UPPSALA UNIVERSITET CAMPUS GOTLAND
Institutionen för informatik och media
Utbildningsprogram:
Kandidatprogram i systemvetenskap Programvaruteknik, 180 HP
Examinator: Marius Mihailescu, marius.mihailescu@im.uu.se
Handledare: Marius Mihailescu, marius.mihailescu@im.uu.se
Handledare: Albin Larsson albin.larsson@raa.se

Datum: 2019-05-15

Examensarbete inom
Programvaruteknik B, 15 hp

Validering av objekttyp med maskininlärning för att främja användning av kulturarvssamling

Daniel Persson
Petter Gullin

Sammanfattning

Det svenska kulturarvet digitaliseras i allt större utsträckning för att tillgängliggöras och användas i samhället. För att kunna användas krävs att de metadata som lagras är korrekt. Vi har fått i uppdrag att undersöka möjligheterna att med maskininlärning identifiera felaktiga metadata i form av felklassificerade objekt i databasen K-samsök. Vi har tränat maskininlärningsmodeller med TensorFlow och NasNet Large och kontrollerat samlingar i K-samsök. Vårt arbete visar att det är fullt möjligt att identifiera felklassificeringar med maskininlärning även om vissa utmaningar finns.

Nyckelord: Maskininlärning, Tensorflow, Kulturarv, Agregationsdatabas, klassificering, Bildklassificering, Metadata.

Abstract

A great amount of the Swedish cultural heritage is being digitized to be made available for use by the society. To be used the metadata stored with the heritage needs to be correct. We have been given the task to determine whether machine learning can be used to identify incorrect metadata like objects in the database K-samsök with incorrect classification. We have trained machine learning models with TensorFlow and NasNet Large and checked collections in K-samök. Our work shows that it is possible to use machine learning to identify objects with incorrect classification even though there are some challenges.

Keywords: Machine Learning, TensorFlow, Cultural heritage, Aggregator database, Classification, Image-classification, Metadata.

Förord

Vi vill tacka vår handledare på riksantikvarieämbetet Albin Larsson för stöd och råd, grunden till koden för hämtning från K-samsök och givande diskussioner om maskininlärning. Tack till hela utvecklingsavdelningen på Riksantikvarieämbetet för en välkomnande och inspirerande miljö.

Vi vill tacka vår handledare från universitetet Marius Mihailescu för stöd i författande av denna rapport.

Till sist tackar vi våra kurskamrater för givande diskussioner vid seminarium under kursens gång.

Innehållsförteckning

1 Beskrivning av uppdrag

I detta inledande kapitel beskriver vi nuläget och förklarar vad som ligger till grund för behovet att utföra uppdraget. Vi går igenom vilka problem det avser lösa och hur det kan hjälpa uppdragsgivaren att nå sina mål. Vidare redovisas våra personliga mål och förväntningar, vilka avgränsningar som gjorts och en kunskapsinventering. Sist i kapitlet följer en redogörelse för hur författarnas bidrag till materialet i rapporten sett ut samt en beskrivning av dispositionen för resterande kapitel.

1.1 Bakgrund och nuläge hos uppdragsgivaren

Riksantikvarieämbetet (RAÄ) är en statlig myndighet som bland annat har till uppdrag att förvalta och tillgängliggöra det svenska kulturarvet (Riksantikvarieämbetet, u.d). En del av tillgängliggörandet är att digitalisera olika kulturarvssamlingar. Digisam är en samverkan mellan olika statliga organisationer med syfte att främja digitalisering, digitalt bevarande och digitalt tillgängliggörande (Digisam, 2015).

RAÄ är en av de organisationer som ingår i Digisam och inhyser även dess sekretariat. En rapport om nuläget i digitaliseringsprocessen visar att endast några få procent av de till Digisam anslutna organisationernas samlingar är digitaliserade. Rapporten visar även att fokus på att höja kvaliteten på lagrade data i många fall saknas (Digisam, 2015). RAÄ förvaltar flera nationella databaser för kulturhistorisk information, exempel på dessa databaser är Fornsök, Kringla, Samla, K-samsök och Bebyggelseregistret. Databasen K-samsök är en så kallad aggregator, vilket innebär att den samlar in data från flera olika databaser för att sammanställa dem i en central databas. För närvarande finns cirka sju miljoner databasobjekt (nedan kallat objekt) i K-samsök från 64 olika institutioners databaser (Riksantikvarieämbetet, u.d). Det kan exempelvis vara olika fysiska museiföremål (nedan kallat föremål), fotografiska bildsamlingar (nedan kallat fotografi) och arkivhandlingar. I många fall finns även en bild av föremålet eller fotografiet lagrat i databasen. Det är de data som finns i K-samsök och dess kvalitet som detta projekt handlar om.

För varje databasobjekt finns metadata lagrade som innehåller en klassificering av vilken typ av objekt det rör sig om. Att korrekt metadata finns för objekten är en förutsättning för att kunna tillgängliggöra samlingen för allmänheten. RAÄ har ett uppdrag att utöver tillgängliggörande av kulturarvet främja användning av det (Riksantikvarieämbetet, u.d). En förutsättning för att kulturarvet skall kunna användas är att det enkelt går att genomföra sökningar. För att hitta det som söks är det avgörande att samlingarnas metadata är korrekt.

Två vanliga typer är just fotografier och föremål. Varje institution är ansvarig för att korrekta metadata lagras i databasen (Digisam, 2014). På RAÄ finns en uppfattning att de metadata som lagras är av varierande kvalitet. Som ett exempel finns ett antal fotografier med fotograferingsdatum innan år 500 medan den första kameran producerades långt senare. Fenomenet kan enkelt ses i den visualisering av innehållet i K-samsök som finns tillgänglig på internet (Riksantikvarieämbetet, u.d). RAÄ har även märkt att det finns ett antal fall där objekt som klassificerats som fotografi egentligen är föremål. En anledning till denna felklassificering kan vara att föremålet representeras av ett fotografi i databasen.

1.2 Projektmål och övergripande mål

Iden till projektet kommer från RAÄ och har beskrivits enligt följande:

”Validering av objekttyp med maskininlärning - Kringla.nu/K-samsök innehåller över sju miljoner poster/objekt kring kulturarv två mycket vanliga objekttyper är fotografi och föremål. Ofta har dessa dock förväxlats. Hur effektivt skulle man kunna identifiera problematiska objekt med objektets bild och maskininlärning (klassificering). Uppdraget består utav:

- *att använda Tensorflow och Python**
- *att med högkvalitativt innehåll från K-samsök träna en klassificeringsmodell*
- *att med högkvalitativt innehåll från K-samsök validera modellen*
- *att testa modellen på eventuellt problematiskt innehåll*

**Förutsättning för att kunna få tekniskt stöd från handledaren.”*

Målet med uppdraget är att undersöka möjligheterna att använda maskininlärning för att identifiera föremål som felaktigt klassificerats som fotografier. Att på detta sätt automatiskt identifiera felaktiga klassificeringar är det något som RAÄ kan använda för att stödja de till K-samsök ansluta institutionerna i arbetet att höja kvaliteten på deras metadata. Detta bidrar direkt till de övergripande målen för RAÄ att främja användning av kulturarvet. För att utföra uppdraget behöver vi skaffa oss kunskaper kring TensorFlow och maskininlärning.

1.2.1 TensorFlow

TensorFlow som vi skall använda oss av är ett bibliotek med funktionalitet för maskininlärning som är skapat av Google. Det kan ses som ett interface som används för att göra maskininlärning mer tillgängligt. Genom att placera bildsamlingar i olika mappar kan TensorFlow tränas till att identifiera den typ av bild som finns i varje mapp. I vårt fall blir det mappar med fotografier och föremål. När TensorFlow tränar skapas en modell som sedan kan sparas för framtida användning. I denna modell kan en ny bild analyseras och modellen ger ett svar på vilken av de två typerna den nya bilden tillhör. Modellen visar även med ett procenttal säkerheten på klassificeringen (Abadi, et al., 2016).

1.2.2 Förväntningar från uppdragsgivare

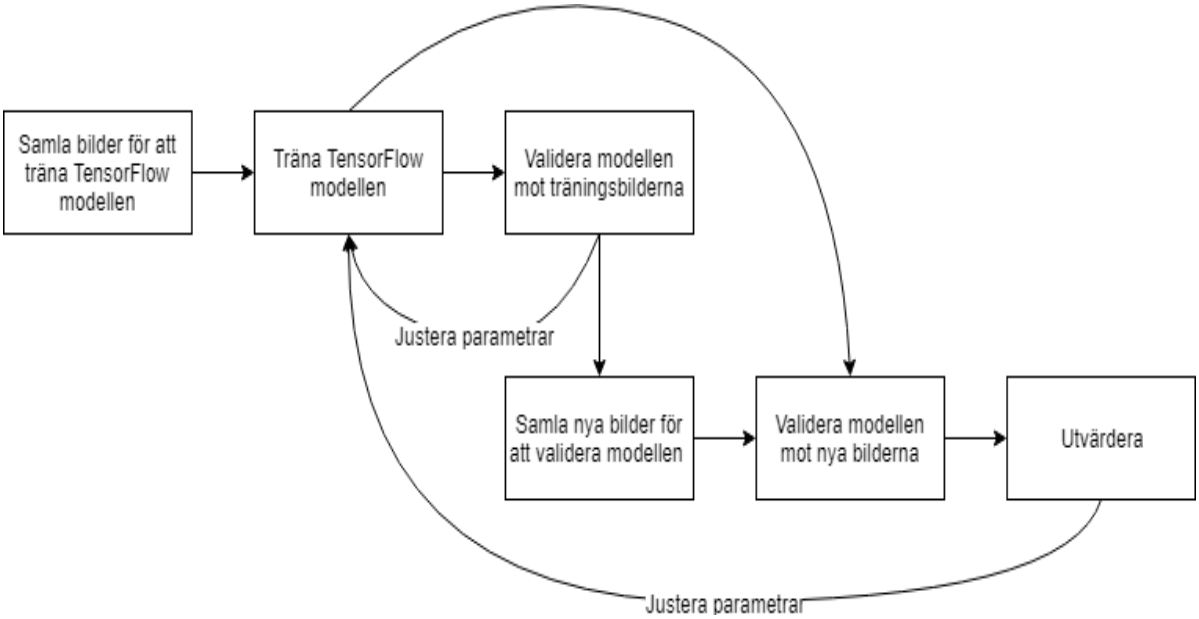
RAÄ förväntar sig att vi skall leverera en rapport där vi beskriver våra försök med att klassificera föremål. I rapporten förväntas vi även dra slutsatser kring hur effektiv metoden är. Utöver vår skriftliga rapport skall vi leverera de färdigtränade modellerna och all övrig producerad källkod som publiceras på internet under öppen licens.

