Detektering av föroreningar i dagvattensystemet

MARKUS LJUNGQVIST NILSSON OCH GUSTAV WETTERBRANDT
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Detektering av föroreningar i dagvattensystemet

Av

Markus Ljungqvist Nilsson och Gustav Wetterbrandt

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

LTH Ingenjörsskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg

Printed in Sweden
Lunds universitet
Lund 2021

Sammanfattning

Målet med examensarbetet var att fastställa om det är möjligt att använda maskininlärning tillsammans med en kommersiellt tillgänglig kamera för att identifiera cigarettfimpar i dagvattensystemet. För att göra detta konstruerades en modell av en dagvattenbrunn. Modellen bestod av en låda som genomskars av ett avloppsrör. I lådans lock hängdes en kamera som filmade en öppning i röret. Vatten som innehöll cigarettfimpar hälldes genom röret och filmades med kameran. Dessa filmklipp användes sedan för att skapa två olika datamängder som användes för att träna två olika faltande neuronnät till att identifiera cigarettfimpar. Båda nätverken uppnådde i slutändan över 95% noggrannhet på träningsdatan och 100% på testdatan. Detta resultat indikerar att det bör vara möjligt att med hjälp av maskininlärning detektera cigarettfimpar i en dagvattenbrunn.

Nyckelord: Maskininlärning, Neuronnät, Dagvattensystem, Bildklassificering, Föroreningar.

Abstract

The aim of this thesis was to determine whether it is possible to use machine learning in conjunction with a commercially available camera to identify cigarette butts in the stormwater system. To do this, a model of a stormwater well was constructed. The model consisted of a box that was cut through by a drain pipe. A camera pointing toward an opening in the drain pipe was attached to the lid of the box. Water containing cigarette butts was poured through the pipe and filmed with the camera. These film clips were then used to create two different datasets that were used to train two different convolutional neural networks on how to identify cigarette butts. Both networks ultimately achieved over 95% accuracy on the training data and 100% on the test data. This result indicates that it should be possible to detect cigarette butts in a stormwater well with the help of machine learning.

Keywords: Machine learning, Neural networks, Stormwater system, Image Classification, Contaminants

Förord

Vi vill först och främst tacka Victor Pelin från VA SYD för den hjälp han bidragit med under examensarbetet och för hans stora intresse kring ämnet som undersökts.

Vi vill även tacka vår handledare Christin Lindholm som hjälpt oss mycket med korrekturläsning av och förbättringsförslag till rapporten samt för att hon såg till att vi fick möjlighet att vara i en lokal på E-huset i Lund. Vi vill även tacka vår examinator Christian Nyberg för att han hjälpte oss med att komma in till lokalen när våra LU-kort inte fungerade samt för hjälp med korrekturläsning.

Miljöbron Skåne är en organisation som förmedlar och koordinerar projekt och examensarbeten mellan näringslivet och universitet och högskolor i Skåne. Fokuset för både projekten och examensarbeten ligger på miljö och hållbar utveckling. Detta examensarbete förmedlades av Miljöbron Skåne för VA SYD:s räkning. Vi vill tacka Helena Ensegård från Miljöbron för att hon hjälpte oss komma i kontakt med VA SYD och för den stöttning hon har bidragit med under examensarbetets gång.

Markus Ljungqvist Nilsson & Gustav Wetterbrandt, Lund, maj 2021.

Innehållsförteckning

1. Inledning

Detta examensarbete gjordes i samarbete med VA SYD och Miljöbron Skåne. Examensarbetets huvudsakliga syfte var att besvara frågan huruvida det är möjligt att med objektklassificering detektera cigarettfimpar i dagvattenbrunnar.

1.1 Bakgrund

VA SYD är ett politiskt styrt kommunalförbund som ansvarar för driften av vatten- och avloppssystemen i Burlöv, Eslöv, Lomma, Lund och Malmö. VA SYD förser en halv miljon människor med rent dricksvatten och varje år renas 65 miljoner kubikmeter vatten i deras 15 avloppsreningsverk. Förbundet arbetar kontinuerligt med förbättringar av VA-systemet genom att byta ut gamla ledningar samt anlägga öppna kanaler och dammar för rening av dagvatten. Detta är ett stort och komplext arbete eftersom delar av VA-systemet är över 100 år gammalt. Som en del i sitt arbete vill de undersöka möjligheten att använda kameror för att detektera föroreningar i dagvattensystemet. För att få hjälp med rekryteringen av studenter till examensarbetet anlitade VA SYD organisationen Miljöbron Skåne.

Miljöbron förmedlar och koordinerar projekt som främjar samarbete mellan näringsliv och universitet och högskolor. Studenterna fick hjälp av Miljöbron Skåne att komma i kontakt med VA SYD och har under examensarbetets gång haft regelbundna möten med Helena Ensegård på Miljöbron Skåne för att stämma av att arbetet gått som planerat.

1.1.1 Beskrivning av avlopp- och dagvattensystem

I VA-systemet finns normalt två separata system som hanterar avloppsvattnet i vårt samhälle, spillvatten- respektive dagvattensystemet. Till spillvattensystemet är alla bostäder samt offentliga och privata fastigheter anslutna. Där hamnar allt vatten som spolas ut från t.ex. tvätt- och diskmaskiner, duschar, toaletter och industrier. Dessa system är kopplade till ett eller flera reningsverk där föroreningarna tas omhand innan vattnet släpps ut i naturen igen.

Vattnet som inte rinner ut i spillvattensystemet avleds i dagvattensystemet och det är främst regnvatten som hamnar där. Detta vatten rinner sedan ut i

sjöar, vattendrag och hav, ofta utan någon kemisk eller annan typ av filtrering. Detta är problematiskt då regnvattnet innehåller föroreningar från luften. Dessutom ökar vattnets föroreningsinnehåll på vägen till och genom dagvattensystemet [2]. Regnvattnet transporterar saker som är miljöfarliga, t.ex. cigarettfimpar och annat skräp som slängts på gatan, partiklar från bildäck, asfalt från vägar, olja från bilar och andra fordon samt olovliga utsläpp från industrier [2]. Även fjärrvärmevatten har i förekommande fall läckt ut från fjärrvärmeledningar och därmed hamnat i dagvattensystemet [3].

Regnvatten för även med sig flera andra typer av objekt och partiklar, till exempel sediment, jord, lera, löv, kvistar, grenar, insekter, råttor och andra smådjur. Detta är inte lika farligt för miljön men kan orsaka stopp eller minskad kapacitet i systemet som kan leda till översvämningar. Det kan även bidra till att vattnets turbiditet ökar. Turbiditet är ett mått på hur grumligt vattnet är och orsakas av mikroskopiska partiklar som flyter runt i vattnet. Hög turbiditet kan i vissa fall störa vattenlevande organismer [4].

Det finns idag inget kostnadseffektivt sätt att detektera vad som rinner ut i dagvattensystemet. Stopp och andra problem upptäcks oftast genom att allmänheten rapporterar till VA SYD då fel påträffas. Därför vill VA SYD i en del av sitt miljö- och digitaliseringsarbete undersöka möjligheten att med hjälp av objektklassificering upptäcka föroreningar som runnit ut i dagvattensystemet för att på så vis kunna hindra att dessa rinner ut i olika vattendrag och förorenar miljön.

1.1.2 Maskininlärning

Maskininlärning är en disciplin inom datavetenskapen där en dator tränas i att lära sig utföra en uppgift utan att programmeraren berättar för datorn hur uppgiften skall lösas. Maskininlärning kan användas för att låta datorer utföra uppgifter som skulle ta väldigt lång tid för en människa att utföra. Det kan t.ex. röra sig om att detektera specifika objekt eller ansikten i stora mängder av bilder eller skapa väderprognoser.

Objektklassificering är en del inom maskininlärning och handlar om att möjliggöra för en dator att “se” bilder och därmed detektera om ett visst föremål finns i en bild.

1.2 Syfte

Undersöka möjligheten att detektera cigarettfimpar i en dagvattenbrunn med hjälp av objektklassificering. Detta skulle leda till förbättringar i VA SYD:s miljöarbete samt minska risken att sjöar och vattendrag blir förorenade.

1.3 Målformulering

Examensarbetet ska undersöka om det går att detektera cigarettfimpar i dagvattensystem med hjälp av en kamera, ljussättning och objektklassificering. En fysisk modell av en dagvattenbrunn ska tas fram och ljussättningen av modellen ska testas för att bestämma vilken ljussättning som ger bäst detekteringsfrekvens. Detektering av cigarettfimpar med hjälp av objektklassificering ska utföras och resultaten sammanställas.

1.4 Problemformulering

Under detta examensarbete kommer följande frågor besvaras:

1. Kan ett maskininlärningsprogram tränas till att detektera cigarettfimpar i ett flöde med hjälp av ett syntetiskt dataset?
2. Kan man med bilder från en modell som ska efterlikna en dagvattenbrunn identifiera cigarettfimpar med hjälp av objektklassificering?
3. Hur påverkar vattnets flödeshastighet möjligheten att identifiera cigarettfimpar?
4. Hur påverkar vattnets turbiditet (grumlighet) möjligheten att identifiera cigarettfimpar?

1.5 Avgränsningar

Examensarbetet är begränsat i vilka typer av föroreningar som ska undersökas. I första hand är det detektering av cigarettfimpar som ska utvärderas. Examensarbetet ämnar inte att utföra tester på en verklig dagvattenbrunn utan endast på en modell av en sådan.

2. Teknisk Bakgrund

I detta kapitel beskrivs de verktyg och tekniker samt teorin bakom dessa som använts under examensarbetet. Vissa termer beskrivs kortfattat i terminologiavsnittet.

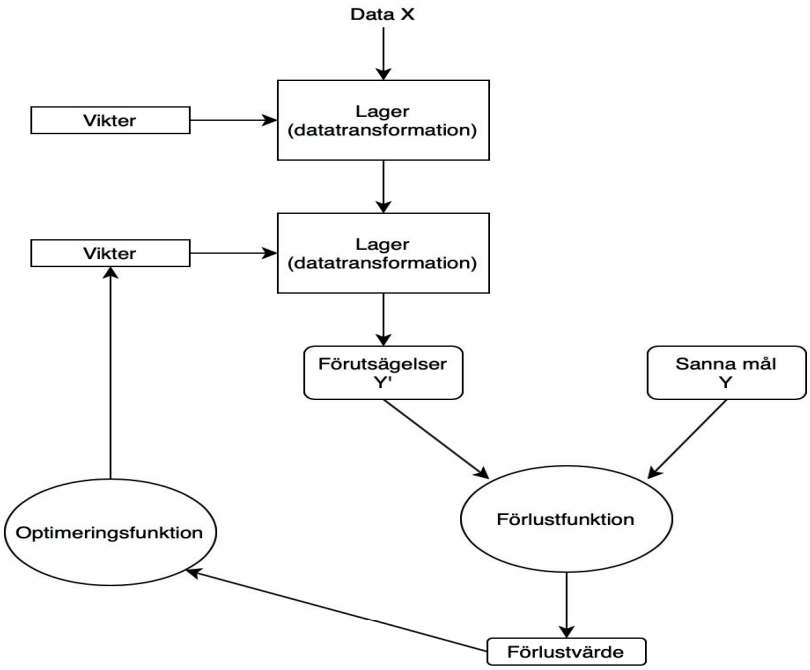
2.1 Maskininlärning med Python

Idéerna som kom att utvecklas till det som idag kallas artificiell intelligens, ett forskningsfält som maskininlärning är en del av, myntades av Alan Turing redan på 1950-talet [1]. Maskininlärning är ett paradigm inom datavetenskapen som skiljer sig från det traditionella sättet att skriva program. Inom traditionell programmering anger utvecklaren vilka regler datorn ska förhålla sig till samt vilken data som ska användas, och datorn svarar på programmerarens frågor. Inom maskininlärning fungerar det annorlunda. Reglerna skrivs inte i koden utan datorn tränas i att lösa uppgifter givet att den har tillgång till någon data och svaren på de frågor som programmeraren ställer [1]. Maskininlärning används idag bland annat i Googles och Apples röstassistenter i Android resp. iOS [5], Teslas autopilot för självkörande bilar [6] samt för automatisk bildbehandling i smarta mobiltelefoner [7]. Maskininlärning kan även användas inom sjukvården för att t.ex upptäcka cancertumörer [8].

