En jämförelse av Eigenface- och Fisherface-metoden tillämpade i en Raspberry Pi 2

Examensarbete 2016

HUVUDOMRÅDE: *Datateknik*
FÖRFATTARE: *Dag Dahl och Gustaf Sterne*
HANDLEDARE: *Bengt Ekeberg*
JÖNKÖPING: 2016-02

Detta examensarbete är utfört vid Tekniska Högskolan i Jönköping inom datateknik. Författarna svarar själva för framförda åsikter, slutsatser och resultat.

Examinator: Ragnar Nohre
Handledare: Bengt Ekeberg
Omfattning: 15 hp (grundnivå)
Datum: 2016-02

Postadress:	Besöksadress:	Telefon:
Box 1026	Gjuterigatan 5	036-10 10 00 (vx)
551 11 Jönköping		

Abstract

The purpose with this report is to demonstrate the possibility to use Raspberry Pi 2 as hardware in a face recognition system. The study will show performance differences regarding the Eigenface- and Fisherface-method.

To demonstrate the possibility the authors have done tests using an experimental study and quantitative method. To review the tests and to understand the result a qualitative literature review was taken.

The tests will be presented as graphs to show the possibility to use Raspberry Pi 2 as hardware in a face recognition system. The same goes for the comparison of the chosen algorithms. The work indicates that Raspberry Pi 2 is a possible candidate to use for smaller face recognition systems. There is also an indication that the Fisherface method is the better choice for face recognition.

Keywords - Face recognition, Eigenfaces, Fisherfaces, Raspberry Pi 2, Python, OpenCV, OpenCV_contrib, psutil.

Postadress:	Besöksadress:	Telefon:
Box 1026	Gjuterigatan 5	036-10 10 00 (vx)
551 11 Jönköping		

Sammanfattning

Syftet med rapporten är att visa möjligheten att använda Raspberry Pi 2 i ett ansiktsigenkänningssystem. Studien redogör för prestandaskillnader mellan Eigenface och Fisherface-metoden.

Studieförfattarna har genomfört en experimentell studie enligt en kvantitativ metod där tester utgör empirin. Resultatet från testerna kommer presenteras genom diagram och påvisa möjligheten att använda Raspberry Pi 2 som hårdvara i ett ansiktsigenkänningssystem. Genom samma testutförande kommer skillnader mellan igenkänningsmetoderna att påvisas.

Studien visar att Raspberry Pi 2 är en lämplig kandidat att använda för mindre ansiktsigenkänningssystem. Vidare framgår det att Fisherface-metoden är det lämpligaste valet att använda vid implementation av systemet.

Nyckelord – Ansiktsigenkänning, Eigenfaces, Fisherfaces, Raspberry Pi 2, Python, OpenCV, OpenCV_contrib, psutil

Innehållsförteckning

Abstract 3
Sammanfattning 4
Ordlista 30
Innehållsförteckning 1
1 2
1.1 Bakgrund 2
1.2 Problembeskrivning 2
1.3 Syfte och frågeställningar 2
1.4 Omfång och avgränsningar 3
1.5 Disposition 3
2 Metod och genomförande 4
2.1 Koppling mellan frågeställningar och metod 4
2.2 Arbetsprocessen 4
2.3 Datainsamling 4
2.4 Dataanalys 5
2.5 Tester 5
2.6 Trovärdighet 7
3 Teoretiskt ramverk 8
3.1 Koppling mellan frågeställningar och teori 8
3.2 Eigenface-metoden 8
3.3 Fisherface-metoden 8
3.4 Verktyg 9
4 Empiri 10
4.1 Testfall 1 10
4.2 Testfall 2 13
4.3 Testfall 3 16
4.4 Testfall 4 18
4.5 Testfall 5 22
5 Analys 26
5.1 Frågeställning 1 26
5.2 Frågeställning 2 26
6 Diskussion och slutsatser 28
6.1 Resultat 28
6.2 Implikationer 28
6.3 Begränsningar 28
6.4 Slutsatser och rekommendationer 28
6.5 Vidare forskning 28
Referenser 29
Bilagor 30

1

Kapitlet ger en bakgrund till det problemområde som studien byggts upp kring. Vidare presenteras studiens syfte och dess frågeställningar. Därtill beskrivs studiens omfång och avgränsningar. Kapitlet avslutas med rapportens disposition.

1.1 Bakgrund

Arbetet grundar sig i ett uppdrag från Combitech AB. Båda parterna fann det motiverat att utforska användningsområdet och tillämpningar av "face recognition" (fortsättningsvis ansiktsigenkänning) i en Raspberry Pi 2.

Ansiktsigenkänning är ett begrepp som syftar till hur datorer genom mjukvara kan identifiera, verifiera och komma ihåg ett ansikte [1]. För själva tillämpningen av ansiktsigenkänningen finns det flera tillvägagångssätt där i huvudsak valet av metod och programmeringsspråk påverkar implementeringen. Två av de vanligaste och mest tillämpade metoderna är Eigenfaces och Fisherfaces [2].

Det vanligaste användningsområdet för ansiktsigenkänning är säkerhetssystem i en övervakningstillämpning. Vidare går det även att använda ansiktsigenkänning för att identifiera demografiska uppgifter om användaren till exempel ålder och kön samt vilket känslomässigt tillstånd personen befinner sig i [1].

Ansiktsigenkänning tillämpas vanligtvis på en PC, dock finns det i dagsläget billigare teknik i form av enkortsdatorer. Prestandan på dessa är sämre kontra en PC men samtidigt är kostnaden lägre, energiförbrukningen och den fysiska storleken mindre. Det finns ett flertal olika enkortsdatorer att tillgå där Raspberry Pi 2 är en av de mest populära [3].

1.2 Problembeskrivning

År 1991 publiceras en artikel skriven av M. Turk vars innehåll presenterar Eigenface-metoden [4]. Sex år senare, 1997, presenteras Fisherface-metoden i en artikel skriven av Peter N. Belhumeur, Joao P. Hespanha, och David J. Kriegman. Förutom att beskriva Fisherface-metoden, så jämför den också Fisherface med Eigenface-metoden. Det som i huvudsak jämförs är hur väl metoderna presterar gällande igenkänning och hur ljusförhållande på test- och tränings-bilder påverkar igenkänningsresultatet. I artiklarna framgår det inte vilken hårdvara som används eller hur krävande de två metoderna är [5].

1.3 Syfte och frågeställningar

I problembeskrivningen framgår det att studierna som gjorts gällandes ansiktsigenkänning är föråldrade och hårdvaran som användes anses idag vara omodern. Syftet med dessa studier har varit att presentera och undersöka de olika metoderna snarare än att påvisa vilken hårdvara som anses lämplig. Det är även brist på studier kring igenkänningssystem vars hårdvara är baserad på enkortsdatorer, mer specifikt Raspberry Pi 2. Därmed är syftet med denna studie: Visa att Raspberry Pi 2 är en tillräckligt kraftfull hårdvara för ett ansiktsigenkännings-system.

Det finns ett antal metoder som kan användas vid skapandet av ett ansiktsigenkännings-system. Valet av metod påverkar både implementeringen och resultatet av igenkänningen. Därmed är även syftet med denna studie: Visa vilken metod av Fisherfaces och Eigenfaces som är mest lämpad att tillämpa i ett system med Raspberry Pi 2.

För att kunna besvara syftet har den brutits ned i två frågeställningar. Den första frågan är att påvisa att Raspberry Pi 2 är tillräckligt kraftfull för att implementera och använda Fisherfaces och Eigenfaces-metoden samt att programexekveringen sker inom en rimlig tidsram. Därmed är studiens första frågeställning: Hur lämpar sig Raspberry Pi 2 som hårdvara för ett igenkänningssystem både gällande träning och igenkänning?

För att besvara den första frågeställningen måste de valda metoderna implementeras. När dessa är implementerade kan den första frågeställningen besvaras, dock inte med någon insikt till varför det blev just det utfallet. Genom att påvisa likheter och skillnader via tester kan det klargöras vilken av metoderna som är mest lämpad för det tänkta systemet. Därför är den andra frågeställningen: Vilka prestandaskillnader finns mellan Fisherface- och Eigenface-metoden gällande träning och igenkänning?

1.4 Omfång och avgränsningar

Studien har genomförts med begränsad tid och ekonomiska resurser. Tidsfaktorn är den mest avgörande parametern gällande omfång och avgränsningar av arbetet.

Systemet kommer använda sig av en Raspberry Pi 2 och tillhörande kamera. Programkod kommer skrivas i Python. Algoritmerna kommer implementeras med hjälp av OpenCV och OpenCV_contrib biblioteken. Egenskrivna hjälpfunktioner kommer förekomma i form av bildbehandling, bildhantering och texthantering.

Vid testförfarandet tar systemet inga egna bilder utan använder sig av YaleB-biblioteket som består av 2470 bilder fördelat mellan 38 individer. Bildbiblioteket sparas och hanteras lokalt på Raspberry Pi 2-enheten och av den anledningen uteblir kameran i testutförandet.

Noggrannheten finns redan studerad [5], därför kommer studien inte att undersöka den parametern närmare.

1.5 Disposition

Kapitel 1: **Inledning**: Består av bakgrund, problembeskrivning, syfte frågeställning och avgränsningar.
Kapitel 2: **Metod och genomförande**: Beskriver de metoder som används för att genomföra studien och de tester som utgör empirin som ligger till grund för resultat och analys.
Kapitel 3: **Teoretiskt ramverk**: Presenterar de teorier som studien bygger på.
Kapitel 4: **Empiri**: Presenter resultat
Kapitel 5: **Analys**: Analys av resultat och återkoppling till frågeställningarna
Kapitel 6: **Diskussioner och slutsatser**: Slutsatsen av arbetet beskrivs. Diskussion av analysen samt förslag på hur studien kan utvecklas presenteras.

2 Metod och genomförande

Kapitlet ger en översiktlig beskrivning av studiens arbetsprocess. Vidare beskrivs studiens ansats och design. Därtill beskrivs studiens datainsamling och dataanalys. Kapitlet avslutas med en diskussion kring studiens trovärdighet.

2.1 Koppling mellan frågeställningar och metod

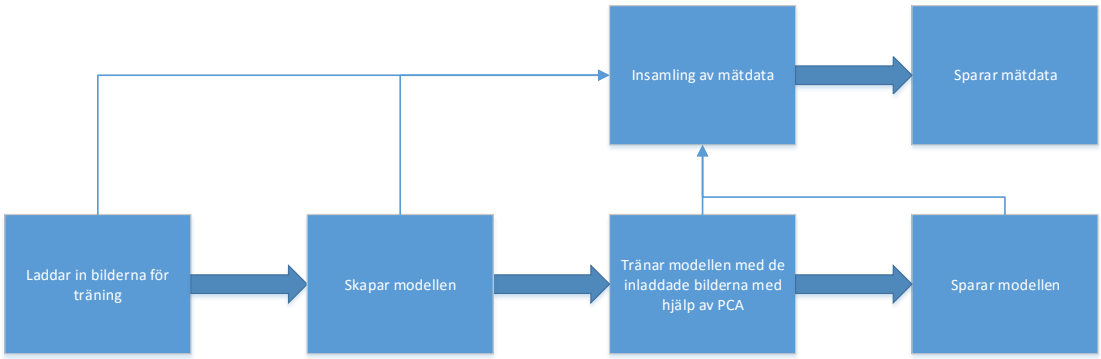
För att besvara studiens första frågeställning har ett antal tester genomförts. Testerna syftar till att mäta processorns arbetsbelastning, RAM-minnets användning, temperatur gällande både träning och igenkänning samt fysiskt minne för träningsmodellen. Det kommer även att göras mätningar kring hur snabbt de olika delarna exekveras. Resultaten från experimenten kommer utgöra grunden för att besvara den första frågeställningen. Insamlade testdata kommer redovisas och genom att analysera resultaten kommer den första frågeställningen kunna besvaras. För att säkerhetsställa att resultaten är korrekta och tolkas rätt genomförs litteraturstudier gällande både implementering och test av systemet. Med hjälp av resultatet från testerna kommer även den andra frågeställningen att kunna besvaras. Testdatan kommer visa på skillnader och likheter mellan de metoder som har undersökts.

