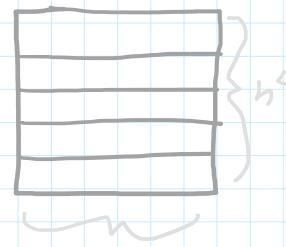


Let's begin from Primary Capsules

Primary Caps Shape:  $[B, h^L, d^L]$ , where  $h^L = w^L \times h^L \times f^L$

For the schematic representations, let's assume that  $n^L=5$ ,  $h^{L+1}=2$ ,  $m=3$   
Under those assumptions,  $C_b^L$ :

Input: PrimaryCaps  $C^L$



SimsOut $_b^L$ :

Output: Similarities SimsOut $_b^L$

SimsOut.shape:  $[B, h^{L+1}]$

Trainable Parameters:  $W$  (transformation Matrices)

Non-Trainable Parameters:  $C^{L+1}$

$C^{L+1}.$ shape:  $[h^{L+1}, d^{L+1}]$

Hyperparameters: reduced\_votes, m, normalize\_votes, norm\_type,

If True, then each capsule  $C_i^{L+1}$  produces  $n^{(L+1)}$  votes. If False, each capsule produces  $m$  votes, where  $m$  is not necessarily equal to  $n^{(L+1)}$ .

Transformation Matrices for each capsule. This is used only if reduced\_votes == False. So, it specifies the number of projections that Each capsule  $C_j^{(L+1)}$  has.

If we set normalize\_votes hyperparameter to True then we transform the vote vectors so as every one of them has length no more than one.

If we set it to 0 then we apply squashing function as a normalization technique. If we set it to 1 we apply tanh normalization to votes and unit normalization to output capsules.

v  
Number of routing iterations. In each iteration, Capsules  $C_j^{(L+1)}$  are updated once.

radical  
If radical is set to True then instead of computing the differences as the votes minus the digit caps, we set the differences to be same as the votes. Radical approach was inspired by the perceptron algorithm (update rule) and differs significantly from the SOM's update rule.

θ  
Specifies the neighborhood function. It is a list of float numbers that determine the gravity of the update for the second, third place. Always, the winner should get the full update so theta at least should contain 1.0. It is worth mentioning that unlike SOM, our lateral distance is the relative to the winner parent capsule distance.

If True then the updates for each digit capsule are scaled according to the number of votes they "won" in a batch.

take-into-account-similarity, softmax, Cr-SOM

If it is set to True then the updates are scaled according to the similarity. Also, if set, the individual thetas in the theta list are not taken into consideration. Just the length of the list to define how many inner loop iterations.

If set to True then instead of hard winners we get soft winners. So even capsules that do not have any winner may get updated with a small coefficient. Of course then there is no point in using a neighborhood.

Similar to softmax. Just the operation is scaled so that dissimilar parent capsules move further away when updated.

Used to normalize digit capsules in every iteration of the outer loop. This, if many iterations are used improves stability.

Start of SOM-Based Routing

At first we compute the votes using the transformation matrices. Depending on the value of the reduced\_votes hyperparameter, we use different transformation matrices and get different votes.

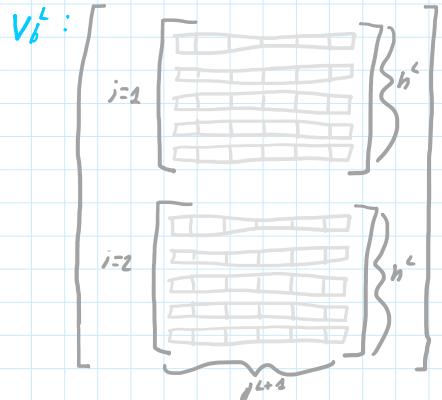
If reduced-votes == True :

$$W^L \text{ shape : } [n^L, d^L, d^{L+1}, n^{L+1}]$$

$$\forall b \in [1, B], \forall i \in \mathcal{R}_L, \forall j \in \mathcal{R}_{L+1} : V_{b|i|j}^L \leftarrow C_{bi}^L \times W_{i::j}^L$$

