Spatial Transformers in Feed-Forward Networks

Max Jaederberg, Karen Simonyan,
Andrew Zisserman and Koray Kavukcuoglu

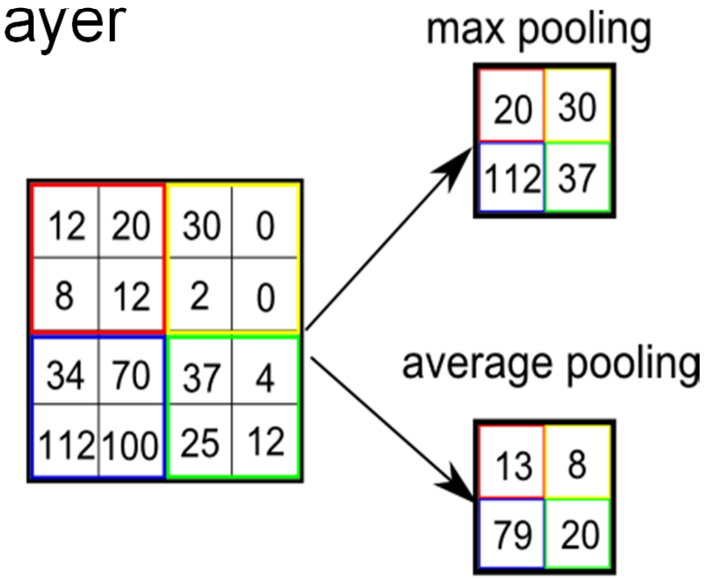
Google DeepMind and University of Oxford

ConvNets

- Interleaving convolutional layers with max-pooling layers allows translation invariance.

- Pooling is simplistic.
- Only small invariances per pooling layer
- Limited spatial transformation
- Pools across entire image

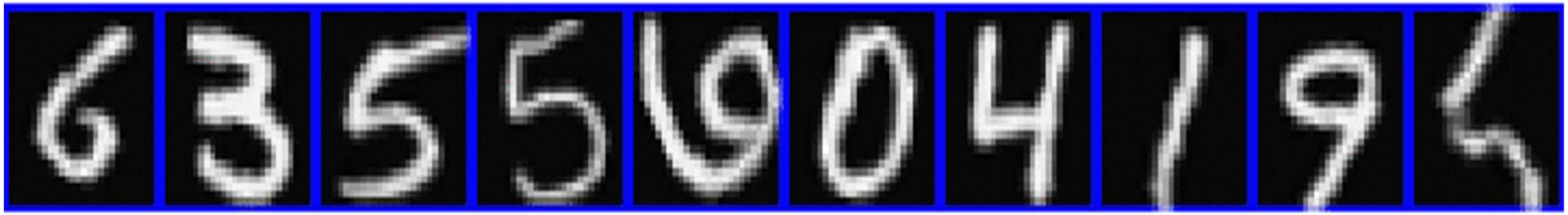
\+ Exceptionally effective

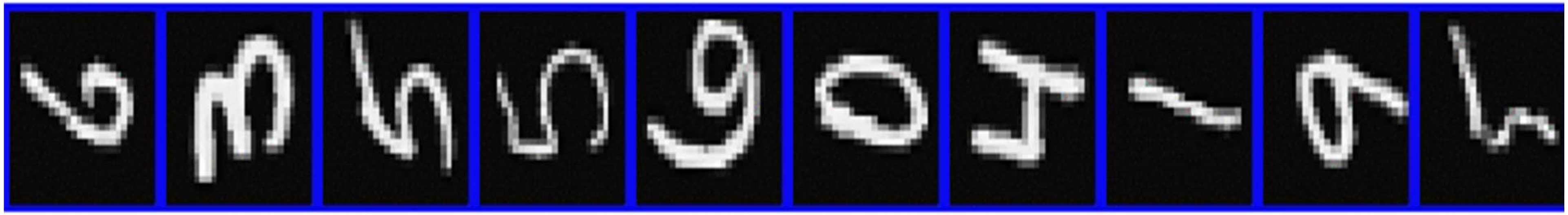


- Can we do better?

Motivation 1: transformations of input data

Rotated MNIST (+/- 90°)



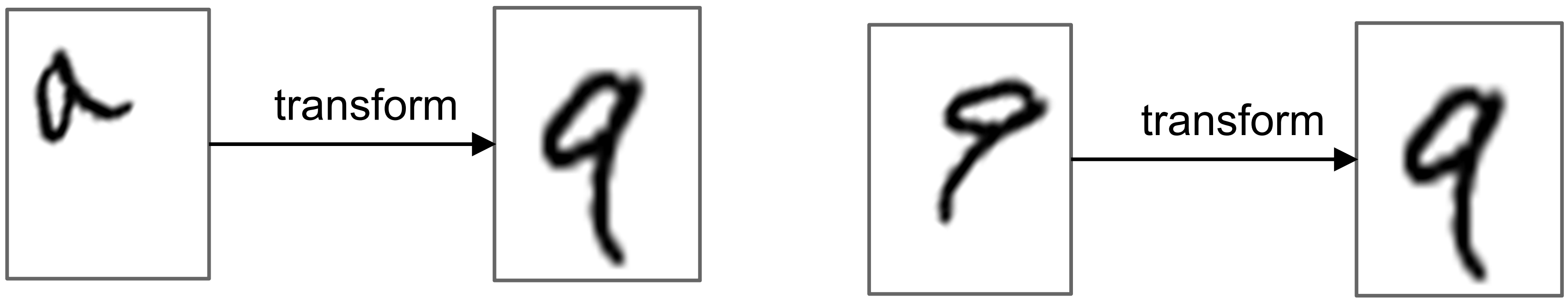


Motivation 2: attention

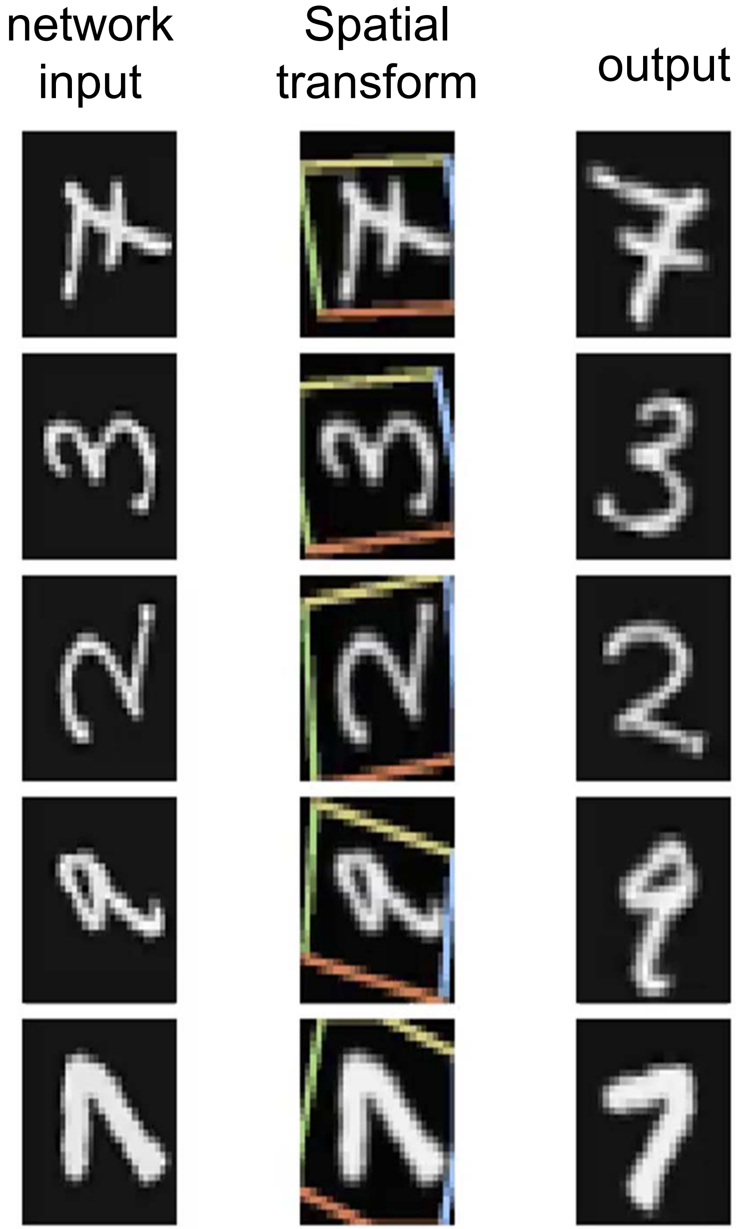
Conditional Spatial Warping

- Conditional on input featuremap, spatially warp image.