1.2.3 Våra förväntningar

Vi förväntar oss att kunna fastställa huruvida det är möjligt att använda maskininlärning för att identifiera felaktigt klassificerade bilder. Vårt mål är att ta fram en klassificeringsmodell som effektivt kan hitta felaktigt klassificerade bilder. Arbetet förväntas ge RAÄ en grund för att fortsätta arbetet med användning av maskininlärning. Vi räknar med att under arbetets gång identifiera en rad utmaningar som vi ställs inför. Våra identifierade utmaningar kan gynna eventuella framtida försök med klassificering på RAÄ. Till vårt förfogande har vi en tilldelad arbetsplats utrustad med datorer och övrig hårdvara som krävs. Vi har

en kompetent handledare som finns på plats de flesta dagarna så att vi kan ställa frågor och få stöd under arbetet. Den arbetsplats vi tilldelats är i ett öppet kontorslandskap med övriga studenter från kursen och RAÄ's utvecklare. Så vi förväntar oss att ha goda möjligheter att få input eller, ställa frågor vid behov. Vi har även möjlighet att nå vår handledare de tillfällen som han inte är på plats.

Med stöd av tidigare kunskaper i utvecklingsmetoder planerar vi att arbeta iterativt enligt figur 1.



Figur 1 - Klassificeringsprocess

Figuren visar de olika steg vi planerar att genomföra under arbetet med att träna en klassificeringsmodell (Abadi, et al., 2016).

1. Vi börjar med att samla lämpliga bilder för träning av modellen i två olika samlingar, dessa måste vi manuellt verifiera att de var för sig innehåller bilder som representerar föremål och fotografier.

2. Träna en klassificeringsmodell i TensorFlow för att lära den känna igen de två objekttyperna.

3. Validera modellen mot Träningsbilderna (Punkt 1 och 2 upprepas vid behov för att justera parametrar).

4. Samla in nya bilder på föremål, här kan vi med fördel inkludera bilder som är fotografier och sådana bilder som vi förväntar oss att modellen kommer att ha svårigheter att identifiera.

5. Validera modellen mot de nya bilderna.

6. Utvärdera resultatet.

7. Upprepa tidigare steg enligt behov.

Det finns ett stort antal olika färdigtränade klassificeringsmodeller att utgå ifrån vid klassificering av bilder. De flesta är tränade till att identifiera vad bilden föreställer. Den förmågan är inget vi har nytta av i vårt arbete. Dock finns möjligheten att använda en befintlig modell och träna om den för det aktuella ändamålet. För att snabbare komma igång med arbetet tänker vi använda NasNet Large modellen för TensorFlow och träna om den till att identifiera föremål och fotografier. Därefter skall vi träna en egen modell som iterativt kommer att utvecklas.

Med den tidigare erfarenheten vi har av grupparbete och utvecklingsmetoder ansåg vi det viktigt med tydliga tidsramar och deadlines. Vi har därför skapat en tidsplanering se bilaga A för hela arbetet och en scrumboard för kodningsuppgifterna se bilaga B. En scrumboard är en planeringstavla där uppgifterna brutits ner till små korta uppgifter. De som arbetar i projektet väljer en uppgift från tavlan och markerar den som pågående och senare som avslutad när uppgiften är löst (Scwaber & Beedle, 2001). Båda dessa planeringstavlor ser vi som levande dokument och kommer att uppdateras efterhand som projektet fortlöper och utformas.

1.3 Avgränsningar

Det finns ett stort antal olika färdigtränade modeller att utgå ifrån vid klassificering av bilder. Vi har dock valt att endast använda oss av NasNet Large modellen då de enligt statistik på TensorFlows hemsida verkar vara väl lämpade för vårt problem. (TensorFlow, u.d.).

Vid hämtning av bilder från K-samsök har vi märkt att vissa institutioner inte tillåter att vi direkt via en länk försöker öppna en bild. Vi får ett felmeddelande att vår session inte är giltig eftersom vi inte är inloggade

eller kommer från en betrodd källa som exempelvis sökmotorn kringla. I ett framtida scenario med försök att hitta felaktigt klassificerade objekt bör detta problem lösas så att det blir möjligt att även genomsöka dessa samlingar. I detta projekt väljer vi dock att ignorera de samlingar som ger detta problem då vi har tillräckligt underlag att dra slutsatser från övriga samlingars data. Vidare kommer vi att fokusera på de samlingar vi kan se har hög kvalitet på metadata vid manuell granskning.

1.4 Kunskapsinventering

Vi har reflekterat kring vilken kunskap vi besitter och vad som behövs för att utföra detta uppdrag. I tabell 1 redovisas identifierade kunskapsbehov både vad gäller kunskap vi besitter och kunskap vi behöver tillskaffa oss.

Tabell 1 - Kunskapsinventering

	Teknisk kunskap	**Tematisk kunskap**	**Procedurell kunskap**
Nuläge (kunskap som finns med i nuläget)	Grundläggande kunskaper i programmering med Java, C# och JavaScript. Grundläggande kunskap kring SQL-databaser och NoSQL-databaser. Grundläggande kunskap i att ställa frågor mot ett API och hantera svar i form av JSON objekt.	Vi saknar kunskaper på temat. Vår handledare har erfarenhet av klassificering av bilder med TensorFlow.	Kvalitetssäkring, innovation och design, samt systemutvecklingsmetoder med scrum.
Fördjupningsbehov	Bekanta oss med syntaxen för Python. Bekanta oss med det API som används för att hämta data från k-samsök och syntaxen	Vi behöver fördjupa oss i klassificering av bilder och förstå hur vi på bästa sätt väljer ut de bilder som klassificeringsmodellen skall	Vi behöver lära oss mer om att utvärdera vårt arbete och presentera resultaten

	för frågespråket CQL. Bekanta oss med TensorFlow och träning av modeller för klassificering av bilder.	tränas utifrån.	

1.5 Översikt / disposition

I kapitel 1 beskrivs projektet och vilka mål det avser lösa för uppdragsgivaren. Vi går också igenom hur nuläget ser ut för uppdragsgivaren

I kapitel 2 finns resultatet av den kunskapsinventering vi utfört. Här redovisas vilka kunskaper vi behöver utöka för att utföra uppdraget.

I kapitel 3 redovisas vårt arbete och de resultat vi kommit fram till analyseras.

I kapitel 4 utvärderar vi det arbete vi gjort med stöd av den litteratur som ligger till grund för arbetet.

I kapitel 5 diskuteras vår syn på resultaten och vi reflekterar kring hela projektet.

1.6 Författarnas bidrag

Då vi i ett tidigt skede och på ett öppet sätt diskuterade våra personliga styrkor och svagheter fann vi en naturlig uppdelning av arbetet. Daniel anser vi är den starkare när det gäller kodning och Petter den starkare när det gäller rapportskrivning. Det är därför vår huvuduppdelning när det gäller arbetsuppgifterna. Båda har dock arbetat med kodning och rapportskrivning. Som en extern bilaga till denna rapport finns loggböcker där vi beskrivit det dagliga arbetet. En bekräftelse av gruppens insats finns i bilaga J.

2 Litteraturgenomgång

Vi har sökt efter forskningsartiklar relaterade till vårt problem för att se om någon ställts inför liknande problem tidigare. Vi har sökt på Google scholar och Uppsala universitetsbiblioteks söktjänst. Här är exempel på de sökord vi använt oss av i olika kombinationer.

- Bildklassificering
- Maskininlärning
- Aggregationsdatabaser för kulturarv
- Metadata för kulturarv
- TensorFlow

Vi har inte lyckats hitta några system som använts för att lösa liknande problem som vi ställts inför trots att det finns flera databaser som fungerar som aggregat för att samla data kring kulturarvssamlingar. Det ger oss anledning att titta på vårt problem på en högre nivå och försöka hitta liknande egenskaper hos andra problem.

2.1 Definition av högnivåproblem

Problemet handlar om att klassificera huruvida ett fotografi tillhör en viss klass eller inte. Detta är ett vanligt användningsområde för maskininlärning. Det kan till exempel handla om att utifrån en mängd olika klasser identifiera vad en bild föreställer. Därför är det inga svårigheter att hitta relaterade problem. Vi ser även att högnivåproblemet innehåller en annan del. De metadata som finns har bristande kvalitet. Informationskvalitet är något som är av stor vikt för ett informationssystems framgång (Delone & McLean, 2003).

2.2 Liknande system

I vår uppdragsbeskrivning rekommenderas att använda ett färdigt system för maskininlärningen (TensorFlow). Med det kan sedan egna klassificeringsmodeller utvecklas. Det finns även flera olika färdiga

klassificeringsmodeller att utgå ifrån vid klassificering av bilder. Som vi tidigare nämnt kommer vi att använda Googles färdiga klassificeringsmodell NASNet Large.

2.3 Relaterad forskning/Litteratur

Vi saknar erfarenhet av att använda maskininlärning därför har vi använt oss av artiklar som beskriver hur maskininlärning fungerar och hur TensorFlow används. Det finns flertalet guider med kodexempel på internet för den som vill börja använda TensorFlow. För att bekanta oss med biblioteket har vi använt oss av några sådana guider. Ett exempel är TensorFlows egen guide (TensorFlow, u.d.). För att förstå neurala nätverk och vilka parametrar som har betydelse för slutresultatet har vi även sökt artiklar inom det området. Nedan följer en sammanställning av de artiklar vi använt oss av under arbetet.

2.4 Implikationer för uppdraget

De artiklar vi valt att stöda oss på påverkar projektet på olika sätt. I tabell 2 redovisas varje enskild artikels påverkan på projektet.

