Abstract

English abstract goes here

Sammanfattning

Svenskt sammanfattning

Contents

Chapter 1

Introduktion

Koppla första mening till titel. söka på speicfika saker i en annons. mysig inledning. Deeplearning bildanalys - google grejer. Ai och hur automatisering vuxit fram. Lite fakta på hur viktig sökfunktioner är för att hitta bostäder? Hur många sökningar /tidsenehet. Om folk enklare hittar bostad på ett mer effektivt sätt så effektiviseras hela köp och sälj processen och därmed hela marknaden.

1.1 Problemformulering

För att hitta nyckelord kopplade till en bild idag så behöver någon manuellt bestämma nyckelord till bilden. Uppgiften är tidskrävande och det är svårt att i efterhand producera relevanta nyckelord utan att gå igenom allt manuellt igen.

Det är också för bostadssökare tidskrävande att leta efter bostäder. Om man kan förfina sökningen än mer skulle detta effektivisera processen.

Syftet med den här studien är att titta på hur maskininlärningsmetoder kan användas för att hitta relevanta nyckelord till bilder på lägenhetsannonser.

1.2 Frågeställning

Går det med nuvarande verktyg inom maskininlärning hitta attribut i bilder från lägenhetsannonser?

1.3 Avgränsningar

Vilka avgränsningar vi gjort i datamängder, testpersoner samt modeller.

Chapter 2

Bakgrund

I det här kapitlet presenteras teori som är relevant för bildklassificering. Målet är att ge läsaren förståelse för de byggstenar som används för att konstruera en modern algoritm för bildklassificering. De vanligaste verktygen inom modern bildkategorisering bygger på djupinlärning, som är en del inom artificella neurala nätverk.

2.1 Maskininlärning

Maskininlärning är ett aktivt forskningsfält inom datalogi och en maskininlärningsalgoritm kan förklaras som en algoritm som lär sig att bli bättre med hjälp utav data [1]. Inom maskininlärning så brukar lärandeprocessen se ut på följande sätt: Om ett datorprogam har som uppgift att med hjälp av en viss erfarenhet E, lära sig vissa förutbestämda uppgifter T, vilket kan mätas med måttet P. Om programmets prestanda P blir bättre, det vill säga att programmet blir bättre på att lösa uppgifterna T, med hjälp av erfarenheten E, då lär sig programmet.

Maskininlärning brukar delas in i två överkategorier: Unsupervised learning och Supervised learning. Det som skiljer dessa åt är att inom supervised learning så är all data redan kategoriserad och uppmärkt för ändamålet medan inom unsupervised learning så är den inte det. Det kan ses som att vi i ena fallet redan har de rätta svaren på vår data. Algoritmer inom unsupervised learning handlar huvudsakligen om att kategorisera data i olika kluster eller på andra sätt försöka förstå den tillgänliga datan. Supervised learning handlar istället om att utgå ifrån den uppmärkta datan och lära sig utav den för att göra samma typ av kategorisering som datan redan är kategoriserad i. Denna funktionsapproximation kan sedan användas för klassificering av ny omärkt data.

Det finns även andra grenar tillämpningsområden inom supervised learning utöver klassificering.

2.2 Artificella Neurala Nätverk

Ett område med många tillämpningsområden inom maskininlärning är artificella neurala nätverk. Neurala nätverk var från början ett försök till att bygga en digital modell av hur det biologiska neuronsystem fungerar. Forskningen inom områden avvek sedan från att efterlikna den biologiska varianten så mycket som möjligt och fokuserade istället på att konstruera neurala nätverk som fungerade så bra som möjligt på maskininlärningsproblem. Grundbyggstenarna i neurala nätverk är dock fortfarande baserade på dess biologiska variant. Den enklaste beräkningsenheten i hjärnan är en neuron och dessa neuroner är ihopkopplade med synapser. En neuron får signal in och skickar sedan signaler ut via synapserna. Hur stark utsignal är simuleras i en dator med hjälp av vikter, (engelska: weights, W). Målet är att träna modellen och göra den bättre genom att justera vikterna. Om summan av flera av dessa viktade insignaler når en viss gräns, så skickar neuronerna vidare en signal. Detta simuleras i en dator med en aktiveringsfunktion. Neurala nätverk är dessa neuroner i en acyklisk graf.

[BILD PÅ NEURALT NÄTVERK]

Ett vanligt neuralt nätverk består vanligtvis först av ett indatalager (input layer). Indatalagret brukar representeras av en neuron per egenskap i indatan. Då indata är bilder så brukar en neuron i indatalagret motsvara en pixel i en bild. Dessa neuroner i indatalagret är sedan ihopkopplade med ett nytt lager med neuroner. Detta lager kallas för det gömda lagret (hidden layer). De gömda lagren kan bestå av godtyckligt många neuroner. Det går att ha en eller fler gömda lager efter varandra och efter det kommer utdatalagret (output layer). Utdatalagret består vanligtvis av lika många neuroner som modellen ska ge olika svar. Om modellen ska kategorisera indata i tio olika kategorier, så borde modellen då ha tio neuroner i sitt utdatalager.

Ett neuralt nätverk blir bättre på sin uppgift genom att ändra sina vikter, vilka även kallas för parametrar. Detta sker i två steg. Första steget är feed-forward pass. Feed-forward pass handlar om att skicka in sin träningsdata genom nätverket och få ut ett svar eller klassificering. Då träningsdatan redan är uppmärkt så jämfört svaret från det neurala nätverket med det riktiga svaret. Beroende på hur många fel nätverket gissade och hur osäker det var när det gissade fel, så beräknas en kostnad. Det finns olika sätt att beräkna denna kostnad men vanligtvis används cross-entropy loss. Målet med att träna modellen är

att få denna kostnad så låg som möjligt. Nästa steg är backward pass, vilket även kallas för backpropagation. Det vi vill göra är att ändra parametrarna så att kostnaden vi beräknade innan blir lägre. Detta görs genom att beräkna gradienten av kostnaden med avseende på alla parametrarna. Vi kan sedan ta ett steg åt motsatt håll som gradienten, för att minska kostnaden. Storleken på detta steg kallas för learning rate.

Stochastic gradient descent

Stochastic Gradient Descent heter den vanligaste algoritmen för att uppdatera parametrarna med hjälp av gradienten. Den bygger på samma tvåstegsmodell som beskriven ovan men istället för att beräkna gradienten för alla datapunkter i träningsmängden så väljs några stycken ut som man beräknar gradienten på. Denna delmängd brukar kallas för batch och dess storlek för batch size. En epoch är när modellen har gått igenom alla datapunker i träningsmängden en gång. Learning rate är då hur stort steg åt gradientens motsatta håll vi ändrar på parametrarna vid varje uppdatering.

[FORMEL PÅ SGD]

En förbättring till learning rate är the momentum method. Stochastic gradient descent med momentum kommer ihåg förändringen av parametrarna vid varje iteration och baserar nästa uppdatering på en linjärkombination av gradienten och den tidigare förändringen.

[FORMEL PÅ MOMENTUM]

2.3 Convolutional Neural Network

Convolutional neural networks (CNN) är en viss typ av neurala nätverk för att processera data som har indata som är placerat i ett rutnät. Det inkluderar data om tidsserier men även bilddata, som kan ses som ett rutnät av pixlar. CNN har fått stort genomslagskraft i praktiska tillämpningar. Convolution är en viss typ av linjär operation och CNN är då neurala nätverk där minst ett av dess lager är ett convolutional layer.