2.2 Arbetsprocessen

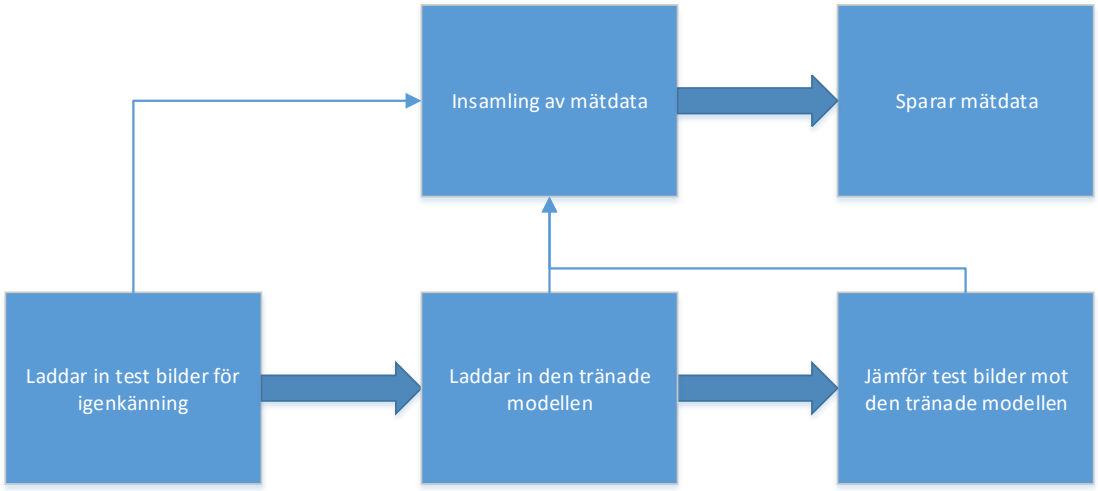
Studieförfattarna kontaktade Combitech i syfte att bestämma tid för ett möte och presentera sin idé angående examensarbete. Efter mötet beslutades att examensarbetet skulle genomföras i samråd med Combitech. Under kommande möten beslutades att arbetet skulle följa den agila arbetsmetoden Scrum och att studien skulle vara experimentell. Parallellt med de första mötena påbörjades en förstudie där relevant litteratur och teorier införskaffats och bearbetats. Det skapades även en planeringsrapport som senare presenterades för handledare på Combitech och Jönköpings Tekniska Högskola(fortsättningsvis JTH). I nästa skede påbörjades fördjupning av litteratur för implementering av Fisherface- och Eigenface-metoden. Vidare inleddes regelbundna sprintar med handledaren på Combitech som pågick under hela arbetsprocessen.

2.3 Datainsamling

Studiens datainsamling består av tester. Testerna är uppdelade i två huvuddelar i form av en träningsdel och en igenkänningsdel. Vid exekvering av programkoden körs det parallellt ett testscript som mäter och sparar data till CSV-textfiler. Resultatet från testerna kommer sedan att presenteras i form av grafer för att få en lättöverskådlig översikt av resultat från mätdatan.



Figur 1: Visar förloppet av datainsamlingen och de ingående delarna i träningsdelen



Figur 2: Visar förloppet av datainsamlingen och de ingående delarna i igenkänningsdelen

2.4 Dataanalys

Arbetet använder sig av en kvantitativ metod för att besvara båda frågeställningarna. De litteraturstudier som utgör studiens kvalitativa metod har använts för att implementera metoderna samt att förstå och tolka utfallet av mätdatan. Eftersom resultaten grundar sig i en experimentell studie så är dataanalysen empirisk.

2.5 Tester

I testvärlden finns det två centrala begrepp, verifiering och validering.
Verifiering är den del som utvärderar programmet genom att fastställa huruvida produkten lever upp till kraven som ställts innan testerna påbörjats.
Validering i sin tur syftar till att konfirmera att slutprodukten lever upp till sitt användningssyfte och stort fokus ligger på slutprodukten [6]. Eftersom igenkänningssystemet inte kommer produceras eller säljas för kommersiellt bruk riktas därför fokus på verifiera att produkten lever upp till följande krav:

- Att programkod går att köra felfri utan felmeddelande, varningar eller krascher.
- Att RAM-minnet inte använder 100 procent av sin kapacitet
- Att temperaturen inte överstiger 80 grader Celsius
- Att processorn inte använder 100 procent av sin kapacitet

För att kunna testa ett program krävs en sekvens av testaktiviteter.
Första steget är att identifiera vad som är relevant att testa och hur testerna ska utformas och genomföras [6]. Det är av yttersta vikt att båda metoderna testas under samma förutsättningar och att yttre faktorer i största möjliga mån inte påverkar testresultaten. När testerna är utförda och datan insamlad återstår det att tolka och dokumentera resultatet.

Testerna är noga planerade och dokumenterade. Alla testfall med tillhörande testdata finns sparad och kan återskapas. Vidare är Raspberry Pi 2:s operativsystem och installerad mjukvara säkerhetskopierade och därmed kan testförhållandena återskapas vid behov.

Bildbibliotek som används är YaleB som totalt består av 38 personer och 2470 bilder. För varje person och mapp finns det 65 bilder. 11 av dessa bilder används för att testa igenkänning och 54 används för att träna modellen. Urvalet sker genom att med start på den första bilden i mappen, samt att var 6:e bild efter denna plockas ut och används för testning. Resterande bilder används för att träna modellen. Detta för att säkerställa variation i testbilderna i avseende ljussättning och ansiktsdrag.

Testfall 1 består av 108 bilder i tränaren och utöver dessa används 22 bilder som testbilder. Bilderna motsvarar 2 personer. Mätintervallet av testdatan är 0,01 sekund det vill säga att för varje 0,01 sekund så hämtas samtliga parametrar(RAM, CPU, Temperatur) och dessa sparas till CSV filen.
Testfall 2 består av 216 bilder i tränaren och utöver dessa används 44 bilder som test bilder. Bilderna motsvarar 4 personer. Mätintervallet av testdatan är 0,01 sekund

Testfall 3 består av 432 bilder i tränaren och utöver dessa används 88 bilder som test bilder. Bilderna motsvarar 8 personer. Mätintervallet av testdatan är 0,01 sekund
Testfall 4 består av 1080 bilder i tränaren och utöver dessa används 220 bilder som test bilder. Bilderna motsvarar 20 personer. Mätintervallet av testdatan är 0,1 sekund
Testfall 5 består av 2052 bilder i tränaren och utöver dessa används 418 bilder som test bilder. Bilderna motsvarar 38 personer. Mätintervallet av testdatan är 1 sekund

Testerna utförs med hjälp av PSUTIL och time-biblioteken samt en temperaturfunktion hämtad från Raspberry Pi's hemsida.
Testanropen itereras i en while loop och testdata sparas i CSV-format till en textfil. Iterationen avslutas när testfall och program som ska testas kört klart.

- Alla bilder redigeras och erhåller 92 * 112 upplösning.
- Både tränings- och testbilder varierar i ljussättning och ansiktsdrag.
- CPU, RAM och temperatur hämtas i ett jämnt tidsintervall.

För att skapa ett tidsuppehåll används time-biblioteket. Programkoden för ett tidsuppehåll på en sekund skrivs som nedanstående:

time.sleep(1)

För att mäta tiden så skrivs:

t1 = time()
#CODE TO MESSURE
time = time() - t1

Vilket ger tillbaka tiden i sekunder[7].

RAM-informationen hämtas med hjälp av anropet

psutil.virtual_memory().percent

Vilket returnerar minnesanvändningen i procent

CPU-informationen hämtas med hjälp av anropet:

psutil.cpu_percent()

Vilket returnerar CPU-användningen i procent.

Temperatur
Temperaturen hämtas med hjälp av följande funktion och anrop.
Funktionen används för att läsa av temperaturen på en av processor pinnarna

```
def getCPUtemperature():
        res = os.popen('vcgencmd measure_temp').readline()
        return(res.replace("temp=","").replace("'C\n",""))
```

Alltså blir anropet:

getCPUtemperature()

Vilket returnerar temperaturen på processorn i grader Celsius.

2.6 Trovärdighet

Studieförfattarna har använt sig av ursprungsartiklarna där de båda metoderna presenterades och använt sig av OpenCV's redan implementerade funktioner, detta för att säkerställa en korrekt implementering och förståelse av systemet. Testerna är noggrant genomförda och testdata, programkod och mjukvara är säkerhetskopierad. Bilaga 1 innehåller en länk till systemets program kod.

3 Teoretiskt ramverk

Kapitlet ger en teoretisk grund och förklaringsansats till studien och det syfte och frågeställningar som formulerats.

3.1 Koppling mellan frågeställningar och teori

För att ge en teoretisk grund till arbetet beskrivs följande områden i det teoretiska ramverket: Eigenface-metoden, Fisherface-metoden, Raspbian, Python, Raspberry Pi 2, OpenCV, OpenCV_contrib och PSUTIL. Eigenfaces och Fisherfaces beskrivs eftersom det är grunden för ett ansiktsigenkänning-system och att det är de metoderna som skall undersökas i studien. Raspbian och Python skall beskrivas på grund av att valet av operativsystem och programmeringsspråk påverkar både implementeringen och hur effektivt systemet är. Vidare beskrivs OpenCV och OpenCV_contrib då de innehåller funktioner som krävs för att implementera metoderna. Testerna använder sig i huvudsak av PSUTIL och därutav kommer biblioteket att beskrivas.

3.2 Eigenface-metoden

Eigenface-metoden använder sig av Principal Component Analysis (fortsättningsvis PCA) vilken bygger på att förvandla ett set av sammanhängande variabler till ett mindre set av osammanhängande. Iden är att man ur ett hög dimensionellt dataset plockar ut den viktigaste informationen och tar bort vad som anses överflödigt. PCA metoden hittar de riktningar som har störst varians i datan [8].

PCA utgör kärnan i Eigenface metoden. Den hittar en linjär kombination av drag som maximerar den totala variansen i datan. Även om det är en kraftfull metod så använder den sig inte av några klasser vilket medför att mycket data kan gå förlorad [8]. Vanligaste orsaken till att data går förlorad är att det är stor skillnad på ljus och ansiktsdrag[5][1]. Metoden har också svårt att hantera variationer i position, till exempel om testpersonen står vinklad eller snett [1].

Eigenface-metoden känner igen ansikten genom att:

- Projicera all träningsdata i ett PCA sub-plan
- Projicera all testdata på detta PCA sub-plan

- Söka efter den närmaste grannen mellan de projicerade träningsbilder och testbilderna [8].

Om ansiktet finns i en bild N(x, y) så anses den vara en två dimensionell NxN array. En bild kan även normaliseras till en N^2 vektor. Alltså blir en bild med storleken 256 * 256 en vektor av dimension 65 536. Ansiktet kan ses som en rad punkter i detta 65 536 dimensionellt plan. Enligt denna premiss så skapar ett set med ansiktsbilder ett låg dimensionellt sub-plan i det totala planet. Det är med hjälp av PCA som dessa sub-plan av ansikten identifieras [1].