Tabell 2 - Litteratur

An introduction to Convolutional Neural Networks (O'Shea & Nash, 2015).	Hjälper oss med teknisk kunskap kring maskininlärning och påverkar designenprocessen.
The DeLone and McLean Model of InformationSystems Success: A Ten-Year Update (Delone & McLean, 2003).	Hjälper oss att förstå effekterna av vårt arbete på RAÄ's mål och därmed designprocessen.
Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis (Simard, et al., 2003).	Hjälper oss med teknisk kunskap kring maskininlärning och med val av material för träning av modell påverkar designen av produkten.
Image Recognition and Study of Hyperparameter Optimization of Convolutional Neural Networks	Hjälper oss med teknisk kunskap kring maskininlärning framförallt vid skapandet av en egen klassifi-

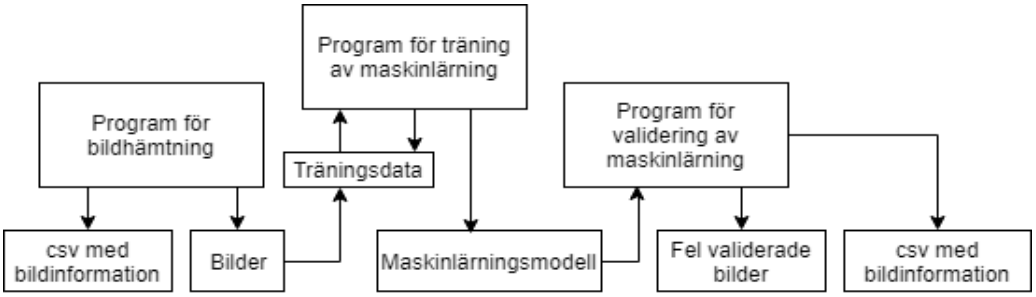
Using TensorFlow and Keras Frameworks (Chari, 2018).	ceringsmodell påverkar designen av produkten.
Learning Transferable Architectures for Scalable Image Recognition (Zoph, et al., 2018).	Används för att förstå NASNET Large och hur det kan implementeras. Påverkar designprocessen.
Machine Learning from Imbalanced Data Sets 101 (Provost, 2000).	Förväntas påverka vårt val av data för att träna klassificeringsmodellerna påverkar designprocessen.
TensorFlow: A System for Large-Scale Machine Learning (Abadi, et al., 2016).	Avhandlar vad TensorFlow är och hur det skapades. Förväntas hjälpa oss att snabbare få förståelse för hur TensorFlow kan användas
TensorFlow Explore overfitting and underfitting (Google, u.d.)	Artikeln beskriver effekterna vid överinlärning och hur vi kan se att en modell är överinlärd. Påverkar
TensorFlow: Custom training: basics (Google, u.d.).	Beskriver grunderna i TensorFlow och används för att stödja designprocessen.

3 Resultat / Programvara / Design

I detta kapitel beskrivs de program vi skapat för att lösa uppdraget, arbetet med att träna klassificeringsmodeller och resultaten av arbetet.

3.1 Programvara

För att utföra uppdraget skapade vi tre olika program, som skulle fungera fristående från varandra och utan något användarinterface. De tre delarna av arbetet vi identifierat är hämtning av fotografier från K-samsök, programmet för att träna modellen och ett program för att validera en tränad modell se figur 2.



Figur 2 Översiktlig förklaring för uppdelning av skapade program

Genom att skapa tre fristående program underlättar vi för framtida användning om uppdragsgivaren endast vill använda en enskild funktion som vi skapat.

3.1.1 Bildhämtnings program

För att träna klassificeringsmodellen behövde vi ett bra underlag av bilder som hämtas från K-samsök. Eftersom vi saknade erfarenhet ville vi börja med ett mindre antal bilder för att sedan gradvis utöka antalet och se vilka skillnader det gav i kvalitet och hur lång tid en träning tar. För att lättare kunna upprepa hämtningen med olika parametrar valde vi att skriva ett program se bilaga C. Vi utgick från en databasfråga som vår handledare hade skrivit för hämtning av bilder. Den modifierade vi till ett körbart program där vi kan välja vilken institution som skall hämtas från och hur många bilder som skulle hämtas. När vi tittade i

institutionernas samlingar med sökmotorn kringla kunde vi se att ofta låg det objekt av samma typ efter varandra. För att få ett bättre resultat med vår modell krävdes att vi har en bra spridning på vad bilderna föreställer och inte en överrepresentation av exempelvis pilspetsar eller krukskärvor KÄLLA. Det program vi skrivit hämtar därför poster i slumpmässig ordning. Uppgifter om varje bild sparas i en csv-fil (ett filformat för lagring av data). På så vis finns det dokumenterat vilka bilder vi hämtat vid varje hämtning. För att få en ännu större spridning på materialet kan programmet även hämta från alla institutioner.

3.1.2 Maskinlärningsträningsprogram

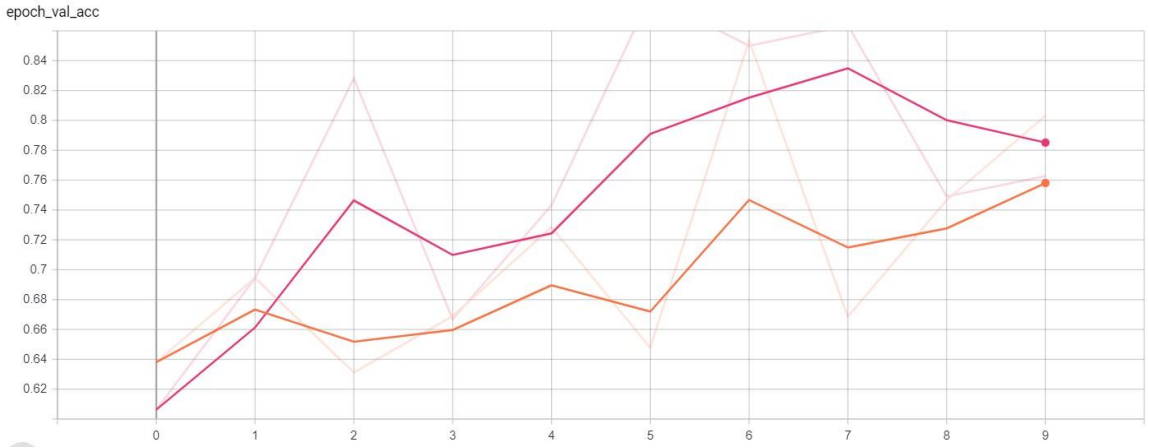
För att träna modellerna skapade vi ett program som läser in de hämtade bilderna och formaterar om dom till att ha samma storlek (bredd/höjd) och gör om dom till svartvit se bilaga D. För att få ett bättre resultat krävs en blandning mellan de två olika klasserna. Att först träna på alla fotografier och därefter föremål skulle ge sämre modeller (TensorFlow, u.d.). Därför randomiseras alla bilder tillsammans innan de används för träning. I Tensorflow används som standard 60 procent av materialet till att träna modellen. 20 procent används för att automatiskt optimera modellen. Träningen upprepas ett valt antal gånger så kallade epoker, om för många epoker görs kommer modellen börja memorera träningsbilderna och lära sig vad som är rätt svar för varje enskild bild kallas överinlärning (Google, u.d.).

De sista 20 procenten används för att validera modellens förmåga efter varje epok. Eftersom dessa 20 procent inte ingår i materialet som används för träningen blir de ett kvitto på om modellen förbättrats eller börjat överlära efter varje epok. En fingervisning om att modellen börjat överlära kan vara ”validation_loss”. Det är ett värde på hur väl en modell vid en given input ger en output som motsvarar det förväntade (Google, u.d.).

När modellen tränas förväntas ”validation-accuracy” (modellens förmåga att klassificera rätt) att stiga för varje epok. ”Validation-loss” däremot sjunker för varje epok. Om modellen börjar överlära kommer validation-accuracy att fortsätta stiga men även ”validation_loss” kommer vända och börja stiga. Generellt vill man ha en modell där

värdet i ”validation-accuracy” är så högt som möjligt och där ”validation_loss” inte vänt och börjat stiga.

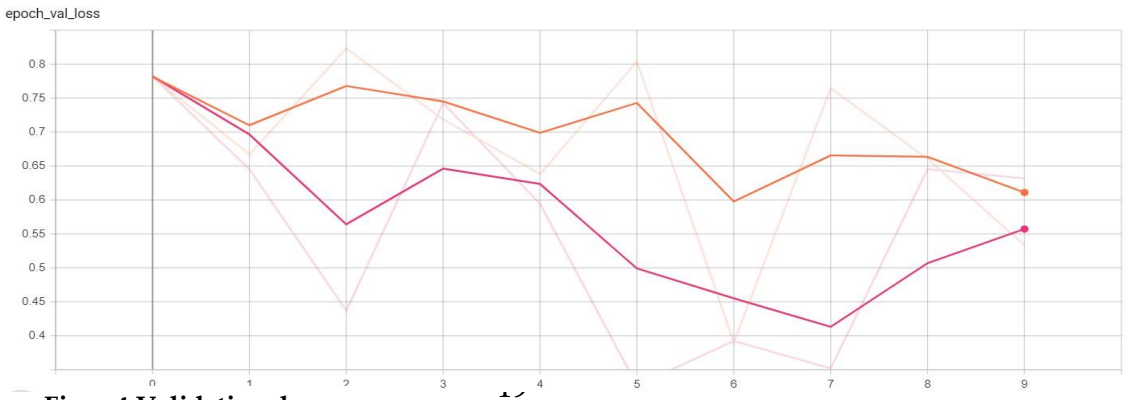
Statistik kring träningen av varje modell sparas ner i en logg-fil och kan sedan visualiseras med hjälp av Tensorboard som finns inbyggt i Tensorflow. Vid träning av modellen används lager av olika typer och antal. I varje sådant lager görs en generalisering av hur bildens pixlar är ser ut för att hitta mönster mellan olika bilder (Simard, et al., 2003). För att hitta rätt kombination upprepas vårt program i en loop där antalet lager av de olika typerna och antalet epoker succesivt ökas. I Tensorboard kan vi sedan se vilken modell som är mest effektiv. Figur 3 nedan visar skillnaden mellan två olika modeller och deras ”validation-accuracy”. Vi kan tydligt se att den röda kurvan når högst. Vid epok sju



når den 83,5% och därefter vänder kurvan nedåt.

Figur 3 Validation-Accuracy

Figur 4 visar ”validation_loss” för samma två modeller, och där ser vi tydligt att samma modell också börjar överinlära vid epok sju, då kurvan vänder uppåt.



