UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Electrònica Industrial i Automàtica

IDENTIFICACIÓ D'ORIENTACIÓ DE ROBA PER A LA DETECCIÓ DE PECES DE DINS CAP A FORA PER MANIPULACIÓ AMB ROBOTS

Memòria i Annexos

Autor: Pau Solsona Boada
Director: Attila Peter Husar
Co-Director: Pablo Jiménez Schlegl

Convocatòria: Juny 2022

Resum

La robòtica d'assistència domèstica inclou un gran nombre de problemàtiques que requereixen una solució per a atendre, amb la màxima flexibilitat, a l'usuari. Un robot ha de poder ser capaç de tractar i manipular un gran nombre d'objectes dins d'una casa, ja siguin rígids o deformables

Entre aquests objectes habituals en una llar hi ha les peces de roba, les quals presenten una alta complexitat de categorització a causa de la dificultat en modelitzar els possibles estats en els que es poden trobar (potencialment infinits degut a la seva deformabilitat) per a tasques com ara guardar la roba, plegar, allisar o simplement transportar d'un punt a l'altre.

Aquest treball presenta una metodologia per a discernir, a partir d'una imatge, l'existència d'un pantaló, i la seva classificació en quatre estats preestablerts, separant aquests entre pantalons que es troben completament de dins cap a fora o amb el costat correcte cap enfora, i determinant si la peça es troba cap per amunt o cap per avall.

Els resultats obtinguts demostren un mètode eficaç i ràpid per a la seva classificació, observant les diferents problemàtiques i aportant un conjunt de solucions i millores aplicables per a treballs futurs.

Resumen

La robótica de asistencia doméstica conlleva un gran número de problemáticas que requieren una solución para atender, con la máxima flexibilidad, al usuario. Un robot debe poder ser capaz de tratar y manipular un gran número de objetos dentro de una casa, ya sean rígidos o deformables.

Entre estos objetos se encuentran las prendas de vestir, las cuales presentan una alta complejidad de categorización debido a la dificultad en modelizar los posibles estados (potencialmente infinitos debido a su deformabilidad) en los que se pueden encontrar para tareas como guardar la ropa, plegar, alisar o simplemente transportarla de un punto a otro.

Este trabajo presenta una metodología para discernir, a partir de una imagen, la existencia de un pantalón, y su clasificación en cuatro estados preestablecidos, separando estos entre pantalones que se encuentran completamente de dentro hacia fuera o con el lado correcto, exterior, hacia afuera, y determinando si la pieza se encuentra boca arriba o boca abajo.

Los resultados obtenidos demuestran un método eficaz y rápido para su clasificación, observando las diferentes problemáticas y aportando un conjunto de soluciones y mejoras aplicables para trabajos futuros.

Abstract

Assistive robots address a large number of problems that require a solution to help, with the maximum flexibility, the user. A robot must be able to treat and manipulate a large number of objects inside a house, whether they are rigid or deformable.

Among such objects there are garments, which have a high complexity of categorization due to the difficulty to model the possible states (potentially infinite, due to their deformability) in which they can be found for tasks such as storing clothes, folding, smoothing, or simply transporting them from one point to the other.

This work presents a methodology to discern, from an image, the existence of pants, and their classification in four pre-established states, separating these between pants that are completely inside-out or right-side out and determining whether the pieces are facing upwards or downwards.

The results obtained demonstrate an effective and fast method for classification, observing the different problems, and providing a set of solutions and improvements applicable for future work.

Agraïments

Aquest projecte no hauria estat possible sense l'ajuda incessant de Pablo Jiménez, codirector del treball i orientador en la seva realització, que m'ha ajudat en tots aquells moments que ho he necessitat i més, i al tutor del treball, Attila Husar pel seguiment que ha fet del treball i les seves aportacions en els aspectes més formals.

A més a més, vull agrair l'ajuda de l'Institut de Robòtica i Informàtica per a solucionar els diferents problemes amb els quals m'he anat trobant durant l'inici del desenvolupament.

També vull agrair el meu company d'universitat, Albert Pla, el qual ha mantingut un interès constant dels avenços obtinguts i m'ha ajudat a tirar endavant el treball en aquells moments de baixa motivació i cansament acadèmic.