2.3.1 Lager

Här presenteras de lager som vanligtvis används i ett CNN. Detta för att sedan kunna gå in på hur de olika arktitekterna inom CNN är uppbyggda. Även om CNN kan användas till olika typer av data, så fokuserar vi här i texten på dess kontext med bilder. En vanlig ordningsföljd av lager för bildkategorisering

är input, convolutional layer, ReLU, pooling layer och sist ett fully-connected layer.

Input

Detta lager håller de råa pixelvärdena från bilden via skickar in. I ett vanligt neuralt nätverk med resnet-arkitekturen så är det 224 x 224 x 3. Då är det en rektangulär bild med 224 pixlar i höjd, 224 pixlar i bred och 3 färgkanaler, RGB.

Convolutional Layer

Ett convolutional layer består av en mängd filter som har parametrar som går att träna. Varje filter är kvadratiskt och små mått i höjd och bredd, men har alltid samma djup som vår indata. En vanlig storlek på ett filter är 5x5x3, det vill säga 5 pixlar brett och högt och 3 färgkanaler. Vid forward pass så glider vi (convolve) detta filter över vår indata-bild. Vi beräknar skalärprodukten av filtret och den 5x5x3-bit av indatan som filter ligger på. Efter beräkningen så glider vi filtret åt sidan. Vanligtvis en pixel, men det kan även vara flera. Utdata från varje filter blir en tvådimensionell activation map. Djupet på utdata motsvarar antal filter vi använt i vårt convolutional layer. Höjd och bredd på utdata beror på storleken på filter.

Storleken på utdata från ett convolutional layer beror på tre hyperparametrar: djup, stride och zero-padding. Djup motsvarar antal filter som används och blir djupet på vår utdata. Stride består hur många pixlar vi förflyttar oss vid varje glidning. Zero-padding bestämmer om vi ska bygga en ram runt bilden med nollor. Detta användas för att kunna behålla storleken på bilden igenom flera convolutional layers men är också bra för att inte bortse viktig information i utkanten av bilden.

Activation Layer / ReLU

Activation Layer består av en elementvis funktion. Det finns flera olika typer av dessa, men det som vanligtvis används inom bildkategorisering är Rectified Linear Units (ReLU). Den applicerar funktionen max(0,x) på varje element i indata. Detta betyder varje element som är positivt är opåverkat och varje negativt värde får värdet noll. Utdata har samma storlek som indata.

Pooling Layer

Ett pooling layer applicerar en nedsampling (eng: downsampling) på indata. Detta sker längs de spatiala dimensionerna, bredd och höjd. Detta sker vanligtvis för att minska antalet parametrar i nätverket och för att minska på beräkningskraften som behövs. Det är även en teknik för att undvika overfitting. Vanligaste typen är max pooling, då man tar ett område, till exempel 2x2 pixlar och väljer den pixel med högst värde. Vanligast är max pooling på 2x2 pixlars filter med en stride på 2. Indata minskar då med 75%.

Fully-connected Layer

Fully-connected layers är den typen av lager som vanligast i normala neurala nätverk. I detta lager har varje neuron en full koppling till alla aktiveringar från det tidigare lagret. Dessa har då vanligtvis en tillhörande matris med vikter och bias, vilket kan beräknas med matrismultiplikation. Storleken på utdata beror på storleken på viktmatrisen. Vanligtvis så byggs det sista FC layer upp så att det har samma storlek som vi vill ha kategorier. Om vi vill klassifiera en bild i tio kategorier, så kan utdata från vårt sista lager ha storleken 10x1, där varje element motsvarar sannolikheten för att bilden tillhör en viss kategori.

2.3.2 Tekniker inom djupinlärning

Batch normalization

Data augmentation

2.3.3 Arkitekturer

Här kommer ett urval av de vanligaste arkitekturerna för CNN för bildkategorisering att presenteras.

Resnet

Det finns flera olika varianter av ResNet, Resnet18, Resnet34, Resnet50, Resnet101 och Resnet152, beroende på storlek. Vi fokuserar här på Resnet18. ResNet bygger på en struktur med convolutional layers, pooling layers och fully-connected layers. Trenden har varit att CNN blir djupare med fler lager.

Men till slut kom modellerna till ett tak när det handlar om accuracy. ResNet är en lösning på det här problemet genom att introducera "identity shortcut connections". Detta betyder att det finns kopplingar som hoppar över vissa convolutional layers. Det betyder att om modellen inte behöver utnyttja

alla convolutional layers så kan den bara använda sig av identiteten istället för ett convolutional layer. Det betyder att nätverket borde kunna vara hur stort som helst, för det går alltid att bara använda indentiteten. Modellerna med residuala funktioner ska även vara enklare att optimera [2]. Resnet beskär bilderna till storleken 224 x 224 pixlar, och huvudsakliga målet var att kategorisera bland 1000 kategorier i imageNet 2012 classification dataset. Det finns flera olika versioner av ResNet, beroende på antal lager. ResNet-18 ser ut så här:

Alexnet

VGG

Densenet

Inception V3

2.4 Transfer Learning

2.4.1 ImageNet

2.4.2 Feature extraction

2.4.3 Finetuning

Chapter 3

Metod

Hur vi gått tillväga. Vilka dataset, hur implementation gått till (verktyg, klassificerare, parametrar), hur vi valt features. Hur evalueringen har gått till (traning, test, validation set). Vi vill mäta både precision och recall. Plotta en PR curve. Vi kan även plotta hur precisionen har gått om i förhållande till mer data, för att skapa en uppfattning av om modellen kan bli bättre med mer data. Vanligtvis gör man detta i log-skala [1]. Gör även att göra en grid search på hyperparametrar. Hur vi kan använda oss av preprocessing kan vi läsa i kapitel 12 av Goodfellow.

Chapter 4

Resultat

Prestandan av de olika modeller kommer presenteras här.

4.1 Balkonger

Här nedan presenteras resultatet av den binära klassificeringen av balkonger i mäklarbilder. Resultatet består av två grafer per modell, som visar hur kostnaden från kostnadsfunktionen samt noggrannheten för både vårt träningssset och evalueringsset. Det finns även en sammanställning av de högst uppnådda noggrannheten och hur lång tid modellerna tog att träna upp. Vi kommer även titta på både när alla lager var frysta och när alla lager tränades.

4.1.1 Feature extraction

Vi kan se kostnaden för båda tränings och valideringsdatan i figur 4.1 för varje epoch. Vi kan även se i figur 4.2 hur träffsäkerheten för de olika modeller var på balkonger.

Modell	Tid	Max. noggrannhet
Resnet	3m 37s	94.63
Alexnet	3m 18s	94.27
VGG-11	6m 21s	93.86
Densenet	5m 59s	96.94
Inception V3	9m 04s	95.80

Table 4.1: Sammanställning av feature extraction

Modell	Tid	Max. noggrannhet
Resnet	06m 06s	96.56
Alexnet	03m 37s	95.41
VGG-11	15m 55s	97.32
Densenet	13m 55s	97.70
Inception V3	21m 54s	98.09

Table 4.2: Sammanställning av finetuning

Modell	Tid	Max. noggrannhet
Resnet	02m 06s	80.50
Alexnet	01m 55s	80.50
VGG-11	03m 45s	77.35
Densenet	03m 35s	73.58
Inception V3	05m 15s	79.24

Table 4.3: Sammanställning av feature extraction

4.1.2 Finetuning

Vi kan se hur kostnadsfunktionen såg ut för varje epoch i figur 4.3 och hur träffsäkerheten var vid finetuning i figur 4.4

4.2 Eldstäder

4.2.1 Feature extraction

Vi kan se kostnaden för båda tränings och valideringsdatan i figur 4.5 för varje epoch. Vi kan även se i figur 4.6 hur träffsäkerheten för de olika modeller var på eldstäder.

