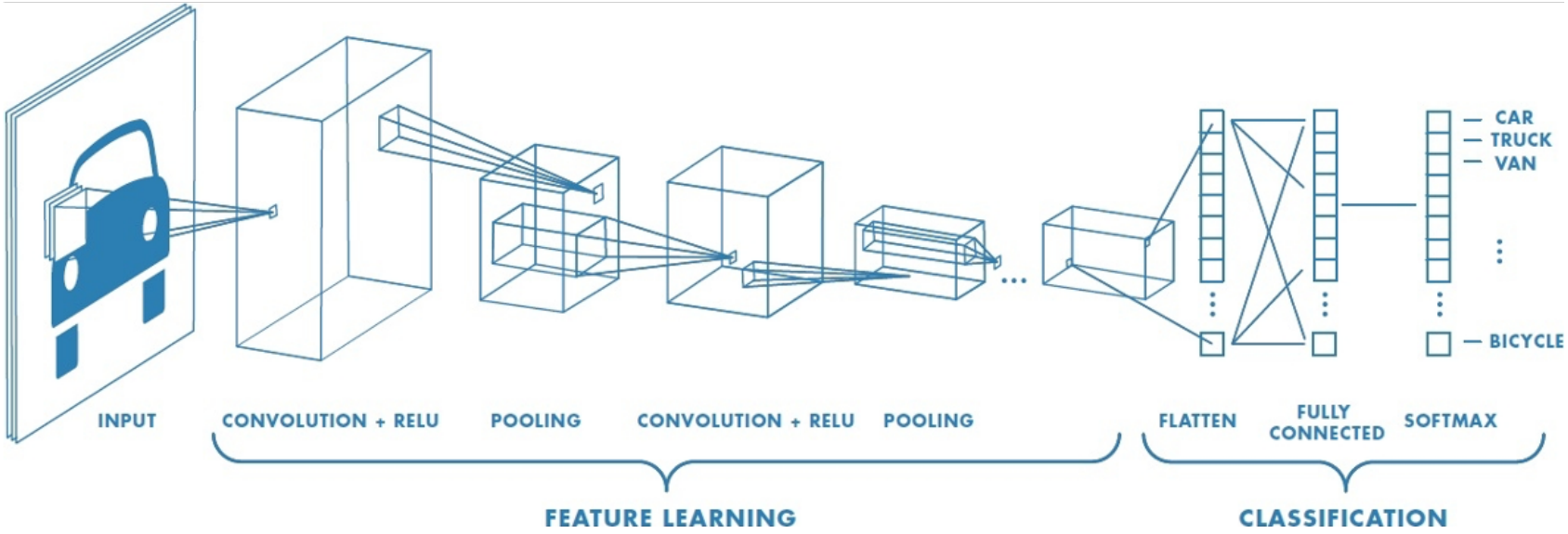
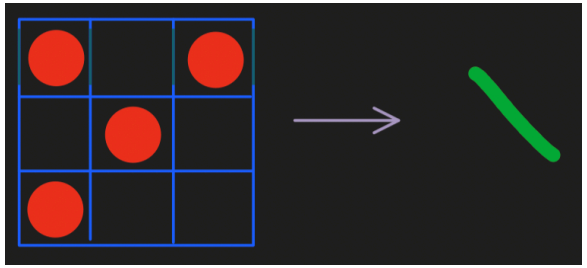
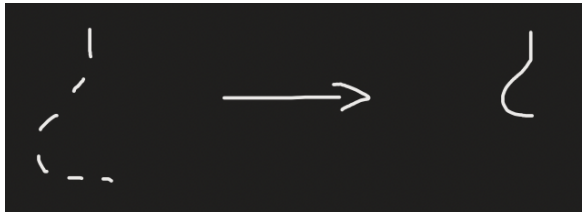
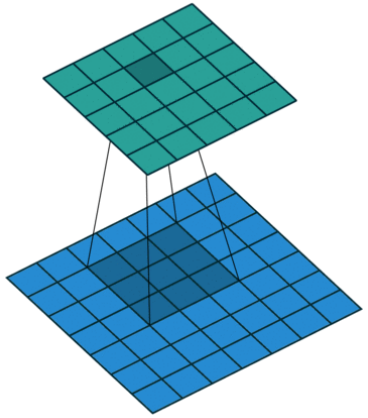
Капсульные нейронные сети