Finalment, vull agrair a la meva família, pel seu suport incondicional i constant durant la realització del treball, que sense ella no em trobaria en aquesta etapa dels meus estudis.

Índex

Taula de figures

1. Prefaci

1.1. Origen del treball

Observant les diferents feines domèstiques que es duen a terme dins d'una casa, la manipulació de roba es troba molt present entre les tasques demandades. Amb la intenció d'automatitzar processos com ara la identificació, plegar o guardar les diferents peces de roba, és molt important considerar els treballs previs de recerca sobre la robotització d'aquestes tasques, analitzant els diferents mètodes aplicats i avaluar la possible aplicació de noves tècniques d'aparició més recent.

Un dels problemes més complexos en la manipulació automatitzada de roba es el d'identificar parts d'una peça de roba que han estat girats de dins cap enfora, i revertir la situació mitjançant la manipulació amb un robot.

La recerca bibliogràfica realitzada demostra com s'han dedicat pocs esforços per a resoldre aquest problema, i l'única referència que hi ha és l'article *"Perception for the manipulation of socks"* (1), completament enfocada als mitjons. Existeix una gran dificultat en extrapolar i reproduir resultats similars al document citat, en altres tipus de peces de roba. Aquesta dificultat resideix en la complexitat de classificació de les diferents configuracions en les quals una peça es pot definir i parametritzar.

El grup de Percepció i Manipulació de l'Institut de Robòtica i Informàtica ha identificat la manipulació de roba com una de les qüestions clau en la robòtica assistencial, tant per la seva rellevància com per la seva dificultat, i és en aquest context que s'ha plantejat l'afer de restablir l'orientació (interior-exterior) correcta de la roba

En col·laboració amb l'Institut, es planteja i defineix els objectius d'aquest treball, tot ajustant la quantitat de recursos requerida per a complir-los, anivellant-los a aquells disponibles per un TFG.

1.2. Motivació

Existeix una gran motivació per a avançar en la investigació i recerca de possibles solucions relacionades amb l'àmbit de la roba i la seva manipulació. Tal com es comenta anteriorment, és molt habitual trobar-se peces de roba que requereixen manipulació dins de la llar, i també és molt freqüent trobar-se aquestes peces en una configuració parcial o totalment revertida.

Tal i com succeeix amb altres aplicacions, com ara el plegat, l'allisat, l'assistència en el vestir o desvestir una persona, és imprescindible determinar l'estat de la roba mitjançant l'observació, per així extreure informació rellevant de la peça, i poder manipular-la amb facilitat.

A més a més, contribuir a la resolució del problema en qüestió ens permet avançar en la investigació i millores per a l'evolució de la automatització de les activitats domèstiques de la llar, obrint portes a nous problemes i avanços.

1.3. Requeriments previs

L'assoliment de l'objectiu d'aquest treball requereix uns certs coneixements de *Machine Learning* i dels diferents algoritmes emprats en aquesta disciplina. Amb la idea de fer una memòria autocontinguda, es descriuen breument els diferents models d'aprenentatge emprats, el seu funcionament i aplicació, així com els motius per seleccionar-ne un o un altre.

La majoria dels algorismes d'aprenentatge estan implementats en el llenguatge de programació Python, així que per entendre el codi associat a les referències consultades, el seu funcionament i possibilitats de millora, és imprescindible conèixer aquest llenguatge. La implementació d'aquest treball ha estat utilitzant Python, tal i com es mostra als annexos.

2. Introducció

Existeix un gran nombre d'aplicacions per a la robòtica en molts àmbits diversos. No obstant, encara hi ha sectors en què la robòtica no s'ha desenvolupat prou, tot i oferir un gran potencial i grans beneficis en la seva aplicació. Ja sigui per la complexitat que aporta la modelització d'algoritmes que funcionin en un nombre il·limitat de situacions, o la dificultat d'implementació d'aquests, l'assistència a la llar conté un gran nombre de problemàtiques que poden ser resoltes mitjançat l'aplicació de la robòtica.

En situacions on l'usuari es troba inhabilitat o amb dificultats de desplaçament, manipulació o percepció, és interessant observar quines accions es poden dur a terme per a millorar l'experiència de vida i assistir-lo en aquelles situacions que duu a terme diàriament. L'ús de la robòtica permet obrir tot un món de possibilitats, com ara aquella de la qual es parlarà en aquest treball.

