Where is the basement? - Image recognition of basement windows on 'street view' images

Marcus Lindell Biehl

Master's thesis
2022:E15

Lund Institute of Technology
Lund University

Centre for Mathematical Sciences
Mathematics

Centrum Scientiarum Mathematicarum

Var finns källaren?

- Bildigenkänning av källarfönster på 'street view'-bilder

Av

Marcus Lindell Biehl

Sammanfattning

Det här examensarbetet hade som mål att ta reda på om neuronnätverk kan användas för bildigenkänning av 'street-view'-bilder för att avgöra om en byggnad har eller inte har en källare. För att göra detta hämtades en datamängd av bilder från Google Street View utifrån en lista med adresser som på förhand hade inventerats av VA SYD. Från filen med bilder på fastigheter som innehar källare valdes det sedan ut de bilder som uppvisade källarefönster så att en diskret klassificering kunde göras. Ett förtränat neuronnät användes som bas till att träna på datamängderna i tre olika körningar, där den varierande parametern var olika typer av data augmentationer. Resultatet gav cirka 90% noggrannhet på testdatan för alla tre körningar med varierande träningsprestanda för de olika data augmentationerna. Detta visar att med bildigenkänning av 'street-view'-bilder borde det vara möjligt att identifiera hurvida en byggnad har källare eller inte.

Nyckelord: Bildigenkänning, Förtränat neuronnät, Källare, Google Street View, Data augmentation.

Abstract

The aim of this thesis was to find out if neural networks can be used for image recognition of 'street-view' images to determine whether or not a building has a basement. To do this, a data set of images was retrieved from Google Street View based on a list of addresses that had been pre-inventoried by VA SYD. From the file with pictures of properties that hold basements, those that exhibited basement windows were then selected so that a discreet classification could be made. A pre-trained neural network was used as a basis for training the data sets in three different runs, where the varying parameter was different types of data augmentations. The result gave approximately 90% accuracy on the test data for all three runs with varying training performance for the different data augmentations. This shows that with image recognition of 'street-view' images, it should be possible to identify whether or not a building has a basement.

Keywords: Image recognition, Pre-trained neural network, Basement, Google Street View, Data augmentation.

Förord

Jag vill först av allt tacka Victor Pelin från VA SYD för hans hjälp med idéer om hur datamängderna skulle hämtas samt den lista med adresser som han tillhandahöll från företaget.

Jag vill även tacka min handledare Ida Arvidsson som har varit ett fantastisk resurs att stödja mig under arbetets gång samt för hennes korrekturläsning och kommentarer till rapporten. Jag vill dessutom tacka min examinator Kalle Åström för hans kontroll av att alla delmoment i examensarbetet är klara och att de möter kvalitetskraven.

Till sist vill jag tacka Helena Ensegård från Miljöbron Skåne för hennes hjälp med att ta kontakt med VA SYD och för hennes fortsatta intresse och stöd under arbetets gång.

Marcus Lindell Biehl, Lund, mars 2022.

Innehåll

1 Inledning

Det här examensarbetet genomfördes tillsammans med VA SYD och Miljöbron Skåne. Syftet med arbetet var främst att ta reda på om ett bildigenkänningsprogram med 'street view'-bilder från karttjänster som data kan användas för att avgöra om en byggnad har källare eller inte.

1.1 Bakgrund

VA SYD är ett kommunalförbund i sydvästra Skåne som förser hushåll med rent dricksvatten och renar avloppsvatten. VA SYD hanterar även dagvatten, som är tillfälligt förekommande rinnande vatten från exempelvis skyfall på öppna markytor, byggnader och andra konstruktioner och förser det med en väg till dagvattensystemet och vidare till åar, sjöar eller andra vattendrag.

Tillsammans med andra aktörer, håller VA SYD genom projektet Future City Flow på att utveckla ett verktyg som kan hantera så kallat tillskottsvatten i spillvattensystemet på ett strategiskt sätt. En del av detta projekt handlar om att beräkna risker för spillvatten och en faktor till detta är om huruvida byggnader har källare eller inte. Traditionellt har VA SYD gjort manuella kontroller av byggnader, vilket är enormt tidskrävande och en alternativ, mer effektiv, digital lösning som använder sig av maskininlärning borde finnas tillgängligt idag.

Att veta om byggnader innehar källare eller inte kan även tillämpas exempelvis till radioplanering för fjärravlästa vattenmätare och för kartläggning av källarnedfarter som har en ökad risk för översvämning.

Den digitaliserade metoden för att kartlägga fastigheter med källare som VA SYD vill utreda är om 'street view'-bilder i karttjänster kan användas för identifiering med hjälp av bildigenkänningsprogram.

Miljöbron är en ideell organisation vars mål är att skapa kontakter mellan näringsliv och universitet/högskola genom att förmedla och koordinera projekt inom miljö och hållbar utveckling. Det är genom Miljöbron Skåne som studenten fick reda på projektet och kontaktade VA SYD.

1.2 Syfte

Ta reda på om ett maskininlärningsprogram kan tränas till att urskilja byggnader som har eller inte har källare genom objektklassificering. Vidare, ska olika typer av *data augmentation* tillämpas på träningsdatan och resultaten av dessa träningar jämföras med varandra på testdata för att se vad som ger högst noggrannhet och prestanda för det här specifika problemet.

1.3 Målformulering

Examensarbetets mål är att undersöka om ett neuronnätverk kan tränas till att avgöra om en byggnad har en källare eller inte och om 'street view'-bilder är en lämplig källa att använda för bildigenkänning av källarlokaler.

1.4 Problemställningar

Följande frågor ska undersökas och försöka att besvaras i det här examensarbetet:

1. Kan ett bildigenkänningsprogram tränas till att identifiera källarlokaler i byggnader med hjälp av 'street view'-bilder som datamängd?
2. På vilket sätt ska neuronnätet byggas upp?
3. Ger data augmentation av 'street view'-bilderna mer robusta modeller?
4. Hur påverkar olika faktorer som exempelvis bildkvalitet och ocklusion klassificeringen av datasetet?

1.5 Avgränsningar

Begränsningar i projektet innefattar användningen av endast Google Street View för insamling av data, då studenten fann att denna tjänst gjorde det möjligt att på ett enkelt sätt hämta bilder på specifika adresser genom deras API-tjänst. Indikationer på källare har begränsats till källarfönster i datamängden, så att en diskret klassificering kan göras för maskininlärningsprogramet samt att studenten fann att det var detta objekt som gav den tydligaste särskiljande egenskapen för de två klasserna. Det är även endast adresser i Malmö som har använts i den här datamängden, då det var dessa adresser som hade efter inventering blivit bekräftat av VA SYD att ha eller inte ha källare. För mer detaljer om detta se metodavsnittet.

1.6 Tidigare liknande arbeten

'Street view'-bilder har använts som datamängd för deep learning med neuronnätverk i en mängd olika tidigare arbeten. Exempel på vad denna metodik har använts till innefattar bland annat:

- Detektering av övergivna byggnader i USA [1].
- Uppbyggnad av instansklassificerare för olika typer av byggnader som exempelvis lägenheter och kyrkor [2].
- Detektering av bilar för att härleda socioekonomiska attribut såsom inkomst, ras och valmönster i USA [3].
- Att utforska hur förares visuella miljö påverkar trafiksäkerheten [4].

2 Tekniskt underlag

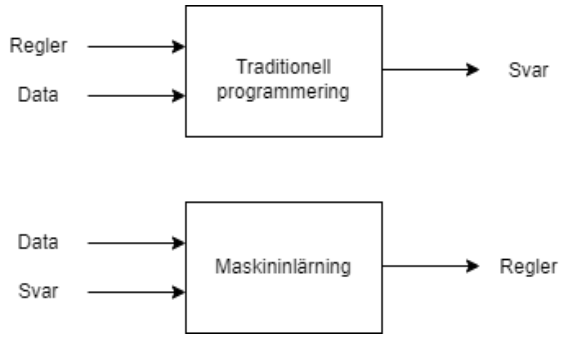
I det här avsnittet beskrivs de tekniska verktyg som har använts i examensarbetet samt de bakomliggande teorierna.

2.1 Artificiell intelligens

Forskning i artificiell intelligens (AI) påbörjades på 1950-talet av Alan Turing, som en härledning från matematisk logik. Turing gav idéen att man skulle kunna använda sig av enkla symboler som ettor och nollor till att simulera vilken logisk slutsats som helst. Därmed skulle man med andra ord kunna göra så att en maskin kan tänka[5].