Figur 4 Validation_loss

Vid omträning av NasNet Large modellen använde vi oss av image feature vectors. Det innebär att de ursprungliga klasserna som modellen tränats för att lära sig väljs bort. Endast själva basen i modellen där den lärt sig känna igen olika mönster bland pixlarna blir kvar (Google, u.d.). Därefter tränades modellen för att identifiera föremål och fotografier. Även de olika NasNet Large modellerna utvärderades i Tensorboard.

3.1.3 Validering med maskinlärnings program

För att kontrollera hur effektiv en modell är skapade vi ett program som hämtar varje objekt från en viss institution som har klassen fotografi eller föremål och där det finns en bild i databasen för att representera objektet se bilaga E. Därefter kontrolleras varje bild med modellen och jämförs med den klassificering den har i databasen. Om modellens klassificering inte stämmer med klassificeringen i databasen sparas objektets miniatyrbild i en mapp och informationen kring objektet sparas i en csv-fil.

Programmet kan även användas för att jämföra skillnaden mellan två modellers resultat. Mappen med miniatyrbilder som skapats av två modeller jämförs och en länk till de objekt som inte finns i båda mapparna lagras i en csv-fil.

3.2 Material för maskinlärning

För att träna en effektiv modell krävs ofta ett stort antal bilder. Ett problem är dock att träningsmaterialet måste gås igenom manuellt för att säkerställa att alla objekt som klassificerats som exempelvis fotografier verkligen är fotografier. Annars kommer modellen att lära sig fel saker. Så även om ett större antal bilder ofta ger bättre resultat måste det ställas mot arbetsinsatsen att manuellt granska alla bilder. För att träna en modell som skall fungera mot alla institutioner behövde vi hämta material från så många institutioner som möjligt. Det är viktigt att inte någon klass är överrepresenterad. I vårt fall var det inga problem att balansera de två klasserna då det fanns gott om fotografier och föremål att tillgå.

3.2.1 Problem med material för maskinlärning

Vid hämtning av träningsmaterial upptäckte vi att det fanns gott om föremål bland fotografier. Vid närmare granskning upptäckte vi dock en del problem. Vissa institutioner hade lagrat även det fotografi som tagits för att dokumentera ett föremål som en post i databasen. Exempelvis den kam som visas i figur 5 förekommer både som föremål och fotografi.

Figur 5 - kam av ben

Den har fått benämningen dokumentationsbild, för att slippa få med den typen av bilder vid hämtning av träningsmaterial exkluderades alla dokumentationsbilder. Trots detta såg vi föremålsbilder bland fotografierna även fortsättningsvis. Det visade sig att det fanns flera olika benämningar som användes. Vi träffade på följande benämningar:

- Dokumentationsbild
- Id-Bild
- Konservationsrapport
- Placeringsbild
- Föremålsbild
- Presentationsbild

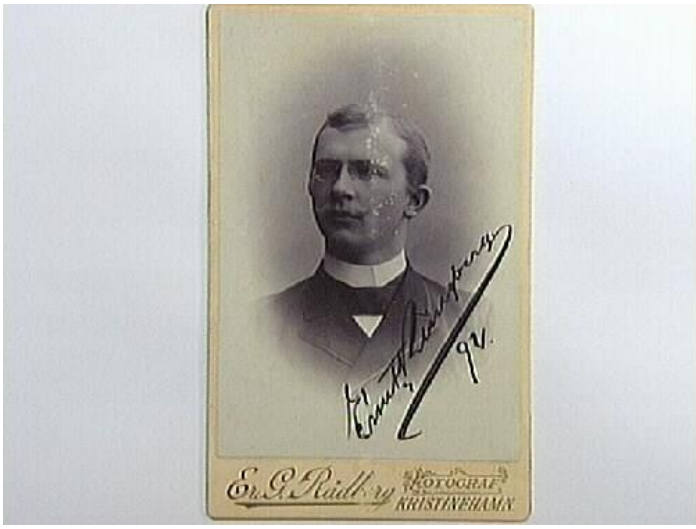
Alla dessa benämningar filtrerades bort vid nedladdning av träningsmaterial.

3.2.2 Urval av träningsmaterial

Vid den manuella genomgången av bilder för träning stötte vi på en del problem. Eftersom vi saknar kunskap kring vad institutionerna faktiskt har i sina samlingar kunde vi inte avgöra korrekt klassificering. Ett exempel är det vykort enligt figur 6 som lagrats som ett fotografi.

Figur 6 - Vykort

En annan vanlig typ av fotografier är uppfodrade porträtt i olika format, ofta med en text på porträttet se figur 7.



Figur 7 - Porträtt

Vi valde vid urval av träningsdata att sortera bort den typen av objekt från klassen fotografier då vi mer uppfattade dem som dokument eller föremål. Därför lät vi dem därmed finnas kvar i klassen föremål.

Då arbetet med att gallra bland träningsmaterialet skedde av oss som saknar kunskap kring innehållet i de aktuella samlingarna ställdes vi inför frågor som hur exempelvis ett vykort bör klassificeras eller vad som uppfattas som ett dokument eller fotografi. Våra beslut kring klassificering kan ha påverkat slutresultatet.

Vi har skapat tre olika samlingar med träningsmaterial se tabell 1 för mer detaljerad information.

Tabell 1 - Träningsdata

Träningsdata			
Storlek	Slumpmässig hämtning	Manuellt rensad	Syfte
1000 bilder	nej	nej	Se om val av väg för skapande av modell var korrekt.
7760 bilder	nej	ja	Se om större träningsmaterial med olika hyperparameterar ger skillnad.
21 850 bilder	ja	ja	För att skapa en slutgiltig modell baserad från tidigare testning.

Det första urvalet med 1000 bilder användes för att komma igång med träningen av modeller och lära oss mer om hur utvärdering av modeller med Tensorboard går till.

Efter att ha tränat några modeller med det första materialet med 1000 bilder samlade vi ett nytt större material för att se vilka skillnader det skulle ge. Det materialet med 7760 bilder sorterades även igenom manuellt och vissa bilder plockades bort.

Det slutgiltiga träningsmaterialet med 21 850 bilder samlades för att se resultatet vid ett riktigt stort underlag. Att manuellt granska en så stor mängd är en utmaning och vi har i efterhand sett att vissa bilder ytterligare borde ha rensats bort.

3.3 Slutgiltiga modeller

Efter att ha tränat flertalet modeller valde vi ut de två som visade bäst resultat i Tensorboard se tabell 2. De valdes utifrån ”validation-accuracy” och ”validation-loss” som beskrivits tidigare.

Tabell 2 - Slutgiltiga modeller

Träningsmetod	Valideringsprocent
Egen tränad TensorFlow modell	80%
Omtränad NASNet-Large modell	96%

Den egentränade modellen hade en precision på 80% vid validering av träningsmaterialet se bilaga F för graf över träningen. NasNet modellen däremot hade en precision på 96% med samma träningsmaterial se bilaga G för graf över träningen. En stor skillnad mellan de båda modellerna var tidsåtgången. Den egentränade modellen tog cirka tre timmar att träna medan NasNet modellen tog ca 15 minuter.

De båda modellerna kontrollerades därefter mot ett material som vi samlat ihop för detta ändamål. Materialet innehöll 220 typiska föremålsbilder och lika många fotografier. Vi ville ha ett kontrollerat material att validera modellerna mot för att kunna dra slutsatser. Här kunde vi se att den egentränade modellen presterade bättre än vad grafen i Tensorboard visade. Mot kontrollmaterialet blev precisionen >86%. NasNet modellen hade en precision på 96% precis som vid träningen se bilaga F för en tabell över resultaten.

Därefter kördes en validering mot sex olika institutioners databaser. Varje fotografi i databasen kontrollerades med de båda modellerna och i de fall då klassificeringen i databasen inte överensstämde med modellens resultat så sparades bildens tumnagel ner i en mapp och uppgif-

ter om objektet sparades i en csv-fil se bilaga H för en sammanställning av resultaten. Här såg vi en stor skillnad i den tid det tog att validera varje bild. Den engentränade modellen var något snabbare per bild. En av de kontrollerade samlingarna innehöll över 12 000 fotografier. Det gjorde att kontrollen tog ca två timmar längre med NasNet modellen. Testerna visade att den egentränade modellen plockade ut fler bilder än NasNet modellen vilket stämmer med modellernas precision vid träning. Dock ser vi att det varierar mellan de olika institutionerna, i Wasamuseets samling plockades nästan lika många bilder ut med de båda modellerna.

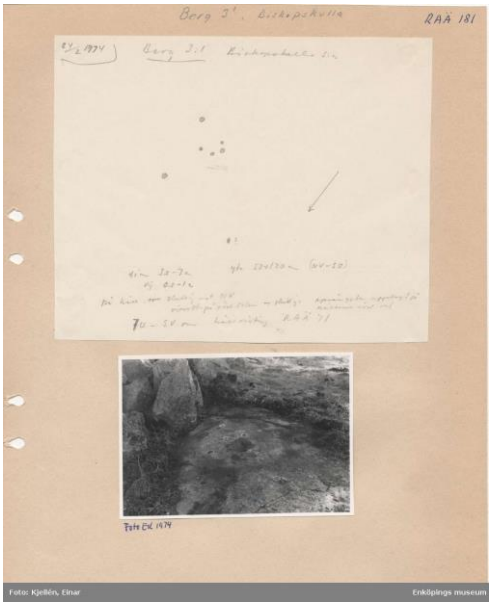
Vi kontrollerade vilka bilder som plockats ut av båda modellerna se bilaga I för en sammanställning. Även här varierar resultaten mellan de olika institutionerna.

3.4 Resultat utifrån valideringsprogram

De objekt som våra modeller identifierade som felaktigt klassificerade analyserades för att se vad resultatet innehåller. Många bilder kunde vi se var falskt positiva. Vilket innebär att de identifierats som felaktigt klassificerade av modellerna trots att de egentligen hade rätt klassificering. Bland resultaten fanns dock även en del andra intressanta exempel.

