Kandidat 19 - Kort Tutorial

Kandidatgrupp18

20 januari 2019

1 Introduktion

Denna tutorial är gjord av kandidatgruppen med motsvarande projekt 2018 och syftet med den är att ge en kort och konkret genomgång av kedjan från datasimulering till ett färdigtränat nätverk. Det viktigaste information i denna tutorial finns i tutorial.py skriptet som man hittar i repositoryn TransferFrom2018To2019 i mappen Tutorial. Vi kan inte nog betona att detta skript är mycket begränsat, konstnadsfunktionen också är förenklad och det är inte bra kodtekniskt. Syftet med tutorial python-skriptet var att hålla allt så avskalat och överblickbart som möjligt. En stor begränsning är att nätverket endast klarar av att tränas på event där tre partiklar avfyras, då vi i vanliga fall ofta tränade nätverken på en blandning av olika multipliciteter. För mer kött på benen kring de olika momenten, se Kandidat19_Komma_igång.pdf.

Tutorialen förutsätter att man har tillgång till ggland, root, python3, tensorflow och relativt kraftig hårdvara.

2 Generera data

Python-scriptet som vi kommer använda oss av i denna tutorial klarar bara av att tränas på händelser där tre partiklar avfyras samtidigt. För att simulera dessa händelser ssh:ar man till en dator med GEANT4 installerat (e.g. planck-o eller vale) och sedan ställer man sig i mappen där man installerade ggland. Sedan går man till mappen ggland: cd land02/scripts/ggland/. Där kör man ggland med: . geant4.sh. Vi simulerar nu 1 000 000 händelser i Crystal Ball (XB) detektorn med tre partiklar avfyrade per händelse, med följande input (allt är på samma rad):

```
./land_geant4 --gun=gamma,E=0.01:10MeV,isotropic
 --gun=gamma,E=0.01:10MeV,isotropic --gun=gamma,E=0.01:10MeV,isotropic
 --events=1000000 --xb=tree-gun-edep --tree=tutorial.root --np
```

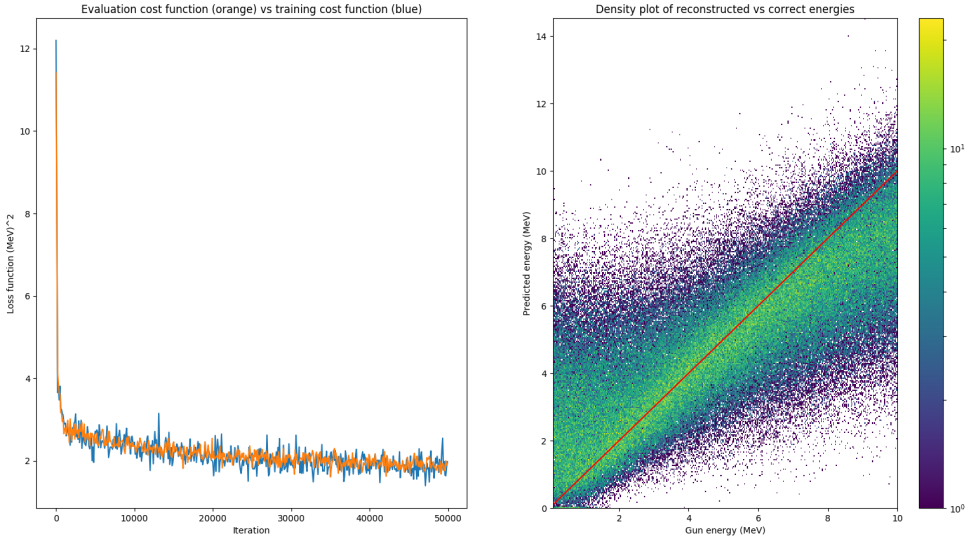
Nästa steg är att extrahera data från root-filen till textfiler. I github repositorin TransferFrom2018To2019 finns filer i mappen DataGeneration. Ladda ned dessa och placera tutorial.root i denna samma mapp som dessa filer. I denna mapp, kör

```
./gen_data_gunedep_XB.sh tutorial.root
```

Detta genererar 3 textfiler, varav två är XB_det_data.txt och XB_gun_vals.txt. Placera tutorial.py skriptet i samma mapp som dessa två textfiler. Förlagsvis läggs tid att förså hur skriptet fungerar och vad som plottas m.h.a. bl.a. kommentarerna. Kör skriptet med:

```
python3 tutorial.py XB_det_data.txt XB_gun_vals.txt
```

och du bör få upp en figur liknande denna efter ett tag:



Vi använde 50 000 träningssteg här, men det var godtyckligt valt. Under vårt projekt så var 500 000 eller mer iterationer inte ovanligt.

3 Efter tutorialen

Förutom att vara väldigt begränsad vad det gäller typen av indata och att kostnadsfunktionen inte är den som vi använde oss av för rapporten, så saknas information i denna tutorial om hur den slutgiltiga utvärderingen gjordes samt att ingen info finns angående convolution (faltning).

Ett misstag vi gjorde var att utvärdera nätverken på realistisk data mot den existerande rekonstruktionsmetoden (addback) ganska sent in i projektet, så idealt skulle denna utvärderingsprocess vara utförligt dokumenterad här. Dock så är utvärderingen inte särskilt komplicerad eller tidskrävande att göra från scratch. Dessutom så var vår lösning inte den smidigaste (vi gick bland annat över till C++ och Root för den processen). Vi har dock lagt in våra kompletta program i FinalizedPythonProgram mappen där utvärderingen mot addback finns med, med användarvänligheten för utomstående är nog inte den bästa.

Med realistisk data menas data där bakrundsbrus är simulerat och där partiklarna är boostade (i.e. de är framåtfokuserade på grund av den höga hastigheten som jonen har vid reaktionen i strålmålet). Vi tränade våra nätverk uteslutande på nätverk utan brus, för att sedan göra den slutgiltiga utvärderingen med brus. Att producera realistisk data kan självklart göras på olika sätt, men i mappen RealisticData finns beskrivit hur vi genererade denna så att det blir tydligt när ni får bättre resultat än vad vi fick. Addback-algoritmen är simpel och kan lätt inkluderas direkt efter nätverks delen i Python-programmen, så att ni får återkoppling direkt när ni tränat nätverken och slipper växla program och programspråk.