2.2 Maskininlärning

Som ett underfält till artificiell intelligens, började maskininlärning att florera på 1990-talet och har fortsatt att växa i samband den ökade tillgängligheten av snabbare hårdvara och större datamängder[6]. Till skillnad från den traditionella typen av programmering, där en utvecklare skriver in regler till hur den ingående datan ska hanteras, ger utvecklaren istället de förväntade svaren till den ingående datan, och reglerna byggs därefter upp (se figur 1). Dessa regler kan sedan användas på ny data för att ge svar, självständigt från de förväntade svaren. Därför säger man att ett maskininlärningsprogram *tränas* istället för att programmeras[6].



Figur 1: Skiss över skillnaden mellan traditionell typ av programmering och maskininlärning.

Metoderna som används för att bygga upp reglerna kräver indata, förväntad utdata samt ett sätt att mäta om modellen gör ett bra jobb. Det sistnämnda är nödvändigt för att mäta skillnaden mellan algoritmens nuvarande utdata och den förväntade utdatan. Detta mått används som en återkopplingssignal så att algoritmen kan justeras och förbättras efter varje iteration som modellen uppdaterar reglerna.

Idag används maskininlärning på en mängd olika områden som exempelvis röstigenkänning i Google- och Apple-produkter[7], Teslas självkörande bilar[8], filmrekomendationer i Netflix[9], etc. En aktuell användning var år 2020 då maskininlärning användes för att ställa diagnoser och hjälpa forskare att utveckla botemedel mot COVID-19[10]. I det här examensarbetet användes bildigenkänning som är en av de vanligaste applikationerna för maskininlärning, som exempelvis Facebooks igenkänning av ansikten i foton[11].

2.2.1 Tränings-, valideringss- och testmängd

Då maskininlärningsmodeller utvärderas delas den tillgängliga datan oftast upp i tre olika delmängder; tränings-, valideringss- och testmängd. Modellen tränas upp på träningsdatan och utvärderas på valideringsmängden. Efter färdiggjord träning så görs en sista utvärdering på testdatan.

Anledningen till att datamängden delas upp i tre delmängder istället för två; tränings- och testmängd, är att olika parametrar justeras under uppbyggnad av modeller för maskininlärning, exempelvis antalet lager. Denna justering görs beroende på hur valideringsmängden utvärderas och om detta görs för

mycket kan det leda till *överträning* på valideringsmängden, vilket betyder att modellen anpassar sig för mycket till valideringsmängden och inte presterar lika bra på ny data. Därför används en separat testmängd för att detta fenomen ska kunna upptäckas och en mer generell modellen ska kunna byggas upp.

2.3 Deep learning med neuronnät

En specifik gren inom maskininlärning kallas för *deep learning*. Termen är inte en referens till att någon djupare förståelse uppnås med den här metoden, utan står för användningen av successiva lager för inlärning. Antalet lager beskriver hur 'djup' modellen är[6].

De flesta *deep learning* modeller är så kallade *neuronnät*, vilket är en term som refererar till hjärnans biologiska struktur med enkla neuroner som är sammankopplade och tillsammans kan lösa komplexa problem. Det är här som analogin slutar dock. Deep learning med neuronnät kan snarare ses som ett filtersystem där varje filter är en neuron som tar indata och transformerar den med hjälp av så kallade *vikter*, som är olika inlärda värden i ett specifikt lager av neuronnätet. Dessa vikter multipliceras med indatan, adderas och skickas till en aktiveringsfunktion som ger utdata från neuronen. Den givna datan transformeras i nätverket så att den alltmer representerar målvärdena och allt mindre representerar indatan. Målet med inlärningen är att finna de optimala värdena på dessa vikter i alla lager så att så mycket som möjligt av den givna indatan slutligen ger de rätta målvärdena.

För att bygga upp ett neuronnät krävs även en förlustfunktion och en optimeringsfunktion. Det är förlustfunktionen som fungerar som återkopplingssignal och visar därför hur bra nätverket presterar. I klassificeringsproblem kan exempelvis *binary* eller *categorical crossentropy* användas för detta ändamål i två eller flera klasser. Förlustfunktionen används sedan i optimeringsfunktionen för att bestämma hur nätverkets vikter ska uppdateras så att förlustvärdet minskar för varje iteration.

2.3.1 Utvärdering av neuronnät

Utvärdering och jämförelse av olika modeller av neuronnät görs som tidigare nämnt på en separat testmängd, som är till för att generalisera modellerna så att de även presterar bra på okänd data. Vad som utvärderas är bland annat hur stor del av testdatan som nätverket klassificerar rätt, vilket anges i en procentsats som kallas noggrannhet. Man mäter även förlustvärdet på testdatan, vilket kommer från förlustfunktionen och anger hur långt ifrån det svar som nätverket beräknade är från det faktiska värdet. Utöver dessa mätvärden redogörs det även i binära klassifikationsproblem ofta om sensitiviteten och specificiteten. Sensitiviteten beskriver sannolikheten att få ett positivt resultat givet att det positiva resultatet stämmer. Specificiteten beskriver sannolikheten att få ett negativt resultat givet att det negativa resultatet stämmer.

Med sensitivitet och specificitet för all testdata kan en så kallad ROC-kurva (receiver operating characteristic curve) ritas upp som plottar sensitiviteten mot 1 - specificiteten då tröskelvärdet, det vill säga värdet som gränsar mellan de olika klassificeringarna, varieras. För att jämföra olika modeller kan sedan arean under dessa kurvor (AUC) beräknas och den modell som ger högst AUC sägs då ha bäst prestanda.

2.3.2 Neuronnät för bildigenkänning

Problem inom bildigenkänning handlar om att avgöra om bilddata innehåller en specifik typ av objekt, aktivitet eller andra utmärkande egenskaper. De för nuvarande bästa typerna av neuronnätverk för sådana problem har så kallade faltande neuronnätverk (CNN) som bas.

Ett faltande neuronnät är en klass av neuronnät som ofta används för problem med visuella representationer. De består till största del av flera faltande lager där ett lager plockar ut alla förekomster av specifika lokala egenskaper i bilden, exempelvis hörnet av ett objekt. Två av de viktigaste egenskaperna i ett CNN är att translation inte har någon betydelse, det vill säga att den specifika egenskapen kan dyka upp var som helst på bilden, samt att det för varje lager byggs upp större mönster baserat på de mindre mönstren i de föregående lagren[6]. De faltande lagren har tredimensionella tensorer som

in och utdata. Som input för bilddata, exempelvis, beskriver de två första axlarna höjd och bredd på bilden och den tredje beskriver färginnehåll.

Efter det faltande lagret används så kallade *pooling* lager vilket har som roll att välja ut de viktigaste egenskaperna som det faltande lagret har lärt sig genom någon specifik operation (oftast max operationen). Detta minskar således storleken på datan.

Då datan har filtrerats genom flera av dessa faltande och pooling lager används typiskt ett *flattening* lager som konverterar den tredimensionella datan till en endimensionell vektor. Till sist används ett *fully connected layer* som ger den slutgiltiga klassificeringen.

För att minska risken för överträning kan ett *dropout*-lager även läggas till efter de faltande och pooling lagren, vilket innebär att den liten delmängd av det modellen precis har lärt sig slumpmässigt avfärdas. Anledningen till att detta görs är så att nätverket inte ska lära sig onödiga mönster som inte bidrar till klassificeringen.

2.3.3 K-delad korsvalidering

När ett neuronnätverk tränas med en liten datamängd, och därmed en liten validationsmängd, kan valideringsresultaten ändras mycket beroende på vilken delmängd av datan som används till validering, det vill säga att det blir hög varians. Utvärderingen av modellen blir därför inte tillförlitlig.

I examensarbetet var det en sådan situation, och på grund av detta användes K-delad korsvalidering. Det innebär att den tillgängliga datan, förutom testdatan, delas upp i K delar och K identiska modeller tränas med en del som valideringsmängd och resterande delar som träningsmängd vid varje iteration. Valideringsresultaten för modellen blir då det genomsnittliga resulatet för de K uppdelningarna.

2.4 Implementering i Python

I det här examensarbetet användes Python som programmeringsspråk till att hämta 'street view'-bilder och för uppbyggnad av träningsmodell som kan identifiera källarfönster i bilderna. Python är idag är det mest använda språket för maskininlärning, på grund av dess lättillgänglighet till en mängd populära verktyg som exempelvis numpy, scipy, matplotlib, sklearn, TensorFlow och Keras som hjälper med bland annat implementering, träning, analys och gränssnitt för neuronnät [6].