L'assistència en la llar és un tema d'interès des dels inicis de la robòtica, i un d'aquells camps que encara requereixen investigació ha estat, i és, la manipulació de roba. La dificultat essencial que l'envolta prové principalment de l'alta complexitat en definir una parametrització que sigui capaç de definir clarament l'estat de la roba.

Una peça de roba es pot trobar en una infinita quantitat de configuracions, degut a la seva deformabilitat, com per exemple plegada, embolicada, del dret, del revés o qualsevol estat entremig o combinació d'estats. Es pot afirmar doncs que l'estat d'una peça de roba es troba definit en un espai de dimensionalitat infinita. És per aquest motiu, que la simple identificació i classificació de la peça en una configuració arbitrària, posant-la en categories com ara pantalons, samarretes, o mitjons és complexa per ella mateixa. I no només això, sinó que un cop determinat el tipus de peça, cal fer un segon anàlisi per a determinar en quina posició, orientació o escala es troba cada una de les parts que la conformen.

Aquest treball pretén contribuir a determinar quin és l'estat d'una peça de roba utilitzant visió per computador, tot imposant certes assumpcions i requeriments per a simplificar el problema i aportar una investigació rellevant.

2.1. Objectius del treball

Els objectius mencionats a continuació són els que s'han establert tan bon punt s'ha finalitzat l'anàlisi i recerca dels treballs previs. Alguns d'aquests objectius s'han complert sense problemes, mentre que altres requereixen explicació per a poder entendre l'assoliment parcial.

Un cop realitzada la recerca s'ha determinat que el treball quedaria repartit en dos objectius principals, inicialment independents, i un cop completats s'integraran en una mateixa línia de funcionament. Aquests són:

- Detecció i classificació de diferents peces de roba per a seleccionar aquelles que pertanyin a la categoria de Pantalons.
- Detecció i classificació dels pantalons en quatre estats diferents, del dret cap amunt, del dret cap avall, del revés cap amunt, del revés cap avall.

En el **Capítol 3 – Assumpcions** s'hi troba una explicació més exhaustiva dels diferents paràmetres a detectar, classificar, definint cada un dels objectius d'una manera més comprensiva i argumentada.

3. Assumpcions

De la mateixa manera que s'han determinat els objectius d'aquest treball, és important establir certes assumpcions que es tindran en compte per a poder realitzar la investigació. En cas contrari, la complexitat de les condicions inicials provocarien una gran dificultat per a determinar com començar el treball, per on avançar, i com millorar aquells aspectes que en requereixen de canvis sense sobrepassar els recursos disponibles.

Les següents condicions ajuden a enquadrar i delimitar els límits del treball, de la mateixa manera que ens demostren i ajuden a determinar de quines maneres es pot millorar el treball. A més a més, mitjançant el desenvolupament d'unes assumpcions clares, aquest treball pretén produir una solució més definida i explicita.

3.1. Assumpció de categoria inicial

Tal com s'ha delimitat en l'apartat d'objectius, aquest treball es troba enfocat en la categorització de peces de roba, més concret els pantalons, en quatre categories diferents. Així i tot, no s'ha argumentat fins ara, el perquè d'aquestes quatre categories i no un nombre alternatiu.

El problema en el que aquest treball està centrat és la identificació de l'orientació de l'interior i exterior d'una peça de roba per a la manipulació amb robots. L'objectiu és determinar l'estat de dins cap a fora d'una peça per així aportar informació per a, en treballs futurs, deformar la peça i retornar-la a un estat òptim pel seu plegat, allisat o transport.

Una peça de roba pot presentar-se de moltes maneres diferents, i en algunes configuracions és molt difícil determinar si estan correctament orientades pel que fa l'interior i l'exterior. Els capritxos de la moda fan que determinats elements que podrien ser clau per determinar si estem veient el costat interior o exterior de la roba, com ara les costures, no resultin enterament fiables.

La categoria de dins cap a fora té sentit en aquelles peces que tenen una forma tubular, com ara samarretes, jerseis, mitjons o pantalons, i, per tant, primerament, és amb aquests tipus de peces amb els quals interessaria treballar. Després és d'interès que continguin certes característiques com ara etiquetes, costures, butxaques o botons, entre d'altres.