3.4.1 Avvikelser i resultatet

Vid körning av valideringsmodellen mot vissa institutioner lyckades vi identifiera en del objekt som enligt oss är felaktigt klassificerade. Exempelvis identifierade vi ett stort antal dokumentationer av hällristningar som klassificerats som fotografier. De bestod ofta av ett flertal fotografier uppfodrade på papper. Många hade även skisser och anteckningar på pappret se figur 8. Vi anser att dessa istället borde klassificeras som



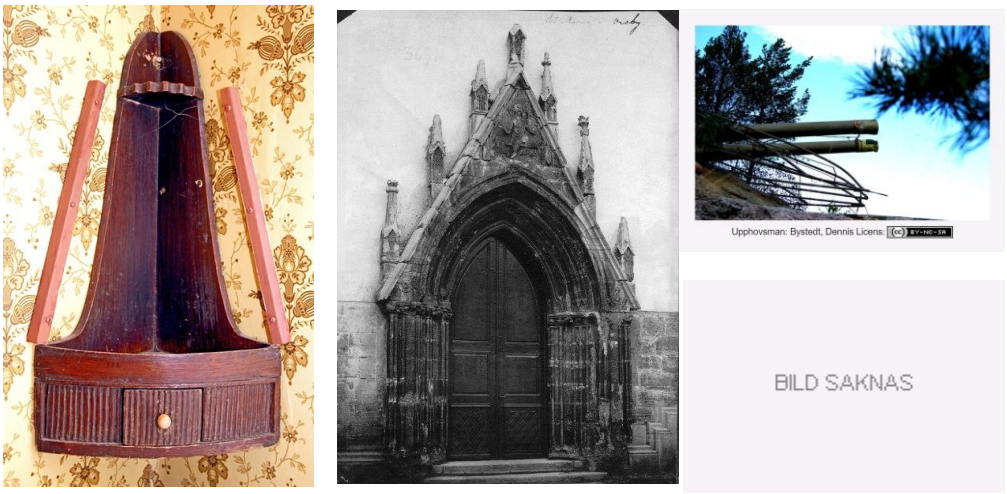
Figur 8 Dokumentation av hällristning

dokument.

Vid körningar för att hitta fotografier bland föremålen hittade vi en del bilder som vår egentränade modell felaktigt klassificerade som fotografier. Det har till stor del att göra med hur dessa bilder är tagna. Exempelvis hyllan i figur 9 klassificerade vår modell som ett fotografi. Vi misstänkte att den röriga bakgrunden var ett problem och provade att klippa ut själva hyllan och placera den på en vit bakgrund. Då klassificerade modellen hyllan som ett föremål. En port i Sankta Maria Domkyrka i Visby klassificerades däremot som ett föremål trots att det är ett foto se figur 9. Här är det troligen den vita kalkstensväggen som ställer till det och ger en falsk träff. En bild tagen på en kanon som finns på Hemsö fästning identifierades som ett fotografi se figur 9. Gemensamt

för alla dessa bilder är att de är tagna i fält och har bakgrunder som gör att de klassificeras felaktigt av våra modeller.

Dessutom ser vi i figur 9 ett exempel där en institution laddat upp en bild med texten Bild Saknas. Då vi i vår databasfråga endast inkluderade objekt som har en bild lagrad ställer detta förfarande till det.



Figur 9 problematiska bilder

3.5 Sammanställning av resultat

Baserat på vår träning av olika modeller och validering av dessa ser vi att det är möjligt att använda maskininlärning för att hitta felaktigt klassificerade objekt i databasen. I vårt arbete hade vi en bred ansats där vi försökte identifiera alla olika typer av felklassificeringar med samma modell. Våra försök visar dock att det är mer effektivt att angripa ett specifikt problem och även skräddarsy det mot en specifik institution. Som vi såg i exemplet med hällristningar som visades i figur 8 så var vår modell mycket bra på att hitta dessa dokument som klassades som fotografier.

Bilder av föremål som är tagna i studio som exemplet med kammen i figur 5 klassificerade våra modeller också som föremål med hög träffsäkerhet. Här upptäcktes även den dubbla lagringen av bilder som görs av vissa institutioner.

4 Utvärdering

I detta kapitel utvärderar vi vårt arbete och argumenterar för vår lösnings validitet baserat på den litteratur vi stödjer oss på.

4.1 Demonstrativ

Vi har visat resultaten av våra tester mot olika institutioners samlingar för vår handledare och fått feedback på de resultat vi fått. Det var här vi fick bekräftat att dokumentationen av hällristningar som diskuterades i kapitel 3.4.1 borde klassificeras som dokument. Ett annat fenomen som egentligen inte rör klassificeringen som upptäcktes vid demonstration var de objekt där fotografier lagrats flera gånger.

4.2 Experimentell

Att träna en klassificeringsmodell är i högsta grad ett experimentellt förfarande. Det är genom att prova oss fram med olika parametrar för modellen som vi i Tensorboard kan se vilken modell som validerar träningsmaterialet mest effektivt. Vi kunde dra slutsatser kring om den aktuella modellen börjat överinlära och fatta beslut om hur vi skulle träna nästa modell. Våra beslut baserade vi på kunskap från artikeln Tensorflow: Explore overfitting and underfitting (Google, u.d.).

4.3 Tolkande

När vi kontrollerat modellerna mot olika institutioners samlingar kan vi se att det finns felaktigt klassificerade objekt i databaserna. Det ser vi som bevis på att felaktiga metadata finns i databasen och att maskininlärning kan användas för att förbättra informationskvaliteten. Vår handledare har bekräftat att de avvikelser vi hittat är felklassificeringar. Kopplat till informationskvalitetens inverkan på ett informationssystems framgång (Delone & McLean, 2003) ser vi att maskininlärning kan användas för att uppnå RAÄ's mål att främja användning av det digitaliserade kulturarvet.

4.4 Metodologisk

Vi har arbetat på ett inkrementellt sätt med att utveckla våra program. För att ha en enkel översikt på vad som gjorts och vad som skall göras i nästa steg har vi använt oss av ett agilt arbetssätt där vi lånat delar från scrum. Vi har delat upp arbetet i sprints med klara deadlines för att se vad som skall göras i nästa steg (Scwaber & Beedle, 2001).

När det gäller träningen av modeller har arbetet snarare varit iterativt. Där träningen av aktuell modell baserats på resultaten av föregående modell. Kunskap kring hur parametrarna för modellerna påverkar har hämtats från artikeln Image Recognition and Study of Hyperpaarameter Optimization of Convolutional Neural Networks Using TensorFlow and Keras Frameworks (Chari, 2018).

4.5 Pragmatisk

Vår handledare blev genom vårt arbete uppmärksammad på att vissa institutioner lagrade samma bild flera gånger. Att de lagrar en bild för att symbolisera själva föremålet är helt rimligt. Att de sedan även valt att lagra själva fotografiet som ett objekt i databasen kan diskuteras. Är just detta fotografi av ett sådant värde att det skall sparas som fotografi också. Det stora problemet som upptäcktes är att bilden faktiskt lagrats flera gånger. Bilden borde bara ha lagrats en gång och sedan skulle båda databasobjekten länkats till samma bild.

Att vissa institutioner väljer att ladda upp en bild med texten ”Bild Saknas” är ytterligare ett problem som uppmärksammats. Det förfarandet gör att sökningar i databasen med frågor efter endast objekt som har bild blir verkningslösa. Uppdragsgivaren har tack vare projektet blivit uppmärksammade på dessa problem.

Att ett informationssystem innehåller korrekt information står i direkt relation till systemets framgång (Delone & McLean, 2003). Därför är det problematiskt att vissa institutioner lagrar data på ett felaktigt sätt i databasen.

5 Diskussion och slutsatser

Detta projekt har visat hur svårt det kan vara att genomföra korrekthet inom maskinlärning. Flera modeller har skapats och två träningssätt har genomförts för att skapa de valda modellerna i resultatet. Att modeller är svårt att skapa beror på flera faktorer. En av faktorerna är att om uppgiften är generell kan det vara svårare att få en korrekt modell. I detta fall skulle objekts letas efter men vad exakt är ett objekt? Här är det mänskliga faktorn som måste bestämma och då kan det skapa fel i modellen, ett exempel är att vår modell plockar ut dokument som objekt detta då det ansågs dokument som objekt. Träningsdata är även en viktig faktor vid modellträning då det utgör grunden för vad modellen ser. I vårt fall krävdes inga speciella ändringar i träningsdatan förutom en genomgång manuellt där felaktiga bilder togs bort, eftersom efter några bildhämtningar märktes det att metadata i K-samsok inte alltid var korrekt. Detta försvårade modellträningen ännu mer då mänskliga faktorn kan skapa okorrekt träningsdata. En annan del är att träningsdata behöver vara så stor som möjligt för att ge en så korrekt bild av vad den ska plocka ut, hur större data är desto svårare blir det att kolla igenom manuellt. De skapade modellerna fokuserar även på alla institutioner något som gör att träningsdata måste hämtas från alla institutioner för att ge modellen en så generell bild som möjligt. Det skulle nog varit bättre att fokuserat på en institution åt gången, då vårt resultat skiljer sig mellan modellerna beroende på institution se bilaga H. Träning av modeller går alltid att förbättra till en bättre procent men tar dock tid. Däremot finns det genvägar som att omskola en modell, detta i vårt fall gav ett bättre resultat. Dock bör man vara varsam vid val av träningssätt då en omskolad modell ger bra resultat på kort sikt och en egen modell bättre på lång sikt (Google, u.d.). Detta kan även ses i vårt resultat där vår omskolade modell plockar ut växter något som vår egentränade modell inte gör. Stickprovs resultatet visar att alla institutioner som testas har fel i sin metadata se bilaga I. Något som kan bero på att modellen fel klassificerar dock bör det tilläggas att manuellt test mot känt material har genomförts utöver TensorFlows valideringstest se bilaga F. Detta ger oss synen att modellerna validerar korrekt inom en viss procent, därav borde de alla institutioner inneha felaktiga metadata som bör ses över.

5.1 Slutsatser

Att skapa en modell för den givna uppgiften är möjligt men kan vara svårt då de givna faktorer som träningsdata, metadata och mänskliga faktorn spelar stor roll. Eftersom mänskliga faktorn spelar roll bör uppgiften vara så specifik som möjligt då det förenklar träningen av en modell. För att modellen ska vara så korrekt som möjligt bör även varje institution ha en egen modell då hur träningsdata ser ut kan variera mellan institutioner. Kvalité av metadata och foton varierar även från institutioner och varje institution bör ses över om det går att applicera maskinlärning på deras data. Uppgiften är genomförbar men för att få en så korrekt modell som möjligt krävs kunskap inom träningsmaterialet och maskinlärningsområdet. Utifrån stickprovernas valideringsresultat som är genomfört visar att alla institutioner har någorlunda fel metadata som bör förbättras. Därav görs slutsatsen att institutioner bör förbättra sin metadata och maskinlärning kan vara till hjälp av detta.