$[2 \times d^L] \quad [d^L \times d^{L+1}]$

$$V^L \text{ shape : } [B, n^L, n^{L+1}, d^{L+1}]$$



else :

$$W^L \text{ shape } [n^L, d^L, d^{L+1}, m^L]$$

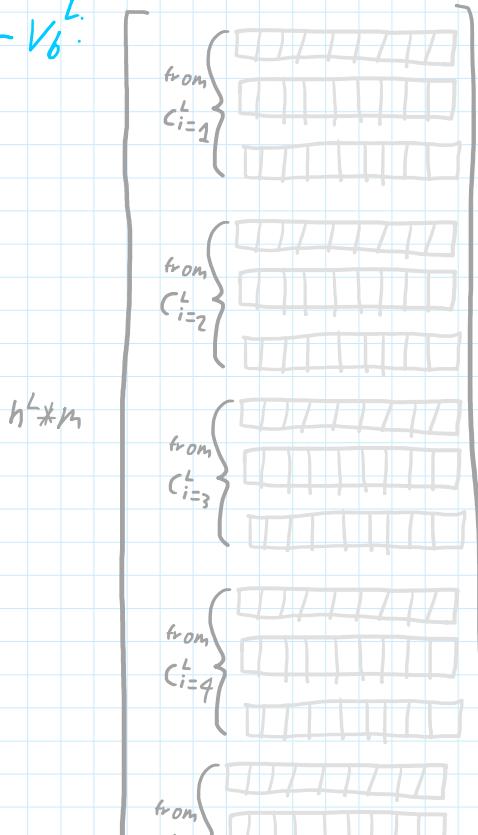
$$\forall b \in [1, B], \forall i \in \mathcal{R}_L, \forall j \in [1, m] : \text{pre-}V_{b(i+n^L*(i-1))d^{L+1}}^L \leftarrow C_{bi}^L \times W_{i::jm}^L$$

$[2 \times d^L] \quad [d^L \times d^{L+1}]$

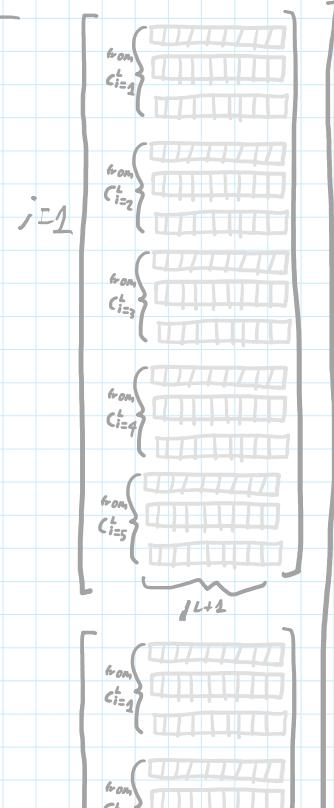
Actually, we compute m projections for each one of the  $n^L$  primary capsules and then we concatenate (we get them all together so we lose track of which vote comes from which primary capsule). We will still use the same index "i" but this, no longer means that this particular vote  $V_{(ij)}^L$  comes from capsule  $C_i^L$ .