4.2.2 Finetuning

Vi kan se hur kostnadsfunktionen såg ut för varje epoch i figur 4.7 och hur träffsäkerheten var vid finetuning i figur 4.8

Modell	Tid	Max. noggrannhet
Resnet	03m 31s	79.24
Alexnet	02m 08s	77.35
VGG-11	08m 56s	81.13
Densenet	07m 55s	85.53
Inception V3	12m 54s	83.64

Table 4.4: Sammanställning av finetuning

Modell	Tid	Max. noggrannhet
Resnet	02m 33s	93.75
Alexnet	02m 18s	93.22
VGG-11	04m 31s	93.75
Densenet	04m 20s	96.87
inception V3	06m 28s	93.75

Table 4.5: Sammanställning av feature extraction

4.3 Rum

4.3.1 Feature extraction

Vi kan se kostnaden för båda tränings och valideringsdatan i figur 4.9 för varje epoch. Vi kan även se i figur 4.10 hur träffsäkerheten för de olika modeller var på rum.

4.3.2 Finetuning

Vi kan se hur kostnadsfunktionen såg ut för varje epoch i figur 4.11 och hur träffsäkerheten var vid finetuning i figur 4.12

Modell	Tid	Max. noggrannhet
Resnet	04m 16s	96.35
Alexnet	02m 34s	94.79
VGG-11	11m 02s	97.91
Densenet	09m 53s	97.91
Inception V3	16m 10s	97.39