2.4.1 TensorFlow

TensorFlow är en open source plattform för maskininlärning utvecklat av Google Brain, en forskningsavdelning på Google för deep learning. Det har stabila API:er, verktyg och bibliotek för framför allt Python och C++, men kan även användas i andra programmeringsspråk som Java och Javascript [12].

Koden för TensorFlow kan köras både i vanliga processorer (CPU) och grafikprocessorer (GPU). När den körs i CPU är TensorFlow själv ett lågnivå bibliotek för tensoroperationer, vilket kallas Eigen. På GPU körs den med hjälp av Nvidia Cuda Deep Neural Network (cuDNN) [6].

2.4.2 Keras

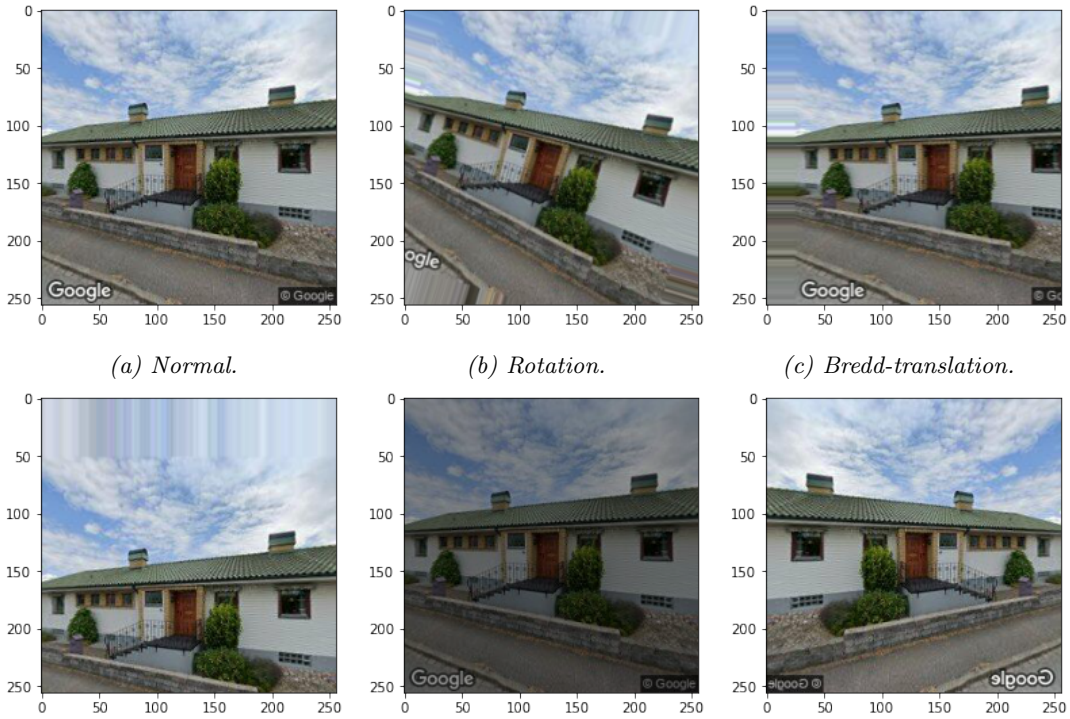
För enklare läsbarhet av kod och användarvänlighet i projekt inom deep learning med neuronnät skrivs koden på ett högnivåramverk som kallas Keras. Med hjälp av Keras kan lager i neuronnätet definieras direkt, vilket förenklar uppbyggnaden av modeller för neuronnät. Processen för utveckling i Keras ser typiskt ut på följande sätt och har använts i det här arbetet [6]:

1. Definiera indata och förväntad utdata.
2. Definiera nätverk med lager (modell) som mappar från indata till utdata.
3. Välj förlustfunktion, optimerare och mätvärden som ska övervakas.
4. Iterera genom träningsdatan med `fit()` metoden.

2.4.3 Data augmentation

Med en liten datamängd kan det även bli problem med överträning, då den upptränade modellen inte har möjlighet att generaliseras till ny data eftersom den inte ser alla möjliga aspekter av data som kan dyka upp. För att undvika detta behövs det skapas mer variation i den tillgängliga träningsdatan, vilket kan göras med data augmentation.

I Keras kan detta göras på inlästa bilder med hjälp av `ImageDataGenerator` metoden som kan göra en mängd slumpmässiga transformationer som ger trovärdiga alternativa datamängder. Dessa transformationer är bland annat rotation, translation, beskärning, zoomning, ljusändring och spegelvändning. Exempel på data augmentationer kan ses nedan i figur 2.



(a) Normal. *(b) Rotation.* *(c) Bredd-translation.*

(d) Höjd-translation. *(e) Ljusändring.* *(f) Spegelvändning.*

Figur 2: Exempel på data augmentationer av en 'street-view'-bild.

2.4.4 Förtränat nätverk - VGG16

Ett förtränat neuronnätverk är något som ofta används i bildigenkänning med en liten datamängd av bilder. Det är ett nätverk som har tränats på en stor datamängd med flera bildklasser vars vikter har sparats undan för användning på nya nätverk. För stora datamängder med hög generalisering kan de inlärda egenskaperna i det förtränade nätverket vara användbara för flera olika typer av objektklassificeringar inom problem i bildigenkänning, även om det är en ny typ av objekt som ska klassificeras som inte ursprungligen gjordes. Detta görs genom att låsa de första lagren i nätverket så att dessa inte tränas och öppnar upp de sista lagren för träning på datamängden för det specifika problemet [6].

Det förtränade nätverket som användes i examensarbetet var VGG16 [14], som kan hämtas direkt i Keras såväl som andra förtränade nätverk [15]. Modellen använde sig av en delmängd av *ImageNet* som tränings- validerings och testmängd, vilket var totalt över 1,4 miljoner högupplösta bilder tillhörande cirka 1000 olika klasser som har skalats ned till en 256 $\times$ 256 storlek [6].

2.4.5 Google Street View Static API

För att hämta den relevanta datamängden i examensarbetet användes Google Street View Static API. Det är ett verktyg för att hämta stora datamängder av statiska Google Street View bilder med en enkel HTTP-begäran, som exempelvis kan implementeras med en Python klass [16]. Parametrar för bildernas storlek, adress/koordinater och placering skrivs direkt in i URL adressen tillsammans med en unik API nyckel. För att få en API nyckel krävs ett Google Cloud Platform konto. Varje HTTP-begäran med API nyckeln har en liten kostnad, men tjänsten har även en gratis testversion som kan användas de första 30 dagarna [17]. När 'street view'-bilderna hämtas med API:n är den nödvändiga erkännandet till Google redan inbakat och ingen ytterligare kreditering behövs [18].

3 Metod

För att besvara målformuleringen och angripa problemställningarna delades arbetet upp i tre huvudfaser:

1. Insamling av datamängder.
2. Uppbyggnad av träningsmodell.
3. Träning och utvärdering av modell med varierande parametrar.

Under arbetets gång utfördes det parallellt med detta informationssökning för att realisera de tre huvudfaserna och bemöta de problem och frågor som uppstod.

I det här avsnittet beskrivs det hur datamängden samlades in, hur träningsmodellen byggdes upp samt vilka parametrar som varierades mellan de olika körningarna och varför. Det nästkommande avsnittet visar resultat för de olika körningarna på träningsmängden samt utvärdering på testmängden med de varierande parametrarna.

3.1 Insamling av datamängder

Den datamängd som har använts i arbetets maskininlärningsmodell kommer ursprungligen från excellistor med adresser som tillhandahölls av Victor Pelin från VA SYD. Listorna innehöll adresser på fastigheter som efter inventering på fält har konstaterats att antingen ha eller inte ha källare. Det fanns 2361 adresser med källare och 639 adresser utan källare. Dessa listor sparades som två `.csv`-filer för att sedan kunna användas i Python.

För att söka upp och ladda ned 'street-view'-bilderna från dessa adresser skapades en klass i Python som implementerade Google Street View Static API med hjälp av `request` paketet[16]. De två `.csv`-filerna importerades till `list` strukturer i Python, där varje element i listan är en string-typ med en specifik adress, som sedan kunde hämtas och sparas i filformatet JPG. Storleken på bilderna valdes till 256×256, så att alla bilder var lika stora och nedladdningen inte var alltför stor och tidskrävande. Av de 2361 adresser med källare och 639 adresser utan källare var 2346 och 637 adresser tillgängliga i Google Street View, respektive. Bilderna på fastigheter med respektive utan källare sparades i separata filer och kan ses i Appendix A.