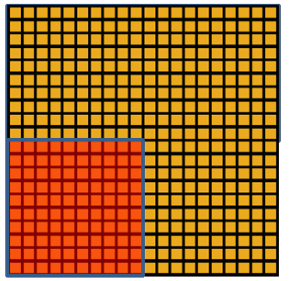
Свёрточные сети



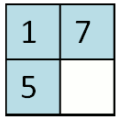








Convolved feature

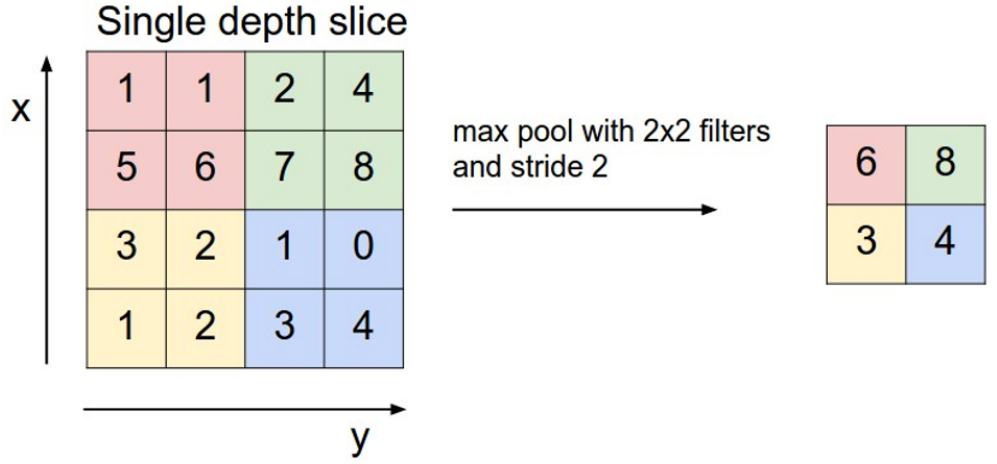


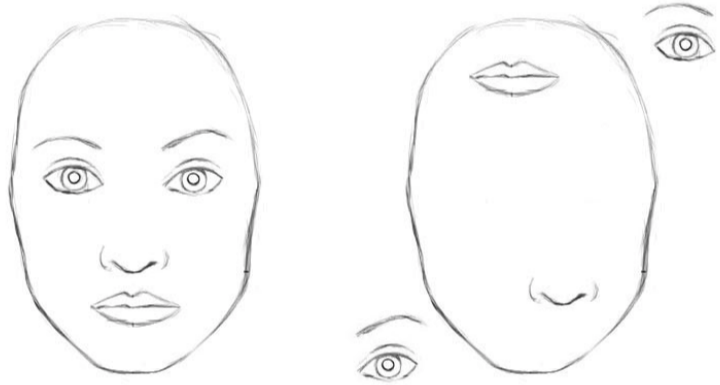
Pooled feature

Invariance vs Equivariance

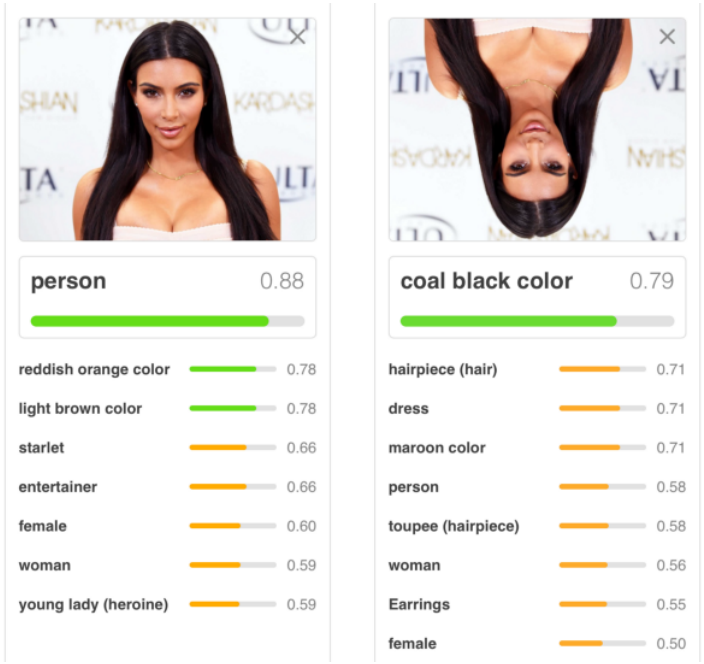
Invariance: внутреннее представление объекта
не изменяется при изменении свойств этого объекта.

Hinton: “The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster.”

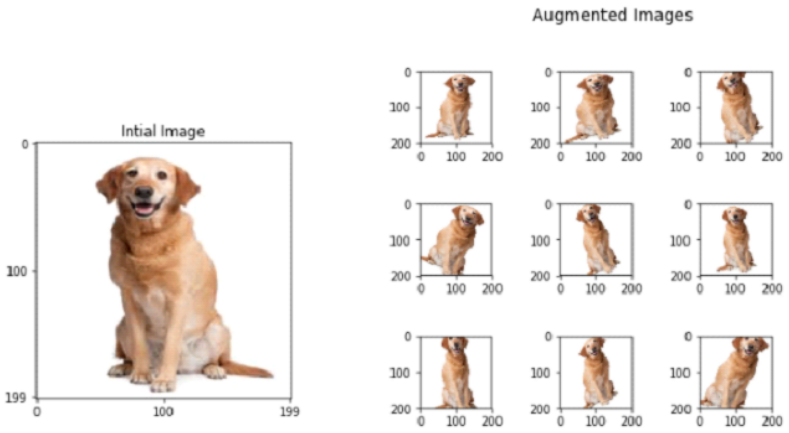




CNN не инвариантна таким преобразованиям изображения, как поворот, изменение цвета или условий освещения.



Augmentation

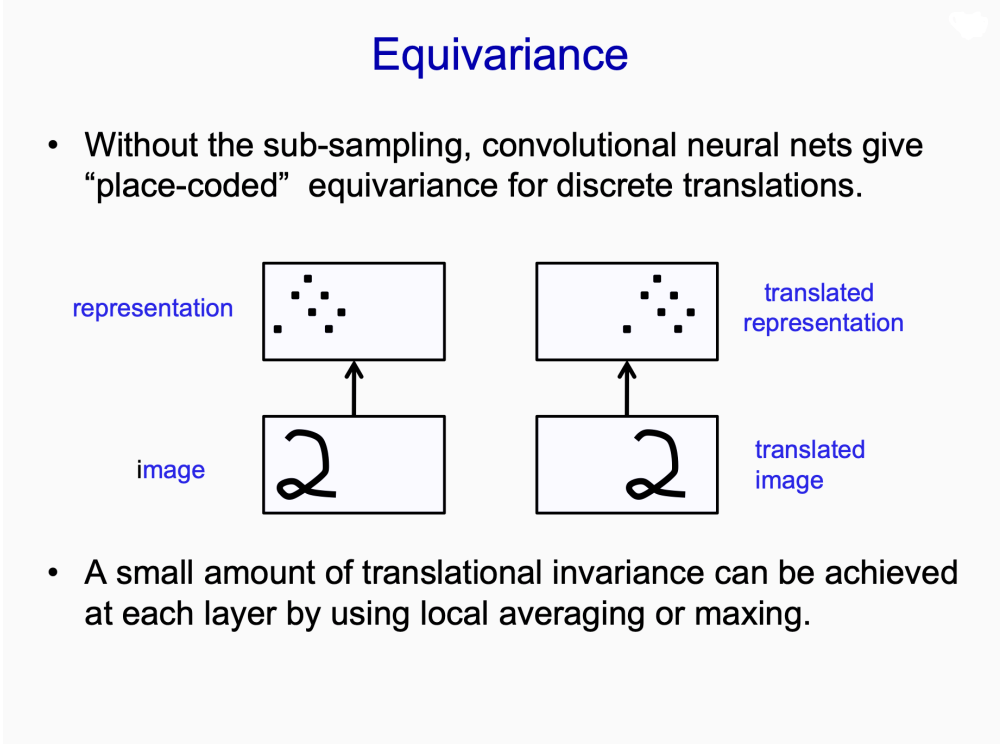


Equivariance

Equivariance: внутреннее представление фиксирует свойства объекта.