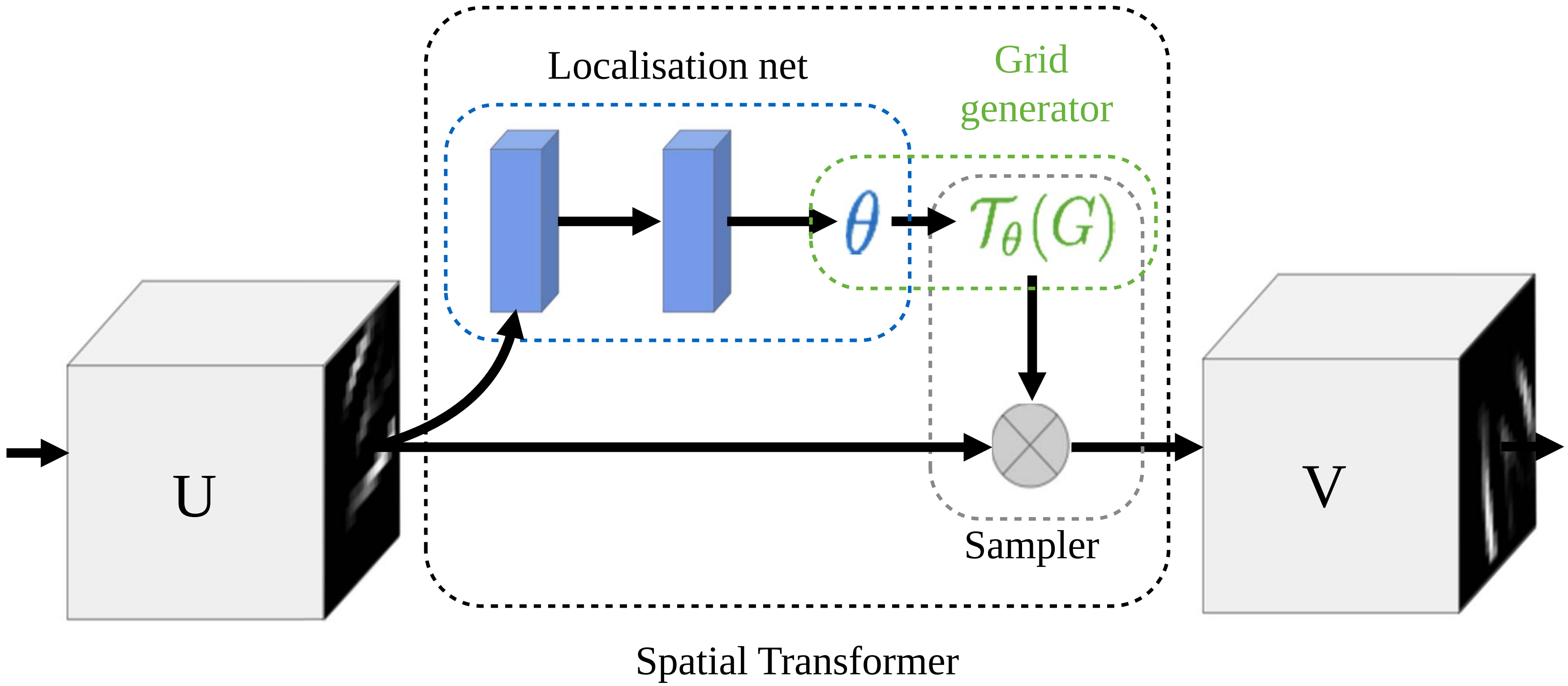
+ Transforms data to a space expected by subsequent layers
+ Intelligently select features of interest (attention)
+ Invariant to more generic warping



Conditional Spatial Warping



A differentiable module for spatially transforming data, conditional on the data itself