På grund av den varierande kvaliteten på 'street view'-bilder, var det många av de insamlade bilderna som inte skulle ge en bra klassificering i modellen. Exempel på dessa kan ses nedan i figur 3, där två av de hämtade bilderna är på fastigheter som är skymda av buskar samt en bild som var tagen mot en gata.

Figur 3: Exempel på hämtade 'street view'-bilder som inte ger någon tillgänglig information för klassificeringen.

För att dessa avvikande bilder inte skulle påverka resultatet i klassificeringen gick studenten för hands först igenom filen med fastigheter som innehar källare, och valde ut de som hade en bra indikation i

form av källarfönster. Detta lämnade 340 bilder på fastigheter med goda indikationer på källare, som användes i tränings-, valideringss- och testmängden. På samma sätt valdes 340 bilder av den andra kategorin, så att de båda kategorierna hade lika stor datamängd.

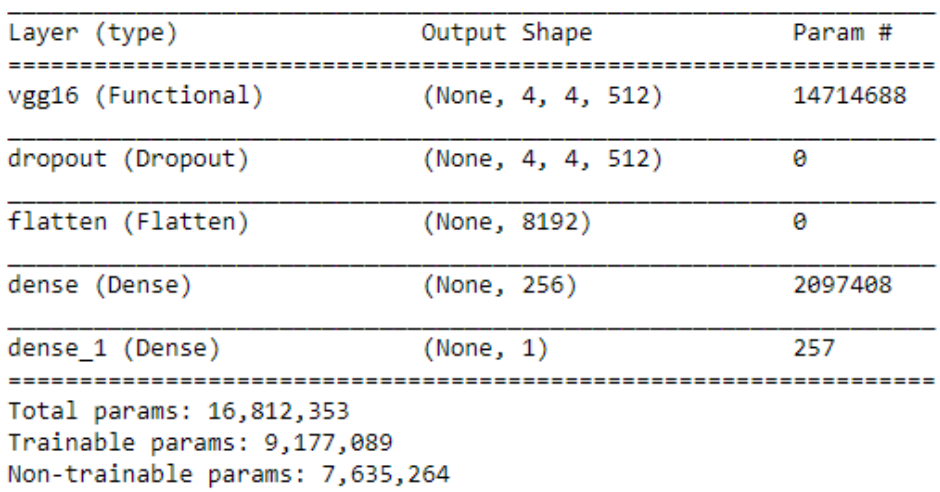
De två kategorierna av datamängder delades upp slumpmässigt med ett *random seed* i Python så att 240 bilder av varje kategori var träningsdata, 60 bilder var valideringsdata och 40 bilder var testdata. Exempel på de två kategorierna, en med och en utan källarfönster, kan ses nedan i figur 4:

Figur 4: Exempel på 'street view'-bilder i datamängden från de två kategorierna, med respektive utan källarfönster.

3.2 Uppbyggnad av träningsmodell

Modellen som användes i arbetet för bildigenkänning av källarfönster skapades i Python med hjälp av Keras. Som bas för modellen användes nätverket VGG16, vars vikter var förtränade med *ImageNet* som databas. För att underlätta träningsperioden så att den inte skulle vara alltför tidskrävande valdes en storlek på 150×150 för bilderna som input.

Strukturen för modellen med information om lagren samt antalet träningsbara och icke-träningsbara parametrar kan ses nedan i figur 5:



Figur 5: Sammanfattande beskrivning av strukturen på modellen som användes för alla körningar.

Storleken på delmängden som avfärdades i dropout-lagret var 20%. I de två sista lagren användes ReLU och sigmoid respektive som aktiveringsfunktioner.

De olika mapparna där datamängderna var lagrade lästes sedan in med os-modulen så att de datamängderna kunde förflyttas till sina respektive tränings, validerings och testkataloger. Därefter valdes 40 bilder av varje kategori slumpmässigt och förflyttades till testkatalogen. Resterande bilder förflyttades till träningskatalogen, vilket var totalt 600 bilder.

Datamängderna lästes in i nätverket med hjälp av Keras-klassen *ImageDataGenerator*. För att se hur data augmentation påverkade träningen gjordes tre olika körningar; en utan någon data augmentation, en med vanliga transformer som rotation, spegelvändning och beskärning och en med tillagd ändring av ljusstyrka. De tre modellerna kallas för M1, M2 och M3 respektive.

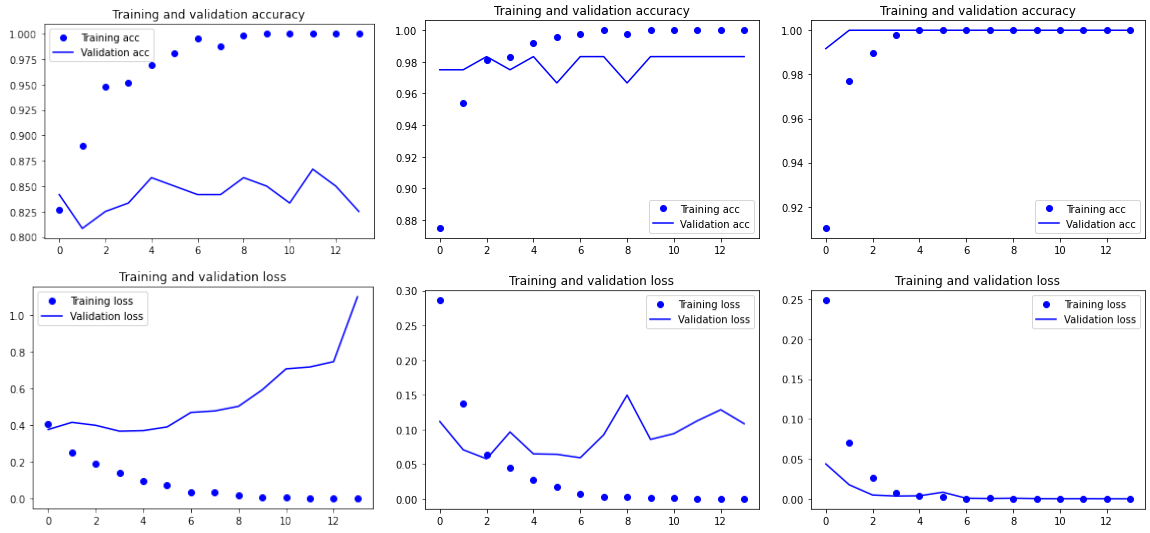
För träning med korsvalidering delades träningsmängden upp i 5 delar, där en av dessa delar valdes som valideringsmängd för varje körning på 14 epoker. Detta repeterades fem gånger och medelvärdet för resultaten av körningarna beräknades och presenteras i resultatavsnittet, tillsammans med grafer på noggrannhet samt förlust på tränings- respektive valideringsdatan för en av korsvalideringsuppdelningarna.

4 Resultat

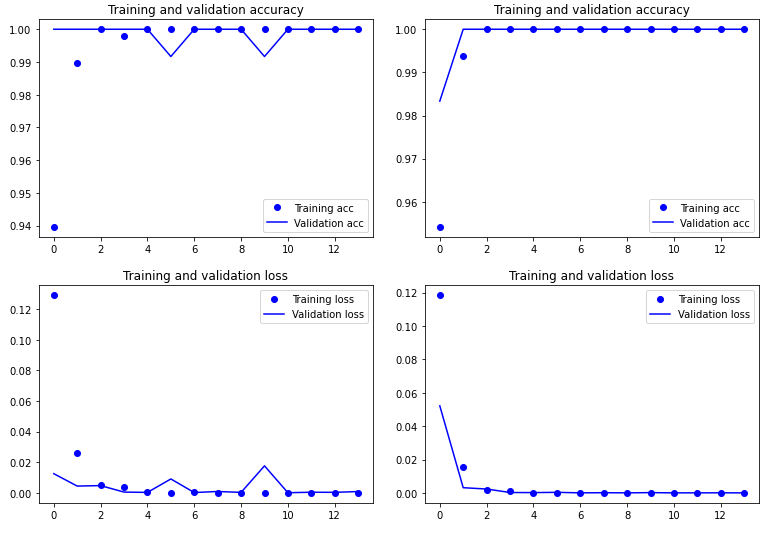
Detta avsnitt är till för att presentera resultaten från tre olika körningar med varierande data augmentationer. För varje delavsnitt visas noggrannheten och förlusten på testmängden för var och en av de fem delarna i korsvalideringen, samt medelvärdet av dessa. Det presenteras även grafer på träningsprestandan i de fem delarna av korsvalideringen. Dessutom presenteras ROC-kurvor på testmängden, med tillhörande area under kurvorna (AUC) efter träning med var del av korsvalideringen. Ytterligare resultat med exakta siffror för körningarna kan ses i GitHub repositoriet för arbetet (se Appendix A).