Table 4.6: Sammanställning av feature extraction

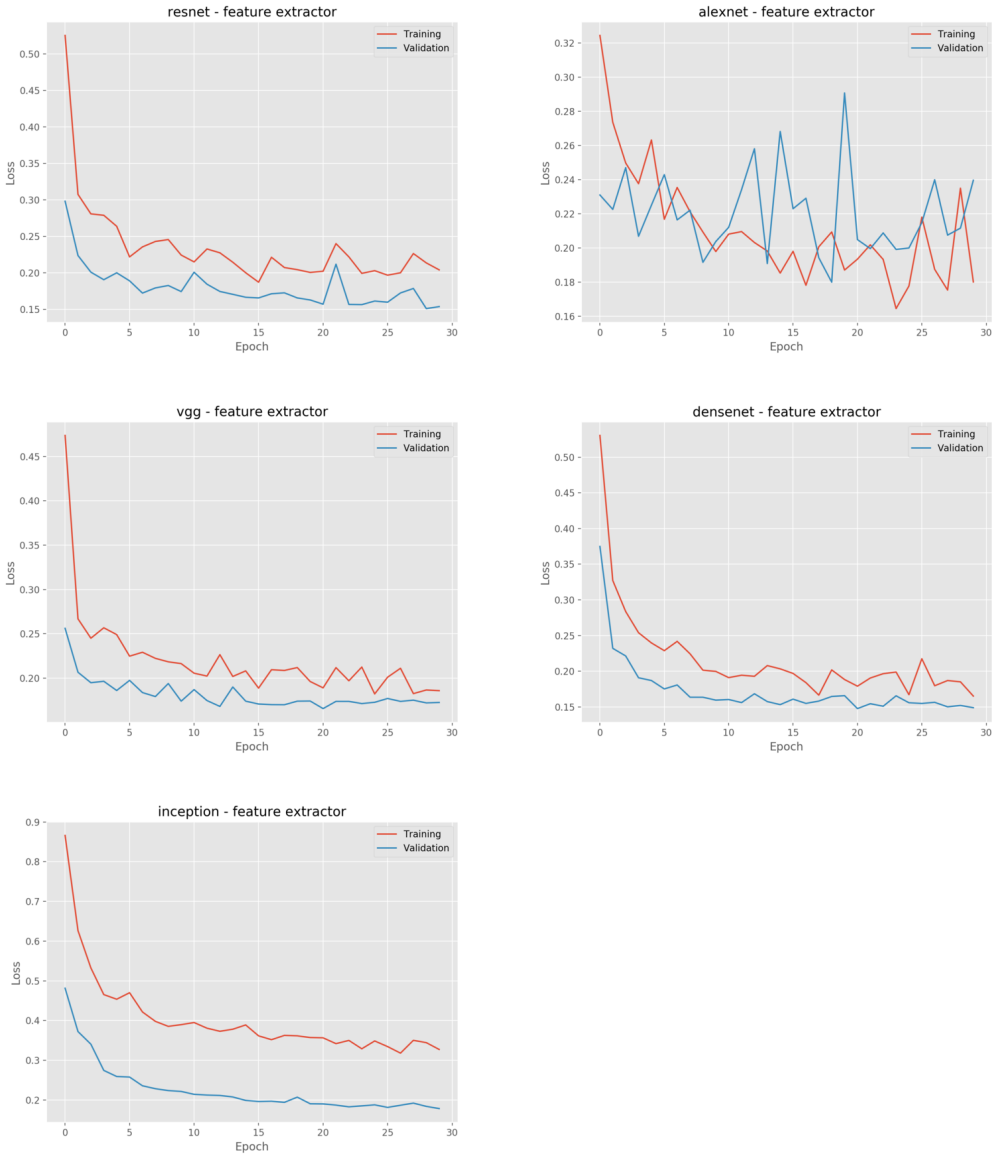

Figure 4.1: Kostnaden vid varje epoch för balkonger med feature extraction

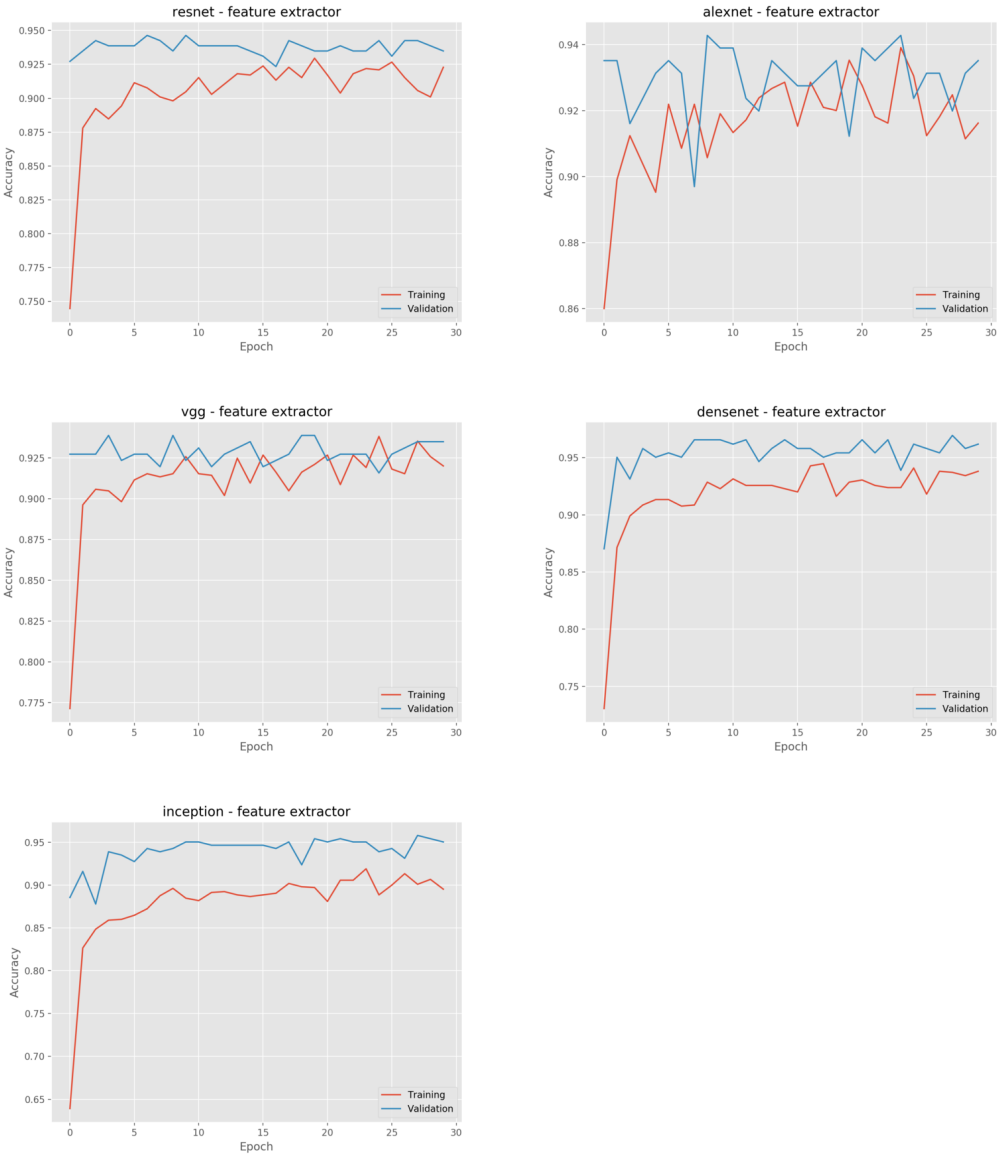

Figure 4.2: Träffsäkerhet för balkonger med feature extraction

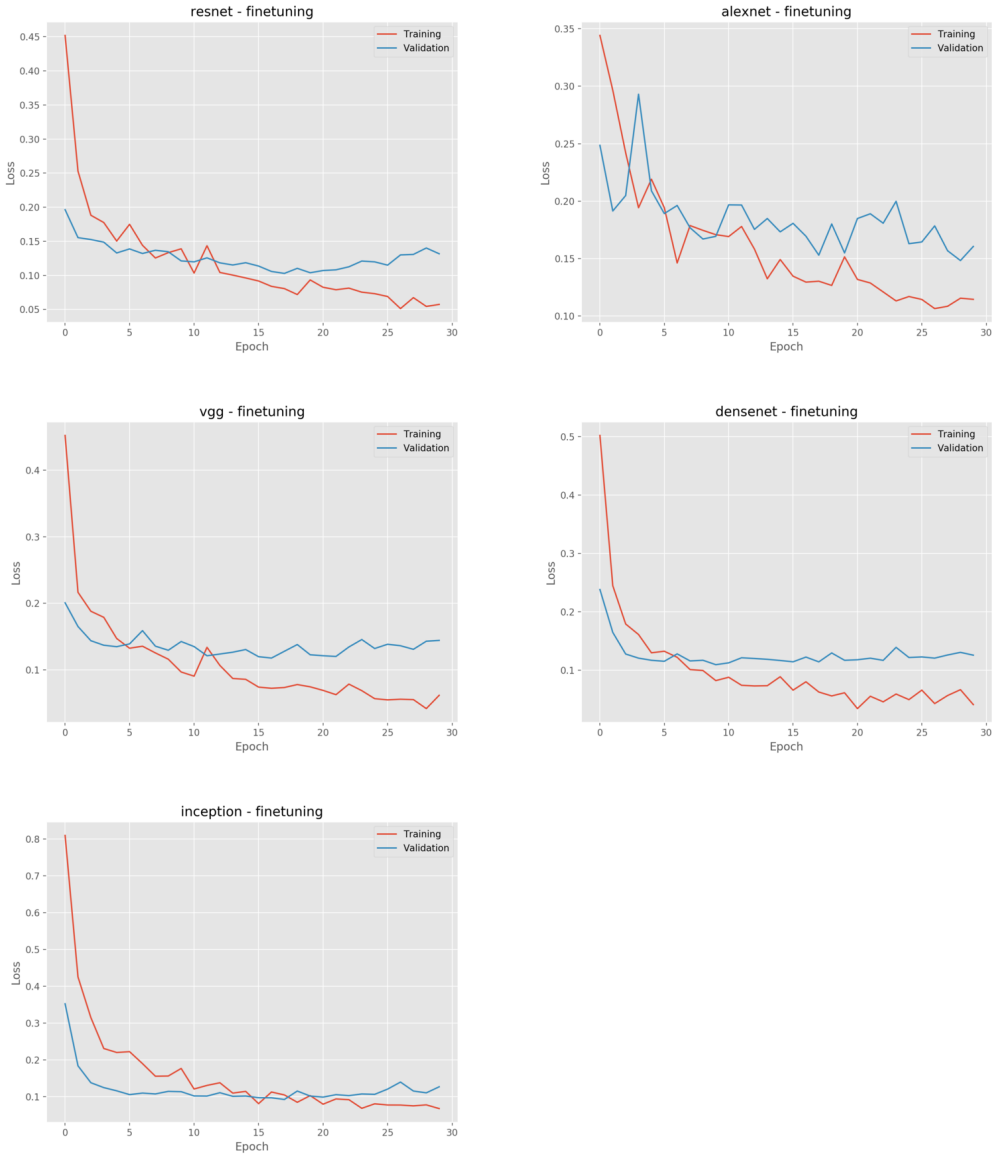

Figure 4.3: Kostnaden vid varje epoch för balkonger med finetuning