f эквивалентна ф-ии g,
если f(g(x)) = g(f(x))

g: I'(x,y) = I(x-1,y)

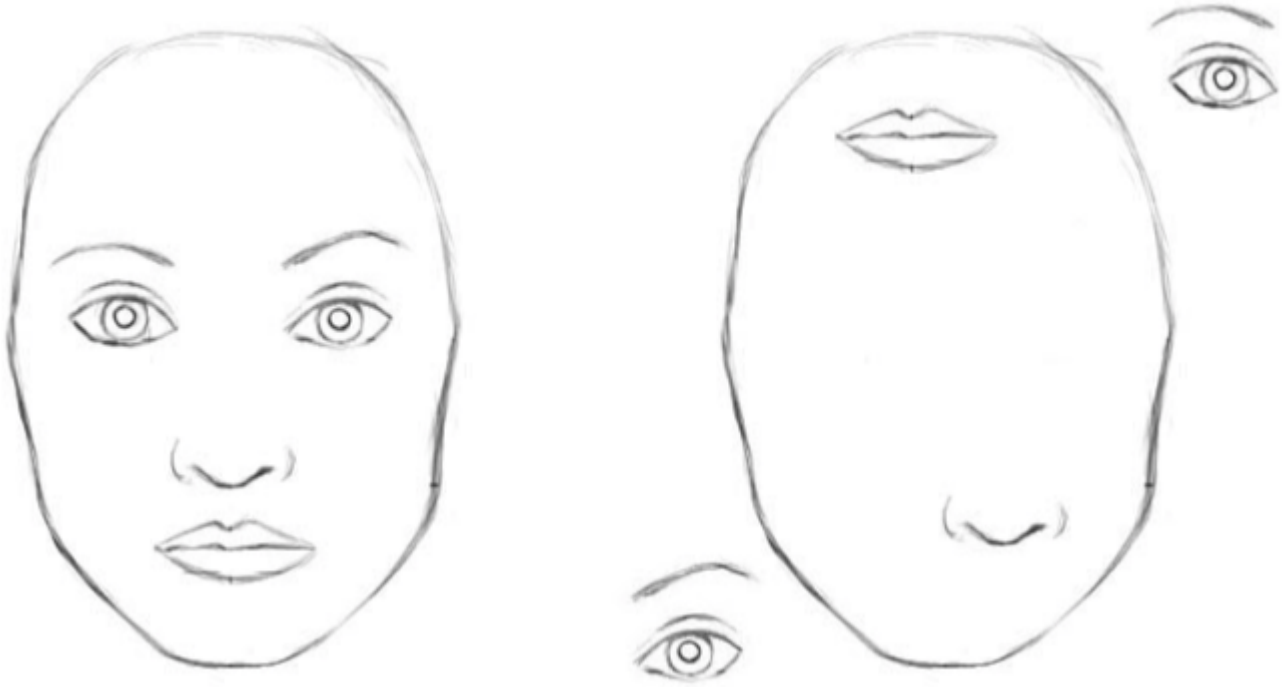


Мы можем с легкостью распознать, что на каждой картинке изображена Статуя Свободы, несмотря на то, что она изображена под разными углами

Все потому что, внутреннее представление Статуи Свободы в нашем мозге не зависит от угла просмотра.

Может быть человек никогда не видел именно этих картинок, но он может узнать, что на них изображено.

Проблемы свёрточных сетей



- Отсутствие информации об ориентации, проблемы при повороте объектов, изменении освещения и т.д.
- Большие потери информации на пуллинге.
- Отсутствие информации о расположении признаков относительно друг друга.