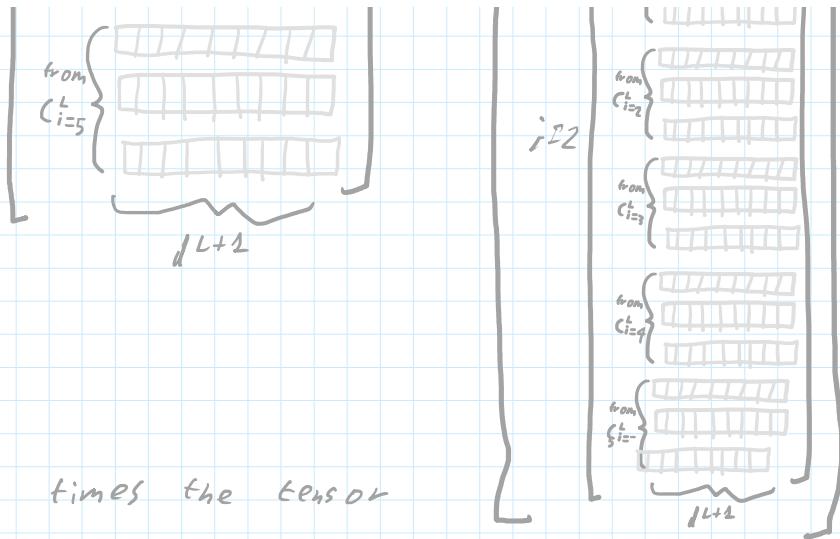
Now  $i \in [1, n^L * m]$  so  $\text{pre-}V_i^L$  comes from  $C_i^L$ .

$\text{pre-}V_b^L$ :



$V_b^L :$





Copy  $n^{L+1}$  times the tensor

pre- $V^L$  so as to get  $V.$ shape:  $[B, \underbrace{n^L \times m}_{n^L}, \underbrace{d^L}_{d^{L+1}}, d^{L+1}]$

$\forall j \in \underline{\Omega}_{L+1}: V_{\cdot \cdot \cdot j}^L \leftarrow \text{pre-}V^L$

In other words, Stack  $n^{L+1}$  copies of pre- $V^L$  vertically.

So,  $V.$ shape:  $[B, \underbrace{n^L \times m}_{n^L}, \underbrace{d^L}_{d^{L+1}}, \underbrace{d^L}_{d^{L+1}}]$

$$\underline{n}^L = \begin{cases} n^L & \text{iff } \text{reduced-votes} == \text{True} \\ n^L \times m & \text{iff } \text{reduced-votes} == \text{False} \end{cases}$$

$$\underline{\Omega}_L = \begin{cases} \Omega_L & \text{iff } \text{reduced-votes} == \text{True} \\ \text{set of all votes in tensor pre-}V_b^L, & \text{iff } \text{reduced-votes} == \text{False} \end{cases}$$

So... in either case:

$V.$ shape:  $[B, \underline{n}^L, h^{L+1}, d^{L+1}]$

if normalize-votes:

Make vote vectors to not exceed length of 1 (L2 norm).

if norm-type == 0:

$\forall b \in [1, B], \forall i \in \underline{\Omega}_L, \forall j \in \underline{\Omega}_{L+1}: V_{b \cdot i \cdot j}^L \leftarrow \text{Squash}(V_{b \cdot i \cdot j}^L)$

$$\frac{\|V_{b \cdot i \cdot j}^L\|^{2k}}{2 + \|V_{b \cdot i \cdot j}^L\|^2} \cdot \frac{V_{b \cdot i \cdot j}^L}{\|V_{b \cdot i \cdot j}^L\|}$$

$\xrightarrow{\text{unit norm}}$

else:

$\forall b \in [1, B], \forall i \in \underline{\Omega}_L, \forall j \in \underline{\Omega}_{L+1}: V_{b \cdot i \cdot j}^L \leftarrow \tanh\left(\frac{\|V_{b \cdot i \cdot j}^L\|}{\text{Scaling term}}\right) \left( \frac{V_{b \cdot i \cdot j}^L}{\|V_{b \cdot i \cdot j}^L\|} \right)$

$V$  has same shape as before.

If take-into-account-win-ratio:

Initialize WinCount with zeros,  $\text{WinCount.} \text{shape: } [n^{L+2}]$

for  $L$  iterations do

Initialize  $SU$  with zeros,  $SU.\text{shape}:[B, n^L, n^{L+2}, d^{L+1}]$   
 → sparse updates

Time to compute the differences between the  $V^L$  and  $C^{L+1}$ .  
 These differences "D" will be used in the update rule to update the parent capsules  $C^{L+1}$ .