3.3 Fisherface-metoden

Fisherfaces utvecklades av Belhumeur, Hespanha och Kriegman, 1997. Deras arbete bygger på statistikern Sir R. A. Fishers metod Linear Discriminant Analysis (fortsättningsvis LDA). Belhumeur, Hespanha och Kriegman var dock först med att tillämpa LDA i ansiktsigenkänningssammanhang, vilket också blev starten för Fisherface-metoden [8]. Till skillnad från Eigenface-metoden så är Fisherfaces robust mot variationer i ljus[5]. Fisherfaces använder sig av en PCA följt av en LDA och har därutav vissa likheter med Eigenfaces.

LDA bygger på en klasspecifik dimensionell reducering [5]. För att urskilja den bästa kombinationen av ansiktsdrag som separerar klasserna, maximeras ration mellan klasserna till inre klass-spridning, till skillnad mot Eigenface-metoden som maximerar övergripande spridning. Iden bygger på att samma klasser skapar ett närliggande kluster, vilket innebär att olika klasser hamnar så långt ifrån varandra som möjligt [8].

Genom LDA så upptäcks ansiktsdrag för att särskilja olika personer. LDA letar efter skillnader och PCA letar efter likheter. Utöver detta så följer Fisherface-metoden samma steg som Eigenface-metoden:

- Projicera all träningsdata i ett sub-plan
- Projicera all testdata på detta sub-plan

- Och sedan leta efter den närmaste granne mellan de projicerade träningsbilderna och testbilderna.

Den stora skillnaden är hur datan projiceras i planet [8].

3.4 Verktyg

Raspberry Pi 2
Raspberry Pi 2 är en lågkostnadsenkortsdator som bland annat har HDMI-utgång, USB 2.0 och Ethernet-anslutning. Den är baserat på en ARM-plattform med en fyrakärning processor med en klockfrekvens på 900 MHz och 1Gb RAM att tillgå. Vidare lämpar den sig till Python och programmeringsprojekt [9].

Python
Python är ett objektorienterat programmeringsspråk som används på olika UNIX plattformar däribland Raspberry Pi 2. Python är ett högnivåspråk som kan tillämpas på flera olika områden och det finns ett stort utbud av bibliotek, bland annat OpenCV och PSUTIL [10].

OpenCV
OpenCV är ett öppet kodbibliotek som innehåller flera olika algoritmer som Haar Cascade, Fisherfaces och Eigenfaces. OpenCV inkluderar även bibliotek i form av grundläggande funktionalitet, bild-video-hantering och 3D rekonstruktion med flera [11].

OpenCV_contrib
OpenCV_contrib består av extramoduler och funktioner. Nya moduler är i regel inte lika beprövade som äldre och mer väletablerade moduler. Därutav släpps dem inte som en del i den officiella OpenCV paketet. Nya moduler börjar alltid i OpenCV_contrib för att inom tid med ökad popularitet och mognad flyttas över till officiella OpenCV [12]. OpenCV_contrib har ett kodbibliotek för igenkänningsalgoritmer, därtill Eigenfaces och Fisherfaces.

Raspbian
Raspbian är ett licensfritt operativsystem baserad på Debian och är optimerad för Raspberry Pi och har stöd för över 35,000 kodbibliotek och den vanligaste mjukvaran.[13].

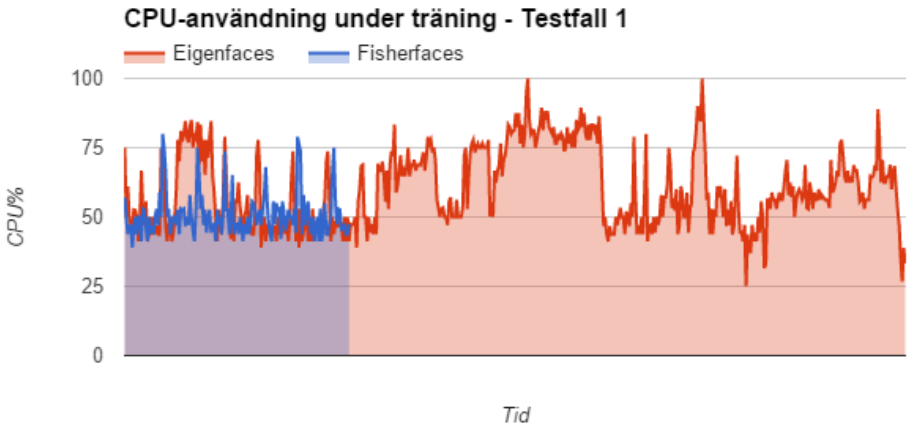
PSUTIL
PSUTIL är ett bibliotek för att få tillgång till information om process och systemkörning till exempel CPU och RAM-användning. PSUTIL är kompatibelt med både Python och Raspberry Pi 2 [14].

4 Empiri

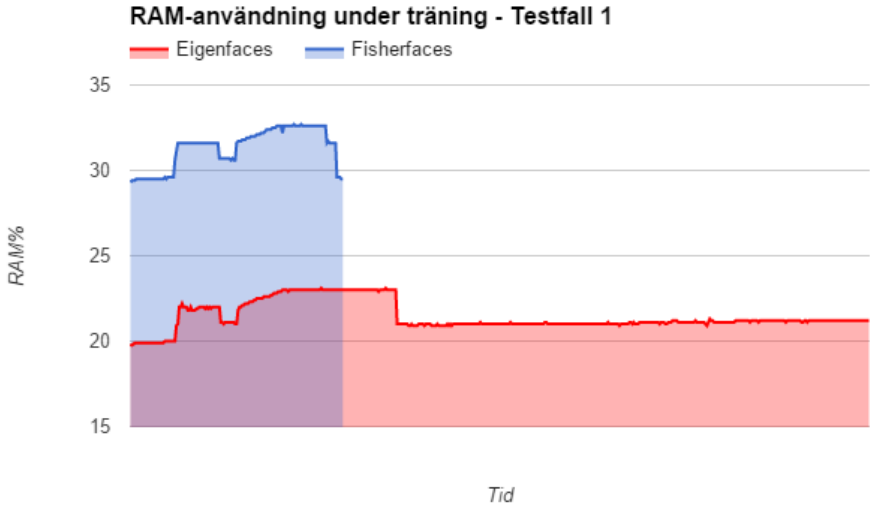
Kapitlet redovisar resultatet av testerna i form av grafer och tabeller. Mätdatan som insamlats kommer ligga till grund för studiens analys och slutsats.

4.1 Testfall 1

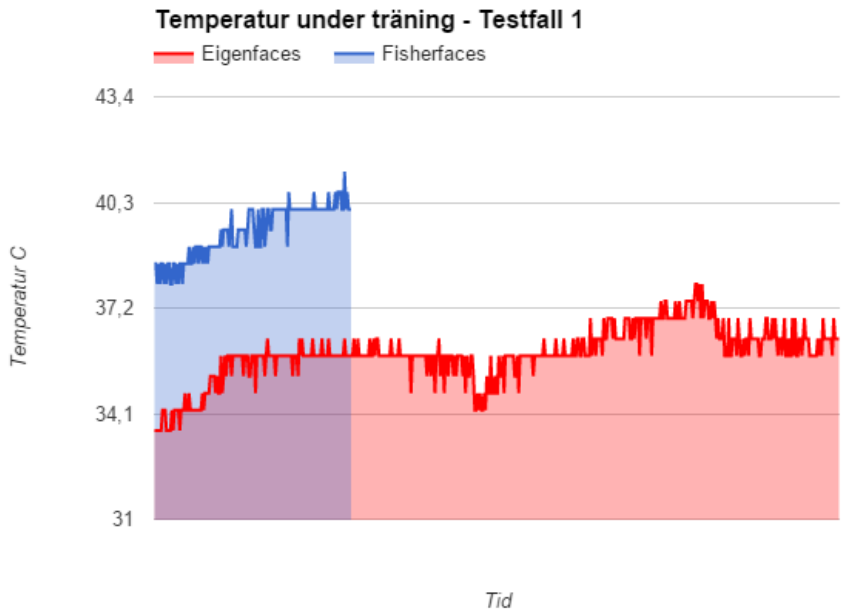
Prestanda vid träning



Figur 3: CPU-användning vid träning av 108 träningsbilder. Fisherface-metodens kurva avslutas tidigare på grund utav att metoden genomför träning och sparandet av modell snabbare än Eigenface-metoden. Detta fenomen upprepas under samtliga tester gällande träningen av modell i Testfall 1.

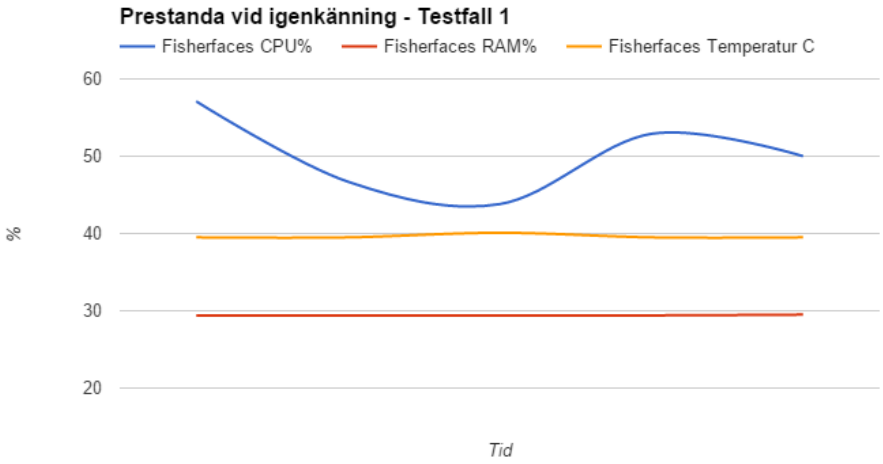


Figur 4: RAM-användning vid träning av 108 träningsbilder.

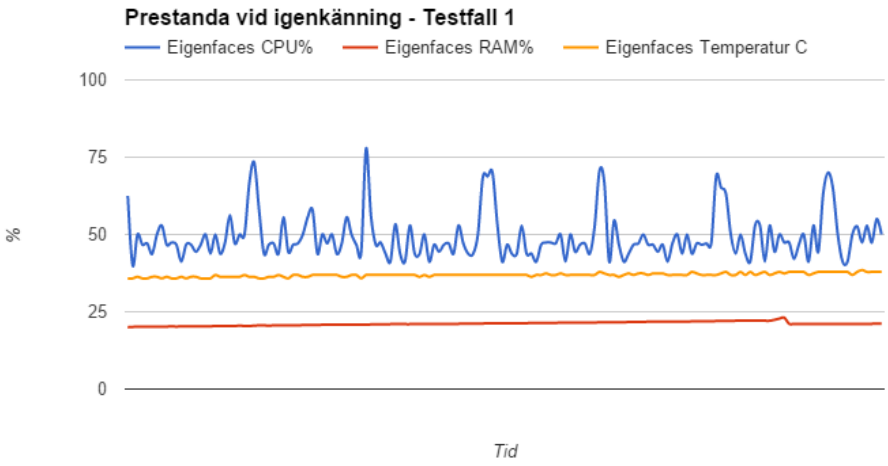


Figur 5: Temperatur vid träning av 108 träningsbilder.