Капсульные нейронные сети

Капсула - расширение искусственного нейрона до векторной формы

Длина
Аналог выходного сигнала искусственного нейрона

Ориентация
Хранит в себе отклонение состояния

Алгоритм динамической маршрутизации по соглашению

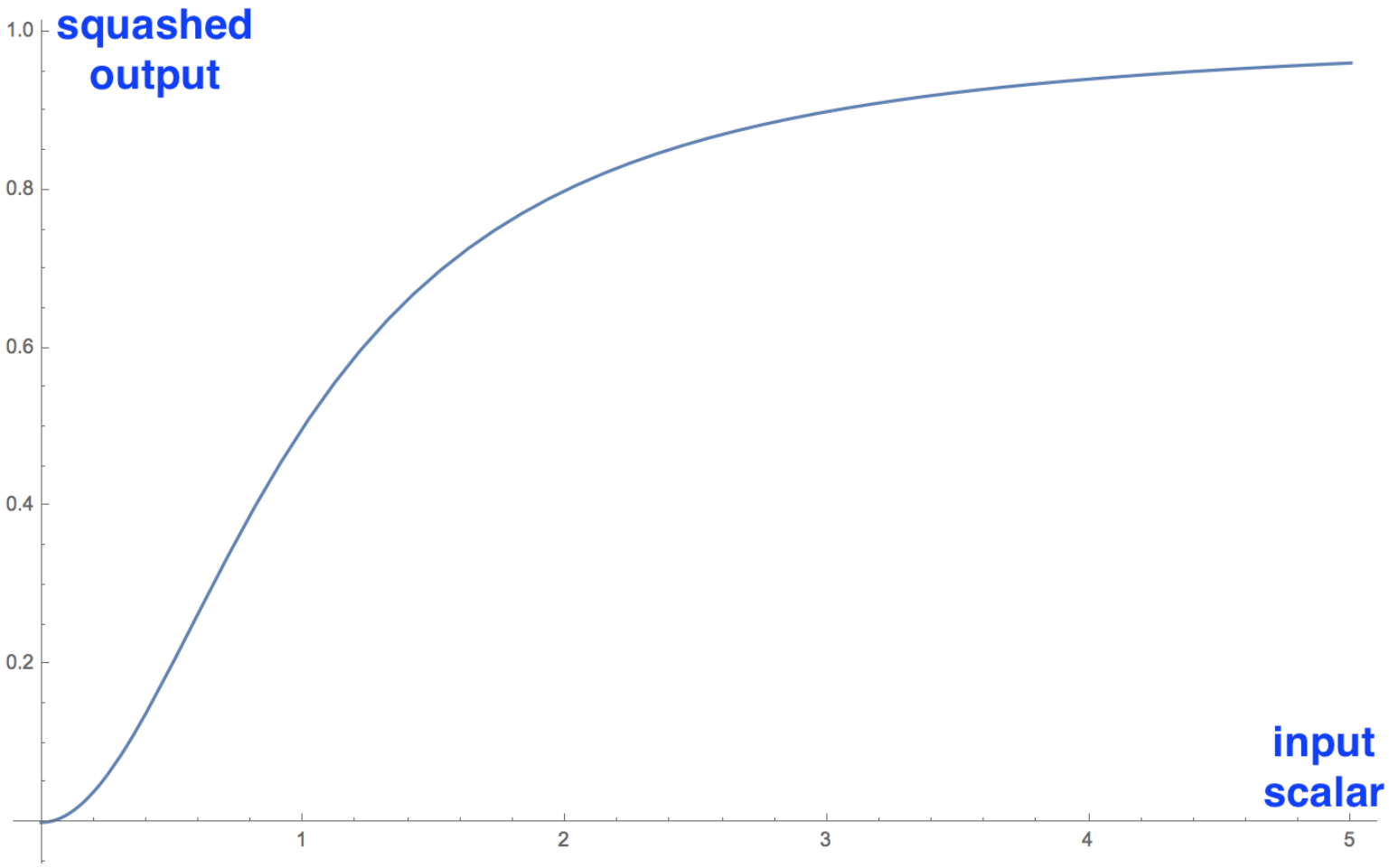
Capsule vs. Traditional Neuron			
Input from low-level capsule/neuron		$\text{vector}(\mathbf{u}_i)$	$\text{scalar}(x_i)$
Operation	Affine Transform	$\hat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear Activation	$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$	$h_j = f(a_j)$
Output		$\text{vector}(\mathbf{v}_j)$	$\text{scalar}(h_j)$

Получение вектора предсказания

Суммирование с учетом коэффициентов связки

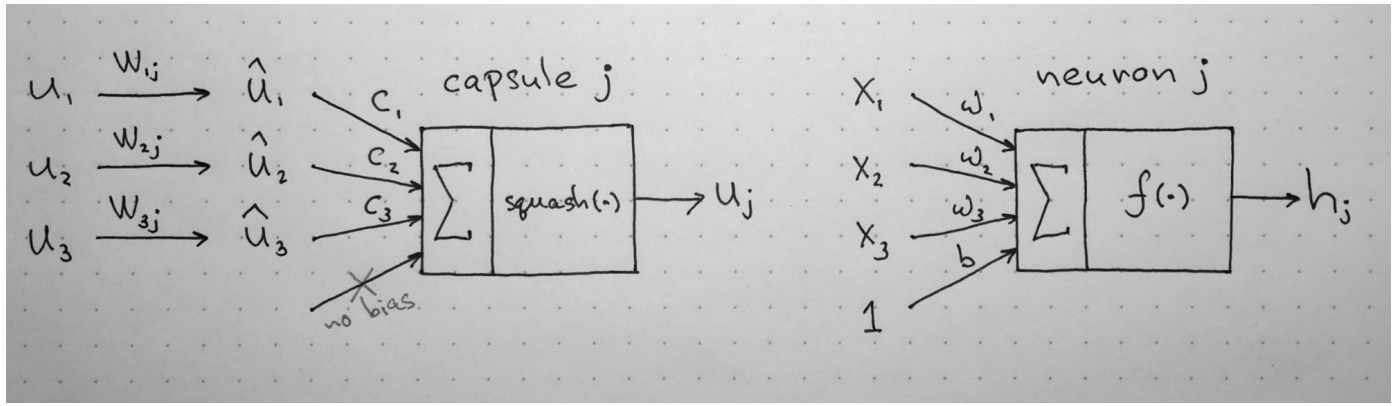
Функция активации (нормализация + нелинейность)

Squash



$$\mathbf{v}_j = \underbrace{\frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}}_{\text{additional "squashing"}} \underbrace{\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}}_{\text{unit scaling}}$$

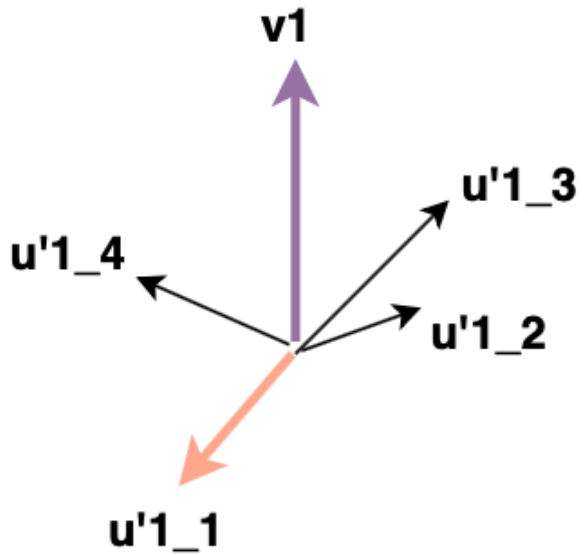
Алгоритм динамической маршрутизации по соглашению

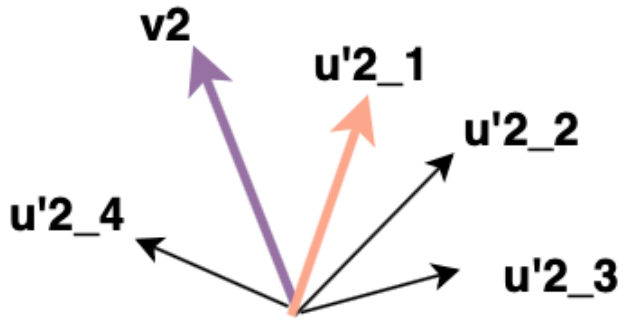


Procedure 1 Routing algorithm.

