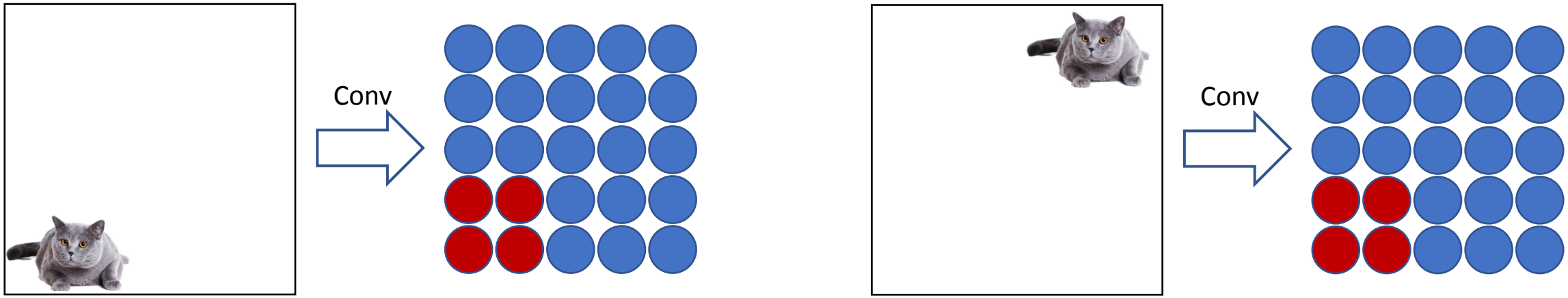
Beyond Convolutions in Deep Learning

Outline

- Проблемы простых сверток
- Spatial Transformer Networks (STN)
- Capsule Networks (CapsNet)

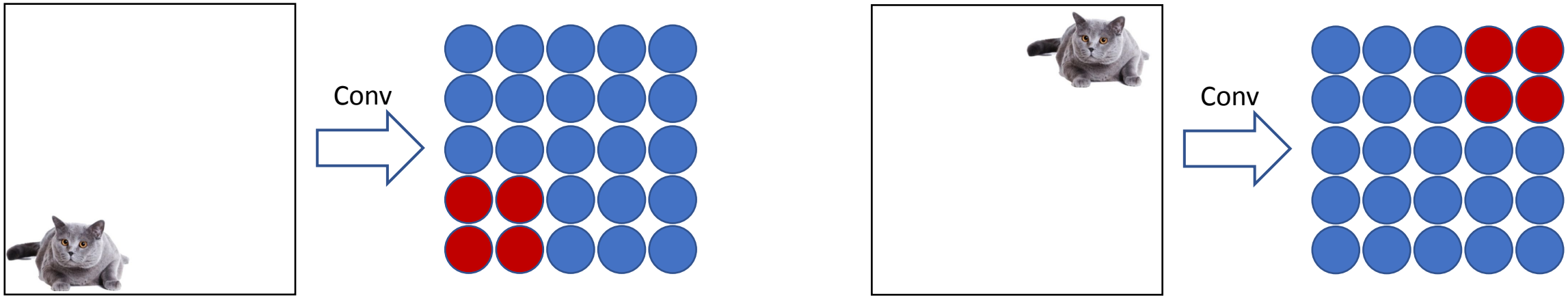
Пара определений

Инвариантность: $f\big(T(x)\big) = f(x)$



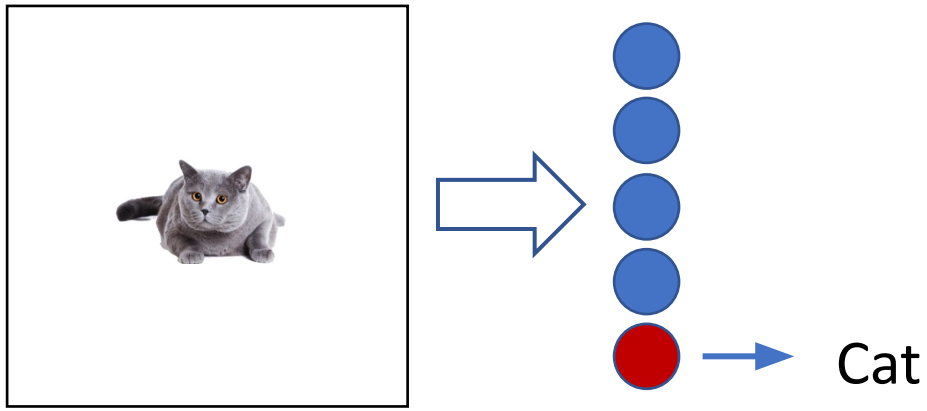
Пара определений

Эквивариантность: $f\big(T(x)\big) = T(f(x))$

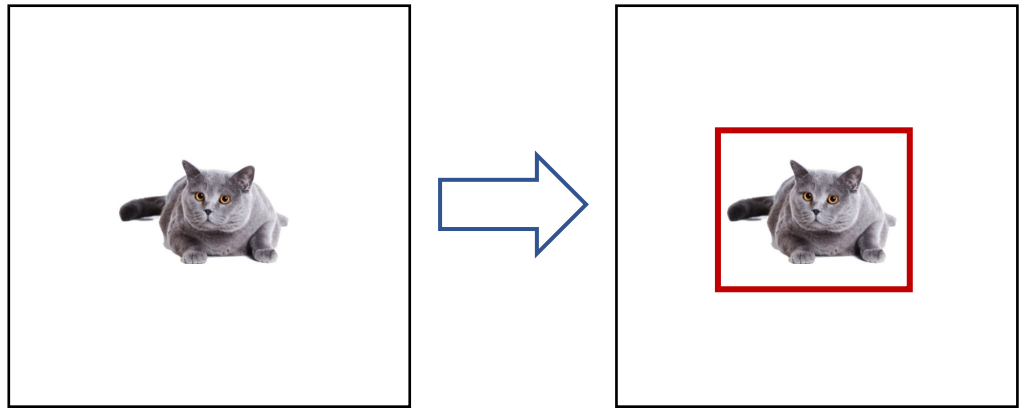


Зачем нам инвариантность / эквивариантность?