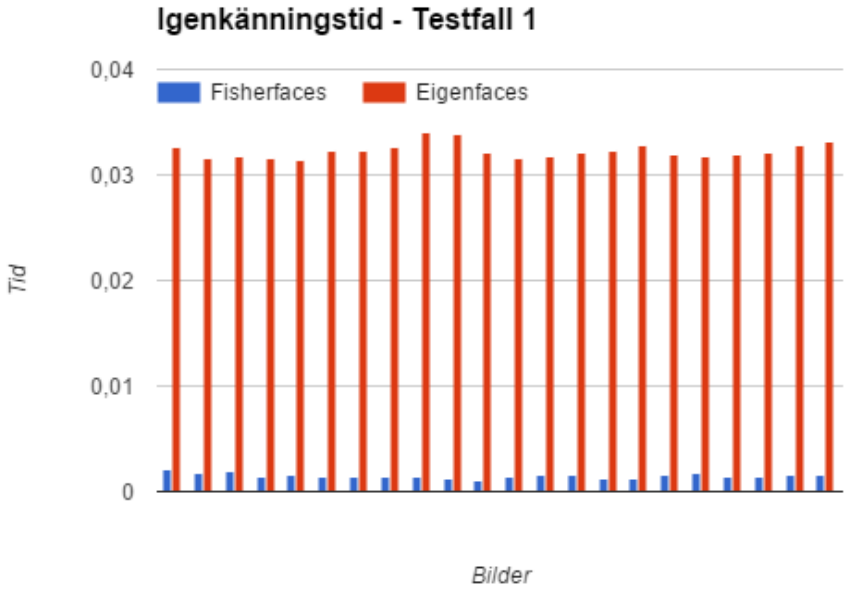
Prestanda vid igenkänning



Figur 6: Prestanda vid igenkänning av 22 testbilder.



Figur 7: Prestanda för igenkänning av 22 testbilder.



Figur 8: Tid för igenkänning av 22 testbilder. Varje stapel motsvarar en bild.

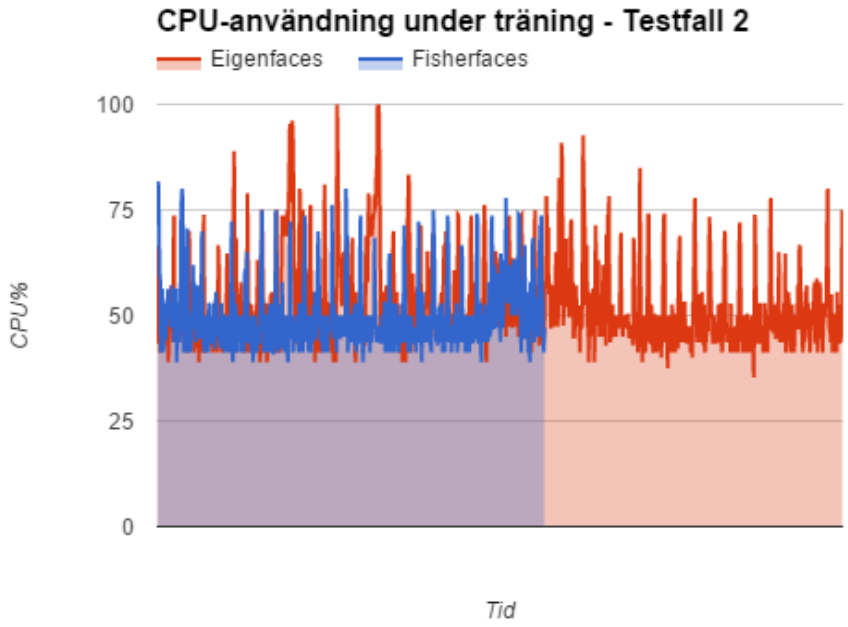
Allmän testdata

Testfall 1	Eigenfaces	Fisherfaces
Träna modellen	8,66s	6,52s
Spara modellen	14,75s	0,18s
Ladda modellen	5,9s	0,1s
Igenkännings sekvens samtliga bilder	0,88s	0,21s
Modellstorlek	27,5 MB	0,51 MB

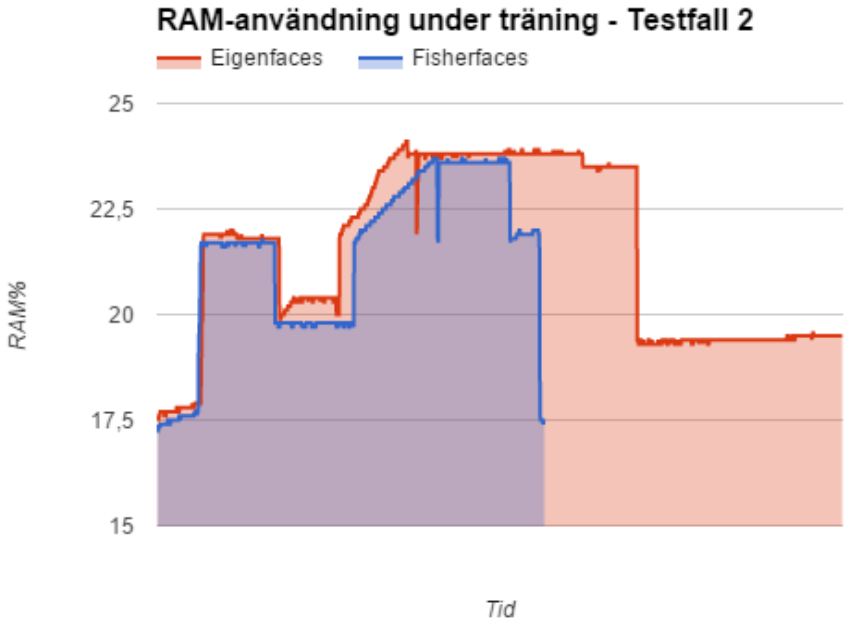
Den totala tiden för träning är uppdelad i tre delar. Dels själva träningstiden, det vill säga tiden det tar att skapa och träna modellen. De två återstående delarna består av tiden det tar att spara den färdigtränade modellen samt tiden det tar att ladda modellen till igenkänningsprogrammet. Igenkänningstiden räknar den totala tiden för att genomföra en igenkänning på samtliga testbilder. Modellstorleken motsvarar den fysiska storleken som den upptar på minneskortet.

4.2 Testfall 2

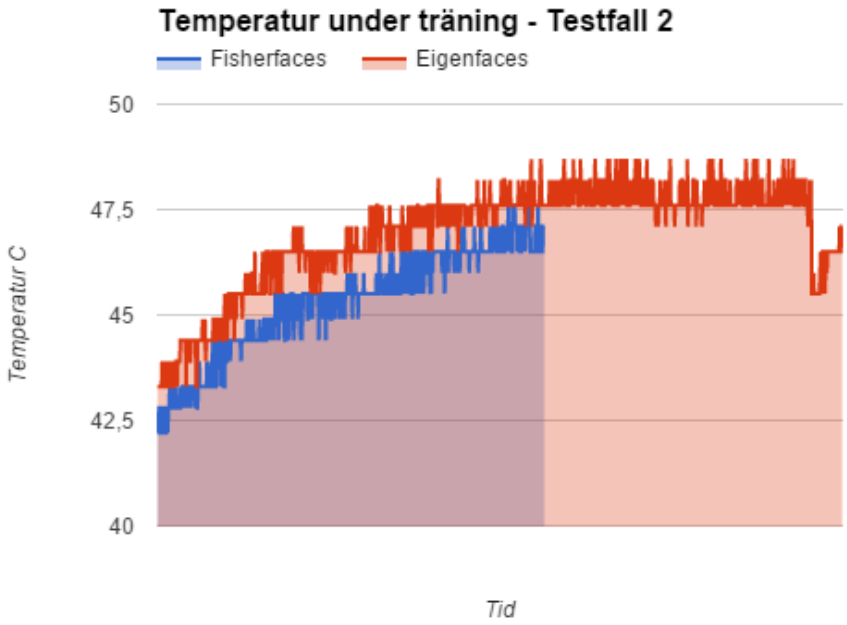
Prestanda vid träning



Figur 9: CPU-användning vid träning av 216 träningsbilder.

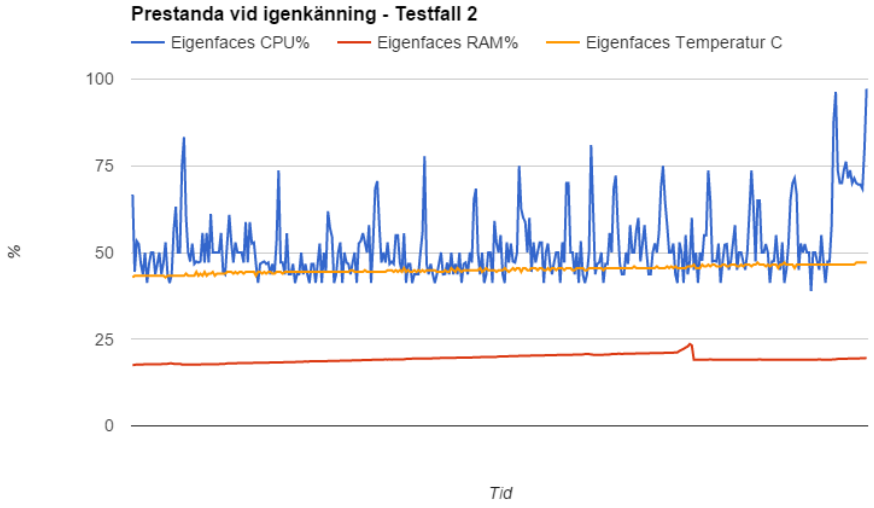


Figur 10: RAM-användning vid träning av 216 träningsbilder.

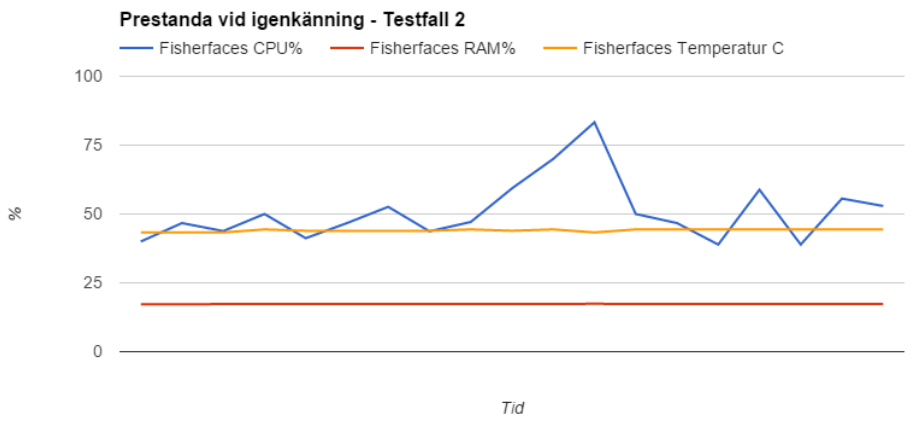


Figur 11: Temperaturen vid träning av 216 träningsbilder.

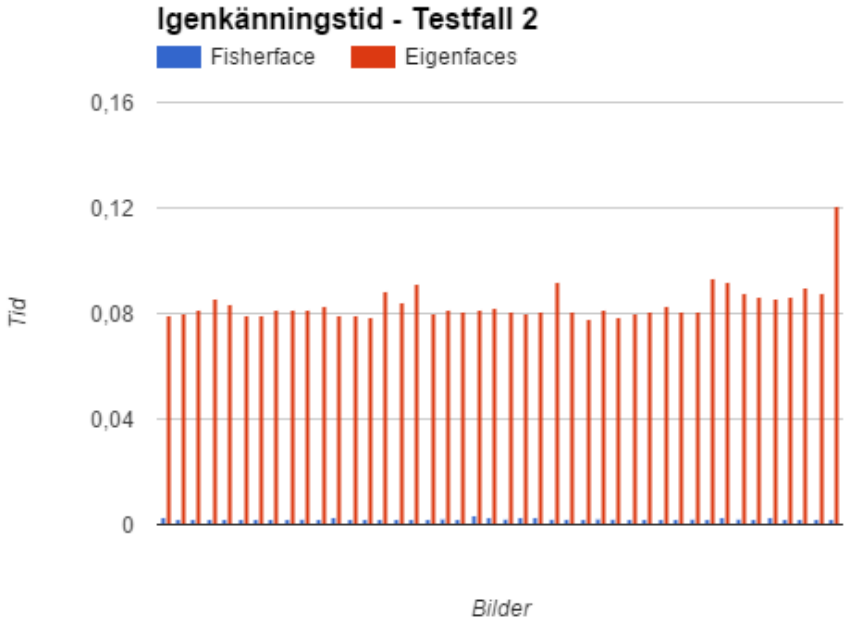
Prestanda vid igenkänning



Figur 12: Prestanda för igenkänning av 44 testbilder.