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: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$ ▷ `softmax` computes Eq. 3
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 $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$ ▷ `squash` computes Eq. 1
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 $\mathbf{v}_j$

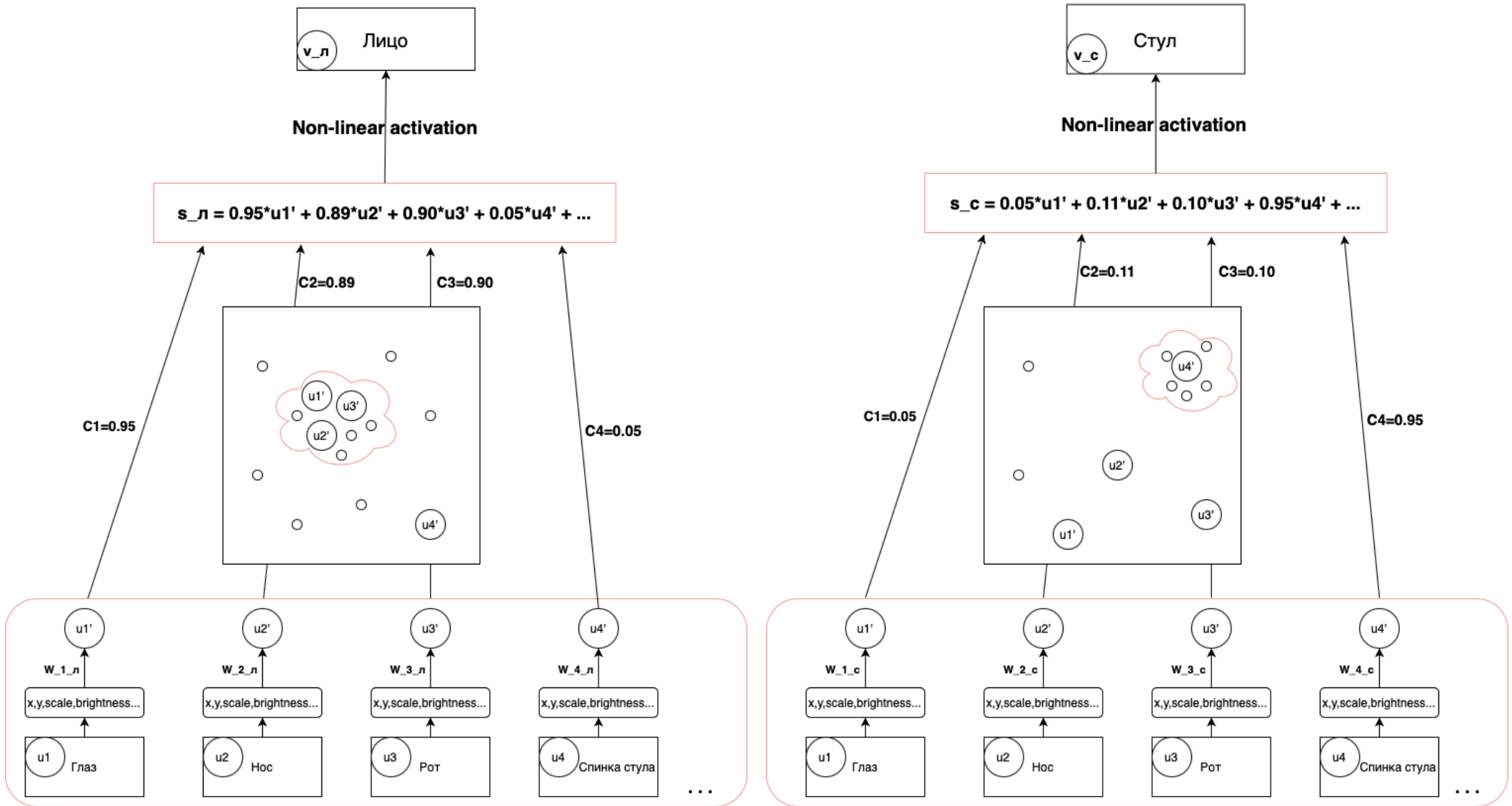
Алгоритм динамической маршрутизации по соглашению

