OnDevice Learning

Hahn-Schickard
Marcus.rueb@hahn-schickard.de
What is OnDevice Training and why is it needed?

Advantages [1]:

- Privacy
- Continuously Learning
- Energy-saving
- Less Communication
- Personalized Models
Applications

Mobile Applications e.g. Pacemaker

Production

Predictive Maintenance

Medical

Robotics

End Products

IOT

Mechanical Engineering

Hahn-Schickard - Marcus Rüb - Marcus.rueb@hahn-schickard.de
What does the workflow look like?

Train Model → Deploy Model → Collect Data → Retrain Model

Hahn-Schickard - Marcus Rüb - Marcus.rueb@hahn-schickard.de
Problems of OnDevice Training

01 Limited Memory
02 Limited Calculation capacity
03 Limited Energie
04 Data Selection
05 Catastrophic Forgetting
Problems of OnDevice Training

01 Limited Memory
02 Limited Calculation capacity
03 Limited Energie
04 Data Selection
05 Catastrophic Forgetting
Data Point Selection

Which data use to train?
How to label Data on Device?
What to do with „old data“? [2]
Active Learning

1. **Least Confidence**: difference between the most confident prediction and 100% confidence
2. **Margin of Confidence**: difference between the top two most confident predictions
3. **Ratio of Confidence**: ratio between the top two most confident predictions
4. **Entropy**: difference between all predictions, as defined by information theory \[^3\]

---

Hahn-Schickard - Marcus Rüb - Marcus.rueb@hahn-schickard.de
Problems of OnDevice Training

01 Limited Memory
02 Limited Calculation capacity
03 Limited Energie
04 Data Selection
05 Catastrophic Forgetting

Hahn-Schickard - Marcus Rüb - Marcus.rueb@hahn-schickard.de
Train more efficient

Backpropagation
100% propagation rate

Sparse Backpropagation [4]
50% propagation rate

Gradient of Input
Hidden Layer
Gradient of Output

Gradient of Input
Hidden Layer
Gradient of Output

Hahn-Schickard - Marcus Rüb - Marcus.rueb@hahn-schickard.de
Train more efficient

Backpropagation
100% propagation rate

\[ \delta_w^l \quad \delta_z^l \quad (a^{l-1})^T \]

Sparse Backpropagation
50% propagation rate

\[ \hat{\delta}_w^l \quad \text{top}(\delta_z^l, k) \quad (a^{l-1})^T \]
Train more efficient

1. Forward Propagation

\[ z^l = W^l \cdot a^l + b^l \]
\[ a^{l+1} = f(z^l) \]

2. Sum magnitudes of local error

\[ Y^l = \sum_{i=1}^{N^l} |\delta_{a,i}^l| \]

3. Decide to train Datapoint

\[ D^l = \left( D_{\text{min}} + \alpha^l \cdot \left( \frac{D_{\text{max}} - D_{\text{min}}}{\alpha_{\text{max}}} \right) \right) \cdot \beta^l \]

4. Calculate k

\[ S^l := \left( S_{\text{min}} + Y^l \cdot \frac{S_{\text{max}} - S_{\text{min}}}{Y_{\text{max}}} \right) \cdot \zeta^{l-1} \]
\[ k^l = S^l \cdot N^l \]

5. Get Top k

(Top k = 2)

6. Sparse back Propagation

(Top k = 2)

Hahn-Schickard - Marcus Rüb - Marcus.rueb@hahn-schickard.de
Problems of OnDevice Training

01 Limited Memory
02 Limited Calculation capacity
03 Limited Energie
04 Data Selection
05 Catastrophic Forgetting
Proposed Workflow

1. Distill Data
2. Train Neural Network
3. Check model accuracy
4. Collect new data
5. Increase model size

If the model size is insufficient, return to step 1.

[5]
What is Dataset distillation

50K Real Training Images

Dataset Distillation

10 Synthetic Training Images

Train

Train

Similar Test Performance
Figure 3. We perform long-range parameter matching between training on distilled synthetic data and training on real data. Starting from the same initial parameters, we train distilled data $D_{syn}$ such that $N$ training steps on them match the same result (in parameter space) from much more $M$ steps on real data.
Train Neural Network

Backpropagation

1. Error - difference between predicted output and actual output

2. Error is sent back to each neuron in backward direction

3. Gradient of error is calculated with respect to each weight

Input Layer  Hidden Layer  Output Layer

W

W

W

x1

x2

x3

Outputs

Predicted output

Error
Check Model Accuracy & Increase Model size

Model accuracy is typically measured using an accuracy score, which is defined as the proportion of correct predictions out of the total number of predictions. If we denote the number of correct predictions as $C_{\text{correct}}$ and the total number of predictions as $N_{\text{total}}$, the accuracy score $A$ can be calculated as follows:

$$ A = \frac{C_{\text{correct}}}{N_{\text{total}}} $$

Let’s denote our original model’s architecture as $\mathcal{M}$ and its accuracy score as $A$. If $A$ doesn’t meet the predefined accuracy standard, $A_{\text{standard}}$, we proceed with the enlargement of the model, creating a new model architecture, $\mathcal{M}'$. This adjustment can be represented as follows:

$$ \mathcal{M}' = \begin{cases} 
\mathcal{M} + \Delta \mathcal{M} & \text{if } A < A_{\text{standard}} \\
\mathcal{M} & \text{otherwise}
\end{cases} $$

where $\Delta \mathcal{M}$ represents the increase in the model’s complexity. This increase can be in the form of additional layers, more
References

[1] Marcus Rüb, Prof. Dr. Axel Sikora, A Practical View on Training Neural Networks in the Edge, IFAC-PapersOnLine, Volume 55, Issue 4, 2022, Pages 272-279, ISSN 2405-8963
[3] https://towardsdatascience.com/uncertainty-sampling-cheatsheet-ec57bc067c0b
Thank you