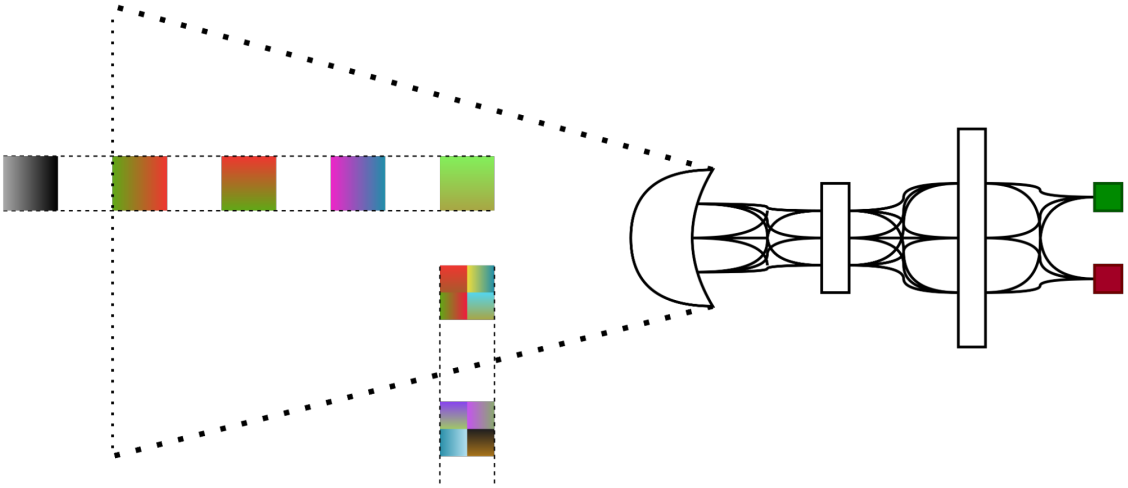
CHALMERS | GÖTEBORGS UNIVERSITET



Förstärkningsinlärning för rekommendationssystem

Utforskning av algoritmer och alternativa modeller för rekommendationer av bilder

Examensarbete inom högskoleingenjörsprogrammet i datateknik

Wincent Stålbert Holm
Oscar Marrero Engström

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2023
www.chalmers.se

Examensarbete 2023

Förstärkningsinlärning för rekommendationssystem

Utforskning av algoritmer och alternativa modeller för rekommendationer av bilder

Wincent Stålbert Holm
Oscar Marrero Engström

GÖTEBORGS UNIVERSITET

CHALMERS

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
Göteborg 2023

Förstärkningsinlärning för rekommendationssystem
Utforskning av algoritmer och alternativa modeller för rekommendationer av bilder
Wincent Stålbert Holm
Oscar Marrero Engström

Handledare:
Joachim von Hacht, Institutionen för Data- och Informationsteknik
Martin Tegner, Ikea

Examinator:
Lars Svensson, Institutionen för Data- och Informationsteknik

Examensarbete 2023
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslagsbild: En visuell representation av en modell som använts i arbetet för att skapa rekommendationer.

Skriven i LaTeX
Göteborg 2023

Sammanfattning

Rekommendationssystem som drivs av maskininlärning är en viktig del av många e-tjänster. De föreslår produkter, filmer, bilder och dylikt till användare i hopp om att de ska vara intressanta och driva engagemang. Att rekommendationer är välanpassade för varje användare är målet med ett rekommendationssystem och maskininlärning används i växande utsträckning för detta ändamål. Ett paradigm inom maskininlärning som visat sig högst aktuell för rekommendationssystem är förstärkningsinlärning. Forskning kring användningen av förstärkningsinlärning för rekommendationssystem är pågående och det finns rum för att experimentera. Denna uppsats presenterar olika förstärkningsinlärningsalgoritmer och modeller som formulerats, implementerats och utvärderats. De modellerna som visat sig mest lovande avviker från ett handlingsrum som innehåller alla möjliga rekommendationer och istället bedömer om en specifik rekommendation är bra eller inte. Resultaten visar att denna princip är värd att undersöka ytterligare. Värdena på hyperparametrarna och andra förutsättningar visar sig ha stor roll i hur algoritmerna presterar. Som indata till modellerna har både träningsbara inbäddningslager och resultat från faltningsnät testats. Det går inte att visa att faltningsnät presterar bättre än träningsbara inbäddningslager och ytterligare undersökning av detta rekommenderas.

Nyckelord: rekommendationssystem, maskininlärning, förstärkningsinlärning, algoritmer, modeller, tränigsbara inbäddningslager, faltningsnät

Abstract

Recommender systems utilizing machine learning are an important part for many internet services. They suggest products, movies, images and more to users in the hope that they will be interesting and increase user engagement. It is paramount that recommendations are well personalized to produce optimal results, and machine learning has proven to be a promising method for this purpose. One paradigm within machine learning which has shown itself particularly suited for recommender systems is reinforcement learning. Research on the subject of reinforcement learning for recommender systems is ongoing and there is room for experimenting. This degree project report presents various reinforcement learning algorithms and models which have been formulated, implemented and evaluated. The models which have shown the most promising results are those which deviate from an action space with all possible recommendations. Instead, the model makes a decision on whether a specific recommendation is good or bad. The results show that this principle is worth exploring further. Hyperparameter values and other conditions which the algorithms operate on have a considerable impact on the results. Both trainable embedding layers and convolutional neural networks have been tested as inputs to the model. It can not be shown that convolutional neural networks perform better than trainable embedding layers, a topic which warrants further research.

Keywords: recommender systems, machine learning, reinforcement learning, algorithms, models, trainable embedding layers, convolutional neural networks

Förord

Vi vill tacka Ikea och Group Digital för möjligheten att genomföra vårt examensarbete hos dem. Speciellt vill vi tacka Martin Tegner som handlett oss under hela processen och alltid erbjudit utomordentlig återkoppling när vi behövt det.

Vi vill också tacka Frida Nilsson som har organiserat och ansvarat för pilotprogrammet som gett oss möjligheten att skriva vårt examensarbete hos Ikea under våren 2023. Ett särskilt tack till Sandhya Sachidanandan som hjälpt oss med många av de tekniska aspekterna av arbetet. Dessutom vill vi tacka Josefin Nyquist och Dino Spirtovic som hjälpt oss med diskussioner kring utvecklingen av examensarbetet.

Slutligen vill vi tacka Joachim von Hacht – vår handledare på Chalmers – som hjälpt oss noggrant med utformningen av rapporten för examensarbetet.

Wincent Stålbert Holm, Oscar Marrero Engström, Göteborg, juni 2023

Akronymer

En lista av akronymer som används i texten sorterad i alfabetisk ordning:

AI	Artificiell intelligens	(*eng.* Artificial intelligence)
ANN	Artificiella neuronnät	(*eng.* Artificial neural network)
CNN	Faltningsnät	(*eng.* Convolutional neural network)
RL	Forstärkningsinlärning	(*eng.* Reinforcement learning)
CF	Kollaborativ filtrering	(*eng.* Colaborative filtering)
MDP	Markov beslutsprocess	(*eng.* Markov decision process)
ML	Maskininlärning	(*eng.* Machine learning)
MSE	Medelkvadratfel	(*eng.* Mean squared error)

Ordlista

Rekommendationsystem: Ett verktyg för att producera rekommendationer till användare, ofta i en kommersiell miljö

Inspirationsbild: En iscensatt bild av en samling produkter på Ikeas webbplatser

Maskininlärning: Ett fält inom datavetenskap och mer specifikt inom artificiell intelligens som ofta kännetecknas av självlärande

Förstärkningsinlärning: Ett av de tre grundläggande paradigmen inom maskininlärning som fokuserar på att maximera en erhållen belöning genom att låta en agent bestämma optimala handlingar

Algoritm: En serie av instruktioner som följs för att utföra en uppgift

Modell: En abstraktion av ett system som ofta bygger på matematiska samband och används för att utifrån ett antal värden producera andra värden som ett resultat

Artificiella neuronnät: En modell som används inom maskininlärning som består av en mängd noder uppdelade i lager och anslutna till varandra

Hyperparametrar: Variabler som bestäms innan en maskininlärningsalgoritm körs och som påverkar dess prestanda

Faltningsnät: Ett typ av neuronnät som generellt används för att analysera bilddata

Symboler

Listor av index, mängder, funktioner, hyperparametrar och variabler som används i texten:

Index

i,j	Används för allmän indexering av mängder
ℓ	Lager i ett neuronnät
t	Tidsindex

Mängder

$\mathcal{S}$	Tillståndsrum
$\mathcal{A}$	Handlingsrum
$\mathcal{B}$	Erfarenhetsbuffer
$\mathcal{D}$	Datapunkter
$\mathcal{T}$	Sekvenser av användarinteraktioner
$\mathcal{I}$	Bilder med produkter
$\mathcal{U}$	Användarprofiler
$\mathbb{Z}$	Heltal
$\mathbb{R}$	Reella tal

Funktioner

φ	Aktiveringsfunktion
$P_a(s, s^+)$	Sannolikheten för en tillståndövergång mellan s och s^+ vid handling a
$R_a(s, s^+)$	Belöning för en tillståndövergång mellan s och s^+ vid handling a
$Pr(a\|b)$	Sannolikheten för händelse a givet händelse b
$\pi(a, s)$	Policyfunktionen
R	Framtida reducerade belöningen
$V_\pi(s)$	Värdefunktionen för ett tillstånd s givet en policy π

Hyperparametrar

m	Storleken på inbäddningsvektorn
γ	Reduktionsfaktorn
ε	Girighetsfaktorn
τ	Mjuk uppdateringsfaktor
`buffer_size`	Storlek på erfarenhetsbuffer
`batch_size`	Storlek på en batch från erfarenhetsbuffern som tränas på
`epoch`	Antal epoker som tränas under inlärning
`positive`	Antalet positiva bilder som väljs vid ett tillstånd
`negative`, r	Förhållandet mellan negativa och positiva bilder i träningsdata
`forward`	Antal framtida användarinteraktioner som datapunkter konstrueras med
`LR`	Inlärningshasitget

Variabler

$w_{i,j}^\ell$	Vikt mellan nod i från lager $\ell - 1$ och nod j från lager ℓ
W^ℓ	Samtliga vikter mellan noderna i lager $\ell - 1$ och ℓ
n	Storleken på en en-kodad vektor
θ	Samtliga vikter i ett neuronnät
$\mathbf{o}$	Inbäddningsvektor
$\mathbf{x}$	En-kodad vektor
E	Inbäddningsmatris

Innehåll

Sammanfattning i

Abstract ii

Förord iii

Akronymer v

Ordlista vi

Symboler vii

Figurer xi

Tabeller xiii

1 Inledning 1
- 1.1 Bakgrund . . . 1
  - 1.1.1 Rekommendationssystem, Ikea och inpirationsbilder . . . 1
  - 1.1.2 Maskininlärning och förstärkningsinlärning . . . 2
- 1.2 Syfte . . . 4
- 1.3 Mål . . . 4
- 1.4 Avgränsningar . . . 4

2 Metod 5
- 2.1 Instudering . . . 5
- 2.2 Konstruktion av algoritmer och modeller . . . 5
- 2.3 Utvärdering av resultat . . . 6

3 Teoretisk bakgrund 7
- 3.1 Maskininlärning . . . 7
  - 3.1.1 Artificiella neuronnät . . . 7
  - 3.1.2 Hyperparametrar . . . 9
- 3.2 Markov-beslutsprocess . . . 9
- 3.3 Förstärkningsinlärning . . . 10
  - 3.3.1 Diskreta och kontinuerliga komponenter inom MDP . . . 11
  - 3.3.2 Numerisk data, kategorisk data och inbäddningslager . . . 11
  - 3.3.3 Epsilon-girig policy . . . 12

3.4 Djup-förstärkningsinlärning . . . 13
3.4.1 Förlustvärde . . . 13
3.4.2 Erfarenhetsbuffer . . . 14
3.4.3 Policynät och målnät . . . 14
3.5 Faltningsnät . . . 15
3.5.1 Faltningslager . . . 16
3.5.2 Samplingslager . . . 16

4 Teknisk bakgrund **17**
4.1 Google Analytics och BigQuery . . . 17
4.2 PyTorch och Hugging Face . . . 17
4.3 Vertex AI . . . 18

5 Genomförande **19**
5.1 Datahantering . . . 19
5.1.1 Generering av datapunkter . . . 19
5.2 Konstruktion av algoritmer och modeller . . . 21
5.2.1 Kategorisering . . . 22
5.2.2 Binär kategorisering . . . 23
5.2.3 Binär kategorisering med faltningsnät . . . 24
5.3 Träning av modeller . . . 25
5.4 Mått för utvärdering av prestanda . . . 25
5.4.1 Validering av modeller . . . 26

6 Resultat **27**
6.1 Kategorisering . . . 27
6.2 Binär kategorisering . . . 27
6.3 Binär kategorisering med faltningsnät . . . 30

7 Diskussion **32**
7.1 Samhälleliga, etiska och miljömässiga aspekter . . . 36

8 Slutsats **37**

9 Framtida arbete **38**

Källförteckning **39**

Figurer

1.1 Inspirationsbilder på Ikeas webbsida [3] 2

3.1 Beräkningen som sker i en nod med tre invärden 8
3.2 Ett framåtgående neuronnät 8
3.3 Tillståndsgrafen av en Markov beslutsprocess 9
3.4 En modell som tar emot tillstånd och producerar ett motsvarande Q-värde för dem 13
3.5 Faltningsnätet VGG-16 [27] 15
3.6 Visualisering av steg i ett faltningsnät 15
3.7 Faltning mellan indata och kärna 16
3.8 Poolingoperation applicerad på indata för dimensionsreducering . . . 16

5.1 Skapandet av en serie av datapunkter 19
5.2 Modell som tillämpas vid kategorisering av bilder för att skapa rekommendationer 22
5.3 Modell som tillämpas vid binär kategorisering av bilder för att skapa rekommendationer 23
5.4 Belöningsfunktion med exempelvärden 24
5.5 Förvirringsmatris med dess fyra möjliga utfall: sann positiv (*eng.* true positive (TP)), sann negativ (*eng.* true negative (TN)), falsk positiv (*eng.* false positive (FP)) och falsk negativ(*eng.* false negative (FN)) . 25

6.1 Träning och validering på 10^5 datapunkter med följande värden för algoritm 5.1: `size` $= 3$, `forward` $= 3$, `positive` $= 5$ och `negative` $= r = 5$ 28
6.2 Träning och validering på 10^5 datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= \infty$, `positive` $= 5$ och `negative` $= r = 5$ 28
6.3 Variationer av belöningsfunktionen 28
6.4 Träning och validering på $2.5{\cdot}10^5$ datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= 6$, `positive` $= 5$ och `negative` $= r = 15$ 29
6.5 Träning och validering på $2.5{\cdot}10^5$ datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= 6$, `positive` $= 5$ och `negative` $= r = 15$ 29

6.6 Träning och validering på $2.5{\cdot}10^5$ datapunkter med följande värden för algoritm 5.1: `size` = 6, `forward` = 6, `positive` = 5 och `negative` = $r = 5$ 30
6.7 Träning och validering på 10^6 datapunkter med följande värden för algoritm 5.1: `size` = 6, `forward` = 6, `positive` = 5 och `negative` = $r = 5$ 30
6.8 Träning och validering på 10^6 datapunkter med följande värden för algoritm 5.1: `size` = 6, `forward` = 6, `positive` = 5 och `negative` = $r = 5$ 31
6.9 Träning och validering på 10^6 datapunkter med följande värden för algoritm 5.1: `size` = 6, `forward` = 6, `positive` = 5 och `negative` = $r = 15$ 31
6.10 Träning och validering på 10^6 datapunkter med följande värden för algoritm 5.1: `size` = 6, `forward` = 6, `positive` = 5 och `negative` = $r = 15$ 31

Tabeller

3.1 Notation för komponenter som beskriver en Markov beslutsprocess . . 10
3.2 Kategorisk data 12
3.3 En-kodad data 12

6.1 Värden för hyperparametrar som inte varierats under arbetet [43] . . 27
6.2 Värden för hyperparametrar som används i resultaten från figur 6.1 och 6.2 28
6.3 Värden för hyperparametrar som används i resultaten från figur 6.1 och 6.2 29

1

Inledning

Detta avsnitt introducerar bakgrunden som gett upphov till detta examensarbete. Syftet med arbetet och målet som vill uppnås definieras också. Dessutom presenteras avgränsningar som gjorts avseende syftet.

