Quaternions Capsule Networks

Barış Özcan, Furkan Kınılı, Furkan Kırcağız
Özyeğin University
Problem

• CNNs require many samples for a specific object to generalize to geometric variations such as novel viewpoints.

• The intrinsic structure of an object is not modelled in the architecture in terms of part-whole relationships.

• This deficiency is proven by the experiments in novel viewpoint setup of smallNORB dataset.

• Capsule Networks address this problem however, using matrix transformations have several disadvantages.
Capsule Networks

• Aims to achieve generalization by modelling the hierarchical structure of parts for object classes in the data.

• Capsule is an encapsulation of multiple neurons allows to flow complex information between layers (i.e. pose, feature vector).

• Each part capsule votes for parent capsules pose by transforming part capsules pose. These transformations are learned during training.

• Parent capsule pose and activations are calculated via clustering-like routing mechanisms over the incoming votes. (i.e. EM-Routing, dynamic routing).
Our assumption

• We constraint the pose representation to 3D rotations between capsules since convolutional connections are translation invariant.

• In the object coordinate system the intrinsic rotations $w'_{ij}$ and $w_{ij}$ between parts and whole is constant from any viewpoint. Thus if we can learn this rotation from the data the network should generalize to novel viewpoints.
Quaternions and 3D rotation

• 4-dimensional complex numbers that represents efficient rotation computation.

• 3D rotation is represented with quaternion product.
  • Rotated quaternion(pose) is a pure quaternion.
  • Rotating quaternion(learned) is a unit quaternion
Advantages of quaternion rotations

• Quaternions do not suffer from gimbal lock.
• Rotation matrices must be orthogonal which is harder than normalizing quaternions to ensure a proper rotation.
• Using quaternions reduces the number of parameters from 9 to 4 for each represented rotation in the network.
Quaternion Capsules

• Each capsule is represented as a pure quaternion.
• Each connection is a learned quaternion rotation.
• Routing is via EM-Routing.
• Learned quaternions are normalized similarly to weight normalization technique.
• Each capsule contains its pose and activation.
Network architecture

Pose branch

Activation branch

Composition of Primary Capsules

# Caps: 32

# classes

output
Generalization to novel viewpoints experiment

- Training and test samples are from different viewpoints of the classes in smallNORB dataset.
- Azimuth experiment: training and test set are divided w.r.t. azimuth angles
- Elevation experiment: training and test set are divided w.r.t. elevations
- Training is stopped where training performances are similar for each model to test generalization fairly.
## Results on novel viewpoints

<table>
<thead>
<tr>
<th>Viewpoints (Models)</th>
<th>Azimuth (%)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QCN</td>
<td>EM*</td>
<td>VB</td>
<td>EM</td>
<td>CNN</td>
</tr>
<tr>
<td>Novel</td>
<td>7.5</td>
<td>13.4</td>
<td>11.3</td>
<td>13.5</td>
<td>20.0</td>
</tr>
<tr>
<td>Familiar</td>
<td>3.7</td>
<td>3.7</td>
<td>3.7</td>
<td>3.7</td>
<td>3.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Viewpoints (Models)</th>
<th>Elevation (%)</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QCN</td>
<td>EM*</td>
<td>VB</td>
<td>EM</td>
<td>CNN</td>
</tr>
<tr>
<td>Novel</td>
<td>11.5</td>
<td>15.8</td>
<td>11.6</td>
<td>12.3</td>
<td>17.8</td>
</tr>
<tr>
<td>Familiar</td>
<td>4.4</td>
<td>4.0</td>
<td>4.3</td>
<td>4.3</td>
<td>4.3</td>
</tr>
</tbody>
</table>
## Results on common datasets

<table>
<thead>
<tr>
<th>Models</th>
<th>smallNORB</th>
<th>MNIST</th>
<th>FashionMNIST</th>
<th>SHVN</th>
<th>CIFAR-10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error (%)</td>
<td># of Params</td>
<td>Error (%)</td>
<td># of Params</td>
<td>Error (%)</td>
</tr>
<tr>
<td>EM [7]</td>
<td>1.8</td>
<td>~310K</td>
<td>0.44</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>EM-IBM [37]</td>
<td>4.6</td>
<td>~335K</td>
<td>1.23$^*$</td>
<td>~337K$^*$</td>
<td>10.44$^*$</td>
</tr>
<tr>
<td>VB [8]</td>
<td>1.6</td>
<td>~169K</td>
<td>–</td>
<td>–</td>
<td>5.2</td>
</tr>
<tr>
<td>EM*</td>
<td>3.40</td>
<td>~317K</td>
<td>0.89</td>
<td>~319K</td>
<td>9.74</td>
</tr>
<tr>
<td>QCN</td>
<td>2.29</td>
<td>~188K</td>
<td>0.37</td>
<td>~187K</td>
<td>6.92</td>
</tr>
</tbody>
</table>
Conclusion

• QCNs generalize better to novel viewpoints in smallNORB.
• QCNs reduce the parameters by a large margin in Capsule Layers with rotation.
• Branching before extracting capsules adds more flexibility to the architecture.
Thank You!

Official implementation in Pytorch is available at https://github.com/Boazrciasn/Quaternion-Capsule-Networks.git