För att ta fram reglerna som används för att lösa ett visst problem används en iterativ process där datorn testar olika sätt att lösa problemet. Efter varje iteration utvärderar datorn sig själv genom att testa reglerna på en datamängd vars innehåll är strikt skiljd från datan i träningsmängden. Datorn uppdaterar reglerna hela tiden genom att beräkna hur stor skillnaden mellan indatan och den faktiska utdatan är. Eftersom reglerna är matematiska konstanter kan dessa då uppdateras proportionellt mot skillnaden mellan indatan och utdatan [1].

I examensarbetet användes programmeringsspråket Python för att bygga en modell som kan detektera cigarettfimpar i bilder. Python är idag det mest populära språket för maskininlärning eftersom det har stöd för många viktiga verktyg och bibliotek som t.ex. Keras, TensorFlow och PyTorch [9]. Dessa ramverk beskrivs mer ingående i kapitel 2.1.2 och 2.1.3.

2.1.1 Neuronnät och deep learning



Figur 2.1. Övergripande skiss av ett djupt neuronnät

Termen neuronnät kommer från och inspireras av hur hjärnan fungerar där sammankopplade neuroner, som i sig är väldigt enkla nervceller, tillsammans kan lösa komplexa uppgifter [10]. En maskininlärningsmodell ska dock inte förväxlas med en modell av den mänskliga hjärnan [1]. En neuron inom maskininlärning är en enkel matematisk funktion som tar en eller flera parametrar som indata. Dessa multipliceras med olika värden som kallas vikter och summeras. Dessa vikter ansvarar i slutändan för nätverkets prestanda i form av hur väl det utför sin uppgift. Summan skickas sedan till en *aktiveringsfunktion* som formar neuronens utdata [11]. Kraften i maskininlärning ligger i att skapa lager av sammankopplade neuroner som kan hitta mönster i stora datamängder. *Deep learning* är ett applikationsområde där neuronnät med många sammankopplade lager

används (se figur 2.1). Man kallar dessa nätverk helt enkelt för “djupa” nätverk [11].

All maskininlärning använder sig av avancerade algoritmer för att lära sig en uppgift. Ett neuronnät accepterar indata av en viss dimension och producerar utdata genom att transformera datan enligt vissa algoritmer beroende på vilka lager nätet består av. Traditionellt sett består indatan av en endimensionell vektor men med moderna tekniker kan flerdimensionella vektorer användas [1].

2.1.2 Typer av neuronnät

Det finns över tjugo olika typer av neuronnät. Några av de mest använda är RNN (engelska: recurrent neural network), CNN (convolutional neural network) och ANN (artificial neural network) [12]. Ett RNN är ett nätverk där indatan splittras och skickas genom nätverket del för del. Utdatan som produceras för varje del av indatan sparas undan i ett tillstånd och kan användas igen när ny indata skickas genom nätverket. RNN har visat sig vara mycket effektivt i text till tal applikationer [13]. ANN kallas även “feed forward neural networks” och används främst för att analysera tabulär data [13].

Ett *faltande neuronnät* (CNN) är ett typ av neuronnät som används för att lära en dator att känna igen mönster i bilder. För att mata in bilddata till ett CNN krävs att bilderna konverteras till tredimensionella matriser. En matris innehåller information om en bilds storlek i x- och y-led samt dess färginnehåll.

Ett neuronnät består av ett eller flera lager. Tillsammans bildar lagren ett nätverk av neuroner. Varje lager är en modul som tar någon typ av indata, transformerar den och ger någon form av utdata [1] (se figur 2.1). För att objektklassificering ska fungera behöver flera olika typer av lager kombineras. Ett faltningslager används överst i modellen för att plocka ut unika särdrag i en bild, t.ex. kanten på ett objekt [14]. Därefter används ett *pooling layer* för att plocka ut de viktigaste egenskaperna från den data faltningslagret producerat, t.ex. hur ett öga ser ut, och skicka vidare dessa egenskaper till nästa lager. Görs inte detta måste nätverket ta hänsyn till alldeles för många parametrar vilket sänker nätverkets prestanda. Ett

flattening layer används sedan för att forma datan till en endimensionell vektor som kan skickas till ett *densely connected layer* där den slutgiltiga klassificeringen sker [1][14].

Utöver lagren och indatan krävs en förlustfunktion och en optimeringsfunktion för att bygga upp nätverket. Förlustfunktionen skapar en återkopplingssignal som representerar hur väl nätverket har lyckats med sin uppgift. För att klassificera bilder kan funktionerna *binary crossentropy* eller *categorical crossentropy* användas för två respektive flera klasser av bilder. Optimeringsfunktionen bestämmer på vilket sätt nätverket ska uppdateras efter en träningsepok baserat på förlustfunktionen [1].

2.1.3 TensorFlow

TensorFlow är ett ramverk för maskininlärning som tagits fram av Google. Det fungerar både med Python och C++ och kan köra kod både på vanliga processorer (CPU) eller på processorer med acceleration från en grafikprocessor (GPU) från Nvidia med hjälp av deras CUDA-API [9][15]. Det går att programmera direkt i TensorFlow men det rekommenderas av [9] att skriva kod i högnivåramverket Keras istället för att göra koden mer läsbar. TensorFlow används bland annat av Twitter, Dropbox och PayPal.

Ett alternativ till TensorFlow är PyTorch som tagits fram av Facebook och finns tillgängligt som öppen källkod sedan 2017 [16]. PyTorch har en del fördelar och nackdelar jämfört med TensorFlow. Syntaxen i PyTorch är väldigt lik vanlig pythonkod vilket gör det lätt för någon som redan är van vid Python att lära sig PyTorch. Ett annat sätt som PyTorch skiljer sig åt från TensorFlow är dess stöd för dynamiska grafer. I TensorFlow är grafer statiska vilket innebär att nätverket måste kompileras om varje gång något ska ändras. I PyTorch uppdateras nätverket dynamiskt under runtime för varje epok som körs vilket förbättrar nätverkets inlärningsförmåga tack vare att optimeringen fungerar bättre [17].

De båda ramverkens olika fördelar och nackdelar har gjort dem populära i olika kretsar. TensorFlows skalbarhet och stora hårdvarustöd har gjort det populärt i kommersiella sammanhang medan PyTorch har blivit populärt inom akademin eftersom det är enklare att använda och i många scenarion har bättre prestanda än TensorFlow [17].

2.1.4 Keras

Keras är ett API som tillåter användaren att enkelt skapa neuronnät och arbeta med deep learning. Keras abstraherar TensorFlows funktionalitet för att göra arbetet med deep learning enkelt för programmeraren. Istället för att, som i TensorFlow, definiera individuella neuroner i ett nätverk låter Keras användaren definiera lager direkt vilket förkortar processen att bygga ett neuronnät avsevärt [9][18].

I Keras skapas ett nätverk genom att en så kallad modell först definieras. Modellen är en klass som implementerar metoder för maskininlärning. Den tillåter användaren att lägga till lager i sin modell likt hur ett element läggs till i en enkellänkad lista. I Keras är en sekventiell modell den mest använda modellen. Där skickas data genom nätverket ett lager i taget tills det kommer ut som utdata. Om användaren behöver en annan typ av modell kan en sådan skapas med hjälp av Keras inbyggda funktionella API. Det är också möjligt att skapa egna lager om standardutbudet inte skulle innehålla det som behövs [18].

2.1.5 Träning av ett faltande neuronnät

För att träna ett neuronnät krävs stora mängder data, t.ex. textsekvenser eller bilder av objekt. Datamängden delas upp i tre delar för träning, validering och testning. Valideringss- och testmängderna ska innehålla data som inte finns i träningsmängden [1].

Modellen tränas iterativt. En iteration kallas för en epok. Under en epok tränas modellen typiskt på all tillgänglig träningsdata. När en träningsepok har körts utvärderas hur väl modellen har lärt sig något från epoken med hjälp av valideringsdatan. Utvärderingen bedöms efter hur stor del av valideringsdatan nätverket klassificerar korrekt vilket anges med en procentsats som kallas för noggrannhet. Testdatan används efter att modellen tränats och ger ett mått på hur bra modellen är på att utföra sin uppgift [1].

Det går att köra oändligt många träningsepoker, men efter ett visst antal blir inte modellen bättre på att detektera det den ska, det vill säga noggrannheten förbättras inte. Ett fenomen som ofta uppstår när ett neuronnät tränas kallas *overfitting*. Det uppstår när modellen börjar lära sig specifika eller onödiga

saker om träningsdatan som inte är generellt applicerbara på data modellen aldrig sett förut. För att undvika overfitting kan man öka mängden och variationen av träningsdatan samt använda olika regulariseringstekniker [1][9].

Träningsprocessen beror på flera parametrar, t.ex inlärningsfaktorn, som styr hur fort nätet lär sig [19], och optimeringsfunktionen som med hjälp av förlustfunktionen uppdaterar nätverkets vikter för att maximera noggrannheten [20]. Under träningen delas data in i olika grupper, så kallade *satser*. Storleken på satserna samt deras indelning kan påverka träningsprocessen. Om satserna är små i förhållande till den totala datamängden varierar resultatet mer än om grupperna är stora [21]. När nätverket lär sig utifrån en liten sats kan en unik egenskap, t.ex. en specifik färg, som endast återfinns i den satsen identifieras. En unik egenskap som återfinns i endast en sats introducerar brus i nätverket som hjälper nätverket att generalisera bättre [21].

Det är viktigt att datamängden som används för träning innehåller tillräckligt med information som överensstämmer med testdatan. Neuronnätet lär sig av informationen i träningsdatan och om inte datan innehåller några egenskaper som återfinns i testdatan kommer nätet inte kunna säga något om testdatan [1].