1.1 Bakgrund

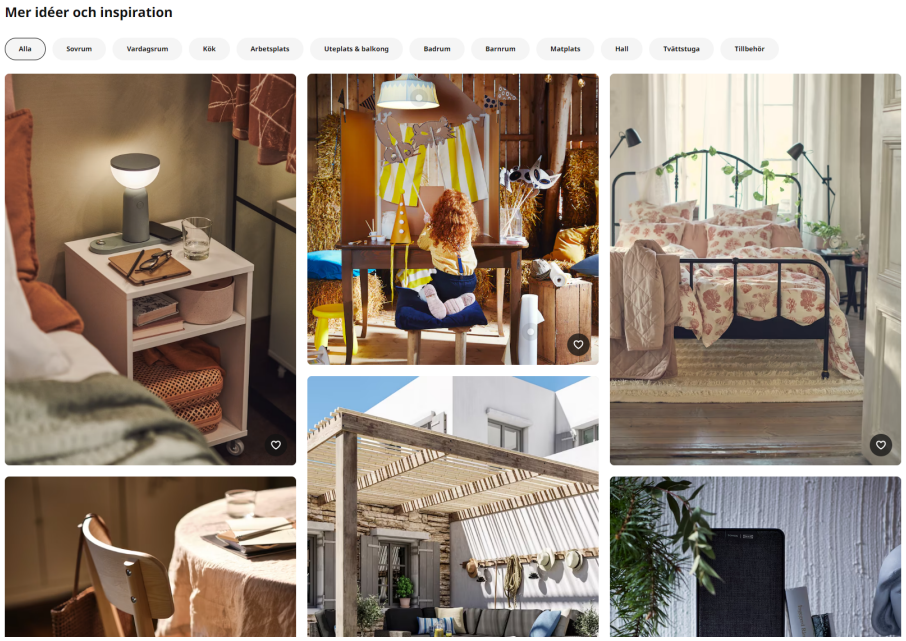
Rekommendationssystem är ett vanligt förekommande verktyg för att marknadsföra olika tjänster eller produkter som erbjuds av företag [1]. Målet med dessa system är att rekommendationerna som erbjuds ska vara skräddarsydda för varje användare och anpassade efter deras preferenser.

1.1.1 Rekommendationssystem, Ikea och inpirationsbilder

Rekommendationssystem bygger vanligtvis på anonymiserad användardata som tillämpas för att föreslå produkter eller tjänster. Användardata kan samlas in från webbsidor, mobilapplikationer och andra digitala plattformar. Informationen kan exempelvis bestå av vilka produkter en användare interagerat med, vilka produkter de har köpt och vilken marknad de tillhör. Insamlad användardata sparas i databaser med hjälp av olika verktyg, exempelvis Google Analytics [2].

Ett exempel på rekommendationssystem finns hos Ikea. På deras webbsidor används data för att skapa dynamiska rekommendationer som avser att visa potentiellt intressanta produkter och inspirationsbilder för användare (se figur 1.1).

Inspirationsbilder från Ikea kan beskrivas som en samling av produkter. Det är en scen med utplacerade produkter som är menade att inspirera användarna och få dem intresserade av relaterade produkter. Inspirationsbilderna presenteras för kunden på hemsidan, produktsidor och sidan man kommer till om man klickar på en inspirationsbild.



Figur 1.1: Inspirationsbilder på Ikeas webbsida [3]

Produkterna som visas i inspirationsbilder kan vara mer eller mindre relaterade sinsemellan. Detta påverkar hur en användare reagerar när en inspirationsbild rekommenderas eftersom den kan innehålla mycket intressanta eller helt irrelevanta produkter för användaren i fråga.

Att rekommendera samlingar av produkter istället för individuella sådana utgör en särskild utmaning. Det gäller inte längre att välja enstaka relaterade produkter, utan att från ett fördefinierat antal på cirka 10^4 inspirationsbilder försöka fånga användarens intresse.

Dessutom krävs det betydande resurser för att skapa inspirationsbilder. Antalet olika möjliga samlingar av produkter är mycket stort. Antalet faktiska inspirationsbilder måste därför begränsas på grund av resurserna de kräver för att produceras. Vissa produkter är följaktligen underrepresenterade i bilderna eller förekommer inte alls. Om en produkt anses relevant men saknas i relaterade inspirationsbilder blir det utmanande att producera en rekommendation för den.

1.1.2 Maskininlärning och förstärkningsinlärning

Rekommendationssystem har undergått omfattande utvecklingar under de senaste åren och dess funktionalitet baseras i dagsläget vanligen på maskininlärning (ML).

Maskininlärning är ett brett område inom datavetenskapen och klassas som en del av artificiell intelligens (AI) [4]. ML kännetecknas av algoritmer som utför någon form av bearbetning av befintlig data. Algoritmen tränar oftast på denna data och lär sig att lösa ett problem.

Exempel på olika typer av dessa problem kan vara att klassificera data i olika grupper, förutse ett värde givet en mängd datapunkter eller detektera kluster av punkter i en datamängd. Problemlösningen som ML erbjuder har tillämpningar inom en mängd olika områden, bland annat i e-postfiltrering [5], taligenkänning [6] och datorseende [7].

Maskininlärningsalgoritmer använder sig av en eller flera modeller som utvecklas under inlärningen. Dessa modeller tillämpas sedan för att lösa problem. Artificiella neuronnät (ANN) är ett exempel av en sådan modell. Ett ANN består av ett flertal sammankopplade noder och efterliknar funktionaliteten av ett biologiskt neuronnät.

Ytterligare ett koncept som är centralt för maskininlärning är hyperparametrar. Dessa parametrar är ofta numeriska variabler som bestämmer hur algoritmen presterar. Hyperparametrar kan också finjusteras för att begränsa komplexiteten av en modell och styr därför hur mycket resurser självinlärningen kräver och hur lång tid den tar att fullföljas [8].

Vissa maskininlärningsalgoritmer har visat sig användbara för att utveckla rekommendationssystem. Exempelvis är kollaborativ filtrering (*eng.* collaborative filtering, CF), en av de mest använda teknikerna bakom rekommendationssystem. Kollaborativ filtrering lider dock av vissa brister, exempelvis genom antagandet att användarnas preferenser inte ändras med tiden [9]. Populariteten hos alternativa varianter av rekommendationssystem har följaktligen växt sedan CF introducerades. Ett sådant alternativ är förstärkningsinlärning (*eng.* reinforcement learning, RL) [10].

Förstärkningsinlärning är ett av de grundläggande paradigmen inom maskininlärning. RL avser att modellera en problemställning och bestämma en optimal handling vid ett givet tillstånd. Målet är att bestämma den handling som för alla möjliga tillstånd maximerar en erhållen belöning [11]. Exempel på förstärkningsinlärningsalgoritmer är Q-inlärning (*eng.* Q-learning) och Djup Q-inlärning (*eng.* Deep Q-learning), som använder sig av det förutnämnda artificiella neuronnätet.

För att bestämma dessa optimala handlingar kräver algoritmerna stora mängder data [12]. Dessutom är ML och RL algoritmer i synnerhet mycket känsliga för vilka värden deras hyperparametrar ges. Eftersom dessa värden påverkar algoritmernas prestanda krävs det vanligen att de anpassas från problem till problem [13].

RL har visat mycket stor potential i olika forskningsprojekt som använt benchmarking för att evaluera förstärkningsinlärning [14][15]. Potentialen förstärkningsinlärning erbjuder tillsammans med de bristande egenskaperna hos traditionella algoritmer för rekommendationssystem som CF leder i dagsläget till att RL används i växande utsträckning för att implementera rekommendationssystem.

1.2 Syfte

Hur inspirationsbilder visas till och väljs av användare är centralt för rekommendationssystemet hos Ikea. Förstärkningsinlärning har möjlighet att välja relevanta rekommendationerna som driver engagemang.

Syftet med arbetet är att undersöka och implementera rekommendationssystem och försöka avgöra hur alternativa förstärkningsinlärningsalgoritmer och modeller kan användas för detta ändamål.

1.3 Mål

Målet med arbetet är att implementera, utvärdera och föreslå algoritmer och modeller för ett rekommendationssystem med förstärkningsinlärning som utgångspunkt. Modellerna ska uppnå en prestanda över en given baslinje, ett minsta prestandakrav som används för jämförelse. Minst två modeller ska implementeras och en jämförelse mellan deras prestanda ska kunna genomföras.

Arbetet ska ge inspiration för hur rekommendationssystem kan utvecklas samt hur system baserade på förstärkningsinlärning generellt kan användas.

1.4 Avgränsningar

Algoritmerna som implementeras kommer endast att utformas för att passa Ikeas användningsfall. Data som används kommer att vara från Ikea.

Möjligheterna till att utföra en fullskalig testning av algoritmerna är begränsade. Därför kommer utvärderingen av hur algoritmerna samt modellerna presterar endast att ske i en simulerad testmiljö med insamlad användardata.

Det finns ett stort antal hyperparametrar som skulle kunna undersökas för att förbättra prestandan. På grund av tidsbrist kommer antalet hyperparametrar som finjusteras under kalibreringen av algoritmerna att begränsas.

2
Metod

Detta avsnitt redogör för metoden som använts vid genomförandet av examensarbetet.

2.1 Instudering

För att öka förståelse för rekommendationssystem och förstärkningsinlärning kommer arbetet att börja med en instudering. I stor del kommer instudering att handla om att söka fram och läsa rapporter och andra publikationer. Forskningen kommer att relatera till ämnen som rekommendationssystem och förstärkningsinlärning. Databaser av artiklar som Chalmers Bibliotek och Google Scholar kommer att användas för att hitta forskningsartiklar. Som komplement till dessa artiklar kommer videomaterial att studeras. Videomaterial kommer internt från Ikea och från Coursera [16].

Möjligheten att diskutera arbetet med anställda på Ikea samt att få handledning kommer att utnyttjas. Kunskapsutbyte med anställda på Ikea kommer att vara en viktig del av arbetet. Diskussioner angående hur det nuvarande systemet har designats är nödvändigt för att utforska alternativa algoritmer. Kunskapsutbyte med handledare på Ikea kommer att ske regelbundet och kan också inkludera hjälp med programmering och instudering.

2.2 Konstruktion av algoritmer och modeller

Arbetet kommer att fokusera på att utifrån vedertagna algoritmer utveckla dessa genom att variera angreppssätt och strategier. Variationerna kan bestå av förenklingar av vissa delar av algoritmerna för att effektivisera dem. Samtidigt kommer ny funktionalitet att undersökas med hjälp av dessa variationer.

För att underlätta implementeringen av algoritmer kommer färdiga kodbibliotek användas. Ett sådant arbetssätt undviker grundarbetet med att implementera bland annat neuronnät. Arbetet med att implementera algoritmer kommer därför att börja med att leta efter relevanta kodbibliotek.

Data som kommer att användas för att kalibrera algoritmerna kan sorteras på bland annat tid och plats. Rangordningen av data kommer att användas för att under träningen begränsa och sedan öka datamängden. Till en början kommer data vara

begränsad till Sverige och under en period inte längre än ett dygn. Datamängden kommer senare att utökas för att finjustera algoritmens kalibrering.

2.3 Utvärdering av resultat

Ett sätt att utvärdera algoritmerna som tas fram under detta arbete är att utföra ett AB-test. En sådan utvärdering innebär att jämföra hur en ny algoritm presterar jämfört med en redan existerande sådan. Denna utvärderingsmetod har fördelen att algoritmen testas i ett verkligt sammanhang. AB-testning har dessutom potential att upptäcka problem med en ny algoritm som inte detekteras på grund av begränsningar i ett simulerat testningssammanhang.

Även om AB-testning i skarp miljö skulle vara önskvärt finns risker med det. Om en ny algoritm visar sig prestera sämre kan det innebära bland annat mindre inkomster och förlorade kunder. På grund av dessa risker med AB-testning kommer algoritmer att utvärderas i en simulerad testmiljö.

Utifrån insamlade användarinteraktioner kommer en kvantitativ utvärdering av algoritmerna genomföras och en uppskattning av algoritmens pricksäkerhet kan beräknas. Pricksäkerheten beskriver hur bra algoritmen förutser och rekommenderar något som användaren har visat intresse för. Pricksäkerhet och andra utvärderingsmått från den simulerade testmiljön kommer användas för att utvärdera algoritmens prestanda.

3

Teoretisk bakgrund

Detta avsnitt introducerar den teoretiska grunden som examensarbetet baseras på.

3.1 Maskininlärning

Maskininlärning är ett fält inom datavetenskap och mer specifikt inom AI. Fältet omfattar många typer av algoritmer som går ut på att lära en maskin ett beteende för att lösa ett problem [4]. Maskininlärningsalgoritmer bearbetar oftast stora datamängder utifrån vilka de kan lära sig ett beteende som avser att lösa ett givet problem utan att de specifikt blivit programmerade för det.

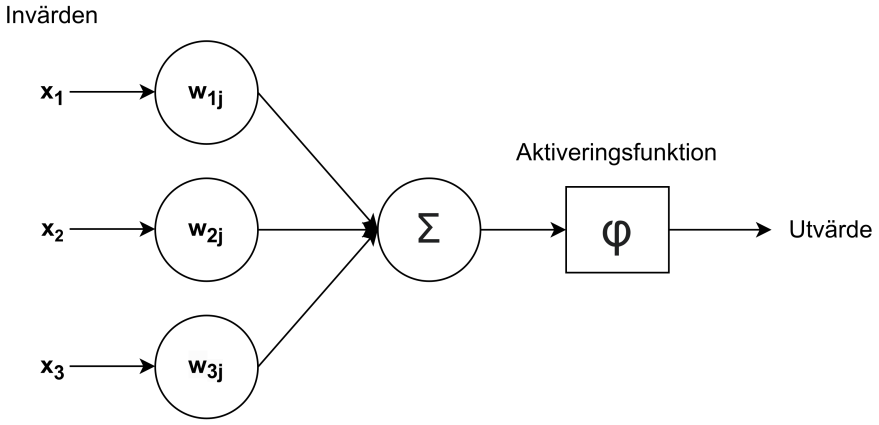
3.1.1 Artificiella neuronnät

Artificiella neuronnät är en typ av modell som tillämpas inom maskininlärning. Dessa neuronnät är ursprungligen inspirerade av dess biologiska motsvarighet och är uppbyggda av noder och anslutningar mellan dem. Noderna struktureras i olika lager och kopplas sedan till noder från andra lager [17].

