CMPT 880 Spring 2019 Topic Presentation
February 26, 2019

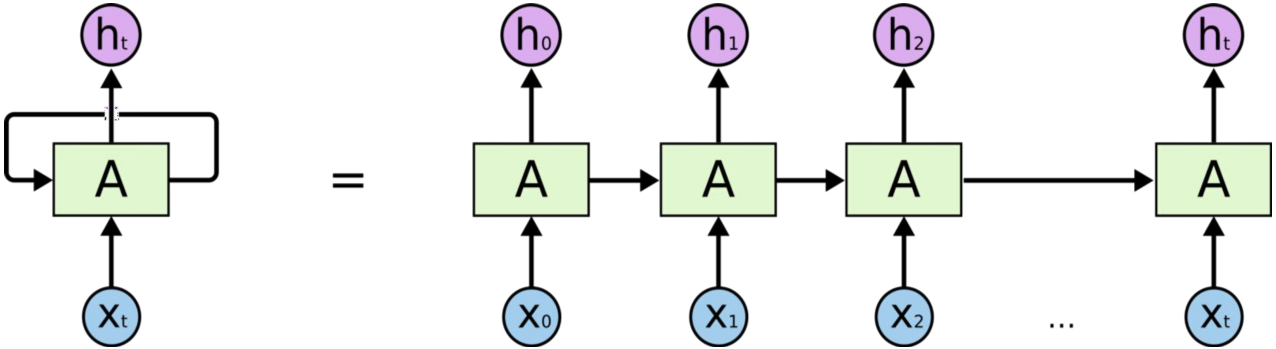
Attention Models

Kumar Abhishek and **Nishant Kambhatla**

Attention in NLP

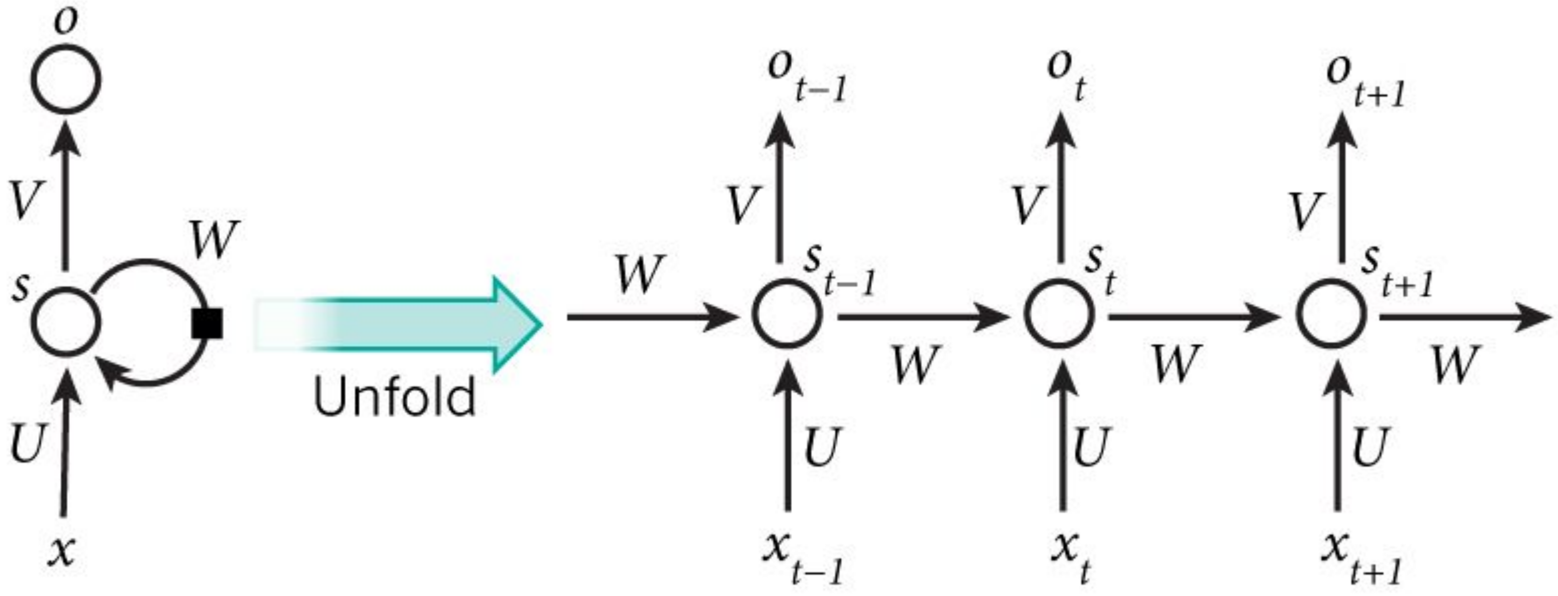
Recurrent Neural Networks (RNNs)