Pel que s'ha comentat anteriorment, és possible que amb una sola d'aquestes característiques no en tinguem prou per determinar el costat d'una part de la peça. De fet, els mateixos humans poden tenir dificultat en classificar o vestir correctament peces de roba que trenquin els esquemes habituals.

Fig. 3.1: Pantalons Kliou, "*Kliou Inside-Out Stacked Women Flare Jeans 2020 High Waist Slim Pockets Blue Demin Pants Autumn Streetwear Female Long Trousers".(Font. (33))*

Per exemple, en la **Fig. *3.1*** es pot observar un tipus de pantalons, que es troben actualment a la venda, que demostra certes característiques corresponents a un pantaló "del revés", caracteritzat per la costura visible, peces d'un color més clar i gastat que del que s'esperaria, i filaments visibles, tot i que es vesteixen tal com es veu a la imatge.

Fig. 3.2: Exemples de pantalons texans a categoritzar. *(Font. Esquerra (34), Dreta (35))*

Això demostra que els mètodes basats en detectar un cert tipus de característica no sempre donaran el resultat esperat, i dependran molt del tipus de roba considerat. Per tant, és important delimitar el tipus de roba amb el que es treballarà, i a quin tipus d'entrada s'aplicarà el mètode seleccionat.

Aquest treball se centrarà principalment en els pantalons, més concretament texans comuns. Els pantalons texans comparteixen entre ells un gran nombre de qualitats similars, com ara la forma, textura, i l'organització i posició dels diferents elements (butxaques, cremallera, botons). A la vegada, els pantalons texans són molt comuns a dins dels armaris de diferents usuaris, i ens serveixen com un exemple ideal per a iniciar el treball.

3.2. Assumpció d'imatge d'entrada

Un cop delimitat l'estat de quin tipus de peça es vol classificar, és important saber quin tipus d'entrada s'espera.

Les peces de roba poden estar en moltes configuracions, com ara embolicades, estirades, plegades, arrugades, i ser percebudes de diferents maneres, com ara vistes des de dalt, des de cert angle, i d'altres infinites possibilitats. Per a limitar aquest nombre infinit de combinacions s'ha determinat una situació simple i fàcil d'obtenir. La configuració la qual s'espera que la peça es trobi és estirada, vista des de dalt amb un cert angle. La peça en qüestió s'ha de trobar completament orientada en el mateix sentit interior-exterior, i, per tant, no es consideraran aquelles que es trobin parcialment de dins cap a fora.

Fig. 3.3: Exemple d'imatge d'entrada

Per aplicar el mètode proposat, es requereix una fotografia de mida 250x250 mínim, una mida inferior pot no contenir suficient informació per a determinar les característiques d'interès, com ara butxaques, cremalleres o botons, de format JPG que contengui únicament una peça de roba visible aproximadament en el centre d'aquesta.

Les fotografies estaran preses prèviament i guardades en l'ordinador. És per això que, tot i ser rellevant en la determinació de l'efectivitat de l'algoritme, el temps de reacció no serà un factor clau per a verificar l'eficiència del mètode proposat.

3.3. Assumpció del resultat final

De la mateixa manera que s'ha determinat quines són les possibles condicions inicials i les característiques que s'esperen de la fotografia d'entrada, és important determinar quin resultat final es vol aconseguir.

Quan es parla d'una peça de roba, la parametrització i modelització d'aquesta és altament complexa a causa de la seva flexibilitat i varietat, i, per tant, s'han de definir unes categories que englobin certes posicions canòniques de les quals puguem extreure informació concreta. Per a poder determinar aquestes posicions s'ha partit de les configuracions inicials en molts dels treballs enfocats en el plec de roba.