Varje nod producerar ett värde som beräknas av den viktade summan av invärdena till noden. Den viktade summan passerar sedan aktiveringsfunktionen φ som producerar det slutliga utvärdet från en given nod. Ett exempel på en aktiveringsfunktion är ReLU [18] som ges av ekvation 3.1.

$$\varphi = f(x) = \max(0, x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{3.1}$$

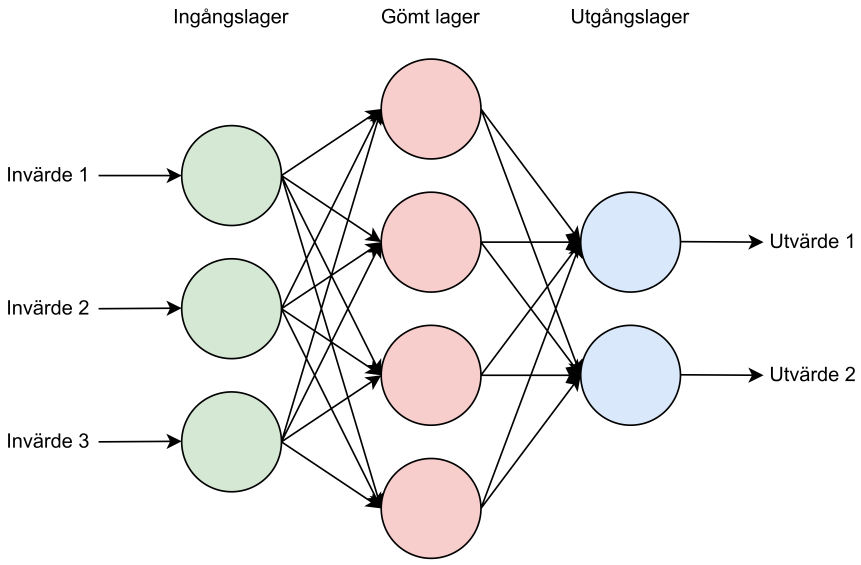
En överblick av beräkningen som genomförs i varje nod i ett neuronnät visas i figur 3.1.



Figur 3.1: Beräkningen som sker i en nod med tre invärden

Mängden av alla vikterna w_{ij} mellan ett lager ℓ och det föregående lagret ℓ−1 i ett neuronnät betecknas W^{ℓ}. Mängden av samtliga vikter för neuronnätet betecknas θ och kallas även för nätets parametrar. Detta ska inte förväxlas med algoritmens hyperparametrar.

Det första lagret i ett neuronnät kallas för ingångslagret, och det sista kallas för utgångslagret. Lagren däremellan kallas för gömda lager. Vanligtvis har alla noder från ett lager anslutningar till samtliga noder i nästa lager, vilket utgör en tät koppling. Vid ingångslagret och utgångslagret saknas föregående respektive påföljande lager. Invärden och utvärden till och från dessa lager genomgår ingen aktiveringsfunktion och viktas inte. Dessa värden är alltså direkt kopplade eller ett direkt resultat från ingångslagret respektive utgångslagret.



Figur 3.2: Ett framåtgående neuronnät

En specifik typ av ANN är framåtgående neuronnät [19] (*eng.* feed-forward neural networks), i vilka kopplingar mellan noder endast förekommer mellan intilliggande

lager (se figur 3.2). Andra typer av ANN som återkommande neuronnät (*eng.* recurrent neural networks) tillåter däremot kopplingar mellan övriga lager [20].

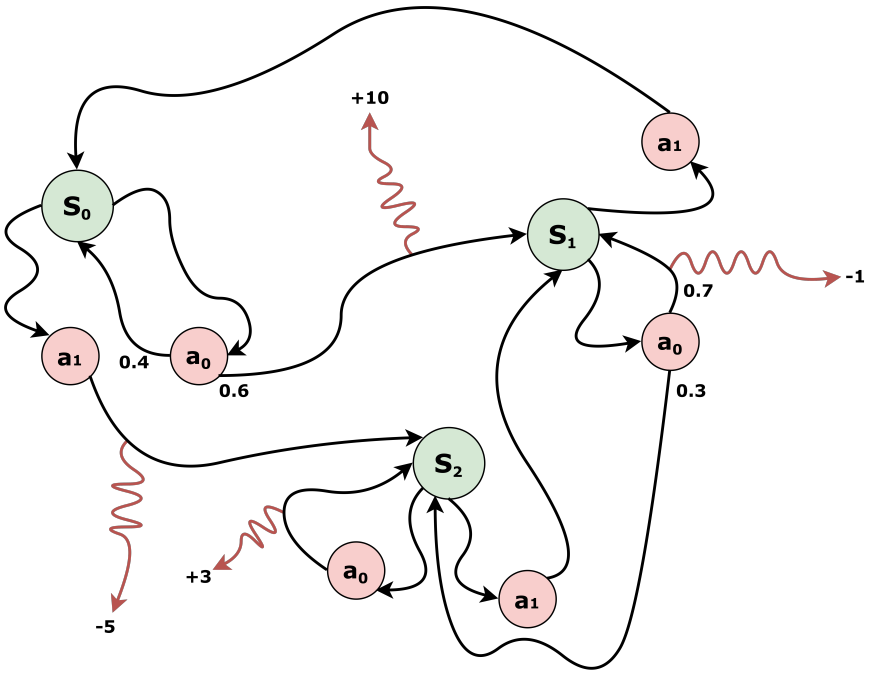
Parametrarna i neuronnät måste tränas – eller optimeras – för att försöka uppnå det problemlösande beteendet som önskas. Bakåtpropagering används vanligen för att justera vikterna baserat på ett beräknat förlustvärde (*eng.* loss). En vanlig algoritm för att optimera neuronnät är Adam, som använder sig av en variant av stokastisk gradientnedstigning (*eng.* stochastic gradient decent) för att minimera förlustvärdet [21].

3.1.2 Hyperparametrar

Hyperparametrar är variabler som bestäms innan algoritmens modell börjar träna, till exempel antal lager och noder i neuronnät. Dessutom ingår hyperparametrar i många av ekvationerna som står till grund för maskininlärningsalgoritmer [13]. Små justeringar i dem kan ha omfattande konsekvenser för algoritmens prestanda. Att justera hyperparametrarna är en del av att kalibrera en maskininlärningsalgoritm.

3.2 Markov-beslutsprocess

En Markov-beslutsprocess (*eng.* Markov decision process, MDP) är ett användbart verktyg för att modellera verkliga situationer där en viss slumpmässighet är förväntad. Beslutsprocessen är uppbyggd av fyra komponenter [22]. Komponenterna beskriver tillsammans en värld (*eng.* environment) och hur den utforskas av en agent. Världen i ett MDP kan illustreras med en tillståndsgraf (se figur 3.3).



Figur 3.3: Tillståndsgrafen av en Markov beslutsprocess

Världen i en Markov-beslutsprocess utforskas av en agent som utför handlingar. Handlingarna har sannolikheter att förflytta agenten från ett tillstånd till ett annat. När en förflyttning sker får agenten en belöning. Dessa komponenter beskrivs med notationen som presenteras i tabell 3.1.

Tabell 3.1: Notation för komponenter som beskriver en Markov beslutsprocess

$\mathcal{S}$	En mängd av tillstånd s kallat tillståndsrum
$\mathcal{A}$	En mängd av handlingar a kallat handlingsrum
$P_a(s, s^+)$	Sannolikheten att en förflyttnings mellan tillstånd s och s^+ sker om handling a väljs, ges av $Pr(s_{t+1} = s^+ \| s_t = s, a_t = a)$.
$R_a(s, s^+)$	Den omedelbara belöningen som erhålls vid en tillståndförflyttning mellan s och s^+ samt val av handling a

3.3 Förstärkningsinlärning

Förstärkningsinlärning är ett av tre grundläggande paradigm inom maskininlärning. Målet i RL är att lösa problemet som ett MDP presenterar. Problemet är att bestämma vilka handlingar agenten ska utföra i varje tillstånd för att maximera den kumulativa belöningen över tiden t.

I förstärkningsinlärning bestäms en policyfunktion som kalibreras för att för att uppnå den maximala kumulativa belöningen [11]. Policyfunktionen (se ekvation 3.2) ger sannolikheten att agenten väljer handling a när den befinner sig i tillstånd s.

$$\pi(a, s) = Pr(a_t = a | s_t = s) \tag{3.2}$$

Policyfunktionen kan användas för att värdera den förväntade framtida reducerade belöningen (se ekvation 3.3) när agenten befinner sig i tillstånd s och tillämpar π för att bestämma vilka handlingar den ska välja framöver [11].

$$R = \sum_{t=0}^{\infty} \gamma^t r_t \tag{3.3}$$

Den förväntade framtida reducerade belöningen - också kallad avkastning - använder sig av reduktionsfaktorn $\gamma \in [0, 1)$ som är en hyperparameter vald baserat på problemet förstärkningsinlärningsalgoritmen ska lösa. Värdet av γ bestämmer hur belöningar viktas beroende på hur långt i framtiden de ligger.

Tillsammans kan policyfunktionen och den förväntade framtida reducerade belöningen användas för att definiera en värdefunktion $V_\pi(s)$ (se ekvation 3.4). Värdefunktionen består av väntevärdet $\mathbb{E}$ (*eng.* expected value) av den framtida reducerade belöningen R givet ett initialt tillstånd s och en policyfunktion π [11].

$$V_\pi(s) = \mathbb{E}[R] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right] \tag{3.4}$$

3.3.1 Diskreta och kontinuerliga komponenter inom MDP

Problemet som modelleras med ett MDP bestämmer om mängderna $\mathcal{S}$ och $\mathcal{A}$ är diskreta eller kontinuerliga. Om det för alla $s \in \mathcal{S}$ och alla $a \in \mathcal{A}$ gäller att $s, a \in \mathbb{Z}$ då är mängderna $\mathcal{S}$ och $\mathcal{A}$ diskreta. Däremot om $s, a \in \mathbb{R}$ då är mängderna kontinuerliga [11].

Eftersom $P_a : \mathcal{S}^2 \to [0, 1]$ och $R_a : \mathcal{S}^2 \to \mathbb{R}$ bestämmer tillståndsmängden $\mathcal{S}$ om P_a och R_a har diskreta eller kontinuerliga definitionsmängder.

Ett exempel på diskreta tillstånd- och handlingsmängder är de som beskriver ett luffarschack. Om mängderna däremot är kontinuerliga kan dessa exempelvis användas för att beskriva tillståndet av en helikopter. Antalet möjliga positioner, rotationer och hastigheter helikoptern kan ha i ett tredimensionellt rum går ej att beskriva enbart med heltal.

En enkel typ av förstärkningsinlärningsalgoritm är Q-inlärning. Denna algoritm arbetar med en Q-tabell – en typ av uppslagstabell – för att bestämma en handling för varje tillstånd. Storleken på en Q-tabell ges av $|\mathcal{S}| \cdot |\mathcal{A}|$ och värdena i tabellen kallas för Q-värden. Den optimala handlingen i ett givet tillstånd kan således väljas utifrån handlingen som ger det högsta Q-värdet.

Även om både $\mathcal{S}$ och $\mathcal{A}$ är diskreta mängder kan tillräckligt stora tillstånds- och handlingsrum göra det opraktiskt att använda Q-inlärning. För att lösa problemet med att dimensionerna blir ohanterbara introduceras en eller flera modeller i andra typer av algoritmer. Neuronnät tillämpas exempelvis av vissa algoritmer för att approximera Q-värden när tillstånds- och handlingsrum är opraktiskt stora. Dessa typer av algoritmer kallas för djupa förstärkningsinlärningsalgoritmer [14].

3.3.2 Numerisk data, kategorisk data och inbäddningslager

Data till en modell går oftast att sortera in i två kategorier, numerisk och kategorisk. Numerisk data har en inbördes ordning. Det kan till exempel vara ett mått för temperatur eftersom skillnader i temperatur har en betydelse. Kategorisk data behöver till skillnad från numerisk data inte ha en inbördes ordning. Exempelvis har olika länder som en person kan komma från ingen självklar inbördes ordning. I en datamängd som innehåller sådan information skulle USA kunna representeras som numeriskt värde 1 och Sverige som 46. Skillnaden mellan 1 och 46 saknar således betydelse i detta fall.

Neuronnät förutsätter att all indata är numerisk, vilket innebär att kategorisk data måste omvandlas till numeriska värden. För att undvika problematiken med kategorisk data som saknar inbördes ordning kan inbäddningslager användas.

En inbäddning består av ett yttre lager som omvandlar indata till en modell från kategorisk data utan inbördes ordning till numerisk data med inbördes ordning.

Kategorisk data kan representeras av en en-kodad (*eng.* one-hot encoded) vektor [23]. I en sådan vektor ersätts kategorisk data av en vektor med ett värde för varje kategori. Den aktuella kategorin ges värde 1 medan övriga kategorier får värde 0. Ett exempel på en sådan omvandling illustreras i tabell 3.2 och 3.3.

Tabell 3.2: Kategorisk data

Land	**Befolkning (mn.)** [24]
Kina	1412
USA	331.9
Sverige	10.42

Tabell 3.3: En-kodad data

Kina	**USA**	**Sverige**	**Befolking (mn.)**
1	0	0	1412
0	1	0	331.9
0	0	1	10.42

Om $\mathbf{x}$ är en en-kodad vektor av kategorisk indata och E är en inbäddningsmatris går det att beräkna värdena för inbäddningsvektorn $\mathbf{o}$ genom en matrismultiplikation (se ekvation 3.5). Värdena i matris E justeras som andra vikter i ett neuronnät under träning.

$$\mathbf{o} = \mathbf{x} \cdot E = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \cdot \begin{bmatrix} \mathbf{e}_1^\intercal & \mathbf{e}_2^\intercal & \dots & \mathbf{e}_m^\intercal \end{bmatrix} \tag{3.5}$$

Antalet kategorier i den en-kodade vektorn $\mathbf{x}$ i ekvation 3.5 ges av n, medan storleken av den resulterade inbäddningsvektorn $\mathbf{o}$ ges av m.

3.3.3 Epsilon-girig policy

Inledningsvis kommer en agent välja handlingar som kan tolkas som slumpmässiga. Att utföra slumpmässiga handlingar tillåter agenten att undersöka vilken belöning som erhålls givet en handling och en tillståndsförflyttning. Värdet av den erhållna belöningen används sedan för att förstärka beteenden som anses positiva och straffa oönskade beteenden. Detta händelseförlopp upprepas för att successivt bestämma den optimala policyfunktionen . De allra flesta algoritmer inom området av förstärkningsinlärning följer detta mönster.

Om agenten strikt tillämpar en policyfunktion som den tror maximerar den erhållna belöningen kallas beteendet girigt. Problemet med girigt beteende är att handlingen sällan är den globalt optimala handlingen, åtminstone i början av modellens träning. För att förhindra att algoritmen fastnar i ett lokalt optimalt beteende kan slumpmässig utforskning tillämpas [16].

Slumpmässig utforskning innebär att agenten utför slumpmässiga handlingar. En metod för att tillämpa slumpmässig utforskning kallas epsilon-girig (*eng.* Epsilon-Greedy) policy. Den introducerar ett värde för hur girig agenten får vara och beskriver hur värdet ändrar på sig över tid (se ekvation 3.6).

$$\varepsilon_t = \varepsilon_{end} + (\varepsilon_{start} - \varepsilon_{end}) \cdot e^{-t/\varepsilon_{decay}} \tag{3.6}$$

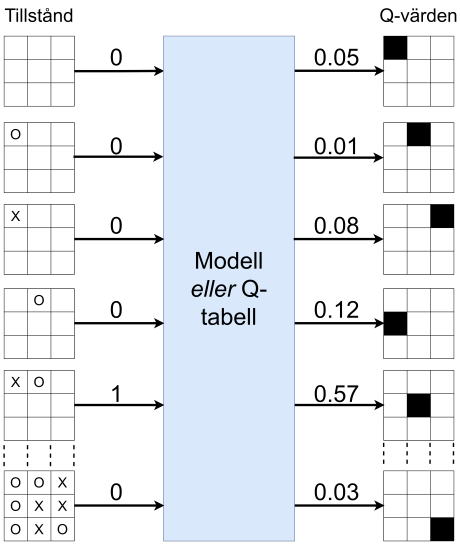
Värdet som ε antar vid tid t bestämmer hur sannolikt det är för algoritmen att slumpmässigt utforska i stället för att tillämpa policyfunktionen π. Slumpmässigt

valda handlingar kan ge algoritmen kännedom om både bra och dåliga val. Agenten kan sedan använda den informationen för att justera sitt beteende i framtiden.