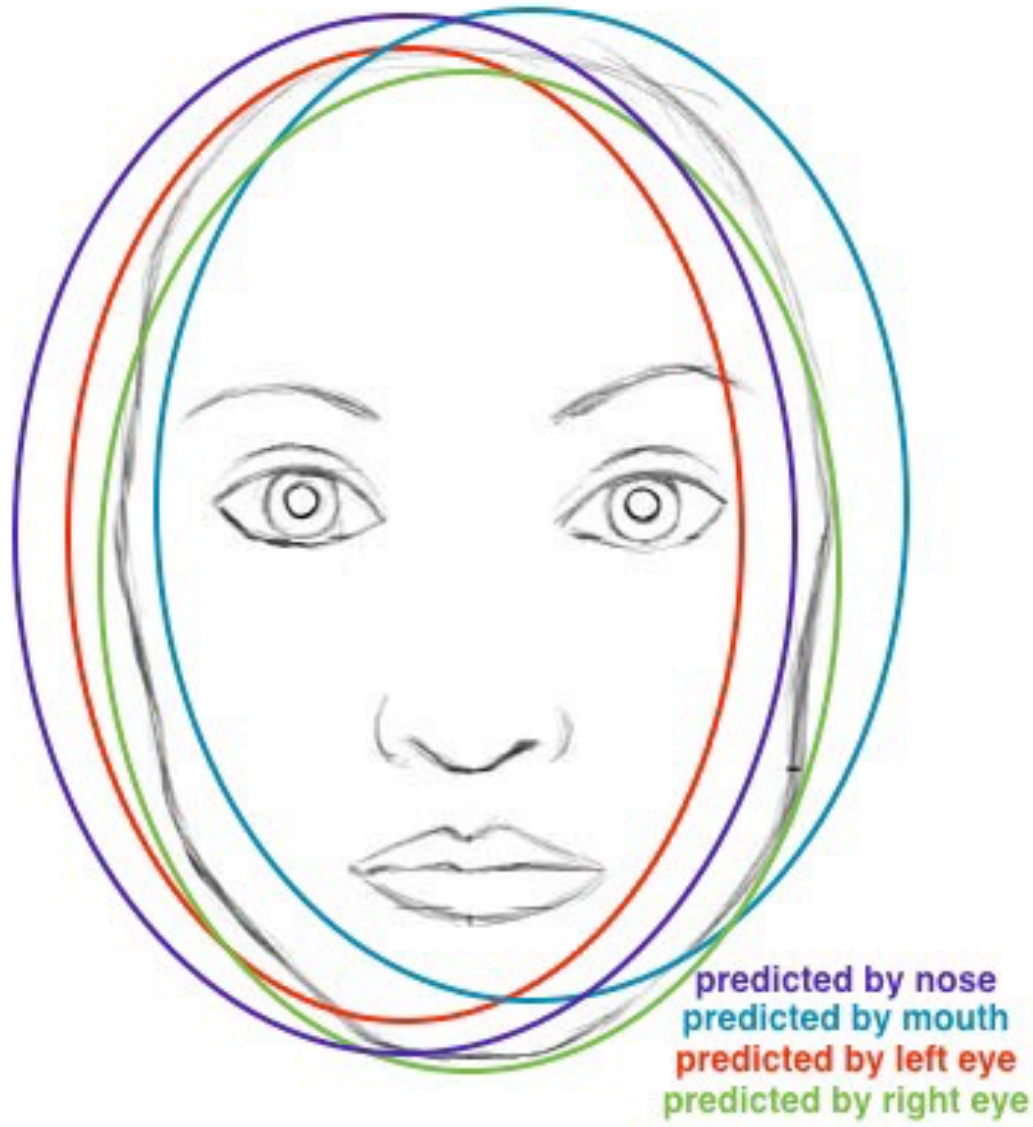




C1_1 ↓

C1_2 ↑


v_л
Лицо
Non-linear activation
s_л = 0.95*u1' + 0.89*u2' + 0.90*u3' + 0.05*u4' + ...
C1=0.95
C2=0.89
C3=0.90
C4=0.05
u1'
u3'
u2'
u4'
u1'
u2'
u3'
u4'
W_1_л
W_2_л
W_3_л
W_4_л
x,y,scale,brightness...
x,y,scale,brightness...
x,y,scale,brightness...
x,y,scale,brightness...
u1
Глаз
u2
Нос
u3
Рот
u4
Спинка стула
...
v_c
Стул
Non-linear activation
s_c = 0.05*u1' + 0.11*u2' + 0.10*u3' + 0.95*u4' + ...
C1=0.05
C2=0.11
C3=0.10
C4=0.95
u4'
u2'
u3'
u1'
u1'
u2'
u3'
u4'
W_1_c
W_2_c
W_3_c
W_4_c
x,y,scale,brightness...
x,y,scale,brightness...
x,y,scale,brightness...
x,y,scale,brightness...
u1
Глаз
u2
Нос
u3
Рот
u4
Спинка стула
...


predicted by nose
predicted by mouth
predicted by left eye
predicted by right eye

Функция потерь

CapsNet Loss Function

calculated for correct DigitCap

calculated for incorrect DigitCaps

loss term for one DigitCap

L2 norm

L2 norm

$$L_c = T_c \max(0, m^+ - ||\mathbf{v}_c||)^2 + \lambda (1 - T_c) \max(0, ||\mathbf{v}_c|| - m^-)^2$$

1 when correct DigitCap, 0 when incorrect

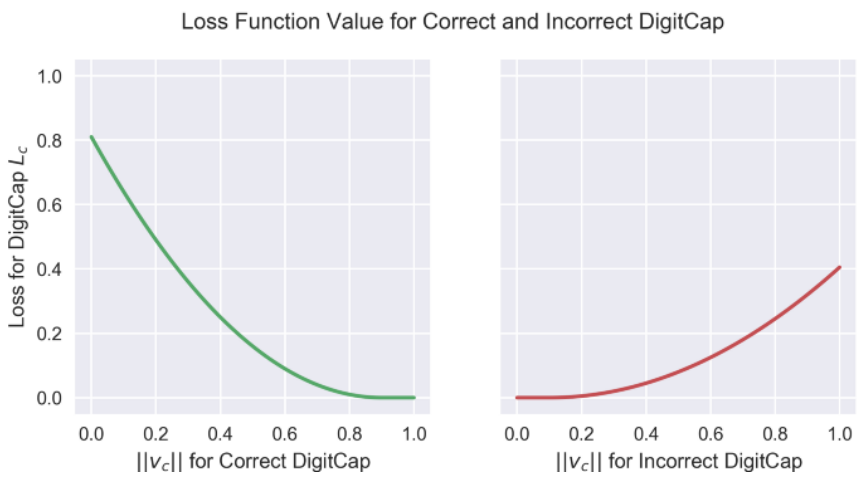
zero loss when correct prediction with probability greater than 0.9, non-zero otherwise