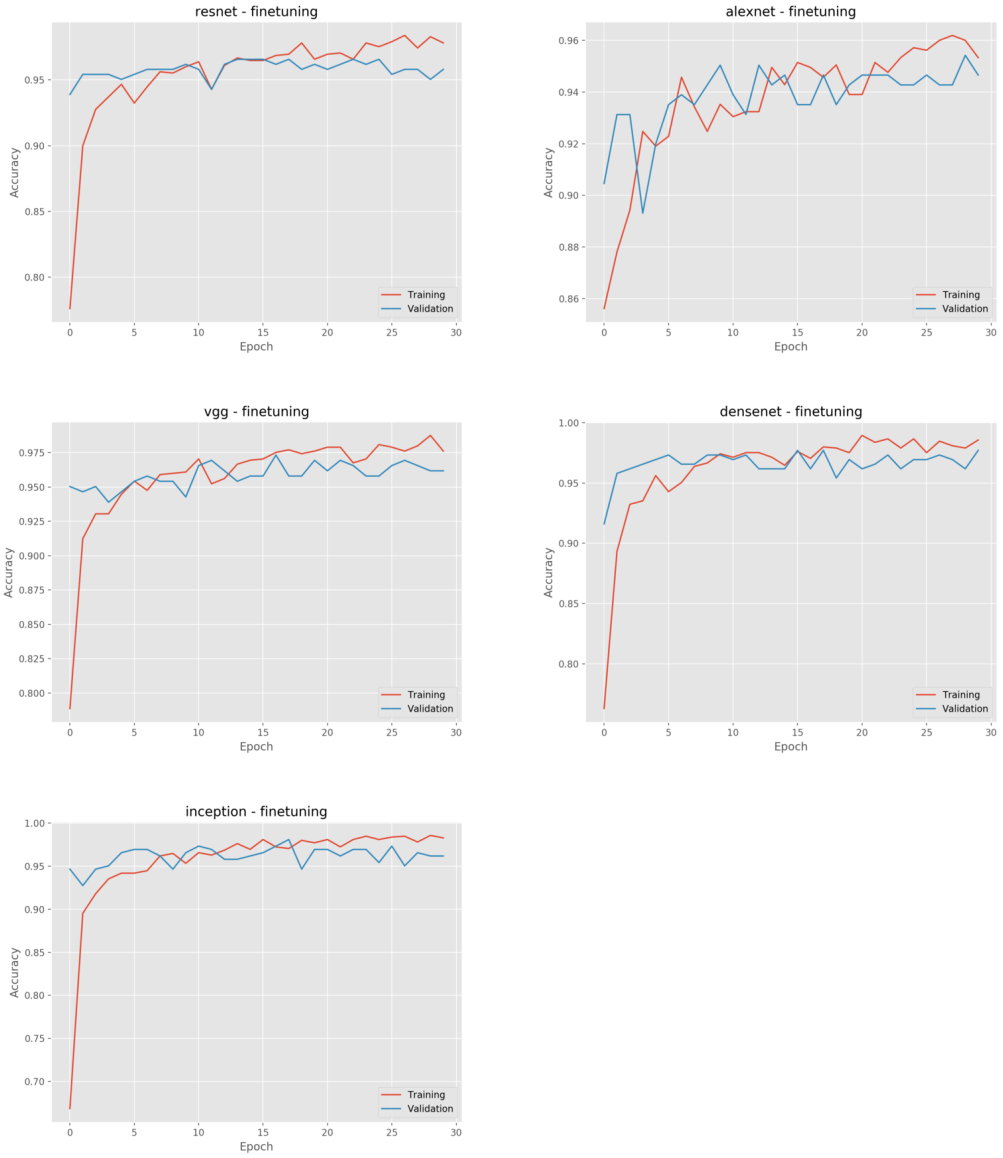

Figure 4.4: Träffsäkerhet för balkonger med finetuning

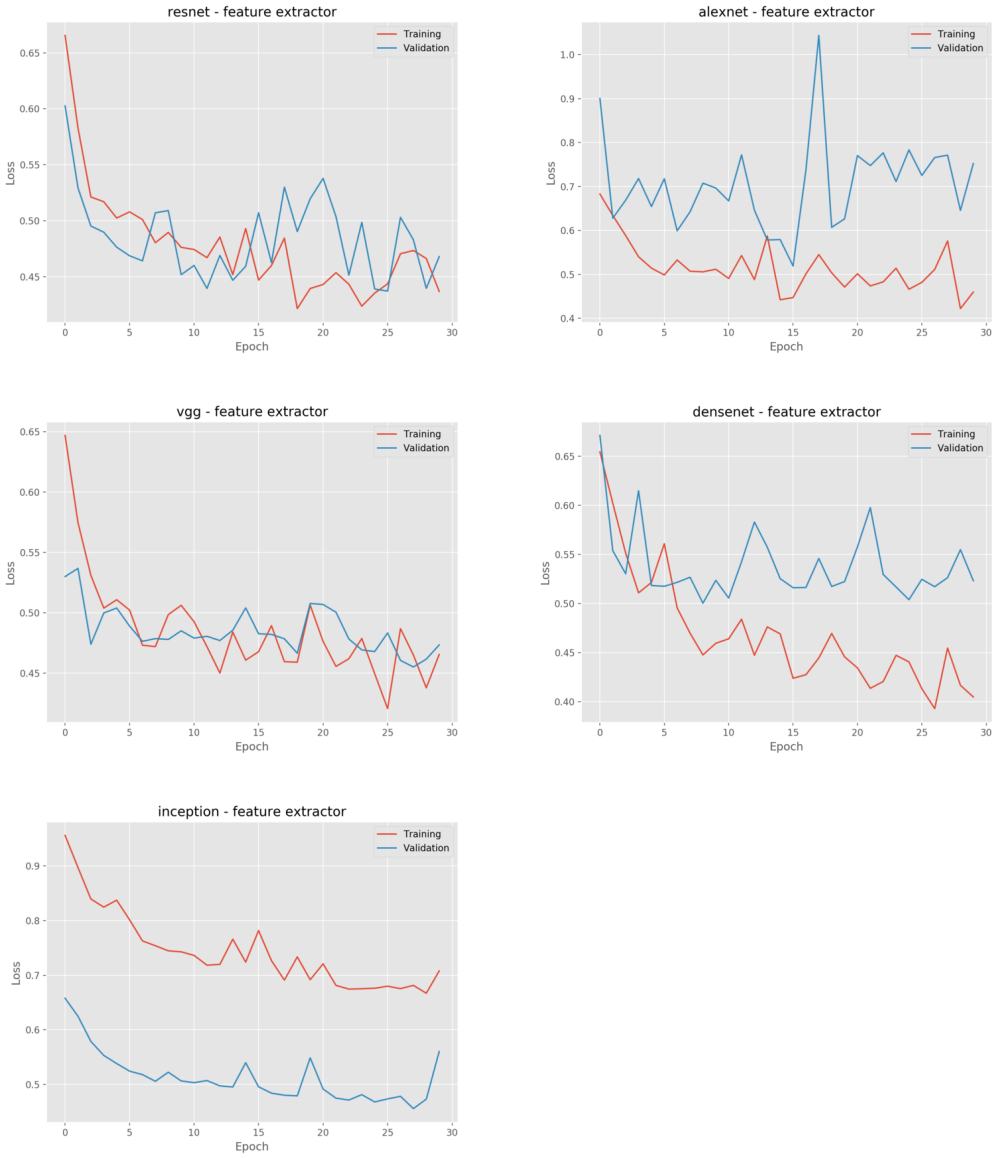

Figure 4.5: Kostnaden vid varje epoch för eldstäder med feature extraction

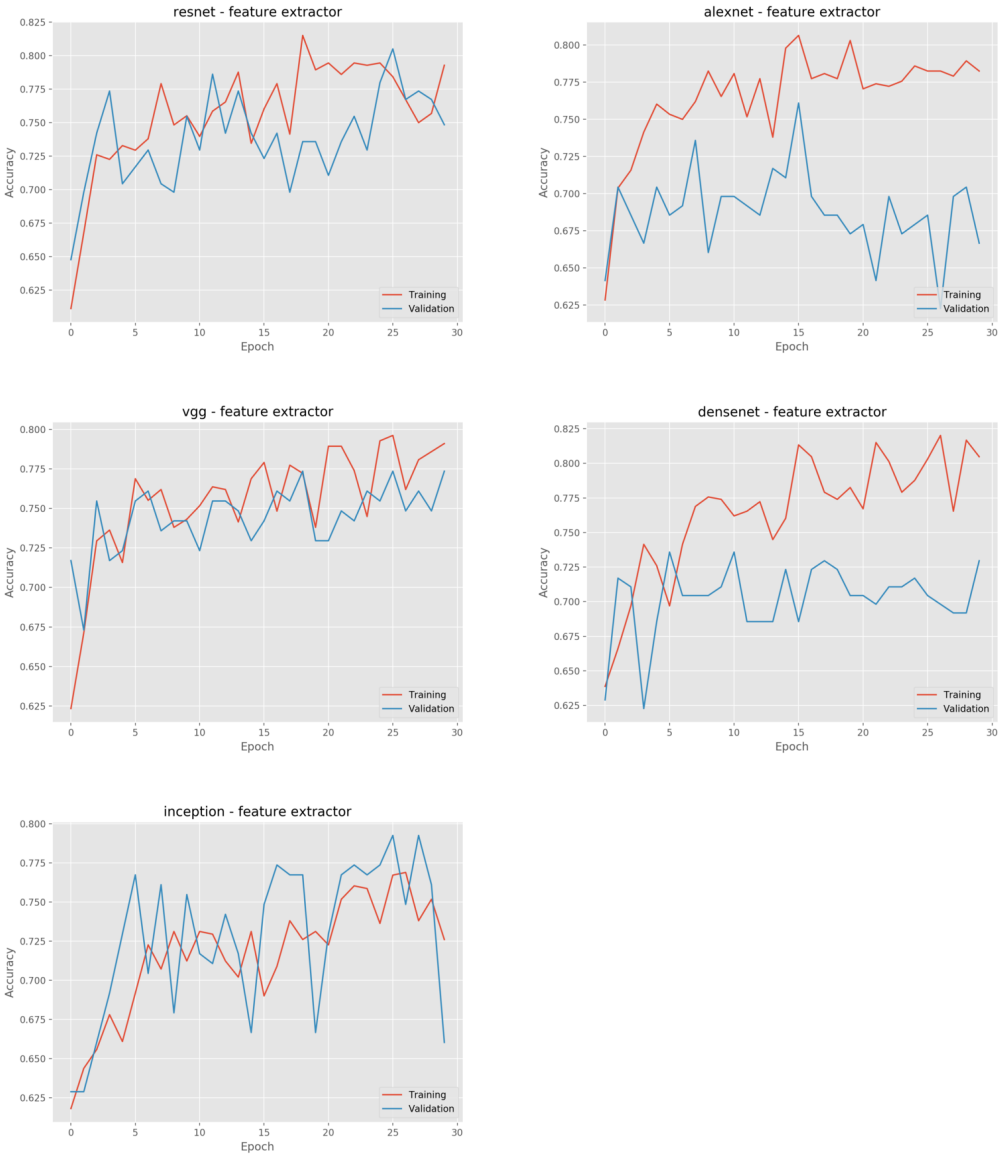

Figure 4.6: Träffsäkerhet för eldstäder med feature extraction

