

Dynamic Routing Between Capsules

Лебедев Антон

Ноябрь 20, 2017

- ▶ Человеческий мозг обрабатывает только небольшую часть оптического массива в полном разрешении.
- ▶ Сцену можно описать как набор заранее зафиксированных признаков.
- ▶ Деревья - это хорошо.

- ▶ Поделим слой нейронной сети на независимые компоненты - капсулы.
- ▶ Каждая капсула будет выбирать себе родителя из числа капсул предыдущего слоя.
- ▶ Выход каждой капсулы - вектор длиной не более 1.
- ▶ Длина вектора - уверенность в наличии признака.
- ▶ Ориентация — описание признака.

- ▶ s_j - общий вход капсулы.
- ▶ v_j - выход капсулы.

$$v_j = \frac{||s_j||^2}{1 + ||s_j||^2} \frac{s_j}{||s_j||}$$

- ▶ u_i - выход капсулы i предыдущего слоя.
- ▶ $\hat{u}_{i|j} = W_{ij} u_i$ - вектор предсказания.
- ▶ $s_j = \sum_i c_{ij} \hat{u}_{i|j}$ - общий вход капсулы.

$$c_{i,j} = \frac{\exp(b_{i,j})}{\sum_k \exp(b_{i,k})}$$

- ▶ Веса хотим ставить большими у тех капсул, с которыми высокая согласованность. $a_{ij} = v_j \hat{u}_{ij}$
- ▶ Итоговые $b_{i,j}$ будем получать с помощью процедуры routing

Procedure 1 Routing algorithm.

```
1: procedure ROUTING( $\hat{\mathbf{u}}_{j|i}, r, l$ )
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  $l$ :  $\mathbf{c}_i \leftarrow \text{softmax}(\mathbf{b}_i)$  ▷ softmax computes
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  $(l + 1)$ :  $\mathbf{v}_j \leftarrow \text{squash}(\mathbf{s}_j)$  ▷ squash computes
7:     for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$ 
   return  $\mathbf{v}_j$ 
```

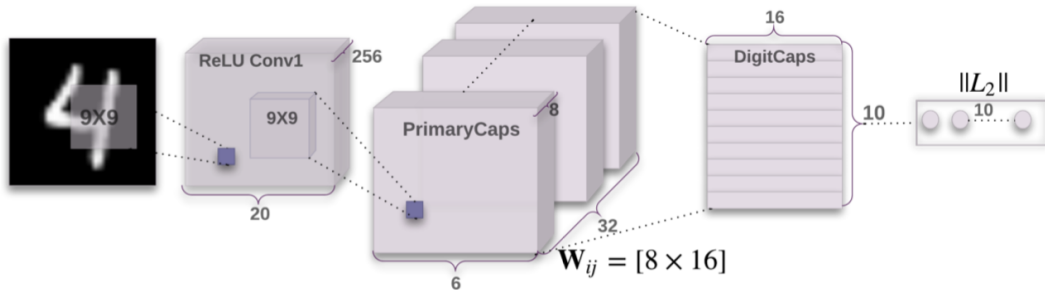
Margin loss for digit existence

- ▶ T_k Индикатор наличия класса под номером k .
- ▶ $m^+ = 0.9$
- ▶ $m^- = 0.1$
- ▶ $\lambda = 0.5$

$$L_k = T_k \max(0, (m^+ - \|v_k\|))^2 + \lambda(1 - T_k) \max(0, (m^- - \|v_k\|))^2$$

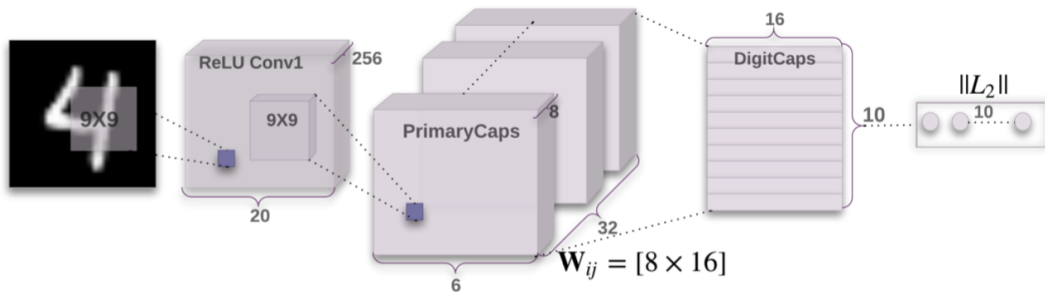
CapsNet architecture (первый слой)

- ▶ На вход: мнист — картинки 28x28.
- ▶ Свёртка 9x9 256 выходов, страйд 1.
- ▶ Relu активация.



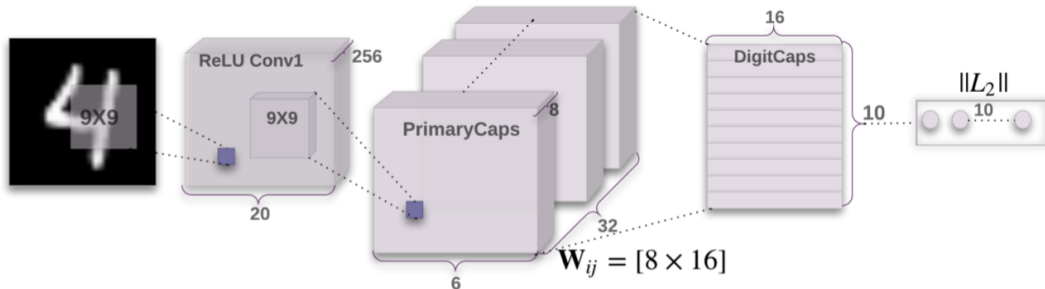
CapsNet architecture (второй слой PrimaryCapsules)

