

PRODUKTMATCHNING EFFICIENTNET VS. RESNET – EN JÄMFÖRELSE

Kandidatuppsats i Informatik

Emil Malmgren
Elin Järdegar

VT 2021:KANI30



HÖGSKOLAN
I BORÅS

Svensk titel: Produktmatchning EfficientNet vs. ResNet

Engelsk titel: Product matching EfficientNet vs. ResNet

Utgivningsår: 2021

Författare: Elin Järdeemar, Emil Malmgren

Handledare: Henrik Linusson

Abstract

E-commerce is steadily increasing and between the years 2010 and 2014, there was an increase in the number of consumers shopping online from 28,9% to 34,2%. Insufficient information about the price of a product forces buyers to search among several different retailers for the best price. There are different ways to produce the information required to be able to compare prices.

One method to compare prices is automated product matching. This method uses image recognition algorithms where its purpose is to detect, locate and recognize objects in images. Image recognition algorithms often have problems finding objects in images due to external factors such as brightness, viewing angles and if the image contains a lot of unnecessary information. In the past, algorithms such as ANN, random forest classifier and support vector machine have been used, but recent studies have shown that CNN is better at finding important properties of objects that make them less sensitive to these external factors.

Two examples of alternative CNN architectures that have emerged are EfficientNet and ResNet, both of which have shown good results in previous studies, but there is not a lot of research that helps one choose which CNN architecture that leads to the best possible result. Our question is therefore: Which of the EfficientNet and ResNet architectures gives the highest result on product matching with the evaluation measures f1-score, precision, and recall?

The results of the study show that EfficientNet is the overall best architecture for product matching on the dataset. The results also show that ResNet was better than EfficientNet in proposing the right matches for the images. The matches ResNet makes are more accurate than the matches EfficientNet suggests when Resnet received a higher precision than EfficientNet. However, EfficientNet achieves a better recall that shows that EfficientNet is better than ResNet at finding more or all correct matches among its potential matches. The difference in recall is greater than the difference in precision between the models, which means that EfficientNet gets a higher f1-score and is generally better than ResNet, but what is most important can be discussed. Is it important that the suggested matches are correct or that you find all the correct matches? If the most important thing is that the proposed matches are correct, ResNet has an advantage, but if it is more important to find all correct matches, EfficientNet has an advantage. The result therefore depends on what is considered to be most important in determining which of the architectures gives the best results

Keywords: EfficientNet, ResNet, CNN, Convolutional Neural Network, image classification, product matching, price matching, object recognition.

Sammanfattning

E-handeln ökar stadigt och mellan åren 2010 och 2014 var det en ökning på antalet konsumenter som handlar online från 28,9% till 34,2%. Otillräcklig information kring en produkts pris tvingar köpare att leta bland flera olika återförsäljare efter det bästa priset. Det finns olika sätt att ta fram informationen som krävs för att kunna jämföra priser.

En metod för att kunna jämföra priser är automatiserad produktmatchning. Denna metod använder algoritmer för bildigenkänning där dess syfte är att detektera, lokalisera och känna igen objekt i bilder. Bildigenkänningsalgoritmer har ofta problem med att hitta objekt i bilder på grund av yttre faktorer såsom belysning, synvinklar och om bilden innehåller mycket onödigt information. Tidigare har algoritmer såsom ANN (artificial neural network), random forest classifier och support vector machine används men senare undersökningar har visat att CNN (convolutional neural network) är bättre på att hitta viktiga egenskaper hos objekt som gör dem mindre känsliga mot dessa yttre faktorer.

Två exempel på alternativa CNN-arkitekturer som vuxit fram är EfficientNet och ResNet som båda har visat bra resultat i tidigare forskning men det finns inte mycket forskning som hjälper en välja vilken CNN-arkitektur som leder till ett så bra resultat som möjligt. Vår frågeställning är därför: Vilken av EfficientNet- och ResNetarkitekturerna ger det högsta resultatet på produktmatchning med utvärderingsmåttan f1-score, precision och recall?

Resultatet av studien visar att EfficientNet är den över lag bästa arkitekturen för produktmatchning på studiens datamängd. Resultatet visar också att ResNet var bättre än EfficientNet på att föreslå rätt matchningar av bilderna. De matchningarna ResNet gör stämmer mer än de matchningar EfficientNet föreslår då ResNet fick ett högre recall än vad EfficientNet fick. EfficientNet uppnår dock en bättre recall som visar att EfficientNet är bättre än ResNet på att hitta fler eller alla korrekta matchningar bland sina potentiella matchningar. Men skillnaden i recall är större mellan modellerna vilket gör att EfficientNet får en högre f1-score och är över lag bättre än ResNet, men vad som är viktigast kan diskuteras. Är det viktigt att de föreslagna matchningarna är korrekta eller att man hittar alla korrekta matchningar. Är det viktigaste att de föreslagna matchningarna är korrekta har ResNet ett övertag men är det viktigare att hitta alla korrekta matchningar har EfficientNet ett övertag. Resultatet beror därför på vad som anses vara viktigast för att avgöra vilken av arkitekturerna som ger bäst resultat.

Nyckelord: EfficientNet, ResNet, CNN, Convolutional Neural Network, bildklassificering, produktmatchning, prismatchning, objektigenkänning.

Innehållsförteckning

Akronymer	- 2 -
1 Inledning	- 3 -
1.1 Bakgrund.....	- 3 -
1.2 Forskningsöversikt.....	- 3 -
1.3 Problemdiskussion	- 5 -
1.4 Problemformulering, Syfte och forskningsfråga.....	- 5 -
1.5 Målgrupp för arbetet	- 5 -
2 Litteratur	- 6 -
2.1 ANN.....	- 6 -
2.2 CNN.....	- 8 -
2.3 ResNet.....	- 10 -
2.4 EfficientNet.....	- 12 -
2.5 Overfitting.....	- 13 -
2.6 Transfer learning	- 13 -
2.7 ImageNet.....	- 14 -
2.8 Precision.....	- 14 -
2.9 Recall	- 15 -
2.10 F1-score.....	- 15 -
2.11 K-Nearest Neighbor	- 15 -
2.11.1 Cosine similarity.....	- 15 -
2.11.2 Euclidean distance.....	- 16 -
2.11.3 Manhattan distance.....	- 17 -
2.12 Data Augmentation	- 17 -
3 Metod.....	- 19 -
3.1 Forskningsmetod.....	- 19 -
3.2 Ramverk för utvecklingsarbetet	- 19 -
3.3 Experiment.....	- 20 -
3.4 Datainsamling	- 21 -
3.5 Utvärderingsstrategi	- 21 -
3.5.1 Utvärderingsmått	- 22 -
3.6 Metodreflektion.....	- 22 -
4 Genomförande	- 24 -
4.1 Affärsförståelse	- 24 -
4.2 Dataförståelse.....	- 24 -
4.3 Förbehandling av data.....	- 25 -
4.4 Modellering.....	- 25 -
4.5 Hårdvara och andra förutsättningar.....	- 27 -
4.6 Evaluering och resultat.....	- 28 -
5 Analys.....	- 29 -
6 Diskussion och slutsatser	- 33 -

Akronymer

ANN - Artificial neural network

CNN - Convolutional neural networks

TLU – Threshold logic unit

MPL – Multilayer perceptron

ReLU - The rectified linear unit function

KNN - K nearest neighbors

IBL - Instance-based learning

CRISP-DM - Cross Industry Standard Process for Data Mining

1 Inledning

1.1 Bakgrund

E-handeln ökar stadigt och mellan åren 2010 och 2014 var det en ökning på antalet konsumenter som handlar online från 28,9% till 34,2%. Otillräcklig information kring en produkts pris tvingar köpare att leta bland flera olika återförsäljare efter det bästa priset (Konopielko & Radkova 2017). Det finns olika sätt att ta fram informationen som krävs för att kunna jämföra priser. En metod som används är att ha en grupp personer som manuellt matchar produkter hos olika återförsäljare för att sedan jämföra de priser som de erbjuder. Denna metod ger sällan en dålig produktmatchning men den är både kostsam och långsam. En annan metod är att använda sig av en algoritm som automatiskt söker efter produkter och matchar produkter med hjälp av produktens attribut, som till exempel produktnamn och beskrivning. Denna metod är betydligt snabbare och billigare men är mer benägen att göra fel, alltså matcha produkter som inte är av samma typ (Competera.net 2021).

Ett annat sätt att använda automatiserad produktmatchning är att använda algoritmer för bildigenkänning där dess syfte är att detektera, lokalisera och känna igen objekt i bilder. Bildigenkänningsalgoritmer har ofta problem med att hitta objekt i bilder på grund av yttre faktorer såsom belysning, synvinklar och om bilden innehåller mycket onödigt information. Tidigare har algoritmer såsom ANN (artificial neural network), random forest classifier och support vector machine används men senare undersökningar har visat att CNN (convolutional neural network) är bättre på att hitta viktiga egenskaper hos objekt som gör dem mindre känsliga mot dessa yttre faktorer (Madakannu & Selvaraj 2019).

Mayer, Huh, och Cude (2005) skriver i sin rapport att hemsidor som gör det möjligt för konsumenter att jämföra produkter från olika återförsäljare sparar konsumenterna både tid och pengar, men för att detta ska fungera så är det viktigt att informationen som presenteras är korrekt.

Företaget Shopee är ett exempel på företag som vill erbjuda sina kunder en korrekt produktmatchning. Idag jämför Shopee priser med hjälp av traditionella maskininlärningsalgoritmer som analyserar bilder och textinformation men med stora skillnader i bilderna, titlar och produktbeskrivningar hos olika återförsäljare är inte dessa metoder tillräckligt effektiva. De vill därför undersöka om de kan få en mer effektiv lösning genom att förbättra algoritmen som analyserar produktbilderna. Problemet idag är att två liknande bilder kan vara samma produkt eller två helt olika produkter.

1.2 Forskningsöversikt

Bildklassificering handlar om att klassificera och identifiera objekt i bilder. Systemet behöver därför lära sig en robust representation av objektet den skall identifiera oavsett vart i bilden objektet är (Madakannu & Selvaraj 2019). Det finns flera olika metoder i hur man utför bildigenkänning där feature extraction, encoding och pooling-based image representation är de mest klassiska metoderna för bildklassificering (Gao, Duan & Tsang 2016). Enligt Madakannu och Selvaraj (2019) har det tidigare använts mer traditionella maskininlärningsalgoritmer såsom ANN, random forest classifier men att senare forskning visar att CNN är bättre på att hitta samband som ger en mer robust representation av objekt. Även Wu, Chen, Wu, Wu och Huang (2021) anser att CNN är bättre på att extrahera en bilds egenskaper och lära sig mer komplex kunskap än traditionella metoder. Inom det liknande

området ansiktsgenkänning har även CNN visat sig vara ett bättre alternativ än traditionella feature extractions-metoder enligt Chang, Li, Wen, Hu och Ma (2019) studie. Vilket även stärks av Grm, Štruc, Artiges, Caron och Ekenel (2017) där flera olika modeller av CNN ger bra resultat på ansiktsgenkänning men de visar också att dessa modeller är robusta som kan stå emot några av de mest förekommande kvalitetsproblemen med bilder.

CNN har dock sina nackdelar, de behöver stora datamängder för träning, de innehåller miljontals parametrar vilket gör träningsprocessen dyr och tidskrävande samt att använda CNN på mindre datamängder kan resultera i överfitting (Liu, Tian & Xu 2019). För att undvika nackdelarna med CNN kan transfer learning användas för att träna CNN på en mindre datamängd. Transfer learning förbättrar en modell inom en domän genom att överföra information från en liknande domän (Weiss, Khoshgoftaar & Wang 2016). Många studier visar att transfer learning ger bra resultat på mindre datamängder (Kaur & Gandhi 2020; Yang et al. 2019; Liu, Tian & Xu 2019; Madakannu & Selvaraj 2019). CNN utvecklas ständigt för att få bättre resultat på olika datamängder, därför har nya familjer av CNN utformats såsom EfficientNet (Tan & Le 2019) och ResNet (He, Zhang, Ren & Sun 2016). Inom de senaste åren har en del studier fokuserat på EfficientNet. Duong, Nguyen, Di Sipio och Di Ruscio (2020) studie handlar om klassificering av frukter där de visar att en mix av EfficientNet och MixNet ger ett bättre resultat än ett traditionellt CNN. Studien visar också att båda teknikerna drar nytta av transfer learning. I en annan studie vid klassificering av växtsjukdomar från bilder jämförs även här förtränade typer av CNN såsom ResNet och EfficientNet. Slutsatsen är att EfficientNet är den mest lämpade tekniken och uppnår högst accuracy i jämförelse med de andra toppmoderna djupa nätverken (Atila, Uçar, Akyol, & Uçar 2021)

Wang, Yang, Huo, He och Luo (2020) behandlar klassificering av fundussjukdomar, de använder en ensemblemodell baserad på CNN som kan klassificera en eller flera fundussjukdomar på bilden. I undersökningen jämförs olika förtränade CNN och slutsatsen blev att EfficientNet var det mest lämpliga förtränade nätverket att använda. De kommer också fram till att bildstorleken har en betydande roll i modellernas prestationer. De anser att en större bild ger modellerna mer textur och information att arbeta med för att fånga bättre egenskaper i bilden som i sin tur ger modellen en bättre prestanda. Men experimentet visar också att det bara stämmer till en viss grad, vid en punkt kommer större bilder ge en negativ inverkan på modellernas prestanda. Vid användning av EfficientNet har även fördelar med normalisering tekniker såsom Reinhard och Macenko gett ett bättre resultat än ej normaliserad data (Munien & Viriri 2021). Marques, Agarwal och de la Torre Díez (2020) visar även att CNN med EfficientNet-arkitektur och 10-faldig stratifierad korsvalidering ger goda resultat för klassificering av COVID-19 infektion i lungorna.