Figure 4.7: Kostnaden vid varje epoch för eldstäder med finetuning

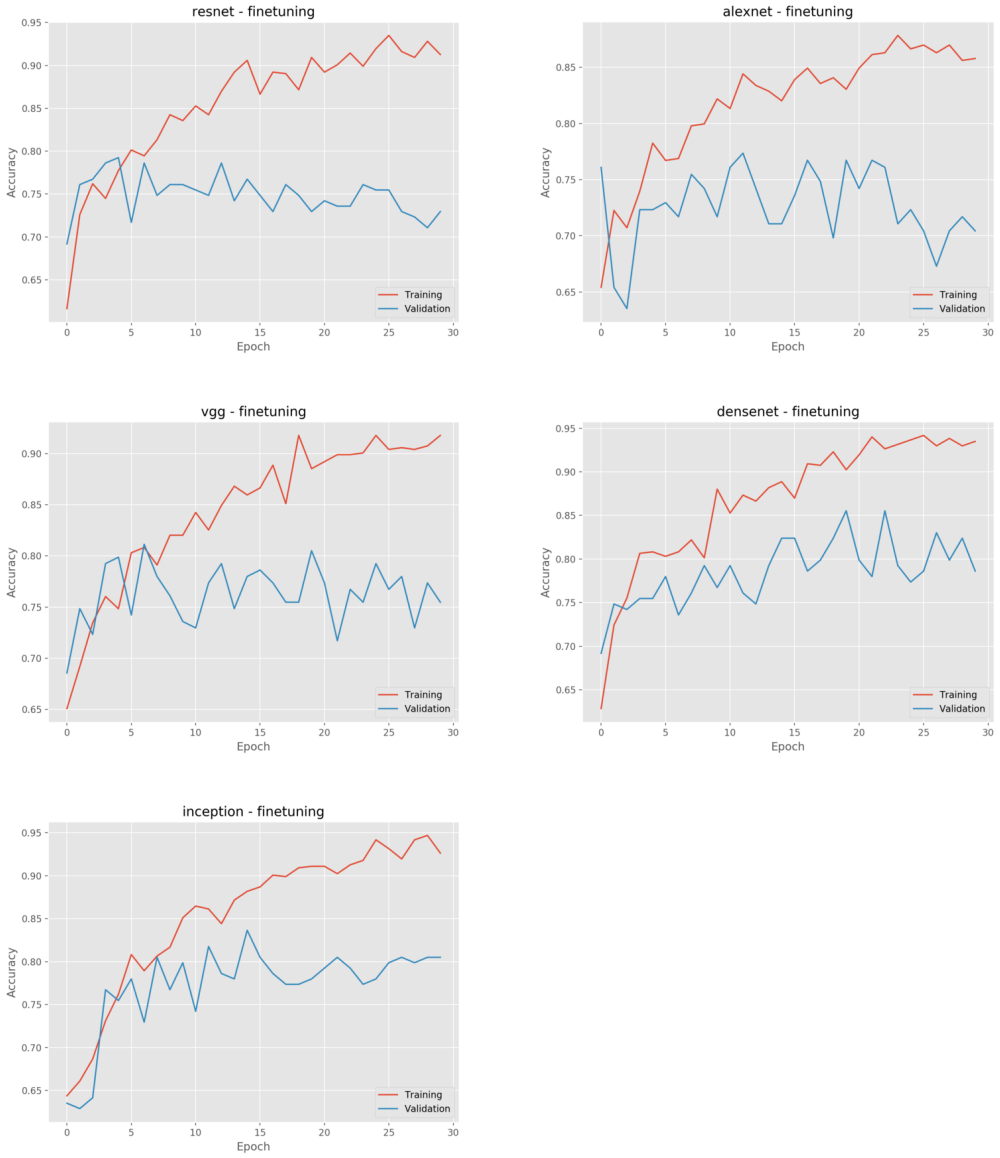

Figure 4.8: Träffsäkerhet för eldstäder med finetuning

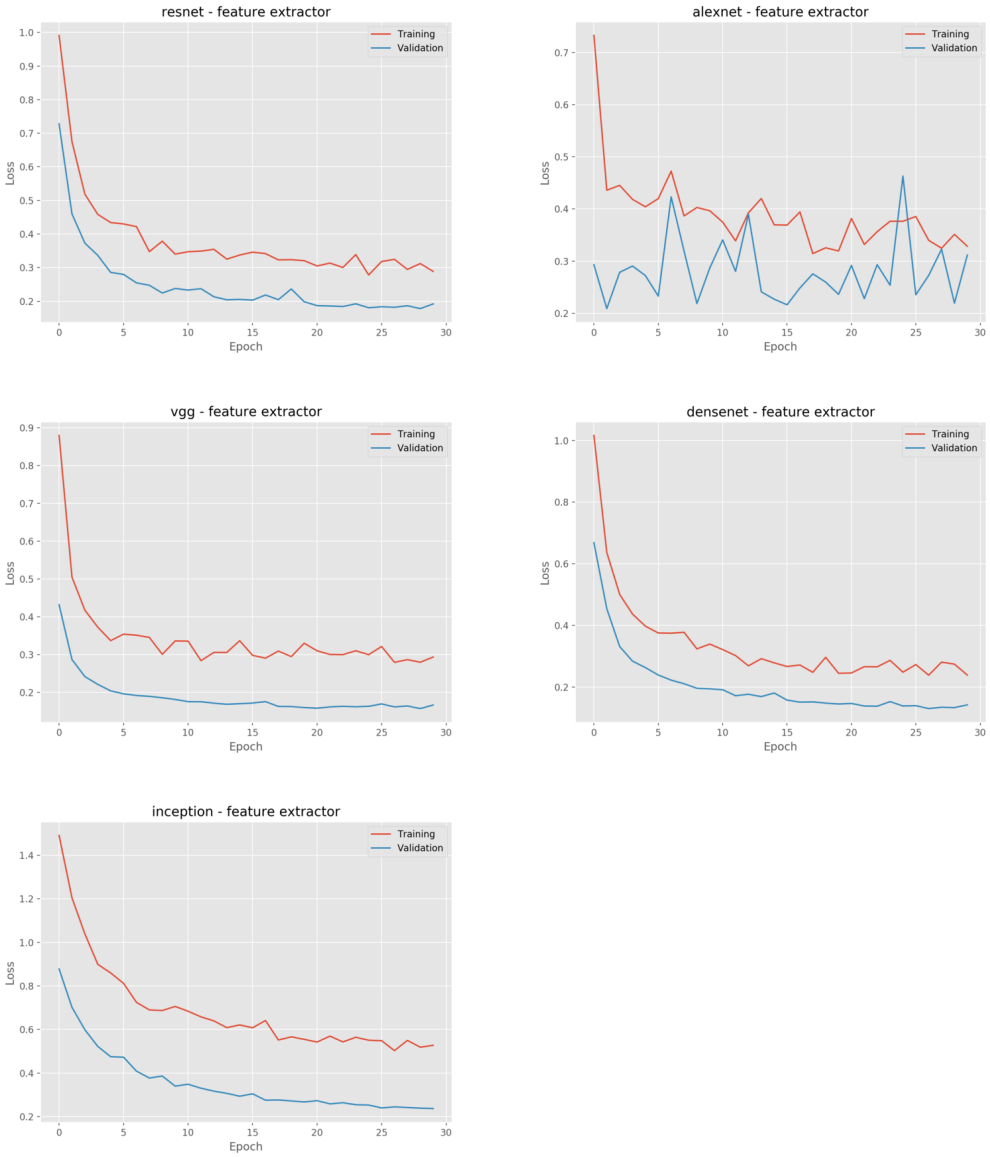

Figure 4.9: Kostnaden vid varje epoch för rum med feature extraction

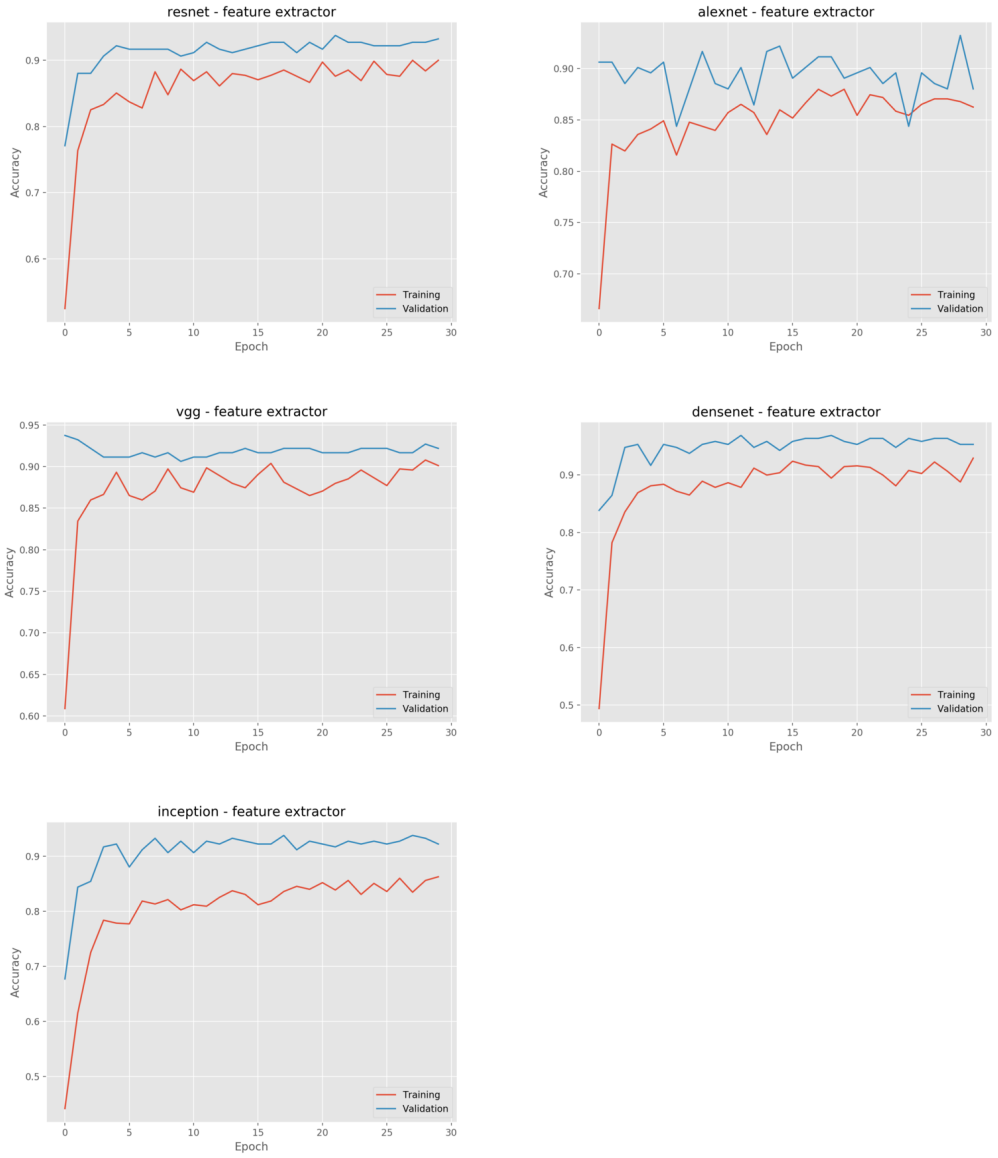

Figure 4.10: Träffsäkerhet för rum med feature extraction