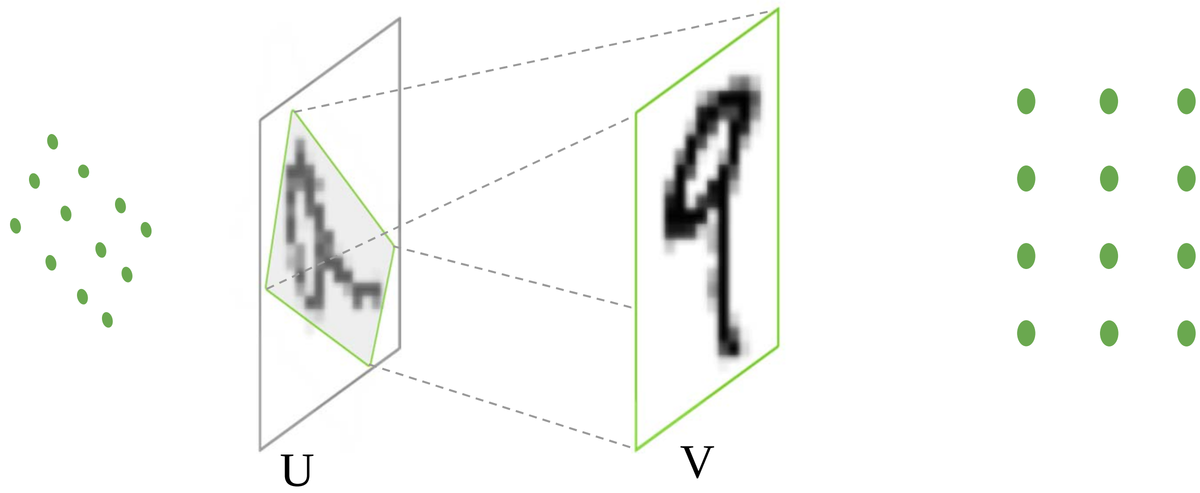


Sampling Grid

Warp regular grid by an affine transformation

Can parameterise, e.g. affine transformation

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

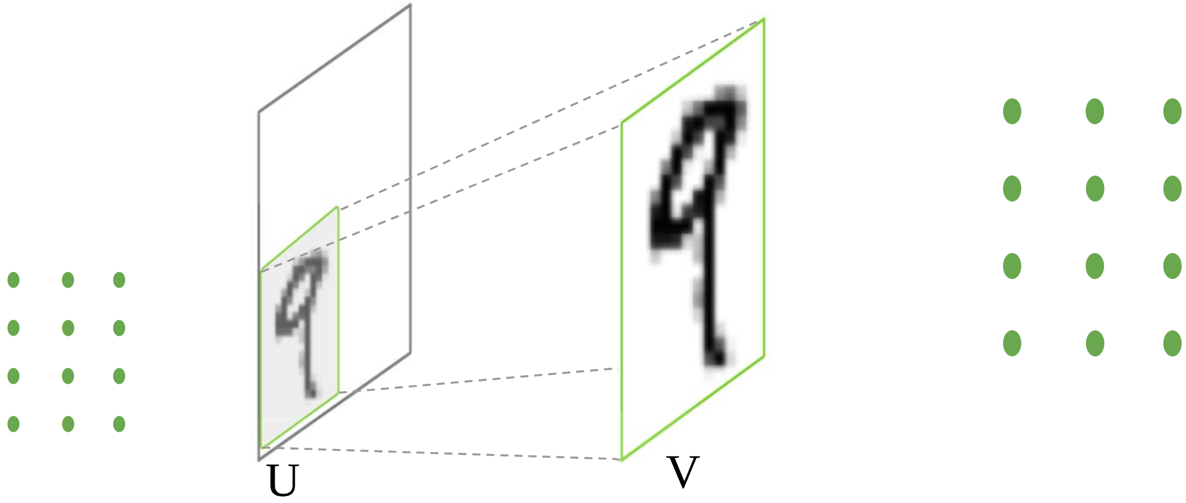


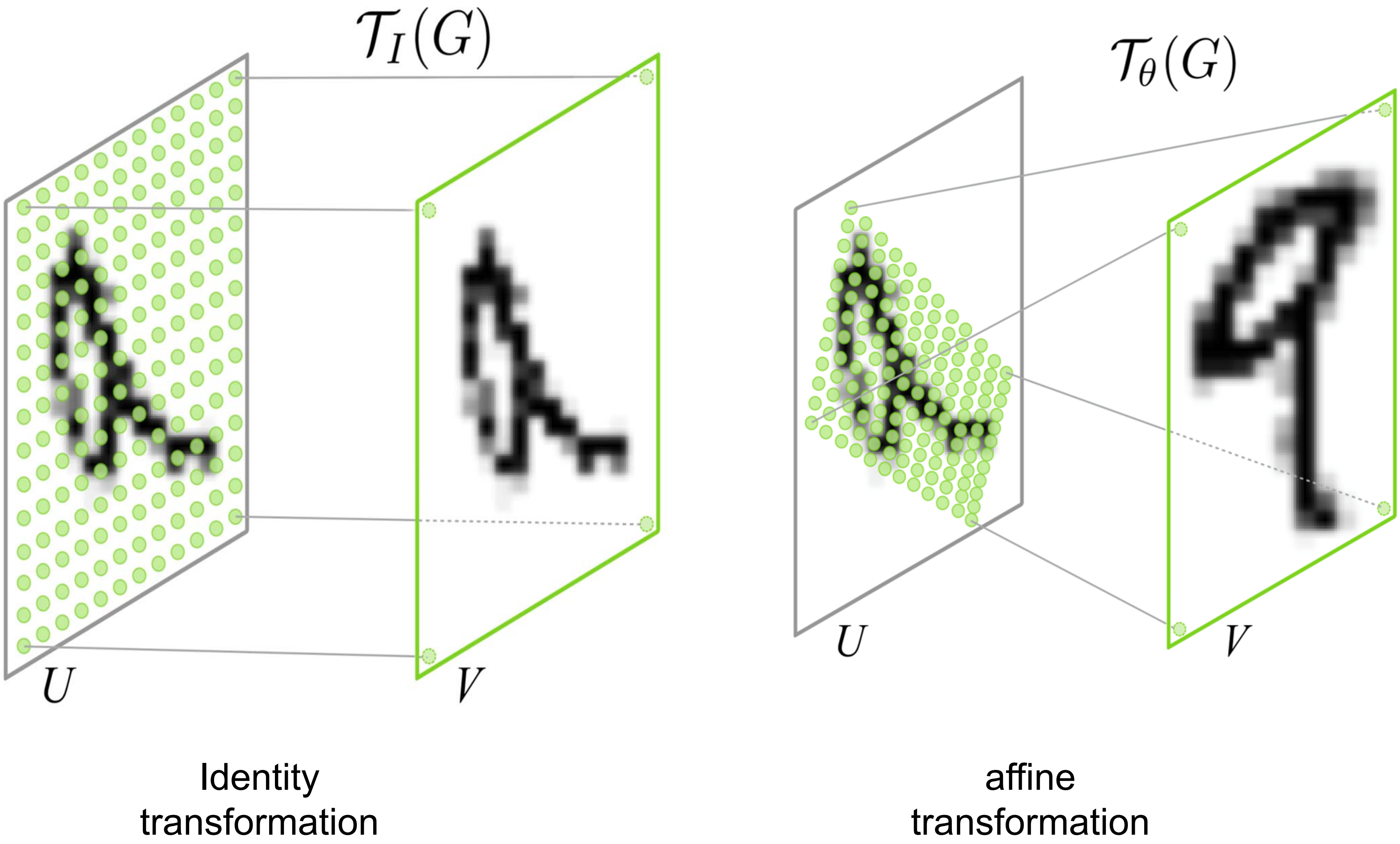
Sampling Grid

Warp regular grid by an affine transformation

Can parameterise attention

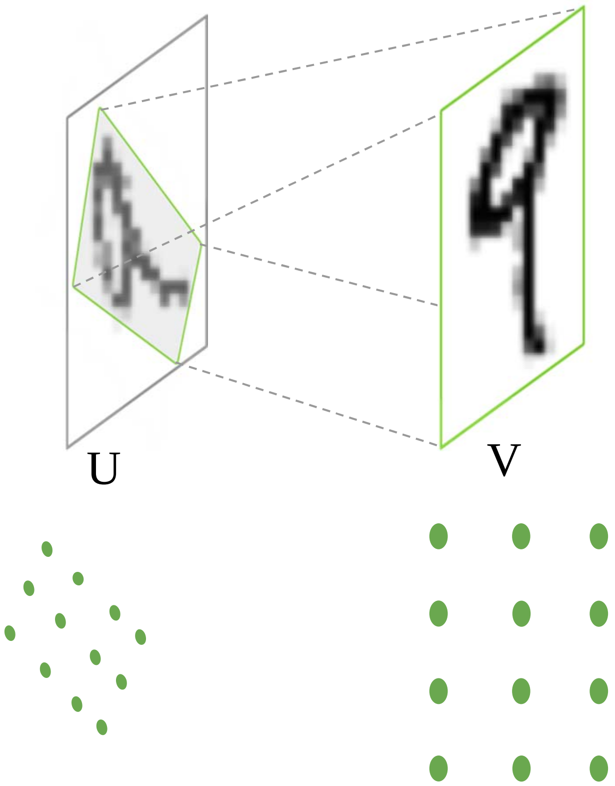
$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$




$\mathcal{T}_I(G)$
U
V
Identity transformation
$\mathcal{T}_\theta(G)$
U
V
affine transformation

Sampler

Sample input featuremap U to produce output feature map V
(i.e. texture mapping)



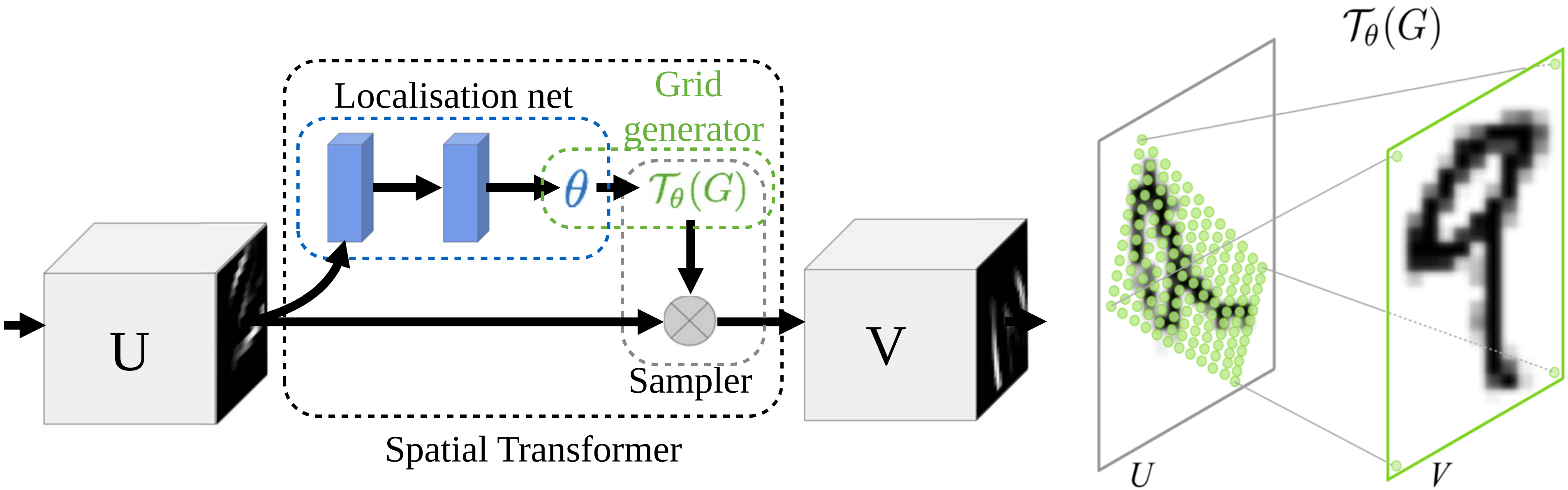
e.g. for bilinear interpolation:

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

and gradients are defined to allow backprop, eg:

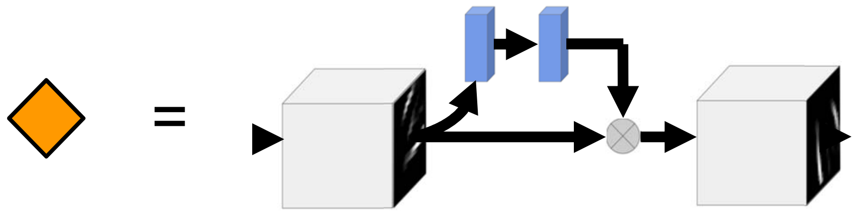
$$\frac{\partial V_i^c}{\partial U_{nm}^c} = \sum_n^H \sum_m^W \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

A differentiable module for spatially transforming data, conditional on the data itself



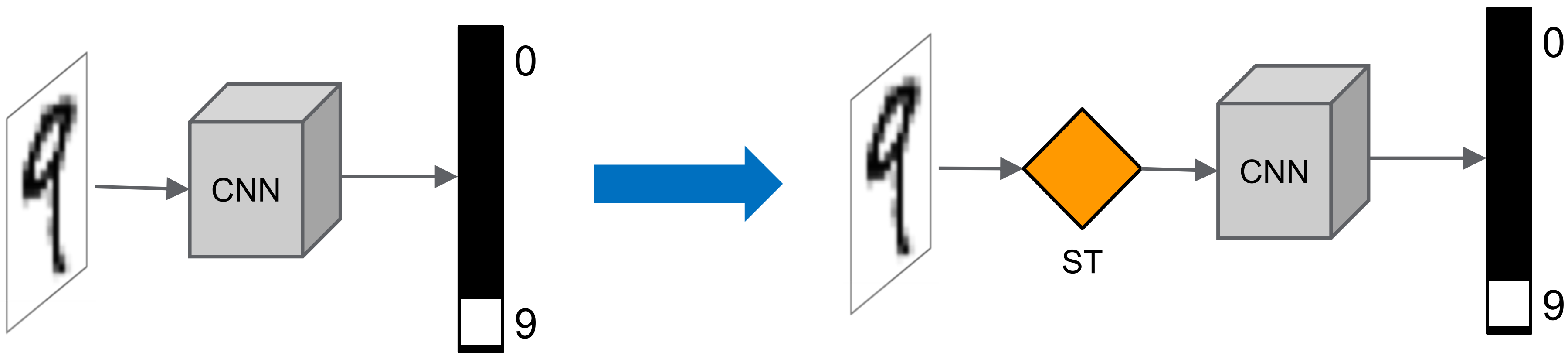
Spatial Transformer Networks

- Spatial Transformers is differentiable, and so can be inserted at any point in a feed forward network and trained by back propogation



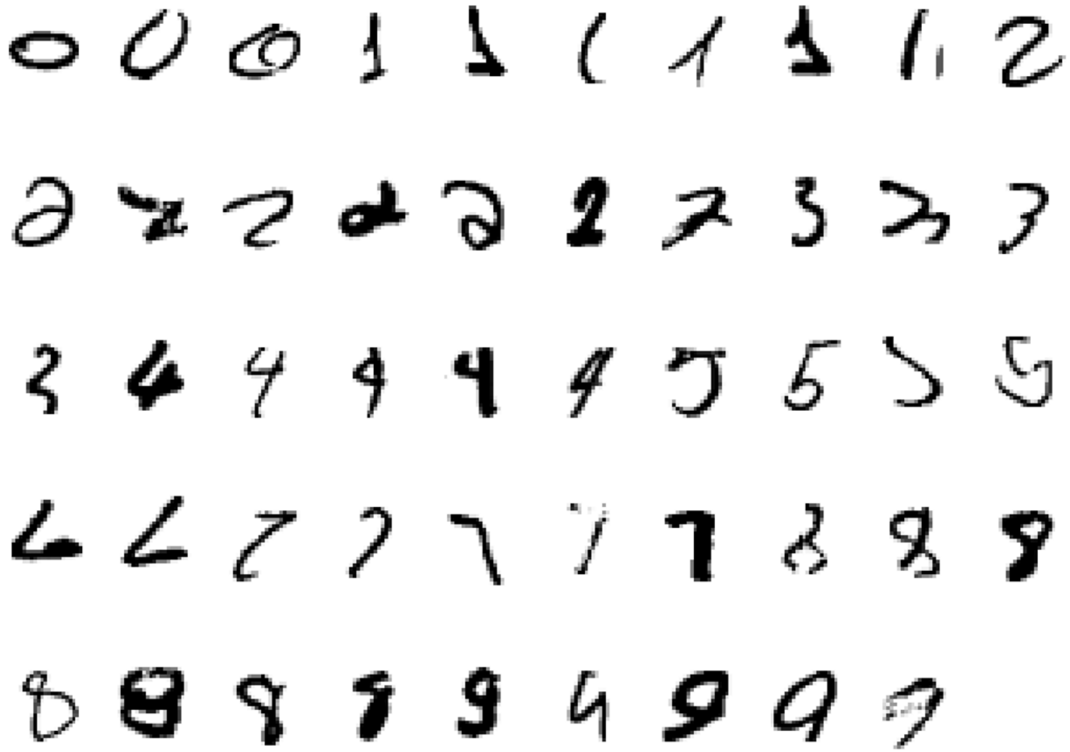
Example:

- digit classification, loss: cross-entropy for 10 way classification



MNIST Digit Classification

Training data: 6000 examples of each digit

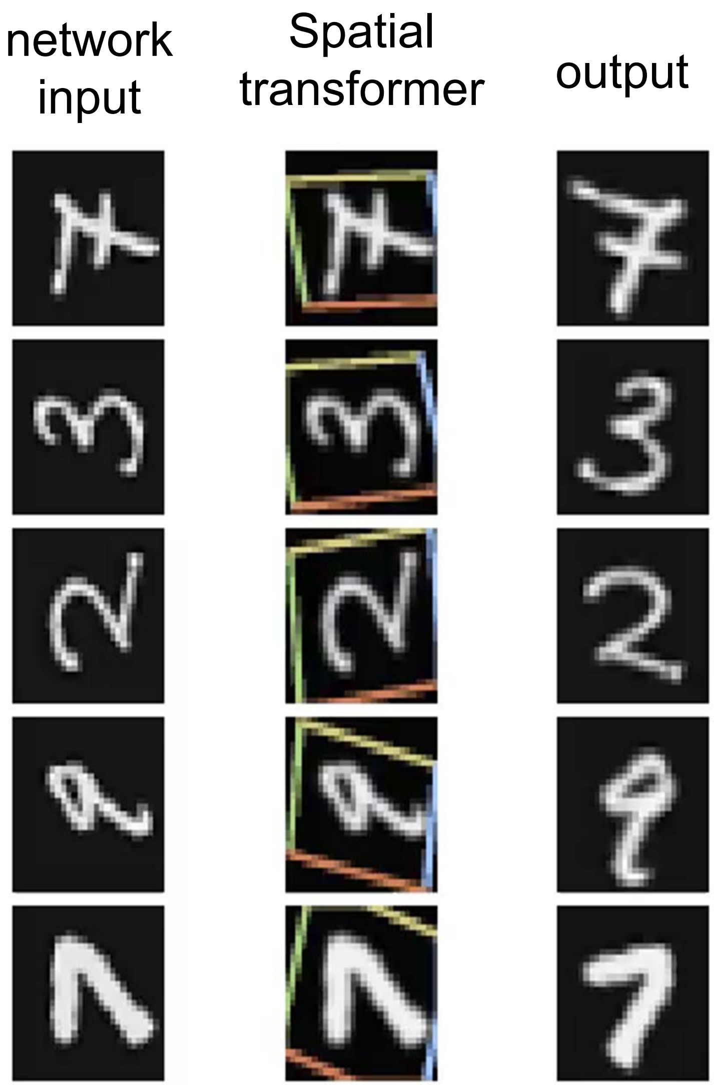


Testing data: 10k images

Can achieve testing error of 0.23%

Task: classify MNIST digits

- Training and test randomly rotated by (+/- 90°)
- Fully connected network with affine ST on input



Performance:

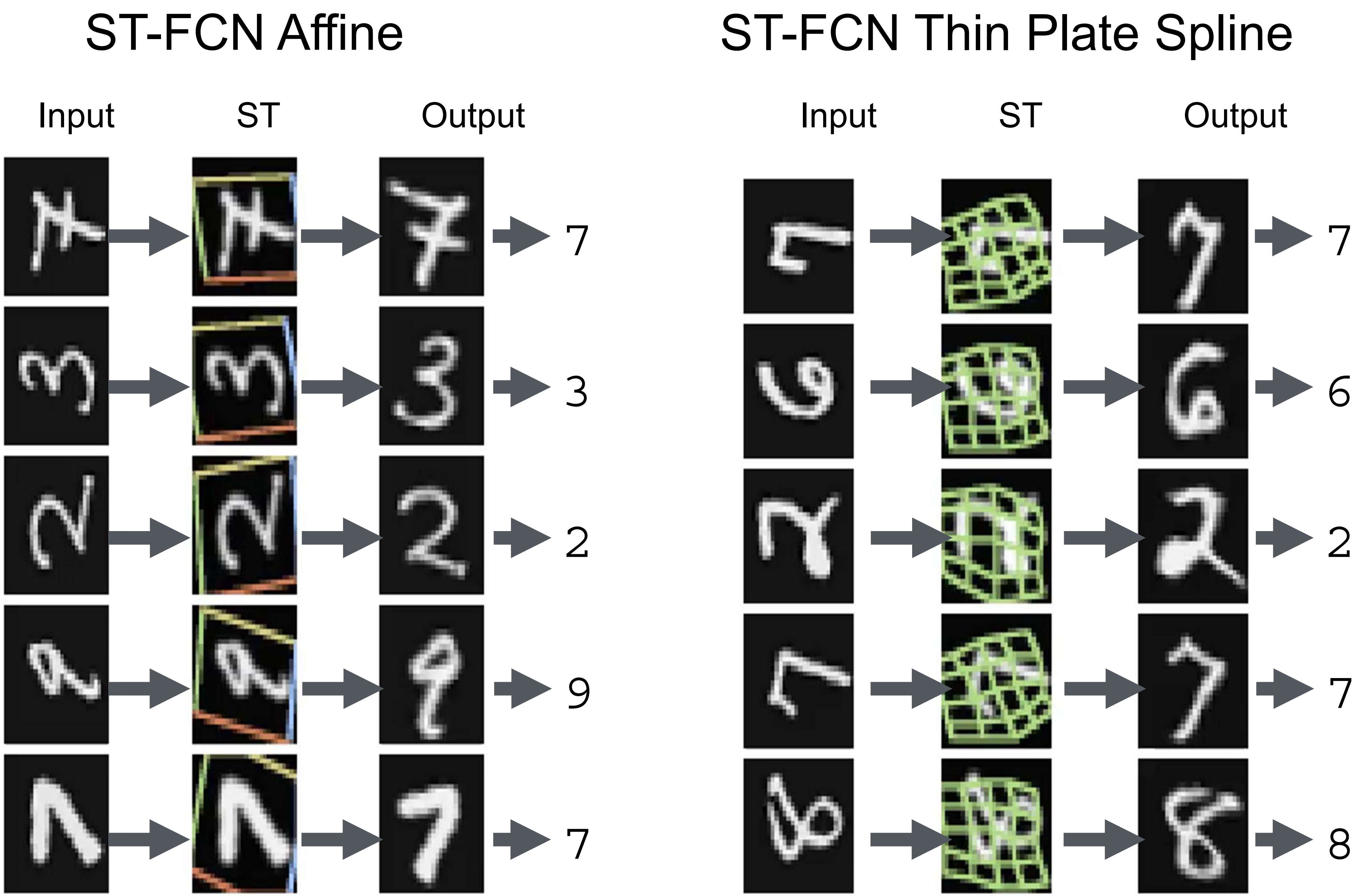
- FCN 2.1
- CNN 1.2
- ST-FCN 1.2
- ST-CNN 0.7

Generalizations 1: transformations

- Affine transformation – 6 parameters
- Projective transformation – 8 parameters
- Thin plate spline transformation
- Etc

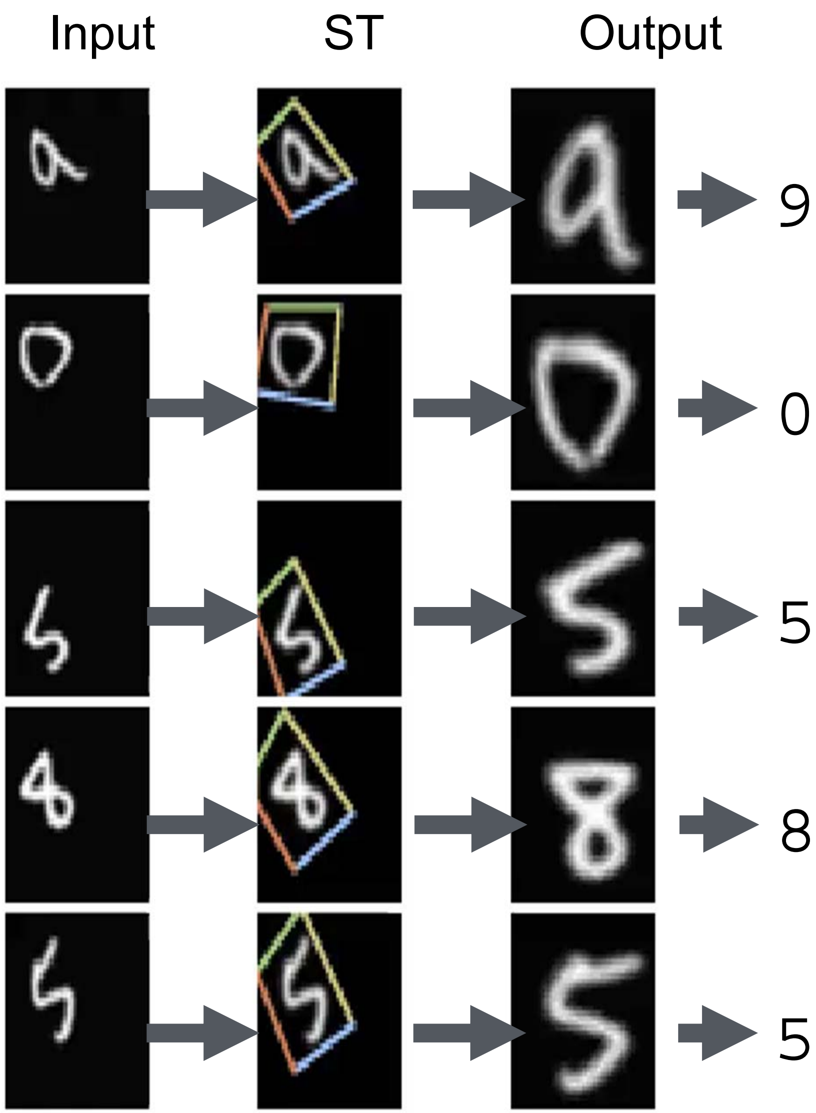
Any transformation where parameters can be regressed

