EXPERTARTIKEL

Neuronnät i fpga ge

Av Nick Ni och Adam Taylor, Xilinx

Nick Ni. I egenskap av senior produktchef planerar han produkter, marknadsför han och skapar han affärsmöjligheter inom området datorseende och SDSoC (mjukvarudefinerade systemchip). Nick Ni kom till Xilinx år 2004 efter att tidigare ha jobbat på ATI, AMD, Qualcomm och Altera. Han han en magisterexamen i datateknik från Universitetet i Toronto, Kanada.

Adam Taylor är erkänt skicklig i att utveckla tillämpningar för inbyggda system och FPGA:er – han har implementerat allt från radar och bildbehandling till säkerhetskritiska styrsystem och kryptering. Han har författat ett flertal artiklar inom FPGA och elektronikkonstruktion, inklusive 130 texter om Zync. Han äger konsultbolaget Adiuvo.

Använd ramverken så får du mycket gratis. Särskilt om du bygger neuronnät i FPGA:er.

Maskininlärning är ett av de hetaste områdena inom datorseende och inbyggda system just nu. Det spänner över flera discipliner och ligger bakom flera supertrender.

Tekniken har en framträdande roll inte bara inom datorseende, utan också inom industriell IoT (IIoT) och molnberäkningar.

För dig som inte känner till det, så implementeras maskininlärning typiskt genom att man konstruerar och tränar ett så kallat neuronnät. Sådana finns av en rad olika slag med olika namn. Neuronnät är modellerade efter hjärnbarken i det avseendet att varje neuron tar emot indata som bearbetas och skickas vidare till nästa neuron. Neuronnät består typiskt av rader av neuroner ordnande i skikt: ett inmatningsskikt, några dolda interna skikt och ett utmatningsskikt.

Neuronnät arbetar på följande sätt. Varje insignal multipliceras med en vikt, varefter de viktade insignalerna summeras. En motvikt subtraheras och till sist appliceras en överföringsfunktion. Resultatet blir insignaler till nästa skikt, och så vidare fram till utmatningsskiktet.

Neuronnät som överför utdata från skikt till skikt utan att bilda cykler kallas framåtkopplade (feed-forward), medan cykliska nät, som exempelvis Elman-nät, kallas återkopplade (recurrent).

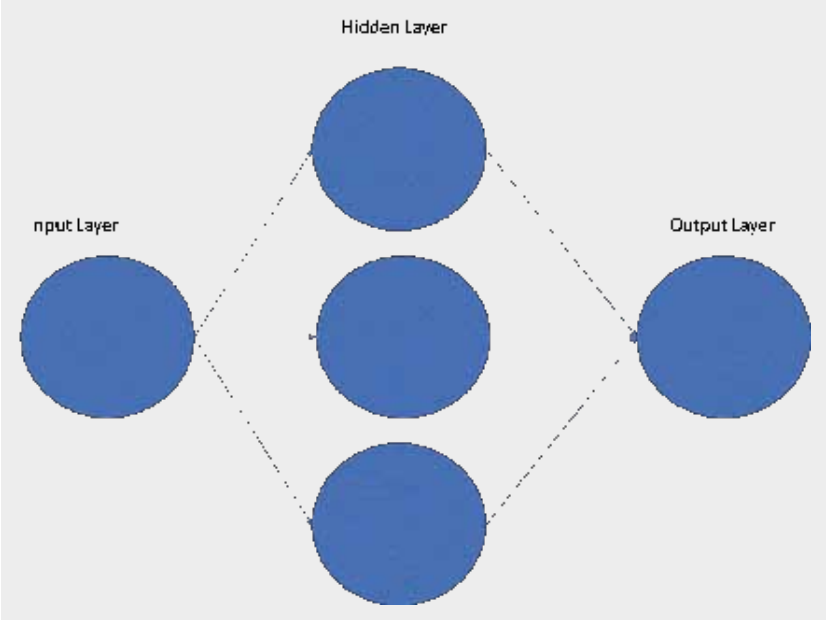
Det begrepp man kanske mest hör talas om inom maskinlärning idag är djupa neuronnät (Deep Neural Network, DNN). Sådana har många dolda skikt vilket gör att de kan lösa mer komplexa uppgifter.