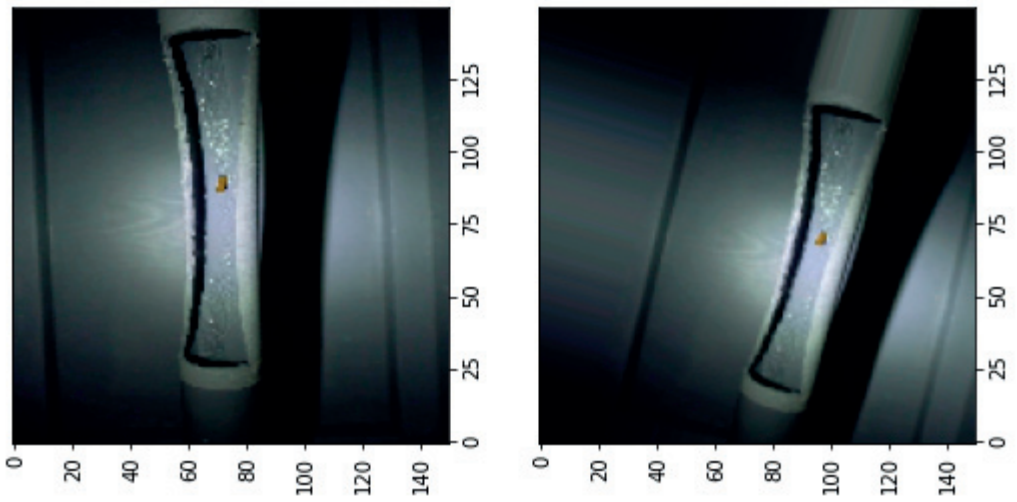
Målet med examensarbetet är att kunna klassificera cigarettfimpar i en dagvattenbrunn. Således behöver träningsdatan innehålla bilder av blöta cigarettfimpar i ett vattenflöde som är belysta med en konstant ljuskälla för att efterlikna de förhållanden som skulle kunna finnas i en dagvattenbrunn.

2.1.6 Regulariseringstekniker

Overfitting, vilket beskrivits i kapitel 2.1.5, är ett fenomen som alltid inträffar förutsatt att ett neuronnät tränats tillräckligt länge. Enligt [1] är det bästa sättet att förhindra overfitting att införskaffa mer träningsdata. Att göra detta är dock inte alltid möjligt och då finns andra tekniker att nyttja. Ett sådant sätt är att förminska neuronnätet genom att reducera antalet lager vilket resulterar i ett mindre antal träningsbara vikter. Eftersom vikterna styr hur bra nätet är på att lösa olika delar av en uppgift, och varje vikt endast

styr en mycket liten del av nätverket, leder en reduktion av antalet vikter till en reduktion av antalet aspekter nätet kan lära sig [1].

En annan teknik som använts under examensarbetet är *data augmentation*. Det används för att skapa mer variation i träningsdatan utan att tillföra någon ny data [1]. Det görs med hjälp av några olika transformationer som modifierar bilderna så att de ser unika ut. Några vanliga transformationer är rotation, spegelvändning och beskärning (se figur 2.2).



Figur 2.2. Bilden till vänster kommer från brunnmodellen. Bilden till höger är skapad med hjälp av data augmentation.

Den mest använda regulariseringstekniken är dropout. Dropout innebär att modellen efter varje epok slumpmässigt förkastar en liten del av det den precis lärt sig för att minska risken för att den lär sig något onödigt. Idén med tekniken är att introducera brus i nätverket som kan reducera risken att nätverket lär sig identifiera tillfälliga mönster som ger felaktiga svar, vilket minskar nätverkets prestanda [1][9].

2.1.7 Förtränade nätverk

Ett förtränat nätverk är ett nätverk som tränats på en stor datamängd och där vikterna från träningen sparats undan. Dessa vikter innehåller information som tillåter nätverket att "se" bättre då den har kännedom om vanliga fysiska egenskaper hos många olika objekt [1]. Vid användning av förtränade nätverk finns möjligheten att låsa specifika lager, vilket betyder att ett lagers vikter förblir oförändrade under träningsfasen. Några av de sista lagren i nätverket kan låsas upp så att de kan tränas på just den data som är relevant för den aktuella uppgiften. På detta sätt kan generell kunskap om många olika typer av objekt utnyttjas för att lösa specifika problem [1]. Förtränade nätverk finns fritt tillgängliga att använda i Keras vilket underlättar utvecklingsprocessen [1]. Ett sådant nätverk är VGG16. Det är tränat på en delmängd av datamängden ImageNet som innehåller 1,4 miljoner bilder i 1000 olika klasser [1].

2.1.8 Förlust- och optimeringsfunktioner

Förlustfunktionen beräknar hur långt det svar nätverket kom fram till är ifrån det faktiska svaret. Det beräknade värdet kallas för förlustvärde och det används av nätverkets optimeringsfunktion för att ändra på nätverkets vikter. Det är viktigt att den förlustfunktion som används matchar den typ av problem som ska lösas. Binary crossentropy används för klassificeringsproblem med två klasser, categorical crossentropy används för klassificeringsproblem med fler än två klasser och mean squared error för regressionsproblem [1].

Optimeringsfunktionen tar de beräknade förlustvärdena över elementen i en sats och uppdaterar vikterna så att förlustvärdena minskar. Detta är en iterativ process där vikterna uppdateras en gång för varje sats [1]. Den optimeringsfunktion som används i examensarbetet heter RMSprop och är en variant av en teknik som kallas för stokastisk lutning, vilken kan användas för att minimera förlusterna [1].

2.2 Mannings formel och flödesberäkningar

Mannings formel är en matematisk formel som används för att göra flödesberäkningar. Givet kända variabler kan flödeshastighet (m/s), flöde

(l/s) och lutning (m/m) beräknas med hjälp av formeln. På flödesform, eller Q-form, kan formeln skrivas som:

$$Q = A * R^{2/3} * M * \sqrt{S_0}$$

Formel 2.1 Mannings formel.

där Q är flödeshastigheten i m/s, A är den våta tvärsnittsarean för kanalen (vilket motsvarar $\pi * r^2$ för en cirkulär sluten ledning som har vatten upp till hjässan), R är den hydrauliska radien (vilket motsvarar *d/4* för en cirkulär sluten ledning) och M är Mannings tal som är en koefficient som beskriver hur skrovlig eller slät ytan på kanalen eller röret är. S_0 är kanalens lutning.

För plast används ett M mellan 90 och 120 [22].

2.3 Kamerautrustning och filminställningar

Kameran som användes under arbetet var en Fujifilm X-T20 tillsammans med ett Fujinon 16mm f/1.4 objektiv. Kameran har en sensor med upplösningen 24 megapixel och kan spela in video i Full-HD i 60 bilder/s och i 4K-upplösning i 30 bilder/s. För examensarbetets syften är bildfrekvensen viktigare än upplösningen och författarna till denna rapport beslutade att under arbetet filma i Full-HD (1080p) och 60 bilder/s istället för i 4K (2160p) och 30 bilder/s.

För belysning av vattenflödet i brunnmodellen användes en pannlampa med ljusstyrkan 100 lm från Clas Ohlson som monterades i modellens lock med hjälp av silvertejp.

3. Metod

Examensarbetet är indelat i fyra faser:

1. Informationssökning och planering
2. Konstruktion av brunnmodell
3. Generering av datamängder
4. Testning och utvärdering av neuronnäten.

Arbetet följde en sekventiell modell [23] där faserna utfördes i den ovan nämnda ordningen.

Kapitel 3 beskriver, i korta drag, hur examensarbetets olika faser planerades och sammanfattar deras genomförande kortfattat. Kapitel 4 beskriver i detalj hur varje fas genomfördes och förklarar ingående hur specifika problem löstes under examensarbetets gång.

3.1 Informationssökning och planering

I startskedet av examensarbetet genomfördes en större informationssökning. Syftet med det var att ta reda på hur maskininlärning fungerar och hur det kan användas i praktiken. Informationssökningen hittade inte något material som dokumenterar att något liknande projekt har genomförts tidigare. Däremot hittades exempel på projekt där vissa delar kunde vara applicerbara på examensarbetet. Till exempel hittades två rapporter som dokumenterade projekt där man använt deep learning för att detektera plast och annat skräp som flyter på floder i Kina [24] och Indonesien [25]. Dessa rapporter visar att det är möjligt att utföra objektdetektering (och därmed också objektklassificering) på objekt som flyter på en vattenyta. Dessutom görs det i en miljö där ljuset inte går att kontrollera på samma sätt som i examensarbetets specifika fall. En annan källa som hittades visade hur ett syntetiskt dataset med cigarettfimpar kan genereras [26], vilket var till stor hjälp under utvecklingsfasen.

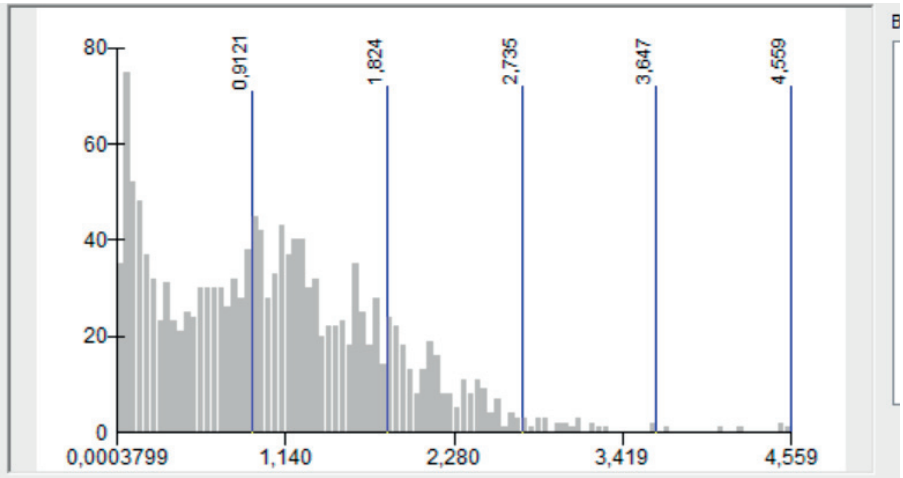
Det praktiska arbetet började med att ett digitalt planeringsmöte hölls tillsammans med Victor Pelin från VA SYD och Helena Ensegård från Miljöbron Skåne för att definiera examensarbetets syfte och målsättning. Under mötet diskuterades, bland annat, vad den genomsnittliga

flödeshastigheten är i en dagvattenbrunn, om någon testning skulle kunna utföras i en dagvattenbrunn och om VA SYD hade tillgång till någon datamängd med föroreningar som skulle kunna användas för att träna neuronnätet.

Eftersom det inte fanns någon datamängd beslutades det att arbetet skulle begränsas till en förorening. Valet föll efter diskussion till slut på cigarettfimpar. Fördelen med cigarettfimpar är att de flyter på vattenytan och att de har en fast form. En vätska, t.ex. olja eller bensin, kan blandas ut i vatten och blir därmed svårare att se med blotta ögat. Utifrån det gjordes antagandet att det skulle bli svårare för ett neuronnät att detektera flytande föroreningar än fasta och därför föll valet på cigarettfimpar.

Vid mötets slut bokades en tid för ett fältbesök i Råbylund där Victor höll en guidad tur där han visade hur olika delar av ett dagvattensystem kan se ut. Fältbesöket gav examensarbetarna förståelse för hur en dagvattenbrunn ser ut vilket kunde användas vid designen av brunnmodellen. Figur 4.5 är en bild av en dagvattenbrunn som fotograferades vid tillfället.

Efter mötet återkom Victor med en graf (figur 3.1) som visade de maximala flödeshastigheterna i ett dagvattensystem under en 10-årsperiod. Y-axeln visar antalet rör där vattnet uppnått en hastighet som korrelerar med värdet på X-axeln.



Figur 3.1 Graf som visar flödeshastigheter i ett dagvattensystem.

