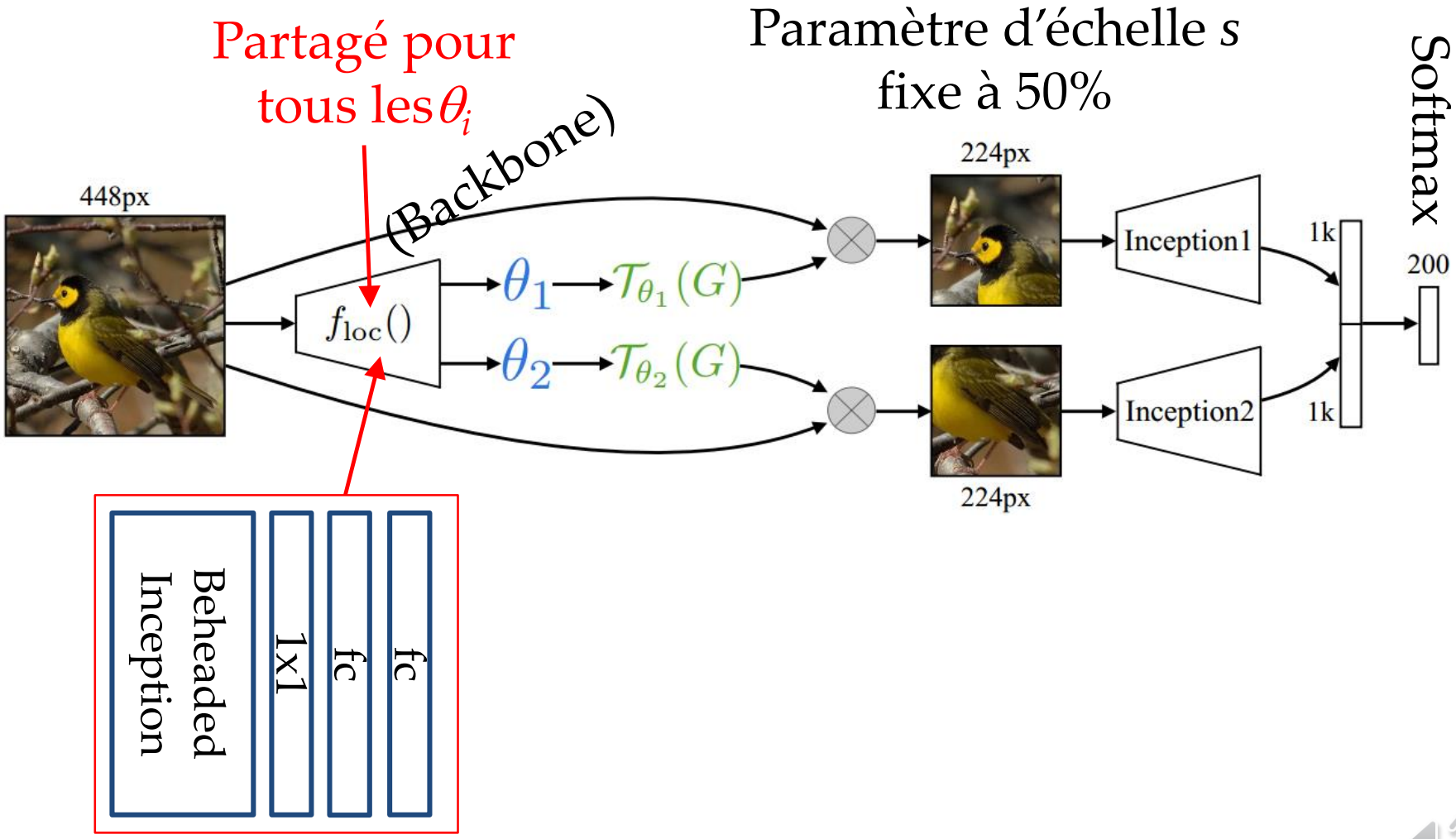
échelle s fixe (0.5)