- **Connections between nodes form a directed graph along the sequence**



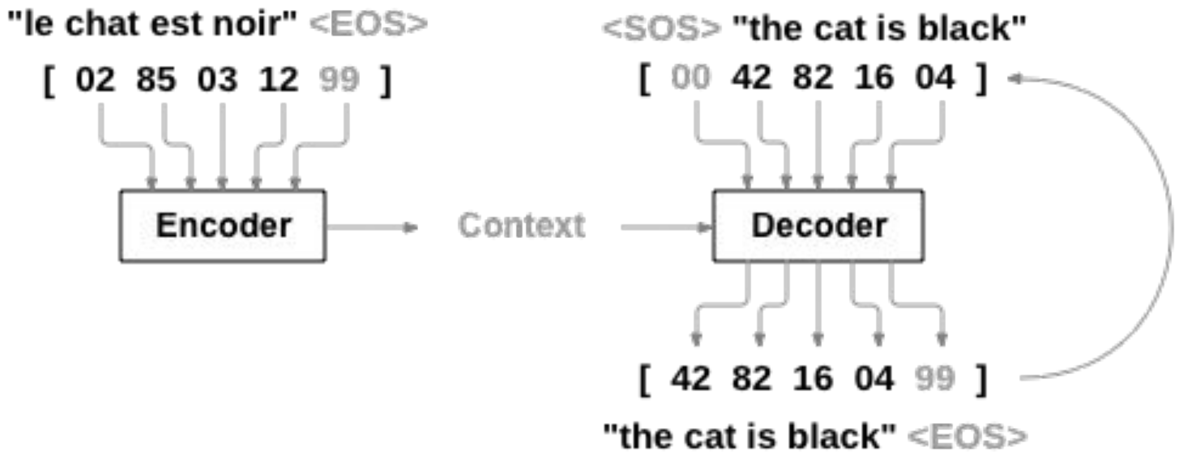
- **Use their internal state (memory) to process sequence of inputs**

Recurrent Neural Networks (RNNs)



Encoder-Decoder

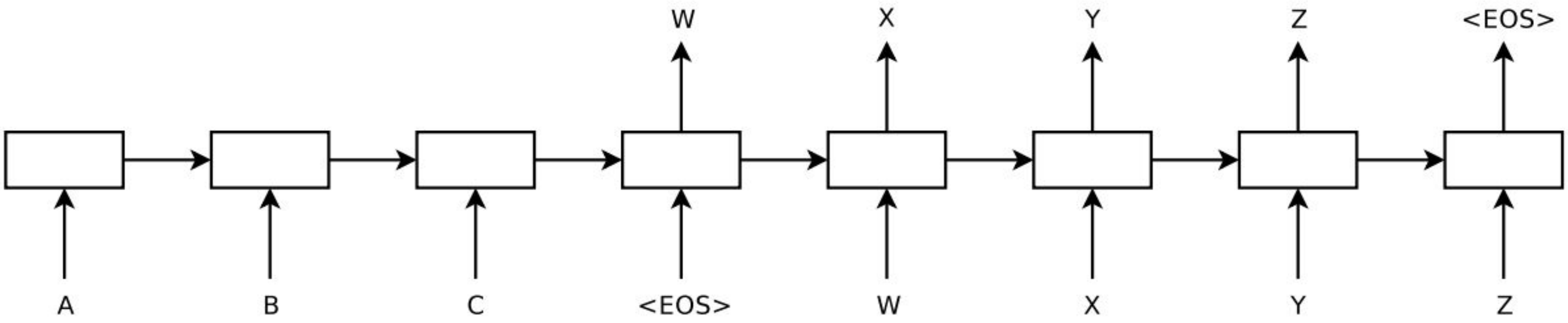
Several applications; predominantly used for Neural Machine Translation (NMT)



Cho, Kyunghyun, et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

Encoder-Decoder

Several applications; predominantly used for Neural Machine Translation (NMT)



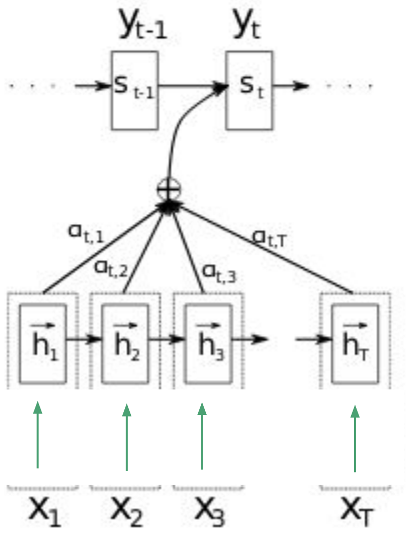
Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." *Advances in neural information processing systems*. 2014.