3.2 Konstruktion av brunnmodell

Eftersom det inte skulle gå att testa neuronnätet i en dagvattenbrunn beslutades det att en modell av en sådan skulle tas fram. Modellen användes för att ta fram både en syntetisk och en icke-syntetisk datamängd. Designen av modellen fastställdes först efter att fältbesöket hade varit. Modellen designades så att den skulle efterlikna de grundläggande egenskaperna hos en brunn. Den var tvungen att förhålla sig till vissa mått, den skulle ha en viss färg inuti och fick i största möjliga mån inte släppa in något ljus utifrån. Efter att ha pratat med Victor Pelin konstaterades att flödeshastigheter på 1 - 1,5 m/s (som är vanligt förekommande hastigheter i dagvattenbrunnar vid låg till normal nederbördsmängd enligt Victor) kunde uppnås med hjälp av gravitationen. Därför beslutades att ett lutande avloppsrör skulle användas för att uppnå den eftertraktade hastigheten. En ingående beskrivning av hur modellen konstruerades återfinns i kapitel 4.1.

3.3 Generering av datamängder

Först genererades en syntetisk datamängd med hjälp av filmklipp som hade filmats i brunnmodellen med kamerautrustningen som beskrivs i kapitel 2.3. Fimparna i bilderna från videoklippen frilades och klistrades in i bilder från ett videoklipp utan cigarettfimpar. Från början var planen att endast en syntetisk och en icke-syntetisk testmängd skulle skapas. Dock visade det sig att en enda cigarettfimp som färdades genom flödet gav upphov till över 10 bilder som innehöll en synlig cigarettfimp. På grund av detta kunde även en komplett icke-syntetisk datamängd av cigarettfimpsbilder genereras. Mängden bilder som krävs för att träna ett neuronnät var färre än vad som initialt hade antagits av examensarbetarna. De första antaganden som gjordes var att ungefär 5000 bilder skulle krävas för att träna neuronnätet men det visade sig i slutändan att 1600 bilder gav tillfredsställande resultat. Genereringen av bilderna och konstruktionen av datamängderna beskrivs utförligt i kapitel 4.2.

Parallellt med att datamängderna togs fram pågick arbetet med att utveckla neuronnäten som skulle tränas och utvärderas på datamängderna. Efter fortsatta litteraturstudier inom maskininlärning beslutade examensarbetarna

att utgå från neuronnät som Chollet beskriver i [1]. Detta på grund av den tidsbesparing det skulle ge jämfört med att utveckla två nya neuronnät.

Chollets neuronnät tränades på både syntetisk och icke-syntetisk tränings- och valideringsdata. Den ena modellen inkorporerade ett förtränat neuronnät (VGG16) medan det andra inte innehöll några förtränade delar.

3.4 Testning och utvärdering av neuronnäten

Neuronnäten som användes var inte anpassade för de datamängder examensarbetarna tagit fram. Detta insågs efter initiala tester då näten presterade sämre på examensarbetets datamängder än på Chollets diton. Därför gjordes förändringar i nätverkens hyperparametrar för att anpassa nätverken till cigarettfimpsdatamängderna. Dessutom lades ett dropout-lager till i nätverken vilket förbättrade inlärningen jämfört med tidigare. Arbetet kring nätverkskonfiguration redogörs för i kap 4.3.

Neuronnäten utvärderades på icke-syntetisk testdata i form av bilder från brunnmodellen. Utvärderingens resultat, som återfinns i kapitel 5, användes sedan för att svara på examensarbetets ursprungliga frågeställningar.

3.5 Källkritik

De primära källor som använts för examensarbetet har varit två läroböcker i maskininlärning, källa [1] och [9]. De anses vara trovärdiga eftersom François Chollet respektive Jeff Heaton är ledande experter inom deep learning och Chollet dessutom är skaparen av Keras som använts för att ta fram modellerna som använts under examensarbetet.

Källa [2] är en rapport framtagen av Lars Sylvén för miljökontoret i Mariestads kommun och avhandlar olika föroreningar som riskerar att hamna i dagvattnet. Sylvén sitter idag i kommunrevisionen och representerar Vänsterpartiet i Mariestad. Han var vid tillfället för rapportens författande Miljö- och hälsoskyddsinspektör i kommunen. Huruvida Sylvéns politiska färg har påverkat rapportens innehåll och trovärdighet är svårt att avgöra och detta bör has i åtanke när rapporten läses.

Källa [3] är en nyhetsartikel från en stor dagstidning. Den får anses vara pålitlig eftersom artikeln i sig inte tar ställning för eller emot något som

relaterar till examensarbetet. Tidningen i sig har reklam på sin hemsida men det är inte något som bör påverka dess trovärdighet.

Källa [4] är en rapport utgiven av en amerikansk federal miljöskyddsmyndighet. Rapporten har inga angivna författare, således kan man tolka det som att hela myndigheten står bakom rapporten. Rapporten bör vara väl granskad innan publicering varför den bör betraktas som pålitlig.

Källa [7] är en sida avsedd att marknadsföra Samsungs mobiltelefoner och deras AI-teknologi. Att den används som referens i den här rapporten är endast för att ge exempel på hur mycket mobiltillverkare satsar på maskininlärning och att de bidrar till att förbättra tekniken. Allting som står på sidan är skrivet i marknadsföringssyfte och text och bilder på sidan ska tas med en nypa salt vad gäller hur korrekt tekniken framställs.

Källa [8] är en populärvetenskaplig artikel från tidskriften Forskning & Framsteg. Man bör därför ha i åtanke att artikelns huvudsyfte är att informera och underhålla, inte att förklara en lösning på ett komplext problem eller presentera ett forskningsresultat. Här refereras den till för att visa på de många olika användningsområden som finns för maskininlärning och att tekniken kan användas för att rädda liv.

Källorna [5], [6], [10], [11], [12], [13], [14], [16], [17], [19], [20], [21], [26], [29], [31] och [35] är artiklar från olika programmeringsfokuserade hemsidor, vissa mer välkända än andra. Artiklarna är skrivna i syfte att lära ut olika koncept inom maskininlärning och är inte ute efter att sälja eller på något annat sätt marknadsföra vissa tekniker eller produkter. Vissa källor, t.ex [10], har ingen specifik författare angiven utan här står endast hemsidan som avsändare. Eftersom hemsidan i detta fall är välkänd för examensarbetarna sedan innan är inte detta något som påverkar förtroendet för att informationen hemsidan tillhandahåller inte skulle vara trovärdig.

Källa [15], [18] och [33] refererar till den officiella dokumentationen för TensorFlow respektive Keras. Båda dessa ramverk är open source, dvs fritt tillgängliga för alla att använda och bidra till utvecklingen av, och därav betraktas de som pålitliga.

[23], [24], [25], [36] och [37] är artiklar publicerade i två olika akademiska tidsskrifter. Det innebär att de har blivit ordentligt granskade (peer reviewed) av opartiska experter inom artiklarnas respektive områden. Detta grundar för att artiklarnas innehåll skall vara trovärdigt och pålitligt.

[27], [30], [32] refererar till den kod som producerats av rapportens författare under examensarbetets gång.

[28] refererar till repositoriet för källkodsbiblioteket JCodec som användes för att extrahera bildrutor från filmklipp i kapitel 4.2.

[34] refererar till en youtubevideo av Gerald Undone som visar relationen mellan slutartid, bildfrekvens och rörelseoskärpa i videoklipp. Gerald arbetar professionellt som videograf och besitter en stor teknisk kompetens inom området varför han kan betraktas som tillförlitlig.

4. Genomförande

I följande tre delkapitel redogörs ingående för hur examensarbetet genomfördes. I kapitel 4.1 behandlas konstruktionen av dagvattenbrunnmodellen för att den i framtiden skall kunna replikeras. Kapitel 4.2 tar upp genereringen av den syntetiska datamängden som användes för att träna de två olika neuronnät som utvecklades av examensarbetarna. Utveckling och träning av dessa redogörs för i kapitel 4.3.

4.1 Konstruktion av brunnmodellen

Eftersom det inte var praktiskt möjligt att utföra testning med en kamera i en brunn beslutades det att en fysisk modell skulle tas fram. Modellen användes både för testning av bildklassificiering och för att generera de båda datamängderna som användes för att träna neuronnätet. Modellen konstruerades av plywood i form av en låda med måtten 100x50x80cm (LxBxH) (se figur 4.1 och 4.2).

Figur 4.1. CAD-modell av stommen till brunnmodellen.

Figur 4.2. Stommen till modellen i konstruktionsfasen.

Längs med kanterna på ovansidan av sidopanelerna monterades en gummilist med hjälp av häftstift för att lådan skulle bli ljustät när locket läggs på (se figur 4.3).

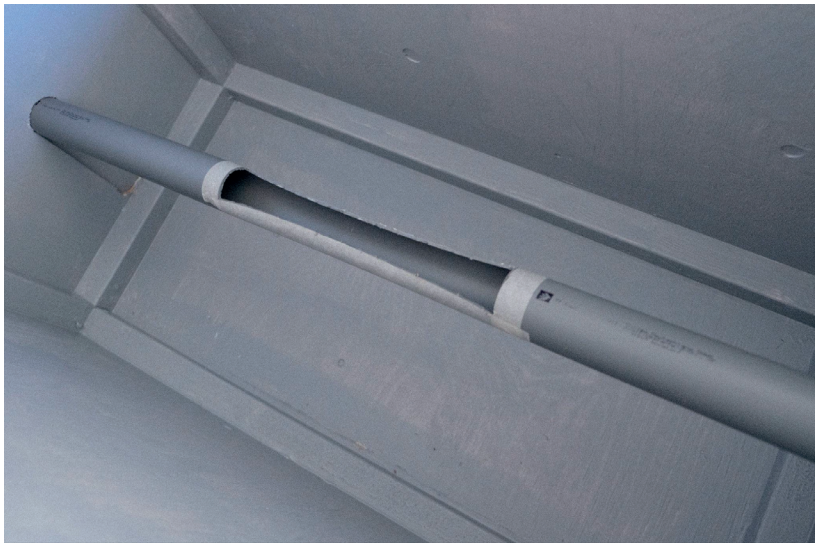
Figur 4.3. Gummilist gör modellen ljustät när locket läggs på.

I locket monterades en ⅜” UNC-bult så att ett ett kulledshuvud från ett kamerastativ kunde monteras (se figur 4.4).

Figur 4.4. Kameran och pannlampan monterade på lockets insida.

Insidan av lådan målades grå i ett försök att efterlikna betongen i en dagvattenbrunn (se figur 4.5 och 4.6).

Figur 4.5. Insidan av en dagvattenbrunn i Råbylund.



Figur 4.6. Insidan av modellen

Enligt figur 3.1 kan den genomsnittliga flödeshastigheten i dagvattensystemet approximeras till 1,0 m/s. Enligt formel 2.1 krävs en lutning på 0,35 m/m för att uppnå hastigheten 1,0 m/s. Detta testades i praktiken med hjälp av en hängränna utomhus innan modellen hade konstruerats. I modellens kortsidor gjordes hål för att montera ett

Figur 4.7. Hålet på utloppssidan som möjliggör justering av rörets fallvinkel.

avloppsrör med diametern 50 mm. På inloppssidan gjordes hålet tillräckligt stort för att röret skulle kunna vinklas uppåt och nedåt lite grann. På motstående sida gjordes ett avlångt hål så att rörets fallvinkel skulle kunna justeras steglöst mellan två ändlägen där det översta läget motsvarar en fallvinkel på 0,35 m/m. (se figur 4.7).