3.4 Djup-förstärkningsinlärning

Djup-förstärkningsinlärning innefattar förstärkningsinlärningsalgoritmer som använder ett neuronnät som modell. Neuronnätet används för att approximera policyfunktionen π och för att utifrån ett tillstånd värdera olika handlingar. Utifrån dessa värderingar kan en handling sedan väljas.

Värderingarna neuronnät producerar kallas – liksom värdena i en Q-tabell – för Q-värden. Q-värden är de värderingar neuronnätet lägger på olika handlingar. Dessa värden erhålls vanligtvis från utgångslagret på neuronnätet och består av en vektor med lika många värden som finns i $\mathcal{A}$. Utifrån Q-värdena kan agenten bestämma en policyfunktion π genom att välja den handlingen som har högst Q-värde givet det nuvarande tillståndet.



Figur 3.4: En modell som tar emot tillstånd och producerar ett motsvarande Q-värde för dem

Med exemplet av ett luffarschack skulle Q-värdena kunna representera varje ruta på spelplanen och hur neuronnätet värderar att placera en markör där (se figur 3.4). När en handling tagits får algoritmen en belöning och kan träna neuronnätet med den informationen.

3.4.1 Förlustvärde

När neuronnät tränas görs det oftast i relation till ett beräknat förlustvärde. Principen med ett förlustvärde är att den ska representera avståndet mellan det nuvarande beteendet och ett bättre – eller optimalt – beteende. Med Adam algoritmen för att

optimera ett neuronnät är justeringen proportionell till förlustvärdet [21].

Det finns flera formler för att beräkna ett förlustvärde, och vilken formel som är mest lämpad beror på problemet som algoritmen ska lösa. En vanlig formel för att beräkna ett förlustvärde i en djup-RL-algoritm är medelkvadratfelet (*eng.* mean square error, MSE).

$$\texttt{MSE} = \frac{1}{n}\sum_{i=0}^{n}(\texttt{error}_i)^2 = \frac{1}{n}\sum_{i=0}^{n}(Q_i \ - \ Q_i^+)^2 \tag{3.7}$$

Ekvation 3.7 beskriver hur medelkvadratfelet beräknas. Q_i representerar ett Q-värde från en mängd sådana, medan Q_i^+ representerar ett värde från en bättre eller optimal mängd Q-värden. Medelkvadratfelet producerar alltså ett mått på hur långt samtliga Q-värden är ifrån sitt motsvarande optimala värdet. Hur Q_i och Q_i^+ väljs varierar från problem till problem.

3.4.2 Erfarenhetsbuffer

Ett vanligt problem som djup-RL-algoritmer kan stöta på är att neuronnätet överanpassar sig. I verkliga problem är den datamängd som algoritmen tränar på inte fullt representativt för problemet i sin helhet. Detta leder ofta till att modellen endast lär sig att lösa problemet för tillstånd den observerar under träningen. Sedan lyckas modellen inte lösa problemet för tillstånd den ej observerat tidigare.

Det finns dock metoder för att försöka motverka att neuronnätet överanpassar sig. En av dessa metoder går ut på att använda en erfarenhetsbuffer $\mathcal{B}$ [14]. Bufferten är en mängd av tupler som beskriver övergångar bestående av ett tillstånd, en handling, nästa tillstånd och den resulterade belöningen.

$$\Big[s, a, s^+, R_a(s, s^+)\Big]$$

Varje gång agenten tar en handling sparas resultaten från tillståndsövergången. När neuronnätet ska optimeras tas ett slumpmässigt urval från $\mathcal{B}$ och används för att beräkna Q_i och Q_i^+. Ett slumpmässigt urval av flera övergångar motverkar att modellen anpassar sig för mycket till individuella tillstånd. Storleken på $\mathcal{B}$ och hur många värden som deltar i det slumpmässiga urvalet är båda hyperparametrar.

3.4.3 Policynät och målnät

Ett koncept kallat målnät kan användas inom djupa-RL-algoritmer. Det är ett sätt att separera lärandet från valet av handlingar. När agenten har tagit en handling kan ett förlustvärde räknas ut och algoritmen kan optimera neuronnätet. Under optimeringen ska Q_i och Q_i^+ beräknas (se 3.4.1). Det kan vara förmånligt att använda separata modeller för att räkna ut dem. Policynätet π_1 används för att räkna ut Q_i och målnätet π_2 används för att räkna ut Q_i^+.

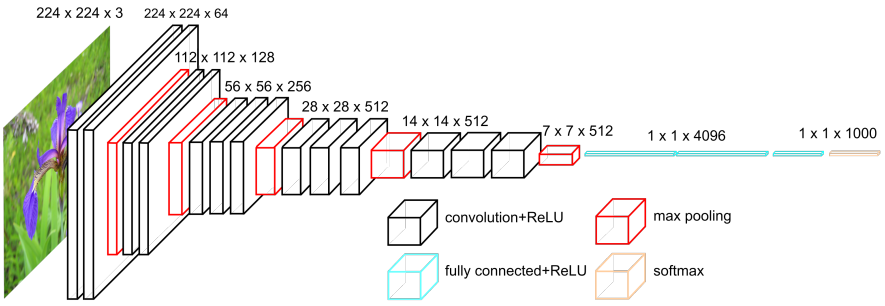
Målnätet är menat att vara mer stabilt än policynätet. Om samma nät användas för beräkningen av Q_i och Q_i^+ löper det större risk att algoritmen blir instabil [25].

θ_1 och θ_2 noterar parametrarna för π_1 och π_2. Parametrarna för π_2 uppdateras proportionellt med hyperparametern τ över tid. Detta beskrivs av tilldelning 3.8 och kallas för en mjuk uppdatering.

$$\theta_2 \leftarrow \tau \cdot \theta_1 + (1 - \tau) \cdot \theta_2 \tag{3.8}$$

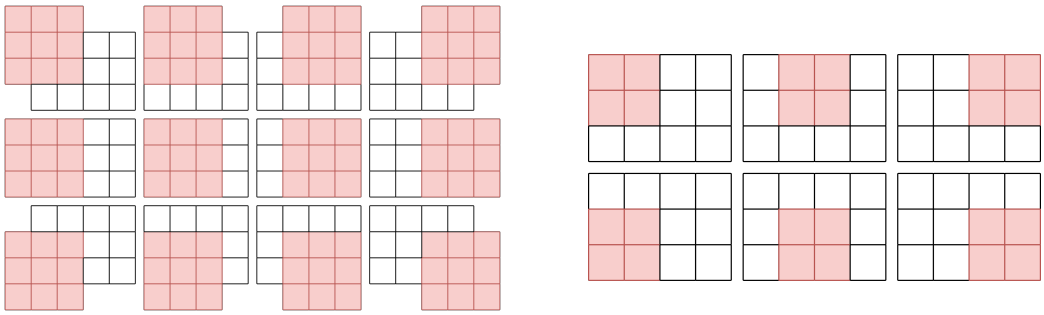
3.5 Faltningsnät

En specifik klass av neuronnät är faltningsnät (*eng.* convolutional neural networks, CNN). Dessa nät tillämpas vanligen för visuell analys av data och kan användas för att behandla bilder, sekvenser av bilder eller ljudfiler. Faltningsnät har exempelvis visat stor potential inom medicinsk bildanalys och används som hjälpmedel vid diagnostisering av sjukdomar [26]. En vanlig CNN-arkitektur visas i figur 3.5.



Figur 3.5: Faltningsnätet VGG-16 [27]

Faltningsnät genomför visuell analys av bilder genom att dela upp tillgänglig data i mindre sektioner och systematiskt utföra steg (*eng.* stride) över dem (se figur 3.6). Stegen avser att analysera delar av bilden genom att utföra olika operationer på de värden som observeras vid ett givet steg.



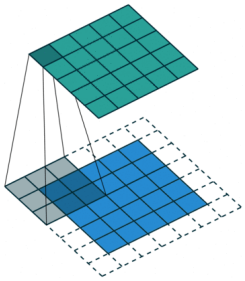
(a) Steg i ett faltningslager **(b)** Steg i ett samplingslager

Figur 3.6: Visualisering av steg i ett faltningsnät

Steg används av faltningsnät i två olika typer av lager, faltningslager och samplingslager.

3.5.1 Faltningslager

Faltningslager är ett av byggblocken i ett faltningsnät. I ett sådant lager används en kärna (*eng.* kernel) för att under stegen utföra en faltning mellan indata och kärnan. Kärnan består alltså av ett antal numeriska värden i form av en matris. Dessa värden används som vikter för att producera resultatet av faltningen. Resultatet erhålls genom att beräkna summan av ett värde och ett antal av dess grannar viktade med värdena i kärnan.



Figur 3.7: Faltning mellan indata och kärna

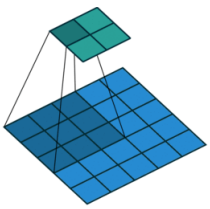
Faltningslager används för särdragsextraktion (*eng.* feature extraction) [28]. Dess funktionalitet är att extrahera egenskaper från indata som kan användas för att identifiera bilderna som analyseras. Vilka egenskaper som extraheras varierar beroende på val av kärna. Exempelvis kan vissa kärnor användas för att utföra kantdetektion på bilder.

Vid tillämpning av faltningslager sker ingen dimensionsreducering. En faltning mellan indata och kärna sker alltså för samtliga värden. Värden vid kanterna saknar således intilliggande värden som ska viktas av kärnan, vilket exempelvis visas i figur 3.7. Ett vanligt sätt att hantera dessa fall är att utöka värdena som saknas med det närmaste värdet.

3.5.2 Samplingslager

Samplingslager är det andra byggblocket som faltningsnät består av. Dessa lager används för att utföra dimensionsreducering på indata vilket underlättar särdragsextraktionen i senare instanser av faltningslager.

Dimensionsreduceringen utförs för varje steg som tas och omvandlar flera värden till ett enda värde (se figur 3.8). Detta kan ske genom att exempelvis välja det största värdet (*eng.* maxpooling) eller medelvärdet (*eng.* averagepooling).



Figur 3.8: Poolingoperation applicerad på indata för dimensionsreducering

4
Teknisk bakgrund

Detta avsnitt introducerar den tekniska grunden som arbetet baseras på.

4.1 Google Analytics och BigQuery

Google Analytics [2] är en av de mest använda tjänsterna för att samla in och sammanställa användardata på webbplatser. När tjänsten till exempel integreras med en internetbutik kan den spara undan bland annat vilka produkter och bilder kunder klickat på. En kund kan identifieras med ett konto på webbplatsen eller ett anonymt identifieringsnummer, information som också kan sparas undan. Dessutom är det möjligt att få ut en grov uppskattning av kundens plats, som går att översätta till land och/eller marknad. Datamängden som samlas in lagras på en annan Google-tjänst — BigQuery.

Googles BigQuery tillhandahåller möjligheten att sammanställa och ladda ner data från bland annat Analytics. Datamängden är samlad i SQL-databaser [29] med stöd för att exekvera SQL-frågor. Frågorna kan aggregera och sortera data för att få fram en önskvärd formatering och filtrering. Gällande en internetbutik skulle exempelvis en filtrering kunna göras för att få fram en sorterad lista med de mest klickade produkterna i Sverige under perioden januari – mars 2023.

När en sammanställning av data är klar – möjligtvis med hjälp av SQL-frågor – kan den laddas ner. Det går att ladda ner i flera format, däribland CSV [30] och JSON [31]. Båda formaten är textbaserade och innehåller samma information. CSV lägger upp data i en tabell såsom den framställts i BigQuery. JSON är istället uppbyggt av nyckel-värdepar. En egenskap hos JSON som kan göra att det är att föredra över CSV är att kolumner med mycket information kan formateras som nyckel-värdepar – som värdet hos ett annat nyckel-värdepar.

4.2 PyTorch och Hugging Face

PyTorch är ett projekt med öppen källkod som erbjuder kodbibliotek för att arbeta med maskininlärning [32]. Det finns färdiga implementationer av neuronnät och verktyg för att bygga upp algoritmer. Projektet är ursprungligen utvecklat i C++ [33] och erbjuder där ett kodbibliotek kallat libtorch [34]. Den primära utvecklingsmiljön för PyTorch är dock Python [35]. För Python har PyTorch ett bibliotek som

överför källkoden skriven i det mer maskinnära språket C++.

För att uppnå ökad prestanda kan PyTorch använda sig av hårdvaruacceleration. Moderna grafikprocessorer har processorkärnor som kan utnyttjas för att utföra beräkningar med hög parallellitet. Att träna neuronnät är ett problem som är bra lämpat för starkt parallella beräkningar. Tiden det tar att träna neuronnät kan därför reduceras markant med hårdvaruacceleration. Grafikprocessorer från båda av de stora tillverkarna – AMD och Nvidia – kan användas för hårdvaruacceleration av PyTorch, men stödet för AMD är sekundärt till Nvidia.

Det finns verktyg för att ladda ner och använda förtränade modeller. En plattform som erbjuder denna funktionalitet för PyTorch är Hugging Face [36]. Plattformen erbjuder många förtränade modeller, däribland flera typer av neuronnät samt faltningsnät.

4.3 Vertex AI

Vertex AI [37] är en Google-tjänst för att träna maskininlärningsalgoritmer på deras hårdvara. Tillgängligt på maskinerna som används finns Nvidia processorer för hårdvaruacceleration. Tjänsten stödjer användningen av egna Docker containers [38] som körs och där den tränade modellen sparas.

5
Genomförande

Detta avsnitt presenterar hur arbetet genomfördes. Hanteringen av data för att skapa datapunkter beskrivs. Dessutom presenteras algoritmerna och modellerna som skapats under arbetets gång.

5.1 Datahantering

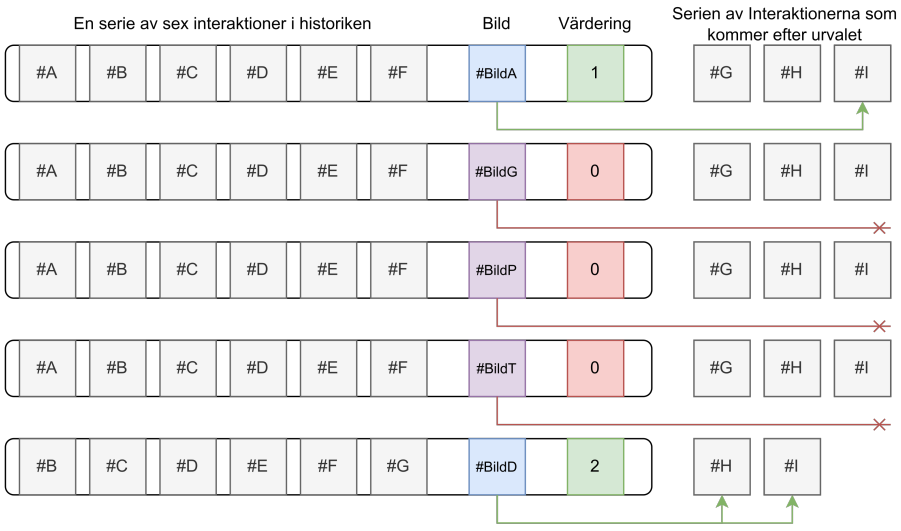
Data som har använts för att träna algoritmerna kommer från anonymiserad användardata. Den inkluderar olika typer av interaktioner från Ikeas hemsida och är insamlad av Google Analytics och lagrad på BigQuery.