Чтобы классифицировать объект нужна инвариантность.



Чтобы сегментировать объект нужна эквивариантность.



Проблемы простых сверток

Свертки эквивариантны к сдвигам.

Q: Как сделать свертки инвариантными к малым сдвигам?

Использовать Pooling.

Проблемы простых сверток

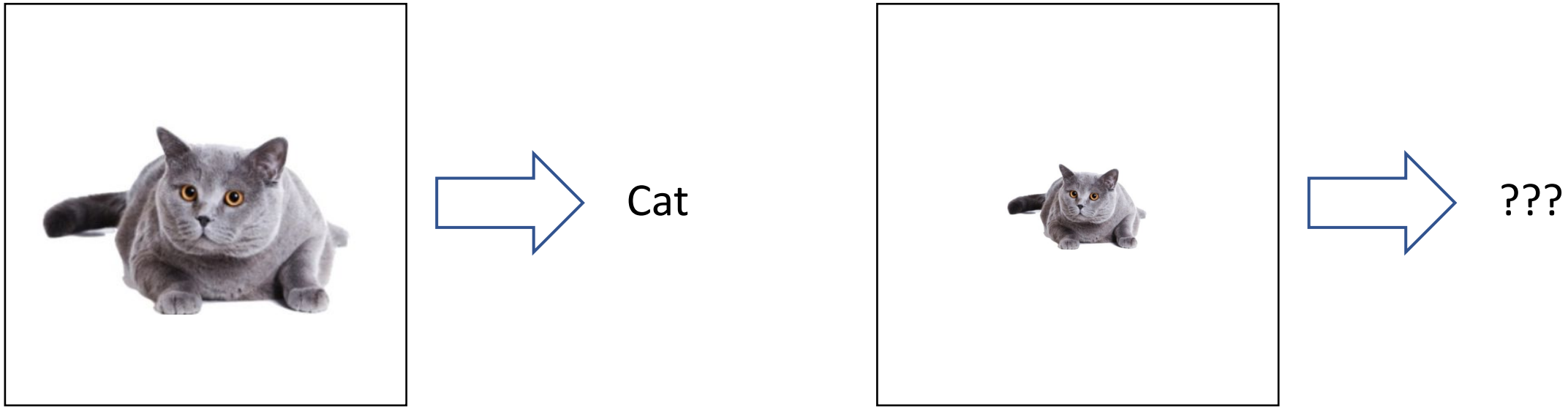
Повороты

Cat

???

Проблемы простых сверток

Изменение масштаба



Проблемы простых сверток

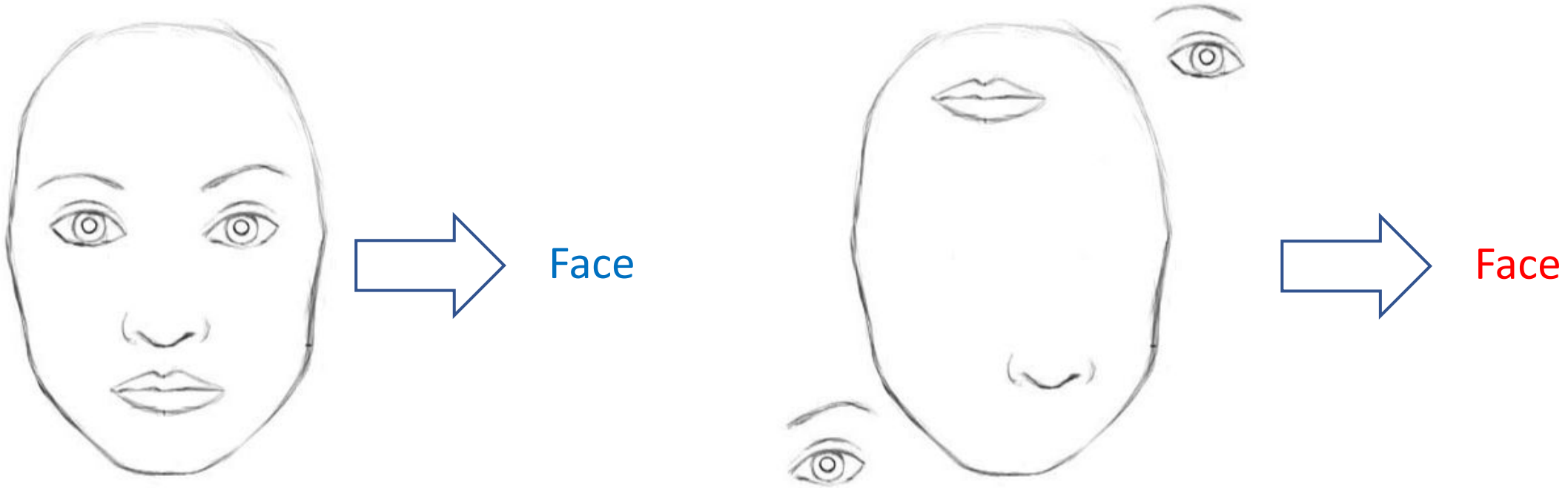
Сети инварианты к сдвигам, но не инвариантны к поворотам и изменению масштаба.

