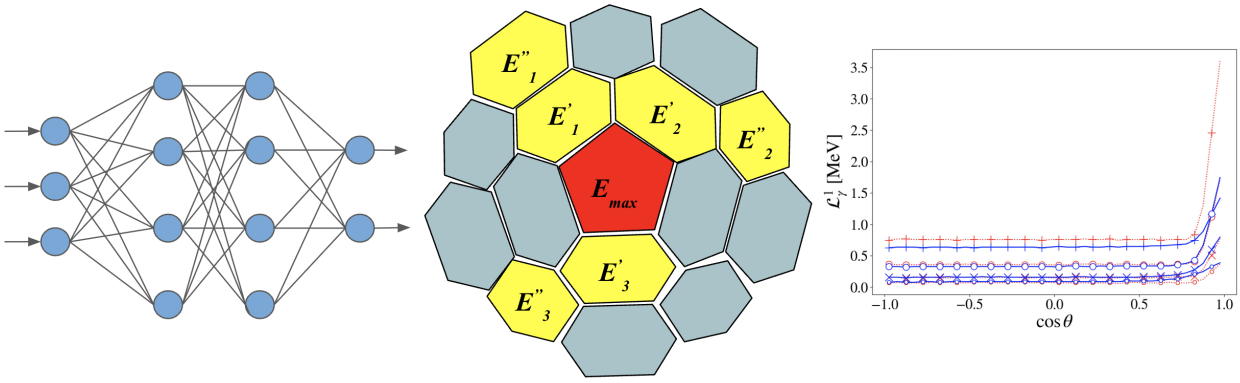
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Djupdykning med djupinlärning

Artificiella neuronnät jämförda med *addback*-algoritmen för detektorrekonstruktion av γ-fotoner

Kandidatarbete i teknisk fysik: TIFX04-22-04

BJÖRN JOHANSSON
ALFRED JUHLIN ONBECK
GABRIEL SHAFIQ AHLGREN
ISABELLA TEPP

Avdelningen för subatomär, högenergi- och plasmafysik, Institutionen för fysik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2022

Kandidatarbete 2022

Djupdykning med djupinlärning

Artificiella neuronnät jämförda med *addback*-algoritmen för detektorrekonstruktion av γ-fotoner

BJÖRN JOHANSSON
ALFRED JUHLIN ONBECK
GABRIEL SHAFIQ AHLGREN
ISABELLA TEPP

CHALMERS
UNIVERSITY OF TECHNOLOGY

Avdelningen för subatomär, högenergi- och plasmafysik
Chalmers tekniska högskola
Göteborg, Sverige 2022

Djupdykning med djupinlärning
Artificiella neuronnät jämförda med *addback*-algoritmen för detektorrekonstruktion av γ-fotoner
BJÖRN JOHANSSON ALFRED JUHLIN ONBECK GABRIEL SHAFIQ AHLGREN ISABELLA TEPP

Handledare: Håkan T Johansson, Andreas Heinz
Examinator: Jan Swenson

Kandidatarbete 2022
Avdelning för subatomär fysik, högenergi- och plasmafysik
Chalmers tekniska högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Skrivet i LaTeX
Göteborg, Sverige 2022

Abstract

To interpret detector data can be difficult and is not always unambiguous. Interactions between γ-photons and matter differ from how charged particles interact. Various scattering processes can occur and the detection of photons gets complex, especially when the aim is to characterize every individual photon, and not just the total deposited energy. Detector reconstruction is essential to determine the multiplicity, energy, and direction of the photons. This study is a continuation of the work of previous bachelor's theses with the goal of developing artificial neural networks that can surpass the conventional *addback* algorithm that is used today. Different types of neural networks are examined, where their structure and hyperparameters are varied, furthermore the residual network architecture ResNeXt has been implemented. All of the networks are trained on data that was simulated with GEANT4, ggland and ROOT. The data is meant to mimic real measurements of isotropic γ-photons with the Darmstadt-Heidelberg crystal ball.

The training of neural networks is an iterative process. For every result presented there has been multiple measurements for the same data point to prove that the values are reproducible and not just random or lucky. The networks are trained multiple times with varying parameters to find the optimum. This multiparameter problem is one of the reasons why the training of neural networks is so complex.

The measurements of performance are made such that the neural networks can be compared with *addback*. With this, a more thorough study has also been made regarding details in the differences between *addback* and neural networks reconstructions.

The developed neural networks perform better than those of previous years. For energies that they have been trained on, $0{,}1 \leq E_\gamma \leq 10\,\text{MeV}$, they also perform better than *addback*. All the three different network structures that were investigated, Fully Connected Networks (FCN), Convolutional Neural Networks (CNN) and ResNeXt, achieved a lower mean momentum error (MME) than *addback* in these intervals. For other energies, $0{,}1\,\text{MeV} - 15\,\text{MeV}$, where the networks were not trained, they did not achieve a lower MME than *addback*.

Sammandrag

Tolkning av detektordata kan vara invecklat och är inte alltid entydigt. Hur γ-fotoner interagerar med materia skiljer sig från hur laddade partiklar gör det. Spridningar av olika slag gör att detektion av fotoner blir komplex, i synnerhet att karaktärisera varje individuell foton, och inte bara den totala deponerade energin. För att urskilja multipliciteten, energin och riktningen hos γ-fotoner krävs detaljerad rekonstruktion av detektordata. Denna studie är en vidareutveckling av tidigare kandidatarbeten med målet att utveckla artificiella neuronnät som överträffar dagens konventionella rekonstruktionsalgoritm, *addback*. Olika typer av neuronnät har undersökts där deras strukturer och hyperparametrar varierats och utöver detta har också residualnätverket ResNeXt implementerats. Alla nätverk har tränats på data som är simulerad med hjälp av GEANT4, ggland och ROOT. Datan efterliknar verkliga mätningar av isotropiska γ-fotoner med Darmstadt-Heidelberg-kristallbollen.

Att träna artificiella neuronnät är en iterativ process. För alla resultat som visas har flera körningar gjorts för samma datapunkt för att verifiera att resultaten är reproducerbara och inte bara slumpmässiga eller lyckosamma. Träningen kan sedan upprepas med varierande parametrar för att optimera dem. Detta multiparameterproblem är en av anledningarna till varför träningen av artificiella neuronnät är väldigt komplex.

Prestandautvärderingen är utformad så att resultaten kan jämföras med *addback*. Med hjälp av detta har mer detaljerade studier gjorts angående om det finns skillnader i hur väl de olika metoderna rekonstruerar för olika typer av händelser.

Neuronnäten som tränats i detta arbeta överträffar de från tidigare arbeten. För de energiintervall som neuronnäten är tränade på, $0{,}1\,\mathrm{MeV} - 10\,\mathrm{MeV}$, presterar de även bättre än *addback*. Alla de tre typer av nätverk som optimerats, fullt kopplade nätverk (FCN), faltande nätverk (CNN) och ResNeXt, hade lägre fel än *addback* när de tränats på data i detta intervall. För andra intervall, $0{,}1\,\mathrm{MeV} - 15\,\mathrm{MeV}$, som nätverken inte var tränade för presterar *addback* fortfarande bättre.

Innehåll

1

Inledning

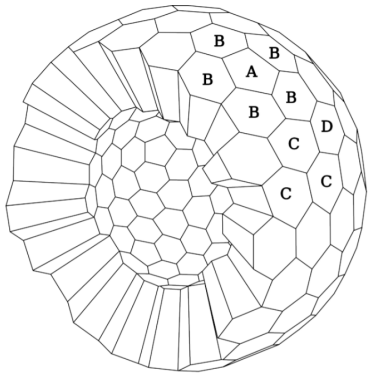
Partikelacceleratorer är bland de största och mest avancerade maskinerna byggda av människan. De är fundamentala för forskning inom subatomär fysik och för att förstå universum. Partikelacceleratorer använder sig av elektromagnetiska fält för att accelerera laddade partiklar till höga hastigheter och energier och sedan behålla dem i bana. Detektorer används för att mäta de reaktioner som inträffar då strålen kolliderar med ett strålmål. Denna rapport visar att analysen av detektordata kan förbättras med hjälp av artificiella neuronnät.

1.1 Gammafotoner vid kärnkollisioner

För att studera egenskaper av atomkärnor kan kollisioner vid relativistiska hastigheter studeras. Olika typer av detektorer används för att ta reda på vad som hänt i en kollision. En sådan detektor är Darmstadt-Heidelberg-kristallbollen, avbildad i figur 1.1. Den har 162 scintillerande NaI-kristaller som detekterar energi och riktning hos γ-fotoner.

Ett problem är att en γ-foton kan aktivera flera kristaller, främst på grund av Comptonspridning (se avsnitt 2.2), vilket gör att den exakta energin och riktningen av γ-fotonen inte mäts direkt utan måste rekonstrueras.

Rekonstruktion av energin och riktningen för γ-fotoner görs i dagsläget med *addback*-algoritmen, se avsnitt 2.4. Enligt S. Lindberg [1] ger *addback* korrekta svar för 70 % av fotoner med energi upp till 1 MeV. *Addback*-algoritmens brister ligger i att korrekt identifiera γ-fotoner med högre energier och att skilja på fotoner som aktiverar kristaller i samma område.



Figur 1.1: Tvärsnitt av Darmstadt-Heidelberg-kristallbollen med dess fyra olika kristallformer, **A, B, C** och **D.** Hela detektorn består av 162 kristaller [2].

1.2 ANN — ett komplext optimeringsproblem

En alternativ metod för att rekonstruera γ-partiklar är att använda artificiella neuronnät (ANN). ANN är en populär metod för att lösa komplexa problem inom många olika områden, bland annat för bildigenkänning och transkribering av tal till skrift. Syftet med detta projekt är att undersöka om artificiella neuronnät kan rekonstruera energi och riktning hos γ-fotoner bättre än den nuvarande metoden *addback*. Detta projekt är en vidareutveckling på tidigare arbeten som ämnat att göra detsamma [2–4]. ANN består av icke oberoende träningsbara och -styrande parametrar som behöver optimeras för γ-rekonstruktion. Meningen är att hitta en optimal uppsättning parametrar och jämföra ANN med *addback*-algoritmen. Eftersom träningen av neuronnät är stokastisk görs tre körningar för varje nätverk som presenteras för att visa reproducerbarhet om inget annat sägs.

1.3 Avgränsningar

Huvudsyftet med det här arbetet är att rekonstruera γ-fotoner med hjälp av neuronnät och jämföra dem med *addback*. Vi kommer därför inte att studera hur verklighetstrogen den simulerade träningsdatan är. Den rekonstruerade datan kommer endast att användas till att utvärdera kvalitén på neuronnäten och inga ytterligare fysikaliska analyser av utdatan kommer att göras. På grund av dess regelbundna geometri kommer Darmstadt-Heidelberg-kristallbollen vara den enda partikeldetektorn som undersöks.

1.4 Arbetet i stort

I kapitel 2 beskrivs Darmstadt-Heidelberg-kristallbollens geometri och den fysikaliska teorin bakom γ-detektion. Kapitel 3 redogör för teorin bakom neuronnät. Metodutvecklingen, träningen och optimeringen av parametrar beskrivs i kapitel 4. Resultaten presenteras i tabeller och grafer som visas i kapitel 5 där prestandan för olika neuronnät och *addback* jämförs och diskuteras. Flera olika typer av neuronnät har tränats och det går att konstatera att dessa lyckas bättre än *addback*-algoritmen som används i dagsläget. Slutligen följer kapitel 6 som ger förslag på framtida utvecklingsmöjligheter samt en sammanfattning av rapporten.

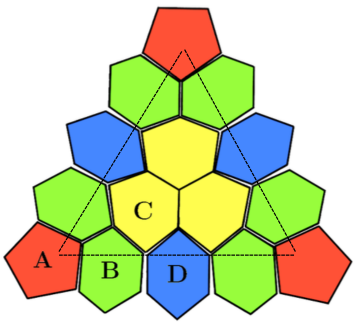
2

Detektion av γ-fotoner

Problemen som diskuteras i denna rapport angår främst maskininlärning, datagenerering och subatomär fysik. För att förstå vad som sker i simuleringarna samt problemen som uppstår krävs en djupare insikt inom subatomär fysik samt teorin om vad som händer i kristalldetektorn.

2.1 Kristallbollen som detektor

En anpassad NaI-detektor med nästintill sfärisk geometri kan ge unik information om γ-sönderfallet från exciterade atomkärnor. Just en sådan detektor, kristallbollen, har byggts i ett samarbete mellan GSI Darmstadt, Max-Planck institutet för kärnfysik och universitetet i Heidelberg [5].



Figur 2.1: En av de 20 liksidiga trianglar som utgör det sfäriska skalet av kristallbollen. Trianglarna har **A**-kristaller i varje hörn, en **BDB**-sekvens längs varje sida och tre **C**-kristaller i mitten [5].

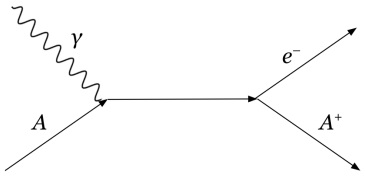
Kristallbollen består av ett 20 cm tjockt sfäriskt skal med en innerradie på 25 cm, se figur 1.1. Skalet utgörs av 162 scintillerande NaI-kristaller som har fyra olika former: en pentagon (**A**) och tre olika hexagoner (**B, C,** och **D**).

Figur 2.1 visar en tvådimensionell projektion av en sekvens från kristallbollen. Denna liksidiga triangel upprepas 20 gånger och bildar det sfäriska skalet hos kristallbollen. Varje kristall täcker en rymdvinkel på 0,076 steradianer vilket innebär att 98 % av den totala ytan är aktiv.

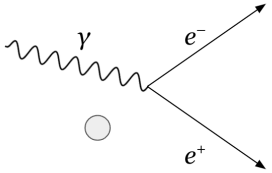
2.2 Interaktion mellan γ-fotoner och kristaller

Sättet som γ-fotoner interagerar med materia skiljer sig från hur laddade partiklar, som elektroner och protoner, gör det. Medan laddade partiklar förlorar sin energi gradvis kan fotoner deponera sin energi vid enskilda växelverkansprocesser på olika sätt beroende på vilken typ av kollision som sker. De vanligaste interaktionerna mellan γ-fotoner

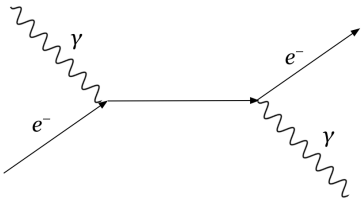
och materia är fotoelektrisk effekt, parproduktion samt Comptonspridning [6]. I de två första fallen förlorar fotonen all sin energi på en och samma gång medan vid Comptonspridning förlorar fotonen en del av sin energi. Figur 2.2 visar schematiskt hur dessa tre interaktioner sker.



(a) Fotoelektrisk effekt.



(b) Parproduktion.



(c) Comptonspridning.

Figur 2.2: Feynmandiagram av hur fotoner interagerar med materia. Fotoelektrisk effekt visas i (a), parproduktion i (b) där den gråa cirkeln representerar en atomkärna i närheten av händelsen, och slutligen Comptonspridning (c).

Fotoelektrisk effekt innebär att fotonen absorberas av en bunden elektron som därefter frigörs med den kinetiska energin $E_k = h\nu - B$, där h är Plancks konstant, ν är frekvensen hos fotonen och B är elektronernas bindningsenergi.

Parproduktion går ut på att en γ-foton omvandlas till ett elektron-positronpar. För att bevara rörelsemängden krävs det att detta händer i närheten av en atomkärna. Eftersom en elektron har en vilomassa ekvivalent med 0,511 MeV sker parproduktion endast för energier $E_\gamma > 2 \cdot 0{,}511 = 1{,}022\,\text{MeV}$.