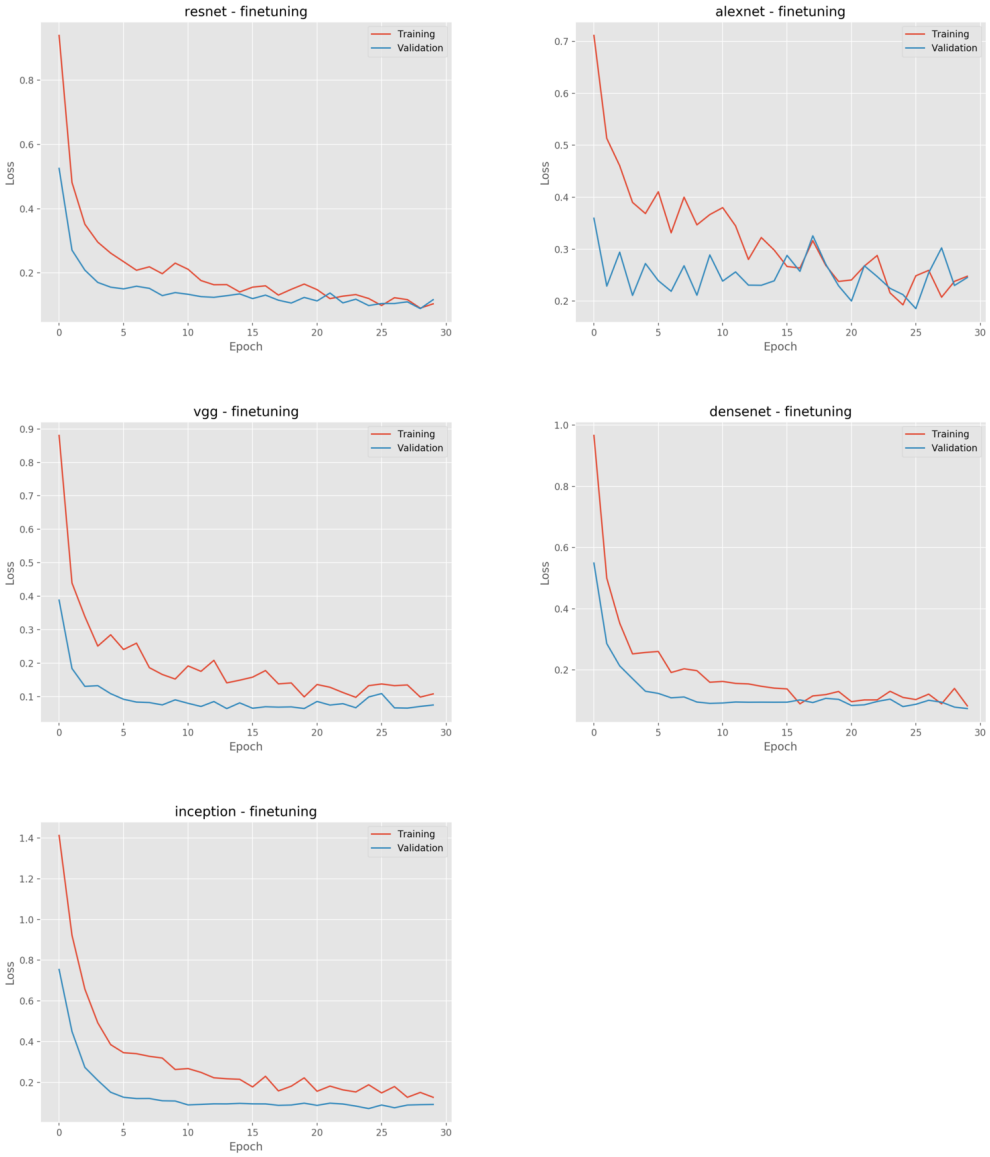

Figure 4.11: Kostnaden vid varje epoch för rum med finetuning

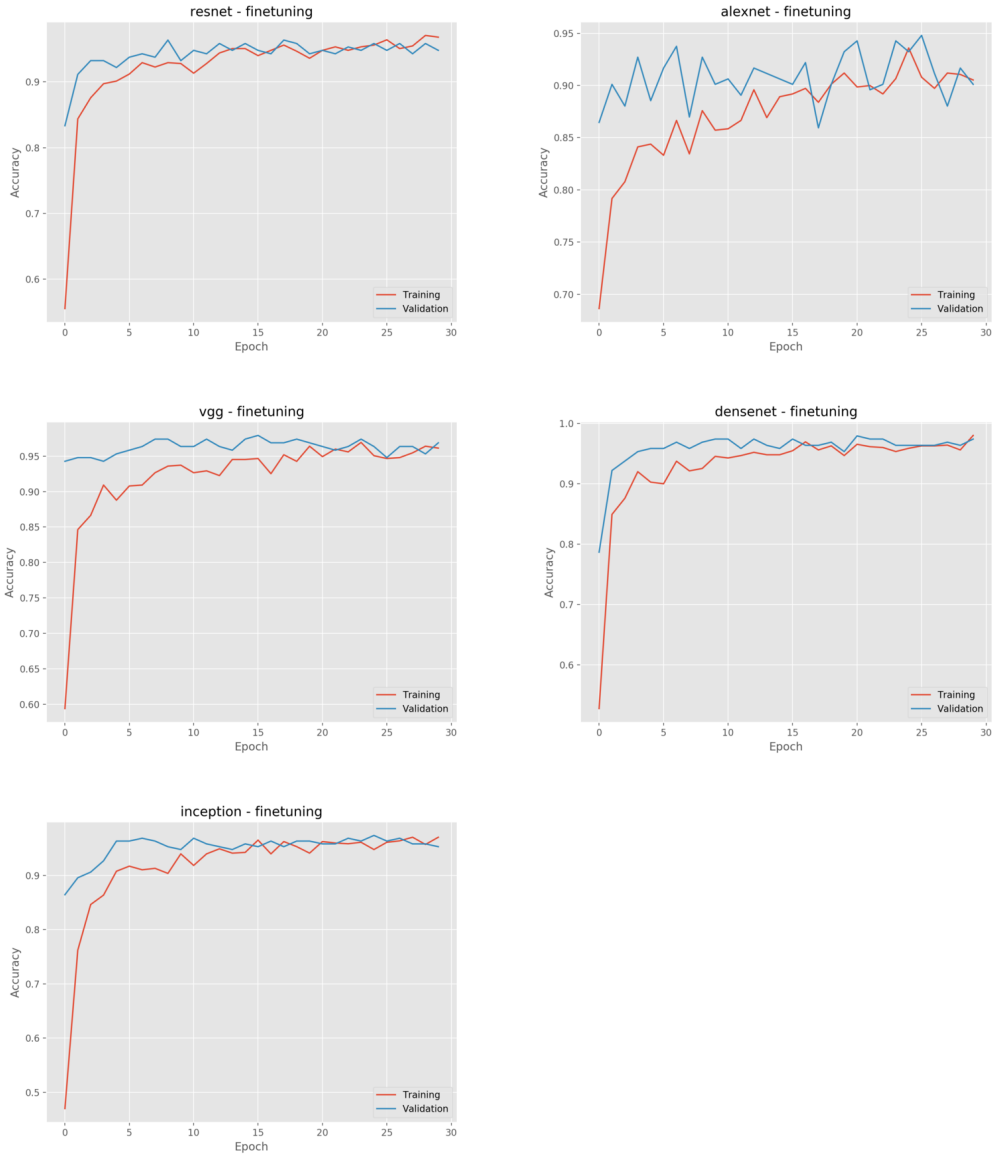

Figure 4.12: Träffsäkerhet för rum med finetuning

Chapter 5

Diskussion

Diskutera resultatet och hur olika delar kan ha påverkat eller påverkade. Diskutera eventuell framtida forskning. Begränsningar med resultatet. Etiska aspekter. Hållbarhet.

5.1 Fortsatt forskning

Vid värdering så är det också intressant att få ut attribut, så kan man räkna med det i värderingskalkylen.

Chapter 6

Slutsats

Slutsats av vad vi kom fram till. burk

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[2] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

Appendix A

Appendix A