Rotated MNIST

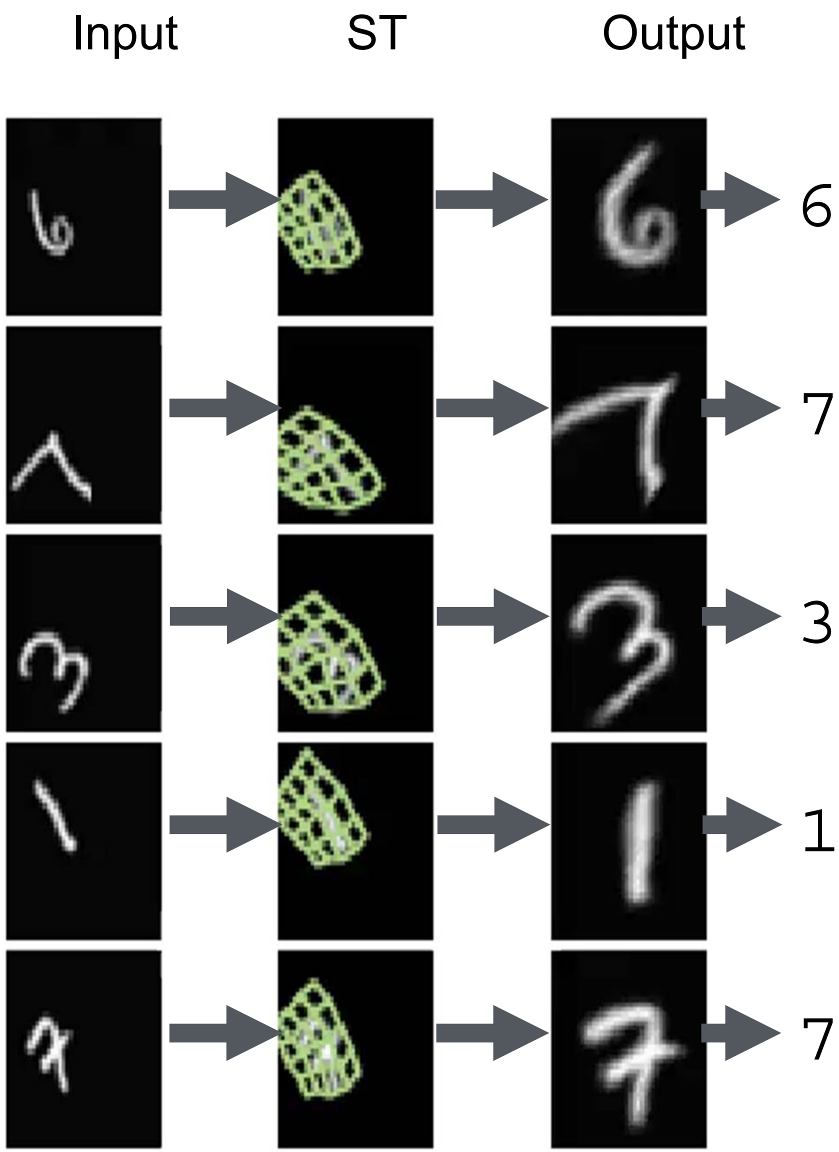


Rotated, Translated & Scaled MNIST

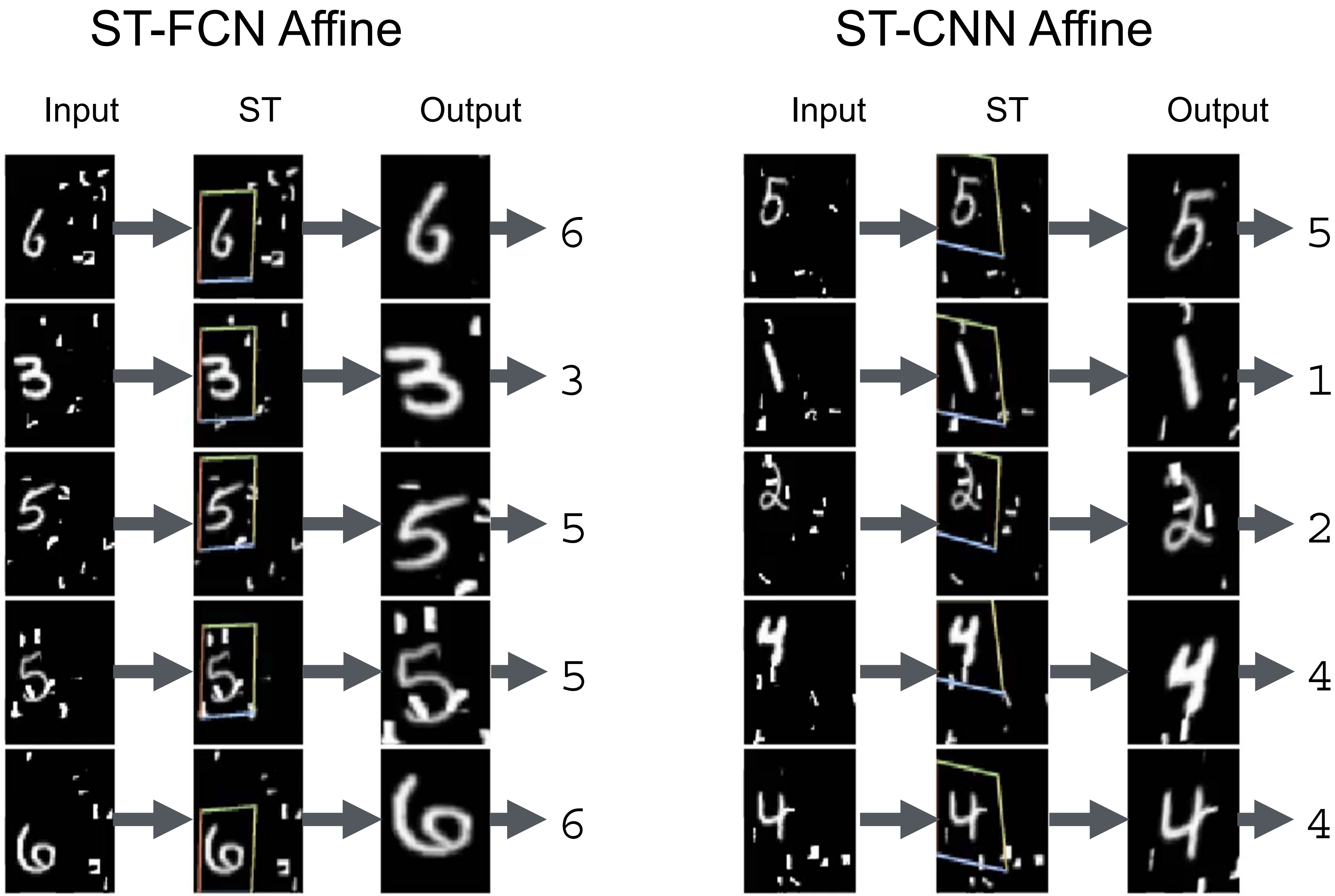
ST-FCN Projective



ST-FCN Thin Plate Spline

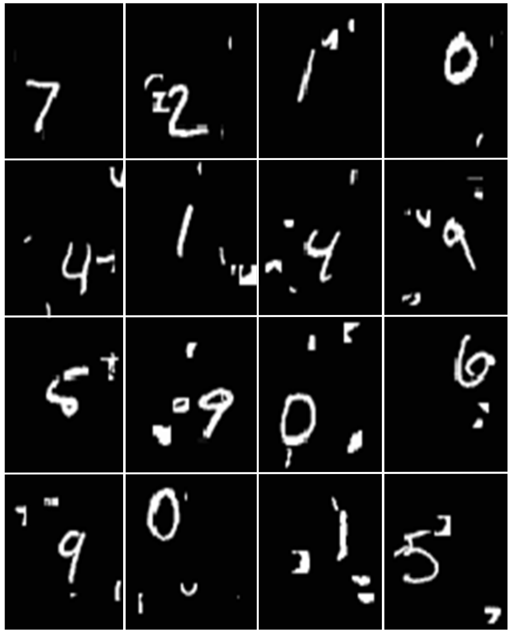


Translated Cluttered MNIST



Results on performance

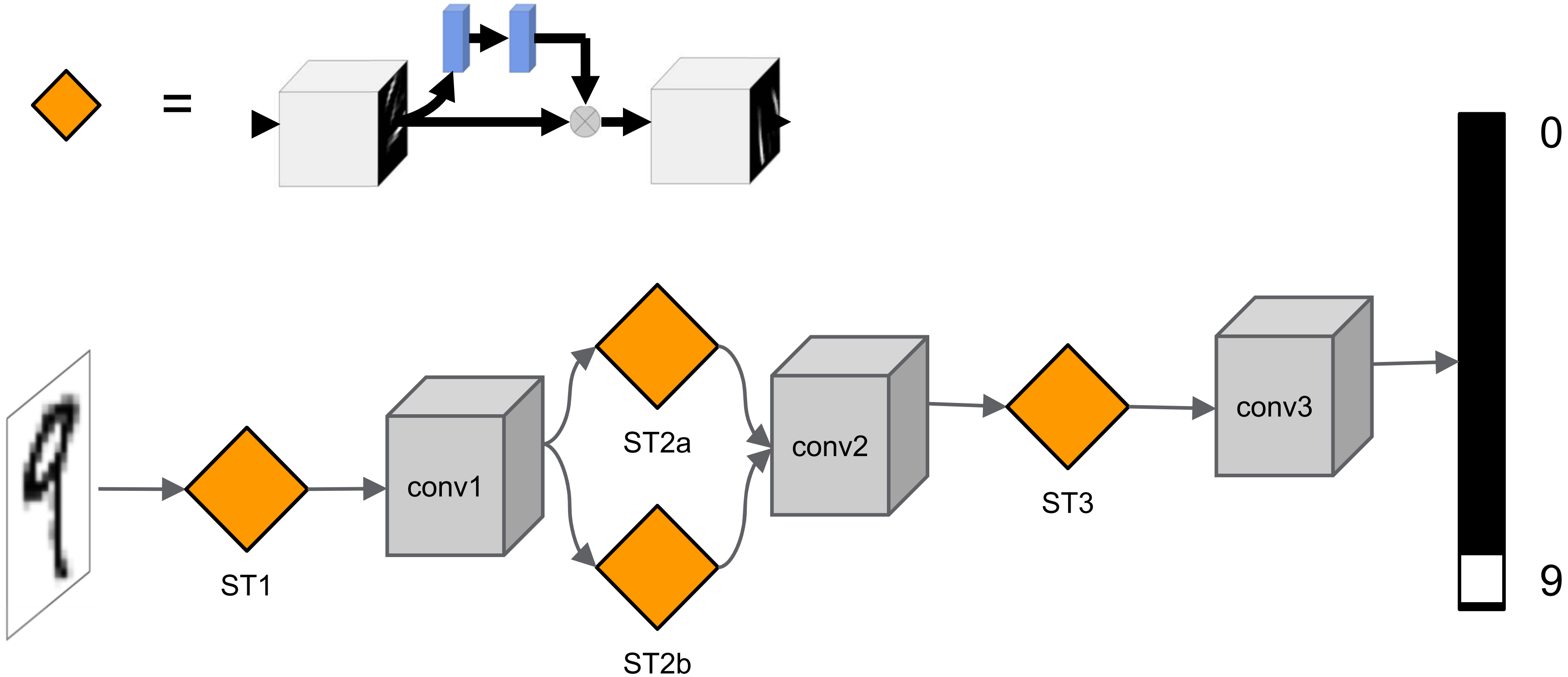
Model		MNIST Distortion R	RTS	P	E
FCN		2.1	5.2	3.1	3.2
CNN		1.2	0.8	1.5	1.4
ST-FCN	Aff	1.2	0.8	1.5	2.7
	Proj	1.3	0.9	1.4	2.6
	TPS	1.1	0.8	1.4	2.4
ST-CNN	Aff	0.7	0.5	0.8	1.2
	Proj	0.8	0.6	0.8	1.3
	TPS	0.7	0.5	0.8	1.1



R: rotation
RTS: rotation, translation, scale