Q: Как сделать сети инвариантными к поворотам и изменению масштаба?

Использовать Data Augmentation.

Проблемы простых сверток

Изменение позиций компонентов
объекта относительно друг друга



Source:
https://medium.com/ai^3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Spatial Transformer Networks

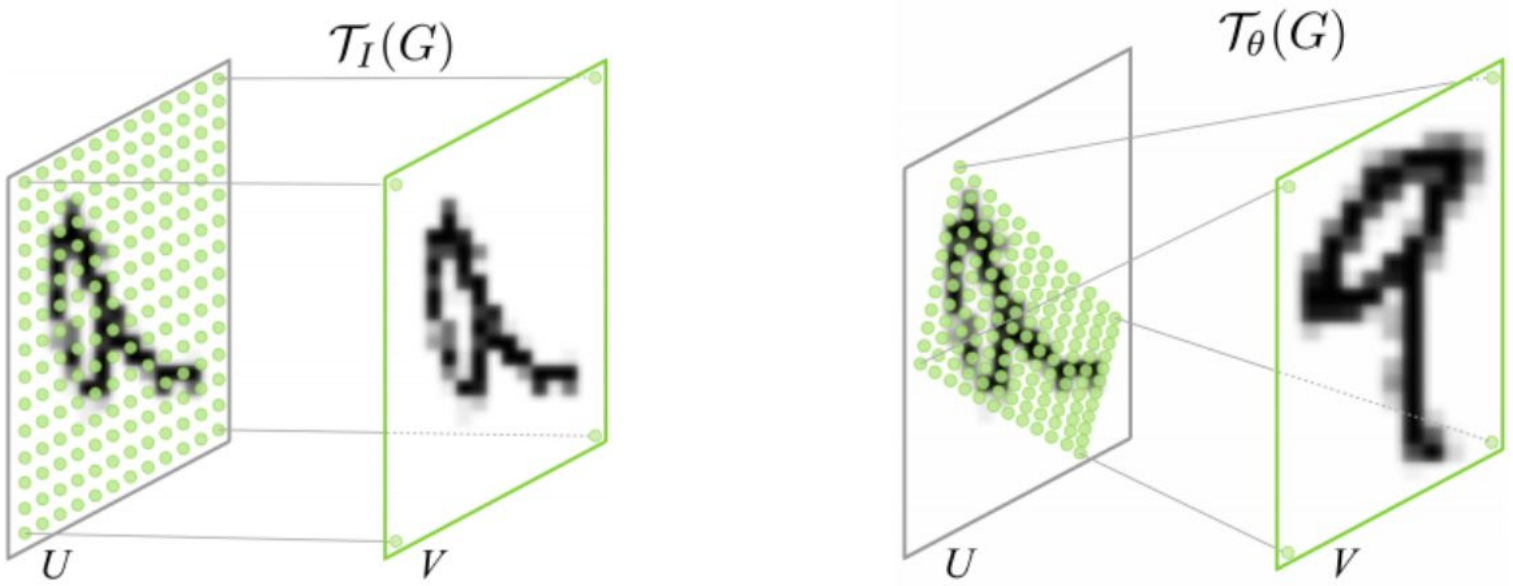
Идея:

Давайте прикрутим еще одну сеть, которая будет учиться трансформировать изображение таким образом, чтобы минимизировать ошибку классификации.

Суть:

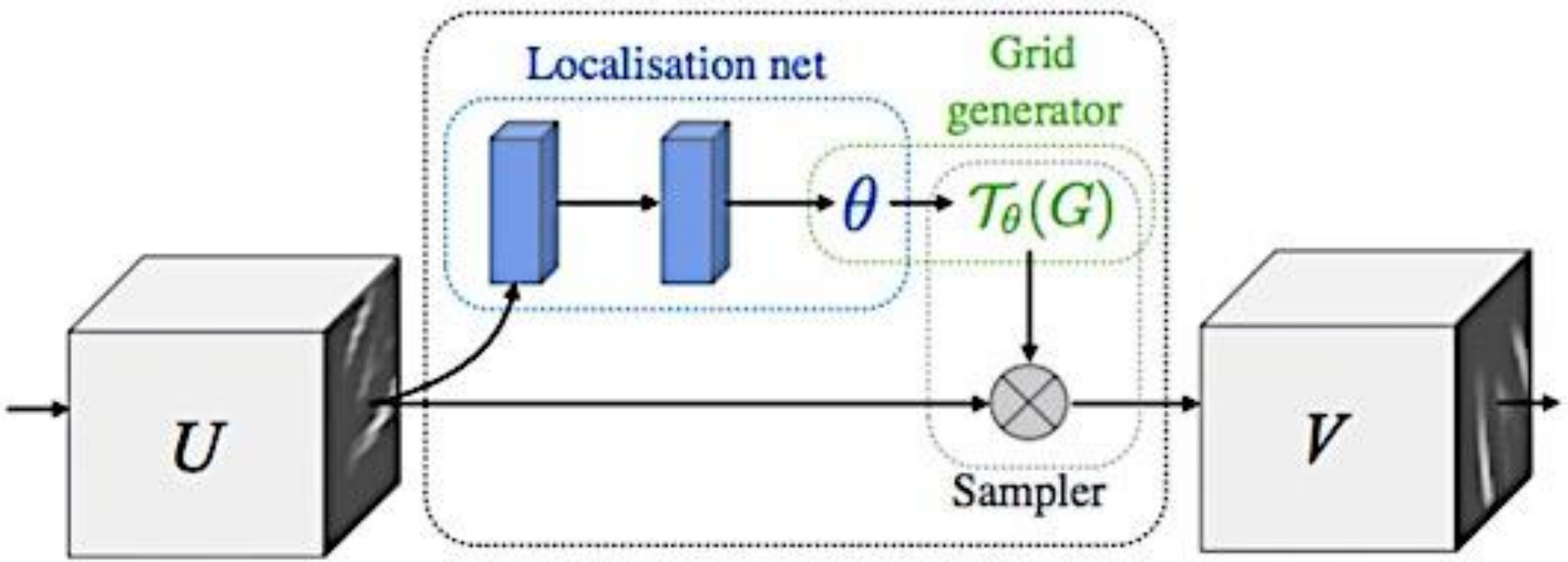
Такой подход позволяет сделать сеть асимптотически инвариантной к поворотам и изменению масштаба.