0.5 constant used for numerical stability

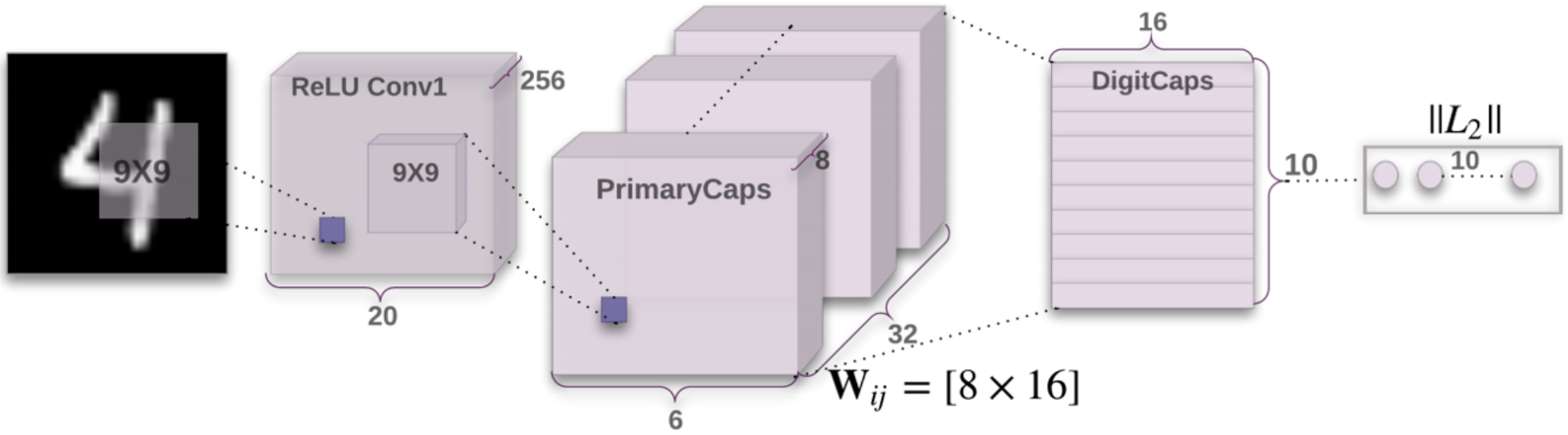
1 when incorrect DigitCap, 0 when correct

zero loss when incorrect prediction with probability less than 0.1, non-zero otherwise

Note: correct DigitCap is one that matches training label, for each training example there will be 1 correct and 9 incorrect DigitCaps

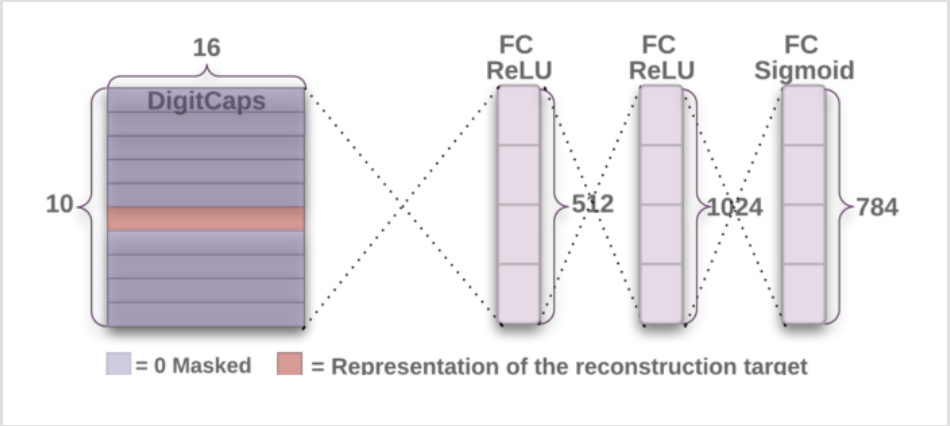


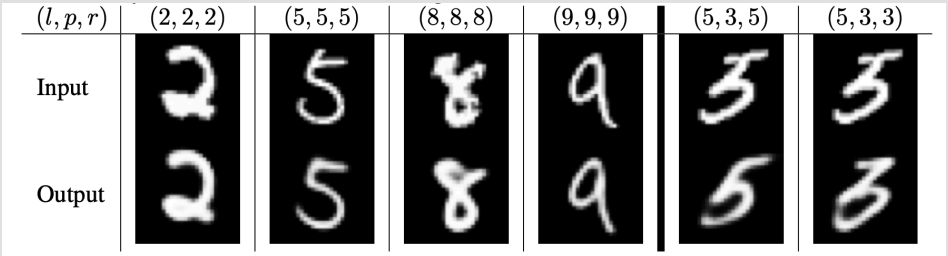
Архитектура CapsNet



Реконструкция

- Реконструктор принимает 16-мерный вектор капсулы соответствующий правильному изображению и пытается восстанавливить это изображение. Используется в качестве регуляризатора.