Figur 13: Prestanda för igenkänning av 44 testbilder.



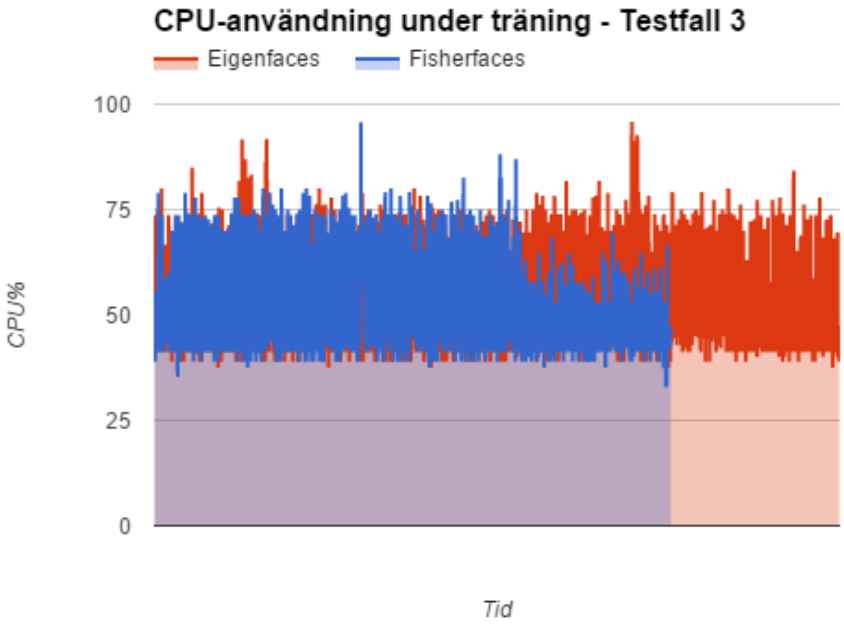
Figur 14: Tid för igenkänning av 44 testbilder. Varje stapel motsvarar en testbild.

Allmän testdata

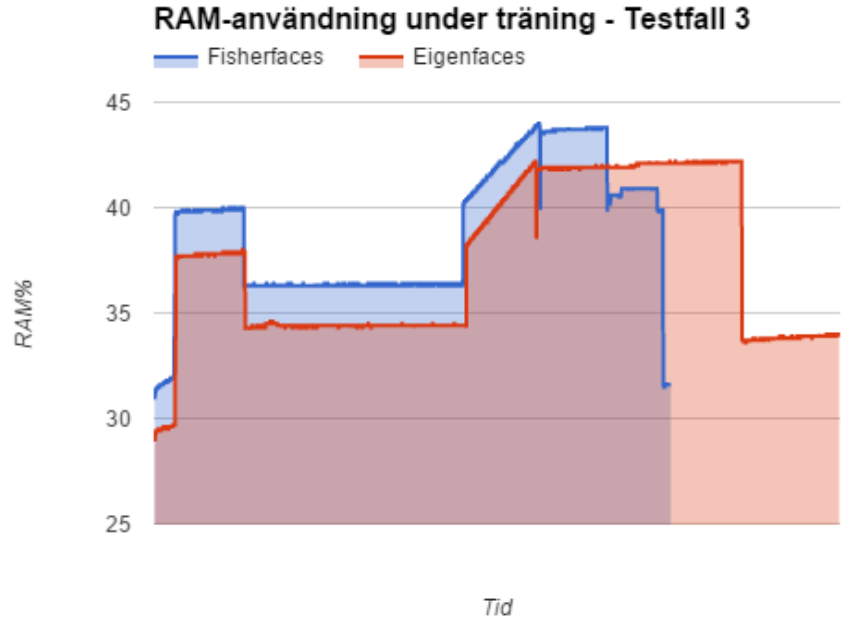
Testfall 2	Eigenfaces	Fisherfaces
Träna modellen	38,19s	27,1s
Spara modellen	18,01s	0,31s
Ladda modellen	12,06s	0,22s
Igenkännings sekvens samtliga bilder	4,34s	0,75s
Modellstorlek	55,3 MB	1 MB

4.3 Testfall 3

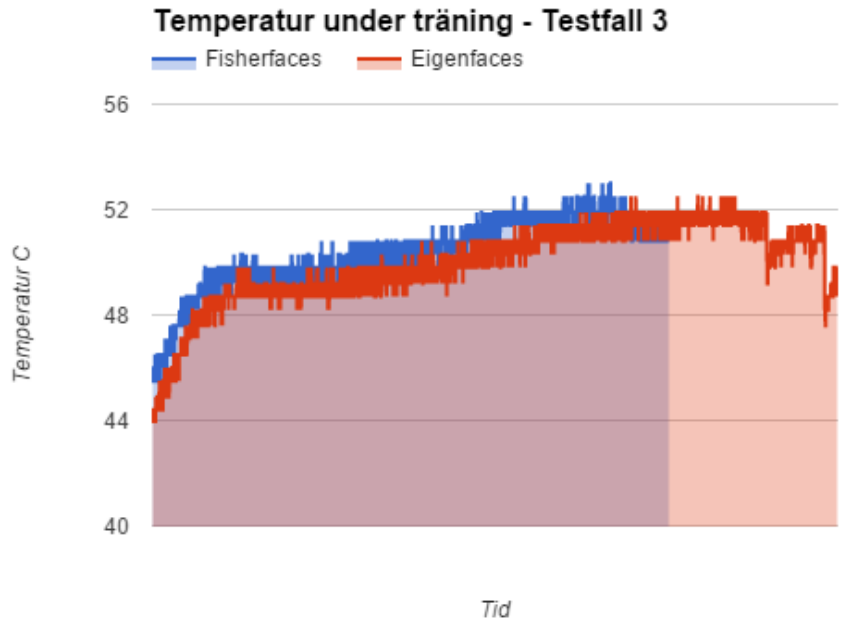
Prestanda vid träning



Figur 15: CPU-användning vid träning av 432 träningsbilder.

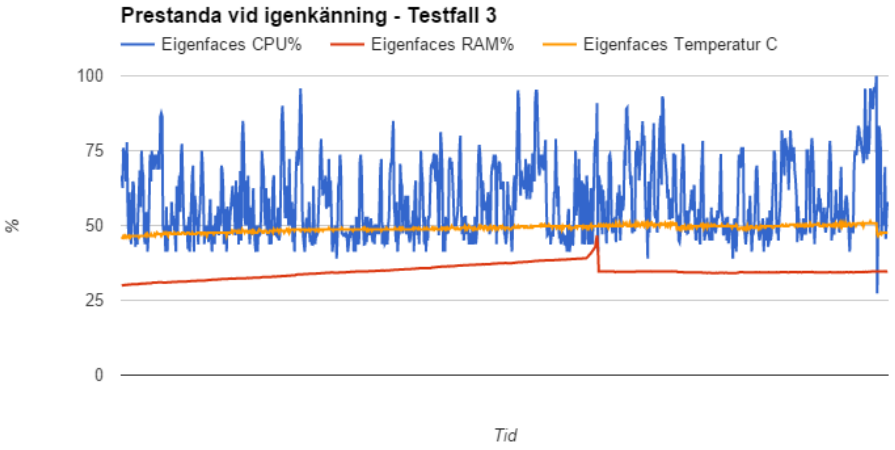


Figur 16: RAM-användning vid träning av 432 träningsbilder.



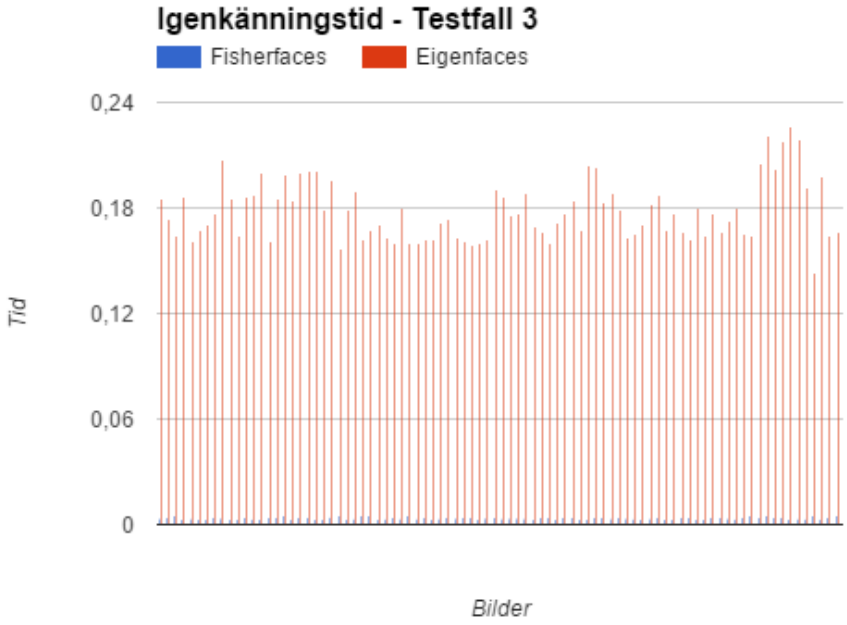
Figur 17: Temperaturen vid träning av 432 träningsbilder.

Prestanda vid igenkänning



Figur 18: Prestanda för igenkänning av 88 testbilder.

Figur 19: Prestanda för igenkänning av 88 testbilder.



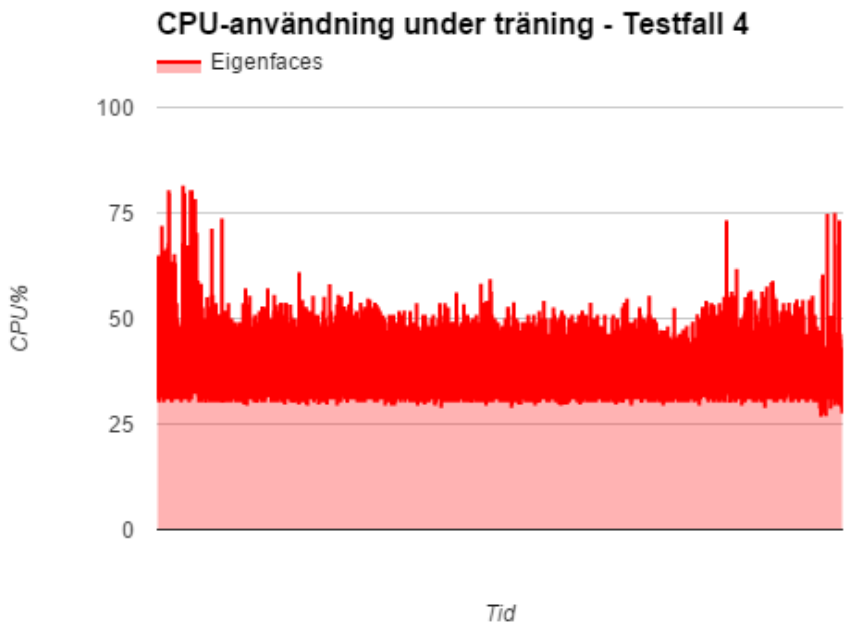
Figur 20: Tid för igenkänning av 88 testbilder. Varje stapel motsvarar en testbild.