Det finns också en del artiklar som tar upp ResNet som en lovande arkitektur. Liu, Tian och Xu (2019) använder sig av ResNet och transfer learning för att identifiera och klassificera scener i bilder. Deras förslag ger goda resultat och är bättre än befintliga toppmoderna modeller. ResNet har även visat framgång med att identifiera och klassificera hjärntumörer, där Lu et al. (2020) studie visar att ResNet med ett pyramidutvidgat convolutionlager i botten är en lämplig teknik. Detta för att öka det mottagliga fältet i de ursprungliga nätverket och förbättra klassificeringsnoggrannheten (Lu et al. 2020). ResNet har även används inom ansiktsgenkänning och flera studier har sett lovande resultat även där. Lu, Jiang och Kot (2018) föreslår Coupled-ResNet som ger ett stabilt och bättre resultat än andra toppmoderna modeller, vilket är en variant av ResNet.

1.3 Problemdiskussion

För att lösa problemet produktmatchning med hjälp av bildklassificering anses CNN vara en lämplig metod enligt tidigare forskning (Madakannu & Selvaraj 2019; Wu et al. 2021; Chang et al. 2019) CNN modellen har dock utvecklats mycket de senaste åren vilket har gett upphov till en mängd olika CNN-arkitekturer.

Två exempel på alternativa CNN-arkitekturer som vuxit fram är EfficientNet (Tan & Le 2019) och ResNet (Liu, Tian & Xu 2019). Enligt Atila et al. (2021) visade det sig att EfficientNet gav bättre resultat än både ResNet och traditionella CNN när det gällde att klassificera växtsjukdomar. I en annan studie gjord av Liu, Tian och Xu (2019) så visade det sig att ResNet gav ett bättre resultat än befintliga toppmoderna modeller vid identifiering av scener i bilder. ResNet stärks även av Lu, Jiang och Kot (2018) studie där de föreslår Coupled-ResNet som ger ett stabilt och bättre resultat än andra modeller. Men i en studie av Tan och Le (2019) så var EfficientNet 5.7 gånger snabbare än ResNet.

Att välja vilken teknik att använda är inte alltid enkelt, Guérin, Thiery, Nyiri, Gibaru och Boots (2021) skriver i sin studie att det inte finns forskning som hjälper en välja teknik. De nämner också att en anledning till detta kan vara att CNN ofta leder till bra resultat, vilket kan leda till att dessa goda resultat kan överskugga faktumet att en annan teknik kan leda till ett ännu bättre resultat.

Båda CNN-arkitekturerna har visat bra resultat i tidigare forskning. ResNet-arkitekturen används i stor utsträckning (Tan & Le 2019) och har bland annat visat lovande resultat inom ansiktsgenkänning (Lu, Jiang & Kot 2018). Men det finns också studier som stödjer EfficientNet överlägsna snabbhet och resultat (Tan & Le 2019; Atila et al. 2021). Precis som Guérin et al. (2021) tar upp blir det svårt att välja vilken teknik som är mest lämplig att använda sig av.

1.4 Problemformulering, Syfte och forskningsfråga

Som tidigare beskrivet så anses CNN vara en kraftfull teknik för bildklassificering och det finns flera olika CNN-arkitekturer, men det finns inte mycket forskning som hjälper till att välja vilken CNN-arkitektur som leder till ett så bra resultat som möjligt.

Syftet med studien är att jämföra CNN-arkitekturerna EfficientNet och ResNet på ett produktmatchningsproblem för att se vilken arkitektur som är mest lämplig.

De frågor som denna rapport skall besvara är:

- ❖ Vilken av EfficientNet- och ResNetarkitekturerna ger det högsta resultatet på produktmatchning med utvärderingsmåten f1-score, precision och recall?

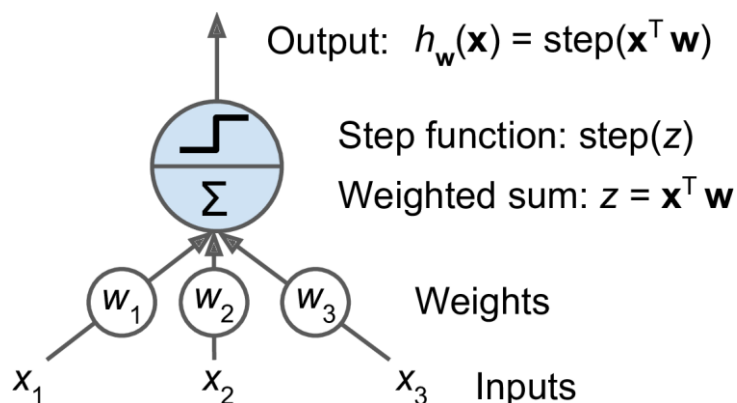
1.5 Målgrupp för arbetet

Intressenter för studien är främst företag och forskare inom produktmatchningsproblem. Där studien kan visa både vilken arkitektur som är mest lämplig att använda och de resultat man kan uppnå med dessa två arkitekturer inom problemdomänen. Men också till studenter inom maskininläring.

2 Litteratur

2.1 ANN

ANN är en datamodell som är inspirerad av den mänskliga hjärnan, precis som en hjärna består ANN:s också av neuroner och kopplingar (Kalogirou 1999). En av de enklaste arkitekturerna av ANN är the perceptron som bygger på en artificiell neuron, threshold logic unit (TLU). TLU neuronerna beräknar en viktad summa av inputen och sedan appliceras en step funktion på den summan som ger output resultatet: $h_w(x) = \text{step}(z)$ där $z = x^t w$ (Géron 2019).



Figur 1: Threshold logical unit (Géron 2019)

The perceptron är sammansatt av ett lager av TLUs, med varje TLU neuron kopplad till varje input. En extra neuron är oftast tillagd, en så kallad bias neuron. När alla neuroner är kopplade till varje neuron i nästa lager kallas det ett fully connected layer. Vid beräkningen av output från ett fully connected layer används ekvationen nedan:

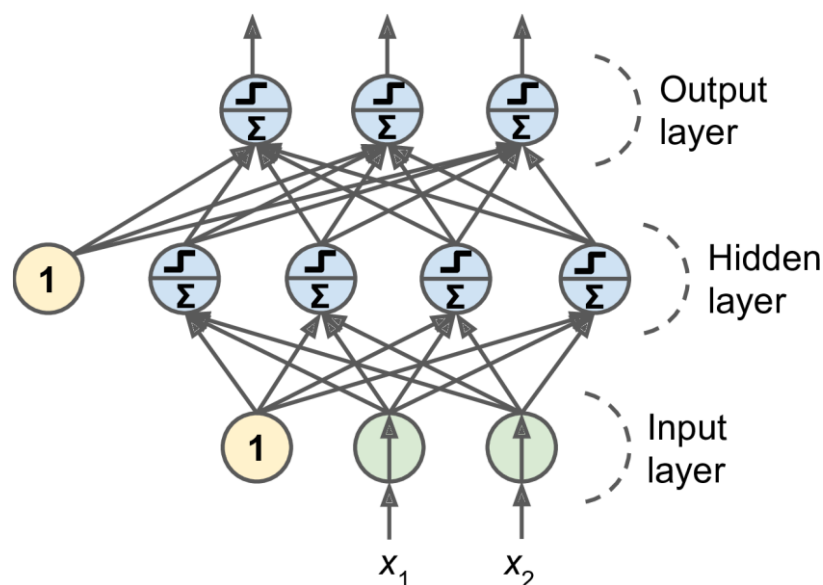
$$h_{w,b}(x) = \phi(xw + b)$$

I ekvationen representeras x som en matris av input attribut, en rad per instans och en kolumn per attribut. W står för viktmatrisen, som innehåller alla kopplingars vikter förutom bias-vikten. B står för bias-vektorn, här finns alla kopplingars vikter mellan bias neuronerna och de artificiella neuronerna. Det finns också en bias vikt för varje artificiell neuron i nätverket. Funktionen ϕ är en så kallad aktiveringsfunktion och när det gäller TLU:s är detta en step funktion.

Vid träning av perceptron nätverk innebär det att hitta de rätta vikterna för nätverket. Träningen av nätverket bygger på Hebb's rule, vilket säger att kopplingens vikt mellan två neuroner ökar när de har samma output. Perceptron nätverk använder en variant av denna regel genom att även ta hänsyn till fel som nätverket gör. Den förstärker kopplingar som reducerar felet. En instans matas genom nätverket och ger en prediktion. För varje output neuron som ger en felaktig prediktion förstärks kopplingens vikt från den input som skulle bidragit till rätt prediktion (Géron 2019).

En vanlig ANN arkitektur är multilayer perceptron (MLP). En MLP består av ett inputlager, ett eller flera lager av TLU:s (hidden layers) och ett sista lager av TLU:s kallat outputlager (Géron 2019). Varje lager innehåller ett godtyckligt antal neuroner där varje neuron har en koppling, även kallad vikt, till alla neuroner i föregående lager. Dessa neuroner använder de värden och vikter som tidigare lager genererat för att beräkna fram ett nytt värde att skicka vidare i nätverket (Kalogirou 1999).

Träningsprocessen går ut på att nätverket först beräknar fram en output, därefter jämför modellen sin output med förväntad output för att sedan stega bakåt i nätverket och uppdatera vikterna för att få en output närmare verkligheten (Kalogirou 1999). En mer ingående beskrivning av träningsprocessens steg: Algoritmen hanterar en delmängd av träningsmängden åt gången och går igenom hela träningsmängden flera gånger. Varje gång den tar sig igenom datamängden kallas det för en epok. Varje delmängd skickas genom nätverkets inputlager, som endast skickar vidare till det första hidden layer. Algoritmen beräknar sedan outputen för hela det lagret och skickar vidare till nästa lager o.s.v. tills nätverket når outputlagret. Sedan beräknas algoritmens outputfel genom att använda en loss funktion som jämför resultatet från nätverket med det önskade resultatet. Därefter beräknas hur mycket varje outputkoppling bidrar till felet genom att applicera kedjeregeln. Algoritmen mäter hur mycket varje koppling till lagret under har bidragit med felet tills att algoritmen har nått inputlagret. Till slut genomförs ett gradient descent steg för att justera vikterna av alla kopplingar i nätverket. I MLP fungerar inte en step funktion och den är i stället utbytt med en aktiveringsfunktion. De mest förekommande aktiveringsfunktionerna är logistic function $\sigma(z) = 1/(1 + \exp(-z))$, the hyperbolic tangent function $\tanh(z) = 2\sigma(2z) - 1$ och the rectified linear unit function (ReLU) $ReLU(z) = \max(0, z)$ (Géron 2019).

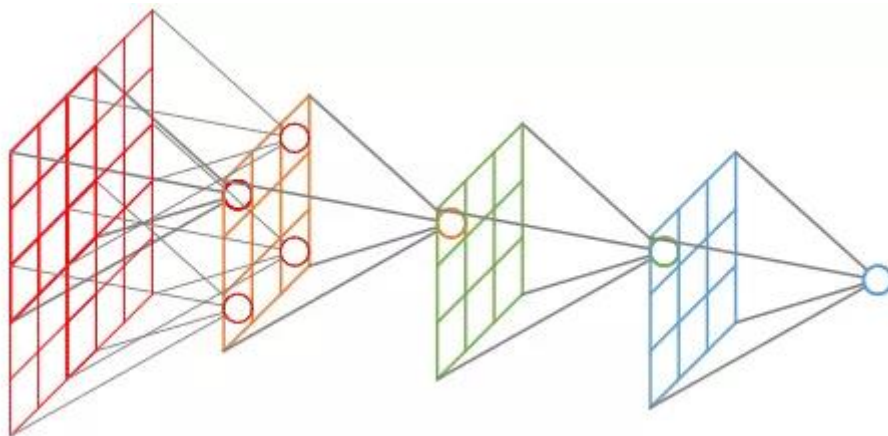


Figur 2: MLP arkitektur (Géron 2019)

Det finns flera olika arkitekturer av ANN, där CNN är en modifierad arkitektur som liknar multilayer feedforward. I stället för att varje nod i är kopplad till alla noder i föregående lager så är CNN utformad i en hierarkisk struktur där varje nod i lagret enbart hämtar information från en delmängd av de noder i föregående lager (Sharifzadeh, Akbarizadeh & Seifi Kaviani 2019).