If radical:

$$D^L \leftarrow V^L$$

Radical updates is inspired by Perception rule.  
 This is significantly different to the updates computed in the original SOM.

else:

$$\forall b \in [1, B], \forall i \in \Omega_L: D_{bi}^L \leftarrow V_{bi}^L - C^L$$

Even if `reduced_votes == False`, if `radical` is not `True`,  $D^L$  tensor has different entries for different  $j$ 's (unlike  $V^L$  which contains  $n^{L+2}$  copies of pre- $V^L$  iff `reduced_votes == False`). That is because pre- $V^L$  votes (in  $V^L$ ) are subtracted by different parent capsules.

$$\text{original SOM update rule: } C_j^{L+1} \leftarrow C_j^{L+1} + \alpha(s) * \text{Dec. function} * (V_{bi}^L - C_j^{L+1})$$

↓ learning rate      ↓ neighborhood function

It is clear that the above update rule can not be parallelizable. Finding the `mincr` (Best Matching Unit - BMU) parent capsule for each datapoint - vote  $V_{bi}^L$  and then updating  $C^{L+1}$  in a serial manner would have been very slow.

We need a new update rule that can be parallel, across parent capsules and across batch items.

$$D^L.\text{Shape}: [B, n^L, n^{L+2}, d^{L+1}]$$

Initialize mask  $M^L$  with ones,  $M^L.\text{Shape}: [B, n^L, n^{L+2}, d^{L+1}]$

for each  $\theta$  in  $\Theta$ :

Our metric for similarity is the inner product.  
 By this criterion we pick the BMUs (winners).

$$\forall b \in [1, B], \forall i \in \Omega_L: \text{Sims}_{bi}^L \leftarrow \sum_k^L \left[ (V_{bi}^L * M_{bi}^L) * C^{L+1} \right]_{n^{L+2}, d^{L+1}}$$

$\sum_k^L$

$\text{Sims}^L.\text{Shape}: [B, n^L, n^{L+2}]$

$\text{Sims}^L$  contains all the similarities between the parent capsules  $C^{L+1}$  and their corresponding pose-predictions (votes  $V^L$ ), for each batch.

Take note that if `reduced_votes == False`, then all the votes (pre- $V^L$ ) are compared to all the parent capsules. So no vote is tied to a particular parent capsule. We leave it to the SOM-based routing to make the discrimination. In other words, if `reduced_votes == False`,  $C^{L+1}$  have same "view" of data.

This is another difference to the original SOM, the criterion used there is the Euclidean Distance.

$$V_{bi}^L: \begin{matrix} & & & \\ & & & \\ & & & \end{matrix} \quad C^{L+1}: \begin{matrix} & & & \\ & & & \\ & & & \end{matrix}$$

$\sum_{k=1}^{n^{L+2}}$

$$M_{bi}^L: \begin{matrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \end{matrix} \quad \sum_{k=1}^{n^{L+2}}$$

$$V_{bi}^L * M_{bi}^L * C^{L+1}: \begin{matrix} & & & \\ & & & \\ & & & \end{matrix} \quad \sum_{k=1}^{n^{L+2}}$$

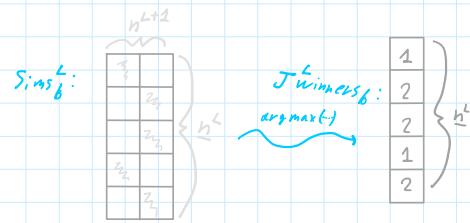
$$\text{Sims}_{bi}^L: \begin{matrix} & & & \\ & & & \\ & & & \end{matrix} \quad \sum_{k=1}^{n^{L+2}} \quad (\text{let } n^L = 5)$$

If not softmax nor tanh-like

We proceed by finding the Best Matching Units (winners). Competition is, as it should be, between the  $C^{L+1}$  as they are trying to explain the datapoints

In our SOM-analogy, the "datapoints" are in our case the votes while the "nodes" are the parent capsules  $C^{L+1}$ .