Encoder-Decoder

- **Encoder**
  - **Input sequence** $\mathbf{x} = (x_1, \cdots, x_{T_x})$
  - **Hidden state** $h_t = f(x_t, h_{t-1})$
  - **Encoded context** $c = q(\{h_1, \cdots, h_{T_x}\})$
- **Decoder**
  - **Probability** $p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c)$

Encoder-Decoder with Attention

- **Attention Decoder**



Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *ICLR* (2014).

Encoder-Decoder with Attention

- **Attention Decoder**
  - **Probability** $p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$

    Hidden state for time i $s_i = f(s_{i-1}, y_{i-1}, c_i)$
  - **Context Vector as weighted sum of hidden state** $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$
  - **Weights** $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$ **where** $e_{ij} = a(s_{i-1}, h_j)$

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *ICLR* (2014).

Encoder-Decoder with Attention

- **Attention Decoder**
  - **Probability** $$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

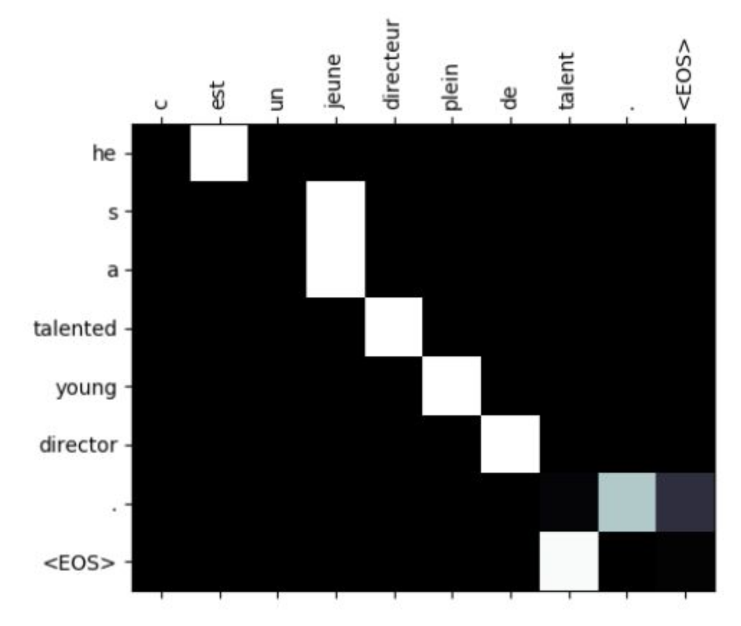
    Hidden state for time i $$s_i = f(s_{i-1}, y_{i-1}, c_i)$$
  - **Context Vector as weighted sum of hidden state** $$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$
  - **Weights** $$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$ **where** $$e_{ij} = a(s_{i-1}, h_j)$$

encoder

En

-

*alignment model which scores how well the inputs around position j and the output at position i match