4.2 Generering av syntetisk datamängd

Det första steget för att skapa ett syntetiskt dataset var att spela in filmklipp med hjälp av brunnmodellen. Vatten hälldes ner i röret med hjälp av en vattendunk och en tratt och cigarettfimparna släpptes i tratten tillsammans med vattnet (se figur 4.8). Filmklippen importerades sedan till programvaran DaVinci Resolve (ett videoredigeringsprogram som liknar Adobe Premiere) där alla sekvenser som saknade en fimp i bilden klipptes bort. I det korta klippet som exporterades fanns det alltså en cigarettfimp i varje bildruta. Därefter användes det nya klippet som indata i ett egenskrivet javaprogram [27] som använde biblioteket JCodec [28]. JCodec innehåller en klass FrameGrab som kan extrahera bildrutor ur ett filmklipp och konvertera dem till enskilda bildfiler. Från ett ca 5 sekunder långt filmklipp kunde på så vis över 300 bilder med cigarettfimpar i flödet extraheras.

Figur 4.8. Vattendunk och tratt som användes för att förse modellen med fimpar och vatten.

Efter att fimpbilderna hade extraherats från videoklippet var de tvungna att friläggas innan de kunde användas för att generera syntetiska bilder. Att

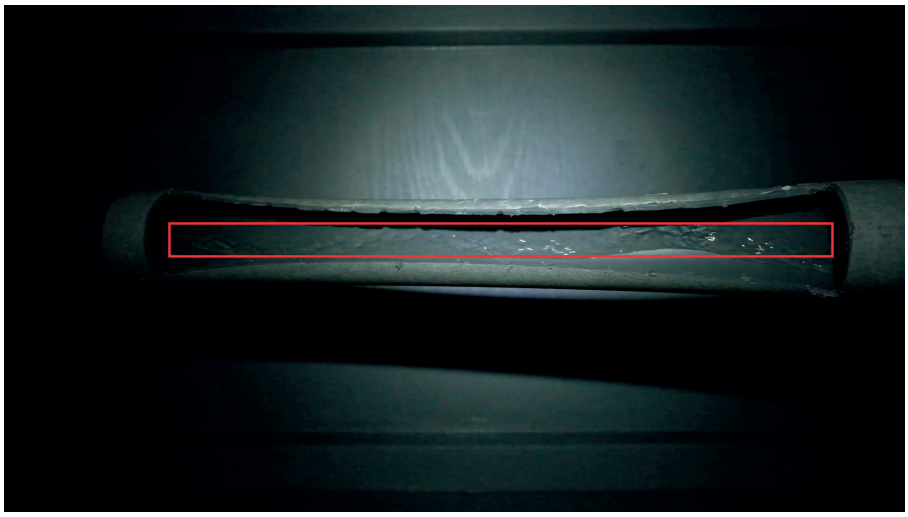
frilägga ett objekt ur en bild innebär att bakgrunden tas bort från bilden så att endast objektet återstår. Detta gjordes för hand med hjälp av Adobe Photoshop. Bilderna som extraherats från filmsekvensen importerades i Photoshop där en lagermask applicerades. I lagret maskerades bakgrunden bort så att bara cigaretterna var synliga. Därefter exporterades lagret som en ny bild i filformatet PNG. Exempel på hur detta ser ut visas nedanför i figur 4.9 och 4.10.

Figur 4.9. Bild av en fimp i modellen som blivit extraherad från ett filmklipp.

Figur 4.10. Fimp som blivit frilagd, dvs bakgrunden har tagits bort. Fimpen bibehåller sin position relativt bakgrunden från originalbilden och är alltså inte centrerad gentemot den genomskinliga bakgrunden.

För att kunna klistra in de frilagda fimparna på en specifik position i bilder från brunnmodellen behöver fimparna vara centrerade. Den yta där en frilagd cigarettfimp kan placeras i en ny bild är begränsad (se figur 4.11). Om ett objekt inte är centrerat vertikalt och det placeras i en annan bild på

en viss y-koordinat kommer det inte hamna på den angivna koordinaten. Detsamma gäller för den horisontella axeln men på grund av hur kameran är positionerad relativt röret är ett större intervall av koordinater möjliga på x-axeln, vilket gör det mindre viktigt hur noggrant fimpen placeras i x-led.



Figur 4.11 Bild från brunnmodellen där den möjliga placeringsytan för en frilagd fimp är markerad. Antalet möjliga placeringar i x-led är fler än i y-led.

För att centrera ett frilagt objekt behövs information kring var objektet befinner sig i bilden. För att bestämma objektets position krävs att bakgrunden har en entydig färg (se figur 4.12). En bild består av en matris där varje värde i matrisen anger vilken färg en korresponderande pixel har [29]. Ett mindre program för att utföra centrering på samtliga bilder i en mapp skrevs i Java [30]. Programmet skannar bilden rad för rad uppifrån och ner. När en pixel påträffas som har en annan färg än bakgrunden markeras pixelns koordinat som objektets högsta punkt. På liknande sätt identifieras objektets lägsta, samt mest högra och vänstra punkter. Därefter skapas en ny svart bild i samma storlek. Området som omsluter objektet i den första bilden kopieras och klistras sedan in i den nya bildens mitt (se figur 4.13).

Figur 4.12. Bild av cigarettfimp innan centrering.

Figur 4.13. Centrering av cigarettfimp från figur 4.11

När fimparna hade centrerats användes ett pythonscript för att klistra in fimparna i bilder från brunnmodellen som saknade cigarettfimpar. Scriptet togs ursprungligen fram av Viraf Patrawala och en ingående beskrivning av hur det fungerar återfinns i [31]. Scriptet modifierades så att cigarettfimparna endast kunde klistras in vid en specifik y-koordinat och i

ett begränsat intervall av x-koordinater [32]. Scriptet kräver att bakgrunden i den bild som ska klistras in i en annan bild är svart. Scriptet gör denna konvertering automatiskt men det fungerade dåligt då nästan alla fimpar innehåller vita pixlar vilket gjorde att de färgades svarta precis som den vita bakgrunden. Därför målades bakgrunden svart för hand i Microsoft Paint. En syntetiskt genererad bild återfinns i figur 4.14.

Figur 4.14. Syntetisk genererad bild med fimp i modellen.

Den syntetiska datamängden som togs fram med pythonscriptet bestod av totalt 3200 bilder. Totalt användes 15 unika fimpar för att generera datamängden. Datamängden delades upp i tränings- och valideringsdata och hälften av bilderna i respektive kategori innehöll cigarettfimpar. Tränings- och valideringsdatan bestod av 2560 respektive 640 bilder. Som nämnts i kapitel 3 var det inte alltför tidskrävande att skapa 1600 bilder med cigarettfimpar från brunnmodellen. Därför skapades en ny, lika stor, icke-syntetisk datamängd med hjälp av nya filmklipp där tränings- och valideringsdatan delades in på samma sätt som i den syntetiska datamängden. Båda datamängderna använde samma testdata. Cigarettfimparna i testmängden återfanns varken i de syntetiska eller icke-syntetiska datamängderna. Testdatan bestod av 200 bilder vardera av cigarettfimpar och icke-cigarettfimpar.

4.3 Utveckling och träning av neuronnäten

Två olika bildigenkänningsmodeller, N1 och N2, togs fram i Python med hjälp av Keras. N1 var ett CNN med fyra faltande lager. VGG16 utgjorde grunden för det förtränade nätverket N2. Nätverket utökades med ytterligare fyra lager.

Nätverken läste in datamängderna med hjälp av klassen *ImageDataGenerator* i Keras. Klassen underlättade nedskalningen av bilderna från 1920 x 1080 till 150 x 150 pixlar, vilket var den storlek neuronnäten förväntade sig. Den huvudsakliga anledningen till bilderna skalades ner var att det går snabbare att träna ett neuronnät på bilder av mindre storlek [33].

N1 var en kopia av neuronnätet som används för att lösa hund- och kattproblemet i [1, p. 134-135]. Chollets konfiguration av neuronnätet utnyttjades eftersom nätets prestanda på hund- och kattproblemet fanns dokumenterad och kunde nyttjas som referensvärde för träningen på cigarettfimpsdatamängden. Målet med att använda Chollets nät var att undersöka hur nätet presterade under träningsfasen med den syntetiska datamängden. Nästa steg i arbetet var att göra små justeringar i konfigurationen för att anpassa N1 för examensarbetets datamängder. Data augmentation introducerades för båda datamängderna för att skapa större variation i träningsdatan och på det sättet frambringa en större mängd egenskaper åt N1 att lära sig. Med hjälp av denna teknik kunde nätverkets noggrannhet förbättras. Slutligen lades ett dropout-lager till i nätverket. N1:s struktur presenteras i kap 5 (se figur 5.1).