Slutligen finns även Comptonspridning, som är den mest dominerande interaktionen för studier som denna när fotonernas energier ligger runt $1\,\text{MeV} - 20\,\text{MeV}$ och interagerar med NaI-kristaller. Comptonspridning är spridningen av fotoner på fria elektroner. I materia är dock elektronerna bundna, men när fotonernas energi är såpass mycket större än bindningsenergin hos en elektron kan elektronen lätt frigöras och approximeras som fri.

2.3 Relativistiska effekter

Acceleratoranläggningar med detektorer som kristallbollen använder jonstrålar för att studera kollisioner. Dessa strålar har hastigheter upp mot $50\,\% - 70\,\%$ av ljushastigheten, vilket motsvarar kvoten $\beta = \frac{v}{c} = 0{,}5 - 0{,}7$. När jonstrålarna interagerar med strålmålet detekteras produkterna i laboratoriets referenssystem och inte i strålens referenssystem. De intressanta och relevanta egenskaperna som forskare önskar studera beskrivs (oftast) bäst i laboratoriets referenssystem, vilket innebär att relativistiska effekter måste beaktas.

De mest framträdande effekterna är Doppler- och strålkastar-effekten (*headlight effect*). Transformationen mellan γ-fotonernas energi och vinkel i strålens referenssystem (E', θ') och laboratoriets referenssystem (E, θ) ges av

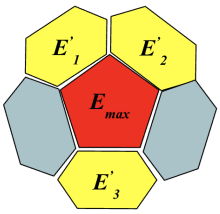
$$E' = \frac{1-\beta\cos\theta}{\sqrt{1-\beta^2}}E, \qquad \cos\theta = \frac{1+\cos\theta'}{1+\beta\cos\theta'}, \tag{2.1}$$

där β är kvoten mellan jonstrålens hastighet och ljushastigheten [7].

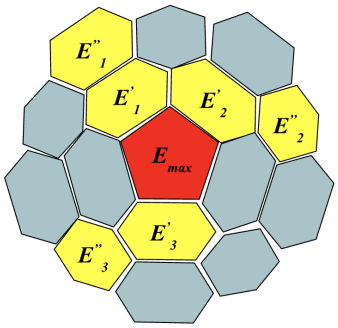
2.4 Rekonstruktion med *addback*

Som nämnts ovan kan γ-fotoner interagera med materia på olika sätt. Om fotoelektrisk effekt och parproduktion var de enda typer av interaktioner som skedde hade γ-fotonerna avgett all sin energi vid första interaktionen med en kristall, vilket hade gjort dataanalysen för varje kristalls detekterade energi väldigt enkel. Tyvärr finns även Comptonspridning som strular till det genom att fotonerna sprids och studsar mellan kristallerna. När en γ-foton träffar en kristall kan därmed en del av dess energi avges till kristallens grannar. Fotonen kan även interagera med ytterligare en kristall efter spridningen och på så sätt ge en kedja av interaktioner med olika kristaller. Spridning utanför detektorn är även möjligt, vilket innebär att ytterligare energi inte deponeras. Detta medför problem redan för rekonstruktionen av energin för en enda γ-foton.

En approximativ lösning är att använda en *addback*-algoritm som går ut på att den största energin som detekterats hos en kristall antas vara från den ursprungliga γ-fotonen. Alla energier som detekteras i närliggande kristaller antas ha orsakats av spridning från denna ursprungliga foton. Genom att summera den största energin och alla de närliggande energierna tar *addback* fram en, oftast bättre, approximation av fotonens ursprungliga energi. Därefter hanteras kvarvarande icke-nära detektioner på samma sätt för att rekonstruera fler fotoner tills alla aktiverade kristaller har behandlats [6].



(a) *Addback* med närmsta-grannar.



(b) *Addback* med två lager grannar.

Figur 2.3: Schematisk bild av *addback*. En enklare variant av algoritmen visas i (a) där endast de närmsta grannarna till den största energin E_{max} räknas med och en mer invecklad version (b), där fler grannar inkluderas.

I figur 2.3a beräknas energin för en γ-foton ut genom att summera den största energin med dess närmsta grannar enligt

$$E_{tot} = E_{max} + \sum_i E'_i, \tag{2.2}$$

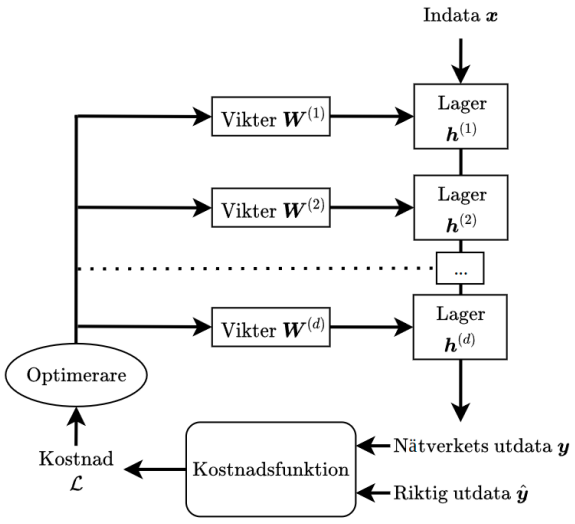
där i är direkta grannar till centralkristallen. Liknande gäller för figur 2.3b med tillägget att även räkna med andra ordningens grannar i

$$E_{tot} = E_{max} + \sum_i E'_i + \sum_j E'_j. \tag{2.3}$$

3

Artificiella neuronnät

Maskininlärning är något som blivit stort de senaste åren. Autokorrekt i våra mobiltelefoner är för de flesta en del av vardagen och på senare tid börjar även ansiktsigenkänning komma fram. Båda är exempel på maskininlärning. Genom att skapa nätverk som efterliknar beteendet hos våra hjärnor kan artificiella neuronnät hantera komplexa problem mycket effektivt. Denna del av rapporten fokuserar på att beskriva hur dessa neuronnät fungerar och hur några olika nätverk skiljer sig åt.



Figur 3.1: Ett ANN tar emot indata och ger utdata. En kostnadsfunktion jämför utdatan med verkliga (önskade) värden och beräknar med en optimeringsalgoritm ut en felkostnad, som används för att uppdatera vikterna i nätverket.

3.1 Artificiella neuronnät — inspirerad av människans hjärna

Djupinlärning är ett delfält av maskininlärning som använder modeller kallade artificiella neuronnät (ANN). Namnet neuronnät kommer från att ANN är inspirerade av neurobiologi även om ANN snarare är en matematisk modell. Idén är att låta successiva lager av noder förbättra indatans representation tills lagren ger tolkningsbar utdata. Ett klassiskt exempel är hur bilder av siffror $0-9$ successivt bryts ned till att representera sannolikheten att bilden är en viss siffra. Antalet lager är nätverkets *djup* och antalet noder per lager är dess *bredd*. Nätverket blir bättre på att representera indatan genom inlärning. Detta sker genom att exponera nätverket för träningsdata och uppdatera dess beteende med en återkopplingssignal. Figur 3.1 ger en överblick av ANN.

Rent matematiskt kan ANN betraktas som

en funktion, f bestående av ickelinjära komponenter som i detta fall tar in detektordata, $\boldsymbol{x}$, från en händelse och approximerar händelsens verkliga utfall, $\hat{\boldsymbol{y}}$. Alltså är f en approximation av $\hat{f} : \boldsymbol{x} \mapsto \hat{\boldsymbol{y}}$, där $\hat{f}$ är den verkliga funktionen som ger den riktiga utdatan $\hat{\boldsymbol{y}}$ och $\boldsymbol{x}$ är indata. I tabell 3.1 redovisas de beteckningar som används i rapporten för att beskriva hur ANN fungerar.

Tabell 3.1: Parametrar som används för att matematiskt beskriva artificiella neuronnäts uppbyggnad.

Notation	Förklaring
$\boldsymbol{x}$	Indata
$\boldsymbol{y}$	Nätverkens utdata
$\hat{\boldsymbol{y}}$	Riktig utdata
$\boldsymbol{h}^{(l)}$	Lager nummer l
$\boldsymbol{W}^{(l)}$	Viktningsmatris (lager l)
$\boldsymbol{b}^{(l)}$	Bias-vektor (lager l)
θ	Parameter i ett neuronnät, t.ex. $\boldsymbol{b}^{(l)}$
d	*Djup*, antal lager
w	*Bredd*, antal noder
g	*Grupper*, ResNeXt
$\mathcal{L}$	Kostnadsfunktion
$\mathcal{A}$	Aktiveringsfunktion
α	Inlärningshastighet

3.2 Principen för neuronnät är framåtfortplantning

Principen för ett artificiellt neuronnät är att noderna i ett lager skickar vidare data till noderna i nästkommande lager med ickelinjära komponenter. Tanken är att neuronnäten ska kunna "se" komplicerade samband som inte går att beskriva med vanlig matrismultiplikation. Framåtfortplantning (*forward propagation*) innebär att varje lager $\boldsymbol{h}^{(l)}$ består av indata från föregående lager med en viktningsmatris, $\boldsymbol{W}^{(l)}$, och en *bias*, $\boldsymbol{b}^{(l)}$. Ett neuronnät som är uppbyggt på detta sätt kallas för ett framåtriktat (*feed-forward*) neuronnät där noderna i varje lager kan skrivas som en funktion av föregående lager,

$$\boldsymbol{h}^{(l)} = \boldsymbol{W}^{(l)}\mathcal{A}(\boldsymbol{h}^{(l-1)}) + \boldsymbol{b}^{(l)}. \tag{3.1}$$

En aktiveringsfunktion $\mathcal{A}$ av utdatan från varje nod läggs till för att få ickelinjära samband mellan alla noder. Det finns många typer av ickelinjära effekter och den funktionen som används mest idag kallas likriktad linjär enhet (*Rectified Linear Unit*, ReLU) och definieras som

$$R(z) = \max(0,z), \tag{3.2}$$

dvs. en funktion som är 0 för $z < 0$ och z för $z > 0$ [8].

Med ekvation (3.1) och (3.2) kan vi låta varje nod vara kopplad till alla noder i nästkommande lager (mer om det i avsnittet om fullt kopplade nätverk, 3.8.1), låta vissa noder vara kopplade till noder flera lager fram (avsnitt 3.8.3) eller utnyttja symmetrier hos indatan (avsnitt 3.8.2) för att bygga några olika typer av neuronnät.

3.3 Utvärdera neuronnät med en kostnadsfunktion

För att kunna bestämma parametrarna i ekvation (3.1) behövs ett sätt att utvärdera hur bra ett nätverk löser sin uppgift. Ett sätt är att introducera en kostnadsfunktion som beskriver skillnaden mellan ett neuronnäts utdata och en önskad utdata. Alltså givet en indata, t.ex. en bild på en sengångare, vill vi kanske att ett neuronnät ska kunna urskilja vilken djurart som är på bilden, eller om bilden faktiskt föreställer en sengångare. I stort sett finns det två typer av problem: regression och klassificering. Klassifi-

cering innebär att nätverken ska kunna placera indata i grupper, vilken djurart bilden visar, medan regression innebär att nätverken ger ett kontinuerligt värde, till exempel sannolikheten att bilden visar en sengångare. Att rekonstruera energin hos γ-fotoner är ett regressionsproblem och därför kommer endast kostnadsfunktioner för regressionsproblem att beskrivas.

En av de vanligaste kostnadsfunktionerna för regressionsproblem är det genomsnittliga kvadratfelet,

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i||^2, \qquad (3.3)$$

där n är antalet noder i det sista lagret, $\boldsymbol{y}_i$ är värdet i den i:te noden och $\hat{\boldsymbol{y}}_i$ är det motsvarande sökta värdet [8]. De funktioner som vi har använt oss av beskrivs i avsnitt 4.5.

Ett annat regressionsproblem är de av binär klassificering. Binär klassificering är problem där observationen kan delas in i två kategorier, till exempel: är detta en sengångare eller en gorilla? Binär korsentropi jämför de förutspådda sannolikheterna till någon av kategorierna, antingen 1 eller 0 [9]. Ett vanligt val för dessa typer av problem är binär korsentropi, H, som är definierad som

$$\begin{aligned} H(\beta,\hat{\beta}) = & -\hat{\beta}\log(\beta) + \\ & + (\hat{\beta}-1)\log(1-\beta), \end{aligned} \qquad (3.4)$$

där $\hat{\beta} \in \{0,1\}$ är det korrekta värdet och $0 < \beta < 1$ är nätverkets resultat.

3.4 Bakåtfortplantning för att träna neuronnät

När vi har hittat en funktion som beskriver felet behövs ett sätt att ändra nätverkets parametrar ($\boldsymbol{W}^{(l)}$ resp. $\boldsymbol{b}$) för att minimera felet. Denna process kallas för träning och är uppdelad i kontrollerad (*supervised*) och okontrollerad (*unsupervised*) träning. [8]

Okontrollerad träning innebär träning på en datamängd utan att ha tillgång till information om det önskade resultatet. Nätverk som tränas okontrollerat är därför oftast till för att identifiera mönster, trender eller gruppera datapunkter.

Kontrollerad träning innebär att träningsdatan på något sätt är kopplad till den önskade utdatan, det går att bestämma felet med till exempel funktionerna i ekvation (3.3) och (3.4). I detta arbete används just kontrollerad träning eftersom vår datamängd är simulerad, mer om det i avsnitt 4.1. Idén är alltså att ge nätverk indata för att se hur stort felet blir och sedan justera deras parametrar utefter felet. För att träna neuronnät som bygger på framtåfortplantning används oftast bakåtfortplantning (*backward propagation*) [10]. Bakåtfortplantning beräknar gradienten av kostnadsfunktionen med avseende på nätverkets parametrar. Genom att gå bakåt i nätverket kan vi räkna ut gradienten för tidigare lager med kedjeregeln och parametrarna uppdateras enligt

$$\theta^{t+1} = \theta^t - \alpha\frac{\partial\mathcal{L}}{\partial\theta}, \qquad (3.5)$$

där θ^t är någon parameter hos nätverket efter t iterationer och α är inlärningshastigheten. Dessa beräkningar är komplexa och kan bli mycket stora. För att underlätta träningen kan träningsdatan delas upp i delar och gradienten beräknas på slumpmässiga mindre delmängder av datan. Detta kallas *Stochastic Gradient Descent* (SGD) [8] och optimeraren `Adam` [11], som används

i detta projekt, bygger huvudsakligen på SGD.

3.5 Varierad steglängd för effektiv träning

Förutom att använda sig av olika optimerare för att effektivisera träningen finns ett antal träningsparametrar som avgör hur bra ett neuronnät kan bli och hur snabbt det går att träna. Att ha ett färdig tränat nätverk innebär att ha hittat en uppsättning parametrar som minimerar kostnadsfunktionen. Inlärningshastigheten, eller steglängden, avgör hur bra träningen kan bli. Ett för litet steg α innebär att ekvation (3.5) är oförändrad medan ett för stort steg gör att ekvationen inte konvergerar mot ett minimum. Att börja med en större steglängd för att sedan avta under träningen har visat sig ge bättre resultat än konstant inlärningshastighet [12]. En av de vanliga metoderna är exponentiellt avtagande inlärningshastighet (*exponential decay rate*)

$$\alpha_r(s) = \alpha_0 \cdot (\mathrm{d}r)^{s/f_s}, \tag{3.6}$$

där α_0 är den initiala inlärningshastigheten, s är antal iterationer, dr är hastighetens avtagande (*decay rate*), och f_s är en parameter (*decay step*) som påverkar förändringen av hastighet [12].