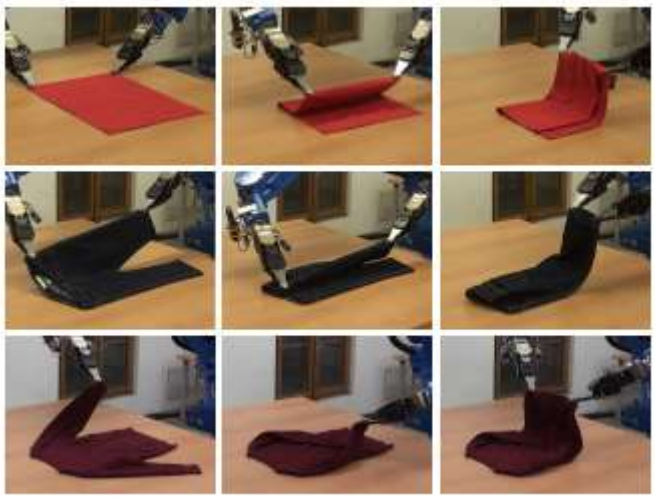


Fig. 3.4: Procediment de plegat de diferents peces de roba.
(Font: (8))

La configuració a determinar prové de l'observació d'una posició que proporcioni el màxim d'informació. Mostrant les dues longituds per a les cames visiblement a la càmera permet determinar que és certament un pantaló.

Tal com s'observen en els objectius, les posicions canòniques que s'esperen són:

- *Right Up (***Fig. *3.6*)*;* Mirant cap per amunt, amb la peça del dret: Determinada per tenir els botons, i cremallera visible a la càmera, juntament amb l'entrada de les butxaques frontals.

Fig. 3.6: Exemple de categoria *Right Up* de pantalons

- *Right Down (**Fig. 3.5**);* Mirant cap per avall, amb la peça del dret: Determinada per tenir les butxaques de darrere, juntament amb una possible etiqueta de la marca visible a la càmera, amb l'entrada a les butxaques de darrere.

Fig. 3.5: Exemple de categoria *Right Down* de pantalons

- *Reverse Up (**Fig. 3.8**);* Mirant cap per amunt, amb la peça del revés: Determinada per tenir la part interna de la cremallera i els botons, amb la part interna de les butxaques, però no la seva entrada, visibles a la càmera.

Fig. 3.8: Exemple de categoria *Reverse Up* de pantalons

- *Reverse Down (***Fig. *3.7);*** Mirant cap per avall, amb la peça del revés: Determinada per tenir l'etiqueta interna, i la part interna de les butxaques de darrere visibles a la càmera.

Fig. 3.7: Exemple de categoria *Reverse Down* de pantalons

Aquestes configuracions permeten delimitar el treball i l'objectiu final d'una manera més clara, tot aportant una simplificació del mètode aplicat. A més a més, tal com es pot veure ens les figures anteriors, s'hi poden trobar detalls en cada una de les imatges que les identifiquen inequívocament amb la categoria en qüestió.

4. Treballs previs

A l'hora de fer un treball d'investigació, és molt important dur a terme una recerca de treballs previs per a observar iteracions anteriors, mètodes alternatius i possibles millores que s'han aplicat a través del temps. Realitzant aquesta documentació, però, es pot veure clarament com hi ha una elevada manca d'informació enfocada al problema concret que es tracta aquí, i no existeixen moltes referències prèvies de les quals es pugui partir.

La robòtica és un dels avenços tecnològics més rellevants de la humanitat. La capacitat d'automatitzar tasques que altrament requeriren humans per a dur-les a terme, permet que la societat es pugui plantejar una redistribució de les feines necessàries i obtenir un estil de vida més lliure i centrat en la creativitat.

És per això que existeix una constant recerca d'aplicacions de robots assistencials en un entorn domèstic. (2) Ha estat demostrat que es produeix un progressiu envelliment de la població, provocant que cada cop ens trobem amb gent més gran durant més temps. Aquest sector de la població pot fer ús de robots assistencials per a mantenir-se i tenir un millor estil de vida. (3).Tal com es veu en la **Fig. *4.1***, actualment existeix una tendència d'envelliment de la població, fet que es demostra amb un augment del percentatge de població per sobre dels seixanta-cinc anys. Amb els anys aquest percentatge anirà augmentat, i conjuntament, el nombre d'usuaris amb requeriments d'assistència.