4.1 Modell M1

I figur 6 nedan presenteras träningsprestandan för de 5 delarna i korsvalideringen då ingen data augmentation användes. Som kan ses så ökar noggrannheten och omvänt så minskar förlusten efter 14 epoker för alla delar i träningsmängden, men endast valideringsmängderna för de tre senare delarna följer samma mönster.



(a) Del 1. *(b) Del 2.* *(c) Del 3.*



(d) Del 4. *(e) Del 5.*

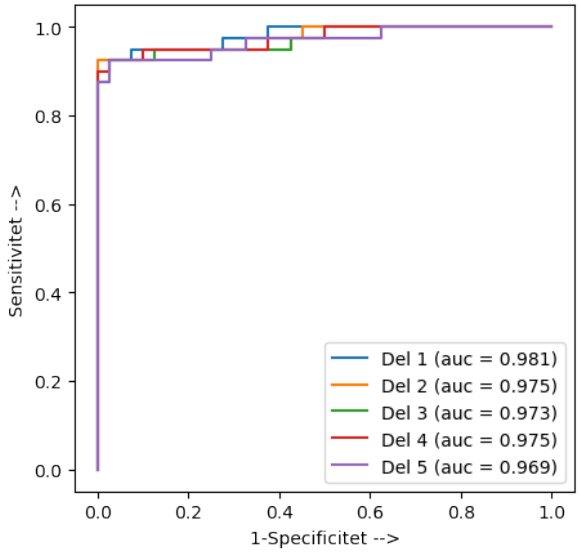
Figur 6: Träningsprestanda under 14 epoker för de 5 delarna av korsvalideringen för modell M1. De övre graferna visar noggrannhet och de undre graferna visar förlust för tränings- och valideringsmängden respektive.

I tabell 1 nedan visas noggrannheten och förlusten på testmängden efter träning med modell M1 för var del av korsvalideringen, samt medelvärdena av dessa.

Del	Förlust	Noggrannhet
1	0,73	90,00%
2	0,54	96,25%
3	0,76	93,75%
4	0,76	95,00%
5	1,01	92,50%
Medelvärde för alla delar:	0,76	93,50 (± 2,15)%

Tabell 1: Noggrannhet och förlust på testmängden för var del i korsvalideringen för modell M1. Resultaten är avrundade till två decimaler.

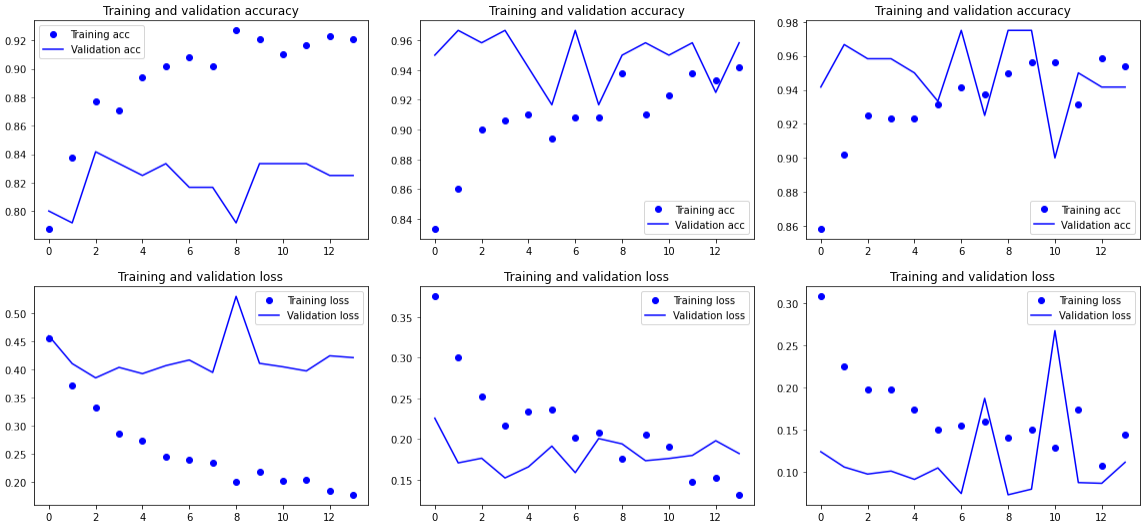
I figur 7 nedan visas ROC-kurvorna och arean under kurvorna (AUC) på testdatan för de fem korsvalideringarna i modell M1.



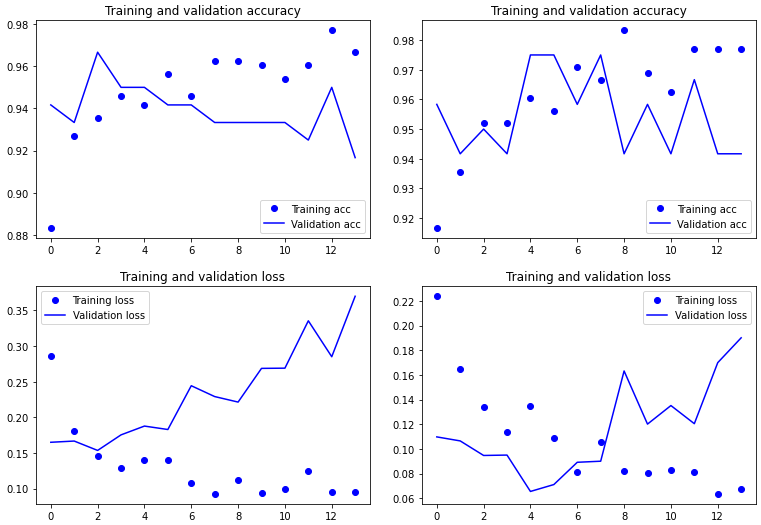
Figur 7: ROC-kurvor och area under kurvorna (AUC) på testmängden för de fem delarna i korsvalideringen för modell M1. Resultaten är avrundade till tre decimaler.

4.2 Modell M2

I figur 8 nedan presenteras träningsprestandan för de 5 delarna i korsvalideringen då data augmentation användes med rotation, bredd-translation, höjd-translation, skjuv-ändring, zoom samt spegelvändning. Som kan ses är det mer variation på noggrannheten och förlusten i alla delar jämfört med resultatet i avsnitt 4.1, både för träningsmängden och valideringsmängden.



(a) Del 1. *(b) Del 2.* *(c) Del 3.*



(d) Del 4. *(e) Del 5.*

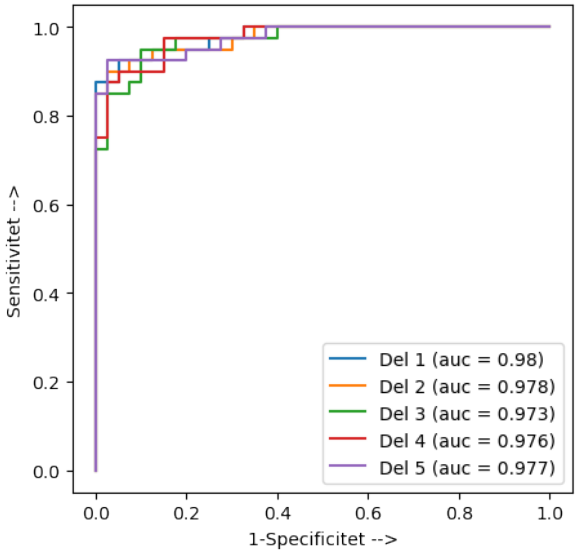
Figur 8: Träningsprestanda under 14 epoker för de 5 delarna av korsvalideringen för modell M2. De övre graferna visar noggrannhet och de undre graferna visar förlust för tränings- och valideringsmängden respektive.

I tabell 2 nedan visas noggrannheten och förlusten på testmängden efter träning med modell M2 för var del av korsvalideringen, samt medelvärdena av dessa.