Allmän testdata

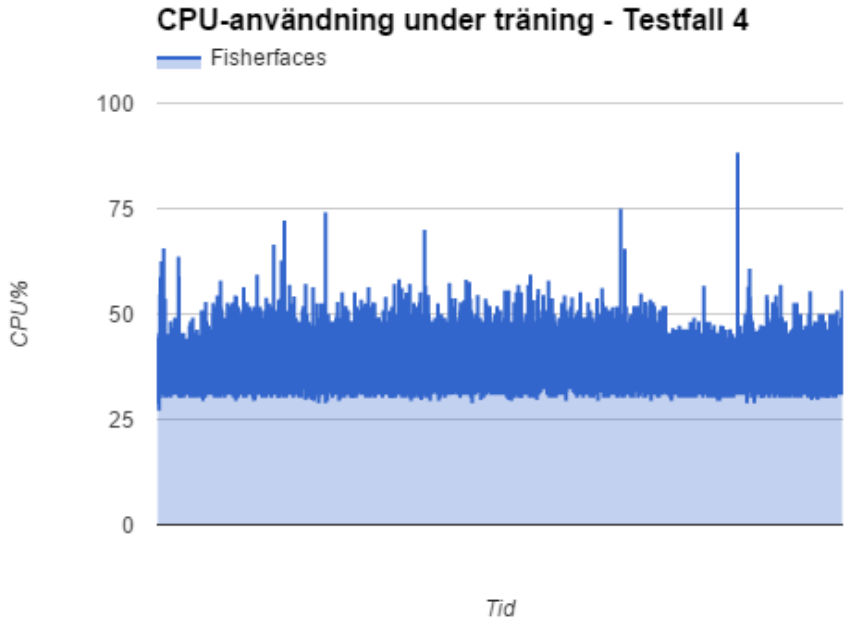
Testfall 3	Eigenfaces	Fisherfaces
Träna modellen	3,24 min	2,73 min
Spara modellen	37,94s	0,58s
Ladda modellen	25,11s	0,47s
Igenkännings sekvens samtliga bilder	31,3s	2,88s
Modellstorlek	112,8 MB	2,1 MB

4.4 Testfall 4

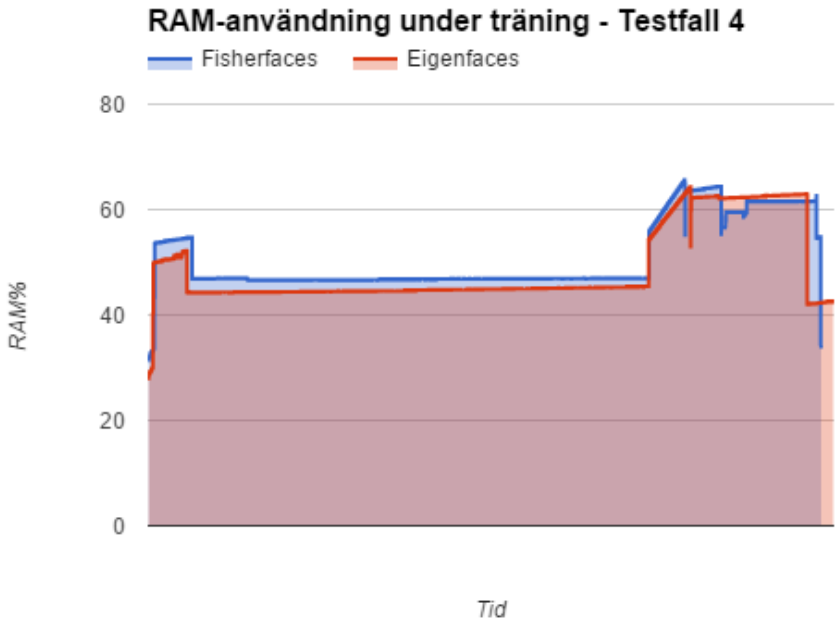
Prestanda vid träning



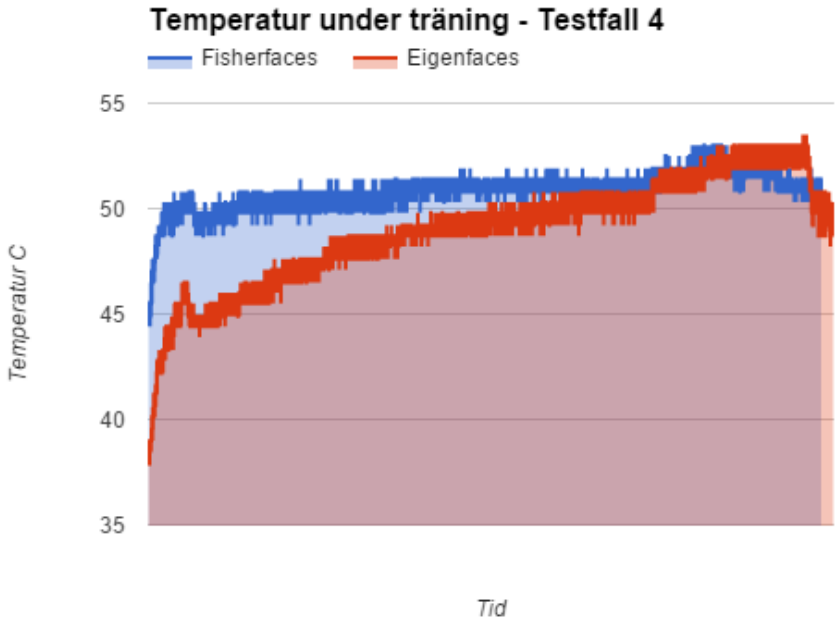
Figur 21: CPU-användning vid träning av 1080 träningsbilder.



Figur 22: CPU-användning vid träning av 1080 träningsbilder.

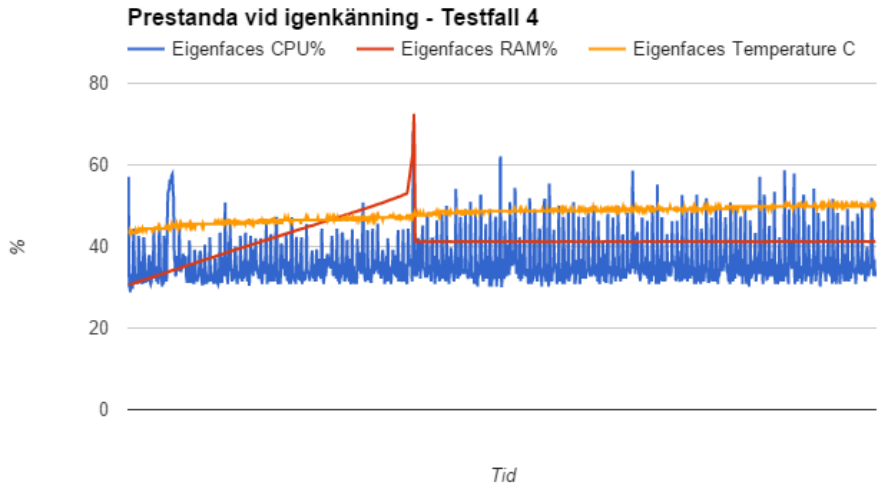


Figur 23: RAM användning vid träning av 1080 träningsbilder.

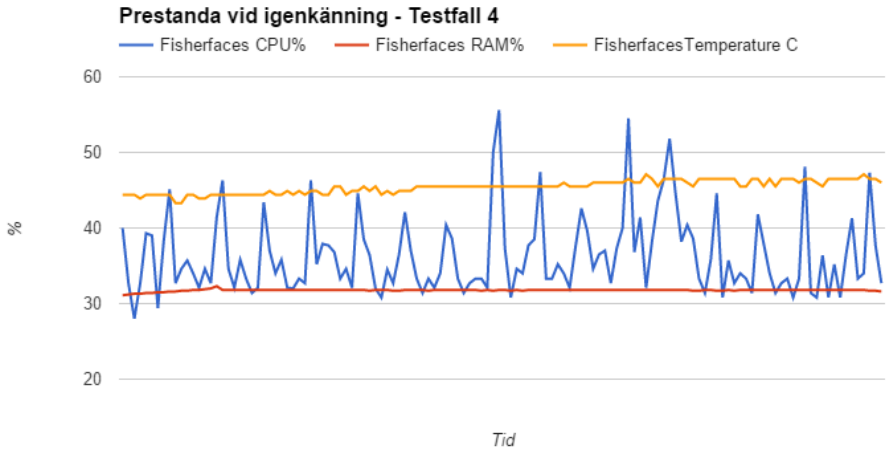


Figur 24: Temperaturen vid träning av 1080 träningsbilder.

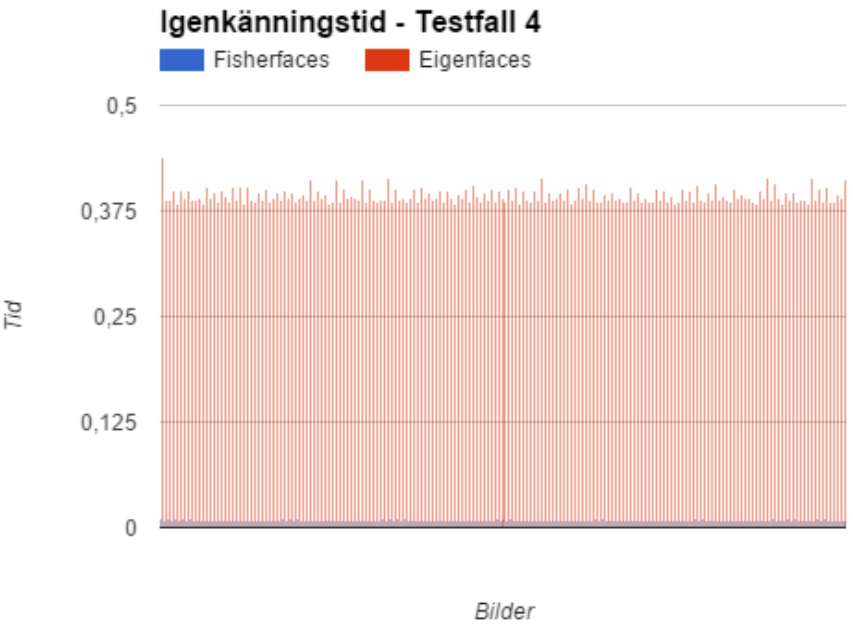
Prestanda vid igenkänning



Figur 25: Prestanda för igenkänning av 220 testbilder.



Figur 26: Prestanda för igenkänning av 220 testbilder.



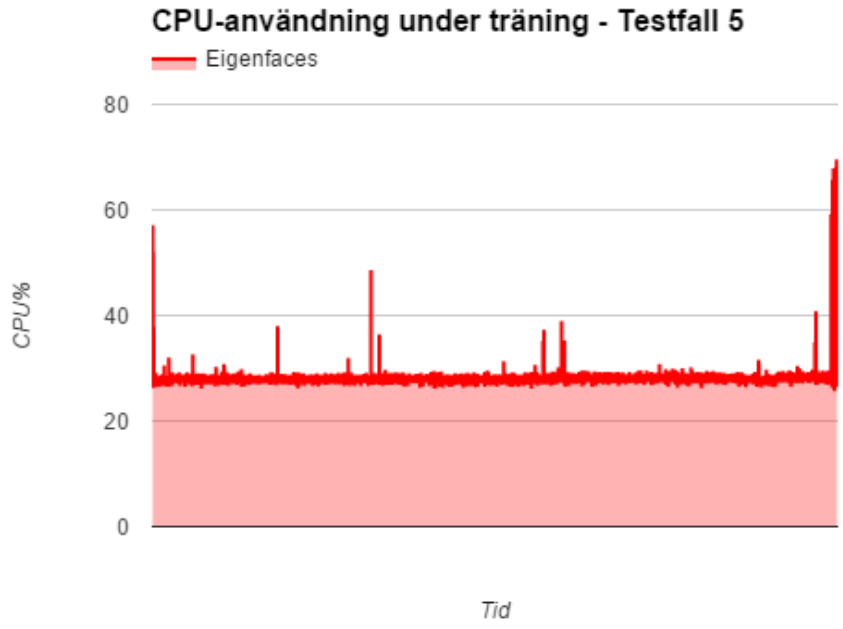
Figur 27: Tid för igenkänning av 220 testbilder. Varje stapel motsvarar en bild.