2.2 CNN

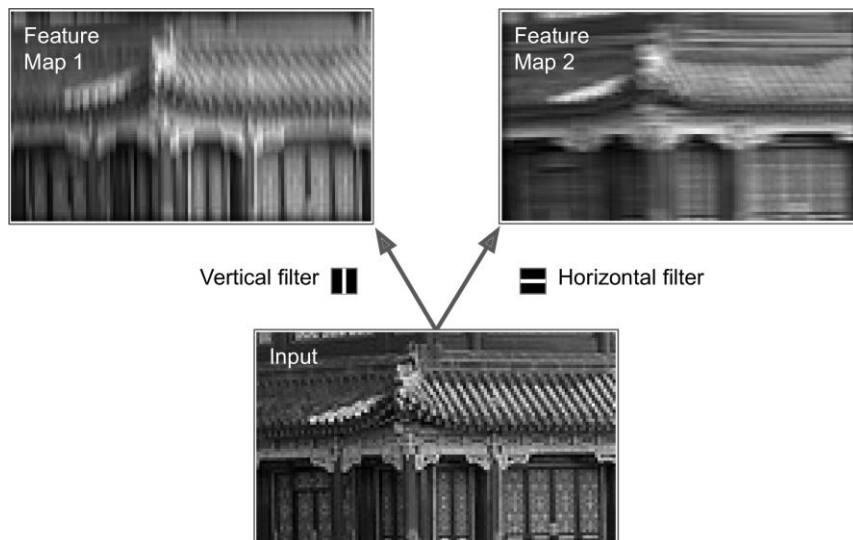
CNN består av flera lager där varje lager delar upp bilden i så kallade receptive fields och en av de viktigaste egenskaperna för ett CNN är dess convolutional-lager. Dessa lager är inte fullt kopplade till föregående lager utan bara till ett antal receptive fields i föregående lager. Receptive fields motsvarar vilken del av bilden som nätverket kollar på när det letar efter egenskaper i bilden. Denna information skickas sedan vidare till nästa lager där flera receptive fields från föregående lager sätts ihop till ett sammanfogat receptive field i nästa lager. Denna process fortsätter framåt i lagerstrukturen, vilket innebär att vid tidiga lager så kollar nätverket bara på en liten del av bilden för att sedan i senare lager kolla på en större del av bilden där information från tidigare lager tas i åtanke (Vo & Lee 2018), vilket enligt Géron (2019) är anledningen till att CNN är bra på bildigenkänning.



Figur 3: Exempel på receptive fields (Vo & Lee 2018)

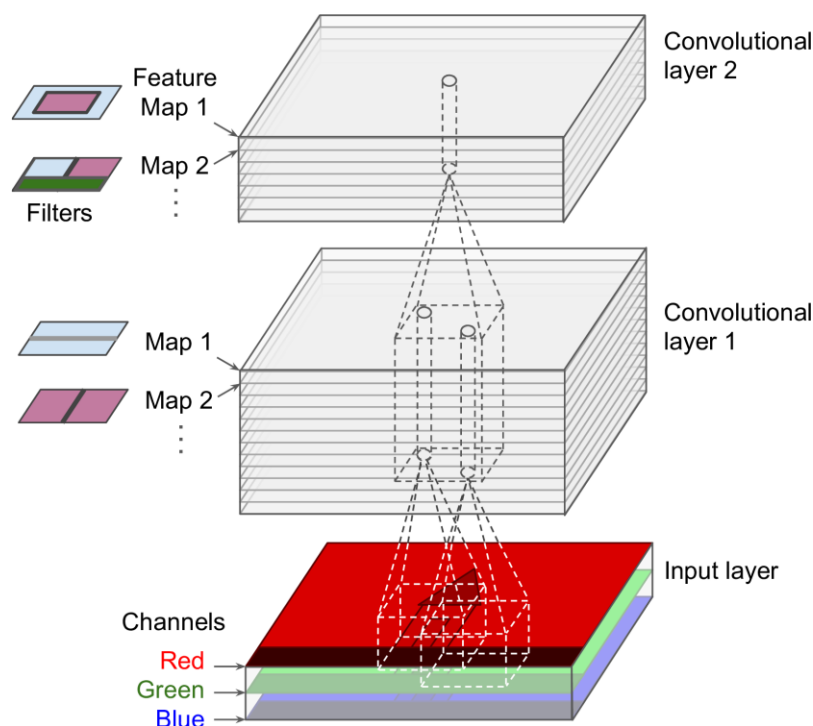
Precis som i neurala nätverk så kan det gömda lagret innehålla flera lager men i ett CNN så har alla noder i ett lager en gemensam viktning och bias. Detta innebär att alla noder i lagret reagerar på samma sätt. Anledningen till att man har denna struktur är för att nätverket inte känner till vart på bilden som det objekt det skall identifiera befinner sig (Farabet, Couprie, Najman & LeCun 2013).

Neuronernas vikter kan representeras som en liten bild med samma storlek som neuronens receptive field så kallat filter eller convolutional kernels. Ett lager som använder samma filter ger en output i form av en feature map som förstärker områden i bilden som aktiverar filtret mest. Ett exempel på detta illustreras av figur 4 där två olika uppsättningar av vikter/filter används. Den första representeras av en svart fyrkant med en vertikal vit linje i mitten. En 7x7 matris med nollor förutom den centrala kolumnen som består av ettor. Neuroner som använder dessa vikter kommer att ignorera allting i deras receptive field utom den centrala vertikala linjen. Vilket sker genom att all input kommer att multipliceras med noll förutom de som finns i den vertikala linjen som kommer multipliceras med 1. För den andra bilden i exemplet används i stället ett filter med en horisontell vit linje i mitten.



Figur 4: Exempel på olika filter (Géron 2019)

Varje convolutional layer består av flera filter som ger en feature map per filter. För att göra detta finns det en neuron per pixel i varje feature map och alla neuroner i varje feature map delar parametrar i form av vikter och bias term. Men neuroner i olika feature maps använder olika parametrar. En neurons receptive field sträcker sig över alla tidigare lagars feature maps. Vilket leder till att ett convolutional layer samtidigt applicerar flera träningsbara inputs som gör det möjligt att detektera flera egenskaper vartsomhelst i bilden. Varje bild är även sammansatt av olika lager av färger så kallade color channel (Géron 2019).

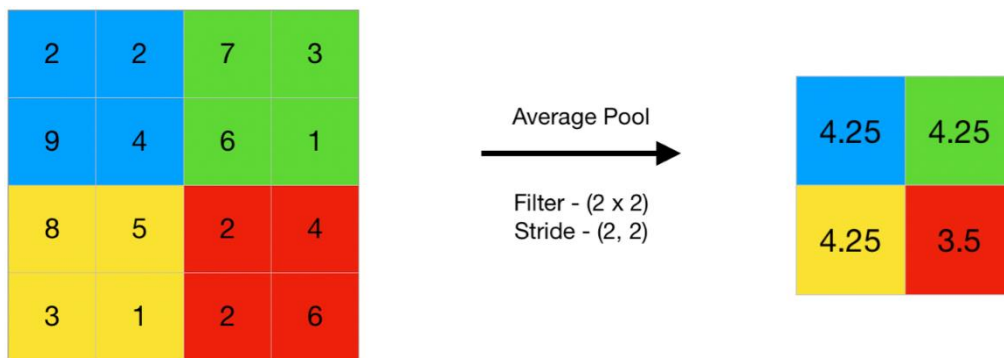


Figur 5: Convolutional layer med flera feature maps och tre color channels (Géron 2019)

Detta kan i sin tur kräva väldigt mycket kraft att utföra dessa beräkningar, för att snabba upp dessa beräkningar används pooling. Pooling bidrar också till att bilden blir mer oföränderlig

och robust mot brus (Chang et al. 2019). Detta gör man genom att kombinera flera outputs från flera noder i tidigare lager till en enkel nod i nästa lager (Géron 2019). Pooling kallas även sub sampling eller down sampling layer och kan utan att påverka informationen filtrera bort ungefär 75% av all information då det finns så pass stor del onödig information i en bild. Pooling är också en vital del för att undvika överfitting, minska beräkningstiden och öka igenkänningsnoggrannheten (Akhtar & Ragavendran 2019).

Figur 6 är en illustration av Average Pooling, vilket innebär att medelvärdet från de tidigare receptive fields matas in i nästa lager vilket kan medföra effektivisering, minskad risk för överfitting och öka igenkänningsnoggrannheten (Chang et al. 2019; Akhtar & Ragavendran 2019).

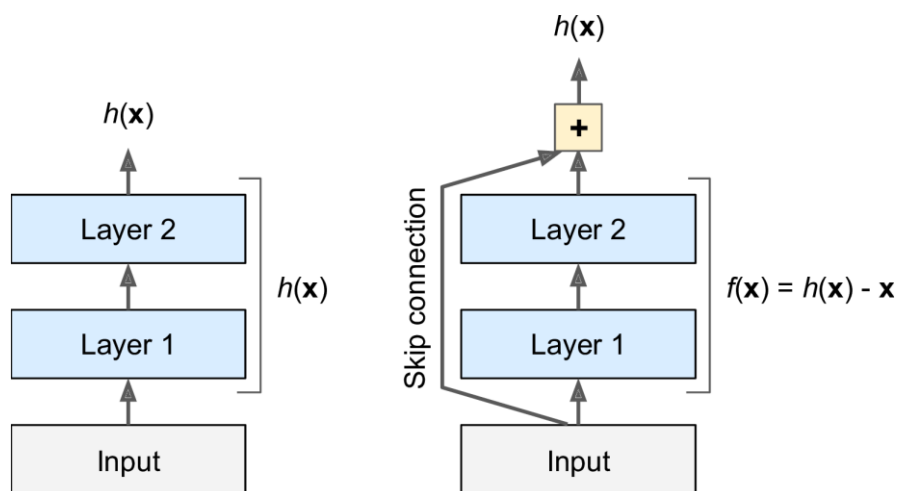


Figur 6: Illustration av Average Pooling (GeeksforGeeks, 2019)

2.3 ResNet

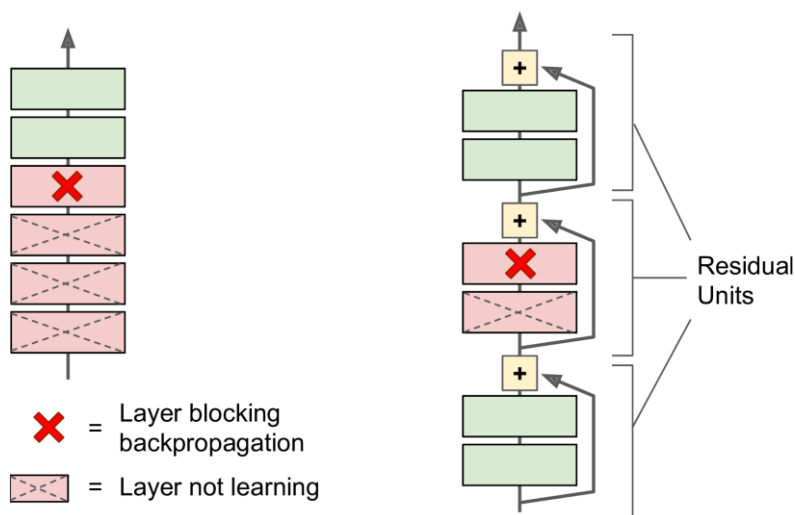
ResNet är en vidareutveckling av CNN med en annorlunda arkitektur för att lösa de problem som har identifierats i djupa nätverk. Forskning har visat att man med fördel kan öka nätverkens djup för att öka träffsäkerheten (engelska: accuracy) då egenskaper kan berikas med ökade antal lager (Simonyan & Zisserman 2014). När djupa nätverk är kapabla till att börja konvergera har ett degraderingsproblem identifierats. Att konvergera menas att vikterna i det neurala nätverket börjar stabiliseras och uppnår ett optimalt värde. Degraderingsproblemet som är identifierat handlar om att med ökat djup på nätverket blir modellens träffsäkerhet mättad och försämras sedan snabbt. Nedbrytningen av modellens träffsäkerhet orsakas inte av överfitting och att lägga till fler lager till en lämpligt djup modell leder till högre träningsfel (He & Sun 2014; He, Zhang, Ren & Sun 2016).

ResNet är en lösning på nedbrytningen av träffsäkerheten. Genom att implementera genvägsanslutningar mellan lagren i arkitekturen. Med dessa genvägsanslutningar kan lager som försämrar arkitekturs resultat hoppas över. På detta sätt kan väldigt djupa nätverk tränas utan att höga träningsfel och undvika nedbrytande av träffsäkerheten (He et al. 2016). Genvägsanslutningen gör att signalen som matas in i lagret också läggs till i outputen av lagret som finns lite högre upp i stacken. När man tränar ett neuralt nätverk är målet att göra det till en target function $h(x)$. Om man lägger in inputen x till outputen av nätverket det vill säga lägga till en genvägsanslutning, då kommer nätverket vara tvunget att modellera $f(x) = h(x) - x$ i stället för $h(x)$. Detta kallas residual learning och illustreras av bilden nedan (Géron, 2019).