5.2 Fortsatt arbete

Vi rekommenderar att Riksantikvarieämbetet jobbar vidare inom området maskinlärning dock mer med en specifik områdesinriktning till exempel en modell som hittar dokument. Att inrikta sig på en institution åt gången med maskinlärning skulle även vara en rekommendation men förståeligt om det inte går att genomföra då det kan vara kostsamt. En utvärdering av varje institution metadata kan även genomföras för att se om maskinlärnings skulle vara applicerbart. Utöver det skulle vi rekommendera att dela upp vår modell i flera mindre modeller och jobba vidare med att förbättra den så gott de går. Om endast en estimering av varje institutions metadata skall genomföras rekommenderar vi att träna på en redan färdig vektormodell som vi gjort i denna uppgift. Men om institutioner vill fortsätta med den givna uppgiften i detta dokument rekommenderar vi att träna en modell från grunden på mer träningsmaterial som är balanserat och specifikt för aktuell institution.

5.3 Egna reflektioner

Texten nedan hänvisas till loggböckerna och bilaga J. Under projektets gång har vi testat på att utveckla ett riktigt projekt i större miljö. Detta i sin tur har visat sig väldigt problematiskt då mycket problemlösning har krävts och planering. En stor del som kunde undvikits med bättre

metoder och utvecklingsätt. En annan stor del är att ett större projekt krävs väldigt mycket kommunikation detta i sin tur har fått oss att inse att i verkliga världen kan det vara problem att utveckla en större programvara då det är många rörliga delar i ett projekt. Att ge sig in i ett nytt programmeringsspråk har också visat sig problematisk då det kan vara svårt att implementera programmerings design principer. Att kanske även ta mer hjälp av handledarna kan ha varit positivt, detta är nog det främsta vi hade gjort annorlunda. Mycket tid lades även på att hitta källor angående ämnet något som kanske kunde minskat. Projektet har varit väldigt lärorikt och lärt oss en del hur vår kunskap går att applicera i verkliga världen. Projektet enligt oss har varit lyckat

Källförteckning

Abadi, M. o.a., 2016. *TensorFlow: A System for Large-Scale Machine Learning.* Savannah, Usenix.

Chari, R. S., 2018. Image Recognition and Study of Hyperparameter Optimization of Convolutional Neural Networks Using TensorFlow and Keras Frameworks, Masteruppsats, California State University, Fresno. Tillgänglig: http://repository.library.fresnostate.edu/bitstream/handle/10211.3/203026/ShyamChari_csu_6050N_10544.pdf?sequence=1 [Använd 08 04 2019]

Delone, W. H. & McLean, E. R., 2003. The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. *Journal of Management Information Systems*, 19(4), pp. 9-30.

Digisam, 2014. *Vägledande principer för arbetet med digitalt kulturarv.* [Online]
Available at: http://www.digisam.se/leveranser/rapporter/
[Använd 28 03 2019].

Digisam, 2015. *Digitalisering av kulturarvet - nuläge och vägvalsfrågor.* [Online]
Available at: http://www.digisam.se/wp-content/uploads/2013/05/Digitalisering%20av%20kulturarvet_nulage_och_vagvalsfragor.pdf
[Använd 30 03 2019].

Google, u.d. *TensorFlow:Custom training: basics.* [Online]
Available at: https://www.tensorflow.org/tutorials/eager/custom_training
[Använd 15 05 2019].

Google, u.d. *Tensorflow:Explore overfitting and underfitting.* [Online]
Available at: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit
[Använd 15 05 2019].

Google, u.d. *TensorFlow:How to Retrain an Image Classifier for New Categories.* [Online]
Available at: https://www.tensorflow.org/hub/tutorials/image_retraining
[Använd 28 05 2019].

O'Shea, K. & Nash, R., 2015. *An Introduction to Convolutional Neural Networks,* u.o.: eprint arXiv:1511.08458.

Provost, F., 2000. *Machine Learning from Imbalanced Data Sets 101.* Menlo Park, AAAI Press.

Riksantikvarieämbetet, u.d. *Kringla - visualiserat.* [Online]
Available at: https://riksantikvarieambetet.github.io/Kringla-Visualized/tidslinje/#skip
[Använd 17 04 2019].

Riksantikvarieämbetet, u.d. *Om K-samsök.* [Online]
Available at: http://www.ksamsok.se/om-k-samsok/
[Använd 17 04 2019].

Riksantikvarieämbetet, u.d. *Om Riksantikvarieämbetet.* [Online]
Available at: https://www.raa.se/om-riksantikvarieambetet/
[Använd 01 04 2019].

Scwaber, K. & Beedle, M., 2001. *Agile Software Development with Scrum.* Upper Saddle River: Prentice-Hall Inc..

Simard, Y. P., Steinkraus, D. & Platt, J., 2003. *Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis,* Redmond: Institute of Electrical and Electronics Engineers, Inc..

TensorFlow, u.d. *Get started with Tensorflow.* [Online]
Available at: https://www.tensorflow.org/tutorials/
[Använd 09 04 2019].

TensorFlow, u.d. *Hosted models.* [Online]
Available at: https://www.tensorflow.org/lite/guide/hosted_models
[Använd 09 04 2019].

Zoph, B., Vasudevan, V., Shlens, J. & Quock, L. V., 2018. *Learning Transferable Architectures for Scalable Image Recognition.* Salt Lake City, IEEE.

Bilaga A: Scrumboard

Sprint - Programkod som krävs för projektet - 1

Name	Assignee	Status	Priority	Estimation
Hämtning av bilder från K-samsök med kriterierna 1.Antal bilder 2.Vad för typ av bild (fotografi/objekt etc) 3.Val av institutioner bilderna får komma ifrån	Daniel Persson	Done	High	2
Skapa en funktion som hämtar ner alla id för bilderna för varje bild-hämtning och lägger de i en array	Petter Gullin	Done	High	0,5
Funktion som skapar en csv fil utifrån skapade arrayer med bildinformation	Daniel Persson	Done	High	0,7
Skapa funktion som sparar ner bilden från en url i en ny mapp	Petter Gullin	Done	High	0,5
Hämtning av randomiserade bilder från K-samsök med kriterierna 1.Antal bilder	Daniel Persson	Done	Medium	0,5
Funktion att kolla bilden inte hämtas hem tidigare (Detta kollas från skapade csv filer)	Daniel Persson	Done	Medium	1
text GUI	Daniel Persson	Done	Low	0,3
				5,5

Sprint - Skapande av TensorFlow modeller- 2				
Name	**Assignee**	**Status**	**Priority**	**Estimation**
Fixa och rensa kod från sprint 1	Daniel Persson	Done	High	1
Formatera hämtade bilder för TensorFlow	Daniel Persson	Done	Low	1
Ladda in Googles färdiga modeller och kör de på hämtade bilder	Daniel Persson	Done	High	1
Skapa TensorFlow modell	Daniel Persson	Done	High	2
Justera TensorFlow modell	Daniel Persson	Done	Best effort	3
Lära sig TensorFlow	Daniel Persson	Done	High	1
Lära sig TensorFlow	Petter Gullin	Done	High	1
Börja på visualisering av Tensor-Flow	Petter Gullin	Done	High	1
				11

Sprint - Upprensning av kod och uppdatering av TensorFlow modeller samt testning av Tensor-Flow modeller - 3				
Name	**Assignee**	**Status**	**Priority**	**Estimation**
Funktion som jämför tagg med resultat från TensorFlow	Petter Gullin	Done	High	1
Skriv in TensorFlow information i csv	Petter Gullin	Done	High	1
Att få TensorFlow att mata in hämtade bilder	Petter Gullin	Done	Medium	1
Testa modeller	Petter Gullin	Done	Medium	1

Få bildprogrammet färdig - Fixa och rensa upp kod	Daniel Persson	Done	High	1
Få TensorFlow programmet färdig - Fixa och rensa upp kod	Daniel Persson	Done	High	1
Hämta hem material och skapa nytt träningsmaterial	Daniel Persson	Done	Medium	1
Skapa modeller från nytt tränings-material med justeringar	Daniel Persson	Done	Medium	4
				11
Backlog				
Name	**Assignee**	**Status**	**Priority**	**Estimation**
Visualisering av TensorFlow informationen	Petter Gullin	On hold	Low	7
Justera Googles TensorFlow modeller	Daniel Persson	On hold	Low	3
				10

Bilaga B: Tidsplanering

PM 1 schema

Name	Timeline - Start	Timeline - End
Beskrivning av uppdrag (kapitel 1)	2019-03-26	2019-04-02
Litteraturgranskning för projektet	2019-03-27	2019-04-09
Revidering av text	2019-04-03	2019-04-10
Beskrivning av kunskapsbas (kapitel 2)	2019-04-02	2019-04-10
Förberedelse presentation och opponering	2019-04-11	2019-04-12
Redovisning PM1	2019-04-15	2019-04-15