3.6 Mängden träningsdata påverkar träningen

Utöver träningsparametrar och optimerare spelar också träningsdatan en stor roll. Målet är att konstruera ett neuronnät som kan ge ett önskat resultat på data som den inte tidigare sett. Den totala datamängden delas oftast upp i tre delar: träningsdata, evalueringsdata och testdata [8]. Träningsdatan presenteras för nätverket flera gånger i så kallade *epoker*. Efter alla epoker används evalueringsdatan för att utvärdera hur bra nätverket presterar. Detta resultat används i sin tur för att justera hyperparametrarna, alltså de parametrar som bestämmer strukturen hos nätverken som till exempel bredd och djup, men även träningsstyrande parametrar som inlärningshastighet.

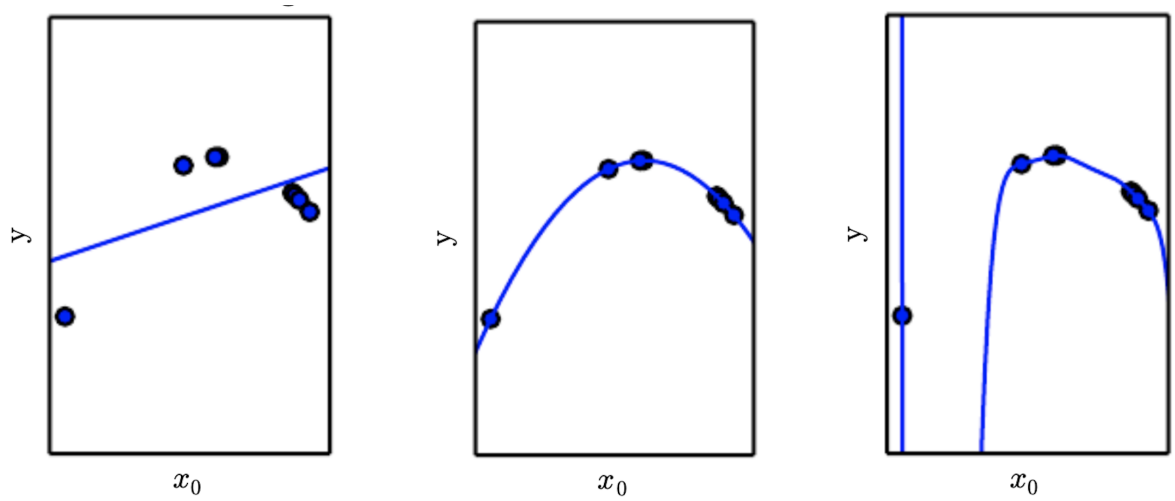
Om nätverken lyckas rekonstruera utdata från träningsdatan väl betyder det inte att nätverken kommer vara lika framgångsrika på testdatan. Det kan ske både över- och underträning (*overfitting* och *underfitting*). Underträning skulle kunna liknas med att plugga för lite inför ett prov medan överträning i så fall blir att hela kursboken har memorerats och endast svar exakt formulerade som i kursboken kan besvaras. Figur 3.2 demonstrerar dessa fenomen. Att öka mängden träningsdata är ett sätt att få nätverken att bli mer generella [8] och alltså minska både under- och överträning.

3.7 Regularisering motverkar överträning

Att öka mängden träningsdata är inte alltid möjligt. Det förutsätter att det finns mer data tillgänglig. Nätverk med många parametrar har större utrymme att hitta ickelinjära samband mellan in- och utdata. Ett problem med många parametrar är dock att nätverken tenderar att övertränas och därför blir sämre på att tolka data som nätverken inte tränats på. Därför har det utvecklats flera metoder för att motverka överträning som kallas regularisering. Nedan beskrivs avhopp (*dropout*), L1- och L2-regularisering som några regulariseringsmetoder.

3.7.1 Avhopp

Avhopp (*dropout*) är en populär regulariseringsmetod. Metoden går ut på att en andel, p, av utdatan från ett lager, $\boldsymbol{h}^{(l-1)}$,



(a) Underträning **(b)** Lagom tränat **(c)** Överträning

Figur 3.2: Demonstration av hur över- och underträning tar form när det gäller tillämpning av funktioner på datapunkter [8]. Bild (a) visar en undertränad funktion, bild (c) visar en övertränad och mittenbilden (b) är det optimala fallet.

slumpmässigt "tappas". Implementationen är i form av ett lager som utför operationen

$$\tilde{\boldsymbol{h}}^{(l-1)} = \boldsymbol{h}^{(l-1)} \cdot \boldsymbol{r}(p) \tag{3.7}$$

innan $\tilde{\boldsymbol{h}}^{(l-1)}$ skickas till nästa lager. Vektorn $\boldsymbol{r}(p)$ är Bernoullifördelad med parameter p som på engelska kallas *dropout rate* [13].

3.7.2 L1- och L2-regularisering

L1- och L2-regularisering används för att reducera komplexiteten av ett nätverk. L2-regularisering går också ofta under namnet *weight decay* på engelska. Metoderna går ut på att ändra nätverkets kostnadsfunktion med

$$\mathcal{L}_{L1} = \kappa_1 \sum_{ij} |\, W_{ij}^{(l)}|\, \tag{3.8}$$

respektive

$$\mathcal{L}_{L2} = \kappa_2 \sum_{ij} |\, W_{ij}^{(l)}|^2 \tag{3.9}$$

för utvalda lager. Värdena κ_1 och κ_2 kallas lagrets L1- respektive L2-värde. Det är även möjligt att addera båda termer samtidigt [14].

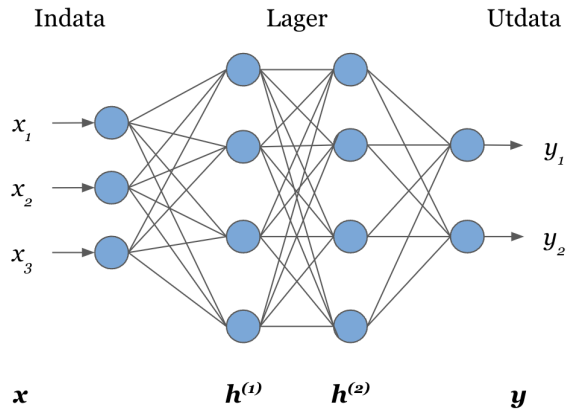
3.8 Tre olika typer av neuronnät

Med fler träningsbara och -styrande parametrar kommer en större utmaning, att optimera ett neuronnät. Det finns olika arkitekturer för neuronnät och i denna rapport beskrivs tre stycken modeller: fullt kopplade neuronnät, faltande nätverk och ResNeXt (en variant av residuala neuronnät). Varje modell har sina för- och nackdelar men alla tre modeller har lyckats rekonstruera γ-fotoner bättre än *addback*, vilket beskrivs i avsnitt 5.1.

3.8.1 Fullt kopplade neuronnät

Ett fullt kopplat neuronnät (*Fully Connected Network*, FCN) är ett framåtriktat nätverk där alla neuroner i ett lager är kopplade till alla neuroner i nästa. Det första lagret är indatan, $\boldsymbol{x}$, som går genom ett antal lager,

$\boldsymbol{h}^{(l)}$, för att nå sista lagret som är utdatan, $\boldsymbol{y}$. Indexet l i $\boldsymbol{h}^{(l)}$ anger vilket lager som berörs. Figur 3.3 visar en skiss över ett enkelt fullt kopplat neuronnät.



Figur 3.3: En illustration av strukturen hos ett FCN med två interna lager, $\boldsymbol{h}^{(l)}$. Indatan är $\boldsymbol{x}$, varje cirkel är en artificiell neuron och $\boldsymbol{y}$ är utdata från nätverken.

En av fördelarna med FCN är att deras struktur inte antar något om indatan på förhand till skillnad från andra typer av neuronnät som genom sin design kan förvänta sig exempelvis en tvådimensionell topologi som hos grafisk indata [15].

3.8.2 Faltande neuronnät

Faltande neuronnät (*Convolutional Neural Networks*, CNN) är specialiserade för att hantera data som har en känd topologi likt ett rutnät [8], exempelvis data av tidsserier som ett endimensionellt rutnät, eller bilddata som ett tvådimensionella rutnät. CNN får sitt namn från den matematiska operationen *faltning* (*convolution* på engelska), som sker i ett eller flera av nätverkens lager. CNN är, likt FCN, framåtpropagerande nätverk men till skillnad från FCN består CNN av en mindre kärna (*kernel*) som faltas med indatan.

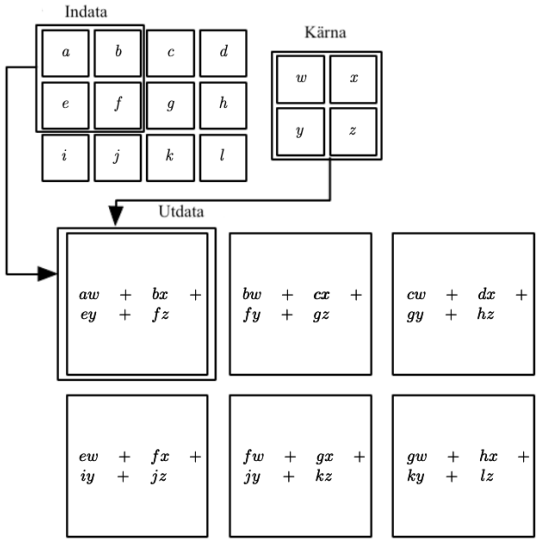
Faltning mellan två funktioner brukar betecknas med en asterisk, $s(t) = (x * w)(t)$, och definieras som

$$s(t) = \int_{-\infty}^{\infty} x(r)w(t-r)dr. \tag{3.10}$$

Om funktionen (x i detta fall) består av diskreta punkter snarare än att vara kontinuerlig fås

$$s(t) = \sum_{r=-\infty}^{\infty} x(r)w(t-r). \tag{3.11}$$

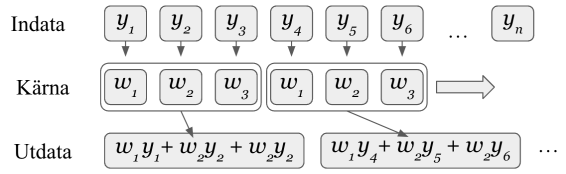
I faltande nätverk är funktionen x indatan, funktionen w kallas kärnan och utdatan $s(t)$ kallas funktionskarta (*feature map*). Inom maskininlärning är alla dessa delar flerdimensionella vektorer, dvs. tensorer. Figur 3.4 illustrerar av hur ett tvådimensionellt faltande lager i ett nätverk fungerar.



Figur 3.4: Exempel på hur ett lager med tvådimensionell indata kan behandlas av faltande nätverk [8]. Kärnan ges i form av en 2×2-tensor och utdatan är faltningen av kärnan och en lika stor del av indatan.

I detta arbete kommer endimensionella faltande nätverk användas. Som visats ovan så faltas indatan med kärnan och i ett 1D-nätverk ser det ut som i figur 3.5.

Kärnan har oftast mindre dimension än indatan, som i figurerna 3.4 och 3.5, och behöver därför upprepas över hela indatans storlek. I fallet med det endimensionella nätverket tar kärnan här dessutom steg om tre, vilket också är kärnans storlek [4].



Figur 3.5: I detta exempel verkar kärnan inte på varje möjlig del av indatan, utan med en steglängd (*stride*) [4], här tre.

Generellt består varje lager i ett CNN av tre steg: i första steget sker faltningen (affin transform), sedan appliceras en ickelinjär aktiveringsfunktion (till exempel ReLU, se avsnitt 3.2) och sist kan en *pooling*-funktion användas. Meningen med *pooling* är i stort sett att sammanfatta utdatan från faltningslagret till färre noder. Genom att göra detta minskar mängden data som ska föras vidare till nästa lager och större områden kan beskrivas med begränsade kärnor. Två vanliga *pooling*-funktioner är max-*pooling* och medelvärdes-*pooling*. De går ut på att ta maxvärdet respektive medelvärdet av noderna från ett område i lagret.

3.8.3 ResNeXt

ResNeXt-modellen bygger på residualnätverkens struktur där ett eller flera lager kan hoppas över. Rent matematiskt så är det att indata-tensorn till ett lager kan vara summan av föregående lagrets utdata och ett annat lagers utdata tidigare i nätverket. Det som karaktäriserar ett ResNeXt-block är att datan delas upp i flera delar och går vidare till parallella grupper, mindre delnätverk för att sedan konkateneras och sedan möjligtvis till fler ResNeXt-block.

4

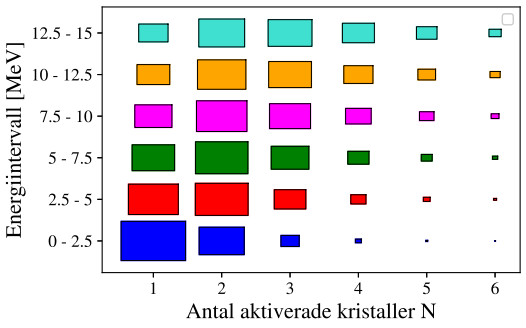
Metodutveckling

Detta kapitel redogör för de centrala aspekterna av den metodutveckling som gjorts. Det huvudsakliga arbetet i detta projekt ligger i att undersöka och utveckla olika nätverk med målet att prestera bättre än *addback*. Neuronnätens alla hyperparametrar påverkar deras förmåga att rekonstruera γ-fotoner. Optimeringsproblemet är alltså stort och multidimensionellt, där en parameters optimala värde är generellt sett beroende på resten av parametrarna och därför måste träningen upprepas flera gånger.

4.1 Datagenerering för träning och evaluering

För att träna ANN för γ-rekonstruktion krävs stora mängder data från Darmstadt-Heidelberg-kristalldetektorn. Istället för att använda experimentell data där den korrekta rekonstruktionen bara är delvis känd, genereras data med Monte Carlo-baserade datasimuleringar via det `C++`-baserade funktions-biblioteket `GEANT4` (med *wrapper*-programmet `ggland`). `GEANT4` simulerar data genom att skapa *händelser* där ett antal γ-fotoner avfyras från kristallbollens centrum isotropt med likformigt fördelad energi. Fotonerna interagerar sedan med detektorns kristaller enligt figur 2.2. För att läsa filerna som skapas används `ROOT` som är ett `C++`-baserat paket för dataanalys utvecklat av CERN [16].

Datasimuleringarna ger två typer av data för varje händelse. Först indata till nätverken och *addback* i form av en lista med 162 flyttal där varje element motsvarar energideponeringen i kristallerna $1 - 162$. Detta motsvarar de direkta mätvärdena från en experimentell mätning av en kärnreaktion. Figur 4.1 ger en överblick över hur många kristaller som aktiveras beroende på energin hos de ursprungliga fotonerna.



Figur 4.1: Lådhistogram av antalet aktiverade kristaller N från ca $1{,}6 \cdot 10^6$ avfyrade γ-fotoner. Majoriteten av alla fotoner deponerar sin energi i en eller två kristaller, $\mu_N = 1{,}96$ och $\sigma_N^2 = 0{,}95$.

Simuleringarna ger också de genererade fotonernas energi och riktning som etiketter för varje händelse bestående av en lista på formen

$$\hat{P} = (\hat{\boldsymbol{p}}_\mathbf{1}, \hat{\boldsymbol{p}}_\mathbf{2}, ..., \hat{\boldsymbol{p}}_\boldsymbol{m}) \in \mathbb{R}^{3m}, \tag{4.1}$$

där $\hat{\boldsymbol{p}}_\boldsymbol{i} = (\hat{p}_{ix}, \hat{p}_{iy}, \hat{p}_{iz})$ är rörelsemängden för en γ-foton och m är det maximala antalet γ-fotoner i en händelse. Detta innebär att en händelse med två γ-fotoner men $m = 3$ representeras av

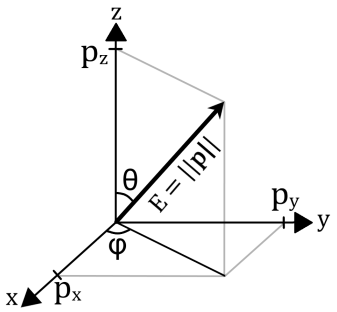
$$\hat{P} = (\mathbf{0}, \hat{\boldsymbol{p}}_\mathbf{2}, \hat{\boldsymbol{p}}_\mathbf{3}) \in \mathbb{R}^9, \tag{4.2}$$

där $\mathbf{0} = (0, 0, 0)$ är nollvektorn. Etiketterna används för jämförelse med utdatan från nätverken och *addback*.