Under arbetet har mängden data varit begränsad. Datamängden som använts består av användardata från Sverige och är insamlad under en begränsad tidsperiod.

5.1.1 Generering av datapunkter

En datapunkt innehåller ett antal av de senaste interaktionerna i historiken och en möjlig rekommendation, en bild. För att förbereda datapunkter som algoritmerna kan träna på har varianter av algoritm 5.1 används.

Figur 5.1 visar ett visuellt exempel på skapandet av datapunkter.



Figur 5.1: Skapandet av en serie av datapunkter

För varje interaktion i den insamlade datamängden skapar algoritmen ett antal datapunkter. Bilden får en värdering som ska representera hur bra den rekommendationen är. Värdering beräknas som antalet produkter bilden innehåller som förekommer framåt i historiken, till en gräns. Algoritmen för att skapa datapunkter tar ut ett antal positiva bilder för varje del i historiken. För varje positivt exempel skapas också ett större antal negativa exempel, bilder med en värdering av noll.

Algorithm 5.1 Algoritm för generering av datapunkter

$\mathcal{D} \leftarrow \{\ \}$ ▷ Datapunkter
$\mathcal{T} \leftarrow \{T_0 = [\texttt{item}_1, \texttt{item}_2, \ldots \texttt{item}_{u_0} =], \ldots T_w = [\ldots]\}$ ▷ Användarinteraktioner
$\mathcal{I} \leftarrow \{I_0 = [\texttt{item}_1, \texttt{item}_2, \ldots \texttt{item}_{v_0} =], \ldots I_x = [\ldots]\}$ ▷ Bilder med produkter
for each element T in $\mathcal{T}$ **do**
  $\mathcal{U} \leftarrow \{\ \}$ ▷ Användarprofilen
  for $0 \ldots$ `size` **do**
    push 0 to the top of $\mathcal{U}$
  end for
  for each element `item` in T **do**
    push $\texttt{item}_i$ to the top of $\mathcal{U}$
    pop from the bottom of $\mathcal{U}$
    $P \leftarrow \{\ \}$ ▷ Positiva exempel
    for each element I_j in $\mathcal{I}$ **do**
      $r \leftarrow |T[i : i + \texttt{forward}] \cup I|$
      if $r > 0$ **then**
        $P \leftarrow P \cup \{\{\texttt{image} = \text{j}, \texttt{profile} = \mathcal{U}\}\}$
      end if
    end for
    $P_{final} \leftarrow$ `positive` random elements from P
    $\mathcal{D} \leftarrow \mathcal{D} \cup P_{final}$
    for $0 \ldots$ `negative` **do**
      for each element `p` in P_{final} **do**
        $rand \leftarrow random(0 \ldots |\mathcal{I}|)$
        if $\mathcal{I}[rand]$ not in P **then**
          $\mathcal{D} \leftarrow \mathcal{D} \cup \{\{\texttt{image} = rand, \texttt{profile} = \mathcal{U}\}\}$
        end if
      end for
    end for
  end for
end for

Antalet olika produkter som förekommer i den insamlade datamängden är i storleksordningen av $3 \cdot 10^4$. Frekvensen av produkterna som förekommer följer en invers relation. Det betyder att en relativt liten delmängd av produkterna utgör en större mängd av användarinteraktionerna. Antalet bilder som kan rekommenderar är ungefär 10^4. Inte alla produkter förekommer bland bilderna, och blir därför omöjliga att rekommendera för. Populära produkter brukar ha fler bilder associerade med dem än mindre populära produkter.

5.2 Konstruktion av algoritmer och modeller

Som angavs i målet skulle flera olika algoritmer och modeller undersökas. De flesta algoritmerna bygger på varann. Hur algoritmerna konstruerades kommer att beskrivas här.

Algorithm 5.2 Algoritm för träning med djup-förstärkningsinlärning

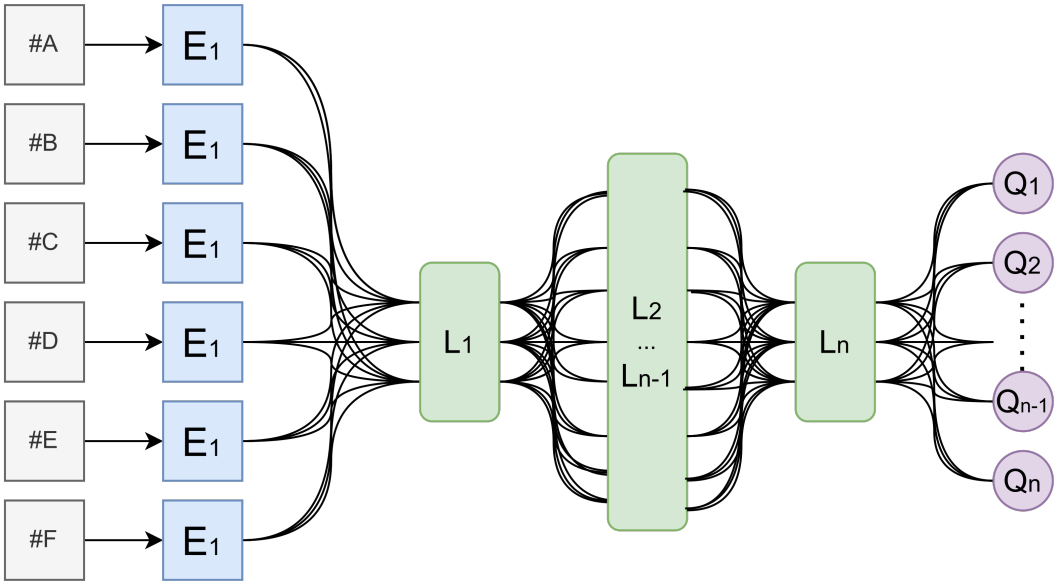
$\mathcal{D} \leftarrow$ dataset
$\mathcal{B} \leftarrow$ empty experience buffer with `buffer_size`
$\theta_1 \leftarrow$ randomly initialized weights
$\pi_1 \leftarrow$ model with θ_1
$\theta_2 \leftarrow \theta_1$
$\pi_2 \leftarrow$ model with θ_2
Optimizer $\leftarrow$ AdamW optimizer with learning rate `LR` for θ_1
for $0 \ldots$ `epoch` **do**
    done $\leftarrow$ false
    t $\leftarrow 0$
    while not done **do**
        $s_t \leftarrow \mathcal{D}[t]$
        if $\text{random}_{\mathbb{R}}(0 \ldots 1) > \varepsilon_t = \varepsilon_{end} + (\varepsilon_{start} - \varepsilon_{end}) \cdot e^{-t/\varepsilon_{decay}}$ **then**
            $a \leftarrow \max_{a_i} \pi_1(s, a_i)$
        else
            $a \leftarrow \text{random}_{\mathbb{Z}}(0 \ldots |\mathcal{A}|)$
        end if
        $s_{t+1} \leftarrow \mathcal{D}$[t+1] or null
        r $\leftarrow R(s_t, s_{t+1})$
        push $[s_t, a, s_{t+1}, r]$ to $\mathcal{B}$
        if $|\mathcal{B}| >$ `batch_size` **then**
            batch $\leftarrow$ `batch_size` random elements from $\mathcal{B}$
            $S_{batch} \leftarrow$ all states in batch
            $A_{batch} \leftarrow$ all actions in batch
            $R_{batch} \leftarrow$ all rewards in batch
            $N_{batch} \leftarrow$ all next states in batch
            $Q \leftarrow Q_{\pi_1}(S_{batch}, A_{batch})$
            $V \leftarrow V_{\pi_2}(N_{batch}) \cdot \gamma + R_{batch}$
            loss $\leftarrow \text{Loss}(Q, V)$
            run Optimizer
            clamp θ_1 to $[-1, 1]$
        end if
        $\theta_2 \leftarrow \tau \cdot \theta_1 + (1 - \tau) \cdot \theta_2$
        done $\leftarrow s_{t+1} ==$ null
        t $\leftarrow$ t+1
    end while
end for

Grunden till alla algoritmer som använts under arbetet har varit djup-förstärkningsinlärning. De har följt samma grundläggande mönster där de största skillnaderna har varit i hur modellen struktureras. I algoritm 5.2 beskrivs det grundläggande mönstret för hur algoritmerna tränas [42]. AdamW – en variant av Adam algoritmen – har använts för att optimera modellens parametrar [43].

5.2.1 Kategorisering

Första modellen som konstruerades under arbetet värderar – baserat på tidigare användarinteraktioner – vilka rekommendationer som är bäst bland alla möjliga. Denna princip skiljer sig från övriga modeller senare i arbetet. På grund av att denna modell skiljer sig från resten använder den sig inte av algoritm 5.1 för att generera datapunkter.

Modellen tar som indata ett antal tidigare interaktioner för en användares historik. Med den informationen ska algoritmen sedan välja vilka rekommendationerna som är bäst från handlingsrummet. Handlingsrummet är lika stort som antalet möjliga rekommendationer (se figur 5.2).



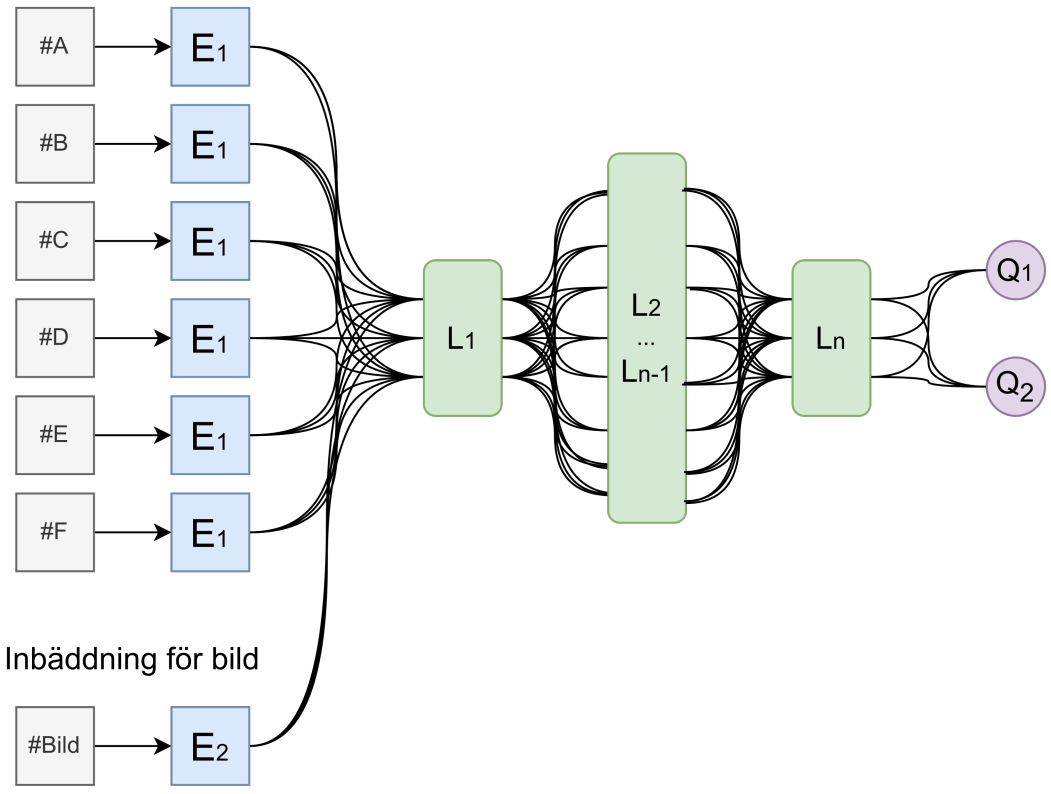
Figur 5.2: Modell som tillämpas vid kategorisering av bilder för att skapa rekommendationer

Modellen som visas i figur 5.2 använder bara ett numeriskt värde för att representera en tidigare interaktion – ett heltal som representerar produkt-id. Denna information ges till inbäddningslager E_1 som tränas tillsammans med modellen för att ge den kategoriska datamängden inbördes betydelse. För att se om det förbättrade prick-säkerheten skapades en variant av modellen som också tar hänsyn till vilken typ av interaktion det var – till exempel klick eller köp.

5.2.2 Binär kategorisering

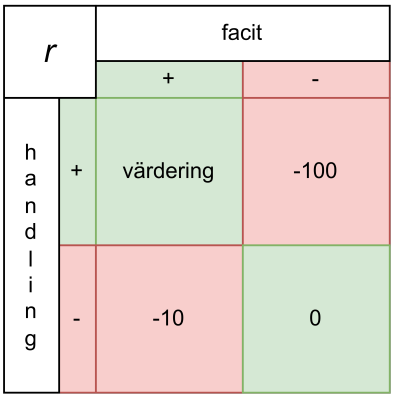
Den andra modellen som konstruerades under arbetet avser att värdera om en rekommendation är bra eller dålig beroende på ett antal tidigare interaktioner. Den princip som modellen utgår från är grunden till samtliga modeller som konstruerades senare i arbetet.

Datapunkterna som används för att träna denna modell är genererade med algoritm 5.1. Den tidigare historiken och bilden – allt som kategorisk data – används som indata. Inbäddningslager som tränas med modellen används för att ge mängden indata en inbördes ordning. Olika inbäddningslager används för produkter och bilder. Inbäddningsvektorn som de två lagerna skapar behöver inte vara samma storlek. Som utdata ger modellen en binär bedömning om bilden är en bra rekommendation eller ej (se figur 5.3). Modellen blir jämförelsevis enkel då handlingsrummet endast består av två handlingar. Förhoppningen med en enklare modell är att den ska vara lättare att träna.



Figur 5.3: Modell som tillämpas vid binär kategorisering av bilder för att skapa rekommendationer

När algoritmen tagit en handling och bedömt om en bild är en bra rekommendation eller ej får den en belöning. Belöningen beräknas utifrån de fyra olika möjliga utfall som finns i en binär kategorisering (se figur 5.4). Positiva eller neutrala belöningar ges när algoritmen korrekt identifierat en rekommendation som bra eller dålig. Om algoritmen misslyckas ges en negativ belöning.



Figur 5.4: Belöningsfunktion med exempelvärden

Med ett binärt val med avseende till en möjlig rekommendation förloras möjligheten att hitta de bästa rekommendationerna med avseende till alla möjliga. I praktiken skulle algoritmen köras flera gånger med ett antal möjliga rekommendationer för att välja ut ett antal som anses bra.

5.2.3 Binär kategorisering med faltningsnät

Ytterligare en modell utformades för att pröva en variant av den som presenterats i 5.2.2. Denna modell skiljer sig från den tidigare vid ingångslagret då den hanterar indata på ett annat sätt.

Hittills har ingångslagret bestått av en konkatenering av värdena som inbäddningslagret E_1 och E_2 producerat under inlärningen. Istället för att utnyttja värden från inbäddningslagret E_1 har faltningsnätet Inception v4 [39] använts för att producera värden för produkterna till ingångslagret.

Inception-v4 är ett förtränat faltningsnät. Värdena på nätets utgångslager motsvarar en gissning angående vad bilden som faltningsnätet ges innehåller. De flesta bilderna får några värden nära 1 på utgångslagret, och resten är värden nära 0. Dessa värden motsvarar sannolikheter för olika kategorier som faltningsnätet tror är representerade i bilden den får som indata. Inception-v4 har 1000 sådana kategorier i sitt utgångslager. Hugging Face (se 4.2) användes för att ladda ned och använda modellen [41].