P: projective
E: elastic

Generalization 2: Multiple spatial transformers

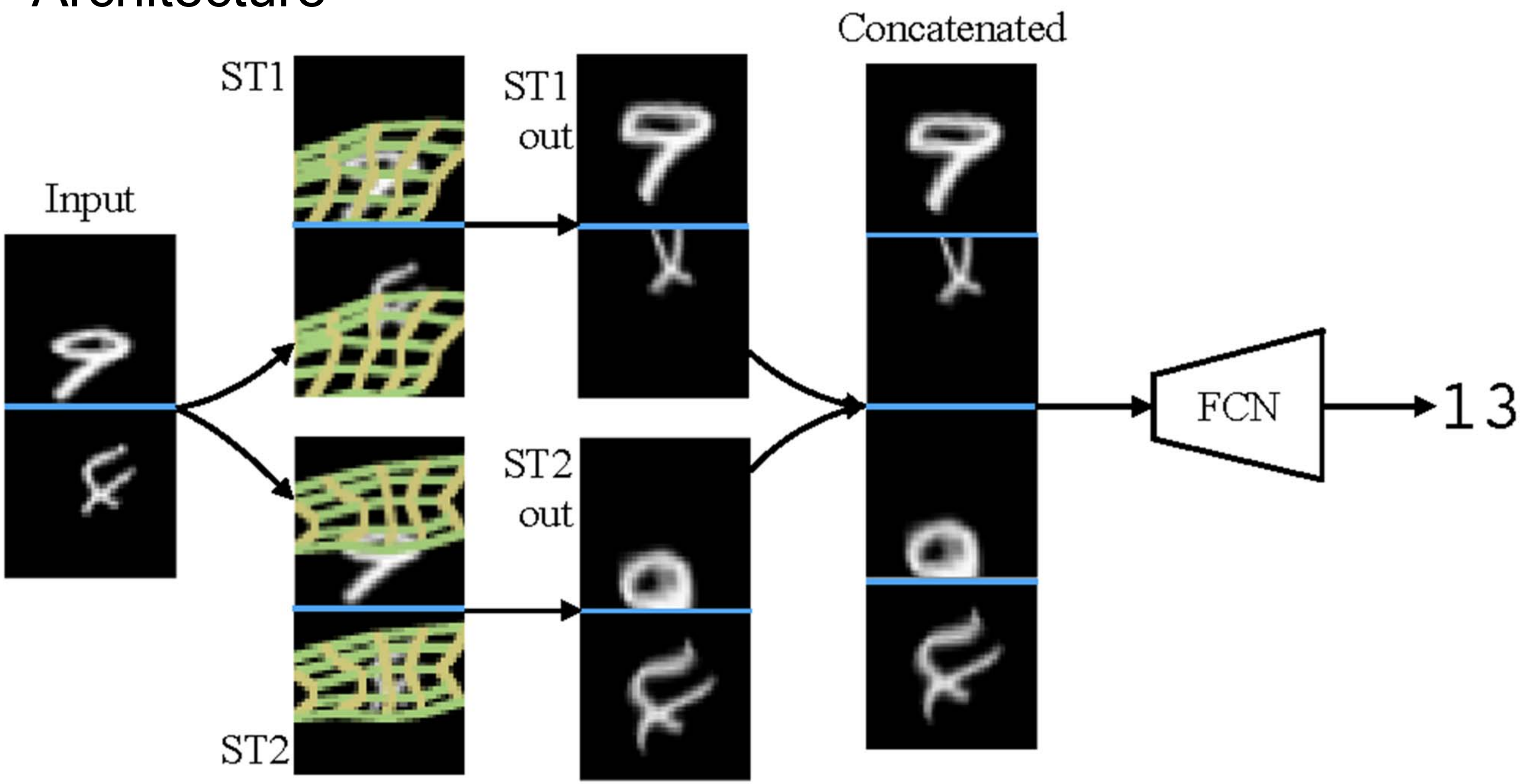
- Spatial Transformers can be inserted before/after conv layers, before/after max-pooling
- Can also have multiple Spatial Transformers at the same level



Task: Add digits in two images

MNIST digits under rotation, translation and scale

Architecture



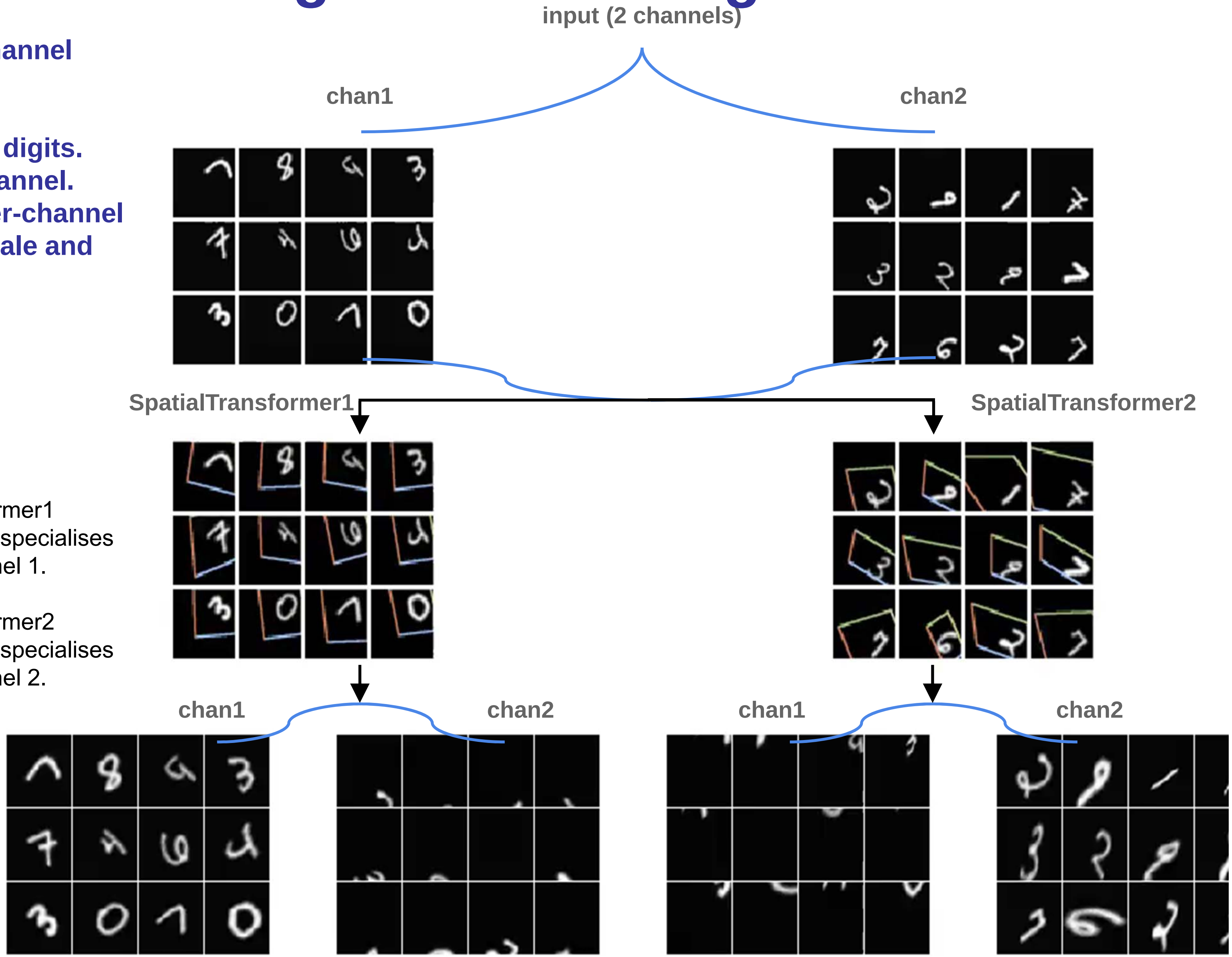
Task: Add digits in two images

MNIST 2-channel addition

Add up the digits. One per channel. Random per-channel rotation, scale and translation.