$c_i)$

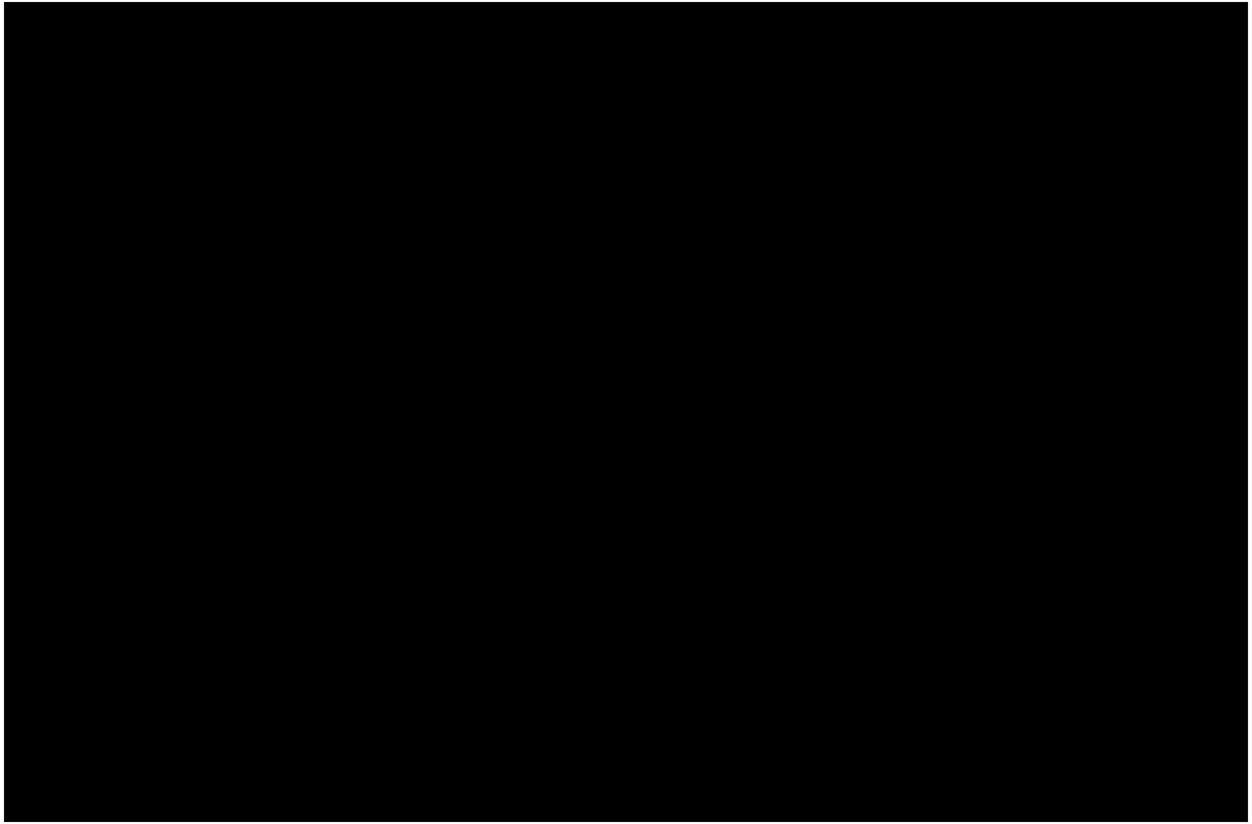
$c_i)$

$j=1$

- Weights $$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$ where $$e_{ij} = a(s_{i-1}, h_j)$$

encoder

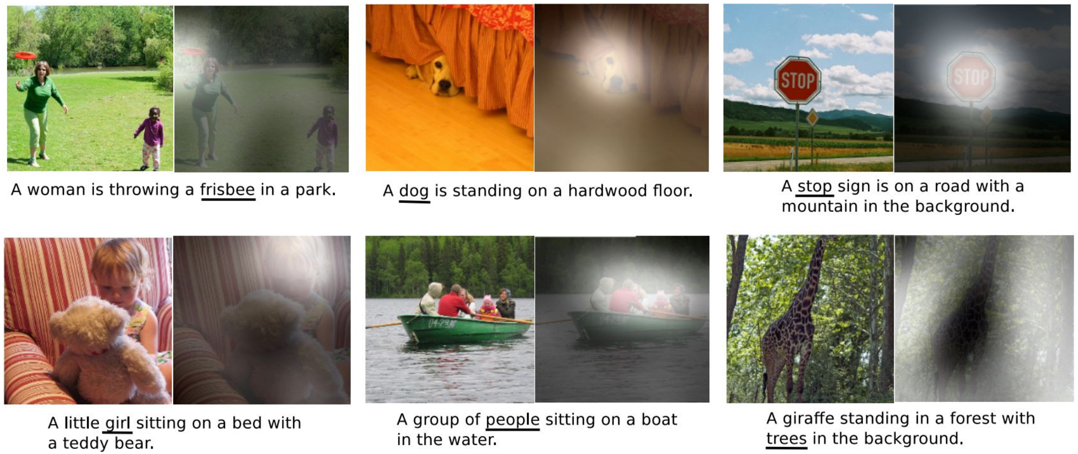
Demo



Attention in Vision

Self Attention

An attention mechanism relating different positions (in space, time, or space-time) of a single sequence in order to compute a representation of the same sequence.



Example: Image Captioning

Kelvin Xu et al., *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*, ICML 2015

Two papers

- **Non-local Neural Networks**

Wang et al., “*Non-local Neural Networks*”, CVPR 2018 [157 citations]

- **Spatial Transformer Networks**

Jaderberg et al., “*Spatial Transformer Networks*”, NeurIPS 2015 [1362 citations]

Non-local Neural Networks

- Inspired by non-local means image denoising*
- Every patch in the image can be expressed as a weighted sum of itself and many other patches in the image.

$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j)$$

x: input signal (image/sequence/video/their features)

y: output signal

g(.): representation of input signal at position j

f(.): computing a scalar (e.g. affinity) between i and j

C(x): normalizing constant

*Antoni Buades et al., *A non-local algorithm for image denoising*, CVPR 2005

Non-local Neural Networks

- Non-local behaviour because all positions (j) considered.
  - In a convolution operation, only a local neighborhood is considered.
  - In a recurrent operation, only the current and the latest time steps are considered.

- Relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$ is based on the relationship between the two locations, and is therefore a function of the input data.

Non-local Neural Networks

Choice of g(.)

$g(\mathbf{x}_j) = W_g \mathbf{x}_j$ **W_g** is a weight matrix to be learned.

Choice of f(.)