Eftersom antalet värden vid ingångslagret ökar kraftigt med denna modifiering har storleken på neuronnätets egna lager också ökats. Utgångslagret bevarar sin storlek eftersom formuleringen av problemet – en förenklad binär kategorisering – fortfarande är likadan.

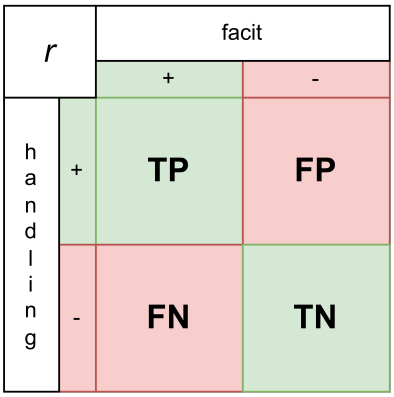
5.3 Träning av modeller

Utvecklingsmiljön där algoritmerna och modellerna implementerades bygger främst på PyTorch. Projektets bibliotek stödjer både Microsoft Windows och Linux – båda har använts under arbetets gång. Utvecklingen med PyTorch har skett främst med deras Python-bibliotek, där den senaste stabila utgåvan av Python och kodbiblioteket har använts – 3.10 respektive 2.0. En implementation med C++ och libtorch gjordes också men användes inte för att producera några resultat som presenteras. Skillnaderna från plattform till plattform är oväsentliga på den nivå utvecklingen skett på.

Träningen av algoritmerna skedde vanligtvis under en perioder på 24 till 72 timmar. Träningstiden varierar beroende på använd hårdvara (se antal tränade episoder per modell i 6 i stället). Hårdvaruacceleration har utnyttjats under arbetet för att uppnå bättre prestanda. För att träna algoritmen under en längre period med mer data har Vertex AI använts.

5.4 Mått för utvärdering av prestanda

För att producera mått för de utvecklade algoritmerna och modellerna och kunna utvärdera dem i de senare fallen – där problemformuleringen är en förenklad binär kategorisering – används en förvirringsmatris (*eng.* confusion matrix). En binär kategorisering har fyra möjliga utfall och det totala antalet av varje utfall visas i en förvirringsmatris.



Figur 5.5: Förvirringsmatris med dess fyra möjliga utfall: sann positiv (*eng.* true positive (TP)), sann negativ (*eng.* true negative (TN)), falsk positiv (*eng.* false positive (FP)) och falsk negativ(*eng.* false negative (FN))

Måttet som har använts för att utvärdera de olika modellerna är pricksäkerhet [40]. För samtliga modeller förutom den första (se 5.2.1) beräknades pricksäkerhet som antalet korrekta bedömningar delat på antalet datapunkter (se ekvation 5.1).

$$\text{pricksäkerhet} = \frac{\texttt{TP + TN}}{\texttt{TP + FP + FN + TN}} \tag{5.1}$$

Från resultaten av validering och validering (se 5.4.1) går det också att beräkna hur bra modellerna är på att korrekt identifiera bra rekommendationer. Detta mått kallas för sensitivitet [40] och ges av ekvation 5.2.

$$\text{sensitivitet} = \frac{\texttt{TP}}{\texttt{TP + FN}} \tag{5.2}$$

Under träning och validering beräknades aldrig något mått på hur bra modellerna var på att identifiera dåliga rekommendationer. Detta mått kallas för specificitet [40]. Däremot var relationen mellan antalet negativa och positiva rekommendationer känd eftersom den väljs som en hyperparameter. Relationen ges av ekvation 5.3

$$r = \frac{\texttt{TN + FP}}{\texttt{TP + FN}} \tag{5.3}$$

En uppskattning av specificiteten kunde således beräknas. Uppskattningen ges av ekvation 5.4. En viss felmarginal finns i måttet varför den referas till som en uppskattning (se 5.4.1).

$$\text{specificitet} = \frac{\texttt{TN}}{\texttt{TN + FP}} = \frac{\text{pricksäkerhet} \cdot (1 + r) - \text{sensitivitet}}{r} \tag{5.4}$$

5.4.1 Validering av modeller

Validering användes för att utvärdera modellerna efter tränigen. En mängd datapunkter plockas ut ur datamängden algoritmen tränar på och används sedan för att beräkna måttet. Validering kan producera mer pålitliga mått eftersom det är ny data algoritmen aldrig har sett förr som den testas på. Pricksäkerheten blir mer representativt för en verklig miljö där unika datapunkter nästan alltid är att förvänta sig.

Eftersom mängden datapunkter delas upp i en träningsmängd och en valideringsmängd är exakta värdet av `TN + FP` och `TP + FN` inte känt. Valideringsmängden är slumpmässigt utplockad ur ursprungliga mängden av datapunkter. Summorna kommer därför ungefär att behålla samma förhållande. Den utplockade valideringsmängden används inte för att träna modellen.

6
Resultat

I detta avsnitt presenteras resultaten från arbetet. Detaljer kring algoritmerna och modellerna såsom värden på hyperparametrar presenteras också här. Modellerna har tränats för att uppnå pricksäkerhet – vilket är synonymt med dess prestanda. Se 5.4 för hur pricksäkerhet är beräknat. Samtliga resultat har erhållits med följande gemensamma värden för vissa av hyperparametrarna som visas i tabell 6.1.

Tabell 6.1: Värden för hyperparametrar som inte varierats under arbetet [43]

LR	0.0001
ε_{start}	0.9
ε_{end}	0.05
ε_{delay}	1000

Dessutom har förlustvärden beräknats med medelkvadratfelet (se 3.4.1) i samtliga fall. De gömda lagren i alla modeller är anslutna tätt med varandra och använder sig av ReLU (se ekvation 3.1) som aktiveringsfunktion.

6.1 Kategorisering

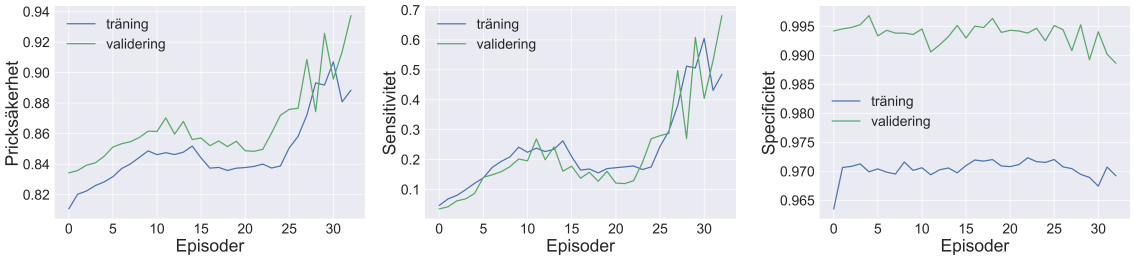
Enbart slumpmässiga handlingar med modellen för kategorisering (se 5.2.1) gav en pricksäkerhet för baslinjen på ungefär 3.5%. Efter att algoritmen tränat visade den prestanda jämförbart med enbart slumpmässiga handlingar eller marginellt bättre. Olika värden på hyperparametrarna visade försumbara skillnader. Skillnaden i prestanda var också försumbar om algoritmen använde sig av bara produkt-id eller inkluderade typen av de tidigare interaktionerna.

6.2 Binär kategorisering

I figur 6.1 och 6.2 visas resultatet från två körningar med modellen för förenklad kategorisering (se 5.2.2). Datapunkter är skapade enligt algoritm 5.1. Ordningen av datapunkterna är slumpmässig. Inbäddningsvektorerna för produkterna och bilderna är av storlek 8. Storleken på de gömda lagren är 256–512–512–256. Belöningsfunktionen som utnyttjas är den som visas i figur 5.4. Övriga hyperparametrar har samma värde för båda fallen visas i tabell 6.2.

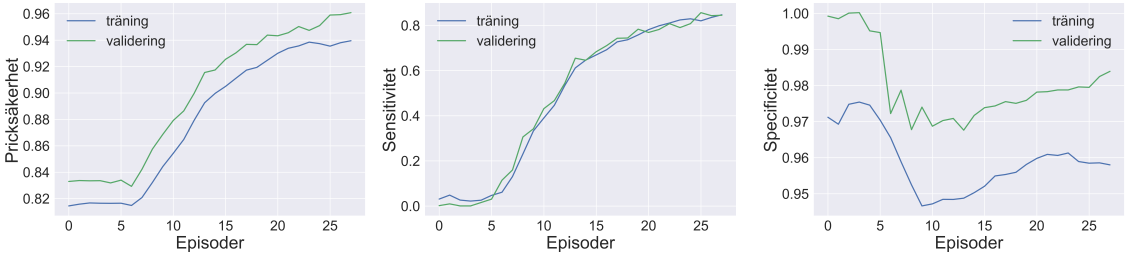
Tabell 6.2: Värden för hyperparametrar som används i resultaten från figur 6.1 och 6.2

τ	0.05
γ	0.99
`batch_size`	256



(a) Pricksäkerhet per epok **(b)** Sensitivitet per epok **(c)** Specificitet per epok

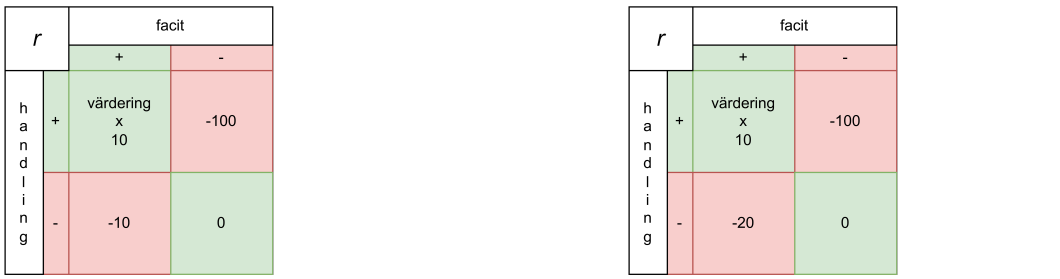
Figur 6.1: Träning och validering på 10^5 datapunkter med följande värden för algoritm 5.1: `size` $= 3$, `forward` $= 3$, `positive` $= 5$ och `negative` $= r = 5$



(a) Pricksäkerhet per epok **(b)** Sensitivitet per epok **(c)** Specificitet per epok

Figur 6.2: Träning och validering på 10^5 datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= \infty$, `positive` $= 5$ och `negative` $= r = 5$

För de resterande två modellerna som testades inom problemformuleringen för binär kategorisering användes variationer av belöningsfunktionen. Dessa variationer visas i figur 6.3.



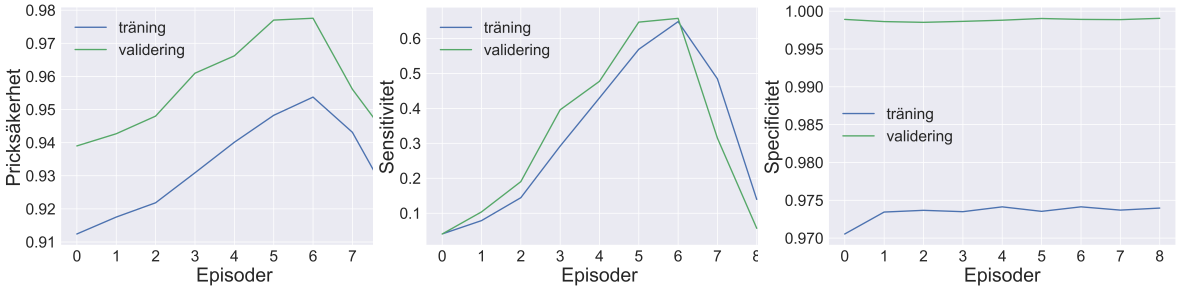
(a) Belöningen av `TP` ökas **(b)** Absoluta belöningen av `TP` och `FN` ökas

Figur 6.3: Variationer av belöningsfunktionen

I figur 6.4 och 6.5 visas resultaten med några modifikationer applicerade på modellen och skapandet av datapunkterna. Inbäddningsvektorn för bildernas storlek ökades till 16. Belöningsfunktionen för resultaten som visas i figur 6.4 är den som presenterades i figur 6.3a, medan resultaten i figur 6.5 tillämpar belöningsfunktionen från figur 6.3b. Värden för övriga hyperparametrar som modifierats visas i tabell 6.3.

Tabell 6.3: Värden för hyperparametrar som används i resultaten från figur 6.1 och 6.2

τ	0.025
γ	0.99
`batch_size`	512



(a) Pricksäkerhet per epok **(b)** Sensitivitet per epok **(c)** Specificitet per epok

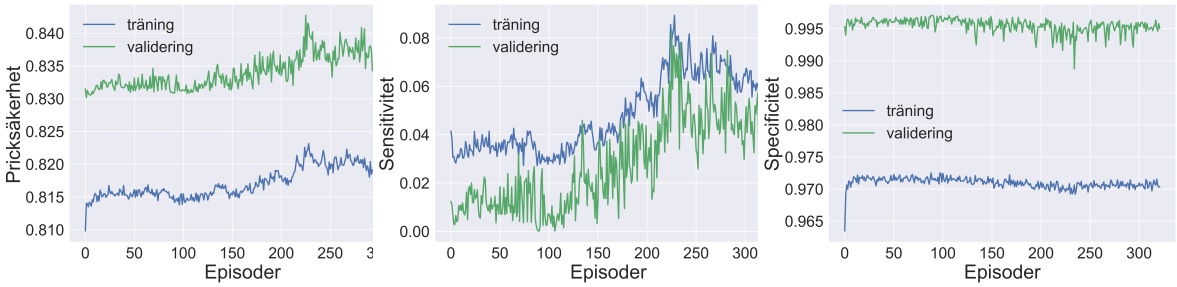
Figur 6.4: Träning och validering på $2.5 \cdot 10^5$ datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= 6$, `positive` $= 5$ och `negative` $= r = 15$

(a) Pricksäkerhet per epok **(b)** Sensitivitet per epok **(c)** Specificitet per epok

Figur 6.5: Träning och validering på $2.5 \cdot 10^5$ datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= 6$, `positive` $= 5$ och `negative` $= r = 15$

6.3 Binär kategorisering med faltningsnät

Den första modellen som testades tillsammans med ett faltningsnät konkatenerade utvärdet från faltningsnätet för respektive produkt tillsammans med utvärdet från inbäddningslagret för inspirationsbilden, vilket sedan blev invärdet till modellen. Belöningsfunktionen som tillämpades var den som visas i figur 6.3a. Storleken på de gömda lagren ökades till 512–1024–1024–512 och samtliga modeller som testades under denna del av arbetet tillämpade hyperparametrarna vars värde visas i tabell 6.3. Resultaten som erhölls visas i figur 6.6.



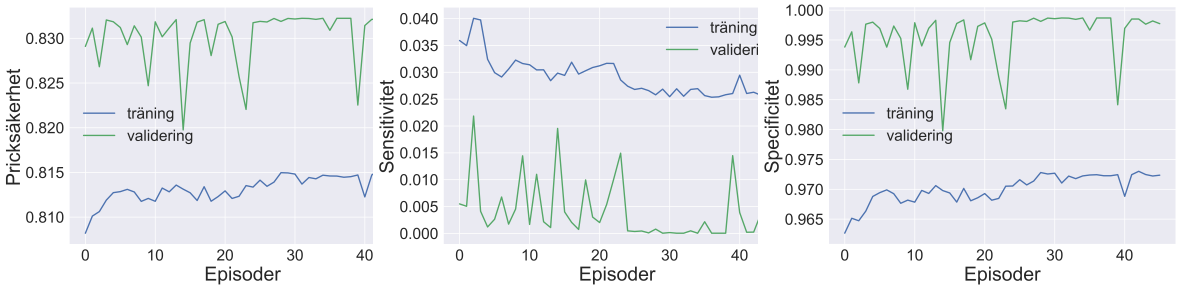
(a) Pricksäkerhet per epok **(b)** Sensitivitet per epok **(c)** Specificitet per epok

Figur 6.6: Träning och validering på $2.5 \cdot 10^5$ datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= 6$, `positive` $= 5$ och `negative` $= r = 5$

En variant av denna modell testades genom att beräkna medelvärdet respektive summan av utvärdena från faltningsnätet i stället för att konkatenera dessa. Konkateneringen med utvärdet från inbäddningslagret för inspirationsbilderna förändrades inte. Däremot gavs modellen 10^6 datapunkter med $r = 5$ att träna på. Resultaten som erhölls visas i figur 6.7 och 6.8.