SpatialTransformer1 **automatically** specialises to rectify channel 1.

SpatialTransformer2 **automatically** specialises to rectify channel 2.

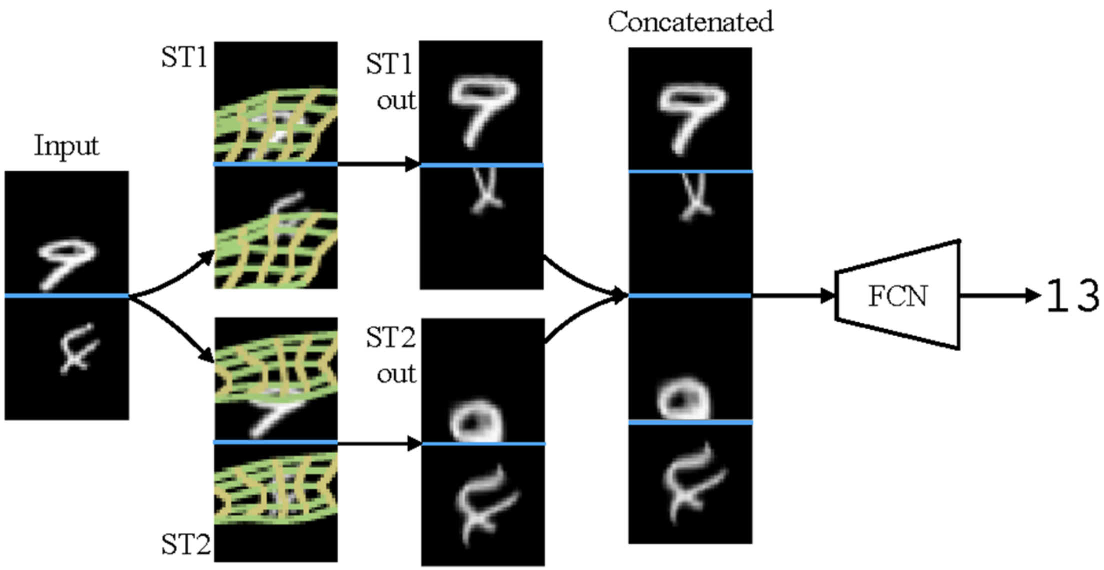


Task: Add digits in two images

MNIST digits under rotation, translation and scale

Performance % error

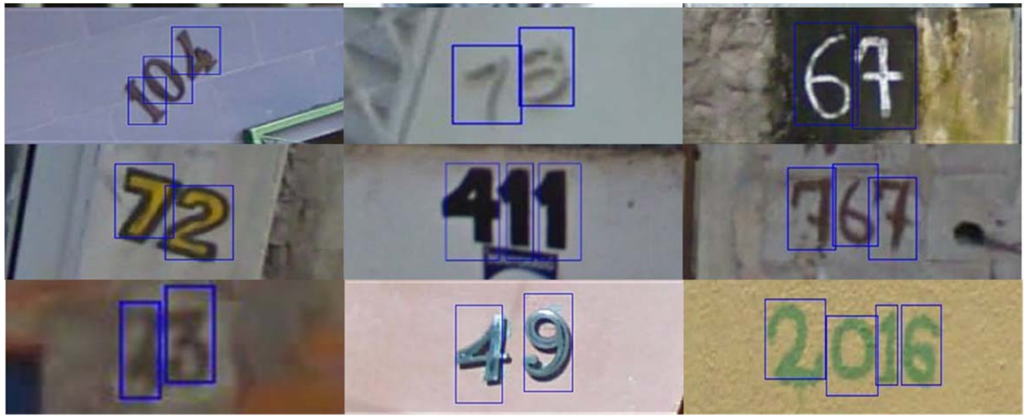
Model		
FCN		47.7
CNN		14.7
ST-FCN	Aff	22.6
	Proj	18.5
	TPS	19.1
2×ST-FCN	Aff	9.0
	Proj	5.9
	TPS	5.8



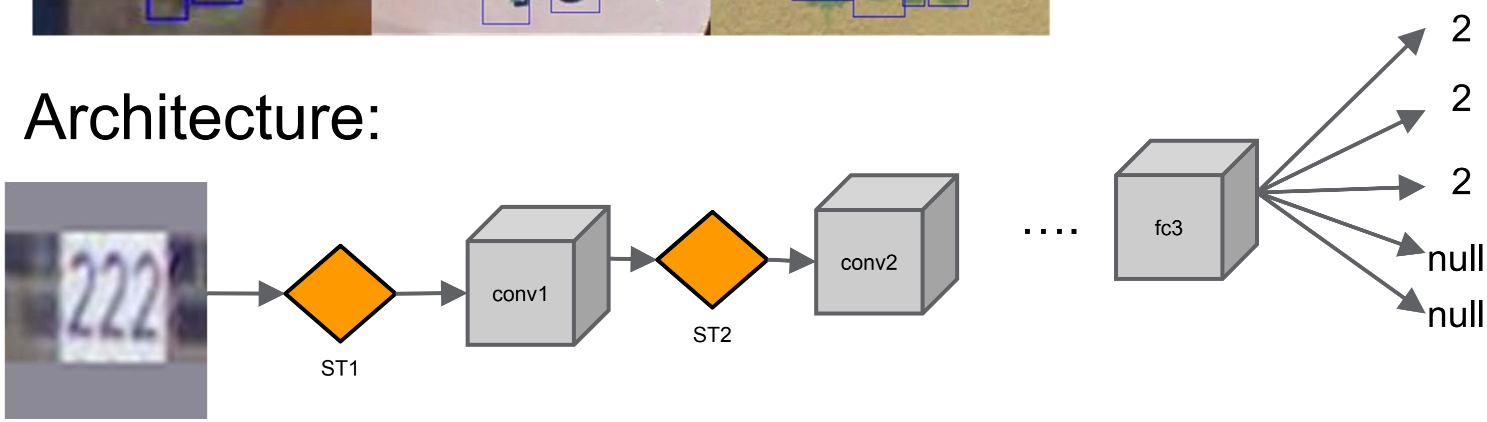
Applications
and comparisons with the state of the art

Street View House Numbers (SVHN)

200k real images of house numbers collected from Street View
Between 1 and 5 digits in each number



Architecture:



4 spatial transformer + conv layers, 4 conv layers, 3 fc layers, 5 character output layers