Fotonerna kan också ekvivalent beskrivas i ett sfäriskt koordinatsystem. Eftersom fotoner är masslösa partiklar ger relativitetsteori att $E = pc$. Genom att också byta till ett enhetssystem där $c = 1$ fås att $E_i = ||\boldsymbol{p}_\boldsymbol{i}||$. En γ-fotons rörelsemängd från både ekvation (4.1) och (4.4) kan då uttryckas som

$$\boldsymbol{p} = (p_x, p_y, p_z)_{xyz} = (E, \theta, \phi)_{r\theta\phi}. \tag{4.3}$$

Figur 4.2 illustrerar koordinatbytet. Medan de tidigaste arbetena främst arbetade i sfäriska koordinater [3, 4] valde det förgående att arbeta i kartesiska för att bli av med ett antal artefakter [2].

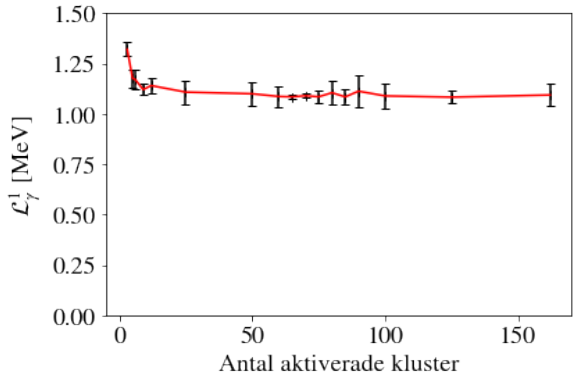


Figur 4.2: Transformation mellan sfäriska och kartesiska koordinater för en γ-fotons rörelsemängd.

Skript från tidigare arbeten används för att generera flera typer av data samtidigt och sedan sammanfoga och komprimera dessa till en fil som kan läsas av nätverken.

Den simulerade datan används för att träna, utvärdera och optimera nätverken i detta arbete. Datan används också till rekonstruktion och utvärdering med *addback*-metoden för jämförelse. Vi väljer att träna och optimera nätverken för händelser med upp till tre γ-fotoner ($m = 3$) där varje fotons energi är likformigt fördelad i intervallet 0,1−10 MeV och där vi lägger till 5 % helt tomma händelser. Vi väljer också att enbart inkludera händelser där 90 % av energin är deponerad. Nätverken och *addback* utvärderas på data med varierande energier (se kapitel 5).

En standardinställning i `ggland` sparar endast de 10 största energideponeringarna från γ-partiklarna. Eftersom en foton kan deponera delar av sin energi i stegvisa interaktioner och flera partiklar kan simuleras samtidigt förloras ofta information om detta tak inte ökas. Tidigare års arbeten [2–4] verkar inte tagit hänsyn till detta. Detta undersöktes genom att simulera flera datamängder där denna parameter varierades och sedan träna nätverk på den datan. Det visade sig då att ett alltför lågt värde inte gav bra resultat, medan det förbättrades upp till 50 kluster och därefter planade ut. Figur 4.3 visar detta.



Figur 4.3: $\mathcal{L}_\gamma^1$ (medelrörelsemängdsfelet per γ-foton) som en funktion av antalet aktiverade kluster. Fem mätningar har gjorts för varje datapunkt på ett FCN med $d = 7$ och $w = 50$. Standardavvikelsen visas med felstaplar.

4.2 Bygga nätverk

De artificiella neuronnäten byggs och modifieras i Googles maskininlärningsplattform `Tensorflow` [17] med `Python`-API:et `Keras` [18]. Med hjälp av `Keras` byggs nätverken upp genom en abstrakt `Model`-klass där deras arkitektur kan bestämmas, till exempel fullt kopplade (FCN) eller faltande nätverk (CNN).

Nätverken ger utdata på formen

$$P = (\boldsymbol{p_1}, \boldsymbol{p_2}, ..., \boldsymbol{p_m}), \tag{4.4}$$

liknande etiketterna i (4.1). Skillnaden är att nätverken aldrig rekonstruerar tomma fotoner ($\boldsymbol{p} = \boldsymbol{0}$). Istället definieras tomma fotoner med hjälp av ett tröskelvärde

$$||\boldsymbol{p}|| < \epsilon \implies \boldsymbol{p} = \boldsymbol{0}, \tag{4.5}$$

där $\epsilon = 0{,}1\,\text{MeV}$ valts eftersom det är den lägsta möjliga energin av en foton under träningen.

4.3 Optimeringsproblemet med hyperparametrar

För varje nätverk finns det ett flerdimensionellt optimeringsproblem som behövs lösas. Genom att köra svepningar i en eller fler dimensioner kan lokala minima hittas för att få fram den lokalt bästa varianten av alla nätverk. Att träna ANN är stokastiskt och nya värden fås varje gång man gör en ny träning. För att visa att resultaten som erhålls är reproducerbara upprepas mätningarna för varje punkt. För alla grafer i denna rapport har tre körningar gjorts för varje datapunkt, om inget annat påstås.

4.4 Prestandautvärdering

För att enkelt utvärdera och jämföra olika nätverk används medelrörelsemängdsfelet (*mean absolute error*), $\mathcal{L}_\gamma^1$, framtagen av tidigare grupp [2]. Den är definierad som

$$\mathcal{L}_\gamma^1 = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i|| \tag{4.6}$$

där N är det totala antalet γ-fotoner i alla händelser.

Anledningen till att $\mathcal{L}_\gamma^1$ används är främst för att underlätta jämförelser med tidigare arbeten. Ett alternativt mått baserat på kvadratroten ur medelkvadratfelet (*root mean square error*) är

$$\mathcal{L}_\gamma^2 = \sqrt{\frac{\sum_{i=1}^{N} ||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i||^2}{N}}. \tag{4.7}$$

Detta innebär också att vi använder ett oviktat skalärmått. $\mathcal{L}_\gamma^2$ medför att fel mellan en verklig och rekonstruerad foton inte viktas upp om $||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i|| > 1$ eller viktas ned om $||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i|| < 1$. Ett problem med $\mathcal{L}_\gamma^1$ och $\mathcal{L}_\gamma^1$ är att nätverken och addback får ett mindre fel om de "hittar på" många γ-fotoner med låg energi. För att kunna mäta

felet kvalitativt hos nätverken och *addback* mäts kvoten av $\mathcal{L}^1$ och det totala antalet händelser,

$$\mathcal{L}_e^1 = \frac{1}{H}\sum_{i=1}^{N} ||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i||, \tag{4.8}$$

där H är antalet händelser. På så sätt minskar inte felet om nätverken eller addback rekonstruerar för många fotoner med låga energier utan endast ökar.

Prestandautvärderingen sker även grafiskt i "lasersvärdsfigurer" (se figur 5.1 i kapitel 5) som jämför de egentliga värdena $\hat{E}, \hat{\theta}, \hat{\phi}$ med de rekonstruerade värdena E, θ, ϕ. Detta görs genom att visa den rekonstruerade energin mot den korrekta i en graf, samma görs för vinklarna θ och ϕ.

4.5 Kostnadsfunktion

Kostnadsfunktioner är en viktig del i träningen av artificiella neuronnät. Utmaningen är att hitta en kostnadsfunktion som passar just detta problem. De första åren använde en icke-relativistisk version av det genomsnittliga kvadratfelet av E, θ och ϕ, vilket dock producerade artefakter, främst i θ och ϕ [3, 4]. Den senaste gruppen [2] använde det genomsnittliga kvadratfelet på rörelsemängden hos den korrekta och rekonstruerade fotonen, definierat som

$$\mathcal{L}^2 = \sum_{i=0}^{m} ||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i||^2, \tag{4.9}$$

vilket åtgärdade artefakterna i de sfäriska koordinaternas vinklar [2].

Vi har dock noterat att en kostnadsfunktion definierad av

$$\mathcal{L}^1 = \sum_{i=0}^{m} ||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i|| \tag{4.10}$$

ger bättre slutlig prestanda än användning av $\mathcal{L}^2$. $\mathcal{L}^1$ grundas, likt prestandamåttet $\mathcal{L}_\gamma^1$, på absolutfel vilket innebär att nätverken tränas på samma mått de evalueras på. Detta innebär också att nätverken inte straffas hårdare för stora fel jämfört med små vilket kan tänkas öka generaliseringsförmågan. Vi har valt att huvudsakligen optimera nätverken efter träning med $\mathcal{L}^1$.

Ytterligare kostnadsfunktioner som använts har varit Huberförlust, $\mathcal{L}_\text{H}$, och medellogfelet, $\mathcal{L}_\text{log}$. Huberförlust är en kombination av $\mathcal{L}^2$ och $\mathcal{L}^1$ definierat som

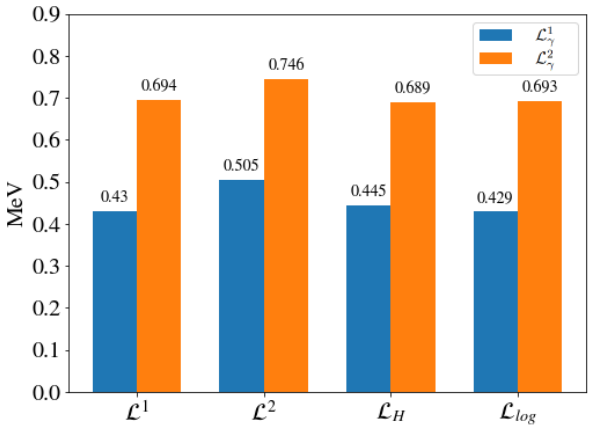
$$\mathcal{L}_\text{H} = \sum_{i=0}^{m} \begin{cases} \frac{1}{2}||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i||^2 \text{ om } ||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i|| < \delta, \\ \delta \cdot (||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i|| - \frac{1}{2}\delta) \text{ annars,} \end{cases} \tag{4.11}$$

där $\delta = 1$.

Syftet är att belöna små fel och inte straffa stora fel för hårt. $\mathcal{L}_\text{log}$ är en logaritmerad version av $\mathcal{L}^2$ definierad som

$$\mathcal{L}_\text{log} = \sum_{i=0}^{m} \log(||\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i|| + 1). \tag{4.12}$$

Denna kostnadsfunktion straffar nätverket proportionellt sett mer för små fel än för stora och testas för att göra nätverken bättre på att rekonstruera fotoner med låga energier.



Figur 4.4: $\mathcal{L}_\gamma^1$ och $\mathcal{L}_\gamma^2$ för `FCN-d6-w1200`-nätverket som introduceras i avsnitt 4.7, tränat med de olika kostnadsfunktionerna.

4.5.1 Permutationsförlust

När flera γ-fotoner detekterats behöver kostnadsfunktionen matcha rätt rörelsemängd $\hat{\boldsymbol{p}}_i$ med de korrekta rörelsemängderna $\boldsymbol{p}_i$. Denna parning görs över alla $m!$ möjliga permutationer eftersom det inte är meningsfullt att kräva att nätverken ska "gissa" i vilken ordning fotonerna etiketterats. Tidigare år har konstruerat en metod som byggts vidare på [2].

4.6 Addback

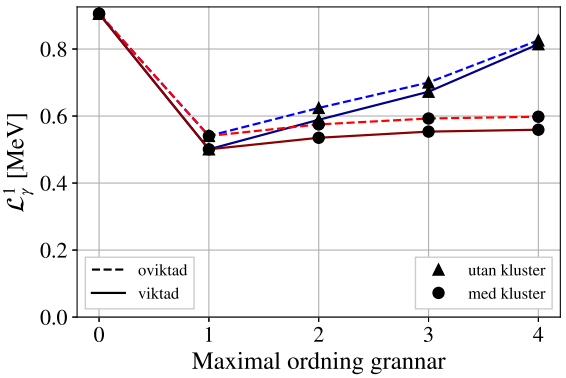
För att skapa en baslinje för prestandajämförelser av och mellan nätverken implementerades *addback*-algoritmen i `Python`. Implementationen hanterar indata och ger utdata på samma form som nätverken vilket underlättar i jämförelserna mellan metoderna. Med hjälp av prestandamåttet $\mathcal{L}_e^1$ presenterat i avsnitt 4.4 kan också nätverken jämföras med *addback*. Eftersom nätverken har ett fixt maximalt antal rekonstruerade fotoner gäller att $\mathcal{L}_e^1 = m \cdot \mathcal{L}_\gamma^1$ för nätverken.

Implementationen gör det möjligt att modifiera vilka grannar som inkluderas och använda två olika riktningsberäkningsmetoder. Om index i noterar en kristall med lokal maximal energideposition och G är mängden av alla kristallindex associerade med träffen i, kan G varieras till att vara alla n:te ordningens grannar till i eller n:te ordningens sammanhängande grannar. Med sammanhängande menas kristaller som är en del av ett kopplat kluster med energideponeringar. Den alternativa riktningsberäkningen viktar varje kristalls positionsvektor med dess energideposition. Den rekonstruerade fotonens θ och ϕ tas sedan från vektorn

$$\boldsymbol{R} = E_i \hat{r}_i + \sum_{j \in G} E_j \hat{r}_j. \tag{4.13}$$

Observera att energideponeringen inte ges av $||\boldsymbol{R}||$ utan av dess totalsumma.

Figur 4.5 visar $\mathcal{L}_e^1$ som funktion av det maximala antalet inkluderade grannar. Det syns att alla metoder är lika då enbart den första träffen inkluderas. Att räkna sammanhängande kluster spelar endast roll då två eller fler lager grannar inkluderas. Det lägsta värdet var $\mathcal{L}_e^1 = 1{,}502\,\text{MeV}$ ($m = 3$ ger då $\mathcal{L}_\gamma^1 \approx 0{,}5$) för både viktad och oviktade kluster då ett lager grannar inkluderas. För att jämföra nätverken med *addback* väljs därför inställningen med ett lager grannar och energiviktade riktningar.

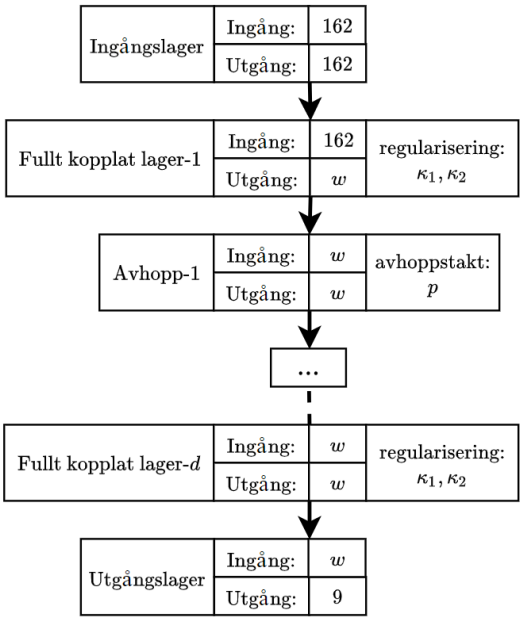


Figur 4.5: $\mathcal{L}_\gamma^1$ för *addback* som funktion av maximalt antal lager av grannar. Värden är beräknade med viktad och oviktad riktning och där både sammanhängande och osammanhängande kluster inkluderats. Det lägsta värdet var $\mathcal{L}_\gamma^1 \approx 0{,}5\,\text{MeV}$ ($\mathcal{L}_e^1 = 1{,}502\,\text{MeV}$) för maximalt ett lager grannar.

