Universität Augsburg
Fakultät für Angewandte Informatik

Deep Learning für Embedded Systems

Praktikum: Programming Parallel Embedded Systems

David Heim, Benedikt Bauer

14.02.2023

Aufgabenstellung

Bibliothek für Neuronale Netze

- Erstellung von neuronalen Netzen mit dieser Bibliothek
- Training dieser Netze auf x86
- Anwendung dieser Netze auf dem Raspberry Pi Pico

Optimisierungen/Erweiterungen

- Ausnutzen des Flash Speichers anstelle des RAMs
- Network Pruning
- Quantisierung

Implementierung der Bibliothek

Einfaches Model zur Bestimmung von XOR

```
auto model = nn::Sequential<float>();
model.add(nn::Linear<float>(2, 3));
model.add(nn::Sigmoid<float>());
model.add(nn::Linear<float>(3, 1));
model.init();
auto optimizer = nn::SGD<float>(model.params(), 0.25);
for (int epoch = 0; epoch < 2000; epoch++) {
  auto& [input, target] = xor_data;
  auto output = model(input);

  auto loss = nn::mse(output, target);

  optimizer.zero_grad();
  loss.backward();

  optimizer.step();
}
```

Größtes Problem der ersten Implementierung

Auf dem Pi Pico 264 kB SRAM verfügbar

```
auto model = nn::Sequential<float>();
model.add(nn::Linear<float>(28 * 28, 100));
```

Für dieses Netzwerk nur für die Gewichte (ohne Bias) :

- 28 * 28 * 100 * 4 Bytes = 313.600 Bytes

Optimierung: Ausnutzung des Flash Speichers

Der Raspberry Pi Pico besitzt 2MB Flash Speicher

- Speichern der Gewichte im Flash
- Für jeden Layer werden nur noch Pointer zu den Gewichten im RAM gehalten

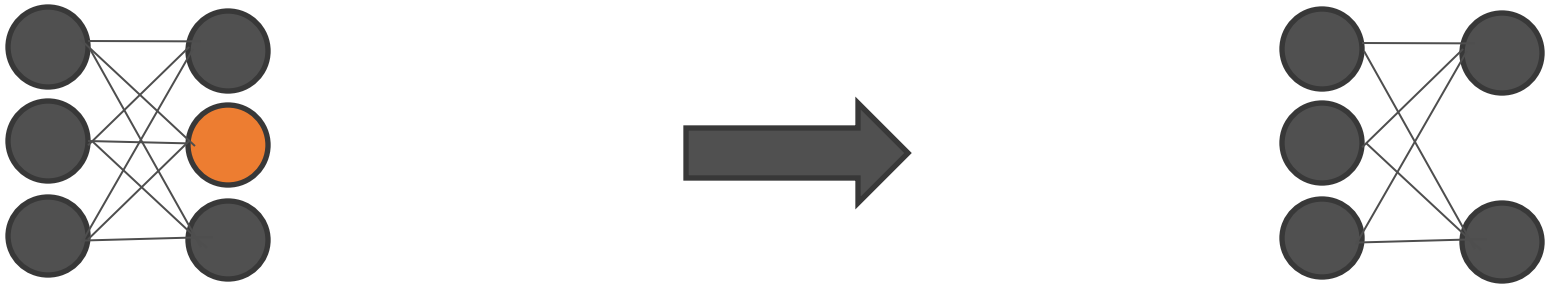
Optimierung: Pruning der Netzwerke

Idee:

- Neuronale Netzwerke beinhalten viele unnötige Informationen
- Lösche die unwichtigsten Informationen aus dem trainierten Netz

Vorteile:

- Trainierte Netze werden kleiner
- Forward-pass wird performanter



Optimierung: Quantisierung

- Gewichte werden nicht als float (4 Byte) sondern als uint8 zusammen mit einem Skalierungsfaktor gespeichert
- Mithilfe des Skalierungsfaktors lässt sich (annähernd) die tatsächliche Zahl errechnen

Vorteile:

- Verringerung des Speicherplatzbedarfs
- Operationen können unter Umständen performanter auf Integerwerten ausgeführt werden

Evaluation: Double vs. Float

	`double`	`float`
Accuracy on full test set (x86)	90.17%	90.16%
Binary size	1.67 MB	1.03 MB
Accuracy on first 20 images of the test set	95%	95%
Inference time	24.85s	19.79s
Time per image	1.24s	0.99s

Evaluation: Verschiedene Größen von Netzwerken

	(50)	(100)	(200)	(300)	(300,100)
Accuracy on full test set (x86)	86.11%	89.02%	90.16%	91.58%	84.89%
Binary size	579kB	738kB	1.03 MB	1.37 MB	1.49 MB
Accuracy on first 50 images of the test set	88%	94%	94%	94%	84%
Inference time	12.6s	24.89s	49.47s	74.05s	83.71s
Time per image	0.25s	0.50s	0.99s	1.48s	1.67s

Evaluation: Pruning

	Original	Pruned
Accuracy on full test set (x86)	92.97%	91.89%
Binary size	1.37 MB	737 kB
Accuracy on first 50 images of the test set	96%	94%
Inference time	73.66s	24.77s
Time per image	1.47s	0.50s

Evaluation: Quantisierung

	Original	Quantized
Accuracy on full test set (x86)	92.97%	10.81%
Binary size	1.37 MB	689 kB
Accuracy on first 50 images of the test set	96%	12%
Inference time	73.66s	73.45s
Time per image	1.47s	1.47s

Vielen Dank für Ihre Aufmerksamkeit