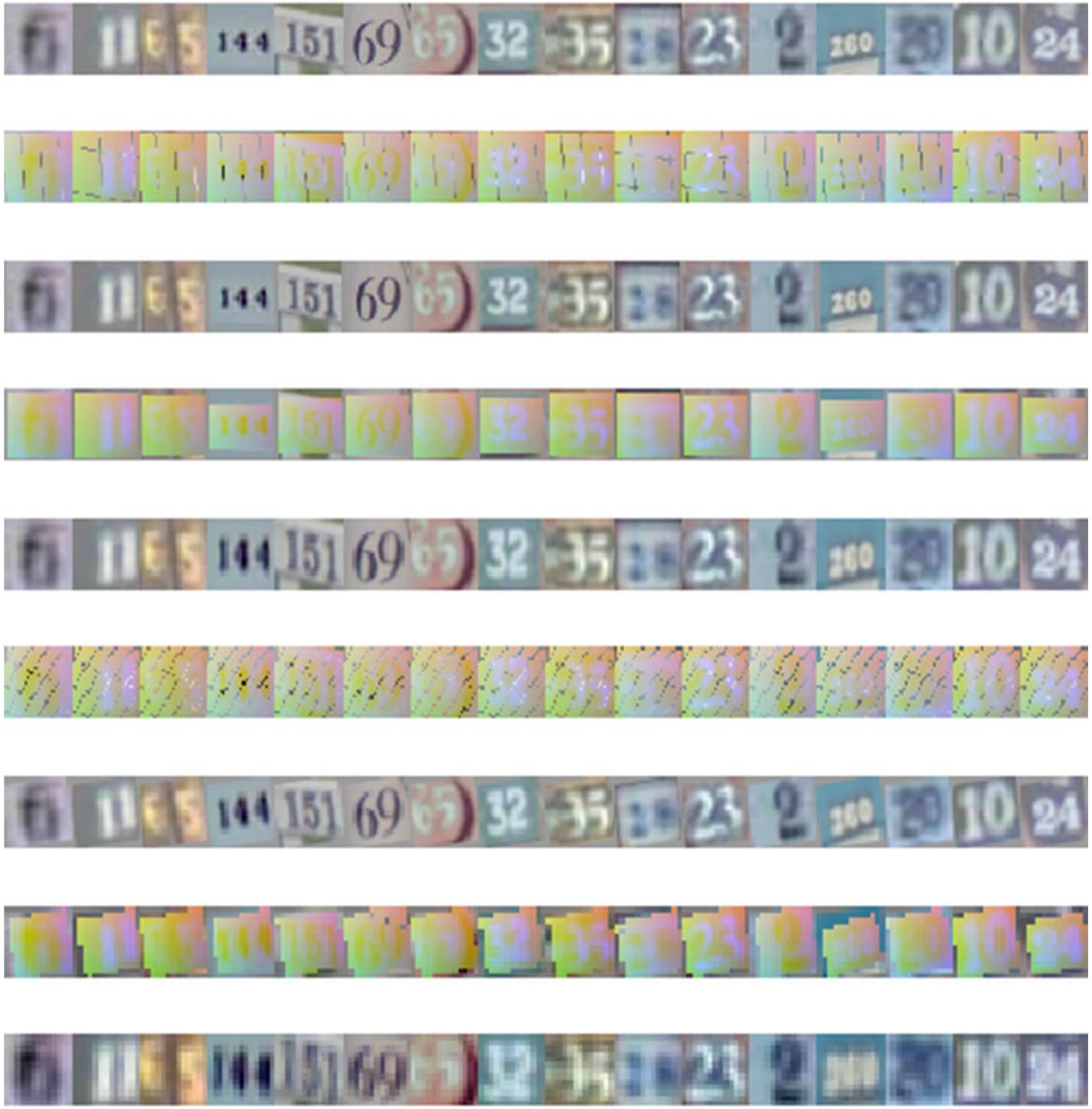
SVHN 64x64

- CNN: 4.0% error
  - (single model) Goodfellow et al 2013

- Attention: 3.9% error
  - (ensemble with MC averaging)

Ba et al, ICLR 2015

- **ST net: 3.6% error**
  - (single model)



SVHN 128x128

- CNN: 5.6% error
  - (single model)
- Attention: 4.5% error
  - (ensemble with MC averaging)

Ba et al, ICLR 2015

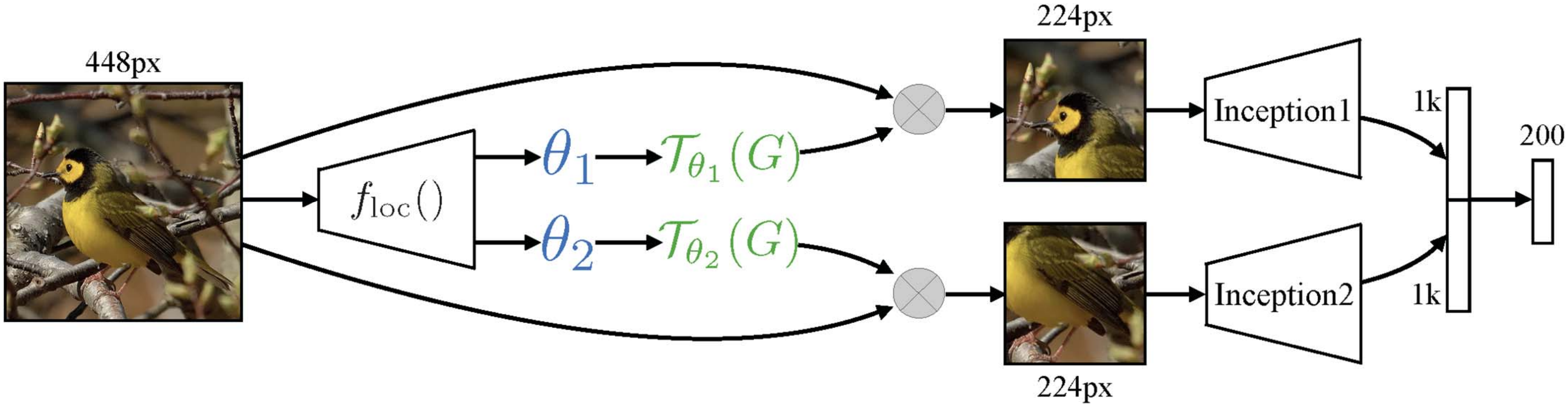
- **ST net: 3.9% error**
  - (single model)

Fine Grained Visual Categorization

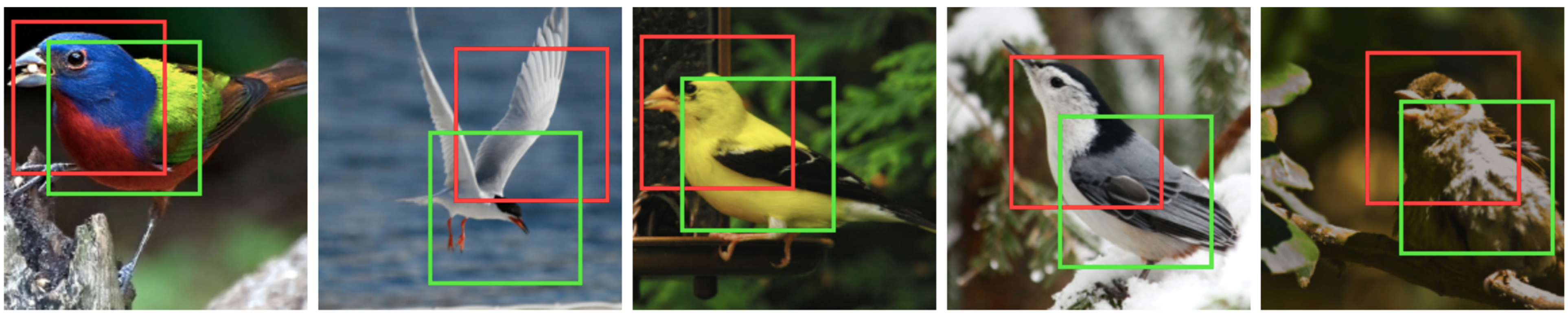
CUB-200-2011 birds dataset

- 200 species of birds
- 6k training images
- 5.8k test images

Spatial Transformer Network

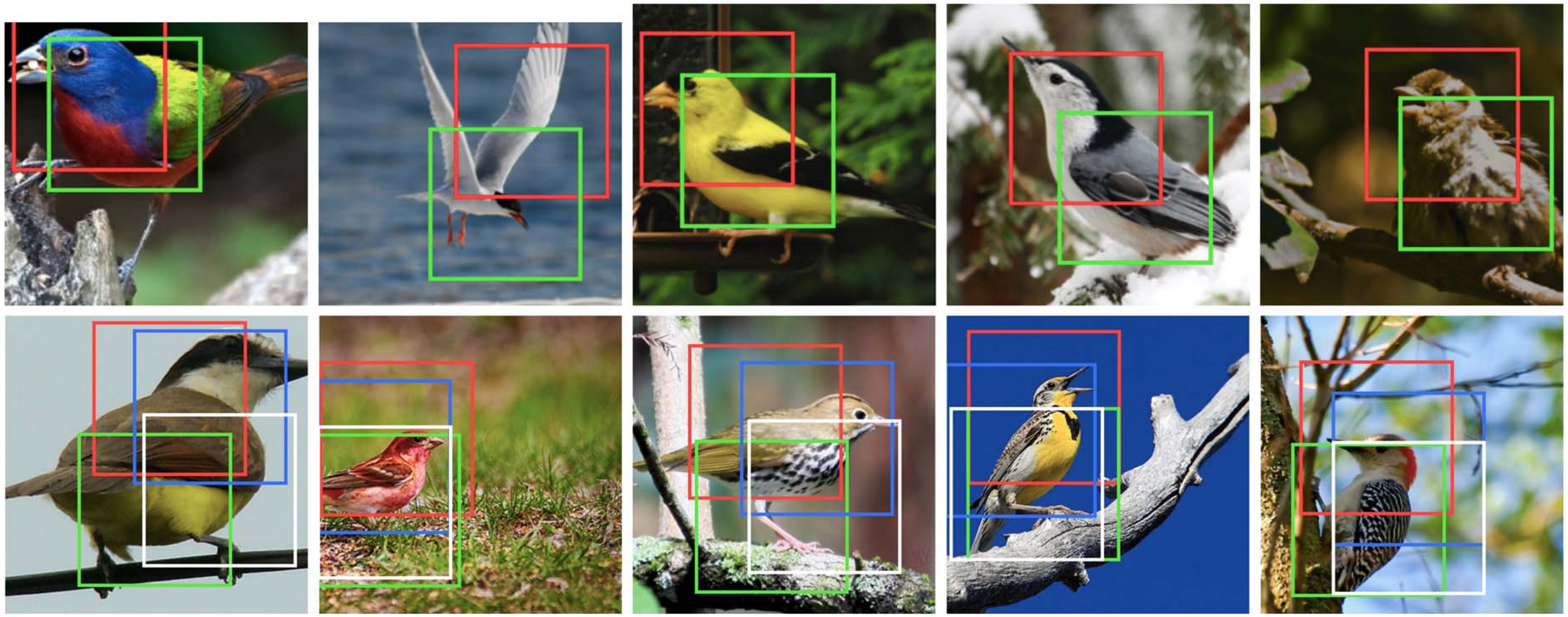


- Pre-train inception networks on ImageNet
- Train spatial transformer network on fine grained multi-way classification



CUB Performance

Model	
Cimpoi '15 [4]	66.7
Zhang '14 [30]	74.9
Branson '14 [2]	75.7
Lin '15 [20]	80.9
Simon '15 [24]	81.0
CNN (ours) 224px	82.3
2×ST-CNN 224px	83.1
2×ST-CNN 448px	83.9
4×ST-CNN 448px	**84.1**



Summary

- Spatial Transformers allow dynamic, conditional cropping and warping of images/feature maps.

- Can be constrained and used as very fast attention mechanism.

- Spatial Transformer Networks localise and rectify objects automatically. Achieve state of the art results.

- Can be used as a generic localisation mechanism which can be learnt with backprop.