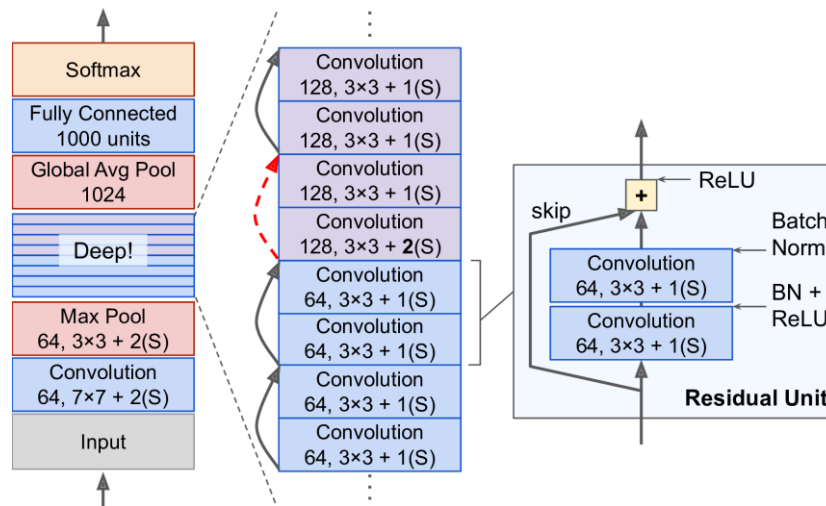
Figur 7: Residual learning (Géron 2019)

Vid initialisering av neurala nätverk är vikterna oftast nära noll, vilket gör att nätverkets outputvärden är nära noll. Om man lägger till en genvägsanslutning blir alltså resultatet att nätverket ger en output som är en ren kopia av dess input. Om target function är ganska lik identity function (vilket ofta är fallet) kommer genvägsanslutningarna att snabba upp träningen av nätverket väsentligt. Om man lägger till många genvägsanslutningar till nätverket kan nätverket göra framsteg även om flera lager inte har börjat inlärningen än. ResNet kan ses som en stack av residual units, där varje residual unit är ett litet neuralt nätverk med en genvägsanslutning (Géron 2019).



Figur 8: Vanligt neuralt nätverk (vänster) och ett residual nätverk (höger) (Géron 2019)

ResNet arkitekturen består av ett inputlager som följs av ett convolutional lager och ett max pool lager. Sedan kommer en väldigt djup stack av enkla residual units. Varje residual unit består av två convolutional layer med batch normalisering och ReLU aktivering. Efter stacken med residual units finns ett global average pooling layer, ett fully connected layer med 1000 units samt ett softmax lager som outputlager (Géron 2019).



Figur 9: ResNet arkitektur (Géron 2019)

2.4 EfficientNet

EfficientNet är också en variant av CNN med en annorlunda arkitektur och förmåga att skala upp nätverket. Det finns många sätt att skala upp CNN:s. De mest vanligt förekommande sätten är att öka nätverkets djup genom att lägga till flera lager (He et al. 2016), öka nätverkets bredd genom att lägga till fler noder i varje lager (Zagoruyko & Komodakis 2016), eller genom att skala upp modellen efter bildupplösning (Huang, Cheng, Chen, Lee, Ngiam, Le & Chen 2018). Dock har processen av att skala upp CNN inte varit väl förstådd och det finns många sätt att göra det på. I tidigare forskning har man endast skalat upp nätverket på en dimension djup, bredd eller bildstorlek. Det är möjligt att skala nätverket på två eller tre dimensioner, men då krävs godtycklig skalning och manuell inställning vilket ger suboptimal träffsäkerhet (engelska: accuracy) och effektivitet (Tan & Le 2019).

Genom att balansera skalningen av alla tre dimensionerna med ett konstant förhållande, så kallat compound scaling method kan CNN:s uppnå en högre träffsäkerhet och effektivitet (Tan & Le 2019).

$$\begin{aligned}
 \text{depth} : d &= \alpha^\theta \\
 \text{width} : \omega &= \beta^\theta \\
 \text{resolution} : r &= \gamma^\theta \\
 \text{s.t } \alpha \cdot \beta \cdot \gamma^2 &\approx 2 \\
 \alpha \geq 1, \beta \geq 1, \gamma &\geq 1
 \end{aligned}$$

Compound scaling method består av en sammansatt koefficient θ . För att uniformt skala alla tre dimensionerna. Där α , β och γ är konstanter som kan bestämmas via en sökning av rymden. θ är en användbar specificerad koefficient som kontrollerar hur mycket mer resurser som finns tillgängliga för att skala upp nätverket. Där α , β och γ specificerar hur resurserna ska fördelas mellan dimensionerna bredd, djup och upplösning. EfficientNet bygger på denna skalningsmetod samt en ny arkitektur. Compound scaling method påverkar inte lageroperationerna och därför ger en justering av arkitektur även en ökning i träffsäkerhet och effektivitet (Tan & Le 2019).

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels \hat{C}_i	#Layers \hat{L}_i
1	Conv3x3	224×224	32	1
2	MBCConv1, k3x3	112×112	16	1
3	MBCConv6, k3x3	112×112	24	2
4	MBCConv6, k5x5	56×56	40	2
5	MBCConv6, k3x3	28×28	80	3
6	MBCConv6, k5x5	14×14	112	3
7	MBCConv6, k5x5	14×14	192	4
8	MBCConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figur 10: EfficientNet arkitektur (Tan & Le 2019)

2.5 Overfitting

Ett centralt problem inom maskininlärning och supervised learning är inlärningen från märkt träningsdata. Ett exempel kan vara vid modellering av en medicinsk diagnos, där träningsdatan består av patienter med deras ärendejournaler och medicinska diagnoser. Modellens jobb är att härleda en funktion som förutsäger patientens diagnos baserad på hans ärendejournaler. Funktionen som ska läras kan representeras som en uppsättning regler. Modellen är tränad på en uppsättning träningsdata men sedan ska den tillämpas på ny data för att göra en prediktion. Målet är att uppnå så hög träffsäkerhet (engelska: accuracy) på den nya data och inte nödvändigtvis på träningsdatan. Arbetar algoritmen för hårt för att hitta den bästa passformen för träningsdatan så finns den en risk att den anpassas efter brus i träningsdatan i stället för att hitta en mer generell regel för egenskaperna. Detta problem kallas overfitting (Dietterich 1995).

Om modellen matchar träningsdatan väl men misslyckas med att predicera nya instanser i problemområdet, är det typiskt för overfitting. Overfitting händer när modellen koncentrerar för mycket på detaljerad information från träningsdatan vilket påverkar modellen förmåga att generalisera negativt. Motsatsen till detta är underfitting. Vilket uppstår när modellen inte är komplicerad nog och för lite fokus på träningsdatan. Resultatet blir att modellen varken passar träningsdatan eller ny data (Czajkowski & Kretowski 2019).

2.6 Transfer learning

Traditionell maskininlärning är karakteriserad av att träningsdata och testdata har samma egenskapsrymd och samma datadistribution. När träningsdatan och testdatan har skillnader i datadistributionen kan en prediktiv modell bli försämrad (Shimodaira 2000). Men informationen som behövs för att träna och testa sin modell kan vara svår och dyr att samla in. Vilket är motiveringen till transfer learning. Transfer learning används för att förbättra en modell från en domän genom att överföra information från en liknande domän (Weiss, Khoshgoftaar & Wang 2016). Definition av transfer learning är, givet en källdomän D_S och inlärningsuppgift T_S , måldomän D_T och inlärningsuppgift T_T . Transfer learnings målsättning är att förbättra inlärningen av målpredictionsfunktionen $f(\cdot)$ i D_T genom att använda kunskapen i D_S och T_S där $D_S \neq D_T$ eller $T_S \neq T_T$. $D_S \neq D_T$ betyder att antingen är egenskapstermerna olika eller att den marginella distribueringen är olika. $T_S \neq T_T$ är när

domänerna är olika och antingen egenskapsrymden är olika i domänerna eller att sannolikhetsdistribueringen är olika. När något av påståendena $D_S \neq D_T$ eller $T_S \neq T_T$ är sanna är det transfer learning (Pan & Yang 2010).

Till exempel vid text-sentimentanalys av produktomdömen när det finns mycket märkt data från digitala kameror. Om både träningsdatan och måldatan kommer från recensioner om digitala kameror är det inte transfer learning då $D_S = D_T$ och $T_S = T_T$ då det handlar om traditionell maskininlärning. Men om i stället träningsdatan är från recensioner för digitala kameror men måldatan är från recensioner av mat, då är det transfer learning efter som $D_S \neq D_T$ är uppfyllt. Datan är från olika men liknande domäner som kan användas då datan har flera egenskaper gemensamt (Weiss, Khoshgoftaar & Wang 2016). Inom CNN kan transfer learning används med fördel genom att tränas på en stor datamängd såsom ImageNet, för att sedan använda dess vikter som initiala vikter till en ny klassificeringsuppgift. Typiskt bara vikterna i convolutional lager och inte alla vikter i hela nätverket. Detta är väldigt effektivt då många datamängder med bilder delar rumsliga egenskaper på låg nivå som nätverken lär sig bättre med hjälp av stora datamängder (Shorten & Khoshgoftaar 2019).

2.7 ImageNet

ImageNet är en storskalig bilddatabas. ImageNet är byggt på en hierarkisk struktur och målet är att ha 50 miljoner uppmärkta bilder med full upplösning (Deng et al. 2009). Den innehåller idag över 14 miljoner bilder inom mer än 20 000 kategorier (image-net 2021). Skaparna bakom ImageNet föreställde sig några olika möjliga applikationsområden. Ett möjligt applikationsområde är som träningsresurs. Många andra större datamängder visar en bias i deras täckning av olika typer av objekt, medan ImageNet innehåller ett stort antal bilder från nästan alla objektklasser till och med sällsynta. Vilket gör att ImageNet kan ge en fördel vid överföring av kunskap. Ett annat applikationsområde är som en referensdatamängd, med sina förnämliga egenskaper såsom hög kvalitet, mångfald och stor skala kan ImageNet främja objektigenkänning och scenklassificering (Deng et al. 2009).

2.8 Precision

Precision är ett utvärderingsmått vid klassifikationer. Det är träffsäkerheten (engelska: accuracy) för de positiva prediktionerna. Precision är representerat som ekvationen nedan där TP är true positives och FP är false positives.

$$Precision = \frac{TP}{(TP + FP)}$$

Precision fångar hur stor del av de positiva prediktionerna som faktiskt är korrekt. Måttet säger hur säkra vi kan vara att den positivt predicerade instansen faktiskt är av den positiva klassen. Måttets värde går från 0–1 där ett högre värde indikerar en bättre modell (Kelleher, MacNamee & D’Arcy 2015).

2.9 Recall

Recall används också som ett utvärderingsmått vid klassificeringsproblem. Recall är representerat som ekvationen nedan där TP är true positives och FN är false negatives.

$$Recall = \frac{TP}{(TP + FN)}$$

Recall säger oss hur säkra vi kan vara att alla instanser med den positiva klassen har hittats av modellen. Måttets värde går från 0–1 där ett högre värde indikerar en bättre modell (Kelleher, MacNamee & D’Arcy 2015).

2.10 F1-score

F1-score är ett sammansatt mått av recall och precision. F1-score är det harmoniska medelvärdet av precision och recall och definieras enligt:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

F1-score har också en skala på 0–1 där ett högre värde indikerar en bättre modell (Kelleher, MacNamee & D’Arcy 2015).

2.11 K-Nearest Neighbor

KNN är en variant av en vanligt förekommande metod kallat instance-based learning (IBL) för maskininlärning. Vilket betyder att metoden använder specifika träningsinstanser för att göra en prediktion utan att ha byggt en modell av träningsdatan. IBL-algoritmer kräver ett avståndsmått/likhetsmått för att beräkna likheten eller avståndet mellan datapunkterna och en klassifikationsfunktion som ger den predicerade klassen baserat på avstånd till datapunkterna. En KNN-klassificerare representerar varje data instans som en punkt i en d-dimensionell rymd, där d är antal attribut. För en given testinstans beräknas avståndet/likheten med övriga datapunkter i rymden. För att avgöra vilken klass som testinstansen tillhör används k antal grannar med i beräkningen för klassificeringen. Testinstansen får den klasstillhörighet som majoriteten av de närmsta k antalet grannarna har. Ovan förklaring av KNN är en supervised metod då KNN har tillgång till klasstillhörigheten för insatserna (Munter et al. 2016). En unsupervised KNN får inte tillgång till klasstillhörigheten, utan skapar kluster av instanserna som liknar varandra (Cariou & Chehdi 2015).

