Tiny Neural Deep Clustering: An Unsupervised Approach for Continual Machine Learning on the Edge
Andrea Albanese¹, Matteo Nardello¹, and Davide Brunelli¹
¹Department of Industrial Engineering, University of Trento, Trento, Italy.

Continual Machine Learning (CML)
CML permits automatic updates of ML systems during inference.
• Overcome the context drift problem.
• Avoid system maintenance due to model re-training.
• Learn new patterns and classes.
• Adapt to environment and context changes.
However, CML systems are challenging:
• Catastrophic forgetting.
• Computationally demanding for tinyML systems.

Tiny Neural Deep Clustering (TinyNDC)
TinyNDC is a fast and light architecture that implements unsupervised CML on microcontrollers.
It is composed of three main components:
• The Frozen model.
• The Active model.
• The Online deep clustering.

State of the Art
Current state-of-the-art CML strategies (e.g., CWR) [1], LWF [2], and tinyOL [3] well adapt to new data.
→ Limi: they need a ground truth for carrying out the backpropagation.
Other works overcome this limitation:
→ Limi: no embedded implementation.
Only the work “On-device training under 256k memory” [6] presents an embedded implementation but in a supervised setting.
What is missing is an embedded solution for implementing unsupervised CML to get closer to real-world exploitation.

Frozen model
The static part of the system which is not updated during runtime. It is composed of two sub-modules, namely the consolidated layer and the training layer.
• The consolidated layer acts as a memory to not forget the previous knowledge.
• The training layer is used for domain adaptation.

Active model
The dynamic part of the system which is updated during runtime. It is composed of an NN truncated in the last layer (e.g., the Softmax classifier).
• Its parameters are fixed after training and cannot be trained in real time.
• Its objective is to extract features from incoming data.

Online deep clustering
A custom algorithm that uses L2-norm to group data.
It is fed with the output of the frozen model and produces the pseudo-label estimation.
• The pseudo-label is fundamental for updating the active model.
• The clustering can update its centroids by using the prediction of the active model.
• The clustering's centroids need to be initialized with a small amount of data (i.e., 10 samples per class).

Case study
Tiny NDC is compared with the same system in a supervised setting (Supervised CML) (i.e., without the pseudo-label estimation with clustering), and with the clustering algorithm acting as an unsupervised CML system (Unsupervised CML).

Experimental results
TinyNDC is implemented in an unsupervised setting. This research proves:
• The real-time execution of such a system in an embedded device is feasible.
• The light and fast performance of the system, even though the high complexity due to the pseudo-label estimation.

2. Measure the execution time of each task to ensure real-time capability and possible employment in real-world applications.

Experimental results
<table>
<thead>
<tr>
<th>Task</th>
<th>Supervised CML (ms)</th>
<th>Unsupervised CML (clustering) (ms)</th>
<th>Tiny NDC (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frozen model</td>
<td>16.20</td>
<td>16.34</td>
<td>16.20</td>
</tr>
<tr>
<td>Clustering update</td>
<td>0.18</td>
<td>0.16</td>
<td>0.13</td>
</tr>
<tr>
<td>Active model</td>
<td>1.58</td>
<td>0.22</td>
<td>0.20</td>
</tr>
<tr>
<td>Active model update</td>
<td>2.54</td>
<td>0.24</td>
<td>2.54</td>
</tr>
<tr>
<td>TOTAL</td>
<td>20.10</td>
<td>10.79</td>
<td>22.50</td>
</tr>
</tbody>
</table>

Conclusion
TinyNDC implements CML in an unsupervised setting. This research proves:
• The real-time execution of such a system in an embedded device is feasible.
• The light and fast performance of the system, even though the high complexity due to the pseudo-label estimation.

The system reaches:
• 99.3% of learning accuracy.
• Execution at 64 FPS.

Future improvements will include:
• Update of the inner layers.
• Add clues for new classes.

References