Del	Förlust	Noggrannhet
1	0,25	93,75%
2	0,27	91,25%
3	0,31	88,75%
4	0,63	86,25%
5	0,40	95,00%
Medelvärde för alla delar:	0,37	91,00 (± 3,20)%

Tabell 2: Noggrannhet och förlust på testmängden för var del i korsvalideringen för modell M2. Resultat är avrundade till två decimaler.

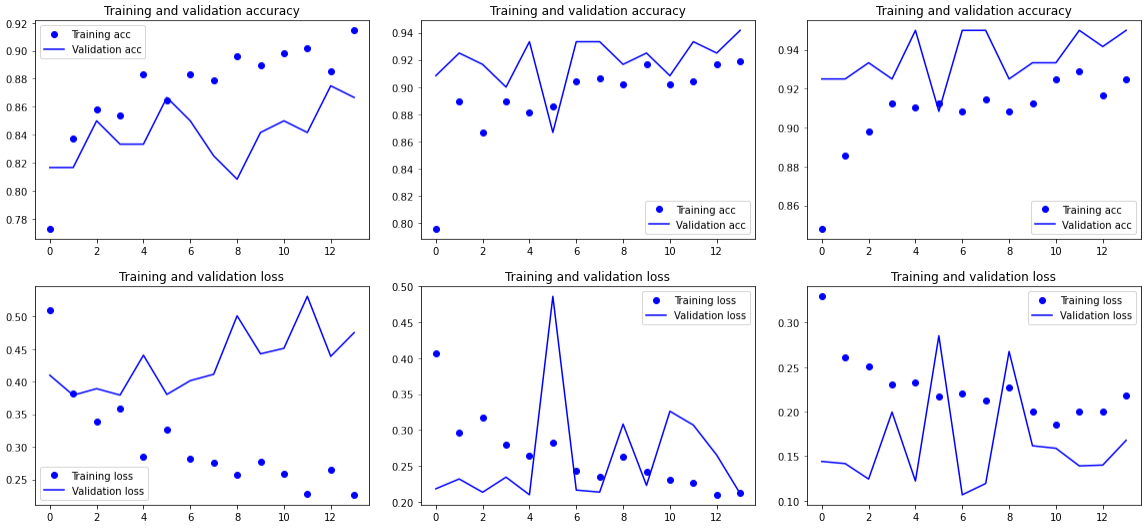
I figur 9 nedan visas ROC-kurvorna och arean under kurvorna (AUC) på testdatan för de fem korsvalideringarna för modell M2.



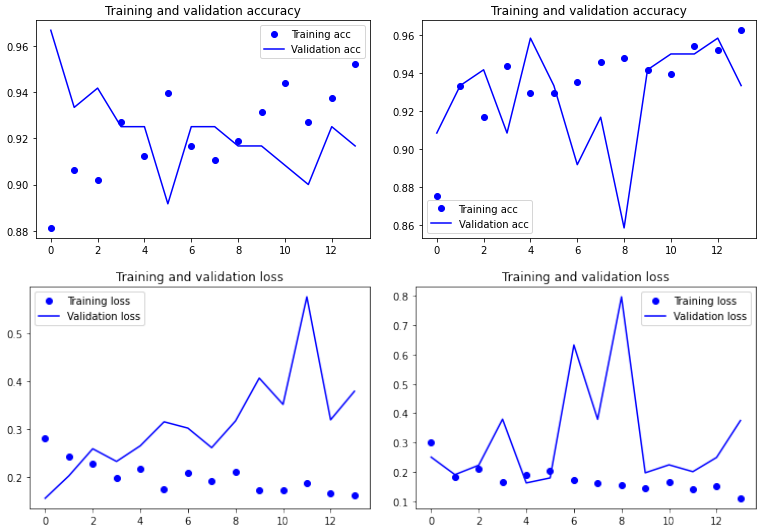
Figur 9: ROC-kurvor och area under kurvorna (AUC) på testmängden för de fem delarna i korsvalideringen för modell M2. Resultaten är avrundade till tre decimaler.

4.3 Modell M3

I figur 10 nedan presenteras träningsprestandan för de 5 delarna i korsvalideringen då samma data augmentationer som i avsnitt 4.2 användes, med tillagd ljusändring. Även här kan större variation i noggrannhet och förlust ses i tränings- och valideringsmängderna för de olika delarna av korsvalideringen.



(a) Del 1. *(b) Del 2.* *(c) Del 3.*



(d) Del 4. *(e) Del 5.*

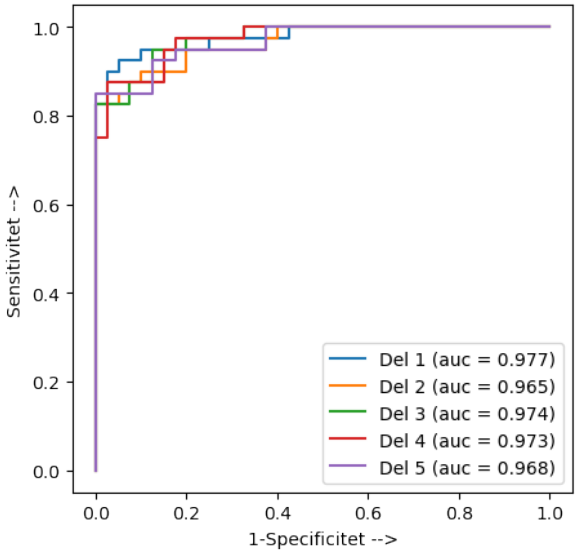
Figur 10: Träningsprestanda under 14 epoker för de 5 delarna av korsvalideringen för modell M3. De övre graferna visar noggrannhet och de undre graferna visar förlust för tränings- och valideringsmängden respektive.

I tabell 3 nedan visas noggrannheten och förlusten på testmängden efter träning med modell M3 för var del av korsvalideringen, samt medelvärdena av dessa.

Del	Förlust	Noggrannhet
1	0,28	92,50%
2	0,30	88,75%
3	0,29	87,50%
4	0,56	88,75%
5	0,43	88,75%
Medelvärde för alla delar:	0,37	89,25 (± 1,70)%

Tabell 3: Noggrannhet och förlust på testmängden för var del i korsvalideringen för modell M3. Resultaten är avrundade till två decimaler.

I figur 11 nedan visas ROC-kurvorna och arean under kurvorna (AUC) på testdatan för de fem korsvalideringarna för modell M3.



Figur 11: ROC-kurvor och area under kurvorna (AUC) på testmängden för de fem delarna i korsvalideringen för modell M3. Resultaten är avrundade till tre decimaler.

4.4 Jämförelse av resultat

När det handlar om noggrannhet, presterar modell M1 bäst på testmängden med 93,50 (± 2,15)% i genomsnitt. Dock så hade M1 något högre förlust (0,76 i genomsnitt) jämfört med M2 och M3 som introducerade data augmentation (0,37 i genomsnitt).

ROC-kurvorna har ungefär samma utseende för de tre modellerna. Då tröskelvärdet ökar når sensitiviteten värdet 1. Då tröskelvärdet är lågt är sensitiviteten fortfarande hög (runt 0,9 är det lägsta som nås för M1 och 0,8 är det lägsta som nås för M2 och M3). Arean under kurvorna (AUC) är likvärdiga för alla tre modeller men det verkar som att när del 1 av korsvalideringen används blir det högst prestanda på testmängden oberoende av vilken modell som används.

4.5 Test på bilder med ocklusion

Nedan visas noggrannhet, förlust och ROC-kurvor för de tre modellerna då de testades på bilder med ocklusion som sorterades bort innan träning då de inte kunde användas för klassificering. Totalt var det 2303 bilder som testades; 2006 av dessa tillhör klassen 'källare' och 297 av dessa tillhör klassen 'inte källare'.

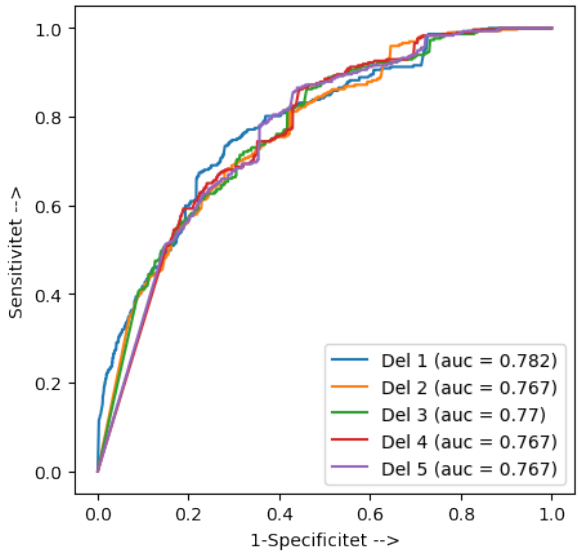
4.5.1 Modell M1

I tabell 4 nedan visas noggrannheten och förlusten för var del av korsvalideringen då modell M1 användes.