N2 baserades på Chollets förtränade nätverk som beskrivs i [1, p. 143-146]. Erfarenheterna från att konfigurera N1 nyttjades och dropout samt data augmentation implementerades från början. N2 finjusterades sedan enligt Chollets beskrivning i [1. p. 152-158]. Med hjälp av dessa tekniker tränades några lager av VGG16 på cigarettfimpar. N2:s struktur presenteras i figur 5.7. De exakta parametervärden som näten konfigurerades med återfinns i tabell 4.1. Inställningarna för data augmentation listas i figur 4.15. Både N1 och N2 tränades på de syntetiska och icke-syntetiska datamängderna. Resultatet av träningen presenteras i kap 5.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)
```

Figur 4.15. Konfigurationen av data augmentation som användes under examensarbetet

Parameter	N1	N2
Data augmentation	se figur 4.15	se figur 4.15
Förlustfunktion	Binary crossentropy	Binary crossentropy
Optimeringsfunktion	RMSprop	RMSprop
Inlärningsfaktor	$1 \cdot 10^{-4}$	$1 \cdot 10^{-5}$
Bildstorlek	150 x 150	150 x 150
Satsstorlek	20	20
Dropout	20 %	20 %
Aktiveringsfunktion	ReLU, Sigmoid	ReLU, Sigmoid

Tabell 4.1. En beskrivning av de parametervärden som användes i de olika neuronnäten.

5. Resultat

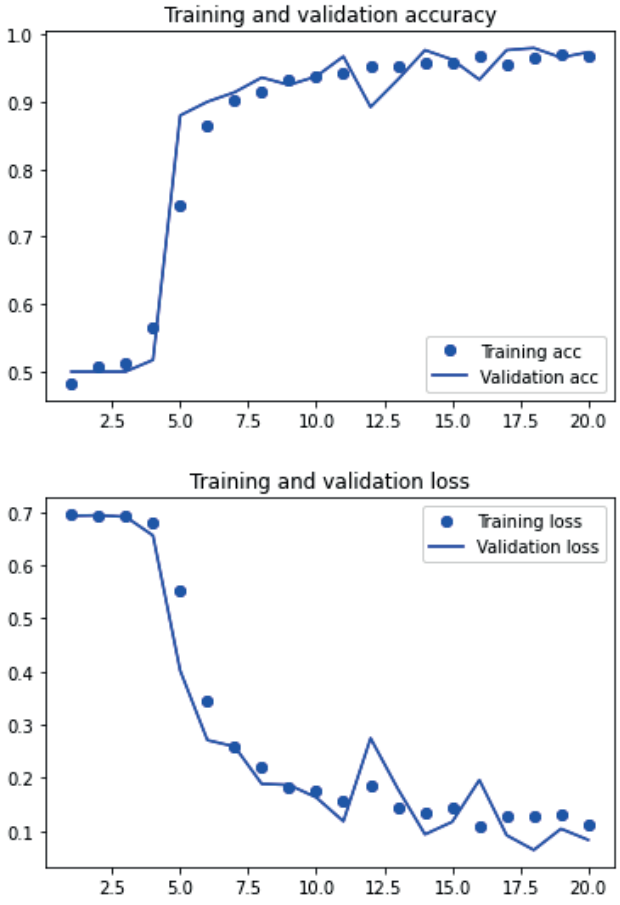
I detta kapitel presenteras resultaten av träningen av de två neuronnäten på de syntetiska och icke-syntetiska datamängderna. Skalorna i de grafer som presenteras i detta kapitel är inte alltid likadana. Y-axlarna visar antingen nätverkens noggrannhet eller förlust på tränings- respektive valideringsmängderna (vilken av dem framgår av texten i figuren). X-axlarna visar antalet utförda träningsepoker.

5.1 N1

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 15, 128)       0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

Fig 5.1. N1 (Chollets faltande neuronnät) med ett dropout-lager tillagt.

Figur 5.1 visar vilka lager N1 består av samt hur många olika träningsbara parametrar N1 innehåller. För neuronnätets inställningar se kolumnen N1 i tabell 4.1.



Figur 5.2. visar träningsprestandan för N1. Den övre grafen visar nätverkets noggrannhet på tränings- respektive valideringsdatan under varje epok. Den undre grafen visar förlusten på tränings- respektive valideringsdatan under varje epok.

Figur 5.2 visar hur N1 presterade på den icke-syntetiska datamängden under träningsfasen. Som syns i den övre grafen börjar neuronnätet hitta de parametrar som är unika för cigarettfimpar efter tre epoker vilket resulterar i

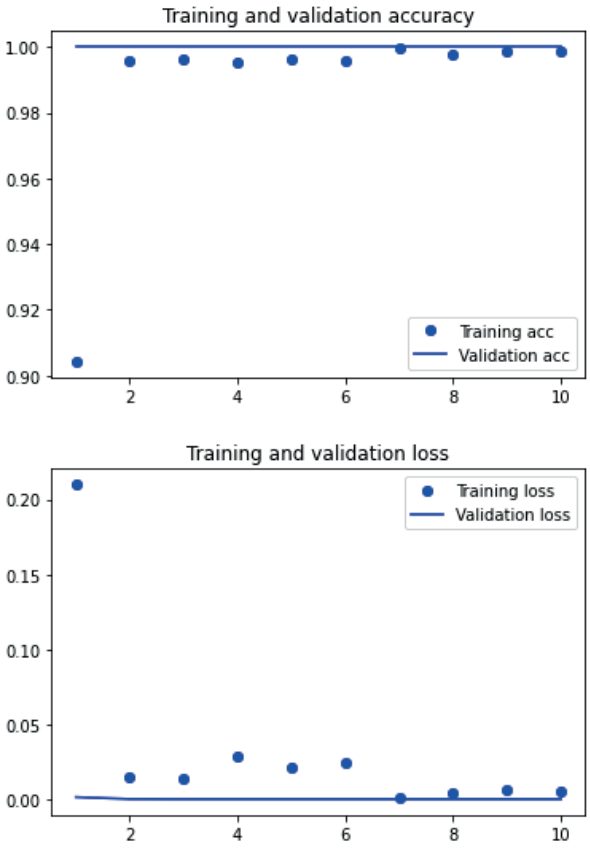
en ökning av tränings- och valideringsnoggrannheten. Omvänt visar den nedre grafen hur träningsförlusten minskar efter tre epoker.

```
Found 400 images belonging to 2 classes.
8/8 [==============================] - 2s 281ms/step - loss: 0.0188 - acc: 1.0000
test acc: 1.0

array([[200,   0],
       [  0, 200]], dtype=int64)
```

Figur 5.3. Testresultat för N1 på testmängden när N1 tränats på icke-syntetisk data.

Figur 5.3 visar hur N1 presterade på testmängden efter att ha tränats på den icke-syntetiska datamängden. Alla cigarett- och icke-cigarettfimpsbilder klassificerades korrekt.



Figur 5.4. visar träningsprestandan för N1. Den övre grafen visar nätverkets noggrannhet på tränings- respektive valideringsdatan under varje epok. Den undre grafen visar förlusten på tränings- respektive valideringsdatan under varje epok.

Figur 5.4 visar hur N1 presterade på den syntetiska datamängden under träningsfasen. Som syns i den övre grafen hittar N1 majoriteten av de parametrar som är unika för cigarettfimpar redan under den första epoken. På grund av att de unika parametrarna hittas så fort och att tränings- och valideringsnoggrannheten är nära 100 % är förlusten i den nedre grafen försumbar.

```
Found 400 images belonging to 2 classes.
8/8 [==============================] - 2s 248ms/step - loss: 3.1363e-07 - acc: 1.0000
test acc: 1.0

array([[200,   0],
       [  0, 200]], dtype=int64)
```

Figur 5.5. Testresultat för N1 på testmängden när N1 tränats på syntetisk data..

Figur 5.5 visar hur N1 presterade på testmängden efter att ha tränats på den syntetiska datamängden. Alla cigarett- och icke-cigarettfimpsbilder klassificerades korrekt. N1 presterar lika bra på testmängden oavsett vilken av datamängderna den tränats på.

5.2 N2

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

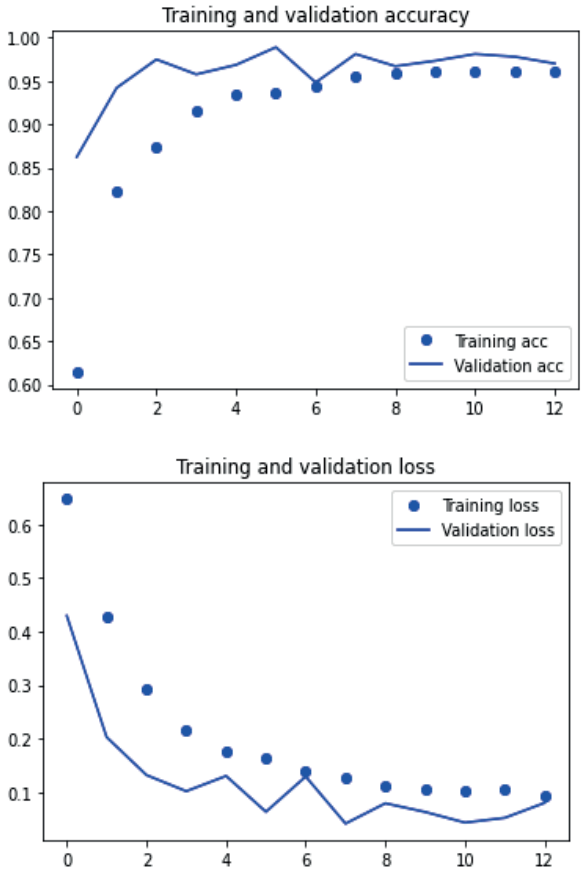
Figur 5.6. Struktur av VGG16

Figur 5.6 visar vilka lager neuronnätet VGG16 består av samt hur många olika träningsbara parametrar VGG16 tillåter.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
dropout (Dropout)            (None, 4, 4, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 256)               2097408
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,812,353
Trainable params: 9,177,089
Non-trainable params: 7,635,264
_________________________________________________________________
```

Fig 5.7. N2 (Chollets förtränade neuronnät) med ett tillagt dropout-lager.

Figur 5.7 visar vilka lager N2 består av samt hur många olika träningsbara och icke-träningsbara parametrar N2 innehåller. För neuronnätets inställningar se kolumnen N2 i tabell 4.1.



Figur 5.8. visar träningsprestandan för N2. Den övre grafen visar nätverkets noggrannhet på tränings- respektive valideringsdatan under varje epok. Den undre grafen visar förlusten på tränings- respektive valideringsdatan under varje epok.

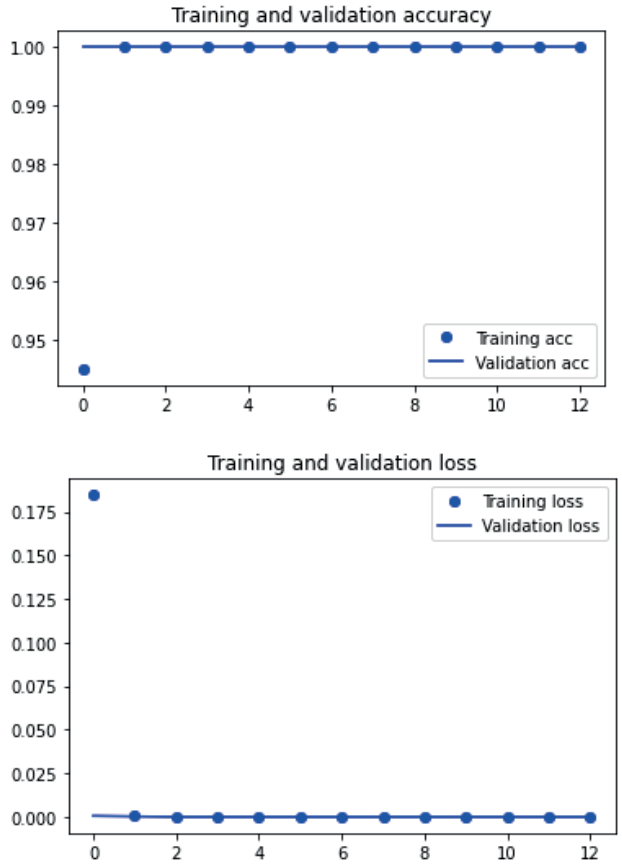
Figur 5.8 visar hur N2 presterade på den icke-syntetiska datamängden under träningsfasen. Som syns i den övre grafen hittar neuronnätet viktiga parametrar för klassificering av cigarettfimpar redan under de första epokerna. Med tiden avtar inlärningen varpå kurvan närmar sig ett gränsvärde. På samma sätt närmar sig förlusterna i nätverket gränsvärdet 0,1 efter att tillräckligt många epoker har körts.

```
Found 400 images belonging to 2 classes.
8/8 [==============================] - 6s 277ms/step - loss: 0.0044 - acc: 1.0000
test acc: 1.0

array([[200,   0],
       [  0, 200]], dtype=int64)
```

Figur 5.9. Testresultat för N2 på testmängden när N2 tränats på icke-syntetisk data

Figur 5.9 visar hur N2 presterade på testmängden efter att ha tränats med icke-syntetisk data. Alla cigarett- och icke-cigarettfimpsbilder klassificerades korrekt.



Figur 5.10. visar träningsprestandan för N2. Den övre grafen visar nätverkets noggrannhet på tränings- respektive valideringsdatan under varje epok. Den undre grafen visar förlusten på tränings- respektive valideringsdatan under varje epok.