- ▶ 32 каналные капсульные светки (32 раза капсульная свертка).
- ▶ Каждая капсульная свертка содержит свертку 9x9 с выходом 8 и страйдом 2.
- ▶ Капсульная свертка - набор капсул. Каждая капсула - вектор фичемапы с измерения глубины. (И Squashing поверх этого)
- ▶ Итого $32 * 6 * 6$ капсул (Паддингов то нет нигде)



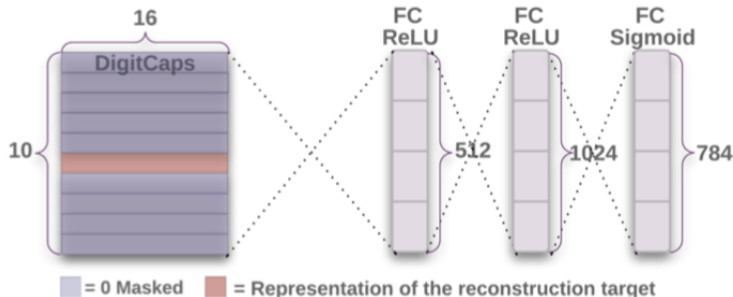
CapsNet architecture (Третий слой DigitCaps)

- ▶ 10 16D капсул - по капсуле на цифру.
- ▶ Каждая капсула получает все выходы капсул с PrimaryCapsules












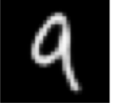


CapsNet architecture - Декодер

- ▶ Выбираем капсулу с правильным ответом и подаем в декодер.
- ▶ Декодер учим восстанавливать картинку, для этого добавлям в лосс квадрат евклидова расстояния между выходом и входом, домноженный на 0.0005.



Результаты - Декодер


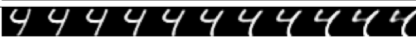
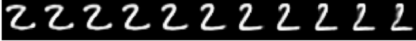
- ▶ (l, p, r) - label, prediction, reconstruction target
- ▶ Первые четыре - правильная работа сети
- ▶ Последние два - ошибочное предсказание. По работе декодера можно понять почему сеть отработала плохо - что заставило сеть увидеть в пятерке тройку.

(l, p, r)	$(2, 2, 2)$	$(5, 5, 5)$	$(\tilde{8}, 8, 8)$	$(9, 9, 9)$	$(5, 3, 5)$	$(5, 3, 3)$
Input						
Output						

Результаты - Интерпретация фичей

- ▶ Благодаря декодеру можем менять фичи DigitCaps и понимать за что они отвечают.
- ▶ Выучивают толщину обводки, наклон и ширину. Так же специфичные признаки, например, длину хвоста у 2.

Figure 4: Dimension perturbations. Each row shows the reconstruction when one of the 16 dimensions in the DigitCaps representation is tweaked by intervals of 0.05 in the range $[-0.25, 0.25]$.

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

MultiMNIST

- ▶ MultiMnist - датасет, полученный наложением примеров мниста с разными метками друг на друга.
- ▶ L — label, R — reconstruction, P — prediction.
- ▶ Звездочкой отмечены примеры, где для реконструкции использовался лэйбл ни из предсказания, ни из истинных меток.

R:(2, 7) L:(2, 7)	R:(6, 0) L:(6, 0)	R:(6, 8) L:(6, 8)	R:(7, 1) L:(7, 1)	*R:(5, 7) L:(5, 0)	*R:(2, 3) L:(4, 3)	R:(2, 8) L:(2, 8)	R:P:(2, 7) L:(2, 8)
R:(8, 7) L:(8, 7)	R:(9, 4) L:(9, 4)	R:(9, 5) L:(9, 5)	R:(8, 4) L:(8, 4)	*R:(0, 8) L:(1, 8)	*R:(1, 6) L:(7, 6)	R:(4, 9) L:(4, 9)	R:P:(4, 0) L:(4, 9)

Accuracy results

- ▶ Baseline - классическая трехслойная сеть, собранная так, чтобы вычислительные затраты были похожи на CapsNet (количество параметров там больше)
- ▶ Итерации роутинга и регуляризация реконструкцией важны.
- ▶ Отмечают, что подобные результаты достигались только более глубокими сетками.

Method	Routing	Reconstruction	MNIST (%)	MultiMNIST (%)
Baseline	-	-	0.39	8.1
CapsNet	1	no	$0.34_{\pm 0.032}$	-
CapsNet	1	yes	$0.29_{\pm 0.011}$	7.5
CapsNet	3	no	$0.35_{\pm 0.036}$	-
CapsNet	3	yes	$0.25_{\pm 0.005}$	5.2

Устойчивость с аффинным преобразованием

- ▶ Учили baseline и CapsNet на датасете, где каждый пример - цифра мниста, расположенная в случайном месте на картине 40x40
- ▶ После затестили на affNIST - примеры мниста после небольшого аффинного преобразования
- ▶ CapsNet: 0.9923 на expanded MNIST test set, 0.79 affNIST
- ▶ baseline: 0.9922 на expanded MNIST test set, 0.66 affNIST

Sara Sabour, Nicholas Fross, and Geoffrey E Hinton. Dynamic routing between capsules. In Neural Information Processing Systems (NIPS), 2017.
<https://arxiv.org/pdf/1710.09829.pdf>