2.11.1 Cosine similarity

Cosine similarity är ett likhetsmått som kan användas vid klassificering och clustering, alltså lämpligt att använda i en KNN-modell. Cosine similarity är ett vanligt och förekommande likhetsmått som är enkelt och effektivt. Måttet fokuserar på riktningar och beräknar cosine av vinkeln mellan två vektorer. Två liknande vektorer är förväntade att ha en liten vinkel mellan dem. Definitionen av cosine similarity är där mönster x och x' definieras av:

$$\cos(\theta) = \frac{\sum_{i=1}^d x_i \times x'_i}{\sqrt{\sum_{i=1}^d x_i^2} \times \sqrt{\sum_{i=1}^d x_i'^2}}$$

Där θ är vinkeln mellan x och x' . Om likheten mellan mönstren ökar, ökar även θ . Det finns nackdelar med cosine similarity genom att den endast fokuserar på orienteringen mellan mönster. Vilket kan illustrera vid till exempel ansiktsgenkänning, vid beräkning av cosine similarity på två bilder av samma ansikte med endast en variation i ljusintensitet. Cosine similarity beräknar att dessa bilder inte är samma bild (Xia, Zhang & Li 2015).



Figur 11: Cosine Similarity (Xia, Zhang & Li 2015)

2.11.2 Euclidean distance

Euclidean distance är det vanligaste och ett av de enklaste avståndsmåtten. Avståndet mellan två punkter är längden av linjesegmentet som förbinder dem. För två mönster x och x' är den euclidean distance:

$$d(x, x') = \sqrt{\sum_{i=1}^d (x_i - x'_i)^2}$$

Likhetssannolikheten minskar när euclidean distance ökar. Dock underpresterar euclidean distance i några verkliga problem tack vare känsligheten till magnituden. Den är känslig för små deformationer i till exempel bildklassificering. Ett exempel är vid tre ansiktusbilder där två bilder är liknande, bild 1 och bild 2, samt en som är inte alls är lik de övriga bilderna, bild 3. Det euklidiska avståndet är mindre mellan de helt olika bilderna bild 2 och bild 3 än mellan bild 1 och bild 2, de mest liknande bilderna (Xia, Zhang & Li 2015).

Bild 1



Bild 2



Bild 3



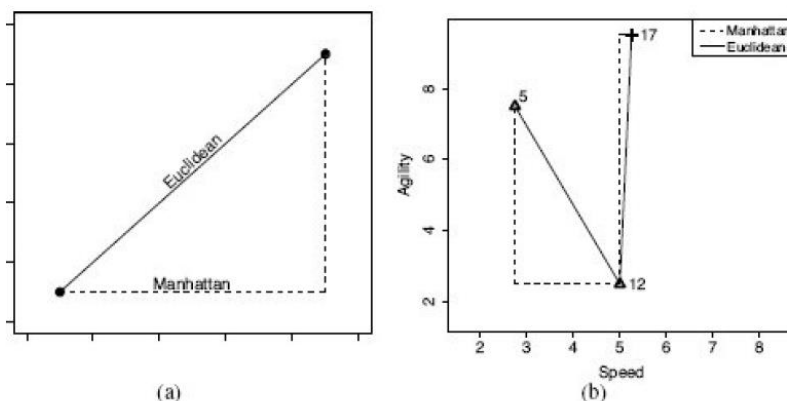
Figur 12: Exempel Euclidean distance (Xia, Zhang & Li 2015)

2.11.3 Manhattan distance

Manhattan distance är också ett vanligt avståndsmått att använda vid KNN. Den har ett övertag mot euclidean distance då den inte kräver lika mycket beräkningskraft, vilket är bra vid stora datamängder. Avståndsmåttet kallas också taxi-cab distance bara för att det är distansen som en taxi skulle behöva köra för att ta sig från en punkt till en annan i ett vägsystem av kvarter likt Manhattans vägsystem. Euclidean distance är den kortaste vägen mellan två punkter medan manhattan distance är summan av det verkliga avståndet mellan punkterna, där linjerna alltid är raka. Manhattan distance definieras enligt nedan där x är en punkt i och x' är en annan punkt (Kelleher, MacNamee & D'Arcy 2015).

$$Manhattan(x, x') = \sum_{i=1}^m (abs(x[i] - x'[i]))$$

Bilden nedan illustrerar skillnaden mellan euclidean distance och manhattan distance.



Figur 13: Skillnad mellan manhattan och euclidean distance (Kelleher, MacNamee & D'Arcy 2015)

2.12 Data Augmentation

Data augmentation inom dataanalys är en teknik som används för att utöka mängden av tillgänglig data genom att lägga till ny data som är en modifikation av tidigare data. Detta hjälper modellen motverka överfitting i träningsprocessen (Shorten & Khoshgoftaar 2019).

När data augmentation utförs på bilder modifieras tidigare bilder genom att vända, rotera, ändra färger, ändra höjd och bredd, zooma in/ut på tidigare bilder och sedan spara ner den modifierade versionen till datamängden (Shorten & Khoshgoftaar 2019).



Figur 14: Exempel på data augmenterade bilder med färgmodifikation (Shorten & Khoshgoftaar 2019)

3 Metod

3.1 Forskningsmetod

Syftet med studien är att ta reda på vilken arkitektur av CNN-arkitekturerna EfficientNet och ResNet som ger bäst resultat på produktmatchningsproblemet. För att svara på frågan kommer två modeller att tas fram genom design science och arbetsprocessen CRISP-DM (Cross Industry Standard Process for Data Mining). CRISP-DM är en cyklisk process som delar upp arbetsprocessen i 6 olika delar, business understanding, data understanding, data preparation, modeling, evaluation och deployment. CRISP-DM anses vara en relevant och omfattande process för dataanalysprojekt (Jaggia, Kelly, Lertwachara & Chen 2020).

När de två modellerna är utvecklade kommer en kvantitativ metod tillämpas med hjälp av experimentell forskning, för att ta reda på vilken arkitektur är mest lämplig att använda på produktmatchningsproblemet.

Experimentell forskning grundar sin forskning på relationen mellan orsak och verkan. Ett exempel på experimentell forskning kan vara när två olika behandlingsgrupper ingår i ett test där bara den ena får behandling, för att sedan jämföra båda gruppernas tillfrisknande (Recker 2013).

Denna metod är lämplig för vår undersökning då två olika modeller separat kommer att testas med samma utgångsläge. Med hjälp av denna metod kommer undersökningen förhoppningsvis kunna ge en bättre förståelse för vilken arkitektur som är mest lämplig att använda på produktmatchningsproblem.

3.2 Ramverk för utvecklingsarbetet

Hevner, March, Park och Ram (2004) har identifierat sju riktlinjer vid design science, som ger vägledning av nyckelprinciper som design science följer. Nedan listas dessa riktlinjer och hur vår studie uppfyller dessa i utvecklingen av en EfficientNet- och ResNetmodell.

Första riktlinjen handlar om design som en artefakt. Design science forskning måste producera en användbar artefakt i form av en konstruktion, modell, metod eller installation. Vår studie kommer att bygga på att ta fram två maskininlärningsmodeller och jämföra modellerna för att se vilken som kan ge bäst resultat på produktmatchning. Studien uppfyller därför riktlinje ett genom att ta fram två artefakter i form av metoder som kan lösa produktmatchningsproblem.

Andra riktlinjen handlar om problemrelevansen. Design science forskningsmål är att utveckla en teknologibaserad lösning till viktiga och relevanta affärsproblem. Det affärsproblem som denna studie bygger på är prisjämförelse där företag vill förbättra hur de utför deras produktmatchning. Detta kan i sin tur leda till att de kan utföra en bättre prisjämförelse. Studien hanterar affärsproblemet genom att ta fram metoder som kan förbättra produktmatchningen, vilket leder till en förbättrad prisjämförelse.

Tredje riktlinjen är designutvärdering. Användningen, kvaliteten och effektiviteten hos en designartefakt måste vara noggrant demonstrerat via väl genomförda utvärderingsmetoder. För att mäta och jämföra våra modeller kommer måttet f1-score, precision och recall att användas.

Fjärde riktlinjen är forskningsbidrag. Effektiv design science forskning måste ge tydliga och verifierbara bidrag inom områdena designartefakt, designfundament och / eller designmetoder. Syftet med studien är att undersöka vilken arkitektur av CNN som är mest lämplig att använda vid produktmatchning. Genom att jämföra två artefakter bidrar studien till produktmatchningsdomänen med en lämplig designmetod.

Femte riktlinjen handlar om forskningens noggrannhet. Design science forskning bygger på tillämpningen av rigorösa metoder för både konstruktion och utvärdering av den designade artefakten. Artefakten som produceras i studien konstrueras genom den vedertagna CRISP-DM processen och utvärderas genom f1-score, recall och precision.

Sjätte riktlinjen är att design science ska vara som en sökprocess. Sökandet efter en effektiv artefakt kräver att man använder tillgängliga medel för att nå önskade mål samtidigt som lagarna i problemmiljön uppfylls.

Sjunde och sista riktlinjen är kommunikationen av forskningen. Design science forskning måste presenteras effektivt både till såväl teknikorienterad som ledningsorienterad publik. Studien presenteras i form av denna rapport, rapporten är skriven med tekniska termer och förklaring till dessa. Vilket gör att såväl teknikorienterad som ledningsorienterad publik kan ta åt sig kunskapen i studien.

Samtliga sju riktlinjer föreslagna av Hevner et al. (2004) kommer att följas i studien för tillämpning av design science.

3.3 Experiment

Experimentet som utförts i studien är en jämförelse mellan CNN-arkitekturerna ResNet och EfficientNet. För att svara på forskningsfrågan och uppfylla syftet med studien. Forskningsfrågan är vilken av EfficientNet- och ResNetarkitekturerna ger det högsta resultatet på produktmatchning med utvärderingsmåten f1-score, precision och recall? Experimentet ska därför jämföra en ResNetmodell med en EfficientNetmodell för att utvärdera vilken modell som presterar bäst på att matcha produkter med hjälp av bilder.

Följande tillvägagångssätt tillämpades för utvärderingen av arkitekturerna.

1. Skapa två modeller enligt genomförandekapitel nedan, en ResNetmodell och en EfficientNetmodell
2. Skicka hela datamängdens bilder genom varje nätverk separat, alltså tillåt nätverken att predicera bilderna och skapa en numerisk representation av varje bild i form av en featurevektor.
3. För att utvärdera hur nätverken har presterat på att extrahera bildernas egenskaper, skicka igenom hela featurevektorn i en KNN-modell som tränar på hela den nya datamängden och sedan låt KNN-modellen ta fram hela datamängdens potentiella matchningar. För att se KNN-modellens konstruktion se genomförandekapitlet nedan.
4. Beräkna f1-score, recall och precision för varje instans av datamängden med de sanna matchningarna i datamängden och de predicerade matchningarna i datamängden

KNN-modellens klassificering har bland annat tre parametrar som kan påverka resultatet, threshold, K och metric. Threshold är det värde som bestämmer när instansens grannar ska klassificeras som instansens klass eller inte beroende på dess beräknade avstånd till den aktuella instansen. K står för antalet närmaste grannar/insatser som avståndet ska beräknas för och instanser att ta hänsyn till vid matchningen av bilderna. Metric står för avståndsberäkningsfunktion alltså sättet KNN-modellen beräknar avståndet mellan insatserna baserat på deras numeriska representation som nätverken har resulterat i.

För att säkerställa att inga parameterinställningar ska påverka resultatet av nätverkets prestanda är KNN-modellens körning automatiserad att genomföra potentiella matchningar med 3 olika avståndsfunktions euclidean distance, manhattan distance och cosine similarity samt med en uppsättning av threshold från 1 till 10 med steg på 0.1. Körningarna utvärderades med hjälp av f1-score och för varje arkitektur kommer den inställning på threshold och avståndsberäkning väljas som gav den högsta f1-scoren. K är satt till 50 då det är ett högt tal för datamängden karaktär då ingen av klasserna ha över 50 instanser vilket säkerställer att K inställningen inte hämmar prediktionerna av potentiella matcher.

3.4 Datainsamling

Data som används i studien är insamlad av företaget Shopee och distribuerad via Kaggle (2021). Kaggle är online-gemenskap för maskininlärningsutövare och data scientist. Kaggle tillåter användare att dela datamängder och utveckla modeller i en online-baserad miljö. Shopee är ett företag som gärna vill lösa problemet med produktmatchning via bilder för att kunna ge sina kunder den bästa prismatchningen. De har därför arrangerat en tävling på Kaggle och släppt denna datamängd för att uppmuntra maskininlärningsentusiaster att lösa problemet.

Datamängden består av en mapp med bilder samt en csv fil med följande attribut `posting_id`, `image`, `image_phash`, `title` och `label_group`. De har gett ut en träningsdatamängd och en testdatamängd med 34 250 bilder respektive 3 bilder. Testmängden har dock inte någon klasstillhörighet i form av `label_group`. Vi vet alltså inte hur de 3 testbilderna ska matchas korrekt och därför bygger studien endast på träningsdatamängden. För mer ingående beskrivning av datans karaktär se genomförandekapitlets avsnitt om dataförståelse.