4.7 Fullt kopplade neuronnät

Tidigare arbeten har haft mest framgång med de fullt kopplade nätverken [2–4]. Som nämnts i avsnitt 3.8.1 är strukturen av ett FCN fullständigt definierat av ett antal lager, d, kallat nätverkets djup, och antalet

noder i respektive lager, w, kallade nätverkets bredd, där alla noder i ett lager är kopplade till noderna i nästa. P. Halldestam m.fl. [2] fann också att varierande bredd i olika lager inte påverkade prestandan nämnvärt och därför är nätverkens bredd densamma för varje lager i våra nätverk.

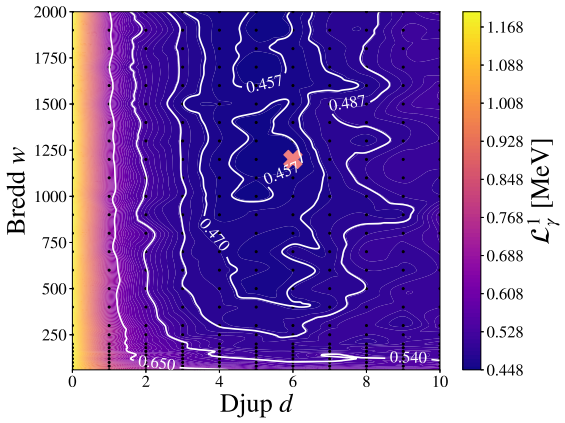


Figur 4.6: Strukturen för ett FCN med avhoppslager med djup $d \geq 1$ och godtycklig bredd w. Nätverket har möjlighet att använda L1/L2 regularisering, där κ_1, κ_2 är L1- respektive L2-värden, eller avhopp, där p är avhoppstakten. Nätverken är anpassat efter indata från kristallbollen och utdata i form av rörelsemängden för tre γ-fotoner. Antalet parametrar i respektive lager anges som Π.

I figur 4.6 visas strukturen för det fullt kopplade nätverket som undersöktes i detta arbete. Det totala antalet parametrar ges av

$$\Pi = dw^2 + 171w + 9, \tag{4.14}$$

då $d \geq 1$. Eftersom djupa FCN har en tendens att överträna introducerades även regulariseringsmetoderna avhopp och L1/L2 regularisering från avsnitt 3.7.

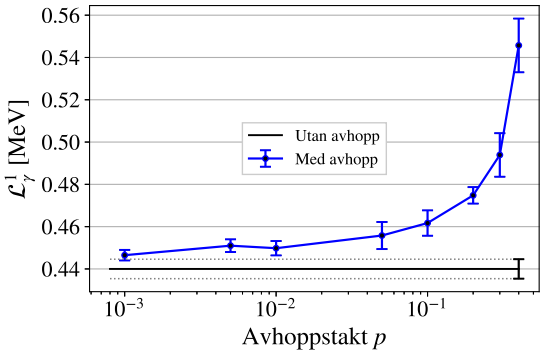


Figur 4.7: $\mathcal{L}_\gamma^1$ för ett FCN med olika djup och bredd. Den bästa prestandan, $\mathcal{L}_\gamma^1 = 0{,}4529\,\text{MeV}$, fås med $d = 6$ och $w = 1200$.

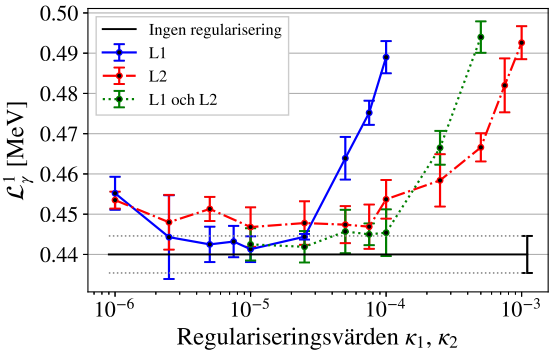
Att hitta den FCN-struktur med bäst prestanda innebär att hitta optimala värden på d, w samt någon av parametrarna p, κ_1 och κ_2. Eftersom strukturen främst definieras av d och w undersöktes dessa först med målsättning att gå djupare och bredare är vad tidigare grupper gjort. I figur 4.7 visas resultatet av träningar av FCN för varierande d och w. Det går att urskilja ett område med $3 \leq d \leq 7$ och $500 \leq w \leq 2000$ där nätverken är som bäst, vilket är brett relativt vad som hittats av tidigare grupper. Det bästa nätverket hittades med $d = 6$ och $w = 1200$ med $\mathcal{L}_\gamma^1 = 0{,}4529\,\text{MeV}$ och namngivs till `FCN-d6-w1200`.

Användningen av regularisering för att motverka överträning visade sig försämra nätvekens prestanda. I figur 4.8 visas prestandan av ett FCN med avhoppslager efter varje fullt kopplat lager för olika avhoppstakter. Nätverket med avhopp går tydligt mot nätverket utan avhopp då $p \to 0$. Speciellt ger

värden av avhoppstakten som ofta rekommenderas, $p = 0{,}5$ [13], en betydlig försämring.



Figur 4.8: Prestandan av `FCN-d6-w1200` med aktiverade avhoppslager för olika avhoppstakter. Den svarta horisontella linjen motsvarar ett FCN utan avhopp. Avhoppen gör nätverket avsevärt sämre.



Figur 4.9: Prestandan av `FCN-d6-w1200` när nätverket straffas för stora vikter enligt L1, L2 och en kombination av L1- och L2-regularisering. När båda är aktiverade används κ_1 och $\kappa_2 = 10\kappa_1$. Den svarta horisontella linjen motsvarar ett FCN utan regularisering. Ingen av metoderna gör nätverket statistiskt sätt bättre och träningen ter sig också vara mer oförutsägbar.

I figur 4.9 visas L1, L2 och både L1 och L2 regularisering när varje lager i nätverket ger ett tillskott till kostnadsfunktionen. Inte heller här ger regularisering en bättre prestanda, även om vissa mätningar kommer under värdet utan regularisering.

4.8 Faltande neuronnät

Den stora fördelen med faltande nätverk är att samma parametrar kan användas över hela kristallstrukturen. CNN vilar dock på antagandet att indatan är translationsinvariant se avsnitt 4.8.2 [14, s. 291]. Eftersom indatan innehåller värden för fyra olika kristaller är antagandet inte uppfyllt och indatan måste omorganiseras dels med hänsyn till translationsinvariansen och dels för att respektera kristallbollens geometri.

4.8.1 Omorganisera indata

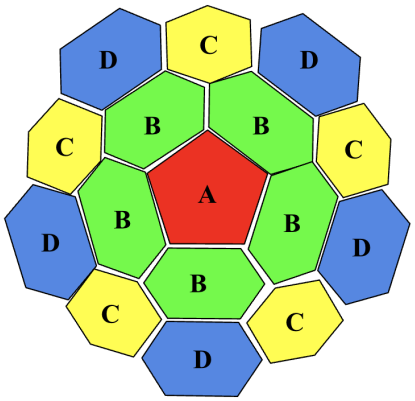
Indatan till CNN består av en lista på 162 energideponeringar och arrangeras om genom att gruppera elementen inom grupper av närliggande kristaller, likt tidigare arbeten [2]. Kristallbollens kristaller kan ordnas i grupper bestående av en kristall i centrum och centrumkristallens första- och andragrannar. På grund av kristallbollens geometri är detta enbart möjligt för **A**- och **D**-kristallerna, se figur 4.10 och 4.11. För grupperna med en **A**-kristall i centrum består gruppen totalt av 16 stycken kristaller medan grupperna med **D**-kristaller i centrum har 19 medlemmar.

Indatan ordnas på **A**- och **D**-grupperna och sorteras utefter tidigare gruppens metod vilket de benämner kristalltypsortering (*Consistent Crystal Type-sorting*, CCT-sortering) [2]. En grupp med kristall k i centrum betecknas med (k **n** **m**), där **n** och **m** representerar listor med första- respektive andragrannarna. I **n** placeras först den kristall som är närmast kristallbollens

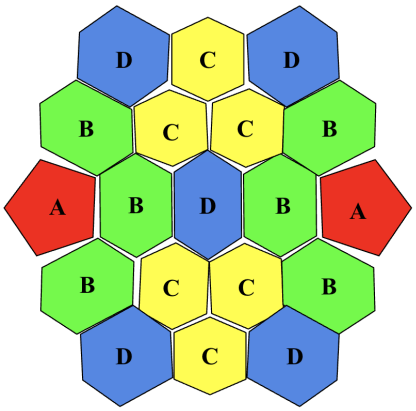
strålutgång och efterföljande förstagrannar sorteras moturs runt centerkristallen. För att sortera andragrannarna väljs den kristall, m_1, som är granne till n_1 och n_2. Två mängder M_1 och M_2 definieras som grannarna till det första och andra elementet i **n**. Matematiskt, kan m_1 beskrivas enligt

$$m_1 = \{M_1 \cap M_2\} \setminus \{k\}. \tag{4.15}$$

Andragrannarna **m** sorteras moturs precis som tidigare.

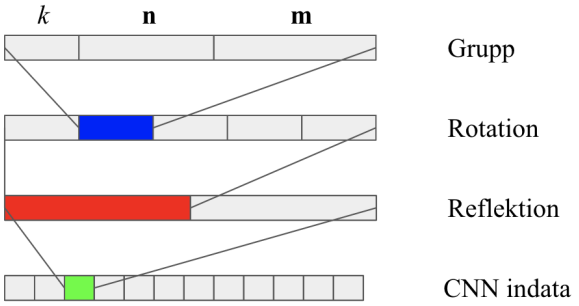


Figur 4.10: En grupp av kristaller med en **A**-kristall i centrum och dess första- och andragrannar.



Figur 4.11: En grupp av kristaller med en **D**-kristall i centrum och dess första- och andragrannar.

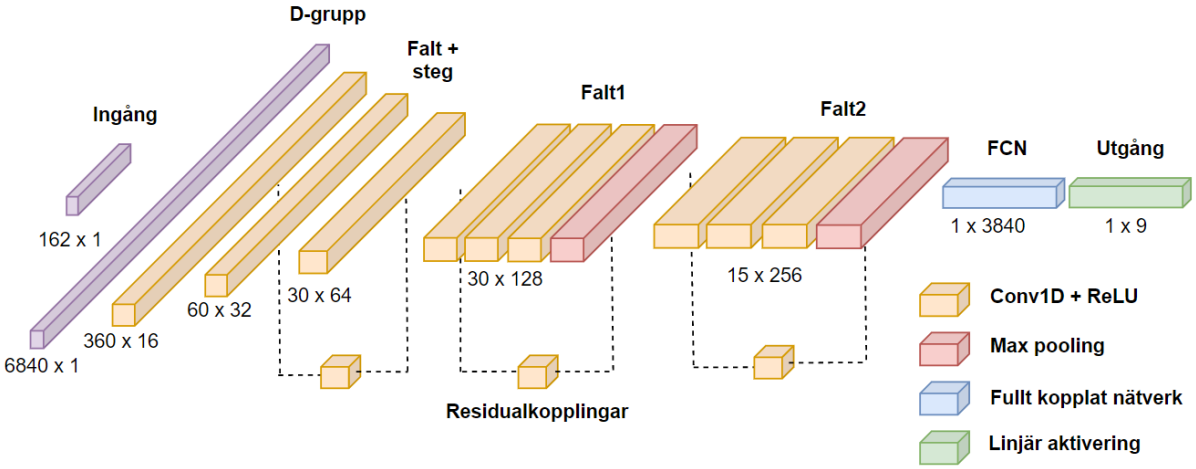
För att underlätta för nätverken att känna igen mönster kan det vara fördelaktigt att utnyttja rotationssymmetrier. Exempelvis bör en foton som sprids mot andragrannar i en grupp detekteras oberoende var de förhåller sig till kristallen i centrum. Samma gäller för spegelbilden av gruppen. Listorna med grannarna som utgör gruppen utökades därför till att även innehålla alla varianter av rotationer och reflektioner av gruppen. I figur 4.12 visas hur utökningen av indatan sker i praktiken för en **A**-kristall.



Figur 4.12: Schematisk bild av hur CNN indatan innehåller reflektionen och rotationer av en grupp **A**-kristaller [2].

4.8.2 Utveckling av bredare, djupare och bättre CNN

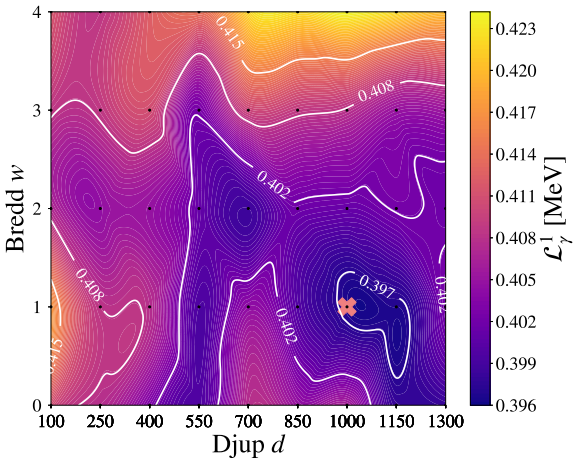
Omorganiseringen producerar två translationsinvarianta indatatyper i form av 12 grupper med **A**-kristaller i mitten och 30 med **D**-kristaller i mitten samt alla rotationer och reflektioner för dessa. Det är därför lämpligt att konstruera två parallella endimensionella faltande lager, likt tidigare gruppers arbeten [2]. Ett **krav** är att det första faltande lagret för varje kristallgrupp har steg- och kärnstorlek $S_A = 16$ eller $S_D = 19$ för **A**- respektive **D**-grupper motsvarande antalet kristaller i gruppen. Detta gör att faltningen alltid sker gruppvis och därför är translationsinvariant.



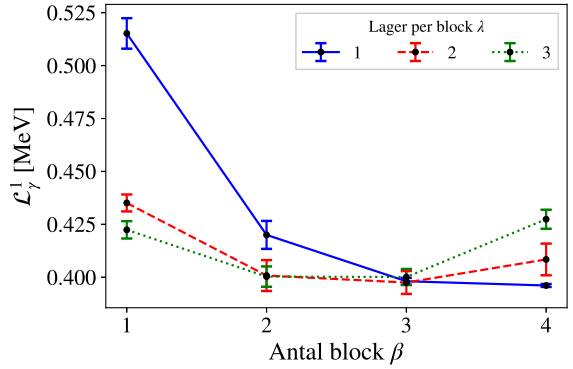
Figur 4.13: Strukturen av ett residualt faltat nätverk skapat för prediktion av tre γ-fotoner. Indatan är transformerad för **D**-kluster och stegas sedan ned för att behålla translationsinvarians. Falt1 och Falt2 indikerar block av faltande lager följt av en max-*pooling*-operation. Ett FCN behandlar datan innan utgången. En liknande faltad del körs parallellt för **A**-kristallerna och matas in i samma avslutande FCN.

För nästkommande lager är friheten större eftersom varje element i datan motsvarar en grupp. Det andra faltande lagret ges steg- och kärnstorlek $r_A = 5$ respektive $r_D = 6$ utefter gruppernas rotation och det tredje ges steg- och kärnstorlek 2 för reflektion. Filterstorleken fördubblas också efter varje lager. Förhoppningen är att att kondensera informationen från grannar, rotationer och reflektioner för varje grupp till ett enda element som sedan skickas till nya lager. Därför hölls denna struktur konstant för alla CNN. Utdatan från alla de faltande lagrena motsvarar nu 12 **A**-grupper och 30 **D**-grupper.