Figur 5.10 visar hur N2 presterade på den syntetiska datamängden under träningsfasen. Som syns i den övre grafen hittar N2 majoriteten av de parametrar som är unika för cigarettfimpar redan under den första epoken.

```
Found 400 images belonging to 2 classes.
8/8 [==============================] - 6s 270ms/step - loss: 6.3700e-12 - acc: 1.0000
test acc: 1.0

array([[200,   0],
       [  0, 200]], dtype=int64)
```

Figur 5.11. N2:s testresultat på den syntetiska datamängden

Figur 5.11 visar hur N2 presterade på testmängden efter att ha tränats med syntetisk data. Alla cigarett- och icke-cigarettfimpsbilder klassificerades korrekt. N2 presterar lika bra på testmängden oavsett vilken av datamängderna den tränats på.

6. Diskussion

För att en modell ska efterlikna en dagvattenbrunn krävs det att den är ljustät samt att flödeshastigheten i modellen liknar de flödeshastigheter som förekommer i en dagvattenbrunn. Modellen som använts under examensarbetet uppfyller de här två kraven och representerar därmed en dagvattenbrunn i stora drag. Examensarbetets resultat visar således att det är möjligt att detektera cigarettfimpar i dagvattenbrunnar med maskininlärning. Anledningen är att båda datamängderna härstammar från bilder tagna från brunnmodellen och de är därmed representativa för en viss mängd av cigarettfimpar som rinner genom dagvattensystemet. I dagvattensystemet är dock variationen mellan cigarettfimparnas utseende större. En fimp som precis hamnat i en brunn ser antagligen inte likadan ut som en som har legat där i några dagar. Examensarbetet visar att om det finns tillräckligt mycket data för att träna ett faltande neuronnät på cigarettfimpars utseenden så är detektering av cigarettfimpar möjligt.

Ett problem i examensarbetet är bristen på variation i datamängden som representerar både cigarettfimpar och icke-cigarettfimpar. I en riktig dagvattenbrunn återfinns allt från pinnar och löv till godispapper och snusprillor. Att träna ett faltande nätverk med större datamängder som innehöll fler objekt hade gett ett rättvisare resultat med avseende på nätverkets uppnådda noggrannhet. Detta testades aldrig då den tid som hade krävts för att samla in objekten och skapa syntetiska bilder med dem i brunnmodellen hade överstigit den tid som fanns tillgänglig för utförandet av examensarbetet.

Det var stor skillnad mellan neuronnätens prestanda under träningsfasen på de två olika datamängderna. Dock resulterade bägge i slutändan i möjligheten för båda neuronnäten att skilja på cigarettfimpar och icke-cigarettfimpar. Som syns i figur 5.2 och 5.4 lär sig neuronnäten att skilja på de två klasserna olika fort beroende på vilken datamängd de tränats med. Med den syntetiska datan hittades de viktiga parametrarna nästan omedelbart, medan det med den icke-syntetiska datan tog flera epoker innan majoriteten av parametrarna hade hittats. Detta beror troligtvis på att de syntetiska cigarettfimparna var lägre upplösta än de riktiga och därmed hade ojämna kanter, vilket neuronnäten lärde sig att urskilja (jämför figur 6.1

med figur 6.2). När ett nät tränas på en sats märker den att samtliga cigarettfimpar är ojämna och ojämnheten fungerar då som en markering för nätverket att det är det området i bilden som är av intresse. Detta resulterar i att nätverket fokuserar på de specifika parametrarna i det området, varpå noggrannheten snabbare går mot 100% än för icke-syntetisk data.

Figur 6.1. Syntetisk cigarettfimp vars kanter är tydligt ojämna till följd av den bildbehandling som genomförts under arbetet.

Figur 6.2. Cigarettfimp från filmklipp som inte har modifierats.

Examensarbetet har inte kunnat visa att flödeshastigheten i en dagvattenbrunn påverkar detekteringsfrekvensen av cigarettfimpar. Dock uppmärksammades det vid beräkning av flödeshastigheten i brunnmodellen att filmklipp med lägre bildfrekvens gav upphov till en viss mängd rörelseoskärpa hos cigarettfimparna, vilket gjorde det svårare att beräkna flödeshastigheten. Fenomenet uppstår när fimpen rör sig i x-led samtidigt som kamerans sensor, som är stationär, exponeras för att registrera en bild.

Den tid det tar för kameran att ta en bild kallas slutartid. Ju längre slutartiden är desto mer rörelseoskärpa skapas när ett objekt rör sig i bilden [34]. Detta beror på att fimpen hinner förflytta sig en viss sträcka under den tid det tar för kameran att ta bilden. Slutartiden bör vara kortare än vad bildfrekvensen anger för att minska rörelseoskärpan. För en bildfrekvens på 60 bilder/s rekommenderas en slutartid på 1/120s per bildruta. För att minska rörelseoskärpan ytterligare kan ännu kortare slutartider utnyttjas på bekostnad av hur mycket ljus som når fram till sensorn (bilden blir därmed mörkare). Detta måste kompenseras om man vill behålla bildens ljusstyrka

genom att antingen öka sensorns ljuskänslighet, vilket gör bilderna brusigare, eller genom att öppna upp objektivets bländare, vilket minskar bildens skärpedjup. När bildfrekvensen höjdes från 30 bilder/s till 60 bilder/s minskade rörelseoskärpan eftersom slutartiden minskade.

Den enskilt viktigaste faktorn för om det ska fungera att detektera ett objekt i rörelse är alltså kamerans möjlighet att ta bilder med så låg rörelseoskärpa som möjligt. Förutsatt att flödeshastigheten i brunnen ligger inom normala värden (se figur 3.1) ska det inte krävas högre bildfrekvenser än 60 bilder/s för att fimparna ska bli skarpa. Det kan enkelt uttryckas som att om det mänskliga ögat kan se en cigarettfimp i en bild så kan även ett faltande neuronnät göra det förutsatt att det tränats tillräckligt för ändamålet.

Frågan kring turbiditet och hur den påverkar möjligheten att identifiera cigarettfimpar lämnades obesvarad. Kombinationen av avloppsröret som användes samt mängden vatten som kunde hällas genom modellen momentant resulterade i en vattennivå som var mindre än cigarettfimparnas diameter, vilket förhindrade testning. Examensarbetarna antar att det enda realistiska sättet för turbiditeten att påverka detektion är om cigarettfimparna döljs av det grumliga vattnet. För att det ska kunna inträffa måste vattnet ha så pass hög turbiditet att objekt under vattenytan inte är synliga för kameran. På grund av dessa faktorer kunde examensarbetarna inte utreda frågan.

I examensarbetets målformulering anges att ljussättningens påverkan på detekteringsfrekvensen skall undersökas. Då testresultaten endast kunde förbättras marginellt och detekteringsfrekvensen var maximal, givet den testmängd som användes, var vidare testning av ljussättning svår. Därmed testades ingen ytterligare belysning. Det är även så att försämrade ljusförhållanden kan kompenseras för med hjälp av kamerans inställningar. Det är därför svårt att säga något om hur sämre belysning hade påverkat resultatet.

6.1 Fortsatt arbete

Som beskrivits i kapitel 6 krävs fortsatt testning av hur noggrannheten påverkas när bilder med andra objekt läggs till i datamängderna. Objekt som lera, småsten, plastskräp, olja, gummirester samt utseendeförändringar hos

fimparna på grund av vattenflödet är bara några av de faktorer som behöver undersökas. Vidare bör testning i en riktig dagvattenbrunn utföras med, för ändamålet, adekvat hårdvara i form av kameror och belysning som klarar av att vistas i en dagvattenbrunn året runt utan att sluta fungera. Detekteringen och klassificeringen av objekten i brunnen bör sedan ske på en videoström som skickas från kameran till ett datacenter. Detta är viktigt eftersom VA SYD i framtiden vill kunna detektera stopp och andra problem i realtid. De ska inte behöva, som i examensarbetets fall, göra klassificeringen i efterhand eftersom det då kan vara för sent att göra något åt de problem man vill förhindra.

Som nämnts i kap 1.1.1 finns det många fler typer av föroreningar i dagvattensystemet än cigarettfimpar. Det är inte säkert att maskininlärning är den bästa metoden för att detektera samtliga. Fjärrvärmevatten är grönt och beroende på om det regnar eller inte kan det vara lätt eller svårt att identifiera då det kan blandas med regnvatten. Beroende på hur mycket fjärrvärmevattnet späds ut kan det vara enklare att detektera det genom att söka efter gröna pixlar än att använda maskininlärning. Skräp som hamnat i dagvattensystemet kan dock identifieras med maskininlärning [35][36]. För att förbättra och generalisera detektering av föroreningar i dagvattensystemet krävs vidare forskning.

För att kunna beräkna vattennivå och flödeshastighet med information från bilder krävs det att bilderna innehåller synliga markörer. Flödeshastigheten kan beräknas med ett synligt objekt som färdas tillsammans med flödet. Om objektets positionsförändring mellan två olika bilder och tidsskillnaden mellan bilderna är kända kan flödeshastigheten beräknas som objektets hastighet. Det är dock inte alltid så att objekt som flyter på ett vattenflöde färdas med samma hastighet som flödet. Att mäta flödeshastigheten på ovan beskrivna sätt är alltså inte en pålitlig metod. Det finns mer avancerade metoder som använder kameror för att lösa detta problem [37]. För vattennivån krävs, likt flödeshastighetsberäkningen, någon form av visuell indikator t.ex. höjdmarkeringar i brunnens kanter.

6.2 Etisk diskussion

Möjligheterna maskininlärning ger för att automatisera genomförandet av uppgifter som tidigare utförts manuellt är mycket stora. Att upptäcka stopp

och miljöfarliga utsläpp i dagvattensystemet är ett sådant område. På sikt bör detta möjliggöra för VA SYD och andra bolag med ansvar för VA-system på olika orter i Sverige att digitalisera övervakningen av dagvattensystemen och därmed bör de kunna förbättra sitt miljöarbete. En lyckad implementation av ett sådant system samt ett aktivt arbete för att minska utsläppen som systemet hittar skulle gynna djurlivet i sjöar och vattendrag. På sikt skulle t.ex. halten av miljöfarliga ämnen i insjöfiskar kunna reduceras.

Som nämnts i kapitel 1.1.1 återfinns många olika typer av objekt och föroreningar i dagvattensystemet, naturliga som olovliga. En av maskininlärningens viktigaste egenskaper är att den kan användas till att hitta mönster i stora mängder data, vilka en människa sedan kan analysera och dra slutsatser ifrån. Om ett företag misstänks för att olovligen släppa ut miljöfarligt avfall som hamnar i dagvattnet kan maskininlärning användas för att insamla bevis för att så har skett. Genom att hitta mönster i hur, var och när dessa utsläpp sker ges åklagare ytterligare verktyg att åtala företag för brott mot miljöbalken.

7. Slutsats

Utifrån examensarbetets problemformuleringar från kapitel 1.4 har följande slutsatser dragits:

1. Det är möjligt att träna en maskininlärningsmodell i att detektera cigarettfimpar med hjälp av ett syntetiskt dataset.
2. Det är möjligt att använda objektklassificering för att detektera förekomsten av cigarettfimpar i en modell som ska efterlikna en dagvattenbrunn.
3. Beroende på kamerans inställningar kan flödeshastigheten orsaka rörelseoskärpa vilket försvårar detektering av cigarettfimpar. Flödeshastigheter som normalt förekommer i dagvattensystemet är inte tillräckligt höga för att objekt som flyter däri inte ska kunna fångas på bild av en modern kamera.
4. Turbiditet och dess påverkan på möjligheten att identifiera cigarettfimpar har inte kunnat undersökas.