- Gaussian → $f(\mathbf{x}_i, \mathbf{x}_j) = e^{\mathbf{x}_i^T \mathbf{x}_j}$
- Embedded Gaussian → $f(\mathbf{x}_i, \mathbf{x}_j) = e^{\theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)}$
- Dot Product → $f(\mathbf{x}_i, \mathbf{x}_j) = \theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$
- Concatenation → $f(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{ReLU}(\mathbf{w}_f^T [\theta(\mathbf{x}_i),\ \phi(\mathbf{x}_j)])$

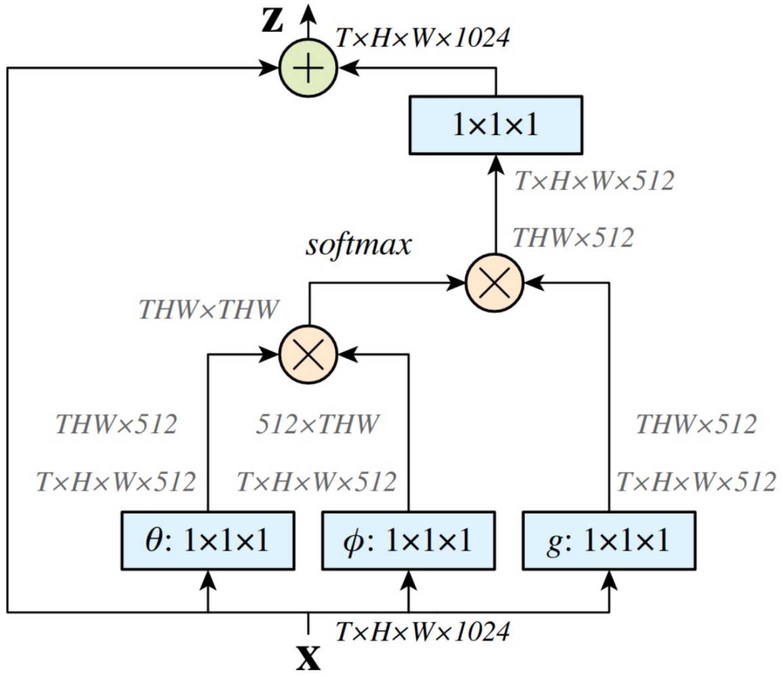
$\theta(\mathbf{x}_i) = W_\theta \mathbf{x}_i$ and $\phi(\mathbf{x}_j) = W_\phi \mathbf{x}_j$ are embedding functions.

Non-local Neural Networks

Non-local block

$$\mathbf{z}_i = W_z \mathbf{y}_i + \mathbf{x}_i$$

Since "+$\mathbf{x}_i$" denotes a residual connection, this non-local block can be inserted into many existing architectures.



Non-local Neural Networks

Finding clues to support prediction on the Kinetics human action video dataset.

[32-frame input shown with a stride of 4 frames.]

Non-local Neural Networks

Experiments and Results

- Tested on
  - Human action classification from videos (Kinetics and Charades datasets)
  - Object detection and segmentation and keypoint detection (MS COCO dataset)

- Addition of non-local blocks results in a solid improvement over baseline performances.

Spatial Transformer Networks

Motivation

CNNs have interleaved CONV layers and POOL layers, leading to partial spatial (translation) invariance.

- Only small invariances per POOL layer
  - Thus spatial invariance to a limited number of transformations.
- NOT invariant to scaling and rotation.

Demo

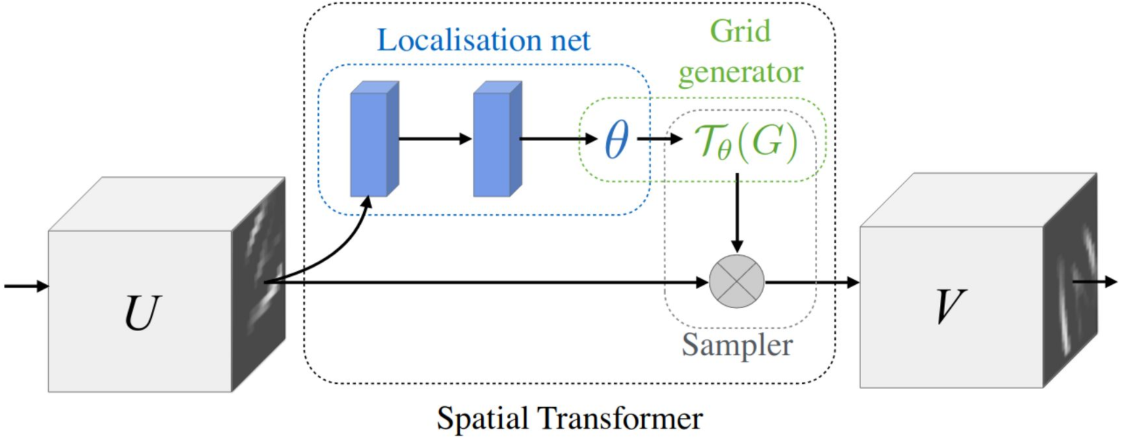
Spatial Transformer Networks

Intuition: Conditional Spatial Warping

- Transform the input to a space that is optimal for the subsequent layers.
- Select features of interest to attend to.
- Increase robustness to more categories of transformations