(a) Pricksäkerhet per epok **(b)** Sensitivitet per epok **(c)** Specificitet per epok

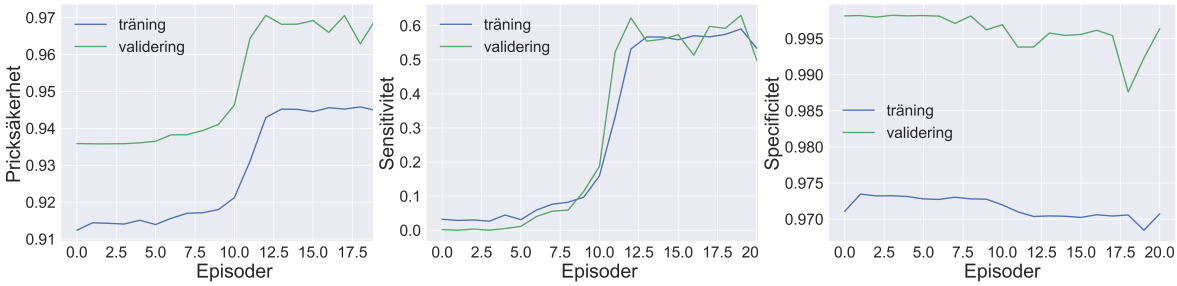
Figur 6.7: Träning och validering på 10^6 datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= 6$, `positive` $= 5$ och `negative` $= r = 5$



(a) Pricksäkerhet per epok **(b)** Sensitivitet per epok **(c)** Specificitet per epok

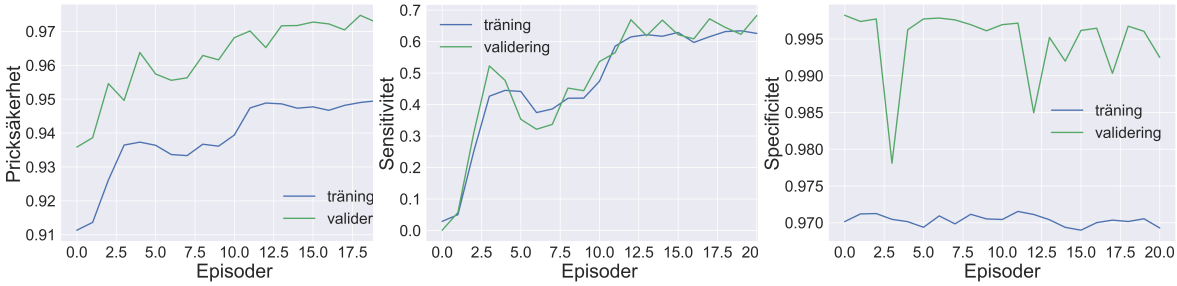
Figur 6.8: Träning och validering på 10^6 datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= 6$, `positive` $= 5$ och `negative` $= r = 5$

Samma modeller testades igen med en datamängd med $r = 15$ och med belönings-funktionen som visas i figur 6.3b. Resultaten som erhölls visas i figur 6.9 och 6.10.



(a) Pricksäkerhet per epok **(b)** Sensitivitet per epok **(c)** Specificitet per epok

Figur 6.9: Träning och validering på 10^6 datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= 6$, `positive` $= 5$ och `negative` $= r = 15$



(a) Pricksäkerhet per epok **(b)** Sensitivitet per epok **(c)** Specificitet per epok

Figur 6.10: Träning och validering på 10^6 datapunkter med följande värden för algoritm 5.1: `size` $= 6$, `forward` $= 6$, `positive` $= 5$ och `negative` $= r = 15$

7
Diskussion

Modellen som skulle markera – bland alla möjliga inspirationsbilder – vilka rekommendationer som var bra visar inga meningsfulla resultat (se 6.1). Pricksäkerheten översteg sällan 3.5% som är baslinjen som används för jämförelse. Det kan bero på flera faktorer. En betydande faktor kan tänkas vara att modellen blev svår att träna. Ett handlingsrum med storleksordningen 10^4 ger agenten ett komplicerat uppdrag. Det är möjligt att modellen skulle producera bättre pricksäkerhet om det fick träna under en längre period med mer data.

Ett problem som förekom under större delar av arbetet var den långa tiden det tog att träna modellerna. Med hårdvaruacceleration tog en epok ungefär tio minuter med 10^5 datapunkter, och tio gånger så länge med 10^6 datapunkter. Under majoriteten av arbetet har det varit opraktiskt att träna modeller under en period längre än ett dygn, vilket begränsar resultaten. Om modellerna skulle tränas för en verklig miljö skulle datamängderna behöva utökas.

Vertex AI användes bara under en kortare period under arbetet. Modellerna blev inte noterbart snabbare att träna på plattformen. Då användning av tjänsten kostar pengar gjordes valet att inte använda den ungefär halvvägs genom arbetet.

Resultaten från modellen med ett förenklat handlingsrum (se 6.2) är mer lovande än dem från första modellen. Med fem negativa exempel till varje positivt exempel är baslinjen för pricksäkerhet $5/6 = 0.833\ldots$. Modellen (se 6.2) uppnår en pricksäkerhet på mellan 88% och 94% beroende på förutsättningar, markant högre än baslinjen. Om modellerna hade tränat under flera epoker och med fler datapunkter hade pricksäkerheten troligtvis ökat ytterligare. Storleken på inbäddningslagerna och de dolda lagren skulle kunna behövas öka för att hantera en större mängd datapunkter.

Det är anmärkningsvärt att trenden för valideringen håller sig tätt intill trenden för träningen. Förväntningen är att valideringen presenterar sämre resultat än träningen för att det är datapunkter modellen inte har fått träna på. En möjlig faktor till beteendet kan vara att datapunkterna var slumpmässigt ordnade. Det gör det sannolikt att modellen har tränat på samma produkter och bilder som den validerar på — bara med annorlunda ordning på dem. Detta trots att tränings- och valideringsmängderna skapades för att undvika detta. Utan slumpmässig ordning på datapunkterna skulle valideringen sannolikt innehålla ett större antal produkter och bilder algoritmen inte tränat på. Sämre validering hade då varit att förvänta sig.

Från resultaten av denna algoritm går det att se att förutsättningarna spelar roll i hur pricksäkerheten utvecklas. Om den tar hänsyn till fler produkter i historiken – både bakåt och framåt – blev utvecklingen snabbare. Det är dock svårt att säga hur relevant historiken framåt är. Vad en användare är intresserad av sex klick framåt kanske inte säger så mycket om vad de är intresserade av i nästa klick [44].

Med ett binärt val är hög sensitivitet och specificitet nödvändig för att motstå prevalensfel (*eng.* base-rate fallacy) [45]. Det är problemet med relationen mellan falska positiva utfall och riktiga positiva utfall. Ta resultaten från modellen som presenterades i figur 6.2 som exempel. Den har enligt sin validering en sensitivitet på 0.845 och en specificitet på 0.983. Om det finns 100 bra rekommendationer bland ett urval på 1000 möjliga skulle den markera 99 rekommendationer som är bra, däribland 15 falskt markerade (se beräkning 7.1). Med resultaten presenterade i figur 6.1 som har - enligt sin validering - en sensitivitet på 0.68 och en specificitet på 0.988. Antalet faktiska positiva markeringar överstiga de falska positiva (se beräkning 7.2).

$$\begin{aligned} \texttt{TP} &= 100 \cdot 0.845 \cdots \approx 84 \\ \texttt{FP} &= (1000 - 100) \cdot (1 - 0.983 \dots) \approx 15 \end{aligned} \tag{7.1}$$

$$\begin{aligned} \texttt{TP} &= 100 \cdot 0.68 \cdots \approx 68 \\ \texttt{FP} &= (1000 - 100) \cdot (1 - 0.988 \dots) \approx 10 \end{aligned} \tag{7.2}$$

Det är svårt att få en siffra på hur många rekommendationer är bra, då det beror på användaren av rekommendationssystemet. Det skulle kunna vara så att en användare bara skulle anmärka ett tiotal rekommendationer som bra. Om det var fallet kan prevalensfelet visa att en stor majoritet av rekommendationerna inte skulle vara intressanta.

De datapunkter som har använts för att träna modellerna är till stor del syntetiska. I datamängden som finns insamlad från Ikeas webbplatser går det att få fram när en kund har klickat på en bildrekommendation. De klicken kan tolkas som kända bra rekommendationer. Mängden av klick på bildrekommendationer är dock begränsat. En kund brukar inte klicka på mer än en bildrekommendation under sitt besök. För att öka mängden data som går att träna på gjordes beslutet att skapa syntetisk data.

Syntetisk data är data som inte kommer från insamling av användardata. I algoritmen som skapar datapunkter används bilder som möjliga rekommendationer utan att de är kända som intressanta för användaren. En kvalificerad gissning görs. Den görs genom att titta fram i användarens historik från en punkt på vilka produkter de kommer att klicka på. En bild anses vara en möjlig bra rekommendation – och något som går att träna på – om den innehåller en av dessa produkter. Datapunkterna har funkat för att få en pricksäkerhet över baslinjen för de flesta modeller utvecklade under arbetet. Huruvida de syntetiska datapunkterna är representativa av verkligheten är svårt att bedöma.

En fördel med syntetisk data är att modellerna inte utgår ifrån att de rekommendationer som är insamlade är de bästa. Om modellerna hade tränat enbart på datapunkter insamlat från Ikeas webbplatser skulle den försöka efterlikna dess beteende. Ett AB-test genomfördes inte under arbetet och ingen jämförelse har kunnat göras mellan det nuvarande systemet Ikea använder och modellerna som tagits fram under detta arbete.

Stabiliteten i resultaten är värt att notera. En stor del av att uppnå stabiliteten var kalibrering av belöningsfunktionen. Betydelsen av de olika belöningarna var till hjälp för att placera in värdena i funktionen. Algoritmen har gjort en bra handling om den korrekt placerar in en möjlig rekommendation som bra eller ej. Den har gjort en dålig handling om den producerar falska positiva eller falskt negativa bedömningar. När den gör en falsk positiv bedömning har algoritmen presenterat något som antagligen inte är intressant för användaren. Om den producerar en falsk negativ bedömning förlorar den möjligheten att presentera något som antagligen är intressant för användaren. Hur dessa händelser skulle belönas för att uppnå en stabil och bra träning för algoritmen var en återkommande fråga under arbetet.

Värdena på belöningarna är bara meningsfulla i relation till varandra. Det viktiga är att värdera utfallen av handlingarna gentemot varandra. Det utfallet som bör värderas högst i ett rekommendationssystem är när en bra rekommendation visas för användaren. En neutral händelse är när en ej intressant rekommendation inte visas. Båda av dessa utfall är positiva. De två övriga utfallen är negativa. Under testningen funkade det bäst att värdera ett falskt positivt utfall lägre än ett falskt negativt. Det går att motivera med att ett falskt positivt utfall tar upp en plats bland rekommendationerna som skulle kunna tas upp av en bättre rekommendation. I förstärkningsinlärning finns möjligheten att introducera negativa belöningar. De har använts i arbetet men de är inte nödvändiga då skillnaden i belöningar är det som har betydelse.

Resultaten som presenteras i figur 6.4 och 6.5 visar hur algoritmen kan bete sig med andra förutsättningar. Med femton negativa exempel för varje positivt istället för fem uppnår de – enligt validering – en specificitet på 0.999 respektive 0.995. Det kan betyda att väldigt få dåliga rekommendationer markeras som bra av algoritmen. Resultaten visar att sensitiviteten når sin topp runt 0.65. Jämfört med resultaten uppmätt i figur 6.1 kommer färre bra rekommendationer markeras och färre dåliga rekommendationer kommer att falskt markeras bra (se beräkningar 7.3 och 7.4).

$$\begin{aligned} \texttt{TP} &= 100 \cdot 0.65 \cdots \approx 65 \\ \texttt{FP} &= (1000 - 100) \cdot (1 - 0.999\ldots) \approx 1 \end{aligned} \tag{7.3}$$

$$\begin{aligned} \texttt{TP} &= 100 \cdot 0.65 \cdots \approx 65 \\ \texttt{FP} &= (1000 - 100) \cdot (1 - 0.995\ldots) \approx 5 \end{aligned} \tag{7.4}$$

I figur 6.4 observeras ett fall i prestanda i de två sista epokerna. Det är svårt att bestämma vad det kan bero på. Denna anomali skulle kunna vara ett exempel på

katastrofal glömska (*eng.* catastrophic forgetting) [46]. Det är ett problem som kan uppstå när en modell tränas på sekvenser av data och glömmer bort det den tidigare har lärt sig när den presenteras med ny data. Ett lägre värde på och andra metoder som kan göra träning mer stabilt skulle kunna användas för att försöka undvika problemet.

De flesta trenderna visar att pricksäkerheten för valideringen är högre jämfört med träningen. Det kan delvis bero på att inget epsilon-girigt beteendet (se 3.3.3) används. Träningen skulle aldrig kunna nå specificitet=1 eller liknande då slumpmässiga handlingar gör det säkert att modellen ibland gör felaktiga bedömningar.

Resultaten som visas i figur 6.6 har en pricksäkerhet vid validering som oscillerar runt den förväntade pricksäkerheten på 3.5% från baslinjen. Modellen visar dessutom mycket låga siffror för sensitiviteten, vilket tyder på att rekommendationssystemet sällan identifierar bilder som anses vara relevanta rekommendationer.

En möjlig anledning till den låga sensitiviteten i resultatet är det nya formatet på indata, som består av en konkatenering av utvärden från faltningsnätet Inception-v4 för varje produkt som användaren har interagerat med vid det givna tillståndet. Antalet värden som modellen får som indata är av storleksordningen 10^4, en markant ökning jämfört med storleksordningen på tidigare indata. Att endast öka storleken på de gömda lagerna i modellen med en faktor 2 var möjligtvis inte tillräckligt.

En alternativ lösning är att minska dimensionerna på modellens indata. Vilken effekt detta har visas i resultaten från figur 6.7 och 6.8. I stället för att konkatenera resultaten från faltningsnätet har det elementvisa medelvärdet respektive summan beräknats i dessa två fall. Resultaten blir inte mycket bättre jämfört med den tidigare modellen. Något noterbart är att måtten för modellen som beräknar medelvärdet från faltningsnätet inte ändras särskilt mycket, medan modellen som beräknar summan visar ett instabilt beteende i sina mått.

Ett stabilare beteende eller ändringar på måtten hade möjligtvis uppnåtts om modellerna fick träna ytterligare eller på mer data. Att jämföra modeller som tränat under olika antal epoker eller inte tränat tillräckligt många epoker är inte heller idealt. Ett jämnare antal episoder för modellerna som testats under detta arbete hade varit önskvärt.

Modellerna testades en sista gång med ny träningsdata med $r = 15$ och och en nya variant av belöningsfunktionen som visas i figur 6.3b. Resultaten från dessa modeller visas i figur 6.9 och 6.10. Dessa blev de bästa resultaten som modellerna med faltningsnät visade. Pricksäkerheten ökade betydligt och översteg baslinjen för $r = 15$ för båda modellerna. Dessutom visades en betydande ökning i sensitiviteten till över 60% under valideringen av båda modellerna, ett beteende som inga av de tidigare modellerna med faltningsnät uppvisat. Dessa resultat erbjuder ytterligare stöd till påståendet om att förutsättningarna spelar stor roll för modellernas prestanda.