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

Correspond à des recherches de *bounding boxes* de demi taille de l’image d’entrée (ici, cas de deux ST)



Architecture pour 2 ST parallèle

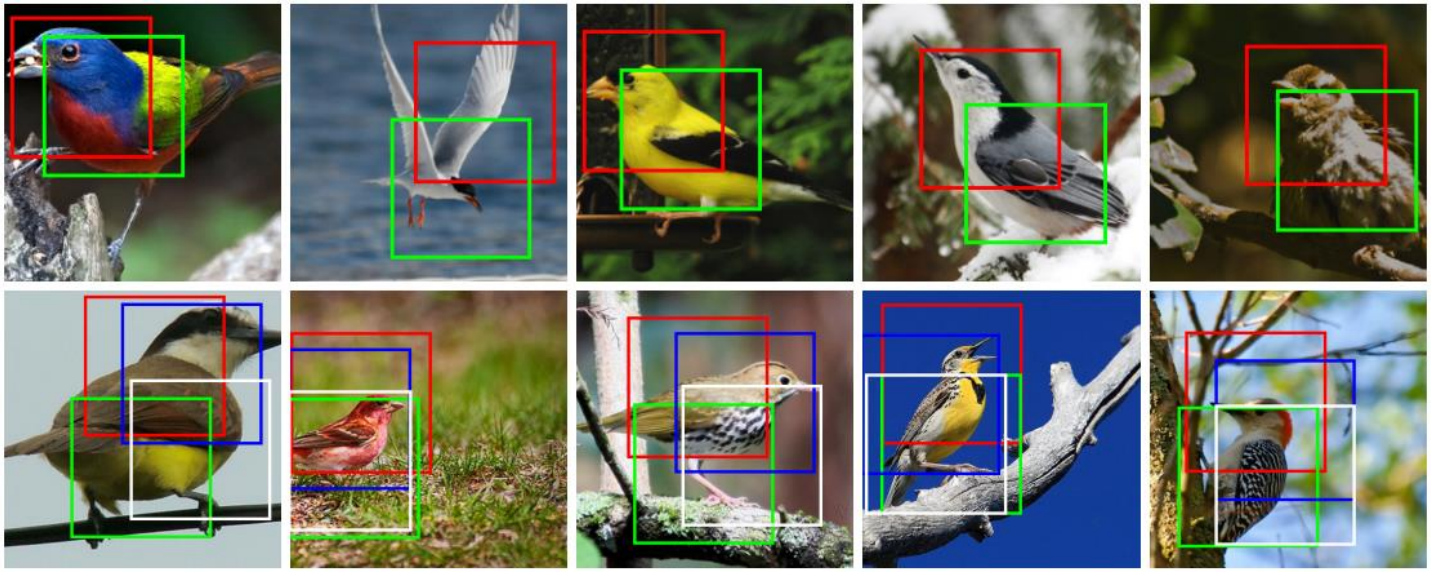


(Détails dans la version arxiv du papier)

Bird data set : CUB-200

Model	
Cimpoi '15 [5]	66.7
Zhang '14 [40]	74.9
Branson '14 [3]	75.7
Lin '15 [23]	80.9
Simon '15 [30]	81.0
CNN (baseline) 224px	82.3
2×ST-CNN 224px	83.1
2×ST-CNN 448px	83.9
4×ST-CNN 448px	**84.1**

TP2 → CNN (baseline) 224px



2×ST-CNN, one of the transformers (shown in red) learns to detect heads, while the other (shown in green) detects the body, and similarly for the 4×ST-CNN. Se specialise sur des parties

Conclusion

- Module confiné pleinement différentiable
- Peut s'ajouter facilement à de nombreuses architecture
- Effectue une transformation spatiale explicite des *feature maps*
- Appris end-to-end, sans altération de la fonction de perte
- Fourni des informations supplémentaires : transformation θ