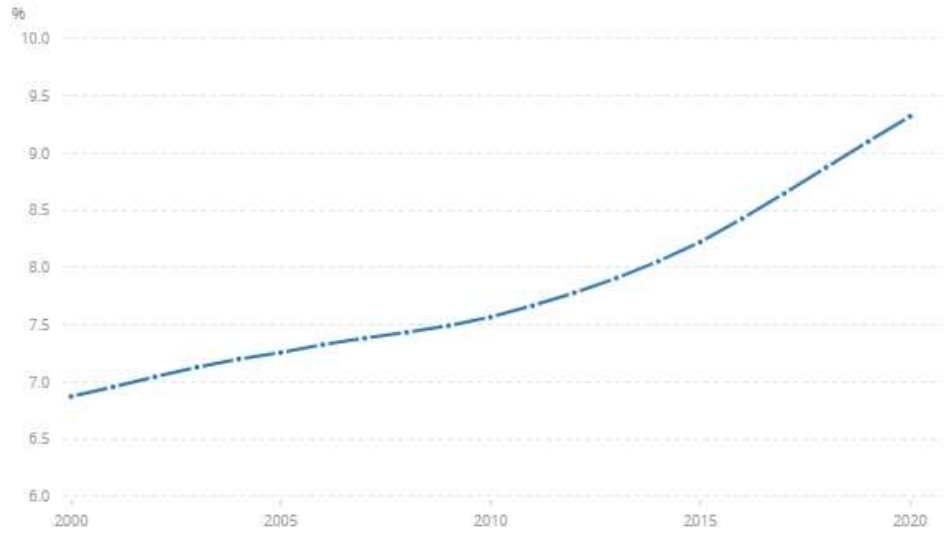


Fig. 4.1: Percentatge de la població del món per sobre dels 65 anys *(Font: (36))*

Això provoca un nombre de problemes que poden ser solucionats mitjançant l'aplicació de robòtica assistencial. A partir de certa edat, o quan el resident té certes discapacitats, existeixen activitats com

ara tenir cura de la llar, alimentar-se o simplement moure's d'un punt a l'altre que són difícils de realitzar d'una manera autònoma.

En concret, aquest treball se centra en la problemàtica de plegar la roba. En la situació mencionada anteriorment, sigui per incapacitats visuals o manipulatives, pot haver-hi un impediment per a finalitzar una tasca com ara recollir una peça de roba, analitzar-la, i actuar en conseqüència (com ara plegar, guardar, planxar, etc.)

L'aplicació de la robòtica, però, no soluciona aquest problema de manera directa i fàcil. Tal com s'ha comentat en l'inici del treball, existeix una alta dificultat per a automatitzar aquesta feina, relacionada amb la complexitat de les peces de roba.

Primerament, és de rellevància poder determinar quin tipus de peça de roba es troba el robot davant. Existeixen moltes peces de roba, amb diferents característiques i propietats, i, per tant, no es poden tractar totes per igual.

"Planning Strategy for Putting away Laundry - Isolating and Unfolding Task" (4)*,* escrita per M. Kakikura i M. Kaneko ens presenta un nombre de mètodes per a l'aïllament, desplegament i plegament de diferents peces de roba apilades. És interessant observar sobretot la tasca de separació o aïllament.

El mètode proposat està compost per dos camins segons la característica de la fotografia. Si es troba de cara a una massa sòlida de diversos colors, separa cada una d'aquestes peces a partir del color en qüestió, aplicant diverses màscares per a aïllar i finalment obtenir els punts per on agafar la peça. En cas contrari, si es troba de cara a una massa d'un únic color, extraurà el punt d'interès a partir de dues ombres, d'on s'obté la discontinuïtat de la peça. Amb aquesta informació més fàcil d'aconseguir se selecciona el punt per on agafar la peça

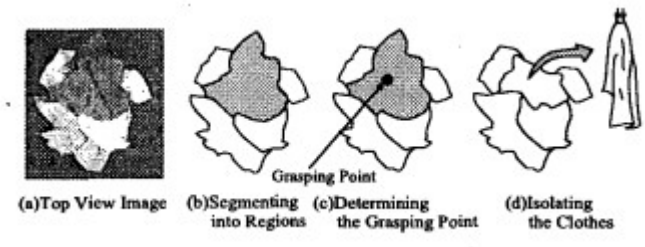


Fig. 4.2: "Isolating Task" de *"Planning Strategy for Putting away Laundry". (Font: (4),* Fig.2)

Un altre article que és d'interès a l'hora d'observar un mètode per a la separació de peces de roba dins d'una imatge formada per una pila d'aquestes és "*A Cloth Detection Method based on Image Wrinkle Feature for Daily Assistive Robots"* (5)*.*

Escrit per K. Yamazaki i M. Inaba presenta un mètode per a la detecció de peces de roba en un entorn domèstic, partint de característiques com ara les arrugues.