Del	Förlust	Noggrannhet
1	1,86	67,04%
2	3,70	57,62%
3	3,74	62,61%
4	5,29	59,18%
5	5,11	61,35%
Medelvärde för alla delar:	3,94	61,56 (± 3,24)%

Tabell 4: Noggrannhet och förlust på datamängd med ocklusion för var del i korsvalideringen för modell M1, samt medelvärden av dessa. Resultaten är avrundade till två decimaler.

I figur 12 nedan visas ROC-kurvorna och arean under kurvorna på datamängden med ocklusion för de fem korsvalideringarna för modell M1.



Figur 12: ROC-kurvor och area under kurvorna (AUC) på datamängd med ocklusion för de fem delarna i korsvalideringen för modell M1. Resultaten är avrundade till tre decimaler.

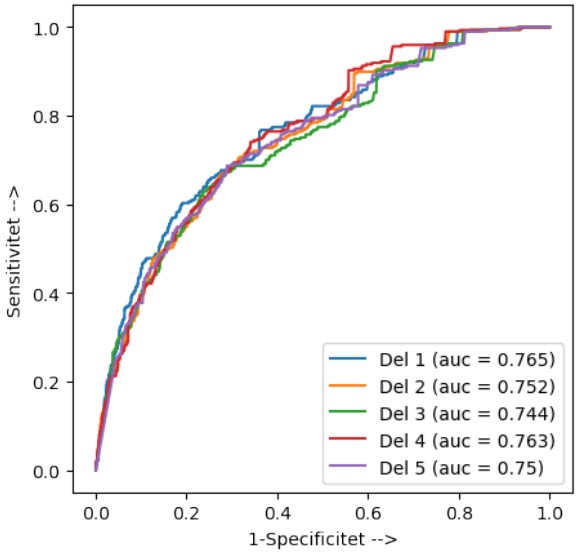
4.5.2 Modell M2

I tabell 5 nedan visas noggrannheten och förlusten för var del av korsvalideringen då modell M2 användes.

Del	Förlust	Noggrannhet
1	1,39	51,11%
2	1,67	53,10%
3	1,22	61,22%
4	1,19	70,21%
5	3,91	48,02%
Medelvärde för alla delar:	1,88	56,73 (± 8,03)%

Tabell 5: Noggrannhet och förlust på datamängd med ocklusion för var del i korsvalideringen för modell M2, samt medelvärden av dessa. Resultaten är avrundade till två decimaler.

I figur 13 nedan visas ROC-kurvorna och arean under kurvorna på datamängden med ocklusion för de fem korsvalideringarna för modell M2.



Figur 13: ROC-kurvor och area under kurvorna (AUC) på datamängd med ocklusion för de fem delarna i korsvalideringen för modell M2. Resultaten är avrundade till tre decimaler.

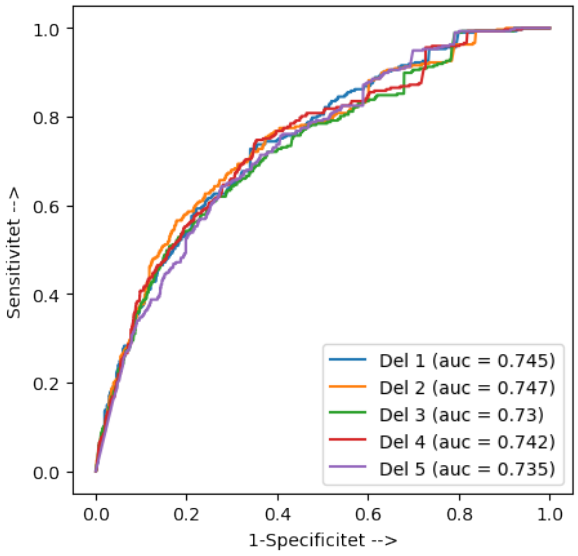
4.5.3 Modell M3

I tabell 6 nedan visas noggrannheten och förlusten för var del av korsvalideringen då modell M3 användes.

Del	Förlust	Noggrannhet
1	1,87	49,33%
2	1,99	48,68%
3	2,58	47,63%
4	1,51	66,35%
5	3,83	46,72%
Medelvärde för alla delar:	2,36	51,74 (± 7,36)%

Tabell 6: Noggrannhet och förlust på datamängd med ocklusion för var del i korsvalideringen för modell M3, samt medelvärden av dessa. Resultaten är avrundade till två decimaler.

I figur 14 nedan visas ROC-kurvorna och arean under kurvorna på datamängden med ocklusion för de fem korsvalideringarna för modell M3.



Figur 14: ROC-kurvor och area under kurvorna (AUC) på datamängd med ocklusion för de fem delarna i korsvalideringen för modell M3. Resultaten är avrundade till tre decimaler.

5 Diskussion

För de tre körningar vars resultat har redovisats i föregående avsnitt kan det observeras att runt 90% av 'street view'-bilderna i testmängden klassificerades rätt, med något lägre förlust för de två körningarna som implementerade data augmentation. Examensarbetet visar därför att det är möjligt att använda sig av 'street view'-bilder som datamängd i ett bildigenkänningsprogram till att identifiera källarfönster.

Ett problem som uppkom under examensarbetets gång var att det fanns brist på tillräckligt stor mängd data för att få pålitliga resultat. Anledningen till detta var att den totala mängden data som kunde användas var begränsat till en lista med adresser som efter inventering bekräftats att ha eller inte ha källarlokaler. Vidare, kunde Google Street View Static API verktyget som användes inte alltid hämta bilder från dessa adresser som gav någon information om det objekt som användes för klassificering i form av källarfönster. Det skulle därför behövas hämtas bilder från dessa adresser manuellt i Google Street View för att få mer träningsdata, vilket är en tidskrävande process.

Givet storleken på datamängden bestämdes det under examensarbetets gång att använda sig av korsvalidering samt data augmentation under träningsfasen för att minska risken för överträning. Träningsprestandan hos de fem delarna i korsvalideringen för modell M1 jämfört med modeller M2 och M3 (se figur 6 och figurer 8, 10 respektive) visar effekten hos den tillagda data augmenteringen i form av större varians, framför allt för valideringsmängderna.

Vad som även kan ses från körningarna, företrädesvis i figur 6, är att valideringsmängden som användes för de olika delarna i korsvalideringen gav väldigt olika noggrannhet och förlust efter 14 epoker. En förklaring till detta kan vara att 'street-view'-bilderna som användes ger olika information från bild till bild och att vissa av delarna i valideringsmängden liknar träningsmängden mer än andra delar på grund av att de exempelvis visar liknande byggnader eller är tagna i samma områden.

Resultaten för klassificeringen på testmängden (se tabeller 1, 2 och 3) visar att det för alla delarna av korsvalideringen gavs ungefär lika stor noggrannhet. Det var dock en större förlust på testmängden för modell M1 jämfört med modeller M2 och M3 vilket kan tyda på att den tillagda data augmentationen ledde till minskad överträning. ROC-kurvorna (se figurer 7, 9 och 11) visar att alla tre modeller har ungefär lika hög prediktiv styrka på testmängden, då de alla är mycket nära utseendet för en perfekt klassificerare.

Data augmentation med tillagd ljusändring verkade inte ge någon märkbar effekt på testresultatet jämfört med utan ljusändring vilket kan tyda på att datamängden som användes inte hade någon stor varians i det här avseendet. Andra augmentationer i bildkvalitet, exempelvis ändringar i färg och kontrast, mättes inte i arbetet eftersom det inte kunde implementeras direkt i klassen *ImageDataGenerator* och en extern metod skulle vara för tidskrävande.

Exempel på bilder i testmängden som klassificerades fel för alla körningar, både utan och med data augmentation, kan ses nedan i figur 15:

Figur 15: Exempel på 'street view'-bilder i testmängden från de två klasserna (med respektive utan källarfönster) som ständigt klassificerades fel.

Anledningen till felklassificeringarna på dessa exempel är oklara men om det finns element i bilderna som dyker upp mer frekvent i den andra klassen tyder det på att storleken på datamängden som användes för träning och validering kan vara förklaringen.