Något som möjligtvis hade kunnat öka prestandan av dessa modeller är att ytterligare experimentera med hur modellens indata utformas. Utgångslagret från faltningsnätet Inception-v4 är konstruerat med aktiveringsfunktionen softmax [47], vilket innebär att utvärdena från nätet motsvarar sannolikheter. Varje sannolikhet är fältningsnätets gissning på vad den inmatade bilden innehåller. Att istället plocka bort utgångslagret och använda värdena från det sista gömda lagret hade varit önskvärt att testa men lämnas som förslag till fortsatt arbete på grund av tidsbrist.

7.1 Samhälleliga, etiska och miljömässiga aspekter

Samhället uppmärksammar idag nödvändigheten av ansvarsfull konsumtion för att värna om klimatet. Rekommendationssystem är ofta menade att driva konsumtion. Deras roll i hur stora företag driver sina kunder till konsumtion är känt [48]. I ett samhälle som behöver minska konsumtionen kan rekommendationssystemens roll vara nyttig att diskutera. Om rekommendationssystem är väldigt personliga – bra på att fånga intresse – är det möjligt att det leder till onödiga köp. Framgångar i cirkulär ekonomi skulle kunna sänka nödvändigheten av att markant dra ner på konsumtionen, men samhället är inte där idag.

Rekommendationssystem som är väldigt personliga kan ses inkräkta på personers integritet. De presenterade modellerna i denna uppsats tar maximalt hänsyn till sex interaktioner i en användares historik. Inga detaljer så som vilket land användaren befinner sig i, deras ålder eller kön tas in i bedömningarna. Det är dock möjligt att bra gissningar angående till exempel en användares åldersgrupp och kön skulle kunna göras utifrån deras interaktioner. Ett neuronnät skulle – på en abstrakt nivå – kunna lära sig identifiera karaktäristiska drag hos en användare från deras historik. Denna typ av risker bör beaktas vid tillämpning av användardata för att implementera rekommendationssystem.

Kvalitén på datamängden som ett rekommendationssystem tränar på visar sig också ha betydande konsekvenser för systemets beteende [49]. Det viktigaste är att datamängden som används vid träningen är representativ för datamängden som systemet kommer att tillämpas på när den är färdigtränad. Rekommendationssystemet som implementerats under detta arbete använde endast användardata från Sverige. Hur skulle ett sådant rekommendationssystem prestera om det användes i ett annat land?

Träning av maskininlärningsmodeller på stora datamängder efterfrågar kraftfull hårdvara och längre träningstid. Att utföra högt krävande beräkningar på den bästa hårdvaran är väldigt energikrävande. Energiförsörjningen består i många länder till stor del av miljöfarliga energikällor. Många större företag har som mål att inte använda fossila energikällor, däribland Google [50], men det är en bit bort.

8

Slutsats

Utifrån resultaten från modellen som presenterades i 5.2.1 går det inte att dra några slutsatser gällande huruvida dess princip är bra eller inte.

Modellen som presenterades i 5.2.2 visar betydligt bättre prestanda än baslinjen. Förutsättningarna för algoritmen visar stor skillnad i pricksäkerhet. Om algoritmen tar hänsyn till mer punkter i historiken kan den få en bättre utveckling av pricksäkerhet.

Antalet negativa exempel i datamängden som modellen tränar på påverkar också sensitiviteten och specificiteten hos algoritmen. Specificiteten ökar med mer negativa exempel. Sensitiviteten sjunker, vilket kan bero på skillnaden i antalet negativa exempel. Resultatet visar att rekommendationssystemet är kapabel att uppvisa en sensitivitet på över 50% och en specificitet på nära 100%.

Den sista modellen som presenteras i 5.2.3 visade avslutningsvis resultat med en god pricksäkerhet. Sensitiviteten som uppnåddes var också god men översteg aldrig resultaten från modellen som presenterades i 5.2.2. Det går inte att visa att tillämpningen av faltningsnät ökar prestandan av ett rekommendationssystem jämfört med ett motsvarande system som implementeras utan ett CNN.

9
Framtida arbete

En algoritm som utvärderades men aldrig implementerades under arbetet var en med ett kontinuerligt handlingsrum. Med ett inbäddningslager producerat med ett faltningsnät eller ett som tränat med modellen skulle dessa värden tillsammans med det på utgångslagret kunna placeras i ett vektorrum. Varje användarinteraktion och rekommendation skulle representeras av en vektor som pekar på en plats i rummet. De rekommendationer som ligger närmast den vektorn producerad av neuronnätet i rummet är de som agenten bedömer är bra.

En algoritm med ett kontinuerligt handlingsrum skulle vara märkbart mer komplicerad i sin implementering. Möjliga fördelar med en algoritm med denna princip kan vara att det blir enklare att introducera nya rekommendationer utan att träna om modellen och att färre datapunkter krävs vid träning. Utforskningen av en sådan modell vid tillämpning av förstärkningsinlärning för att implementera rekommendationssystem föreslås som fortsatt arbete.

Dessutom föreslås också att undersöka hur tillämpningen av faltningsnät på inspirationsbilder (alternativt det som systemet ska rekommendera) på samma sätt som på produkterna en användare interagerat med påverkar rekommendationssystemets prestanda. Ytterligare en aspekt som kan undersökas är om det gör någon skillnad att använda det sista gömda lagret på faltningsnätet i stället för utgångslagret för att konstruera all indata till neuronnätet.

Källförteckning

[1] F. Ricci, L. Rokach, B. Shapira, *Introduction to Recommender Systems Handbook*, Springer, 2011.

[2] "Welcome to Google Analytics", *Google*. [Online] Tillgänglig: `https://analytics.google.com/analytics/web/provision/#/provision` Hämtad: 7 mars, 2023.

[3] "Mer idéer och inspiration", *Ikea*. [Online] Tillgänglig: `https://www.ikea.com/se/sv/` Hämtad: 26 maj, 2023

[4] R.F. de Mello, M.A. Ponti, *Machine Learning: A Practical Approach on the Statistical Learning Theory*, 1st ed., Springer, 2018.

[5] H.K. Alanzani, R.H. Alruwaili, "Exploring the role of machine learning in Email Filtering", in *International Conference on Business Analytics for Technology and Security (ICBATS)*, Dubai, United Arab Emirates, 2022. [Online] Tillgänglig: `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9759057`

[6] T. Jamtsho, K. Powdyel, R.K. Powrel, R. Bhujel, K. Muramatsu, "OCR and Speech Recognition System Using Machine Learning", in *Innovations in Power and Advanced Computing Technologies (i-PACT)*, Kuala Lumpur, Malaysia, 2021. [Online] Tillgänglig: `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9697030`

[7] R. Cipolla, S. Battiato, G.M. Farinella, *Machine Learning for Computer Vision*, 1st ed., Springer, 2013.

[8] L. Yang, A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice", *Neurocomputing*, vol. 415, pp. 295-316, 20 november, 2020.

[9] V. Aziz, S. Mitra, X. Chen, "Collaborative Filtering Guided Deep Reinforcement Learning for Sequential Recommendations", in *IEEE International Conference on Big Data*, Osaka, Japan, 2022. [Online] Tillgänglig: `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10020921`

[10] X. Gao, M. Qiu, "Recommendation System Design for Social Media using Reinforcement Learning" in *IEEE 9th International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2022, pp. 1-2. [Online] Tillgänglig: `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9842879`

[11] R.S. Sutton, A.G. Barto, *Reinforcement learning: an introduction*, 2nd ed., Cambridge, MA: The MIT Press, 2018.

[12] F.J. Fan, Y. Shi, "Effects of data quality and quantity on deep learning for protein-ligand binding affinity prediction", *Bioorganic & Medicinal Chemistry*, vol. 72, no. 1, oktober, 2022.

[13] M. Aach, R. Sedona, A. Lintermann, G. Cavallaro, H. Neukirchen, M. Riedel, "Accelerating Hyperparameter Tuning of a Deep Learning Model for Remote Sensing Image Classification", in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Kuala Lumpur, Malaysia, 2022. [Online] Tillgänglig: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9883257

[14] V. Mnih *et al*, "Playing Atari with Deep Reinforcement Learning", *DeepMind Technologies*. [Online] Tillgänglig: https://arxiv.org/pdf/1312.5602.pdf

[15] S. Fujimoto, E. Conti, M. Chavamzadeh, J. Pineau, "Benchmarking Batch Deep Reinforcement Learning Algorithms", *Facebook AI Research*. [Online] Tillgänglig: https://arxiv.org/pdf/1910.01708.pdf

[16] DeepLearning.AI, Stanford University. (2023). Unsupervised Learning, Recommenders, Reinforcement Learning. [Online] Tillgänglig: https://www.coursera.org/learn/unsupervised-learning-recommenders-reinforcement-learning

[17] L. Hardesty, "Explained: Neural networks". *MIT News Office*. [Online] Tillgänglig: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 Hämtad: 9 april, 2023.

[18] "ReLU", *PyTroch*. [Online] Tillgänglig: https://pytorch.org/docs/stable/generated/torch.nn.ReLU.html Hämtad: 23 april, 2023.

[19] K. Mohanapriya, N. Sangavi, A. Kanimozhi, V.R. Kiruthika, P. Dhivya, "Optimized Feed Forward Neural Network for Fake and Clone Account Detection in Online Social Networks" in *2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS)*, Erode, India, 2023. [Online] Tillgänglig: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10104616

[20] K. Nanthini, D. Sivabalaselvamani, D. Selvakarthi, D. Pavethran, N. Srinaath, K.S. Vignesh, "Performance of Recurrent Neural Networks in Liver Disease Classification", in *Second International Conference on Electronics and Renewable Systems (ICEARS)*, Tuticorin, India, 2023. [Online] Tillgänglig: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10085202

[21] A. Ajagekar, "Adam", *Cornell University*. [Online] Tillgänglig: https://optimization.cbe.cornell.edu/index.php?title=Adam Hämtad: 19 mars, 2023.

[22] M.L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*, New York, NY: Wiley-Interscience, 2005.

[23] D. Harris, *Digital Design and Computer Architecture: From Gates to Processors*, 2nd ed., San Francisco CA, Burlington: Elsevier Science & Technology, 2007.

[24] "Population, total", *The World Bank*. [Online] Tillgänglig: https://data.worldbank.org/indicator/SP.POP.TOTL Hämtad: 15 maj, 2023.

[25] T.P. Lillicrap et al, "Continuous control with deep reinforcement learning", *Google Deepmind*. [Online] Tillgänglig: https://arxiv.org/pdf/1509.02971.pdf

[26] S. Patel, "An Overview and Application of Deep Convolutional Neural Networks for Medical Image Segmentation", in *Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, 2023. [Online] Tillgänglig: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10073857

[27] "VGG-16 convolutional neural network", *MathWorks*, 2023. [Online] Tillgänglig: https://se.mathworks.com/help/deeplearning/ref/vgg16.html Hämtad: 13 april, 2023.

[28] H.H. Aghdam, E.J. Heravi, *Guide to Convolutional Neural Networks*, Cham, Switzerland: Springer, 2017.

[29] *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*, ISO/IEC 9075-1:2016, december 2016. [Online] Tillgänglig: https://www.iso.org/standard/63555.html

[30] *Common Format and MIME Type for Comma-Separated Values (CSV) Files*, RFC 4180, SolidMatrix Technologies, Inc. oktober 2005. [Online] Tillgänglig: https://datatracker.ietf.org/doc/html/rfc4180

[31] *Information technology – The JSON data interchange syntax*, ISO/IEC 21778:2017, november 2017. [Online] Tillgänglig: https://www.iso.org/standard/71616.html

[32] *Features | Pytorch*, Pytorch. [Online] Tillgänglig: https://pytorch.org/ Hämtad: 7 maj, 2023.

[33] *About: Standard C++*, Standard C++. Accessed May 4, 2023. [Online] Tillgänglig: https://isocpp.org/about Hämtad: 4 maj, 2023.

[34] *Pytorch C++ API – Pytorch main documentation*, Pytorch. [Online] Tillgänglig: https://pytorch.org/cppdocs/ Hämtad: 4 maj, 2023.

[35] *About Python*, Python. [Online] Tillgänglig: https://www.python.org/about/ Hämtad: 4 maj, 2023.

[36] *Hugging Face – The AI community building the future.* [Online] Tillgänglig https://huggingface.co/ Hämtad: 4 maj, 2023.

[37] *Vertex AI*, Google Cloud. [Online] Tillgänglig https://cloud.google.com/vertex-ai Hämtad 28 maj, 2023.

[38] *Docker: Accelerated, Containerized Application Development*, Docker. [Online] Tillgänglig: https://www.docker.com/ Hämtad: 4 maj, 2023.

[39] C.Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning", *Google Inc.* [Online] Tillgänglig: https://arxiv.org/pdf/1602.07261v2.pdf

[40] C.M. Virbin, "Recommendations for reporting measures of diagnostic accuracy", *Cytopathology: official journal of the British Society for Clinical Cytology*, vol. 34, no. 3, pp. 185-190. [Online] Tillgänglig: https://onlinelibrary.wiley.com/doi/full/10.1111/cyt.13208

[41] *Inception v4*, Hugging Face. [Online] Tillgänglig: https://huggingface.co/docs/timm/models/inception-v4 Hämtad: 4 maj, 2023.

[42] *Reinforcement Learning (DQN) Tutorial*, Pytorch. [Online] Tillgänglig: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html Hämtad: 4 maj, 2023.

[43] I. Loshchilov, F. Hutter, "Decoupled Weight Decay Regularization", *University of Freiburg.* [Online] Tillgänglig: https://arxiv.org/pdf/1711.05101.pdf

[44] F.S. Passino, L. Maystre, D. Moor, A. Anderson, M. Lalmas "Finding Structure in Users' Evolving Listening Preferences", *Spotify.R&D Research*, 19 Apr. 2021. [Online] Tillgänglig: https://research.atspotify.com/2021/04/

finding-structure-in-users-evolving-listening-preferences/ Hämtad: 19 april, 2023.

[45] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection", *ACM Transactions on Information & System Security*, vol. 3, no. 3, pp. 186-205, augusti 2000. [Online] Tillgänglig: https://dl.acm.org/doi/pdf/10.1145/357830.357849

[46] M. McCloskey, N. J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem", *Psychology of Learning and Motivation*, vol. 24, pp. 109-165, 1989. [Online] Tillgänglig: https://www.sciencedirect.com/science/article/abs/pii/S0079742108605368

[47] "Softmax", *PyTorch*, 2023. [Online] Tillgänglig: https://pytorch.org/docs/stable/generated/torch.nn.Softmax.html Hämtad: 23 april, 2023

[48] I. MacKenzie, C. Meyer, S. Noble, "How retailers can keep up with consumers", *McKinsey & Company*. [Online] Tillgänglig: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers Hämtad: 9 maj, 2023

[49] J. Camacho, K. Wasielewska, "Dataset Quality Assessment in Autonomous Networks with Permutation Testing", in *Network Operations and Management Symposium (NOMS)*, Budapest, Hungary, 2022. [Online] Tillgänglig: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9789767&tag=1

[50] "24/7 Carbon-Free Energy by 2030", *Google Data Centers*. [Online] Tillgänglig: https://www.google.com/about/datacenters/cleanenergy/ Hämtad: 15 maj, 2023

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2023
www.chalmers.se

GÖTEBORGS
UNIVERSITET

CHALMERS