Datan skickas sedan till ett regelbundet CNN bestående av β antal block innehållande λ antal faltande lager med steg- och kärnstorlek 3 och en max-*pooling* i slutet av blocket som halverar storleken. Residualkopplingar mellan första och tredje faltande lagret samt över varje block introducerades också med förhoppningen att det tillåter ännu djupare nätverk. Nätverken för de två kristallgrupperna kopplas slutligen samman följt av ett FCN med djup d och bredd w. I figur 4.13 visas nätverkets struktur för en **D**-gren av nätverket.



Figur 4.14: Prestanda för faltande nätverk baserade på figur 4.13 med tre block och två faltande lager per block. Bredden och djupen av det FCN som följer de faltande lagren har varierats. Det optimala värdet hittades vid $d = 1000$ och $w = 1$ med $\mathcal{L}_\gamma^1 = 0{,}3967$.



Figur 4.15: Prestanda för CNN med varierande antal block och antal lager per block. Nätverket är byggt enligt figur 4.13. Punkterna verkar konvergera till optimala värden vid tre block. Fem körningar per datapunkt.

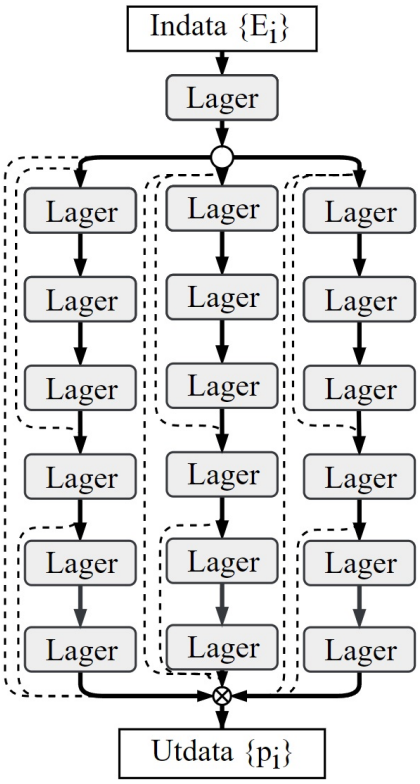
4.8.3 Optimering av CNN

Likt de andra nätverken finns det en rad strukturparametrar som kan varieras för det faltande nätverket. Vi valde att främst undersöka antal block β, faltande lager per block λ, bredden w av det fullt kopplade nätverket i slutet och djupet d av densamma. Nätverket med residualkopplingar presterade något bättre i genomsnitt än det utan. Gruppnormalisering verkade också förbättra nätverket och filterstorleken verkade enbart ha en liten inverkan, även om det måste medges att detta inte undersöktes grundligt. Eftersom välkända och speciellt välpresterande nätverk ofta utnyttjar residualkopplingar, gruppnormalisering och en successivt ökande filterstorlek efter varje block valdes att också inkludera dessa.

Det nätverk som presterar bäst utifrån detta blev ett med $\beta = 3$, $\lambda = 2$, $d = 1000$ och $w = 1$. I figur 4.15 visas prestandan av nätverket för varierande antal block och antal lager per block. Den lägsta mätningen togs vid $\beta = 3$ och $\lambda = 2$. Dessa användes sedan för träningar där det fullt kopplade nätverket varierades i bredd och djup vilket visas i figur 4.14. Skillnaden mellan olika värden var inte särskilt stor och det minsta värdet hittades vid $d = 1000$ och $w = 1$ med $\mathcal{L}_\gamma^1 = 0{,}3967$.

4.9 ResNeXt

ResNeXt-arkitekturen är en utveckling av residualnätverksstrukturen och genom att göra nätverken mer avancerade med fler justerbara parametrar kan de optimeras i ytterligare en dimension. I det enklaste fallet med en grupp g är nätverket ett residualnätverk.



Figur 4.16: Flödesdiagram för ett ResNeXt med tre grupper och sex lager djupt. Den vita cirkeln i början av nätverket illustrerar en uppdelning av datan, den nedre cirkeln med kryss är en sammanfogning av utdatan. De heldragna linjerna visar fullt kopplade anslutningar och de streckade är residuala hopp. "Lager" står för fullt kopplade lager.

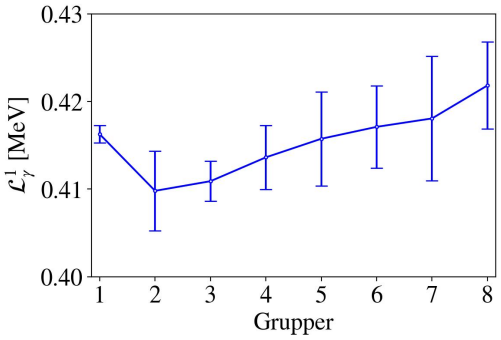
ResNeXt-strukturerna som har studerats är för det mesta likformiga i både gruppernas djup och bredd. Nätverk med både minskande bredd och djup som varierar bland grupperna presterade, i bästa fall, lika bra men oftast något sämre än likformiga och kommer därmed inte att visas. Gruppernas bredd och djup skrivs som w och d, där bredden är antalet noder i varje lager i alla grupper och d ej inkluderar det första lagret. För att uppdelningen ska garanterat fungera har det första lagrets bredd definierats som $g \cdot w$.

Flödesdiagram för ett likformigt ResNeXt-nätverk med tre grupper och sex lager i varje grupp ses i figur 4.16.

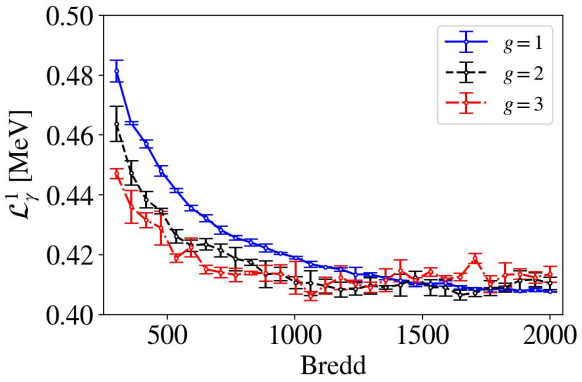
De tränade ResNeXt-nätverken som visas här har några fixa hyperparametrar: initial inlärningshastighet $\alpha_0 = 10^{-4}$, förfallhastighet $\mathrm{d}r = 96\,\%$, uppdaterar inlärningshastigheten stegvis (se avsnitt 3.5), och är tränade på $\approx 5 \cdot 10^6$ händelser med tre fotoner som mest.

4.9.1 Optimering av ResNeXt

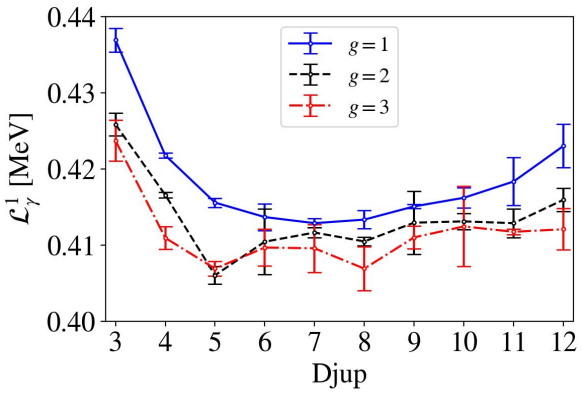
Eftersom ResNeXt har en dimension mer än FCN är det rimligt att det bästa djupet kommer att vara beroende av hur många grupper nätverket omfattar. Med en bredd på 1250 och ett djup på 6 har ResNeXt med olika antal grupper undersökts och prestandan varierar mellan ca 0,41 MeV − 0,42 MeV, se figur 4.17. Bredden och djupet har sedan varierats för en, två och tre grupper som figur 4.18 och 4.19 illustrerar. Likt FCN så presterar djupa nätverk sämre. Värt att notera är att prestandan varierar väldigt lite, ungefär 0,1 MeV mellan det högsta och lägsta värdet.



Figur 4.17: $\mathcal{L}_\gamma^1$ för ResNeXt-nätverk med $1 \leq g \leq 8$. Nätverket har $w = 1250$, $d = 6$, och förfallsteg $f_s = 4000$. Fem körningar per datapunkt.



Figur 4.18: Tre ResNeXt-nätverk med $g =$ 1,2,3 och varierande bredd. Notera intervallet på y-axeln.

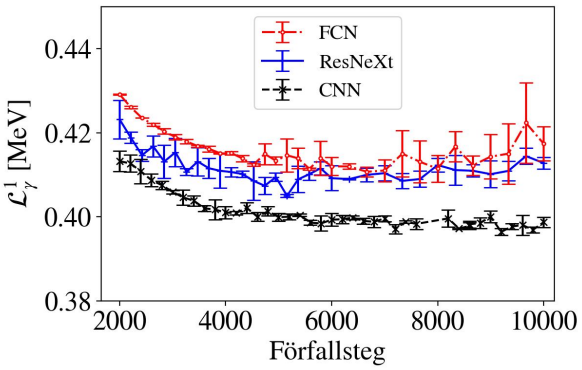


Figur 4.19: Tre olika ResNeXt-nätverk med $g = 1,2,3$ och $3 \leq d \leq 12$ jämförs baserat på $\mathcal{L}^1$.

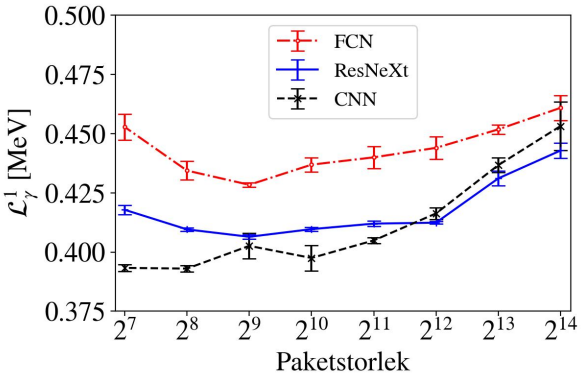
4.10 Träningsstyrande parametrar

Ett antal träningsstyrande parametrar har testats på de olika arkitekturerna. Vid höga paketstorlekar (*batch size*) ser felet vid rekonstruktion ut att bli sämre, se figur 4.21. För FCN verkar en paketstorlek på 2^9 ge bäst prestanda, medan CNN och ResNeXt inte har ett lika tydligt minimum. Med låg paketstorlek tar träningen mycket längre tid, så mycket som upp till åtta gånger längre än vissa högre paketstorlekar och risken ökar att träningen avslutas i förtid för att ha uppnått maximalt antal epoker, se appendix A.1.

Förutom paketstorlek har också förfallsteg för de olika arkitekturerna varierats och prestandan ökar generellt för en större steglängd men FCN får samtidigt en större spridning, se figur 4.20. CNN och ResNext är inte testade med avhopp, L1- och L2-regularisering de inte gav en önskad effekt på de fullt kopplade neuronnäten.



Figur 4.20: Prestanda för tre olika nätverk, FCN, ResNeXt ($g = 1$) och CNN beroende på förfallstegen.

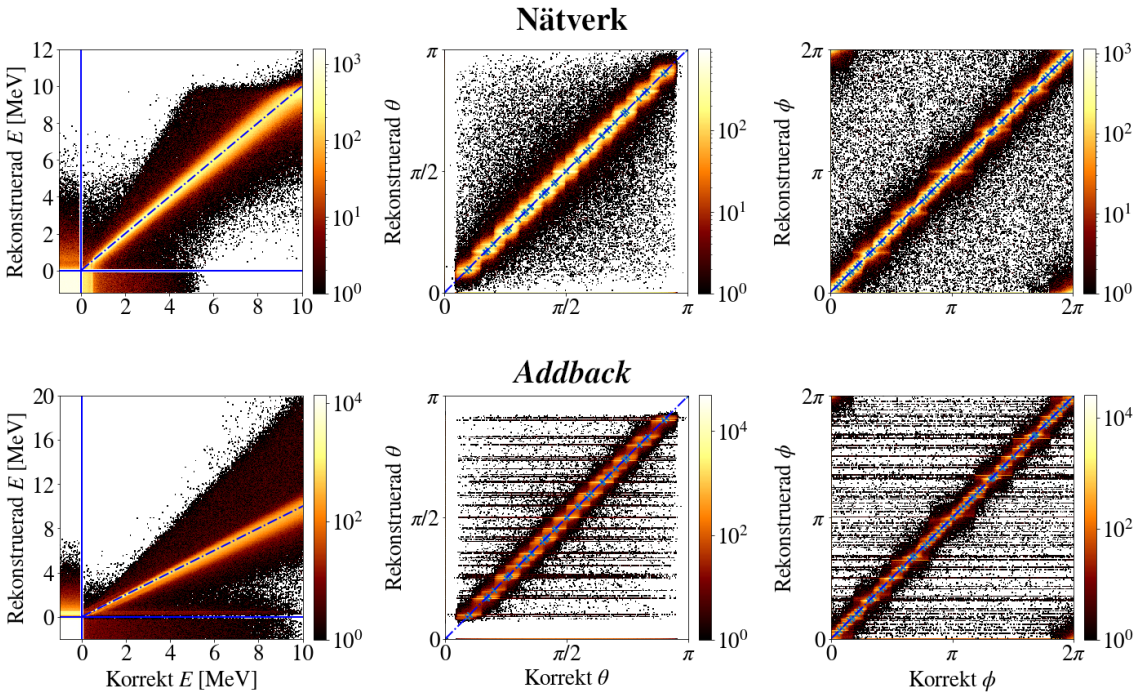


Figur 4.21: Prestanda för tre olika nätverk, FCN, ResNeXt ($g = 1$) och CNN beroende på paketstorleken.

5

Resultat och diskussion

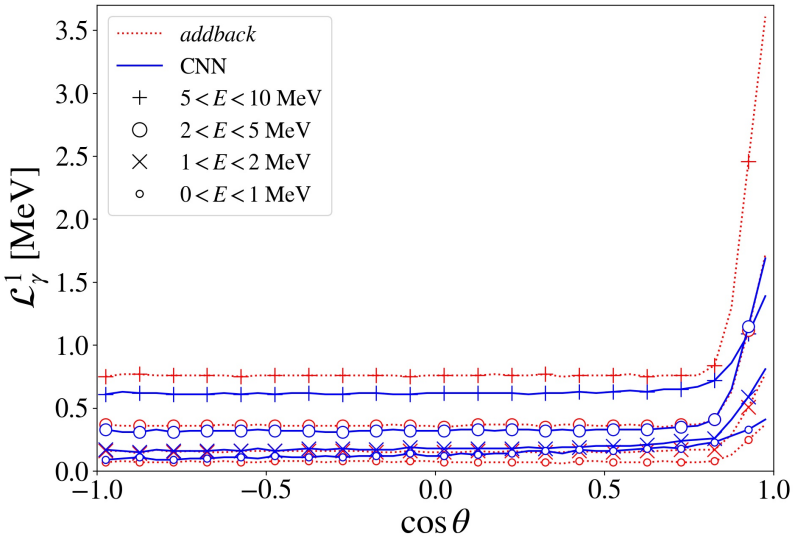
Våra ANN jämförs med *addback*-algoritmen med prestandamåttet $\mathcal{L}^1_\gamma$. I intervallet 0,1 MeV – 10 MeV lyckas FCN, CNN och ResNeXt prestera bättre än *addback* för rekonstruktion av maximalt tre fotoner. Vi visar att mer träningsdata upp till tio miljoner händelser förbättrar resultaten. Vi visar också att rekonstruktion utanför det tränade intervallet fungerar dåligt, men att nätverken oftare identifierar rätt antal fotoner.