När de tre modellerna testades på de bilder som inte kunde användas i träningen resulterade det i en betydligt lägre noggrannhet, som i genomsnitt inte gav mycket bättre resultat jämfört med en slumpmässig klassificerare (se tabeller 4, 5 och 6). ROC-kurvorna visar på likvärdig tendens (se figurer 12, 13 och 14), med lägre AUC och grafer som är närmre den diagonala linjen som går mellan (0,0) och (1,1) och betecknar en slumpmässig klassificerare. Detta resultat var förväntat då den klassificering som gjordes var tydligt avgränsat till hurvida källarfönster är närvarande i bilderna och modellerna därför inte är lämpliga att använda för andra objekt som möjligen skulle användas till klassificering av dessa bilder.

5.1 Fortsatt arbete

Som framställdes i förra sektionen kan en större datamängd till träning och validering vara behövligt för att få mer pålitliga resultat, även om metoder i form av korsvalidering och data augmentation användes i arbetet för att försöka motverka detta. Vidare, skulle fler objekt kunna användas till klassificeringen istället för endast källarfönster. Exempel på dessa kan vara trappor som leder nedåt under jord längs en husvägg, andra typer av källaringångar samt källarnedfarter.

För att skala upp arbetet krävs det även en automatiserad metod för att ta bort de avvikande bilderna som inte ger någon information i klassificeringen, då en manuell metod skulle bli allt för tidskrävande. Ett sätt att ta bort dessa är att använda sig av VGG16 modellen med ett förtränat nätverk som redan har tränats på en bilddatamängd som är märkta med de relevanta kategorierna som ska undersökas. Exempel på en sådan datamängd är Places365, som innehåller 1,8 miljoner bilder och 365 olika kategorier, där exempelvis kategorierna *apartment* och *house* skulle kunna användas[19].

Vidare experiment med olika typer av data augmenteringar skulle även vara behövligt, speciellt färg och kontrastaugmentationer, då det är dessa typer som kan tänkas variera mest från bild till bild.

6 Slutsats

Baserat på problemställningarna i avsnitt 1.4 har följande slutsatser dragits:

1. Ett bildigenkänningsprogram kan tränas till att identifiera källarlokaler i byggnader med hjälp av 'street view'-bilder som datamängd.

2. Neuronnätet som användes för problemet kunde byggas upp med hjälp av den förtränade modellen VGG16 utökat med ytterligare fyra lager.

3. Den typen av data augmentation av 'street view'-bilder som tillämpades gav inte någon förbättring av prestandan för den datamängd som användes i arbetet, dock gav det något minskat överträning. En större datamängd och eventuellt fler typer av data augmentationer skulle vara behövligt för att få mer tillförlitliga resultat.

4. Ocklusion och sämre bildkvalitet minskar prestandan avsevärt för klassificering med de modeller som tränades upp i arbetet. En större datamängd och/eller en metod för att automatiskt ta bort dessa avvikande bilder skulle vara nödvändigt.

Med dessa slutsatser menar examensarbetaren att 'street view'-bilder går att använda som en lämplig källa för bildigenkänning av källarlokaler samt att neuronnätverk kan användas som ett verktyg för detta ändamål.

Referenser

[1] Shengyuan Zou, Le Wang, "*Detecting individual abandoned houses from google street view: A hierarchical deep learning approach*", ISPRS Journal of Photogrammetry and Remote Sensing, Volume 175, 2021, Pages 298-310, ISSN 0924-2716, https://doi.org/10.1016/j.isprsjprs.2021.03.020.

[2] Jian Kang, Marco Körner, Yuanyuan Wang, Hannes Taubenböck, Xiao Xiang Zhu, "*Building instance classification using street view images*", ISPRS Journal of Photogrammetry and Remote Sensing, Volume 145, Part A, 2018, Pages 44-59, ISSN 0924-2716, https://doi.org/10.1016/j.isprsjprs.2018.02.006.

[3] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, m.fl., "*Using deep learning and Google Street View to estimate the demographic makeup of neighborhoods across the United States*", PNAS, Volume 114, No 50, 2017, 13108-13113, https://doi.org/10.1073/pnas.1700035114

[4] Qing Cai, Mohamed Abdel-Aty, Ou Zheng, Yina Wu, "*Applying machine learning and google street view to explore effects of drivers' visual environment on traffic safety*", Transportation Research Part C: Emerging Technologies, Volume 135, 2022, 103541, ISSN 0968-090X, https://doi.org/10.1016/j.trc.2021.103541.

[5] D. Berlinski, "*The Advent of the Algorithm*", Harcourt Books, ISBN: 978-0-15-601391-8, 2000.

[6] F. Chollet, "*Deep learning with Python*", Shelter Island, NY, Manning Publications Co, ISBN: 9781617294433, 2018.

[7] R. Midya, "*AI Revolution - Voice Assistants & Their Smartness*", Medium, 20 oktober, 2020, [Online], Tillgänglig:
https://medium.com/swlh/ai-revolution-voice-assistants-their-smartness-c7afe2580e74,
Hämtad: 2022-02-04.

[8] Y. Bouchard, "*Tesla's Deep Learning at Scale: Using Billions of Miles to Train Neural Networks*", Towards Data Science, 7 maj, 2019, [Online], Tillgänglig:
https://towardsdatascience.com/teslas-deep-learning-at-scale-7eed85b235d3,
Hämtad: 2022-02-04.

[9] Springboard India, "*How Netflix's Recommendation Engine Works?*", Medium, 5 november, 2019, [Online], Tillgänglig:
https://medium.com/@springboard_ind/how-netflixs-recommendation-engine-works-bd1ee381bf81,
Hämtad: 2022-02-04.

[10] R. Vaishya, M. Javaid, I. H. Khan, A. Haleem, "*Artificial Intelligence (AI) applications for COVID-19 pandemic*". Diabetes & Metabolic Syndrome: Clinical Research & Reviews, vol 14, iss 4, 2020, p 337–339, ISSN 1871-4021, https://doi.org/10.1016/j.dsx.2020.04.012.

[11] T. Simonite, "*Facebook Creates Software That Matches Faces Almost as Well as You Do*", Technology Review, 17 mars, 2014, [Online], Tillgänglig:
https://www.technologyreview.com/2014/03/17/13822/facebook-creates-software-that-matches-faces-almost-as-well-as-you-do/,
Hämtad: 2022-02-04.

[12] TensorFlow, "*TensorFlow README.md*", [Online], Tillgänglig:
https://github.com/tensorflow/tensorflow#readme,
Hämtad: 2022-02-12

[13] C. Versloot, "*How to use K-fold cross-validation with Keras*", [Online], Tillgänglig:
https://github.com/christianversloot/machine-learning-articles/blob/main/how-to-use-k-fold-cross-validation-with-keras.md,
Hämtad: 2022-02-16

[14] Karen Simonyan, Andrew Zisserman, "*Very deep convolutional networks for large-scale image recognition*", Visual Geometry Group, Department of Engineering Science, University of Oxford, https://doi.org/10.48550/arXiv.1409.1556.

[15] Keras, "*Keras Applications*", [Online], Tillgänglig:
https://keras.io/api/applications/,
Hämtad: 2022-02-18

[16] C. Ouyang, "*How to Query Google Street View Static API with Python (UPDATED IN 2020)*", Elvin Ouyang's Blog, Uppdaterad: 31 maj, 2020, [Online], Tillgänglig:
https://elvinouyang.github.io/project/how-to-query-google-street-view-api-with-python/,
Hämtad: 2021-11-16.

[17] Google, "*Street View Static API overview* ", [Online], Tillgänglig:
https://developers.google.com/maps/documentation/streetview/overview,
Hämtad: 2022-02-18

[18] Google Maps & Google Earth Permissions, Tillgänglig:
https://www.google.com/intl/en-GB_ALL/permissions/geoguidelines/
Hämtad: 2022-02-04

[19] B. Zhou, A. Lapedriza, A. Khosla, A. Torralba, A. Oliva, Massachusetts Institute of Technology, *Places Download*, [Online], Tillgänglig:
http://places2.csail.mit.edu/index.html,
Hämtad: 2022-03-13

Appendix A

I det här repositoriet på GitHub redovisas all kod som har använts i arbetet för att hämta datamängder samt för uppträning och analys av modeller. De datamängder som användes finns även tillgängliga här, såväl som exakta siffror för alla träningsepoker i körningarna som gjordes:

https://github.com/Thraxos/FMAM05

Master's Theses in Mathematical Sciences 2022:E15
ISSN 1404-6342

LUTFMA-3465-2022

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/