3.5 Utvärderingsstrategi

Utvärderingen av arkitekternas prestation på datamängden görs genom en KNN-modell. Den numeriska representationen som nätverken resulterar i är ett gäng kolumner med en mängd värden. EfficientNetmodellen ger 1536 attribut/kolumner och ResNetmodellen ger 2048 attribut/kolumner. För att utvärdera vilken arkitektur som har lyckats att ge den bästa numeriska representationen av bilderna i datamängden, körs EfficientNets featurevektor på 1536 attribut och ResNets featurevektor med 2048 attribut och tränas sedan av en unsupervised KNN-modell. KNN-modellen har sedan hela datamängdens bildrymd i en numerisk representation. När man sedan låter hela datamängden gå igenom KNN-modellen en gång till men i form av en avståndsmätning till de andra instanserna i datamängden resulterar detta i att varje instans blir tilldelad sina potentiella matchningar, därefter beräknas f1-score, recall och precision för den instansen matchningar och label group.

3.5.1 Utvärderingsmått

De utvärderingsmått som tillämpas är f1-score, recall och precision. Instanserna i datamängden har blivit tilldelade potentiella matcher samt faktiska matchningar. Därifrån kan man beräkna precision, recall och f1-score. Precision beräkningen ger en indikation på hur säkra vi kan vara på att de potentiella matchningarna är rätt, medan recall ger en indikation på hur bra man har lyckats hitta alla korrekta matchningar för bilden. F1-score ger det harmoniska medelvärdet för recall och precision, som ger en indikation på hur bra man har lyckats hitta potentiella matcher samt hur bra man har varit på att hitta samtliga matchningar som finns. Arkitekturerna kommer därför att utvärderas och jämföras på hela datamängdens medelvärde på f1-score, recall och precision.

Beräkningarna utförs på varje instans i datamängden genom att använda instansens sanna matchningar, potentiella matchningar och korrekta matchningar. Där de sanna matchningarna är de matchningar som finns i datamängden, potentiella matchningarna är de matchningar modellen har tagit fram för bilden och de korrekta matchningarna är de potentiella matchningarna som modellen har gjort som faktiskt stämmer med datamängdens matchningar.

Beräkningen för recall utförs på varje instans genom att ta antalet korrekta matchningar dividerat på antalet sanna matchningar. Enligt exemplet nedan är recall beräkningen $2/5$, vilket ger instansen ett recall på 0,4 som ger en indikation på hur bra modellen har lyckats hitta samtliga matchningar till bilden. I detta exempel har modellen missat 3 av 5 matchningar vilket gör att recall inte blir speciellt högt. Beräkningen av precision utförs på varje instans genom att ta antalet korrekta matchningar dividerat med antalet totala potential matchningar. Beräkningen av precision blir följande enligt exemplet nedan $2/2$ vilket ger instansen en precision på 1 som beskriver att modellens potentiella matchningar faktiskt är korrekta matchningar. F1-score beräknas på varje instans genom $2*((\text{korrekta matchningar}/ (\text{sanna matchningar} + \text{potentiella matchningar}))$. F1-score beräknas på exemplet nedan $2 * (2/(5+2))$ som ger resultat 0,571.

Instans	Sanna matchningar	Potentiella matchningar	Korrekta matchningar	Recall	Precision	F1
train_4271849129	train_1383791735 train_4271849129 train_1816141615 train_2419947469 train_2259188081	train_4271849129 train_2419947469	train_4271849129 train_2419947469	0,4	1	0,571

3.6 Metodreflektion

En styrka för metodens giltighet som experimentell forskning ger är möjligheten att isolera, kontrollera och utvärdera specifika variabler och orsaker, för att sedan jämföra vilka konsekvenser dessa har på slutresultatet (Recker 2013). Undersökningen kommer utnyttja detta genom att använda identisk datamängd för båda arkitekturerna. Det enda som är olika mellan modellerna kommer att vara arkitekturerna vilket gör att experimentet blir isolerat till den faktorn.

En svaghet hos experimentell forskning där man isolerar och kontrollerar specifika variabler är att de inte längre reflekterar verkligheten (Recker 2013). Vilket också kan hänvisas till den externa giltigheten där man uppskattar till vilken grad studiens resultat kan generaliseras

(Recker 2013). Denna svaghet som experimentell forskning besitter kan göra att det finns ett hinder för studiens resultat att vara generell för hela produktmatchningsproblematiken och därmed skada studiens externa giltighet. Då vi endast har isolerat experimentet till en datamängd vilket inte riktigt reflekterar hela verkligheten då det finns olika typer av datamängder inom produktmatchningsproblemet. Vi kan därför inte säga med säkerhet att vårt metod har samma resultat på andra datamängder. Den externa giltigheten är därför tvetydig i undersökningen.

Undersökningens pålitlighet beskriver i vilken utsträckning en variabel eller en uppsättning variabler är konsekventa i vad den är avsedd att mäta. Pålitlighet innebär att operationerna i en studie kan upprepas med lika inställningar med samma resultat. Problem med pålitlighet uppkommer oftast på grund av beroende av subjektiva observationer och datainsamling (Recker 2013). Datamängden är insamlad externt och undersökningen har därför inget beroende av subjektiva observationer eller datainsamling. Experimentet beskrivs utförligt i varje steg vilket gör det enkelt att replikera undersökningen. Studiens pålitlighet anses därför vara hög.

Design science är ett forskningsområde där det finns många debatter om hur man bäst går till väga. Det finns mer forskning som föreslår andra objektiva än Hevner et al. (2004), såsom Jones och Gregor (2007) åtta design teorier. Vi har valt att använda Hevner et al. (2004) sju riktlinjer. Recker (2013) diskuterar om att det finns en debatt om dessa 7 riktlinjer är lämpliga, eller om de är missledande. Det finns åsikter som att dessa riktlinjer kan göra design science generiska och skapar en för fast struktur, då de oftast tolkas alltför mekaniskt och procedurellt. Recker (2013) insisterar på att artefakten ska vara i fokus och inte stegen i riktlinjerna. Det viktigaste är att artefakten ska åstadkomma visad nytta. För att uppnå visad nytta bör tre kriterier tas hänsyn till:

- Ett, nyhet i den påvisade nyttan med en artefakt.
- Två, en positiv skillnad i nyttan av en artefakt jämfört med befintlig.
- Tre, en grundlig utvärdering som ger avgörande bevis för överlägsen nytta av en artefakt.

Undersökningen följer Hevner et al. (2004) sju riktlinjer för design science men för att säkerställa att inte fokuset försvinner från artefakten och dess visad nytta kommer även dess tre kriterier vara grunden för vår design science metod och utvärderingen av artefaktens visad nytta.

4 Genomförande

Utvecklingen av artefakten kommer att följa CRISP-DM processen. Nedan följer rubriker för varje steg i CRISP-DM och förklaring till studiens utveckling av artefakterna, en modell med EfficientNetarkitektur och en med ResNetarkitektur.

4.1 Affärsförståelse

Artefakterna ska utvecklas för att tackla ett produktmatchningsproblem. Problemet består av att identifiera samma produkt på olika bilder. Vilket är viktigt vid prisjämförelse där konsumenterna har blivit alltmer smarta och letar efter det bästa priset. För att underlätta för smarta konsumenter finns prisjämförelsesidor som ger konsumenterna ett verktyg för att göra smarta inköp till det bästa priset. För att prisjämförelsen ska ge konsumenterna den bästa beslutsgrunden kan inte produkter kategoriseras fel och påverka det bästa priset för konsumenten. Innan maskininlärning var ett alternativ har prisjämförelsesidor hanterat data manuellt vilket är otroligt tidskrävande och dyrt. Problemet med automatisk datahantering är att det blir mer felaktigheter. Därför är det viktigt att modellerna matchar produkterna korrekt.

4.2 Dataförståelse

Data som kommer att användas för att utveckla dessa modeller är Shopees datamängd hämtat från kaggle. Datamängden består av 32 412 bilder och 34 250 datainstanser, detta för att vissa datainstanser hänvisar till samma bild i datamängden. Datamängdens attribut är `posting_id` som är en unik identifierare, `image` som är bildfilens namn, `image_phash` vilket är bildens fingeravtryck, titeln på bilden och `label_group` som är målvariabeln där varje unik produkt har en unik `label_group`. Alla attribut i datamängden representeras som en sträng. Dessa rader finner man i en CSV-fil och till filen finns också en mapp med bilder som identifieras genom `image` attributet.

Det finns också en testmängd på tre bilder men dessa har ingen `label_group` vilket gör att vi inte har tillgång till information om vilken `label_group` dessa faktiskt tillhör. Därför kommer inte testmängden ingå i vår studie. Nedan visas tre exempel från träningsmängden samt deras bilder.

posting_id	image	image_phash	title	label_group
train_2771755203	001e11145b8e9bf5ac51110c0fdd8697.jpg	eab5c295966ac368	PASHMINA KUSUT RAWIS POLOS...	509010932
train_3117535609	00416d439a613fb6cbede5cfc95176e6.jpg	bb3fcc4c013c3e1	WINGs BRA-BRA TEMPEL SEAMLESS ..	532279668
train_615566263	007fca8ce9a042f9e1656ce8f96ba19d.jpg	e29b9d52d8a649ac	TUNIK DZUVIA ...	1356633425



001e11145b8e9bf5ac51110c0fdd8697.jpg
(train_2771755203)



00416d439a613fb6cbede5cfc95176e6.jpg
(train_3117535609)



007fca8ce9a042f9e1656ce8f96ba19d.jpg
(train_615566263)

Produktmatchning är ett komplext problem som går att lösa på flera sätt. Vi har valt att enbart fokusera på datamängdens bilder för att utföra vår produktmatchning.

4.3 Förbehandling av data

Den förbehandling av data som utförts på data är en uppdelning till tränings- och valideringsdata. 80% av instanserna blir träningsdata och 20% valideringsdata. Alla bilders bildstorlek både i tränings- och valideringsmängden omvandlas till 512x512 vilket kan göra att CNN-nätverken presterar bättre då större bilder är att föredra enligt Wang et al. (2020). Träningsmängdens bilder genomgår sedan en rad olika data augmentations operationer. Vändning av bilderna vertikalt med $p=0.5$ och horisontellt med $p=0.5$. Sedan utförs en rotation med inställningen $limit=120$ och $p=0.8$. Till sist görs även en ljuskorrigerings operation med parametrarna $limit=(0.009,0.6)$ $p=0.5$. Data augmentationer utförs för att generera mer träningsdata för modellerna att träna på och kan göra att modellen inte blir överfittad enligt Shorten och Khoshgoftaar (2019).

Både tränings- och valideringsdatamängden normaliseras som också tidigare studier har visat kan ge ett bättre resultat än icke normaliserad data (Munien & Viriri 2021). Till slut delas hela datamängden upp i olika delmängder s.k. batchar där varje batch innehåller 16 bilder. Anledningen till att dela upp all data i delmängder är för att låta modellen uppdatera sina inre parametrar efter varje delmängd i stället för efter hela mängden och på så sätt effektivisera hela processen (Kandel & Castelli 2020).

4.4 Modellering

För att bygga modellerna för problemet har vi använt oss av transfer learning. Dels för att flertalet tidigare studier har visat att det är en framgångsrik teknik (Kaur & Gandhi 2020; Yang et al. 2019; Liu, Tian & Xu 2019; Madakannu & Selvaraj 2019) men också för att det kräver för mycket datakraft att träna dessa nätverk själva. Därför hämtade vi ResNet och EfficientNet från Pytorch, där båda modellernas vikter var förtränade med hjälp av ImageNet. Average pooling användes för båda modellerna eftersom både Chang et al. (2019) och Akhtar och Ragavendrants (2019) tidigare studier har visat att det kan bidra till en ökad effektivisering, minskad risk för överfitting och öka igenkänningsnoggrannheten.

Såsom Chang et al. (2019) tar upp så är det svårt att veta hur mycket man skall modifiera de förtränade nätverken för att få ett bra resultat på ett specifikt problem. Vi har därför experimenterat fram en lösning. Olika konstellationer testades och resulterade i att ett linjärt lager ovanpå det förtränade lagret i de olika modellerna var lämpligast för studien. Därför använde vi ett linjärt lager som outputlager vilket innehöll 1536 noder för EfficientNet och 2048 noder för ResNet.

Träningen av modellerna sker genom 5 epoker och 1712 steg för träningsmängden och 429 steg för valideringsmängden. Träningen sker med en softmax som loss funktion. Modellerna tar in träningsmängden och validerar nätverkets resultat mot valideringsmängden och använder loss funktionen för att korrigera vikterna vid nästa epok, tills träningen är klar. Träningarnas output syns nedan där man tydligt kan se problematiken med överfitting då första epoken skapar den bästa modellen för båda arkitekturerna på valideringsdatan.