Neuronnät ”tränas” genom att man justerar fram optimala värden på neuronernas vikter och motvikter. Under träningen appliceras både korrekta och inkorrekta indata, och en felfunktion används för att styra neuronnätet mot önskat beteende. Träning av ett DNN kan kräva en mycket stor datavolym innan nätet beter sig korrekt.

INOM DATORSEENDE för inbyggda system finns några av de viktigaste tillämpningarna av maskininlärning. Det pågår en utveckling där system som förut använt sig av datorseende övergår till att vara autonoma och att styras av datorseende.

Något som skiljer datorseende från enklare maskininlärningstillämpningar är att indata är tvådimensionella. Därför används en nätverksstruktur kallad Convolutional Neural Network (CNN), som kan bearbeta tvådimensionella indata.

Ett CNN är ett framåtmatande djupt neu-



Figur 1. Ett enkelt neuronnät.

```
name: "AlexNet"
layer {
  name: "data"
  type: "Input"
  top: "data"
  input_param { shape: { dim: 10 dim: 3 dim: 227 dim: 227 } }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 96
    kernel_size: 11
    xtride: 4
  }
```

Figur 2. En Prototxt-fil som definierar ett neuronnät.

r seende system

ronnät som består av en rad faltnings- och delprovtagningsskikt samt ett separat fullt kopplat nät som utför en slutlig klassificering.

I faltningsskikten delas det tvådimensionella indatat upp i överlappande mindre block som matas in i ett aktiveringsskikt och bildar en karta över aktiveringen.

Därefter sker ytterligare delprovtagning i flera steg, varefter det sista fullt kopplade skiktet appliceras. Den exakta definitionen av CNN-nätverket varierar med vilken nätverksarkitektur som implementeras, men den innehåller typiskt åtminstone följande skikt:

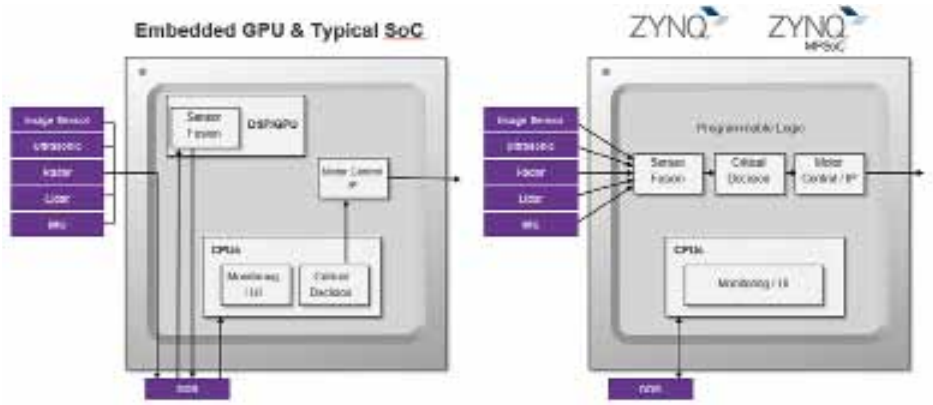
- Faltning – används för att identifiera mönster i bilden
- Rectified Linear Unit (reLU) – ett aktiveringsskikt som skapar en aktiveringskarta efter faltningen
- Max Pooling – utför delprovtagning mellan skikten
- Fullt kopplat – applicerar den slutgiltiga klassificeringen

VIKTERNA FÖR VART OCH ETT av elementen bestäms via träning, och en av fördelarna med CNN är att träningsmetoden är relativt enkel. Träning kräver dock många bildexempel både på det objekt som ska kännas igen och objekt som inte tillhör rätt kategori. På grund av de tunga beräkningskraven sker bearbetningen oftast i högpresterande datormoln.