Spatial Transformer Networks



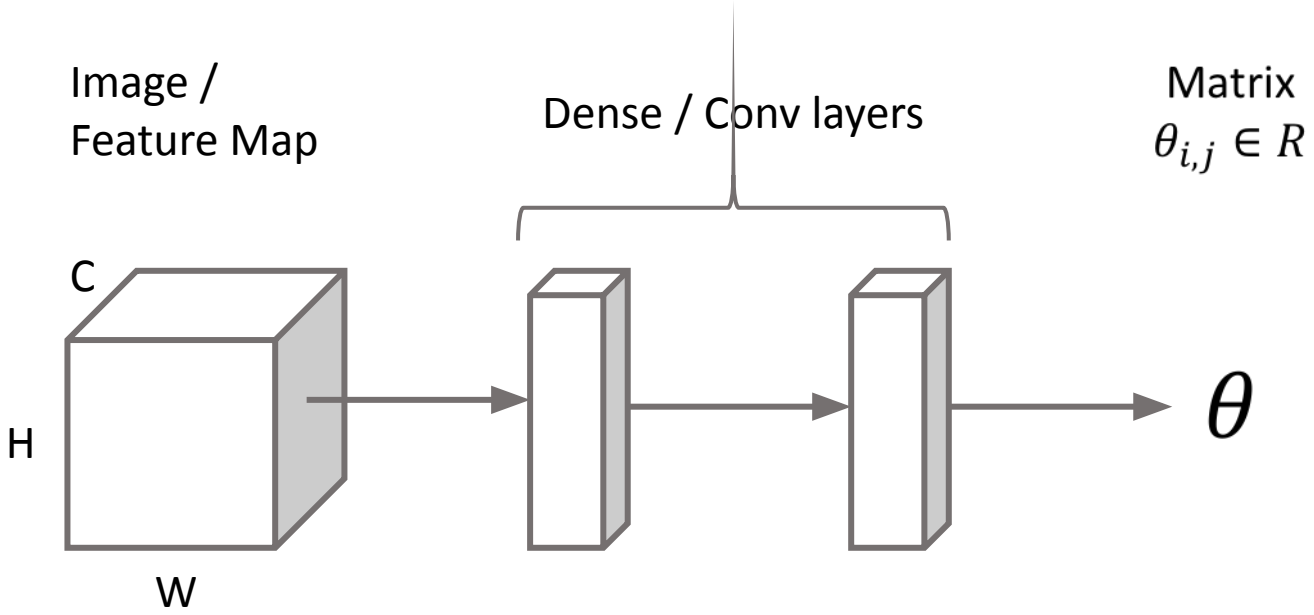
Source: http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf

ST \ Архитектура



Source: http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf

ST \ Localization Net



In: изображение или Feature Map размера (H, W, C).

Out: матрица θ размера (6,).

ST \ Grid Generator

Output from Localization Net

$$\begin{bmatrix} x^s \\ y^s \end{bmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{bmatrix} x^t \\ y^t \\ 1 \end{bmatrix}$$

s – source image/FM, *t* – target image/FM

Для каждого пикселя в выходном изображении мы получаем координаты пикселя во входном изображении, которые нам понадобятся на следующем шаге.