Spatial Transformer Network Module

- Three **differentiable** blocks:
  - Localization Network
  - Grid Generator
  - Sampler



Spatial Transformer Network Module

Localisation Network:

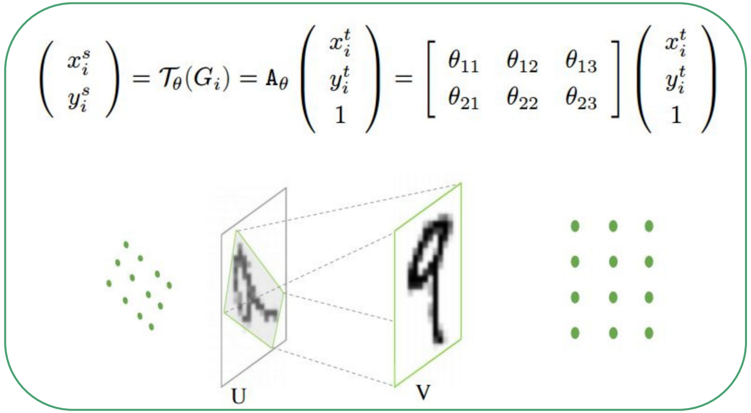
A “mini-network” consisting of say, 2 fully-connected layers.

Takes the feature maps as input and regresses the parameters of the transformation to be applied.

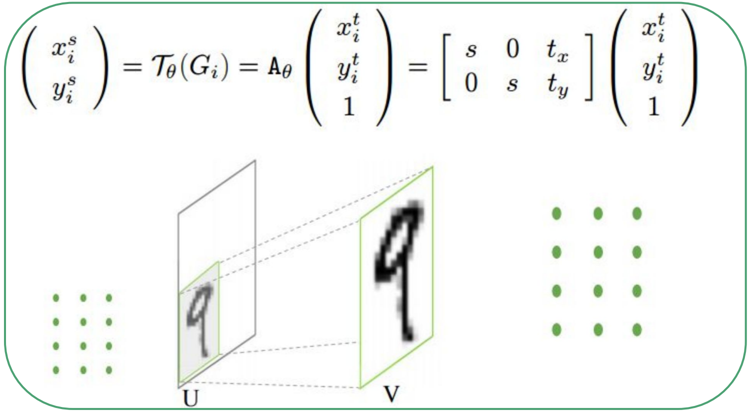
Spatial Transformer Network Module

Grid Generator:

Takes the transformation parameters θ and produces the sampling grid, mapping each pixel in the output to a corresponding pixel in the input.



Affine Transform

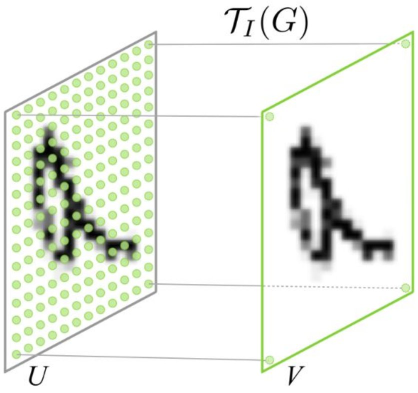


Attention Model

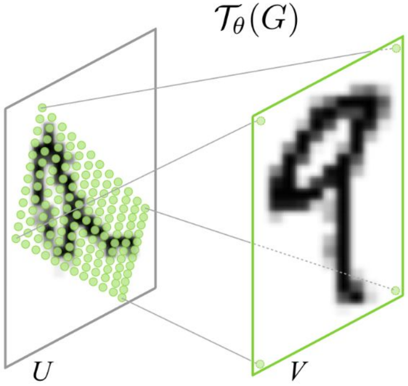
Spatial Transformer Network Module

Sampler:

Takes the sampling grid and applies it to the input, producing the transformed outputs.