Allmän testdata

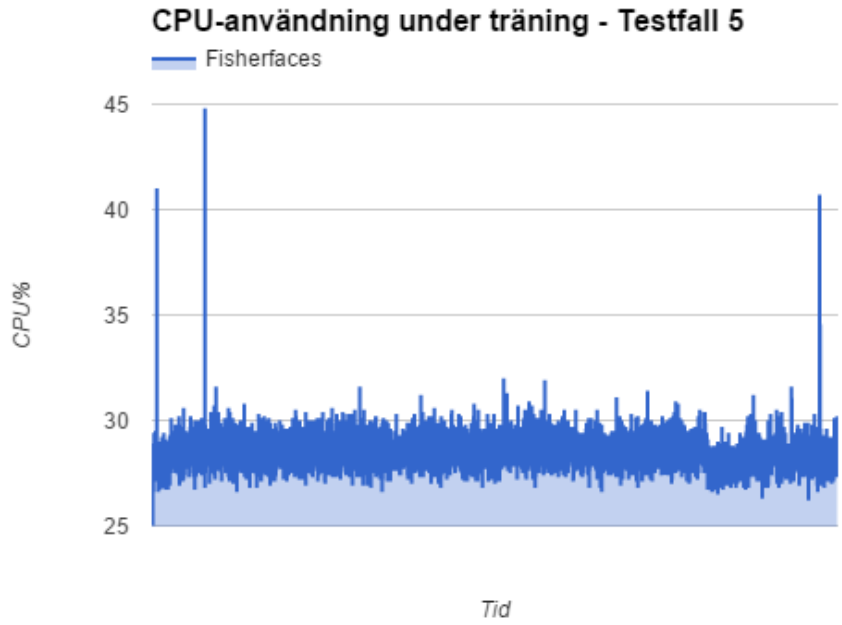
Testfall 4	Eigenfaces	Fisherfaces
Träna modellen	36,69 min	37,26 min
Spara modellen	100,09s	1,34s
Ladda modellen	59,35s	1,15s
Igenkännings sekvens samtliga bilder	99,53s	15,3s
Modellstorlek	299,4 MB	5,6 MB

4.5 Testfall 5

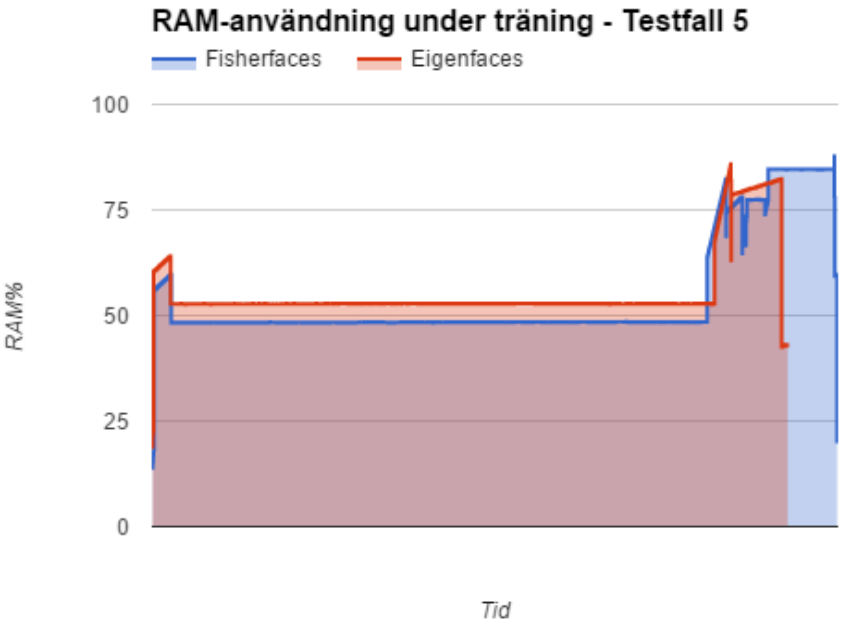
Prestanda vid träning



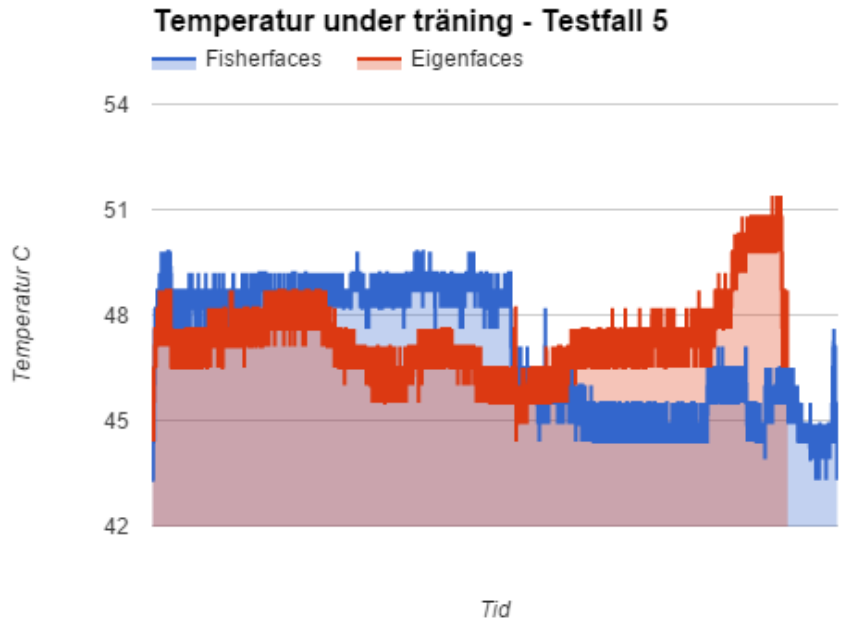
Figur 28: CPU-användning vid träning av 2052 träningsbilder.



Figur 29: CPU-användning vid träning av 2052 träningsbilder.

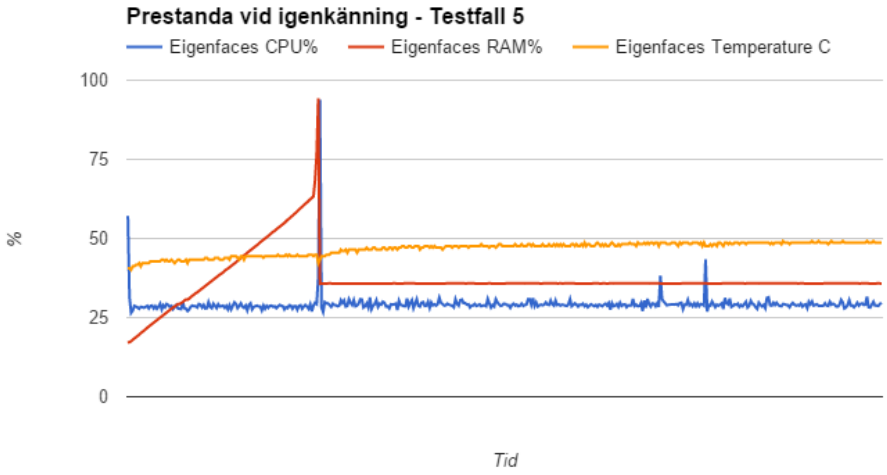


Figur 30: RAM användning vid träning av 2052 träningsbilder.

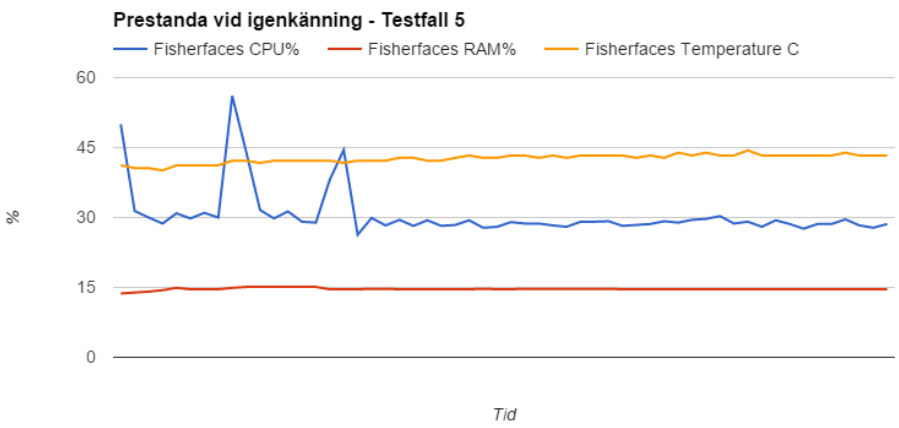


Figur 31: Temperaturen vid träning av 2052 träningsbilder.

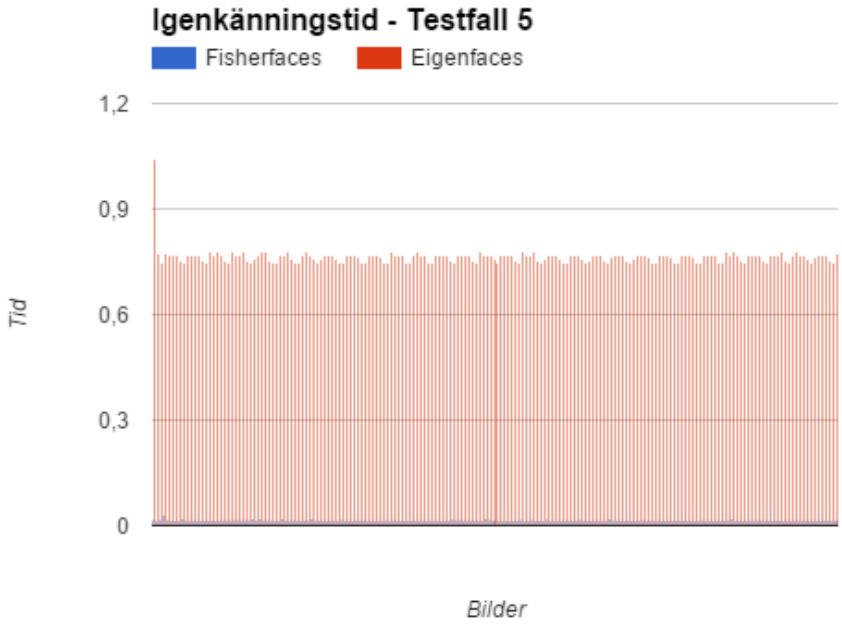
Prestanda vid igenkänning



Figur 32: Prestanda för igenkänning av 418 testbilder.



Figur 33: Prestanda för igenkänning av 418 testbilder.



Figur 34: Tid för igenkänning av 418 testbilder. Varje stapel motsvarar en bild.

Allmän testdata

Testfall 5	Eigenfaces	Fisherfaces
Träna modellen	4.36 h	4.75 h
Spara modellen	189,74s	2,59s
Ladda modellen	138,40s	2,35s
Igenkännings sekvens samtliga bilder	366,38s	54,44s
Modellstorlek	619 MB	11,6 MB

5 Analys

Kapitlet ger svar på studiens frågeställningar genom att behandla insamlad empiri och teoretiskt ramverk.