ST \ Image Sampler

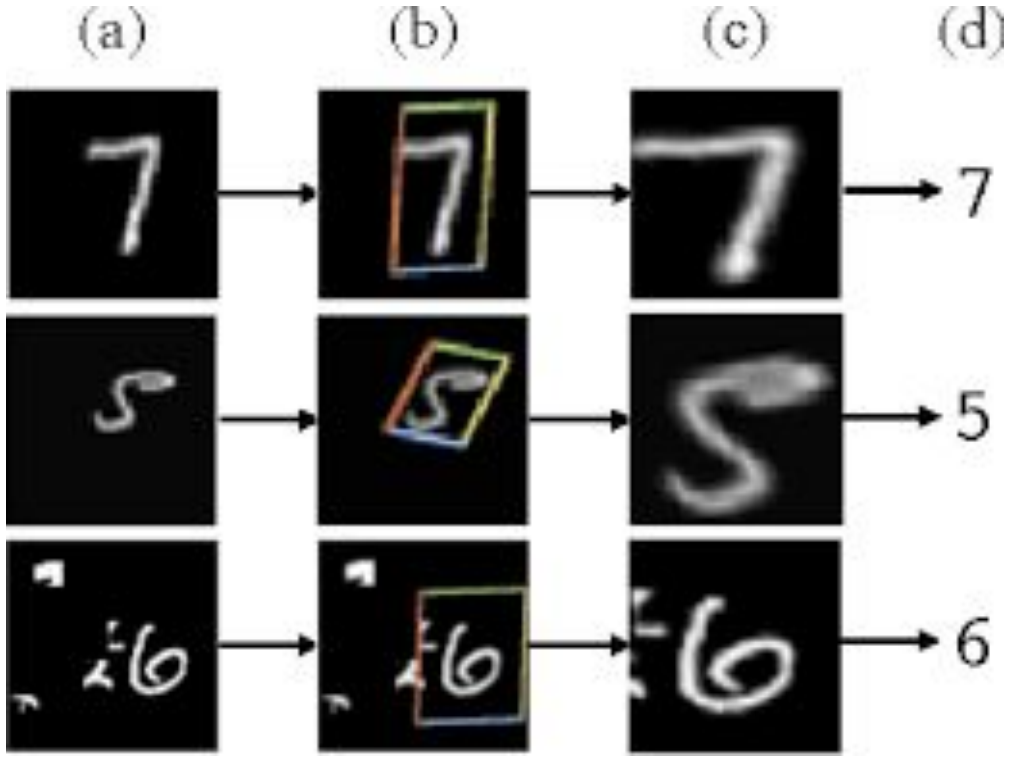
$$pixel\ value\ at\ (x^t, y^t) = pixel\ value\ at\ (k(x^s, y^s))$$

k – интерполяция (например, билинейная).

Для каждого из пикселя (x^t, y^t) мы берем пиксель (x^s, y^s) и каким-нибудь хитрым образом (*k*) вычисляем значение, которое присвоим пикселю (x^t, y^t).

Главное - операция присвоения значения пикселю (интерполяция) должна быть дифференцируема.

ST \ Результаты



Source: http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf

ST \ Результаты

batch = 0/200 theta = 1.02 0.02 -0.02
-0.02 1.02 -0.02

Source: https://habrahabr.ru/company/newprolab/blog/339484/

Spatial Transformer Networks

Q: Что можно делать с помощью ST?

поворачивать

обрезать

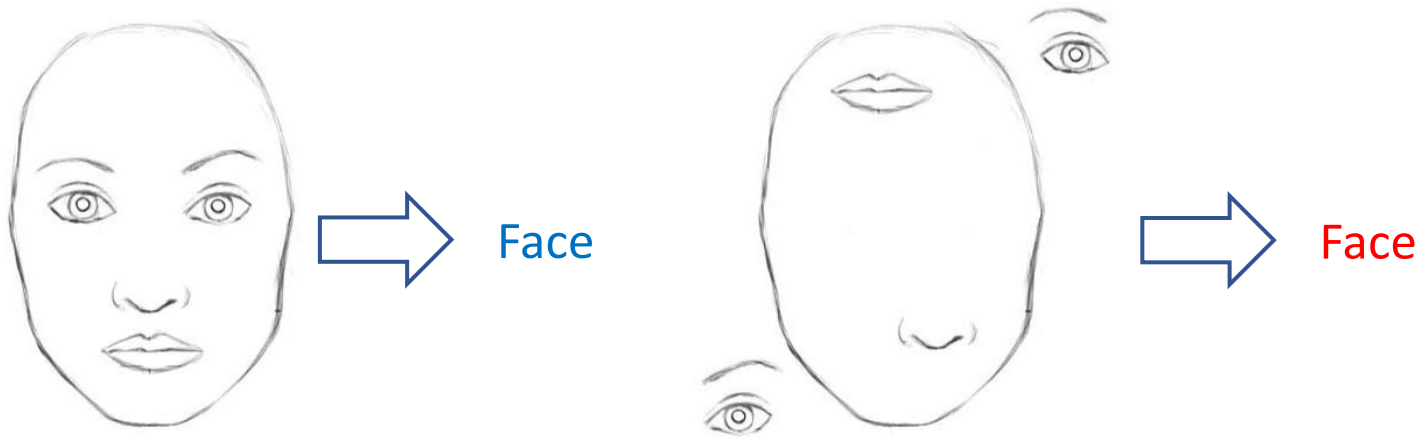
повышать/понижать размерность

Q: Что самое приятное в ST

То, что сеть сама поймет что и как ей делать. Все учится end-to-end без дополнительных функций потерь и т.п.

Capsule Networks

Что если мы хотим, чтобы сеть в классификации опиралась не только на присутствие компонентов какого-либо объекта, но и использовала информацию о взаимном расположении этих компонентов?



Source:
https://medium.com/ai^3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Capsule Networks \ Капсулы

Капсула – группа нейронов, чей вектор характеризует:

- Уверенность сети в том, что объект (или его часть) присутствует в какой-то части картинки.
- Параметры расположения этого объекта относительно других объектов.
- Параметры освещения, деформации, поворота и т.п.

Вообще, капсулы можно делать по-разному, но мы посмотрим на то, как это делает Geoffrey Hinton в оригинальной статье.

Capsule Networks \ Капсулы

В имплементации Хинтона:

- Длина вектора капсулы ($\|v_j\|$) – вероятность того, что объект присутствует на картинке.
- Значения вектора капсулы (v_j) – все остальные параметры объекта.