Maskininlärning är en komplex uppgift, särskilt om man varje gång måste börja om från början med att definiera lämpliga nätverk, arkitekturer och träningsalgoritmer. För att hjälpa konstruktörer att både implementera och träna neuuronnät finns några ramverk som blivit branschstandarder, bland dem Caffe och Tensor Flow. I Caffe finns bibliotek i programspråket C ++- med bland annat modeller och förtränade nät. Dessutom finns bindningar till programspråken Python och Matlab.

Att använda ett ramverk innebär för utvecklaren att den slipper starta från början med att skriva kod för att skapa och träna nät. Caffe-användare kan dessutom återanvända varandras modeller via ett gemensamt ”zoo” befolkat av modeller som kan användas och om så önskas skräddarsys för aktuell uppgift. Nätverk och vikter definieras i en prototxt-fil, som i maskininlärningsmiljön är vad som används för att definiera en inferensmotor.

Allt oftare används lösningar baserade på programmerbar logik i inbyggnadstillämpningar för datorseende. De heterogena systemkretsarna Zynq-7000 och Zynq UltraScale+ är två plattformar som kombinerar programmerbar logik med högpresterande ARM-kärnor.



Figur 3. Fördelar med implementering i programmerbar logik.

Kombinationen gör det möjligt att bygga energieffektiva lösningar med bra svarstid som dessutom är flexibla för eventuella framtida modifieringar.

Att den slinga som klassificerar indata och bestämmer en respons har låg fördröjning är av avgörande betydelse för många tillämpningar inom datorseende, exempelvis för autonoma robotar, där svarstiden är kritisk för att undvika skador på människor och miljö.

DEN FÖRBÄTTRADE SVARSTIDEN kommer sig av att den rörledning som bearbetar kameradata är implementerad i programmerbar logik, liksom även den inferensmotor som används för maskininlärningen.

Programmerbar logik reducerar systemflaskhalsarna jämfört med traditionella lösningar. En metod som baseras på CPU och GPU kräver accesser till externt DDR-minne i varje beräkningssteg, eftersom bilderna inte kan överföras mellan funktionerna via det begränsade interna cacheminnet. Den programmerbara logiken betyder att du kan strömma data via internt RAM med buffring efter behov. Att behovet att lagra mellanliggande data i ett externt minne försvinner, reducerar inte bara fördröjningarna i bildbehandlingen utan sänker också energianvändningen och ökar determinismen eftersom det inte finns något behov att dela åtkomst med andra systemresurser.

Mjukvarustacken reVISION från Xilinx gör det enkelt att implementera både bildbehandling och maskininlärning i en heterogen systemkrets. reVISION stöder tillämpningar både inom traditionell bildbehandling och inom maskininlärning baserade på verktyget SDSoC.

Både ramverken OpenVX och Caffe stöds. För OpenVX går det att bygga en rörledning som accelererar grundläggande funktioner i programmerbar logik. Också inferensmotorn i maskininlärningssystemet kan hårdvaruoptimeras genom att implementeras i programmerbar logik.

Integrationen med Caffe i reVISION är enkel. Du specificerar en maskininlärnings- ▶

Figur 4. reVISION Stack.

motor i en prototxt-fil med tränade vikter. Ramverket hanterar resten.

Prototxtfilen konfigurerar en schemaläggare i C/C++ att accelerera neuronnätsinferenser i hårdvaruoptimerade bibliotek. Programmerbar logik används för att implementera inferensmotorn och innehåller funktioner som Conv, ReLu, Pooling, med flera.