Teknisk schema

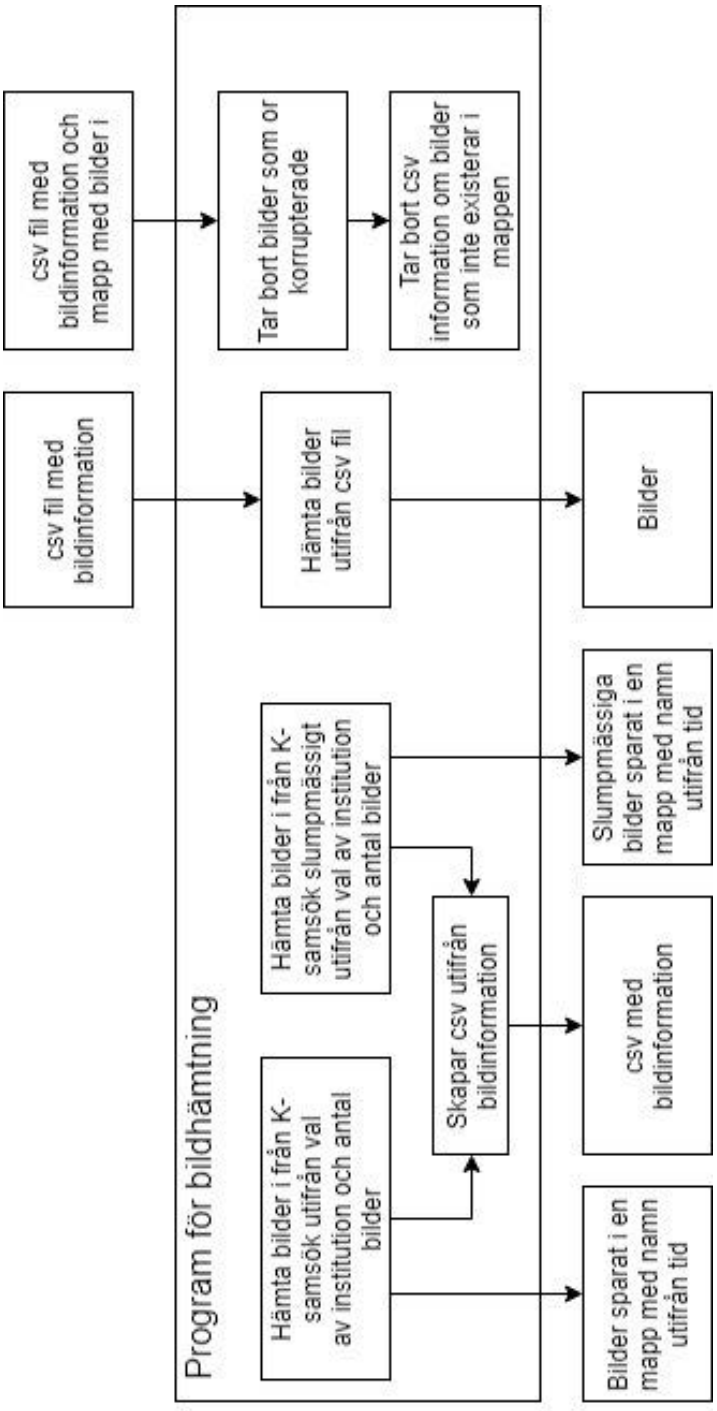
Name	Timeline - Start	Timeline - End
Bekanta sig med Pythons syntax samt TensorFlow	2019-03-27	2019-03-31
Hämta träningsmaterial till modell från K-samsök	2019-04-05	2019-05-05
Gå igenom insamlat material	2019-04-14	2019-05-10
Träna Googles färdiga TensorFlow modell	2019-05-01	2019-05-10
Skapa egen TensorFlow modell	2019-04-07	2019-05-10
Skapa bildhämtnings program	2019-04-02	2019-04-24
Skapa TensorFlow tränings program	2019-04-05	2019-04-22
Skapa validerings program	2019-05-13	2019-05-16
Publicering av koden på Github	2019-05-20	2019-05-25

PM 2 Schema

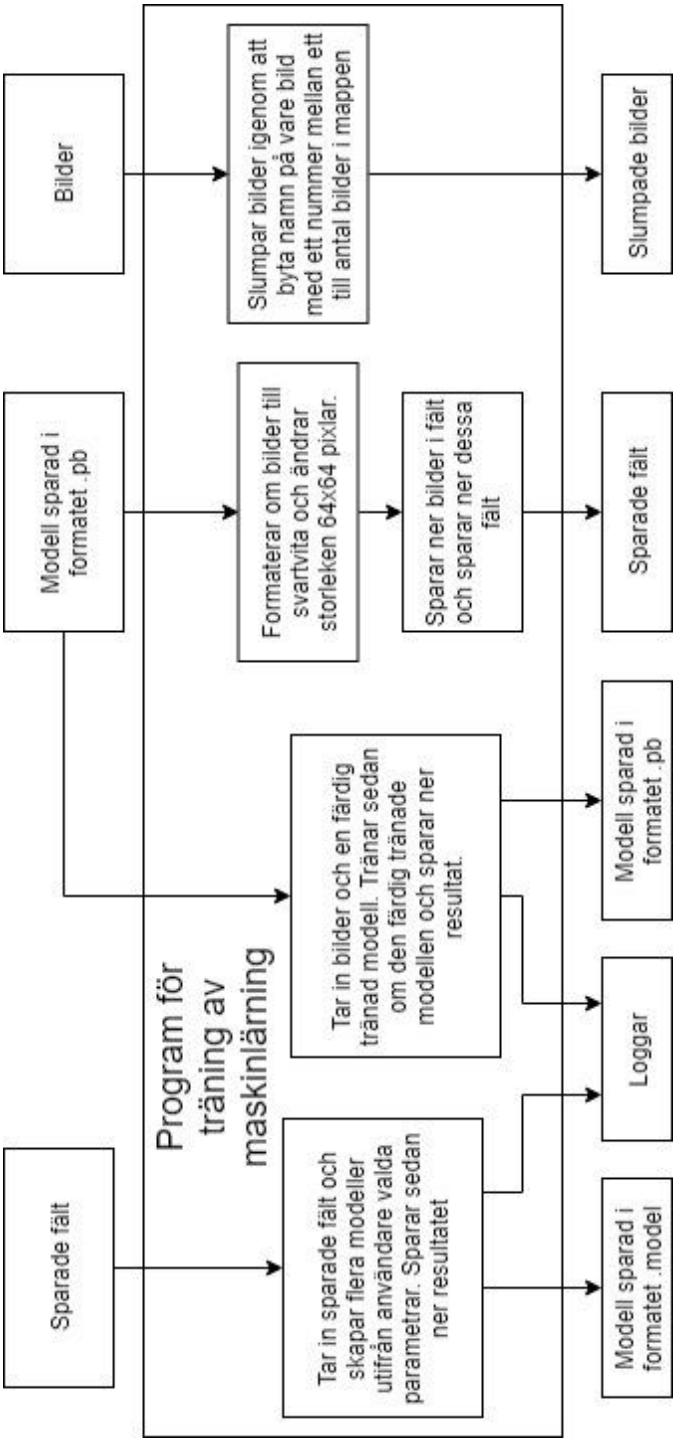
Name	Timeline - Start	Timeline - End
Beskrivning av resultat, programvara och design (kapitel 3)	2019-04-17	2019-05-14
Beskrivning av utvärdering kapitel (4)	2019-04-30	2019-05-14
Beskrivning av diskussion och slutsatser (kapitel 5)	2019-04-22	2019-05-14

Förberedelse presentation och opponering	2019-05-15	2019-05-17
Redovisning PM2	2019-05-20	2019-05-20
Färdigställande av rapport		
Name	**Timeline - Start**	**Timeline - End**
Färdigställa rapport med synpunkter från opponering	2019-05-21	2019-05-31
Förberedelse presentation	2019-05-30	2019-06-03
Muntlig redovisning	2019-06-04	2019-06-04

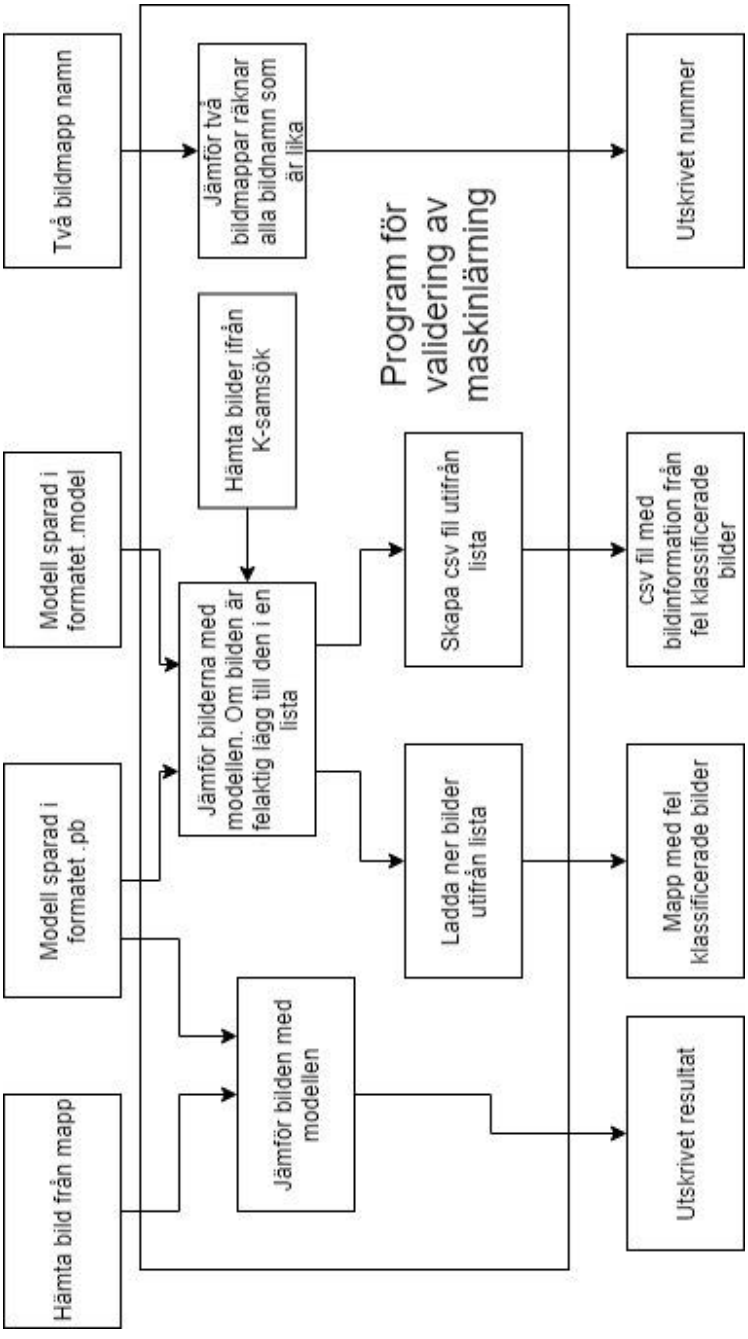
Bilaga C: Program för bildhämtning funktionalitet