Per a fer-ho, primerament s'aplica un "*Gabor Filter* [1]" el qual permet obtenir les diferents arrugues de la peça i després es genera una *SVM (Support Vector Machine)* [2] a partir d'una gran quantitat d'imatges dels diferents plecs. Finalment, s'aplica un "*Graph Cut Algorithm*" [3] i s'extreu aplicant màscares, la selecció final de la peça.

Aquests mètodes comentats permeten la detecció de peces de roba en una superfície, distingint-les d'altres objectes que puguin haver-hi en la imatge. Són doncs un bon inici per a determinar la localització de peces de roba i d'aquesta manera prosseguir a la seva categorització.

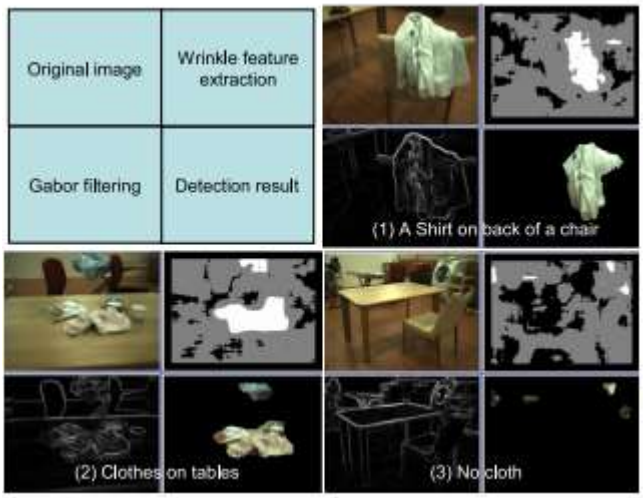


Fig. 4.3: Procediment de detecció del article "*A Cloth Detection Method based on Image Wrinkle Feature for Daily Assistive Robots*". *(Font: (5), Fig .5.)*

[1] *"Gabor Filter"* s'anomena a un filtre lineal utiltizat per a la detecció de cantonades, anàlisis de textures i extracció de característiques. (37)

[2] Una "*Support Vector Machine"* consisteix en un algoritme d'aprenentatge per ordinador utilitzat per a classificació i regressió. Aquest busca un pla que pugui separar les dades de diferents tipus. (39)

[3] El "*Graph Cut Algorithm"* es un algoritme de visió per computador que permet segmentar una imatge en diverses particions, a partir de cert criteri. (38)

Un article que demostra un mètode eficaç es *"Pose and Category Recognition of Highly Deformable Objects Using Deep Learning"* (6), redactat per I. Mariolis, G. Peleka, A. Kargakos, i S. Malassitois. En aquest article, es demostra un mètode per a la detecció de la configuració de diferents peces de roba, juntament amb la seva classificació en tres categories, *"Shirt", "Pants" i "Towel".*

Per a fer-ho, han creat un model d'aprenentatge autònom que permet detectar les característiques d'interès. Més concretament, existeixen dues capes, la primera determinant de la categoria en qüestió, mentre que la segona determina la posició de la peça.

Per a l'obtenció d'imatges s'ha fet ús d'una càmera amb visió de profunditat, prenent un nombre limitat d'imatges, i s'ha augmentat aquestes mitjançant software 3D, reproduint diferents posicions de manera automàtica.