5.1 Frågeställning 1

Testerna visar att exekveringstiden för träningen är den mest centrala faktorn att ta hänsyn till för att besvara den första frågeställningen. Resultatet från testerna visar att det är möjligt att genomföra träningen lokalt på Raspberry Pi 2 men när modellen innehåller 2052 bilder tar träningen närmare 5 timmar. Testerna visar att det tar tre minuter för Eigenface-modellen bestående av 2052 bilder att laddas in i systemet. Det finns möjlighet att träna modellen externt på en kraftfullare hårdvara för att sedan ladda in modellen lokalt på Raspberry Pi 2. Denna metod erbjuder en mer tidseffektiv lösning men det förutsätter att det finns en tillräckligt kraftfull extern hårdvara att tillgå.

Vid jämförelse av de två största testfallen rörande storlek på modellerna är storleken på Fisherface-metodens modell 5,6 MB respektive 11,6 MB och för Eigenfaces-metoden 299,4 MB respektive 619 MB. Det innebär att vid en fördubbling av bilder fördubblas även storleken på modellerna. Vid en närmare undersökning av resterande testfall fastställs att storleken förändras linjärt.

Vid ett försök att träna båda metodernas modeller med 2470 bilder uppkom ett kompileringsfel som hänvisade till att det inte fanns tillräckligt med RAM-minne att tillgå. Dock kunde båda metoderna genomföra sin träning utan felmeddelanden eller varningar vid Testfall 5. Detta indikerar att gränsvärdet gällande antalet bilder som modellen kan innehålla för att uppnå testkraven ligger mellan 2052 och 2470 bilder. Figur 30 visar att metoderna i slutfasen uppnår 80 procent av RAM-minnet. Detta stödjer indikationerna om att gränsvärdet för att träna lokalt är mellan 2052 och 2470. Detta medför att RAM-minnet är en mer avgörande faktor än storleken på modellen framförallt för Eigenface-metoden vars modellstorlek i sammanhanget är utrymmeskrävande.

Det finns en viss svårighet att mäta processorns verkningsgrad då operativsystemet använder bakgrundsprocesser parallellt som testscripten och programkoden exekveras. Dessa faktorer är svåra att påverka och kan medföra att CPU-värdena får plötsliga spikar. Även om detta kan förekomma påverkar det inte resultatet märkbart och systemet begränsas mer av storleken på RAM-minnet än av processorns kapacitet. Figur 28-29 visar att processoranvändningen är förhållandevis låg och det är endast vid ett fåtal tillfällen som processoranvändningen närmar sig 50-70 procent. Processorns arbetsbelastning är inom samma intervall under de första fyra testfallen och påverkas därmed inte märkbart av hur många bilder som är i modellen.´

Testfall 1-5 visar att temperaturen endast varierar med några grader och det är ingen märkbar temperaturskillnad mellan de två metoderna. Temperatur överstiger aldrig 55 grader Celsius och bör därför inte påverka hårdvaran i sådan utsträckning att det blir en väsentlig faktor att ta hänsyn till.

Resultaten från testerna visar att hårdvarans arbetsbelastning är lägre under igenkänningsfasen jämfört med träningsfasen. Resultatet visar på låga värden på samtliga parametrar och i enighet med resultatet finns det inga begränsningar för att använda Raspberry Pi 2 för igenkänningen.

5.2 Frågeställning 2

Resultaten från testerna rörande att spara och ladda modellen påvisar att Fisherface-metoden är snabbare än Eigenface-metoden. Vad som i huvudsak påverkar tidsskillnaderna är Eigenface-metodens modellstorlek. I testfall 5 har Fisherface-modellen en storlek på 11.6 mb och Eigenface-modellen är 619 mb stor, vilket är en differens på cirka 610 mb. Det innebär att modellen som ska sparas och laddas är större för Eigenface-metoden och är därutav mer tidskrävande att behandla.
Dessa skillnader är genomgående för alla testfall dock är den tydligast i testfall 5. Mätdatan visar att det tar cirka 3 minuter för Eigenface-metoden att spara den skapade modellen jämfört med Fisherfaces som genomför träning på 2.5 sekunder. Vid laddning av modellen tar det cirka 2.5 minuter för Eigenfaces-metoden jämfört med Fisherface-metodens 2 sekunder.

Vid en jämförelse av modell-träningstiderna för de två första testfallen skiljer det endast någon sekund mellan de båda metoderna varav Fisherface-metoden är mer effektiv. Tabell 3 visar att Fisherface-metoden genomför träningen en halv minut snabbare än Eigenface-metoden. Vid testfall 4 har tiden för

träningen ökat till dryga 37 min för de båda metoderna. Märkbart är att Eigenface-metoden genomför träningen snabbare än Fisherface-metoden och likaså vid testfall 5. Tabell 5 visar att tidsskillnaden har ökat till cirka (12.5 minuter) där Eigenface-metoden genomför träningen på 4.36 timmar och Fisherface-metoden på 4.75 timmar.

Figur 28-29 visar att processorn under träningen har en verkningsgrad på cirka 25 procent med vissa spikvärden upp mot 75 procent. I testfall 1-4 är Fisherfaces och Eigenfaces CPU-användning likvärdiga utan några märkbara skillnader. Testfall 5 uppvisar dock en lägre och jämnare processor-användning. En möjlig orsak till detta är att mätdatan sparas med ett längre intervall och på så vis inte förekommer lika många höga värden.

Figur 30 visar att metoderna initialt använder 50 procent av RAM-kapaciteten. I slutet av testet sker en ökning upp mot 80 procent. Testfall 1-4 visar att RAM-användningen är förhållandevis likvärdig mellan metoderna dock till Fisherface-metodens fördel.

6 Diskussion och slutsatser

Kapitlet ger en sammanfattande beskrivning av studiens resultat. Vidare beskrivs studiens implikationer och begränsningar. Dessutom beskrivs studiens slutsatser och rekommendationer. Kapitlet avslutas med förslag på vidare forskning.

6.1 Resultat

Resultatet från analysen visar att Raspberry Pi 2 lämpar sig som hårdvara för ett igenkänningssystem dock med vissa begränsningar rörande träning. Träningstiden vid 2052 bilder uppgår till dryga 4.5 timmar som i praktiken inte kan anses rimlig. Vidare begränsas träningen av RAM-minnets storlek vid 2470 bilder i modellen då RAM-minnet tar slut. Därför är det lämpligt att använda sig av en kraftfullare hårdvara som utför träningen externt för att sedan ladda in modellen i systemet. Om det ändå är önskvärt att träna lokalt är det lämpligt att göra när tidsaspekten inte är ett hinder eller när bild antalet i modellen är färre. När modellen är tränad och igenkänningsfasen inleds kan man konstatera att Raspberry Pi 2 klarar av testerna med stor marginal och uppvisar inga hinder att vara hårdvara i ett igenkänningssystem.

Analysen pekar på tydliga skillnader kring de två metoderna. Kapitlet redogör att dessa skillnader i första hand utgörs av exekveringstid och storlek på modell.

6.2 Implikationer

Det finns ett tydligt samband mellan ökning av bilder och verkningsgraden av de olika parametrarna. Antalet bilder påverkar tiden för träning, igenkänning, sparandet och inladdning av modellen. Vidare finns det en implikation mellan mängden bilder och storlek på modell, det innebär att storleken på modellen ökar vid användandet av flera bilder.

6.3 Begränsningar

En påtaglig begränsning av Raspberry Pi 2 är förmågan att träna lokalt. Vid testfall 5 genomför metoderna träningen på dryga 4.5 timmar.

RAM-minnets kapacitet begränsar bild antalet för träning till ett maxantal mellan 2052 och 2470.

6.4 Slutsatser och rekommendationer

Det är möjligt att använda Raspberry Pi 2 för både träning och igenkänning, dock med begränsningar kring träningsdelen. Med anledning av dessa begränsningar är rekommendationen att använda en extern hårdvara för att träna modellen och istället endast använda Raspberry Pi 2 för igenkänningsdelen.

Fisherface-metoden är snabbare, har mindre modellstorlek och är mindre krävande än Eigenface-metoden. Eigenfaces-metoden tränar snabbare än Fisherface-metoden vid testfall 4 och 5. Trots detta anser studieförfattarna att Fisherface-metoden är det lämpligare valet av de båda metoderna.

6.5 Vidare forskning

Testerna genomfördes med en bildupplösning på 92*112. Studier med fokus på prestanda vid högre upplösning vore av värde eftersom denna studie inte undersöker denna parameter närmare. Vidare ligger igenkänning via en videosekvens nära studiens fokusområde och är därmed ett lämpligt område att utforska ytterligare.

Referenser

[1] D. Georgescu ”A Real-Time Face Recognition System Using Eigenfaces”, *Journal of Mobile*, vol.3, nr 4, s.193-204, 2011

[2] M. Grgic and K. Delac, “Algorithms”
M. Grgic and K. Delac [Online] Tillgänglig:
http://www.face-rec.org/algorithms/ [Hämtad: 28:e februari, 2016]

[3] Raspberry Pi Foundation “FIVE MILLION SOLD!“
Raspberry Pi Foundation [Online] Tillgänglig:
https://www.raspberrypi.org/blog/five-million-sold/ [Hämtad: 28:e februari, 2016]

[4] M. Turk, A. Pentland, ”Eigenfaces for Recognition”, *Journal of Cognitive Neuroscience*, vol.3, nr 1, s.71-86, 1991

[5] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection”, IEEE *TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, VOL. 19, NO. 7, JULY 1997

[6] K. Naik, P. Tripathy,”Software Testing and Quality Assurance”, *Theory and Practice*, New Jersey: John Wiley & Sons, 2008

[7] G. Lanaro,”Python High Performance programming”, *Boost the performance of Python programs using advanced techniques*, Birmingham: Packt Publishing, 2013

[8] OpenCV dev team “Facerec Tutorial”
OpenCV dev team. Senast uppdaterad 9 februari 2016 [Online] Tillgänglig:
http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html
[Hämtad: 9 februari, 2016]

[9] Raspberry Pi Foundation, "What is raspberry pi?"
Raspberry Pi Foundation, [Online] Tillgänglig:
https://www.raspberrypi.org/help/what-is-a-raspberry-pi/ [Hämtad: 13 oktober, 2015].

[10] Python Software Foundation, "General Python FAQ"
Python Software Foundation, Senast uppdaterad 12 oktober 2015 [Online] Tillgänglig:
https://docs.python.org/3/faq/general.html#what-is-python/ [Hämtad: 13 oktober, 2015].

[11] OpenCV dev team, "OpenCV API Reference"
OpenCV dev team, Senast uppdaterad 25 februari 2015 [Online] Tillgänglig:
http://docs.opencv.org/modules/core/doc/intro.html. [Hämtad: 13 oktober, 2015].

[12] OpenCV dev team, “OpenCV_contrib repository”
OpenCV dev team. Senast uppdaterad: 1 februari 2016 [Online] Tillgänglig:
https://github.com/Itseez/opencv_contrib [Hämtad: 9 februari, 2016]

[13] Raspberry Pi Foundation “Raspbian”
Raspberry Pi Foundation [Online]Tillgänglig:
https://www.raspbian.org/ [Hämtad: 9 februari, 2016]

[14] Python Software Foundation “Psutil”
Python Software Foundation [Online] Tillgänglig:
https://pypi.python.org/pypi/psutil [Hämtad: 9 februari, 2016]

Ordlista

Träningsmodellen: - Modellen - Tränaren: skapar en modell som består av vektor data. En så kallad support vector matrix (fortsättningsvis SVM). Det är denna modell som sedan används för att kunna känna igen ansikten som finns tränade i modellen.

Eigenface-metoden: - Eigenfaces - Eigenface-algoritmen - Innefattar både träning och igenkänningsdelen

Fisherface-metoden: - Fisherfaces - Fisherface-algoritmen - Innefattar både träning och igenkänningsdelen

Bilagor

Bilaga 1 Länk till programkod

https://drive.google.com/folderview?id=0B9fU6BcbSUOQVmgzOXZfTk1QaVU&usp=sharing