Matching Units (winners). Competition is, as it should be, between the  $C^{L+1}$  as they are trying to explain the datapoints (votes). The parent capsule that explains best a child capsule  $C_b^L$  (i.e. has the largest similarity with the corresponding vote) wins.



$$\forall b \in [1, B], \forall i \in \mathcal{Q}_L : J^{winners}_{bi} \leftarrow \underset{j}{\operatorname{argmax}}(s_{imsb_{ij}})$$

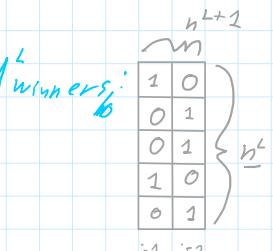
$J^{winners}.shape : [B, n^L]$

$J^{winners}$  contains one winner index per child capsule.

$$\forall b \in [1, B] : M^{winners}_b \leftarrow \operatorname{OneHot}(J^{winners}_{bi} | h^{L+1})$$

$M^{winners}_b$  is a mask that has 1's on the positions of  $s_{imsb}$  where the maximum value (across parent capsules) is located.

$$M^{winners}.shape : [B, n^L, h^{L+1}]$$



else if softmax:

we produce "soft" winners.

$$\forall b \in [1, B] : M^{winners}_b \leftarrow \underset{\text{across;}}{\operatorname{Softmax}}(s_{imsb}^L)$$

$$\text{Where } M^{winners}.shape : [B, n^L, h^{L+1}]$$

$$\text{and } \underset{\text{across;}}{\operatorname{Softmax}}(s_{imsb}^L) = \begin{bmatrix} \operatorname{softmax}(s_{imsb_1=1}^L) \\ \operatorname{softmax}(s_{imsb_1=2}^L) \\ \vdots \\ \operatorname{softmax}(s_{imsb_1=n^L}^L) \end{bmatrix}$$

else if tanh-like

With tanh-like dissimilar parent capsules will point further away.

$$\forall b \in [1, B] : M^{winners}_b \leftarrow \left( \underset{\text{across;}}{\operatorname{Softmax}}(s_{imsb}^L) - 0.5 \right) * 2$$

where "-" and "\*" are pointwise operations.

$$\text{Again, } M^{winners}.shape : [B, n^L, h^{L+1}]$$

If not take-into-account-similarity and not take-into-account-win-ratio:

Original, simple case: Compute the update of the winning nodes (capsules). For the first iteration of thetas, we will update the winners. On the next iterations we will update the winners' 1st degree neighbours. On the third iteration of thetas (if, of course the hyperparameter  $\Theta$  contains 3 items) we will update the 2nd degree neighbours of the winning nodes.

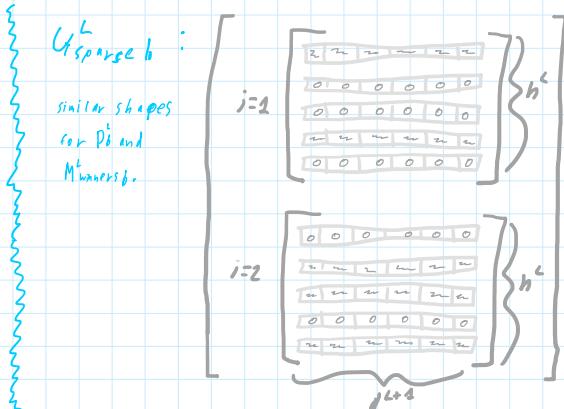
$$\forall b \in [1, B], \forall i \in \mathcal{Q}_L : U_{sparse}^L_{bi} \leftarrow \left( \frac{M^{winners}_b \times P^L_{bi}}{\sum_{j=1}^{n^{L+1}} M^{winners}_{bj}} \right) * L_{SOM} * \theta$$

$$\forall b \in [1, B], \forall i \in \Omega_L : U_{sparse}^L_{bi} \leftarrow \underbrace{\left( M_{\text{winners}}^L_{bi} \times D_{bi}^L \right)}_{\substack{[B \times 1] \\ [n^{L+1}, d^{L+1}]}} * \text{lr-SOM} * \theta$$

pointwise  
Masked Differences

$U_{sparse}^L$  contains the updates of selected capsules. If on the first iteration, then  $U_{sparse}^L$  contains the updates of the winner capsules.

