We discuss the evaluation of $(v_i^i)^T P^m_\phi$
In the backward process, the following operation is applied.

\[(\mathbf{v}^i)^T P^m_\phi,\]  

where

\[
\mathbf{v}^i = \text{vec} \left( (\mathbf{W}^m)^T \frac{\partial \xi_i}{\partial S_{m,i}} \right) \]  

Consider the same example used for explaining \(\phi(Z^{in,i})\)
We have

\[
(\nu^i)^T P^m_\phi \parallel
\]

\[
P^m_\phi = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]
Thus

$$(v^i)^T P^m_\phi = [v_1 \ v_2 + v_5 \ v_6 \ v_3 \ v_4 + v_7 \ v_8], \quad (3)$$

which is a kind of “inverse” operation of $\phi(\text{pad}(Z^{m,i}))$.

We accumulate elements in $\phi(\text{pad}(Z^{m,i}))$ back to their original positions in $\text{pad}(Z^{m,i})$. 

Chih-Jen Lin (National Taiwan Univ.)
In MATLAB, given indices

\[
\begin{bmatrix} 1 & 2 & 4 & 5 & 2 & 3 & 5 & 6 \end{bmatrix}^T
\]

(4)

and the vector \( \mathbf{v} \), a function `accumarray` can directly generate the vector (3).

Example:
\[(\mathbf{v}^i)^T P^m_{\phi} \mathbf{V}\]

octave:18> [v a]
ans =

1   0.406445
2   0.067872
4   0.036638
5   0.279801
2   0.490535
3   0.369743
5   0.429186
6   0.054324
\[(\mathbf{v}^i)^T P^m_{\phi} \mathbf{V} \]

octave:19> accumarray(v,a)
ans =

0.406445
0.558407
0.369743
0.036638
0.708987
0.054324
\[(v^i)^{T} P^m_{\phi} \] VII

- We can see that the second position is
  \[a(2) + a(5)\]
  \[= 0.067872 + 0.490535\]
  \[= 0.558407\]

- To do the calculation over a batch of instances, we aim to have
  \[
  \begin{bmatrix}
  (v^1)^{T} P^m_{\phi} \\
  \vdots \\
  (v^l)^{T} P^m_{\phi}
  \end{bmatrix}^{T} \Rightarrow \text{a vector} \begin{bmatrix}
  (P^m_{\phi})^{T} v^1 \\
  \vdots \\
  (P^m_{\phi})^{T} v^l
  \end{bmatrix} (5)
  \]
We can apply MATLAB's `accumarray` on the vector

$$v^T P^m \phi$$

by giving the following indices as the input.
\[
(\mathbf{v}^i)^T P^m_\phi \mathbf{X}
\]

where

\[
a^m_{\text{pad}} b^m_{\text{pad}} d^m
\]

is the size of \( \text{pad}(Z^m,i) \)

and

\[
h^m h^m d^m a^m_{\text{conv}} b^m_{\text{conv}}
\]

is the size of \( \phi(\text{pad}(Z^m,i)) \) and \( \mathbf{v}_i \).

That is, by using the offset \( (i - 1)a^m_{\text{pad}} b^m_{\text{pad}} d^m \), accumarray accumulates \( \mathbf{v}^i \) to the following positions:

\[
(i - 1)a^m_{\text{pad}} b^m_{\text{pad}} d^m + 1, \ldots, ia^m_{\text{pad}} b^m_{\text{pad}} d^m.
\]
\[(\nu^i)^T P^m \phi X\]

- (7) can be easily obtained by the following outer sum

\[\text{vec}((4) + [0 \ldots l - 1] a^m_{\text{pad}} b^m_{\text{pad}} d^m)\]

- To obtain

\[
\begin{bmatrix}
\nu^1 \\
\vdots \\
\nu^l
\end{bmatrix}
\]

we note from (2) that it is the same as

\[\text{vec} \left( (W^m)^T \left[ \frac{\partial \xi_1}{\partial S^{m,1}} \ldots \frac{\partial \xi_l}{\partial S^{m,l}} \right] \right).\]
Thus we do a matrix-matrix multiplication

From (9), we have a reason that in our implementation

\[
\frac{\partial \xi_i}{\partial \text{vec}(S_{m,i})^T}
\]

over a batch of instances are stored in the form of

\[
\left[ \frac{\partial \xi_1}{\partial S_{m,1}} \ldots \frac{\partial \xi_l}{\partial S_{m,l}} \right] \in \mathbb{R}^{d_{m+1} \times a_{conv}^m b_{conv}^l}.
\]
A Simple Code I

```matlab
a_prev = model.ht_pad(m);
b_prev = model.wd_pad(m);
d_prev = model.ch_input(m);

idx = net.idx_phiZm(:) +
    [0:num_v-1]*d_prev*a_prev*b_prev;
vTP = accumarray(idx(:), V(:),
                  [d_prev*a_prev*b_prev*num_v 1])';
```
Here we assume $V = [v_1 \ldots v_l]$ and `num_v` is the number of columns.

Note that the third parameter of `accumarray` is to specify the size of the resulting vector as some entries may not accumulate any value (so we have zero there).
Discussion I

- If a package provides efficient implementations of the following operations
  - matrix-matrix products
  - matrix expansion for $\phi(\text{pad}(Z^{m,i}))$
  - outer sum
  - accumarray

then we can easily have a good CNN implementation

- Unfortunately, the difficulty to optimize these operations may vary
Discussion II

- To work on instances together, it’s difficult to decide the best storage settings
- Further, storage settings affect the implementations