Identity Mapping



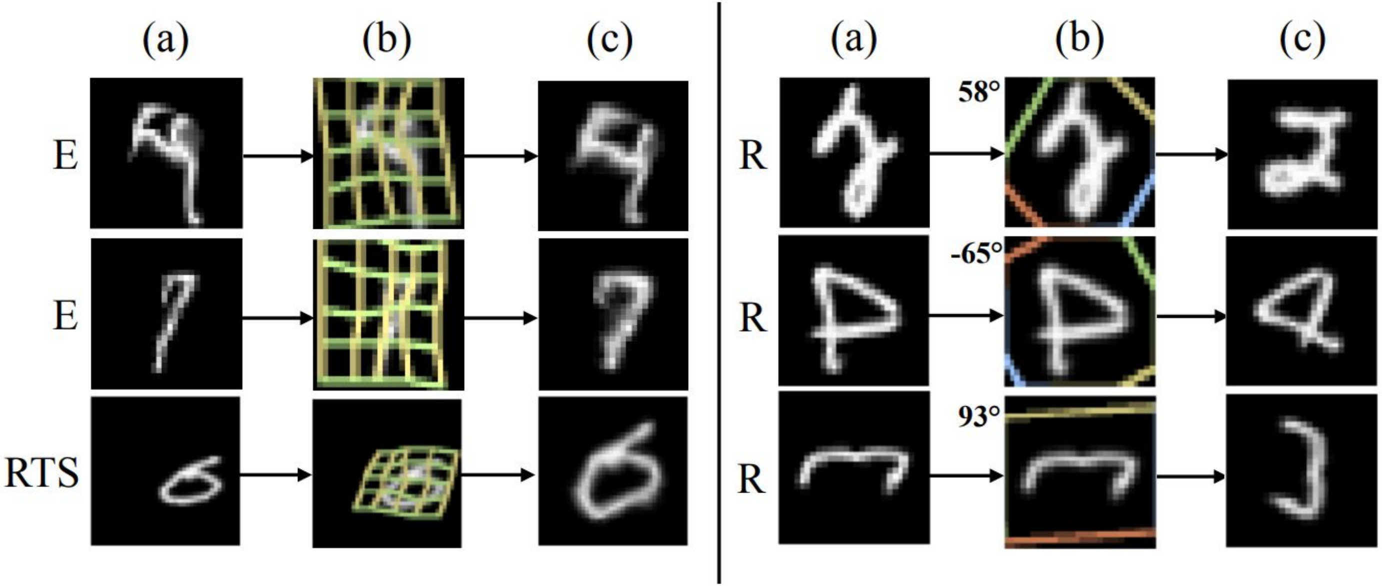
Affine Mapping

Spatial Transformer Networks

- Differentiable module.
  - Well defined gradients w.r.t the input.
  - Can be inserted anywhere in the network.
  - Can be used in parallel for fine grained classification.
- Tested on
  - Distorted MNIST dataset
  - SVHN (Street View House Numbers) dataset **[state-of-the-art results]**
  - CUB-200-2011 birds dataset **[state-of-the-art results]**