$U_{sparse}^L$ . Shape:  $[B, h^L, h^{L+1}, d^{L+1}]$



As you can see (I suppose), each parent capsule  $C_j^{L+1}$  will be updated according to the capsules  $C_{bi}^L$  that it attracted.

Child capsules that a parent capsule  $C_j^{L+1}$  did not win, will not contribute to its update (thus, their row will be zero).

That is the case for the 1st iteration.

On the next iteration, the capsule  $C_j^{L+1}$  that won children capsule  $C_{bi}^L$  can not "claim" that capsule again. On the second iteration, the winner capsule for datapoint  $C_{bi}$  will now be the parent capsule  $C_j^{L+1}$  that has the second best similarity.

The above note is another major difference to the Original SOM algorithm. Lateral distance, in our case, is defined as the rank distance on the similarity "leaderboard".

### else if take\_into\_account\_similarity:

We will perform similar operations as before. The difference now is that we will scale the Mask of the winners according to the similarities. In that way, the gravity of the update will be proportional to the agreement.

$$\bullet \text{Simsparse}^L \leftarrow \text{Sims} * M_{\text{winners}}^L$$

$[B, h^L, n^{L+1}] \xrightarrow{\text{pointwise}} [B, h^L, h^{L+1}]$

$\text{Sims Sparse. Shape: } [B, h^L, h^{L+1}]$

$$\bullet b \in [1, B], \forall i \in \Omega_L : U_{sparse}^L_{bi} \leftarrow \underbrace{\left( \text{Simsparse}_{bi}^L \times D_{bi}^L \right)}_{\substack{[B \times 1] \\ [h^{L+1}, d^{L+1}]}} * \text{lr-SOM}$$

pointwise  
Scaled, Masked Differences

$U_{sparse}^L$ . Shape:  $[B, h^L, h^{L+1}, d^{L+1}]$

No need to scale more using  $\theta$ . Similarity-scaling does this for us. (as we go from iteration to iteration, similarities will decrease)

### If take\_into\_account\_win\_ratio :

This is another update-scaling term that will be used later.

$$\bullet \text{WinCountPartial}^L \leftarrow \sum_i^B M_{\text{winners}}^L$$

$[n^{L+1}]$

Contains the number of wins that each

$$* \text{WinCountPartial}^L \leftarrow \sum_i^B \sum_b M_{\text{winners}}^L$$

Contains the number of wins that each parent capsule made in one iteration of  $\theta$ .

$$\text{WinCountPartial Shape: } [n^{L+1}]$$

$$* \text{WinCount} \leftarrow \text{WinCount} + \text{WinCountPartial}$$

$\hookrightarrow$  contains the number of  $C_i^L$ 's that each  $C_j^{L+1}$  "won" (in all iterations).

If not softmax and not tanh-like:

Update mask  $M^L$  so that in the next iteration of  $\theta$  (neighbors with distance 1), we will not have the same winners. In other words, we want to find the neighbors of the winner capsules, not the winners themselves.

$$* \forall b \in [1, B], \forall i \in \underline{\Omega}_L, \forall j \in \Omega_{L+1}: M_{bij}^L \leftarrow M_{bij}^L - M_{\text{winners}}^L$$

pointwise  
[1]  $\hookrightarrow$   $bij$

$$M \text{ shape: } [B, h^L, h^{L+1}, d^{L+1}]$$

$$[B, h^L, h^{L+1}, d^{L+1}]$$

$M$  has now zeros at the places where  $M_{\text{winners}}$  had ones.

Aggregate the updates across thetas.

$$* SU^L \leftarrow SU^L + U^L_{\text{spars}}$$

$\downarrow$  partial updates  
 $\downarrow$  total updates

$$SU^L \text{ shape: } [B, h^L, h^{L+1}, d^{L+1}]$$

End of neighbour loop.