Figur 15: Träning av EfficientNetmodellen



Figur 16: Träning av ResNetmodellen

```

Building Model Backbone for efficientnet_b3 model

Downloading: "https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-weights/efficientnet_b3_ra2-cf984f9c.pth" to /root/.cache/torch/hub/checkpoints/efficientnet_b3_ra2-cf984f9c.pth
100%|██████████| 1712/1712 [16:22<00:00, 1.74it/s, Epoch=0, LR=1e-5, Train_Loss=9.26]
100%|██████████| 429/429 [01:57<00:00, 3.66it/s, Eval_Loss=9.82]

best model found for epoch 0

100%|██████████| 1712/1712 [16:04<00:00, 1.77it/s, Epoch=1, LR=7e-5, Train_Loss=7.77]
100%|██████████| 429/429 [01:53<00:00, 3.78it/s, Eval_Loss=13.8]
100%|██████████| 1712/1712 [16:06<00:00, 1.77it/s, Epoch=2, LR=0.00013, Train_Loss=4.21]
100%|██████████| 429/429 [01:52<00:00, 3.83it/s, Eval_Loss=16.6]
100%|██████████| 1712/1712 [16:02<00:00, 1.78it/s, Epoch=3, LR=0.000128, Train_Loss=1.19]
100%|██████████| 429/429 [01:52<00:00, 3.83it/s, Eval_Loss=18.2]
100%|██████████| 1712/1712 [16:05<00:00, 1.77it/s, Epoch=4, LR=8.24e-5, Train_Loss=0.289]
100%|██████████| 429/429 [01:55<00:00, 3.70it/s, Eval_Loss=18.9]

```

Figur 17: Träning av EfficientNetmodellen

```
Building Model Backbone for resnext50_32x4d model

Downloading: "https://github.com/rwightman/pytorch-image-models/releases/download/v0.1-weights/resnext50_32x4d_ra-d733960d.pth" to /root/.cache/torch/hub/checkpoints/resnext50_32x4d_ra-d733960d.pth
100%|██████████| 1712/1712 [18:04<00:00, 1.58it/s, Epoch=0, LR=1e-5, Train_Loss=9.27]
100%|██████████| 429/429 [02:06<00:00, 3.40it/s, Eval_Loss=10.1]

best model found for epoch 0

100%|██████████| 1712/1712 [18:03<00:00, 1.58it/s, Epoch=1, LR=7e-5, Train_Loss=6.12]
100%|██████████| 429/429 [01:54<00:00, 3.76it/s, Eval_Loss=17.7]
100%|██████████| 1712/1712 [18:03<00:00, 1.58it/s, Epoch=2, LR=0.00013, Train_Loss=1.24]
100%|██████████| 429/429 [01:55<00:00, 3.73it/s, Eval_Loss=21]
100%|██████████| 1712/1712 [18:02<00:00, 1.58it/s, Epoch=3, LR=0.000128, Train_Loss=0.299]
100%|██████████| 429/429 [01:54<00:00, 3.74it/s, Eval_Loss=22.2]
100%|██████████| 1712/1712 [18:03<00:00, 1.58it/s, Epoch=4, LR=8.24e-5, Train_Loss=0.126]
100%|██████████| 429/429 [01:55<00:00, 3.71it/s, Eval_Loss=21.9]
```

Figur 18: Träning ResNetmodellen

För att kunna ta fram vilka produkter som matchar varandra använde vi en unsupervised KNN-modell från cuML. KNN-Modellen tränas på CNN-modellernas output och sedan körs hela datamängden för att hitta potentiella matchningar av produkter. För att säkerställa att inga parametrar påverkar experimentets resultat automatiseras KNN-Modellens threshold och avståndsfunktion för att väljas optimalt för modellen utifrån f1-score. K för KNN-modellen är satt till 50 vilket gör att när hela datamängden körs ta de 50 närmsta matchningarna fram och genom thresholdvärdet väljs vilka av de 50 närmsta grannarna som kommer att vara potentiella matchningar för instansen.

4.5 Hårdvara och andra förutsättningar

Ett av studiens största hinder har varit datakraft och minne. Flera olika alternativa miljöer undersöktes, där colab med lokal GPU och minne för körningarna var en miljö som testades. Körningar med hög upplösning för bilderna blev väldigt tunga och tog mycket tid. Men Kaggles egna notebook-miljö erbjöd en lösning. Majoriteten av utvecklingen har skett via kaggles notebook-miljö som erbjuder CPU kraft på 4 CPU kärnor och 16 GB RAM, GPU kraft på 2 CPU kärnor 12 GB RAM samt TPU kraft på 4 CPU kärnor och 16 GB RAM. Kaggle erbjuder 36 timmar varje vecka med dessa resurser och allt som krävs är ett kagglekonto.

All kod i studien och utvecklingen av modellerna är skriven i python. Python är ett vedertaget språk för användning vid maskininlärning och språket har många maskininlärningsbibliotek vilket gör att det är det bra verktyg vid maskininlärning. Bibliotek som används i studien är följande: pandas användes för dataundersökning och pandas dataframe användes som datastruktur. Numpy-biblioteket användes för att utföra snabba och effektiva beräkningar på numpy arrayer. Biblioteket albumentations användes vid data augmentationen där metoderna Resize, HorizontalFlip, VerticalFlip, Rotate, RandomBrightness användes för att generera nya träningsbilder. Timm och Torch är de huvudsakliga maskininlärningsbiblioteken som innehåller de förtränade nätverken. För att modifiera de förtränade nätverken används

torch.nn där det linjära lagret hämtades som användes vid utveckling av modellerna. För KNN-modellen användes biblioteket `cuml.neighbors`.

4.6 Evaluering och resultat

I detta avsnitt presenteras de resultat som denna studie uppnått med hjälp av de modeller och experiment som tidigare beskrivits. Alla experiment är utförda i Kaggles Notebook. Körningen använde ett spann av värden på threshold på tre olika avståndsberäkningar, euclidean distance, manhattan distance och cosine similarity.

Körningen med EfficientNet gav följande resultat. EfficientNet presterade bäst med euclidean distance med ett threshold på 5.4 som gav ett f1-score på 0.623, recall på 0.555 och precision på 0.920. Med manhattan distance som avståndsberäkning uppnåddes ett f1-score på 0.492, recall på 0.353 och precision på 0.999 med ett threshold på 9.9. Det avståndsmått som gav EfficientNet det lägsta f1-score var cosine similarity med 0.122 i f1-score, 0.728 i recall och 0.075 i precision med 2.0 som threshold.

Metric EfficientNet	Threshold	F1-score	Recall	Precision
Euclidean distance	5.4	0.623	0.555	0.920
Manhattan distance	9.9	0.494	0.353	0.999
Cosine similarity	2.0	0.122	0.728	0.075

Körningen med ResNet visade att även här var euclidean distance den avståndsberäkning som gav bäst resultat dock med ett annan threshold på 3.7, vilket resulterade i ett f1-score på 0.592, recall på 0.487 samt precision på 0.958. Med manhattan distance som avståndsberäkning uppnås ett f1-score på 0.494, recall på 0.352 och precision på 0.999 med ett threshold på 9.9. Likt EfficientNet gav även cosine similarity det lägsta f1-score 0.122 med 2.0 som threshold samt 0.728 i recall och 0.075 i precision för ResNetmodellen.

Metric ResNet	Threshold	F1-score	Recall	Precision
Euclidean distance	3.7	0.592	0.487	0.958
Manhattan distance	9.9	0.494	0.352	0.999
Cosine similarity	2.0	0.122	0.728	0.075

För EfficientNet var euclidean distance det bästa avståndsberäkningen med ett threshold på 5.4 och ResNet presterade också bäst med euclidean distance men med ett threshold på 3.7. Nedan visas de körningar med det bästa resultatet från respektive modell.

Modell	F1-score	Recall	Precision
EfficientNet	0.623	0.555	0.920
ResNet	0.59	0.487	0.958

Resultatet ovan illustrerar att EfficientNet arkitekturen uppnår ett f1-score på 0.623, ett precision på 0.920 och en recall på 0.555. ResNet däremot uppnår ett f1-score på 0.59, ett precision på 0.958 och en recall på 0.487. EfficientNet har den högsta f1-score av de två arkitekturerna. Dock har ResNetmodellen en högre precision än EfficientNetmodellen men EfficientNet har ett högre recall.

5 Analys

De tre avståndsmåtten som användes under experimenten gav olika resultat. Avståndsmåtten manhattan distance och cosine similarity gav näst intill identiska resultat för både ResNet och EfficientNet. Där manhattan distance gav ett högre resultat på precision men ett lägre resultat på recall, vilket tyder på att modellerna gjort bra gissningar men missade många bilder med samma label_group. Cosine similarity gav ett högre resultat på recall men ett lägre resultat på precision, vilket visar att den hittat många korrekta matchningar men den har också gjort många matchningar som är fel. Cosine similarity gjorde många felmatchningar för varje produkt vilket resulterade i ett lågt f1-score jämfört med manhattan distance och euclidean distance. Manhattan distance gjorde många säkra gissningar då många av dess matchningar var korrekta men den missade också många bilder med samma produkt. Detta resulterade i ett något lägre f1-score jämfört med euclidean distance. Även om manhattan distance uppnådde en precision på 0.99 är detta resultat missvisande då manhattan distance i många fall enbart lyckats hitta instansens egen bild som matchning vilket ger ett högt värde på precision. Euclidean distance gav ett mer jämnt resultat på precision och recall, den lyckades hitta många av de matchningarna som fanns samtidigt som många av de matchningar som togs fram var korrekta matchningar. Det var också vid euclidean distance som skillnader mellan ResNet och EfficientNet påvisades.

Euclidean distance gav det bästa resultatet och är det avståndsmått som används för att ta fram vårt slutresultat. Slutresultatet av studien visar att EfficientNet är den över lag bästa arkitekturen för produktmatchning på datamängden. Resultatet visar också att ResNet var bättre än EfficientNet på att föreslå rätt matchningar av bilderna. De matchningarna ResNet gör stämmer mer än de matchningar EfficientNet föreslår, eftersom ResNet fick ett högre precision än vad EfficientNet fick. EfficientNet uppnår dock ett bättre recall som visar att EfficientNet är bättre än ResNet på att hitta fler eller alla korrekta matchningar bland sina potentiella matchningar. Men skillnaden i recall är större mellan modellerna vilket gör att EfficientNet får ett högre f1-score och är över lag bättre än ResNet, men vad som är viktigast kan diskuteras. Är det viktigt att de föreslagna matchningarna är korrekta eller att man hittar alla korrekta matchningar. Är det viktigaste att de föreslagna matchningarna är korrekta har ResNet ett övertag men är det viktigare att hitta alla korrekta matchningar har EfficientNet ett övertag. Resultatet beror därför på vad som anses vara viktigast för att avgöra vilken av arkitekturerna som ger bäst resultat.

Resultatet kan vara något missvisande då datamängden körs genom en unsupervised KNN med hela datamängden som feature space. Vilket gör att den även har möjligheten att hitta sig själv och har med den instansen i alla 3 utvärderingsmått. Detta kan vara något missvisande men det blir också konstigt om man inte får ta hänsyn till detta då det finns exempel där en label_group enbart innehåller en instans. En annan anledning till att vi valde att inkludera dessa matchningar i våra resultat var för att det även fanns identiska bilder inom datamängden och om två återförsäljare använder sig av två identiska bilder anser vi att det är viktigt att kunna matcha även dessa.

Vid utforskning av modellernas potentiella matchningar för instanser ser vi att EfficientNet och ResNet är duktiga på olika bilder. Ett exempel då EfficientNet gav ett bättre resultat var när den skulle matcha bild 1 och bild 2 där enda skillnaden är den svarta texten ner till höger i bild 2. I datamängden som vi använde fanns det ingen annan bild som skulle matchas och EfficientNetmodellen lyckades både identifiera och matcha ihop dessa bilder korrekt. ResNet lyckades också matcha dessa bilder men valde också att klassificera bild 3 och 4 till samma

produkt. Vilket motsäger resultaten ovan men illustrerar att de inte är generaliserbart till alla potentiella matchningar. En anledning till att ResNet även klassificerat dessa bilder till samma klass kan vara att objekten har liknande form.



Figur 19 Bild 1



Figur 20 Bild 2



Figur 21 Bild 3



Figur 22 Bild 4

Ett exempel då ResNet enbart matchade en bild till identiska bilder var bild 5. Bild 6 innehåller samma typ av produkt vilket betyder att dessa två bilder skulle matchat, men då ResNet enbart identifierade identiska bilder fick den en recall på 0.5 samtidigt som precision hamnade på 1 då alla matchningar korrekta. Däremot matchade EfficientNet ihop fler bilder än vad som stämmer i verkligheten. Den fick ett precision på 0.17 vilket tyder på att många av de matchningar den gjort är fel och ett recall på 0.5 vilket betyder att den enbart hittat hälften av de matchningar som den skulle hittat (i detta fall 1 matchning). Bild 7 och bild 8 är två exempel på matchningar EfficientNet gjorde fel. Ser man på bilderna kan man dra slutsatsen att EfficientNet ändå matchar liknande produkter.

Även här kan vi se att bild 5 och bild 6 visar samma produkt på två olika sätt. Ingen av modellerna lyckades matcha bild 6, vilket kan bero på att produkten inte är riktigt lika mycket i fokus som på bild 5.