Figur 5.1: Energi-, θ-, och ϕ-rekonstruktion för ett CNN som är tränat med likformigt fördelade händelser i intervallet $0{,}1 \leq E_\gamma \leq 10\,\text{MeV}$ och *addback*. Nätverkets prestanda är $\mathcal{L}^1_\gamma = 0{,}383(1)\,\text{MeV}$ och *addback*s är $\mathcal{L}^1_\gamma = 0{,}503(1)\,\text{MeV}$. Notera hur *addback* rekonstruerar betydligt högre energier upp till 20 MeV än nätverken. De horisontella strecken i vinkelrekonstruktionen kommer från detektorkristallernas vinklar.

Tabell 5.1: Uppmätt prestanda $\mathcal{L}_\gamma^1$ för olika nätverk och *addback* evaluerade över olika energier. Alla nätverk är tränade på $0{,}1 \leq E_\gamma \leq 10\,\text{MeV}$, 16 miljoner händelser. Varje kolumn visar prestandan för ett visst energiintervall. Energierna är angivna i MeV.

	$0{,}1 \leq E \leq 10$	$0{,}1 \leq E \leq 15$	$0{,}1 \leq E \leq 5$	$5 \leq E \leq 10$	$10 \leq E \leq 15$
FCN	0,399(1)	1,317(6)	0,243(1)	0,546(1)	2,371(6)
ResNeXt	0,395(2)	1,381(25)	0,243(1)	0,539(4)	2,577(89)
CNN	0,384(1)	1,374(4)	0,246(1)	0,522(2)	2,669(33)
Addback	0,503(10)	0,751(26)	0,250(6)	0,821(22)	1,362(40)



Figur 5.2: Prestanda för CNN och *addback* för händelser med olika öppningsvinklar θ mellan två γ för händelser med två simulerade γ. Nätverket är tränat på samma intervall som evalueringsdatan, $0{,}1 \leq E \leq 10\,\text{MeV}$. Nätverkets prestanda är $\mathcal{L}_\gamma^1 = 0{,}383(1)\,\text{MeV}$.

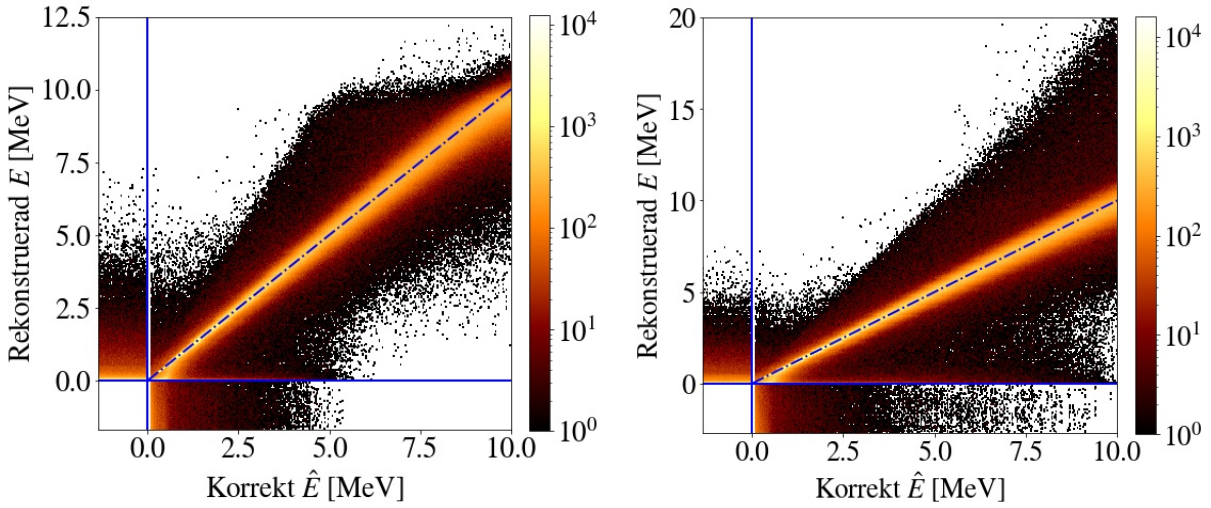
5.1 Nätverken presterar bättre än *addback*

För att jämföra *addback* med nätverken behövs ett prestandamått som rättvist bedömmer båda metoderna. I avsnitt 4.4 introducerades medelfelet per rekonstruerad γ-foton, $\mathcal{L}_\gamma^1$ (ekvation (3.3)), och medelfelet per händelse, $\mathcal{L}_e^1$, som används för att mäta prestandan (lägre värde innebär bättre prestanda). Eftersom nätverken ger utdata med tre γ-fotoner fås sambandet

$$\mathcal{L}_\gamma^1 = \frac{\mathcal{L}_e^1}{3} \tag{5.1}$$

och det är detta samband som gäller i hela rapporten. Prestandan för addback mäts enligt ekvation (5.1) och får alltså inte bättre prestanda för att ha rekonstruerat många fotoner med låg energi.

Enligt prestandamåttet $\mathcal{L}_\gamma^1$ presterar alla nätverk bättre än *addback* i utvärderingsintervallet $0{,}1 \leq E_\gamma \leq 10\,\text{MeV}$. De faltande nätverken hade bäst prestanda överlag, följt av ResNext, FCN och slutligen *addback*, se första kolumnen i tabell 5.1. För låga energier presterar alla tre neuronnät likvärdigt med *addback* medan neuronnäten



(a) ResNeXt-nätverk tränat för intervallet $0{,}1 \leq E_\gamma \leq 10\,\text{MeV}$, $1{,}6 \cdot 10^7$ händelser. Prestanda: $\mathcal{L}^1_\gamma = 0{,}395(2)\,\text{MeV}$

(b) ResNeXt-nätverk tränat för intervallet $0{,}1 \leq E_\gamma \leq 35\,\text{MeV}$, $2{,}4 \cdot 10^7$ händelser. Prestanda: $\mathcal{L}^1_\gamma = 0{,}439(2)\,\text{MeV}$

Figur 5.3: Energirekonstruktion med två ResNeXt-modeller tränade i olika intervall. Båda har försökt rekonstruera samma data.

presterar bättre vid högre energi (se kolumn tre och fyra i tabell 5.1). I energiintervall som neuronnäten inte är tränade på presterar *addback* bättre. Eftersom neuronnäten genom sin design inte kan rekonstruera fler än tre fotoner per händelse och är tränade på energier upp till 10 MeV kan det förklara en del av skillnaderna i prestanda jämfört med *addback*.

Eftersom *addback* inte har någon gräns för antalet γ-fotoner den kan rekonstruera så rekonstruerar *addback* många fler fotoner med låg energi än CNN, vilket visas i figur 5.1. I figuren syns också linjer i rekonstruktionen av vinklarna för *addback* vilket motsvarar kristallernas position i kristallbollen. För rekonstruktion av vinkeln ϕ syns också två områden i övre vänstra och nedre högra hörnet som är förväntat beror på att vinkeln är 2π-periodisk.

Addbacks största svårigheter är att rekonstruera höga energier och att skilja två γ-fotoner som träffar samma, eller närliggande kristaller, se figur 5.2. CNN har också svårare att skilja närliggande fotoner åt, för låga energier har den något svårare, eller lika svårt som *addback*.

5.2 Träningsdata

En av anledningarna till att neuronnäten presterar bättre än *addback* är att neuronnäten tränats på energier upp till 10 MeV och därför tenderar att inte rekonstruera högre energier. Ett ResNeXt tränat på energier upp till 10 MeV rekonstruerar ytterst sällan fotoner över det, se figur 5.3a. Ett ResNeXt som istället är tränat på en datamängd i intervallet $0 \leq E_\gamma \leq 35\,\text{MeV}$ (där ca 25 % av händelserna innehåller fotoner med energier över 10 MeV), har inte samma tak, utan rekonstruerar fotoner med energier upp till 20 MeV, figur 5.3b. Prestandan för nätverket är $\mathcal{L}^1_\gamma = 0{,}439(2)\,\text{MeV}$, vilket fortfarande är något bättre än *add-*

back. Även ett neuronnät som är tränat med likformigt fördelade händelser i intervallet 0,1 MeV − 35 MeV har en prestanda på 0,48 MeV. En förklaring till att neuronnäten försämras kan vara att de inte har tränats på en större datamängd i proportion till det ökade intervallet.

Mer träningsdata får prestandan hos neuronnäten att öka markant ända upp till 10 miljoner händelser (≈ 12 GB komprimerad data), se figur 5.4. Det visar sig att neuronnät tränade med en miljon händelser presterar bättre än *addback*. Mer träningsdata gör dock att träningen tar längre tid.

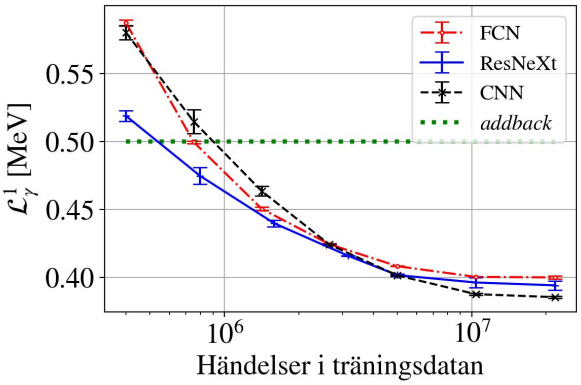
Träningstiderna har delvis varit väldigt varierande, troligtvis på grund av att endast delar av GPU-noder har använts. Träningstiderna kan då bero på hur kraftiga beräkningar resten av noden utför, vilket även figurerna A.1-A.7 i appendix A.1 visar.

Tabell 5.2: Sammanställning av träningstid för olika nätverk. Nätverken tränades på $5 \cdot 10^6$ händelser och har valts med de bäst uppmätta hyperparametrarna och träningen utfördes på 1×*NVIDIA A100 Tensor Core GPU*. För samband mellan träningstid och hyperparametrar, se appendix A.1.

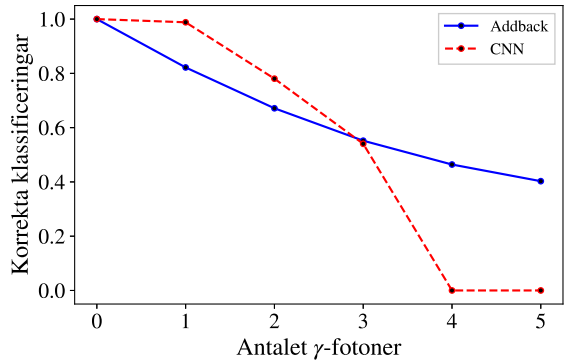
Arkitektur	Tid [min]	Träningsbara parametrar
FCN	77(6)	$7{,}41 \cdot 10^6$
ResNeXt	163(69)	$3{,}30 \cdot 10^7$
CNN	334(30)	$4{,}50 \cdot 10^6$

5.2.1 Klassificera γ-fotoner

Ett ytterligare illustrativt exempel på hur nätverken presterar bättre än *addback* där de är tränade men sämre annars är förmågan att klassificera antalet γ-fotoner. I figur 5.5 visas hur nätverken och *addback* avgör korrekt antal fotoner i en händelse. Observera att båda metoder utnyttjar toleransvillkoren från avsnitt 4.2 ekvation (4.5). Antalet fotoner som *addback* kan rekonstruera beror på antalet kristaller i detektorn till skillnad från nätverken som inte kan rekonstruera fler än tre fotoner på grund av sin design. Detta ger nätverken ett onaturligt stort övertag i vissa fall.



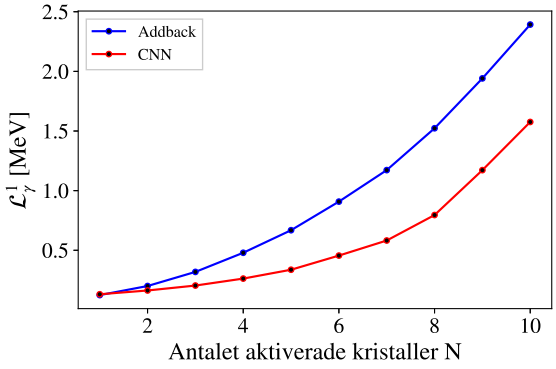
Figur 5.4: Prestanda för de tre bästa nätverken av respektive typ för olika träningsdatamängder. Notera den logaritmiska skalan för antalet händelser.



Figur 5.5: Förmåga att klassificera korrekt antal γ-fotoner i en händelse för *addback* och CNN. De andra nätverken har i stort sätt identiska kurvor. Notera att våra nätverk inte kan klassificera fler än tre fotoner.

5.2.2 Antalet aktiverade kristaller påverkar rekonstruktionen

Indatan är relativt gles, i genomsnitt aktiverar en γ-foton en till tre kristaller (se figur 4.1 i avsnitt 4.1). Om en foton deponerar all sin energi i en enda kristall kommer CNN och *addback* ge ungefär samma resultat och när fler kristaller aktiveras presterar CNN bättre än *addback*, se figur 5.6.



Figur 5.6: Förmåga hos nätverken och *addback* att rekonstruera händelser med en γ-foton där olika antal kristaller aktiverats av fotonen.

5.3 Slutsats

Till skillnad från tidigare arbeten [2–4] har vi lyckats träna ANN till att prestera bättre än *addback* i intervallet $0{,}1 \leq E_\gamma \leq 10\,\text{MeV}$ för maximalt tre γ-fotoner, utvärderat med ett medelabsolutfel per simulerad foton, $\mathcal{L}_\gamma^1$. Alla nätverk som studerats, FCN, ResNeXt och CNN har presterat bättre än *addback*, där CNN presterar bäst överlag med $\mathcal{L}_\gamma^1 = 0{,}384(1)$ MeV följt av ResNext, $\mathcal{L}_\gamma^1 = 0{,}395(2)$ MeV, och FCN, $\mathcal{L}_\gamma^1 = 0{,}399(1)$ MeV. Prestandan för *addback* är $\mathcal{L}_\gamma^1 = 0{,}503(1)$ MeV. ResNext presterar bättre för träning med färre händelser än övriga nätverk.

För energier och antal fotoner som neuronnäten inte är tränade på presterar *addback* dock fortfarande bättre än neuronnäten. Detta är i sig varken förvånande eller problematiskt eftersom nätverken inte rekonstruerar något de aldrig sett. Därför bör de exponeras för större energiintervall och fler antal fotoner. Tidigare arbeten [2, 4] har haft förhoppningen att prestandan på $0{,}1 - 10\,\text{MeV}$ då förblir densamma, men vi har funnit att detta inte är fallet. Speciellt leder det faktum att nätverken inte rekonstruerar fotoner *högre* än 10 MeV till att de har ökad prestanda *under* den energin. Nätverkens oförmåga att detektera fler än tre fotoner leder möjligtvis till liknande effekter. *Addback* har inte samma begränsningar som nätverken och kanske skulle tillämpningen av sådana påverka algoritmen positivt.

Fler undersökningar av nätverkens generalitet jämfört med *addback* krävs för att veta om nätverken är en lämplig metod för detektorrekonstruktion i verkligheten, även om nätverken presterar avsevärt bättre än *addback* för de fall vi har studerat.

6

Utvecklingsmöjligheter och sammanfattning

Projektet har syftat till att förbättra tidigare gruppers artificiella neuronnät [2–4] och undersöka hur träningsförfarandet påverkar resultaten. Vi diskuterar här framtida utvecklingar som kan göras för att förbättra neuronnäten och ger slutligen en sammanfattning av rapporten.

6.1 Undersökning av den genererade datan

Det vore intressant att träna, optimera och utvärdera nätverk skapade för att vara mer generella genom att använda data med ett större energiintervall och ett högre antal γ-fotoner. Vi tränade och optimerade enbart för $0 \leq E_\gamma \leq 10\,\mathrm{MeV}$ och fick där mycket goda resultat. Kanske skulle ett liknande förfarande för sådan data också leda till nätverk som är bättre än *addback* på att rekonstruera energier upp till energinivåer och fotonmultipliciteter som bortom rimligt tvivel kan komma från de kärnreaktioner som studeras med vår detektor.