Capsule Networks \ Нелинейность

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||}$$

s_j – вход капсулы j, v_j – вектор капсулы j

Такая нелинейность приводит очень длинный вектор к длине около 1, а очень маленький к длине около 0.

Capsule Networks \ Вход капсулы

$$\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j|i} \ , \qquad \hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij} \mathbf{u}_i$$

s_j – вход капсулы j

u_i – вектор капсулы i с предыдущего слоя

W_{ij} – матрица весов, которая учится backprop'ом

c_{ij} – особый коэффициент, который вычисляется алгоритмом со следующего слайда

Capsule Networks \ Dynamic Routing

Procedure 1 Routing algorithm.

1: **procedure** ROUTING($\hat{\boldsymbol{u}}_{j|i}, r, l$)
2: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
3: **for** r iterations **do**
4: for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
5: for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$
6: for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
7: for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} . \mathbf{v}_j$
return $\mathbf{v}_j$

$$\text{squash}\,(s_j) = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||}$$

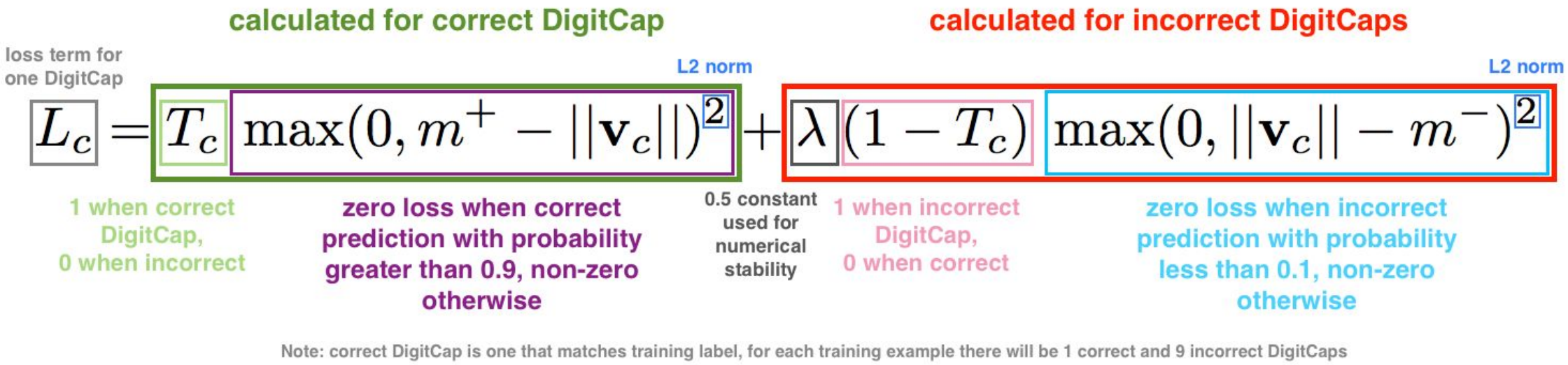
Source: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf

Capsule Networks \ Loss

$$L_k = T_k \ \max(0, m^+ - ||\mathbf{v}_k||)^2 + \lambda \ (1 - T_k) \ \max(0, ||\mathbf{v}_k|| - m^-)^2$$

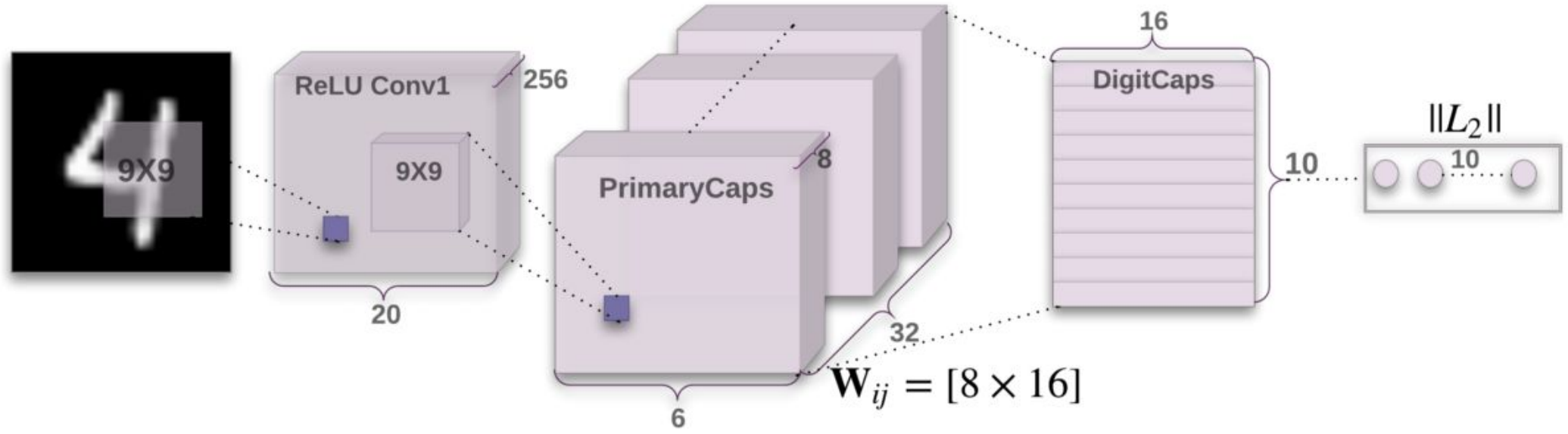
Давайте разберемся что это значит!