1 epoch

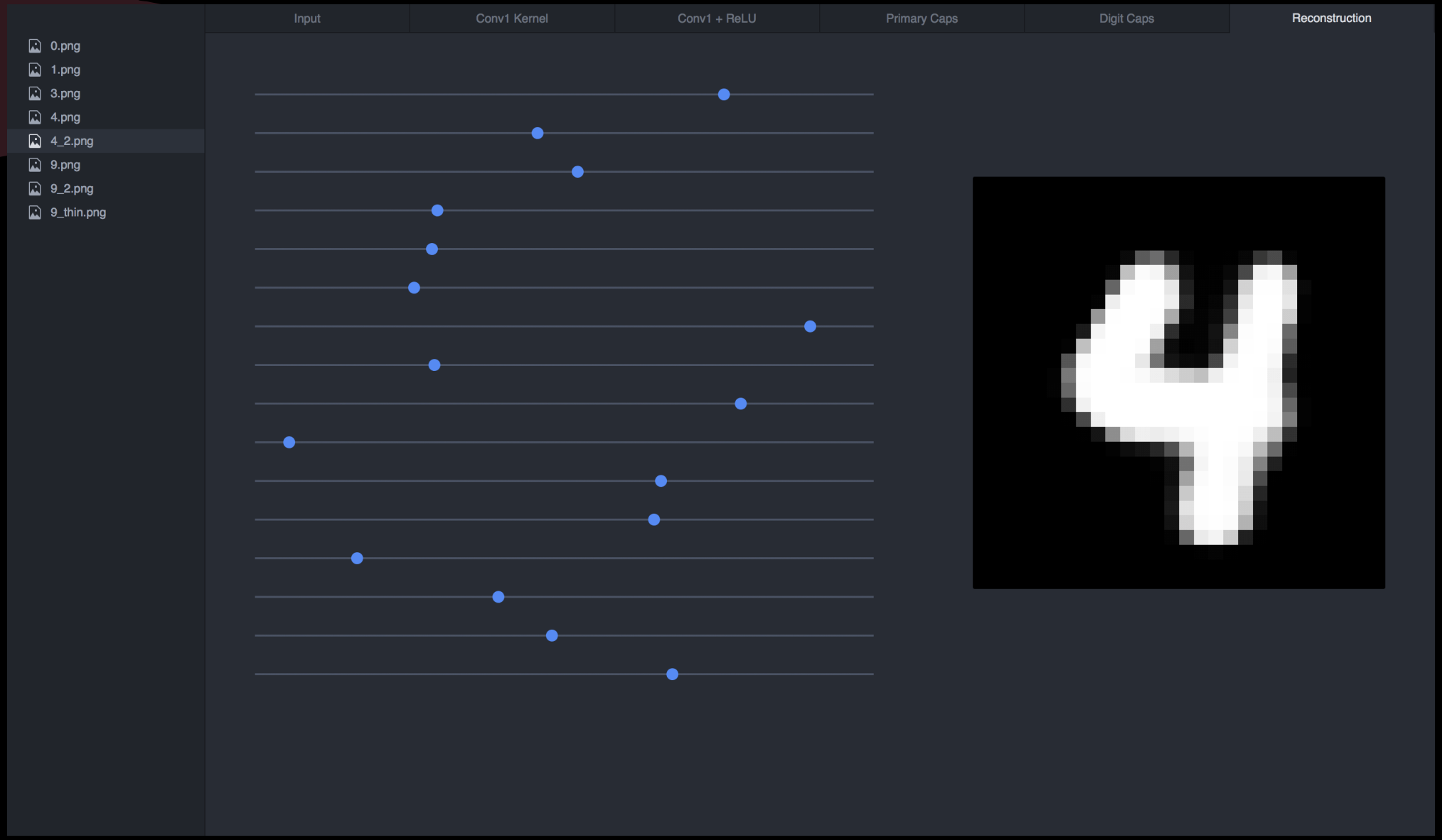
5 epochs

10 epochs

15 epochs

20 epochs

50 epochs

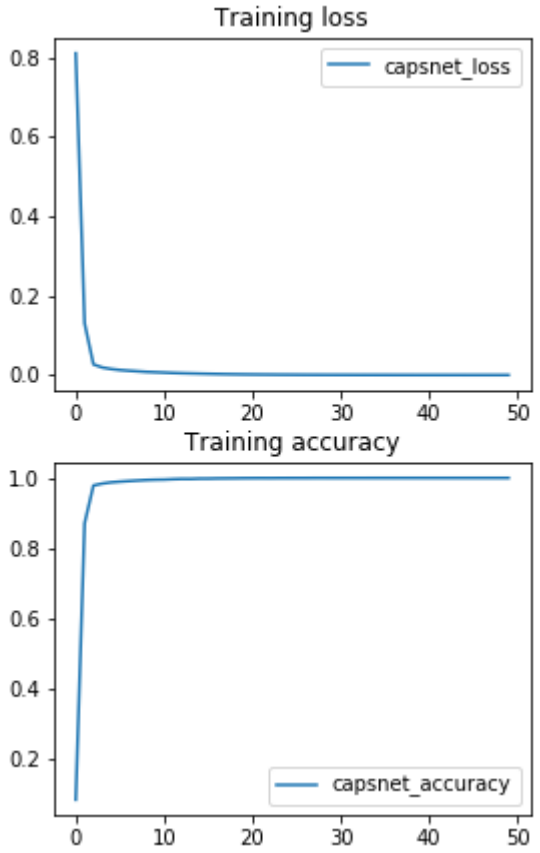

Input
Conv1 Kernel
Conv1 + ReLU
Primary Caps
Digit Caps
Reconstruction
0.png
1.png
3.png
4.png
4_2.png
9.png
9_2.png
9_thin.png

Результаты на MNIST

Table 1: CapsNet classification test accuracy. The MNIST average and standard deviation results are reported from 3 trials.

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	$\mathbf{0.25}_{\pm 0.005}$	**5.2**

epochs	Test accuracy
1	0.9674
5	0.9866
10	0.9877
20	0.9906
50	0.9931



Результаты на других датасетах

Fig. 1. Example images from the Yale face database B.

Fig. 2. Example images from the MIT CBCL dataset.

Fig. 3. Example images from the Belgium TS dataset.

Fig. 4. Example images from the Cifar-100 dataset

Table 3. Comparison of classification results.

			Baselines			CapsNet	
Dataset	**Classes**	**Instances**	**Algorithm**	**Avg. training time**	**Test accuracy**	**Avg. training time**	**Test accuracy**
Yale Face Database B	38	5850	Fisherface	~5 minutes*	98.2%**	~24 hours***	95.3%
MIT CBCL (faces)	10	5240	Fisherface	~1 minute*	98.3%**	~14 hours***	99.87%
BelgiumTS (traffic signs)	62	7000	Modified LeNet	<1minute*	98.2%	16 hours***	92% (40 epochs)
CIFAR-100 (objects)	100	60000	Resnet 50	20 hours (200 epochs)	65.5%	18 hours***	18% (35 epochs)

** Core i7 extreme 7th generation, 16 GB RAM, Nvidia GeForce 1050 Ti Mobile*

*** With best parameters we discovered using multiple tests with training and validation sets*

**** Google cloud, Intel Xeon, 5 Gb RAM, NVIDIA Tesla K80*

Итоги

Плюсы:

- Эквивариантность нейронной активности в отношении изменений входных данных и инвариантности в вероятностях обнаружения признаков
- Инкапсуляция ценной информации вместо простого выбора максимальных значений

Минусы:

- Вычислительная сложность
- Малая исследованность => Низкая точность