<table>
<tr><th rowspan="2">cases</th><th>doc_1</th><th>doc_2</th><th>decision</th><th>id</th></tr>
<tr>
<td>
<table>
<tr><th>authors</th><td>Amir Gholami
Kurt Keutzer
George Biros</td></tr>
<tr><th>title</th><td>ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs</td></tr>
<tr><th>publication_date</th><td>2019-02-27 01:48:32+00:00</td></tr>
<tr><th>source</th><td>SupportedSources.ARXIV</td></tr>
<tr><th>journal</th><td>None</td></tr>
<tr><th>volume</th><td></td></tr>
<tr><th>doi</th><td></td></tr>
<tr><th>urls</th><td>http://arxiv.org/pdf/1902.10298v3
http://arxiv.org/abs/1902.10298v3
http://arxiv.org/pdf/1902.10298v3</td></tr>
<tr><th>id</th><td>id-2155191757629502724</td></tr>
<tr><th>abstract</th><td>Residual neural networks can be viewed as the forward Euler discretization of an Ordinary Differential Equation (ODE) with a unit time step. This has recently motivated researchers to explore other discretization approaches and train ODE based networks. However, an important challenge of neural ODEs is their prohibitive memory cost during gradient backpropogation. Recently a method proposed in [8], claimed that this memory overhead can be reduced from O(LN_t), where N_t is the number of time steps, down to O(L) by solving forward ODE backwards in time, where L is the depth of the network. However, we will show that this approach may lead to several problems: (i) it may be numerically unstable for ReLU/non-ReLU activations and general convolution operators, and (ii) the proposed optimize-then-discretize approach may lead to divergent training due to inconsistent gradients for small time step sizes. We discuss the underlying problems, and to address them we propose ANODE, an Adjoint based Neural ODE framework which avoids the numerical instability related problems noted above, and provides unconditionally accurate gradients. ANODE has a memory footprint of O(L) + O(N_t), with the same computational cost as reversing ODE solve. We furthermore, discuss a memory efficient algorithm which can further reduce this footprint with a trade-off of additional computational cost. We show results on Cifar-10/100 datasets using ResNet and SqueezeNext neural networks.</td></tr>
<tr><th>versions</th><td></td></tr>
</table>
</td>
<td>
<table>
<tr><th>authors</th><td>A. Gholami
K. Keutzer
G. Biros</td></tr>
<tr><th>title</th><td>ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs</td></tr>
<tr><th>publication_date</th><td>2019-02-27 00:00:00</td></tr>
<tr><th>source</th><td>SupportedSources.SEMANTIC_SCHOLAR</td></tr>
<tr><th>journal</th><td></td></tr>
<tr><th>volume</th><td></td></tr>
<tr><th>doi</th><td>10.24963/ijcai.2019/103</td></tr>
<tr><th>urls</th><td>https://www.semanticscholar.org/paper/8cc20e88e7da6a1055ab9c617e495efcabb1628a</td></tr>
<tr><th>id</th><td>id82227827608381077</td></tr>
<tr><th>abstract</th><td>Residual neural networks can be viewed as the forward Euler discretization of an Ordinary Differential Equation (ODE) with a unit time step. This has recently motivated researchers to explore other discretization approaches and train ODE based networks. However, an important challenge of neural ODEs is their prohibitive memory cost during gradient backpropogation. Recently a method proposed in arXiv:1806.07366, claimed that this memory overhead can be reduced from LNt, where Nt is the number of time steps, down to O(L) by solving forward ODE backwards in time, where L is the depth of the network. However, we will show that this approach may lead to several problems: (i) it may be numerically unstable for ReLU/non-ReLU activations and general convolution operators, and (ii) the proposed optimize-then-discretize approach may lead to divergent training due to inconsistent gradients for small time step sizes. We discuss the underlying problems, and to address them we propose ANODE, a neural ODE framework which avoids the numerical instability related problems noted above. ANODE has a memory footprint of O(L) + O(Nt), with the same computational cost as reversing ODE solve. We furthermore, discuss a memory efficient algorithm which can further reduce this footprint with a tradeoff of additional computational cost. We show results on Cifar-10/100 datasets using ResNet and SqueezeNext neural networks.</td></tr>
<tr><th>versions</th><td></td></tr>
</table>
</td>
<td>DUPLICATES</td>
<td>303</td>
</tr>
</table>