Capsule Networks \ Loss



Source:
https://medium.com/ai^3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Capsule Networks \ Архитектура

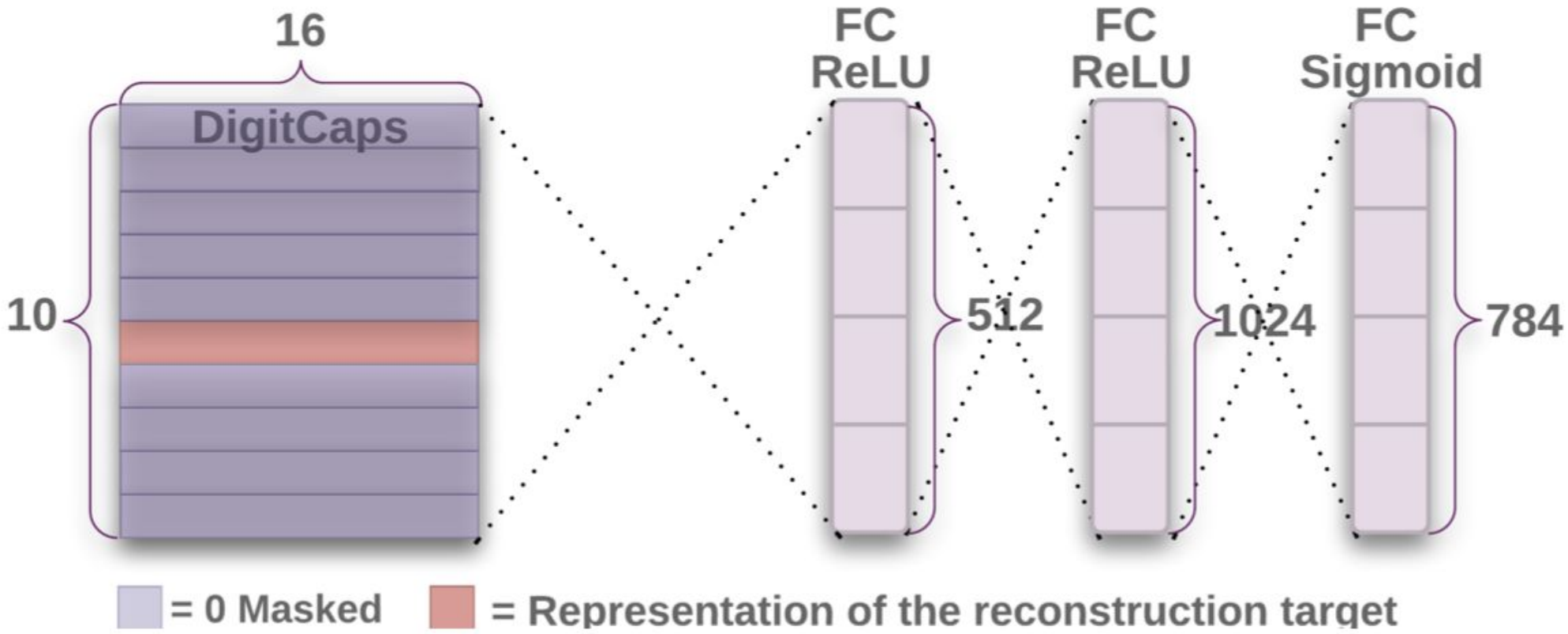


Все учится backprop’ом.

Dynamic routing делается только во время forward prop’а.

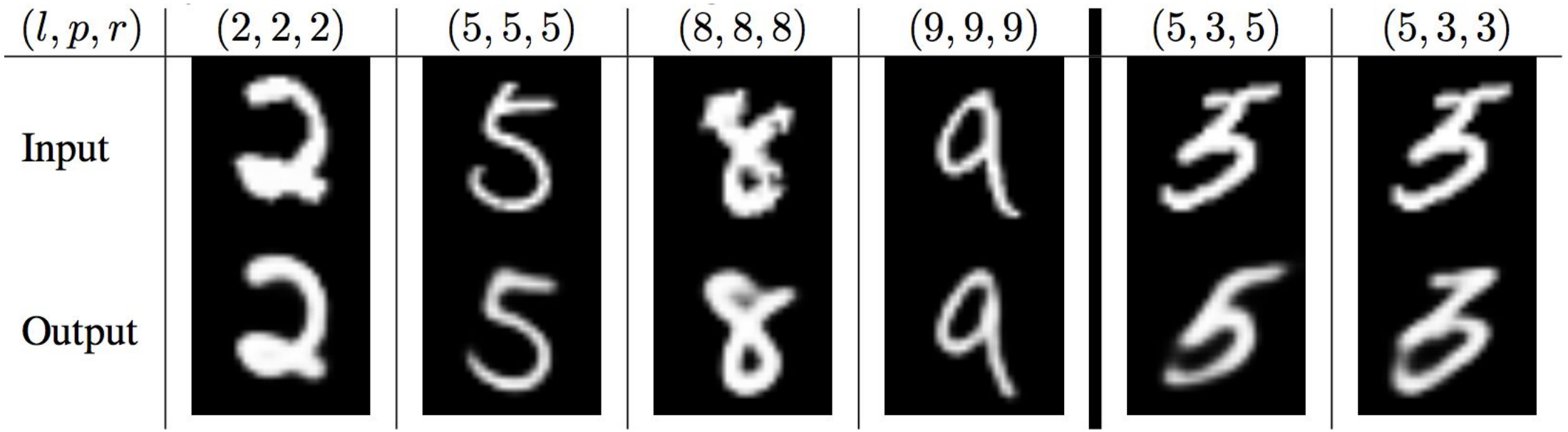
Source: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf

Capsule Networks \ Реконструкция



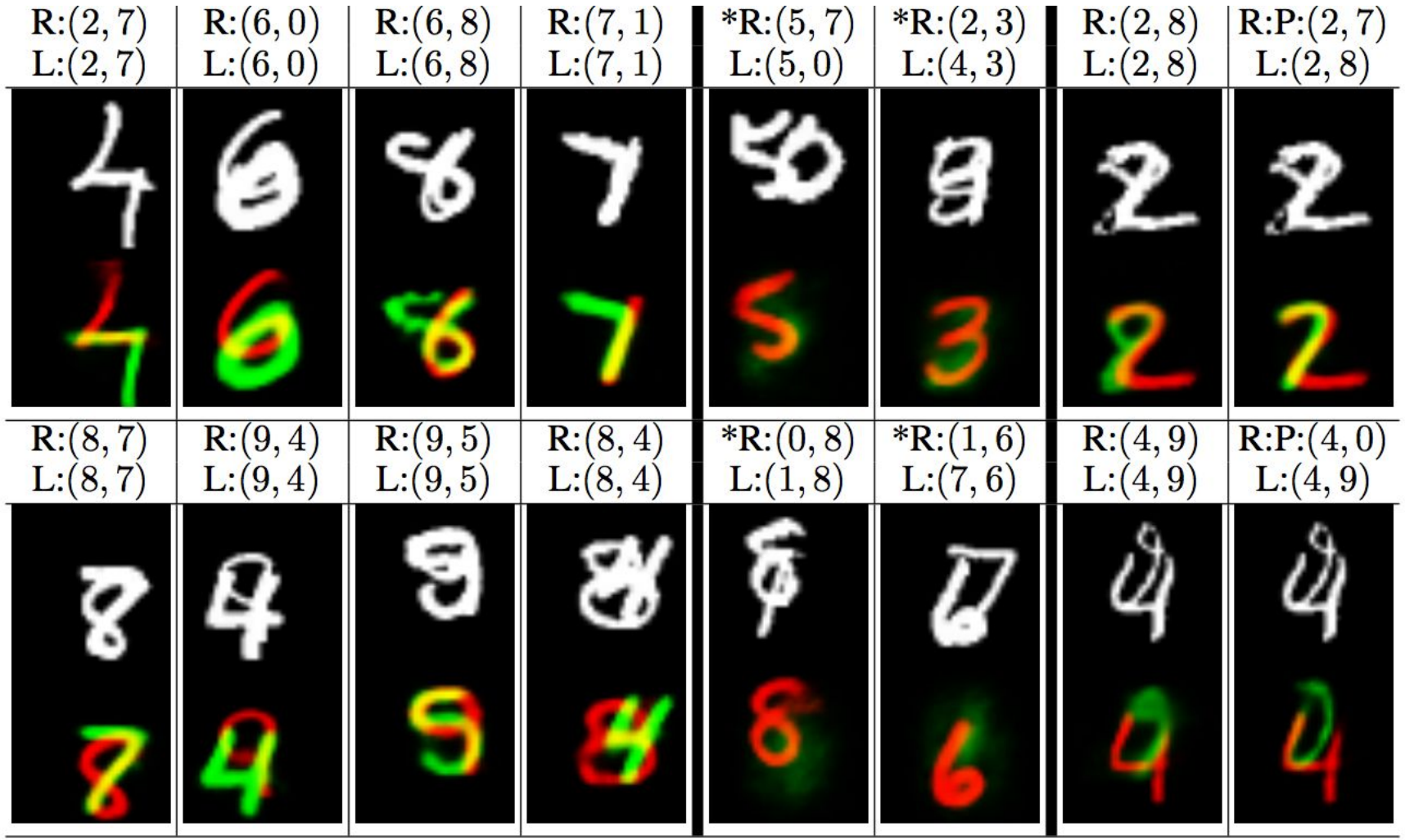
Source: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf

Capsule Networks \ Результаты



Source: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf

Capsule Networks \ Результаты



Source: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf

Capsule Networks \ Результаты

Scale and thickness	
Localized part	
Stroke thickness	
Localized skew	
Width and translation	
Localized part	

Source: http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf

Recap

1. У сверток есть недостатки
    - Они не инвариантны к поворотам
    - Они не инвариантны к изменению масштаба
    - Они плохо справляются с изменением взаимных позиций компонентов объекта

Recap

2. Spatial Transformer дает инвариантность к поворотам и изменению масштаба. А также, позволяет дифференцируемо обрезать, растягивать и вращать изображения.

Recap

3. Capsule Network помогает справится с проблемой изменения взаимных позиций компонентов объекта. Еще капсулы получают очень хорошие и иногда даже интерпретируемые репрезентации изображений.

Немного полезных материалов

Базовый туториал по ST:
http://pytorch.org/tutorials/intermediate/spatial_transformer_tutorial

Объяснение ST на русском:
https://habrahabr.ru/company/newprolab/blog/339484/

Отличный туториал по CapsNet:
https://github.com/higgsfield/Capsule-Network-Tutorial

Объяснение CapsNet в 4-х частях:
https://medium.com/ai^3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Страничка с тысячей ресурсов по CapsNet:
https://github.com/aisummary/awesome-capsule-networks

Оригинальные статьи:

ST
http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf

CapsNet
http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.pdf