Figur 23 Bild 5



Figur 24 Bild 6



Figur 25 Bild 7



Figur 26 Bild 8

När de båda modellerna skulle matcha bilderna som innehöll den produkt som finns på bild 9 så har ResNet även här bara matchat de bilder som är identiska med bild 9. Detta ger ett precision på 1 men då de missat att matcha bilder såsom bild 10 och 11 så fick ResNet ett recall på 0,625. EfficientNet lyckades identifiera alla bilder som innehöll den produkten vilket gav en recall på 1, däremot så matchade den ihop bild 12, vilket är av en annan produkt. Detta resulterade i att den fick ett precision på 0.89. Detta exempel illustrerar att det generella resultatet stämmer. Men även här ser man på bilderna att EfficientNet gör ett bra jobb med att matcha produkterna även om den matchade med bild 12 som inte var korrekt. Eftersom vi anser att även en människa hade behövt mer information än bara bilder för att sära på dessa produkter.



Figur 27 Bild 9

Figur 28 Bild 10



Figur 29 Bild 11



Figur 30 Bild 12

6 Diskussion och slutsatser

Syftet med studien var att undersöka vilken CNN-arkitektur av EfficientNet och ResNet som var mest lämplig att använda vid produktmatchning. Svaret på vår forskningsfråga är att EfficientNet var den över lag lämpligaste arkitekturen baserat på f1-score, recall och precision.

Studiens resultat stärker tidigare forskning såsom Tan och Le (2019) som visar att EfficientNet är en bättre arkitektur än till exempel ResNet. Även vid klassificering av växtsjukdomar (Atila et al. 2021), frukt (Duong et al. 2020), fundussjukdomar (Wang et al. 2020) och Covid diagnosering (Marques, Agarwal & de la Torre Díez 2020) är EfficientNet den arkitektur som anses vara mest lämplig. Slutsatsen av vårt resultat visar att EfficientNet även är den lämpligaste tekniken utav EfficientNet och ResNet inom området för produktmatchning. Dock är studien inte speciellt generaliserbar, eftersom studien är endast isolerad till en datamängd med produkter som inte speglar hela verkligheten då det finns betydligt mer produkt och bilder än de som är med i denna studie. Den externa giltigheten är därför inte speciellt stark och gör att resultatet i studien inte är lämpligt för att generaliseras på hela produktmatchningsområdet.

Även fast analysen visar att EfficientNet-arkitekturen gav ett bättre resultat än ResNet-arkitekturen så uppnår inte modellen i vår studie speciellt högt f1-score, vilket gör att vår modell inte är brukbar i verkligheten. Vi tror att med fortsatt forskning kan ett betydligt bättre resultat uppnås. Förslag på fortsatt forskning är att undersöka EfficientNet på liknande datamängd. Jämföra EfficientNet med andra CNN-arkitekturer. Lägga till mer lager till EfficientNet-arkitekturen såsom normalisering och/eller dropout och även tillämpning av ArcFace som loss funktion. Ett annat tillvägagångssätt som vi tror hade kunnat öka modellens resultat är att även kolla på faktorer såsom titel och beskrivning tillsammans med bilden för att ta fram potentiella matchningar. Vi anser att EfficientNet lyckats identifiera bra matchningar till bilderna även om inte datamängdens label group var densamma. Vilket illustrerar att EfficientNet lyckats identifiera bildernas egenskaper bra, och med hjälp av texttolkning av titel eller beskrivning kunna ge EfficientNet ett bättre resultat.

Källor

Akhtar, N., & Ragavendran, U. (2019). Interpretation of intelligence in CNN-pooling processes: a methodological survey. *Neural Computing & Applications*, 32(3), 879–898. doi: 10.1007/s00521-019-04296-5

Atila, Ü., Uçar, M., Akyol, K., & Uçar, E. (2021). Plant leaf disease classification using EfficientNet deep learning model. *Ecological Informatics*, 61, ss. 101182–. doi: 10.1016/j.ecoinf.2020.101182

Cariou., C. & Chehdi, K.(2015). Unsupervised nearest neighbors clustering with application to hyperspectral images. *IEEE Journal of Selected Topics in Signal Processing*, 9(6), ss. 1105-1116. doi: 10.1109/JSTSP.2015.2413371

Chang, J., Zhang, L., Gu, N., Zhang, X., Ye, M., Yin, R., & Meng, Q. (2019). A mix-pooling CNN architecture with FCRF for brain tumor segmentation. *Journal of Visual Communication and Image Representation*, 58, 316–322. doi:10.1016/j.jvcir.2018.11.047

Chang, T., Li, H., Wen, G., Hu, Y., & Ma, J. (2019). Facial expression recognition sensing the complexity of testing samples. *Applied Intelligence (Dordrecht, Netherlands)*, 49(12), ss. 4319–4334. doi:10.1007/s10489-019-01491-8

Competera.net. (2021). Smart Product Matching for Price Comparison. <https://competera.net/solutions/by-need/product-matching> [2021-05-08].

Czajkowski, M., & Kretowski, M. (2019). Decision tree underfitting in mining of gene expression data. An evolutionary multi-test tree approach. *Expert Systems with Applications*, 137, 392–404. doi:10.1016/j.eswa.2019.07.019

Deng, J., Dong, W., Socher, R., Li L., Li K. & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. doi: 10.1109/CVPR.2009.5206848

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Computing Surveys*, 27(3), 326–327. doi: 10.1145/212094.212114

Duong, L., Nguyen, P., Di Sipio, C., & Di Ruscio, D. (2020). Automated fruit recognition using EfficientNet and MixNet. *Computers and Electronics in Agriculture*, 171, ss.105326–. doi: 10.1016/j.compag.2020.105326

Farabet, C., Couprie, C., Najman, L. & LeCun, Y. (2013). Learning Hierarchical Features for Scene Labeling. *IEEE transactions on pattern analysis and machine intelligence*. 35 (8), ss. 1915–1929. doi: 10.1109/TPAMI.2012.231

Gao, S., Duan, L. & Tsang, I. (2016). DEFEATnet-A Deep Conventional Image Representation for Image Classification. *IEEE transactions on circuits and systems for video technology*. 26 (3), ss. 494–505. doi: 10.1109/TCSVT.2015.2389413

GeeksforGeeks. (2019). *CNN / Introduction to Pooling Layer*

<https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/> [2021-05-29].

Géron, A. (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Sebastopol: O'Reilly Media, Incorporated.

Grm, K., Štruc, V., Artiges, A., Caron, M., & Ekenel, H. (2017). *Strengths and Weaknesses of Deep Learning Models for Face Recognition Against Image Degradations*. doi: 10.1049/iet-bmt.2017.0083

Guérin, J., Thiery, S., Nyiri, E., Gibaru, O., & Boots, B. (2021). Combining pretrained CNN feature extractors to enhance clustering of complex natural images. *Neurocomputing (Amsterdam)*, 423, ss.551–571. doi: 10.1016/j.neucom.2020.10.068

He, K., & Sun, J. (2014). *Convolutional Neural Networks at Constrained Time Cost*

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ss.770–778. doi: 10.1109/CVPR.2016.90

Hevner, A. R., March, T. S., Park, J. & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75–105. doi:10.2307/25148625

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V., & Chen, Z. (2018). Gpipe: Efficient training of giant neural networks using pipeline parallelism.

Image-net (2021). about image-net. <https://image-net.org/about.php> [2021-05-03]

Jones, D., & Gregor, S. (2007). The Anatomy of a Design Theory. *Journal of the Association for Information Systems*, 8(5), 312–335. doi: 10.17705/1jais.00129

Kaggle (2021). *Shopee – Price Match Guarantee*. <https://www.kaggle.com/c/shopee-product-matching/data> [2021-04-25]

Kandel, I. & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6(4), ss. 312-315. doi:10.1016/j.ict.2020.04.010

Kalogirou, S. A. (1999) Applications of artificial neural networks in energy systems: A review. *Energy conversion and management*. 40 (10), ss. 1073–1087.

Kaur, T., & Gandhi, T. (2020). Deep convolutional neural networks with transfer learning for automated brain image classification. *Machine Vision and Applications*, 31(3). doi: 10.1007/s00138-020-01069-2

Kelleher, J., MacNamee, B., & D'Arcy, A. (2015). *Fundamentals of machine learning for predictive data analytics : algorithms, worked examples, and case studies* . Cambridge, Mass: The MIT Press.

- Konopielko, Ł. & Radkova, D. (2017). Price dispersion and online markets maturity in Poland, *Acta Universitatis Lodzianensis.Folia Oeconomica*, 329, ss. 7-21. doi: 10.18778/0208-6018.329.01
- Liu, S., Tian, G., & Xu, Y. (2019). A novel scene classification model combining ResNet based transfer learning and data augmentation with a filter. *Neurocomputing (Amsterdam)*, 338, ss. 191–206. doi: 10.1016/j.neucom.2019.01.090
- Lu, Z., Bai, Y., Chen, Y., Su, C., Lu, S., Zhan, T., Hong, X., & Wang, S. (2020). The classification of gliomas based on a Pyramid dilated convolution resnet model. *Pattern Recognition Letters*, 133, ss. 173–179. doi: 10.1016/j.patrec.2020.03.007
- Lu, Z., Jiang, X., & Kot, A. (2018). Deep Coupled ResNet for Low-Resolution Face Recognition. *IEEE Signal Processing Letters*, 25(4), ss.526–530. doi:10.1109/LSP.2018.2810121
- Madakannu, A. & Selvaraj, A. (2019) DIGI-Net: a deep convolutional neural network for multi-format digit recognition. *Neural computing & applications*. 32 (15), ss. 11373–11383. doi: 10.1007/s00521-019-04632-9
- Marques, G., Agarwal, D., & de la Torre Díez, I. (2020). Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network. *Applied Soft Computing*, 96, ss. 106691–106691. doi: 10.1016/j.asoc.2020.106691
- Munien, C., & Viriri, S. (2021). Classification of Hematoxylin and Eosin-Stained Breast Cancer Histology Microscopy Images Using Transfer Learning with EfficientNets. *Computational Intelligence and Neuroscience*, 2021. doi:10.1155/2021/5580914
- Munter, A., Othman, R. R., Abualhaj, M., Anbar, M. & Yaakob, S. N. (2016). A preliminary performance evaluation of K-mean, KKN and EM unsupervised machine learning methods for network flow classification. *International Journal of Electrical and Computer Engineering*. 6(2), ss. 778-784. doi:10.11591/ijece.v6i1.8909
- Pan, S., J. & Yang., Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), ss.1345–1359. doi:10.1109/TKDE.2009.191
- Recker, J. (2013). *Scientific research in information systems : a beginner's guide*. 1st ed. 2013. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Mayer, R., Huh, J., & Cude, B. (2005). Cues of credibility and price performance of life insurance comparison Web sites. *The Journal of Consumer Affairs*, 39(1), 71–94. doi: 10.1111/j.1745-6606.2005.00004.x
- Sharifzadeh, F., Akbarizadeh, G., & Seifi Kavian, Y. (2019). Ship Classification in SAR Images Using a New Hybrid CNN–MLP Classifier. *Journal of the Indian Society of Remote Sensing*. 47 (4), ss. 551–562. doi: 10.1007/s12524-018-0891-y

- Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2), ss.227–244. doi: 10.1016/S0378-3758(00)00115-4
- Shorten, C. & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of big data*. 6 (1), ss. 1–48. doi: 10.1186/s40537-019-0197-0
- Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*.
- Tan, M. & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of the 36th International Conference on Machine Learning*, 97, ss. 6105–6114.
- Keras (2021). Keras documentation: Image data preprocessing. <https://keras.io/api/preprocessing/image/>. [2021-05-10]
- Vo, D. M. & Lee, S.-W. (2018). Semantic image segmentation using fully convolutional neural networks with multi-scale images and multi-scale dilated convolutions. *Multimedia tools and applications*. 77 (14), ss. 18689–18707. doi: 10.1007/s11042-018-5653-x
- Wang, J., Yang, L., Huo, Z., He, W., & Luo, J. (2020). Multi-Label Classification of Fundus Images With EfficientNet. *IEEE Access*, 8, ss. 212499–212508. doi:10.1109/ACCESS.2020.3040275
- Weiss, K., Khoshgoftaar, T., & Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(1), ss. 1–40. doi:10.1186/s40537-016-0043-6
- Wightman, R. (2021). timm: (Unofficial) PyTorch Image Models. PyPI.<https://pypi.org/project/timm/> [2021-05-14].
- Wu, X., Chen, H., Wu, X., Wu, S., & Huang, J. (2021). Burn Image Recognition of Medical Images Based on Deep Learning: From CNNs to Advanced Networks. *Neural Processing Letters*. doi: 10.1007/s11063-021-10459-0
- Xia, P., Zhang, L., & Li, F. (2015). Learning similarity with cosine similarity ensemble. *Information Sciences*, 307, ss. 39–52. doi: 10.1016/j.ins.2015.02.024
- Yang, Z., Yu, W., Liang, P., Guo, H., Xia, L., Zhang, F., Ma, Y. & Ma, J. (2019). Deep transfer learning for military object recognition under small training set condition. *Neural Computing & Applications*, 31(10), ss.6469–6478. doi: 10.1007/s00521-018-3468-3
- Zagoruyko, S. & Komodakis, N.(2016). Wide residual networks. BMVC.



HÖGSKOLAN I BORÅS

Besöksadress: Allégatan 1 · Postadress: 501 90 Borås · Tfn: 033-435 40 00 · E-post: registrator@hb.se · Webb: www.hb.se