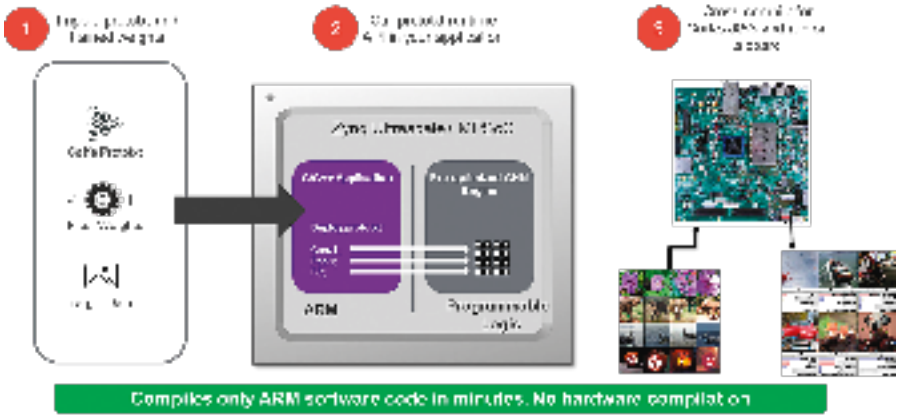
VILKEN TALREPRESENTATION som används i inferensmotorn spelar en stor roll för prestanda. Inom maskininlärning använder man allt oftare effektivare representationer med fast decimalkomma och reducerad precision, som åttabitars heltal (INT8).

Det ger inga signifikanta förluster i noggrannhet jämfört med traditionella 32-bitars flyttal (FP32) och eftersom fastpunktsaritmetik är betydligt lättare att implementera än flytpunktsaritmetik, ger förändringen också en effektivare lösning i vissa implementationer.

Fasttal passar perfekt med programmerbar logik eftersom reVISION kan jobba med INT8-representation i dedikerade DSP-block. Upp till två MAC-operationer i INT8 kan köras parallellt, om de använder samma vikter. Detta ger inte bara hög prestanda utan också minskad strömförbrukning.

Flexibiliteten i den programmerbara logiken gör det enkelt att implementera talrepresentation med ännu lägre precision, allteftersom sådan börjar användas.

reVISION-stacken ger en stor prestandavinst i faktiska tillämpningar. Ett exempel på en tillämpning med både maskininlärning och datorseende är kollisionsavvärjning för fordon. Du får en klart bättre respons om du använder en Xilinx UltraScale+ MPSoC, utvecklar tillämpningen i reVISION och utnyttjar SDSoC för att skapa acceleratorer i programmerbar logik.



Figur 5. Integrering i Caffe Flow.

SKILLNADEN ÄR STOR när man jämför svarstiden från en reVISION MPSoC med en GPU-baserad metod, när de båda implementerar Googles 22 lager djupa bildklassificerare GoogLeNet. ReVISION-implementationen upptäcker en potentiell kollision och aktiverar fordonets bromsar på 2,7 ms (med en batchstorlek på 1) medan en GPU-baserad lösning tar 49–320 ms (med stor batchstorlek) beroende på implementation. GPU-arkitekturen behöver den stora batchstorleken för att nå en rimlig genomströmning, men det sker till priset av en signifikant sämre svarstid, medan Zynq kan uppnå hög prestanda även vid ett batchstorlek på 1 med lägsta latens. Denna skillnad i reaktionstid kan vara skillnaden mellan att undvika en kollision eller inte.

Maskininlärning kommer att fortsätta att vara en viktig drivkraft i många tillämpningar, särskilt i seende robotar eller ”cobots” som de allt oftare kallas. Heterogena systemkretsar som kombinerar processorkärnor med programmerbar logik gör det möjligt att skapa mycket effektiva, responsiva och omkonfigurerbara lösningar. Med mjukvarustackar som reVISION öppnas för första gången fördelarna med programmerbar logik upp för ett bredare skikt av utvecklare, samtidigt som utvecklingstiden minskar. ■

	FP-32	FIXED-16 (INT16)	FIXED-8 (INT8)	Difference vs FP32
VGG-16	86.6%	86.6%	86.4%	(0.2%)
GoogLeNet	88.6%	88.5%	85.7%	(2.9%)
SqueezeNet	81.4%	81.4%	80.3%	(1.1%)

Figur 6. Noggrannhet i nätverk med olika viktrepresentationer.