Make updates dense by finding, for each parent capsule  $C_j^{L+1}$  the mean of all the votes from which capsules he won (across batch size). This is done to make less updates but more "educated".

$$* U^L \leftarrow \frac{\sum_b \sum_i SU_{bi}^L}{B * h^L}$$

$U^L \text{ shape: } [h^{L+1}, d^{L+1}]$

Time to change the DigitCaps (perform update step).

If not take-into-account-win-ratio:

If not radical:

Basic, "simple" case:

$$* C^{L+1} \leftarrow C^{L+1} + U^L$$

$$C^{L+1} \text{ shape: } [h^{L+1}, d^{L+1}]$$

> else if radical:

As you can see,  $C^{L+1}$  is not dependent on a single input. Rather, it changes given a batch at training. So, it is not a prediction. The capsule vectors  $C^{L+1}$  are learned to represent the invariant properties of the image features. With these instantiations

$C$ . Shape:  $[h^{L+1}, d^{L+1}]$

else if radical:

Average over  $r$  iterations.

$$* C^{L+1} \leftarrow \frac{C^{L+1} * (r-1) + U^L}{r}$$

The capsule vectors  $C^{L+1}$  are learned to represent the invariant properties of the image features. With these instantiation invariant vectors the votes are compared and the similarities are produced. As we will see in a moment, these aggregated similarities are also the output of our model. Please note that during testing, preferably  $r=0$  so as to not update  $C^{L+1}$  anymore.

else: Scale updates relative to the number of wins.

$$* WinRatio^L \leftarrow \frac{WinCount^L}{n^L * B} \quad WinRatio.shape: [h^{L+1}]$$

$$* SoftWinRatio^L \leftarrow \text{softmax}(WinRatio^L) \quad \begin{cases} \text{SoftWinRatio is like assignment} \\ \text{probabilities.} \end{cases}$$

$$* \forall j \in \Omega_{L+1}: U_j^L \leftarrow SoftWinRatio_j^L * U_j^L \quad \begin{matrix} \text{shape} \\ [h^{L+1}, d^{L+1}] \end{matrix}$$

If not radical:

Basic, "simple" case:

$$* C^{L+1} \leftarrow C^{L+1} + U^L$$

$C$ . Shape:  $[h^{L+1}, d^{L+1}]$

else if radical:

Average over  $r$  iterations.

$$* C^{L+1} \leftarrow \frac{C^{L+1} * (r-1) + U^L}{r}$$

- If normalize-in-loop:

Normalize digit capsules before next iteration.

if norm-type == 0:

$$\cdot \forall j \in \Omega_{L+1}: C_j^{L+1} \leftarrow \text{Squash}(C_j^{L+1})$$

else:

$$\cdot \forall j \in \Omega_{L+1}: C_j^{L+1} \leftarrow \underbrace{\tanh(\|C_j^{L+1}\|)}_{\text{Scaling term}} \underbrace{\frac{(C_j^{L+1})}{\|C_j^{L+1}\|}}_{\text{Unit norm}}$$

Repeat if  $r > 1$

L

End of  $L$  iterations

Compute the similarities now, using the updated  $C^{L+1}$ .

$$*\forall b \in [1, B], \forall i \in \mathcal{I}_L : \text{FinalSims}_{bi}^{L+1} \leftarrow \sum_{k=1}^{n^{L+1}} (\mathbf{V}_{bi}^L * C^{L+1})_{bit}$$

Shape:  $[B, n^L, n^{L+1}]$

$$*\forall b \in [1, B] : \text{SimsOut}_b^{L+1} \leftarrow \frac{\sum_i \text{FinalSims}_{bi}^{L+1}}{n^L}$$

Shape:  $[B, n^{L+1}]$

\* Return  $\text{SimsOut}^{L+1}, C^{L+1}$

In test mode, these two lines below may be the only steps of the algorithm after computing the votes.

For a behaviour that closely resembles the one of capsules, we can set batch size = 1 and use a bigger LK-SOM. In this way, capsules  $C^{L+1}$  will capture the equivariant properties of the data.