Examensarbetarna hävdar, utifrån ovan angivna slutsatser, att det bör vara möjligt att detektera cigarettfimpar i dagvattensystemet med hjälp av maskininlärning.

Referenser

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

[2] L. Sylvén, "*Föroreningar som riskerar att hamna i dagvatten",* Mariestads kommun, Mariestad, Sverige, Dnr 2003.74, 2004, [Online],Tillgänglig: https://mariestad.se/download/18.7e2db5ad15996c107651605/1485769088265/F%C3%B6roreningar+som+riskerar+att+hamna+i+dagvatten.pdf, Hämtad: 2021-04-14

[3] G. Elbied Pettersson, "Kraftig läcka i Borås - grönt vatten forsade ut", *Göteborgs-Posten,* Sep, 2, 2019, [Online], Tillgänglig: https://www.gp.se/nyheter/v%C3%A4stsverige/kraftig-l%C3%A4cka-i-bor%C3%A5s-gr%C3%B6nt-vatten-forsade-ut-1.17687572 , Hämtad 2021-04-14

[4] "*National Management Measures to Control Nonpoint Source Pollution from Urban Areas",* United States Environmental protection agency, Washington. D.C, USA, EPA-841-B-05-004, 2005,Tillgänglig: https://www.epa.gov/sites/production/files/2015-09/documents/urban_guidance_0.pdf, Hämtad: 2021-04-15

[5] R. Midya, *"AI Revolution - Voice Assistants & Their Smartness"*, Medium, Okt, 20, 2020, [Online], Tillgänglig: https://medium.com/swlh/ai-revolution-voice-assistants-their-smartness-c7afe2580e74, Hämtad: 2021-04-30

[6] Y. Bouchard, "*Tesla's Deep Learning at Scale: Using Billions of Miles to Train Neural Networks",* Towards Data Science, Maj, 7, 2019, [Online], Tillgänglig: https://towardsdatascience.com/teslas-deep-learning-at-scale-7eed85b235d3, Hämtad 2021-04-30

[7] Samsung, *"AI Camera, Redefining Mobile Photography",* 2020. [Online]. Tillgänglig: https://www.samsung.com/semiconductor/minisite/exynos/technology/ai-camera/, Hämtad: 2021-05-01

[8] P. Snaprud, "*AI bättre än läkare på att upptäcka hudcancer*", Forskning och framsteg, Juni, 25, 2020, [Online], Tillgänglig:

https://fof.se/tidning/2020/6/artikel/ai-battre-lakare-pa-att-upptacka-hudcancer, Hämtad: 2021-04-30

[9] J. Heaton, "*Applications of Deep Neural Networks*," Heaton Research, Inc. ePrint: 2009.05673, 2021. Tillgänglig: https://arxiv.org/abs/2009.05673 Hämtad: 2021-04-22

[10] Tutorialspoint, *"Artificial Intelligence - Neural Networks"*, 2020. [Online], Tillgänglig: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm, Hämtad: 2021-05-02

[11]S. Ronaghan, "*Deep learning: Overview of Neurons and Activation Functions*", Medium, Jul, 26, 2018, [Online], Tillgänglig: https://srnghn.medium.com/deep-learning-overview-of-neurons-and-activation-functions-1d98286cf1e4, Hämtad: 2021-05-02

[12] A. Mehta. "*A comprehensive guide to types of Neural Networks*", Digital Vidya, Jan, 25, 2019, [Online], Tillgänglig: https://www.digitalvidya.com/blog/types-of-neural-networks/, Hämtad: 2021-04-29

[13] A. Pai, "*CNN vs. RNN vs. ANN - Analyzing 3 types of networks in Deep Learning* ", Analytics Vidhya, Feb, 17, 2020, [Online], Tillgänglig: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/, Hämtad: 2021-04-29

[14] D. Nelson, *"Image Recognition in Python with TensorFlow and Keras"*, Stack Abuse, [Online]. May. 2019. Tillgänglig: https://stackabuse.com/image-recognition-in-python-with-TensorFlow-and-keras/, Hämtad: 2021-04-23

[15] TensorFlow, "*TensorFlow README.md*", 2021 [Online], Tillgänglig: https://github.com/TensorFlow/TensorFlow/blob/master/README.md Hämtad: 2021-04-23

[16] V. Kurama, "*PyTorch vs. TensorFlow: Which Framework is Best for Your Deep Learning Project?*", builtin, Sep, 9, 2020, [Online], Tillgänglig: https://builtin.com/data-science/pytorch-vs-TensorFlow, Hämtad: 2021-05-03

[17] G.Boesch, "*Pytorch vs. TensorFlow: A head-to-head comparison*", viso.ai, Mar, 2, 2021, [Online], Tillgänglig: https://viso.ai/deep-learning/pytorch-vs-TensorFlow/, Hämtad: 2021-05-03

[18] Keras Special Interest Group, "About Keras", 2021 [Online], Tillgänglig: https://keras.io/about/ Hämtad: 2021-04-24

[19] J. Brownlee, "*Understanding the Impact of Learning Rate on Neural Network Performance*", machinelearningmastery.com, Jan, 25, 2019, [Online], Tillgänglig: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/, Hämtad: 2021-04-29

[20] J. Brownlee, "*Loss and Loss Functions for Training Deep Learning Neural Networks",* machinelearningmastery.com, Jan, 28, 2019, [Online], Tillgänglig: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/, Hämtad: 2021-04-29

[21] J. Brownlee, "*How to Control the Stability of Training Neural Networks With the Batch Size*", machinelearningmastery.com, Jan, 21, 2019, [Online], Tillgänglig: https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/, Hämtad: 2021-04-29

[22] Svenskt Vatten AB, "*Publikation 110. Avledning av dag-, drän- och spillvatten*", Stockholm, Sverige: Svenskt Vatten AB, 2016.

[23] A. Vijay Patil, "Comparative Analysis between Sequential and Iterative Project Management Approaches", International Research Journal of Engineering and Technology (IRJET), vol. 06, iss. 07, p. 3035, July 2019.

[24] L. Zhang et al., *"Real-Time Water Surface Object Detection Based on Improved Faster R-CNN,"* Sensors, vol. 19, no. 16, p. 3523, Aug. 2019.

[25] C. vanLieshout et al., *"Automated River Plastic Monitoring Using Deep Learning and Cameras,"* Earth and Space Science, Vol 7, Iss 8, 2020.

[26] A. Kelly, *"Training an AI to Recognize Cigarette Butts"*, Medium, Jun, 6, 2018, [Online], Tillgänglig: https://medium.com/@aktwelve/training-an-ai-to-recognize-cigarette-butts-5cff9e11c0a7, Hämtad: Januari 2021.

[27] M. Ljungqvist, G. Wetterbrandt, "*Image_Processing/src/FrameExtractor.java*" https://github.com/MarkusLjungqvist/EITL05/tree/master/Image_Processing, Publicerad: 2021-05-11

[28] JCodec Project, "JCodec readme", [2012], Tillgänglig: https://github.com/jcodec/jcodec, Hämtad: 2021

[29] A. Cepalia, "*How computers see images*", Real Python, 2019, [Online], Tillgänglig: https://realpython.com/lessons/how-computers-see-images/, Hämtad 2021-05-06.

[30] M. Ljungqvist, G. Wetterbrandt, "*Image_Processing/src/ImageCentering.java*" https://github.com/MarkusLjungqvist/EITL05/tree/master/Image_Processing, Publicerad: 2021-05-11

[31] V. Patrawala, "*Create A Synthetic Image Dataset — The "What", The "Why" and The "How"*", Towards Datascience, 2020, [Online], Tillgänglig: https://towardsdatascience.com/create-a-synthetic-image-dataset-the-what-the-why-and-the-how-f820e6b6f718, Hämtad: april 2021.

[32] M. Ljungqvist, G. Wetterbrandt, "*Image_Processing/Image_generator.ipynb*" https://github.com/MarkusLjungqvist/EITL05/tree/master/Image_Processing, Publicerad: 2021-05-11

[33] Keras API Reference, "*Image data preprocessing*", 2021 [Online], Tillgänglig: https://keras.io/api/preprocessing/image/#imagedatagenerator-class, Hämtad: 2021-05-11

[34] G. Undone, "*Motion Blur, Shutter Speed, & 180° Shutter Angle // TESTING the RULES*" YouTube, Okt. 19, 2020, [Video]. Tillgänglig: https://www.youtube.com/watch?v=UPPSdCrqcFQ

[35] M. Kortas, "*Creating an AI to Combat Environmental Pollution*", Towards Datascience, April, 2, 2021, [Online],

Tillgänglig: https://towardsdatascience.com/ai-to-combat-environmental-pollution-6d58b0bf6a1, Hämtad: 2021-05-21

[36] M. Wolf et al.,"*Machine learning for aquatic plastic litter detection, classification and quantification (APLASTIC-Q)*", Environmental Research Letters, vol. 15, no. 11, p. 114042, Nov 2020.

[37] D. Jeanbourquin et al. "*Flow measurements in sewers based on image analysis: automatic flow velocity algorithm*", Water Science & Technology, vol. 64, no. 5, p. 1108-1114, 2011.

Terminologi

Maskininlärning *- ett forskningsområde inom artificiell intelligens*
Deep learning *- En gren av maskininlärning*
Neuron *- Enkel matematisk funktion som tar en eller flera parametrar som indata*
Aktiveringsfunktion *- Funktion som skapar utdatan från en enskild neuron. Används i kombination med ett densely connected layer för att göra nätverkets slutgiltiga klassificering.*
Neuronnät *- Ett kluster av flera lager av neuroner.*
Faltande neuronnät *- Ett neuronnät med ett eller flera faltande lager som används för att detektera objekt i bilder.*
Pooling layer *- Ett lager som sammanfattar och förminskar utdatan från ett tidigare lager enligt någon operation. (max, min)*
Flattening layer *- Ett lager som transformerar föregående lagers outputvektor från en godtycklig dimension till en endimensionell vektor.*
Densely connected layer *- Lager som beräknar skalärprodukten av den indata lagret ges tillsammans med någon förutbestämd vikt samt resultatet från aktiveringsfunktionen.*
Binary crossentropy *- En förlustfunktion för att göra binära klassificeringar av objekt*
Sats *- Indelning av data i mindre grupperingar som används för att träna ett neuronnät..*
Epok *- En fullständig träningsperiod där ett neuronnät tränas på hela träningsdatan.*
Overfitting *- Ett fenomen som kan inträffa under träning av ett neuronnät. Neuronnätet överspecialiserar sig på träningsdatan och blir mindre noggrann på validerings- och testdatan.*
Data augmentation *- En teknik för att skapa ny data från redan existerande data.*
Dropout *- Ett lager som introducerar brus i neuronnät.*
Syntetisk datamängd *- En datamängd av bilder som skapats genom sammanfogning av bilder.*
Icke-syntetisk datamängd *- En datamängd av bilder som inte manipulerats.*

Appendix A

All källkod som använts i examensarbetet och de använda datamängderna finns samlade i det här repositoriet på github: https://github.com/MarkusLjungqvist/EITL05

Printed by Tryckeriet i E-huset, Lund 2021

LUND
UNIVERSITY

Series of Bachelor´s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2021-817
http://www.eit.lth.se