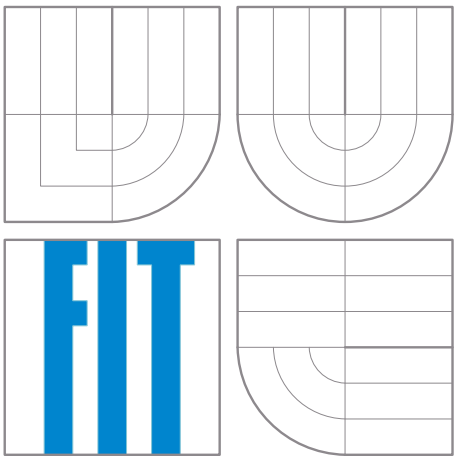

Projekt do předmětu GMU – Grafické a multimediální procesory

Neuronové sítě v OpenCL

5. ledna 2021

Řešitel: Jakub Zárybnický (xzaryb00@stud.fit.vutbr.cz)
Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

1 Zadání

Formální zadání:

- Nastudujte neuronové sítě (back propagation, inference, ...)
- Implementujte framework umožňující konfiguraci topologie sítě, nastavování vlastních aktivačních funkcí, trénování).
- Implementujte jednoduchou aplikaci využívající framework.

Interpretace:

- Několik pokusů o řešení: CUDA (následné zjištění, že se CUDA nedá emulovat), Vulkan (pomocí compute shaders), raw OpenCL v C++ (po správném přečtení zadání "v OpenCL"), raw OpenCL v Haskellu a poslední varianta:
- Poté, co se mé řešení začalo nebezpečně podobat jednomu existujícímu frameworku jsem zvolil alternativu takovou, že tento framework využiji a vylepším/paralelizuji.

2 Použité technologie

- OpenCL
- HMatrix
- Grenade

- Namísto CMake použit Nix, který narozdíl od CMake dobře podporuje Haskell
- Žádné další závoslosti,

3 Použité zdroje

- https://github.com/HuwCampbell/grenade
- https://github.com/IFCA/opencl
- https://github.com/acowley/CLUtil

4 Způsob akcelerace

- Paralelizace procesu trénování i inference sítě v OpenCL (matrix-vector multiplication, matrix-matrix multiplication, outer vector product)
- Konkrétně implementace tří maticových algoritmů pro FCNN
- Následné zjištění, že soupeřím s BLAS/LAPACK v původní verzi software a ladění výkonu FFI mezi Haskell a OpenCL

První pokusy o zrychlení pouze `cbits` částí frameworku — konkrétně gradient descent a konvoluční algoritmy — vedly k pětinásobnému zpomalení a rychle jsem je zavrhl - *overhead* neustálého kopírování dat mezi programem a OpenCL byl příliš vysoký.

Implementoval jsem namísto toho typ vrstvy neuronů, která celá věží na OpenCL, ale je zároveň interoperabilní se zbytkem kódu psaného čistě v Haskellu.

5 Ovládání vytvořeného programu

Je potřeba nainstalovat Nix, konkrétně `nixFlakes`, experimentální Nix verze 3.0, který pak nainstaluje veškeré potřebné závislosti.

Jelikož se má jednat o framework, demonstrační program samotný je triviální a dá se spustit v kořenovém adresáři uvnitř prostředí `nix shell` pomocí

```
cabal run grenade-examples:exe:feedforward
```

popř. pro profilování je možné použít

```
cabal run --enable-profiling -- grenade-examples:exe:feedforward +RTS -p -s
```

6 Vyhodnocení

Testováno na `intel-ocl` na vestavěné grafické kartě, jelikož nemám přístup k GPU zařízením.

Rychlost akcelerovaných vrstev je přibližně konstatní při rostoucí velikosti, nebo roste zanedbatelně, zatímco rychlost vrstev používajících BLAS roste lineárně — zřejmě FFI *overhead* je při použití příliš malých matic příliš vysoký, protože maticové operace pouze pomocí BLAS bez nutnosti kopírování matic jsou znatelně rychlejší. Při velikosti vrstev kolem 700 neuronů už se rychlosti vyrovnaly.

7 Rozdělení práce v týmu

Sólový projekt.