Vidare vore det intressant att jämföra våra nätverk med nätverk tränade på data som inte enbart är likformigt fördelad. Som indatan genereras i `ggland` är γ-energin och multipliciteten likformigt fördelad och riktningen isotrop. Det är möjligt att data där fotonernas energi och multiplicitet är logaritmiskt fördelade mellan noll och mycket höga energier och multipliciteter kan producera ett nätverk med högre generalitet. Det vore även önskvärt att studera hur nätverken presterar på data som är relativistiskt korrigerad till strålen masscentrumsystem.

6.2 Andra kostnadsfunktioner

Den kostnadsfunktion som använts i detta projekt var $\mathcal{L}^1$. I avsnitt 4.5 diskuterades bland annat andra kostnadsfunktioner, en som visade sig vara sämre, $\mathcal{L}^2$, men även en som inte studerats lika noggrant, $\mathcal{L}_{\log}$. Medellogfelet ($\mathcal{L}_{\log}$) verkar till och med vara bättre än $\mathcal{L}^1$ men hann inte studeras lika mycket. För denna kostnadsfunktion verkar det vara möjligt att vikta kostnaden istället för datan. Kostnadsfunktionen straffar nätverket mer för små fel än för stora, vilket skulle innebära en eventuell förbättring för rekonstruktionen av låga energier.

6.3 Kombination av ResNeXt och CNN — en möjlig förbättring

I tidigare arbeten var FCN de mest framgångsrika nätverken. I detta projekt har vi lyckats träna både ResNeXt och CNN till att prestera bättre än FCN. ResNeXt-nätverken och de faltande nätverken är väldigt spännande då vi ännu inte har spenderat lika mycket tid på att optimera dessa jämfört med FCN. Ytterligare intressanta aspekter är hur dessa två nätverk skulle interagera med varandra.

De faltande nätverken som tränats i detta arbete har avslutas med fullt kopplade lager. Något som kan vara värt att vidare undersöka är hur nätverken beter sig om ett ResNeXt-nätverk används i slutet av CNN. För ResNeXt-neuronnäten används fullt kopplade lager och även där hade man kunnat testa byta ut fullt kopplade mot faltande, alltså en övergripande ResNeXt-struktur med faltande lager.

6.4 Djupgående studier av ResNeXt och CNN

Något som inte heller fokuserats särskilt på är själva strukturen i CNN och ResNeXt. Till exempel har vi inte gjort en djupgående undersökning av hur antalet lager som hoppas över i ResNeXt påverkar träningen, eller av olika kombinationer av lager-hopp som kan göras. CNN har inte heller optimerats lika noggrant som FCN, ändå ger de faltande nätverken väldigt låga fel.

6.5 Andra typer av optimerare

Många avgränsningar har genom åren behövt göras för detta arbete. Hur datan genereras, om den är verklighetstrogen eller inte och även valet av optimerare för de artificiella neuronnäten. I [19] visas grafer över hur olika optimerare påverkar tränings- och evalueringsfelen på bland annat CNN och ResNet (ytterligare en typ av residualnätverk likt ResNeXt). Även om `Adam` verkar optimera bra finns det andra typer av optimerare som kan vara ännu bättre, t.ex. `NAdam`, `Nesterov` och `RMSProp` [19].

6.6 Hur påverkar finare segmentering *addback* och neuronnäten?

Kristallbollen har 162 kristaller som detekterar energier. *Addback* har under en lång tid fungerat väldigt bra för just denna typ av detektorer medan neuronnäten inte riktigt nått samma nivå förrän nu. En fråga som kan ställas är hur *addback* och neuronnäten skulle prestera om antalet kristaller ökade, dvs. om segmenteringen av detektorn blev finare? Var skulle gränsen gå där *addback* inte kan konkurrera med neuronnäten längre? Att undersöka detta skulle också öppna upp till att studera grafneuronnät mer, då de antas prestera bättre på en detektor med högre geometrisk upplösning.

6.7 Grafneuronnät för detektorrekonstruktion

I detta arbete har som sagt endast FCN, CNN och ResNeXt-nätverk studerats men det finns självklart många andra typer av neuronnät. Tidigare år [2] har grafneuronnät (*Graph Neural Networks*, GNN) undersökts men inte gett något speciellt bra resultat. Å andra sidan presterade inte deras andra nätverk på samma nivå som våra och grafneuronnäten kanske har samma potential. Halldestam m.fl. nämner att det kan

vara intressant att se hur GNN presterar om antalet indatavärdena ökar. I fallet där kristallbollen skulle bestå av fler kristaller tros GNN kunna ha en fördel i kombination med FCN då det kan innebära färre träningsbara parametrar [2].

6.8 Hårdvara och sammanlagd träningstid

Majoriteten av träningen av neuronnäten har genomförts på superdatorn Alvis vid C3SE (Chalmers Centre for Computational Science and Engineering), genom SNIC (The Swedish National Infrastructure for Computing), under projektnamnet SNIC 2022/5-74. Ungefär 13 000 GPU-timmar har använts under projektet och de GPU:er som använts är framförallt NVIDIA A40 och NVIDIA A100.

6.9 Sammanfattning

Vi har undersökt flera olika typer av neuronnät och tagit fram ett prestandamått, $\mathcal{L}_\gamma^1$, för att kunna jämföra dem med varandra och med *addback*. hur olika träningsparametrar påverkar resultaten har också studerats, för att maximera nätverkens möjligheter att lyckas. För första gången har neutronnät presterat bättre än *addback* för rekonstruktion av γ-fotoner med energier i de intervall som nätverken tränades på ($0{,}1 \leq E_\gamma \leq 10$). Av de nätverken som vi har studerat presterade CNN allra bäst. För andra energiintervall har *addback* fortfarande ett övertag även om alla våra neuronnät i helhet presterar mycket bättre än tidigare arbeten [2–4]. Flera parametrar hos neuronnäten har studerats, generellt sett presterar neuronnäten som är tränade på större datamängder bättre men fler träningsbara parametrar innebär inte nödvändigtvis ett bättre nätverk. Olika versioner av *addback* har analyserats för att kunna jämföra den version som bäst rekonstruerar γ-fotoner med neuronnäten. Flera utvecklingsmöjligheter inom området finns, bland annat djupgående studier av faltande neuronnät och ResNeXt.

Litteratur

[1] S. Lindberg. *Optimised use of detector systems for relativistic radioactive beams*. Göteborg: Instutitionen för fysik, Chalmers tekniska högskola, 2013.

[2] Peter Halldestam m. fl. *Detector reconstruction of γ-rays*. Gothenburg, Sweden: Chalmers Univeristy of Technology och University of Gothenburg, 2020.

[3] Jacob Olander m. fl. *Rekonstruktion från detektordata med hjälp av neurala nätverk*. Gothenburg, Sweden: Chalmers Univeristy of Technology, 2018.

[4] Jesper Jönsson m. fl. *Event reconstruction of γ-rays using neural networks*. Gothenburg, Sweden: Chalmers Univeristy of Technology, 2019.

[5] R. S. Simon. *The Darmstadt-Heidelberg crystal ball*. Journal de Physique Colloques, 1980.

[6] W. R. Leo. *Techniques for nuclear and particle physics experiments: a how to approach*. Springer, 1987.

[7] W. Rindler. *Relativity: Special, General and Cosmological*. Oxford Press, 2006.

[8] Ian Goodfellow m. fl. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[9] Analytics Vidhya. *Binary Cross Entropy*. 2021. URL: https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/ (hämtad 2022-04-26).

[10] John McGonagle m. fl. *Backpropagation*. 2022. URL: https://brilliant.org/wiki/backpropagation/ (hämtad 2022-03-31).

[11] Diederik P. Kingma och Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2015. URL: https://doi.org/10.48550/arXiv.1412.6980.

[12] Suki Lau. *Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning*. 2022. URL: https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1 (hämtad 2022-05-08).

[13] Nitish Srivastava m. fl. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". I: *Journal of Machine Learning Research* 15.56 (2014), s. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[14] François Chollet. *Deep Learning with Python*. Manning Publications Co., 2021.

[15] Pooja Mahajan. *Fully Connected vs Convolutional Neural Networks*. 2020. URL: https://medium.com/swlh/fully-connected-vs-convolutional-neural-networks-813ca7bc6ee5 (hämtad 2022-01-31).

[16] Thomas Nilsson. *ROOT analysis and simple simulation write-up for the course FUF065/FIM465GU Advanced Subatomic Detection and Analysis Methods*. Gothenburg, Sweden: Chalmers University of Technology, 2015.

[17] *Tensorflow: Overview*. 2020. URL: https://www.tensorflow.org/overview/ (hämtad 2022-03-23).

[18] *Keras: The python deep learning library*. URL: https://keras.io/ (hämtad 2022-03-23).

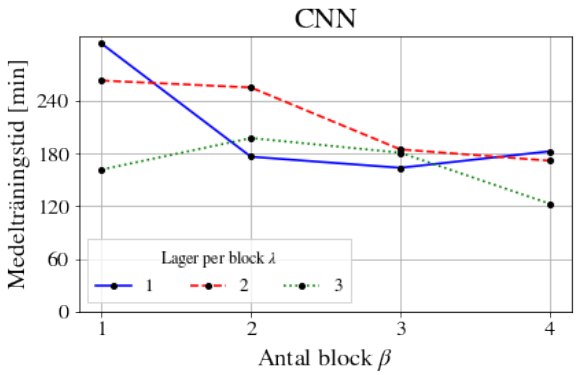
[19] Dami Choi m.fl. *On empirical comparisons of optimizers for deep learning*. 2019. URL: https://arxiv.org/pdf/1910.05446.pdf.

A

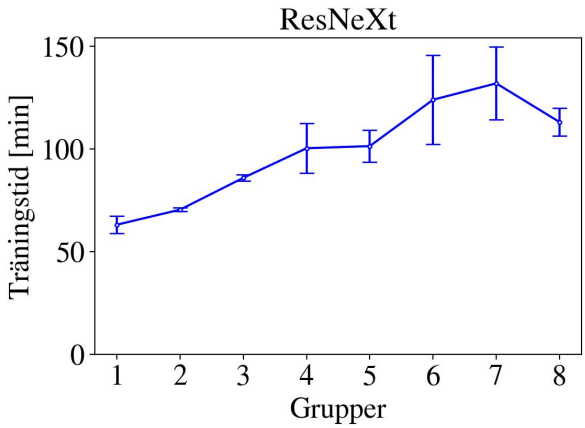
Träningstider

A.1 Träningstider

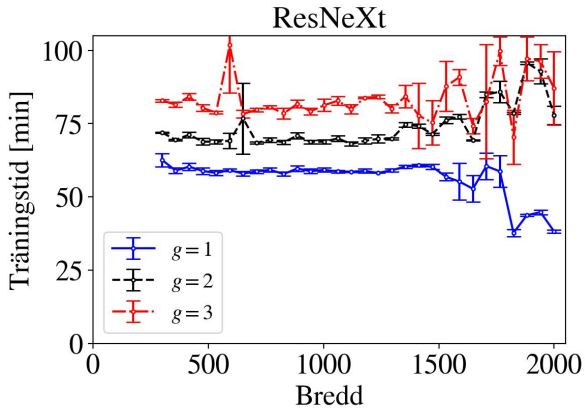
Figurerna A.1-A.7 visar träningstider för olika nätverk som funktion av några relevanta parametrar. Notera att ju fler grupper i ResNeXt desto längre tid tar träningen medan ju fler block i CNN desto mindre tid tar träningen. I figur A.5 syns det att ResNeXt också har längre träningstider än FCN. Alla träningar som visas här har använt en *A100 NVIDIA Tensor Core GPU* och en datamängd på $\approx 5 \cdot 10^6$ händelser (ungefär 6 GB av komprimerade data)



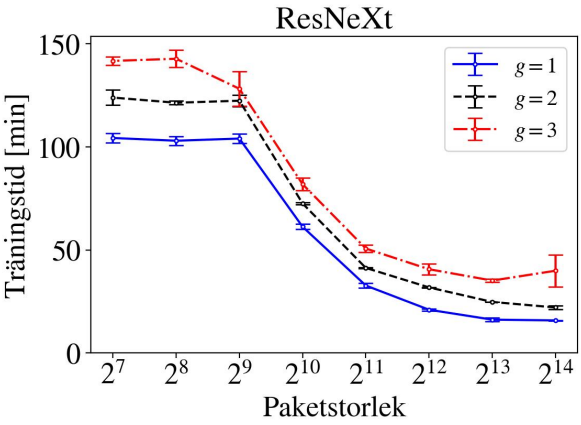
Figur A.1: Visar träningstiden för motsvarande graf i metodutvecklingen, figur 4.15



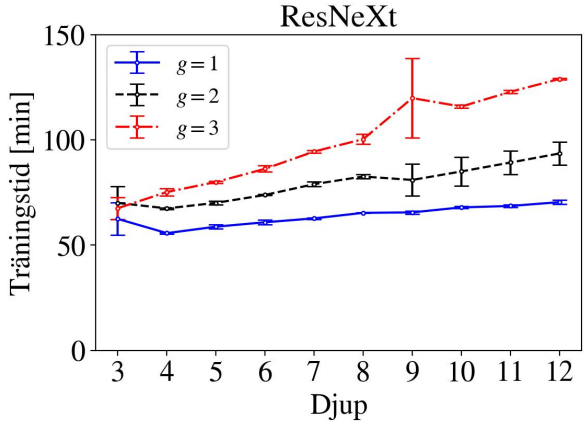
Figur A.2: Visar träningstiden för motsvarande graf i metodutvecklingen, figur 4.17.



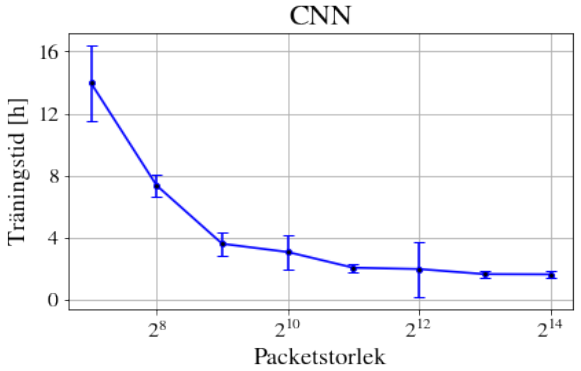
Figur A.3: Visar träningstiden för motsvarande graf i metodutvecklingen, figur 4.18.



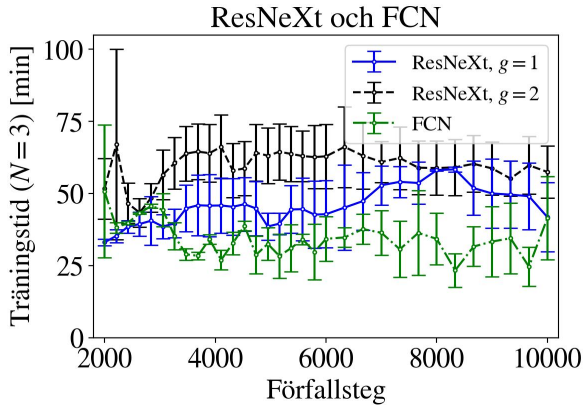
Figur A.6: Träningstiden för ResNeXt-nätverken i figur 4.21.



Figur A.4: Visar träningstiden för motsvarande graf i metodutvecklingen, figur 4.19.



Figur A.7: Visar träningstiden för de faltande nätverken för olika packetstorlekar. Notera att CNN prestarar bäst för låga packetstorlekar, visar i figut 4.21.



Figur A.5: Visar träningstiden för motsvarande graf i metodutvecklingen, figur 4.20.