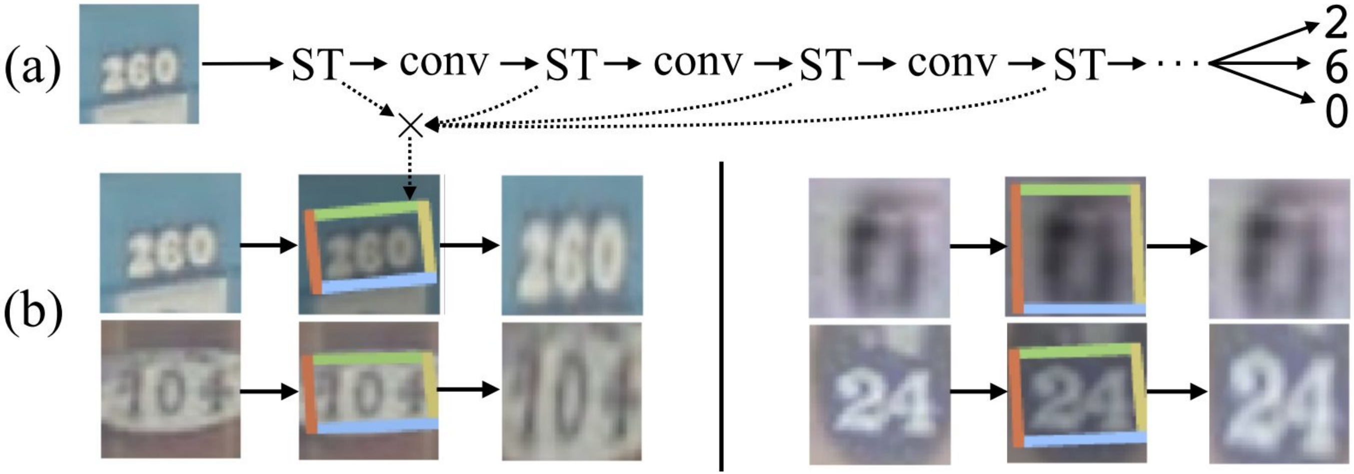
Spatial Transformer Networks

Distorted MNIST



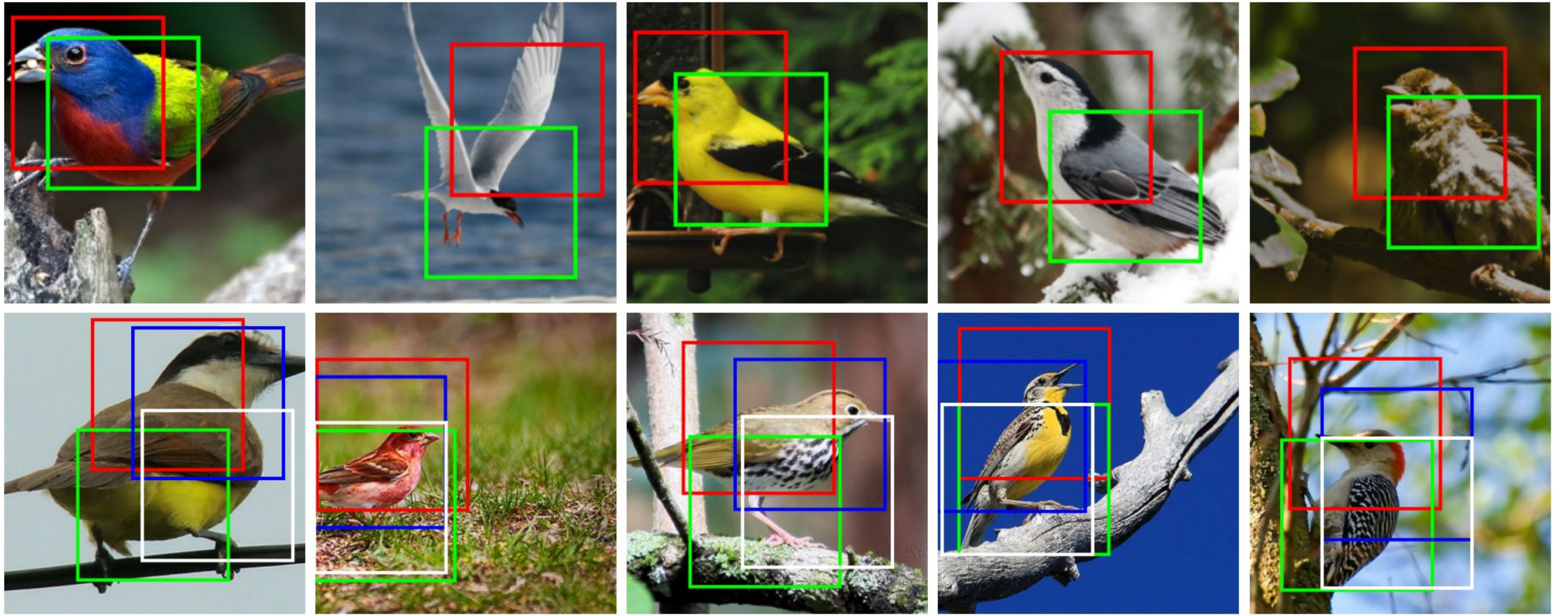
Spatial Transformer Networks

SVHN (4 STN modules)

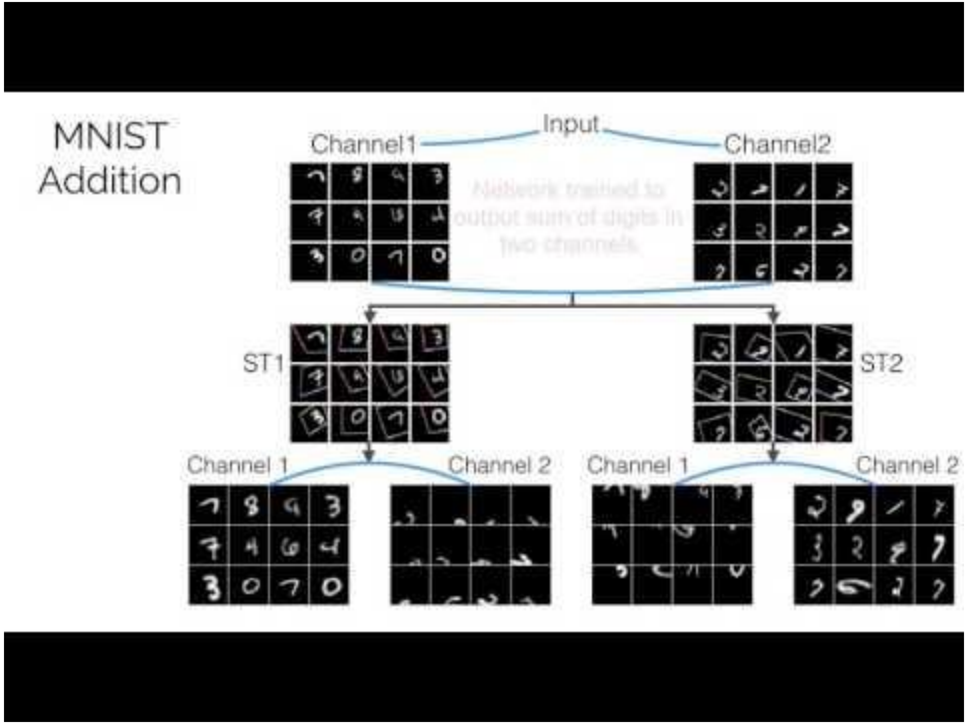


Spatial Transformer Networks

Birds Dataset (2/4 STN modules in parallel)



Spatial Transformer Networks



Thank You.