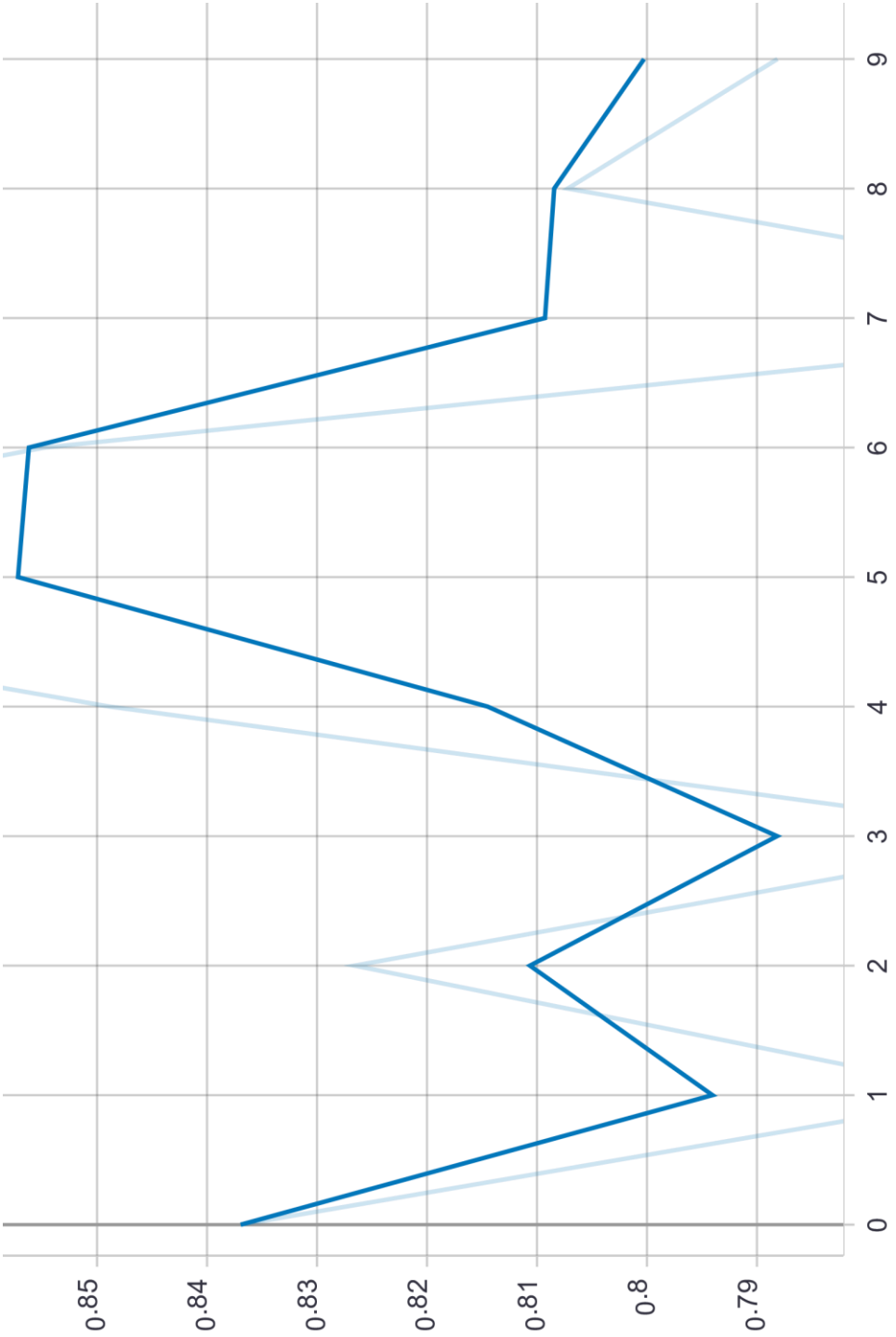
Bilaga D: Program för träning av maskinlärning



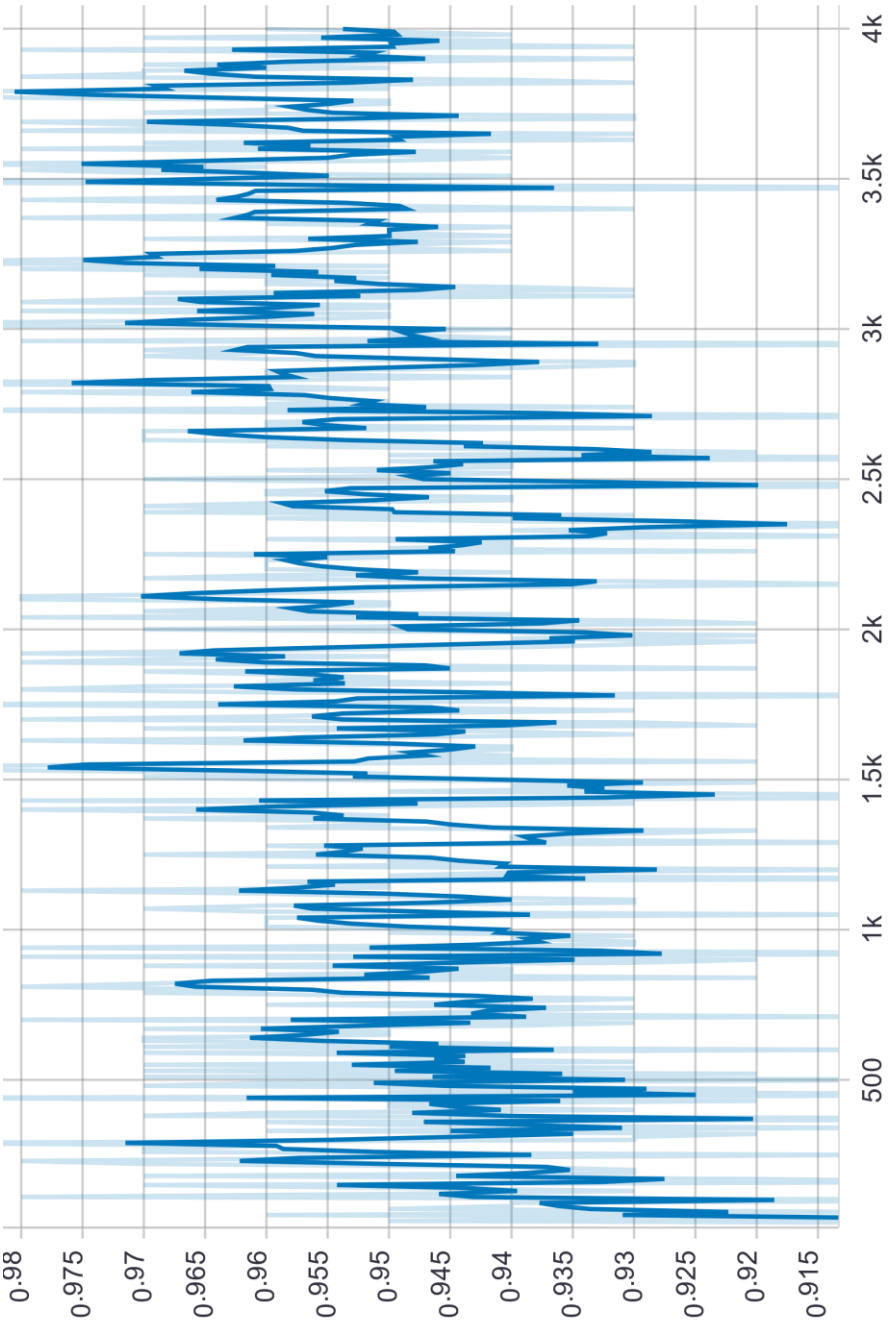
Bilaga E: Program för validering av maskinlärning



Bilaga F: Statistik från egen tränad TensorFlow modell



Bilaga G: Statistik från omskolad modell ”NasNet-A(Large) version 3 feature vector”



Bilaga F: Stickprov mot känt material

Stickprov mot känt material				
Bildmängd	Typ av objekt	Modeller	Antal fel klassificerade	Procentuellt rätt modellen har mot känt material
220	Föremål	***Egen tränad TensorFlow modell***	30	86%
		NasNet-A(Large) version 3 feature vector	10	95%
220	Foto	***Egen tränad TensorFlow modell***	29	87%
		NasNet-A(Large) version 3 feature vector	9	96%

Bilaga H: Stickprovs statistik utifrån validering mot K-samsök

Museum	**Antal bilder i samling**	**Fel validerade bilder i samling** *"Egentränad Tensor-Flow modell"*	**Fel validerade bilder i samling** *"NasNet-A(Large) version 3 feature vector"*
Skansen	317	52	29
Teleseum	1061	499	236
Enköpings museum	1658	513	427
Östgötalands museum	2673	507	224
Vasamuseet	4286	749	760
Armémuseum	12 443	7806	6576

Bilaga I: Sammanställning av stickprovs resultat

Museum	**Antal bilder i samling**	**Fel validerade bilder i samling som före-kommer i båda modeller**	**Fel validerade bilder i samling som före-kommer i båda modeller i procent**
Skansen	317	23	7%
Teleseum	1061	199	19%
Enköpings museum	1658	390	24%
Östgötalands museum	2673	188	7%
Vasamuseet	4286	389	9%
Armémuseum	12 443	4370	35%

Bilaga J: Gruppens insats

Gruppens insats

Detta dokument syftar till att ni som arbetar tillsammans ska reflektera över hur ni fördelat arbetet er emellan. Orsaken till att ni ska göra det är att betygen som måste graderas sätts individuellt och då ni till största delen gör examensarbetet på egen hand, d.v.s. utanför fakulteten och utan att er handledare på fakulteten är närvarande. Följaktligen, behövs det ett underlag för bedömningen. Det behövs inte att ni har gjort exakt samma uppgifter och arbetat samtidigt, utan det handlar om den totala insatsen. Om ni svarar "nej" på en eller båda påståendena nedan så bestämmer vi ett möte där vi reder ut skillnaderna i arbetsfördelningen.

Vi som arbetat tillsammans är: Daniel Persson
Petter Gullin

Projektens namn: Validering av objekttyp med maskin-inlärning.

Vi bedömer att vi har fördelat arbetet lika

Ja	Nej
☒	☐

Vi bedömer att vi deltagit lika i arbetet

Ja	Nej
☒	☐

Kommentarer kring fördelningen av arbetet
Daniel har haft större fokus på kodning och Petter på rapport-skrivning. Båda har varit delaktiga i alla delar och totalt bidragit lika mycket.

Underskrifter: Daniel Persson
Petter Gullin