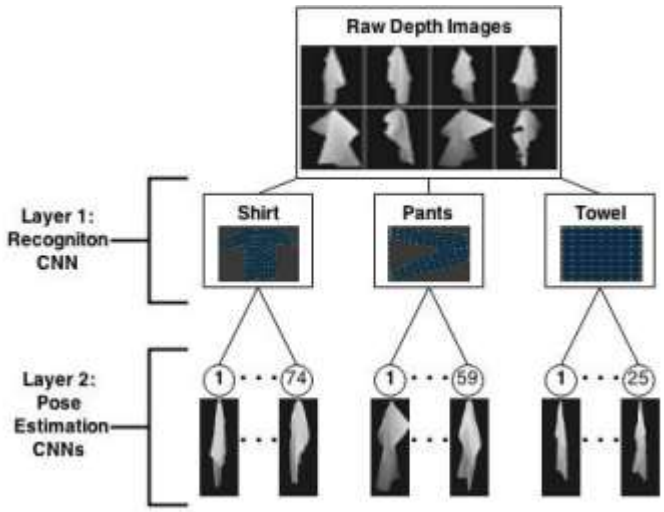


Fig. 4.4: Arquitectura bàsica de *"Pose and Category Recognition of Highly Deformable Objects Using Deep Learning*". *(Font: (6). Fig. 2.)*

Seguint amb l'estimació de posició es troben dos articles fortament relacionats, "*Parametrized Shape Models for Clothing"* (7) i *"Garment Perception and its Folding Using a Dual-arm Robot"* (8). El primer article demostra l'ús d'un algoritme que permet la parametrització geomètrica d'una peça de roba.

Aquesta parametrització, en forma de diferents punts, s'obté mitjançant l'aplicació de diverses fórmules de pesos i d'energia en un conjunt d'imatges simplificades, obtenint finalment, uns punts similars als mostrats en la **Fig. *4.5***.

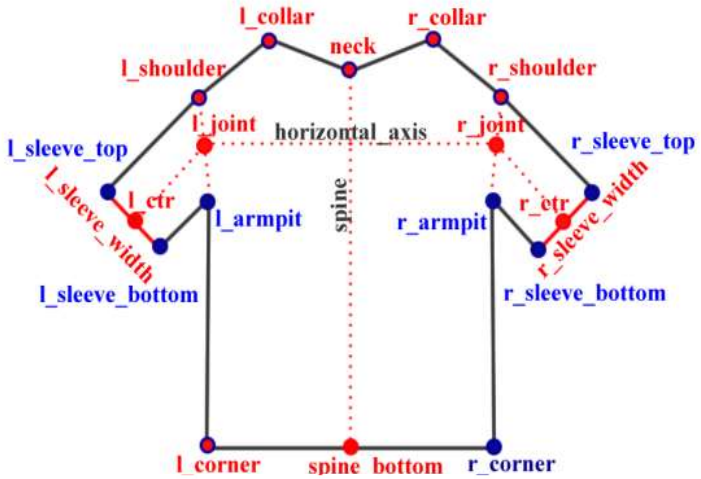


Fig. 4.5: Parametrització objectiu d'una samarreta en l'article "*Parametrized Shape Models for Clothing". (Font: (8), Fig. 4)*

A partir d'aquí, es determina quina és la forma geomètrica més adequada, i finalment la posició en la qual es troba, delimitada pels punts mencionats. Aquests punts permeten després, determinar possibles posicions del robot per a plegar, i actualitzar-se per adaptar-se a la nova posició.

El segon article amplia el primer, optimitzant i millorant l'algoritme de parametrització, tot implementant un mètode per a moure un robot format per dues mans.

S'ha observat, doncs, com els documents comentats es troben especialitzats sobretot a detectar la forma o la categoria de la peça, però cap d'aquests mencionats se centra en el problema del treball, detectar si la peça es troba del dret o del revés.

Malauradament, no existeix molta documentació referent al tema. Aquest és un problema que no s'ha visitat gaire sovint, i, per tant, no hi ha (almenys fins a la redacció d'aquest document) un treball rellevant que demostri un mètode eficient com a solució general al problema. És per això que interessa buscar articles que solucionin problemes semblants o de proximitat, que es puguin aplicar al tipus de roba d'interès.

L'article "*Perception of cloth in assistive robòtic manipulation tasks"* (9), escrit per Pablo Jiménez (codirector del treball) i Carme Torras (Professora d'Investigació a l'IRI), fa una recopilació de diferents mètodes emprats en la percepció de roba en la robòtica assistiva. En aquest document s'hi pot trobar mètodes emprats per a separar la peça de roba del seu entorn, detectar els diferents punts per on es pot agafar, classificació de la peça i detecció de la seva posició, i finalment, i el punt més rellevant per aquest treball, mètodes per al reconeixement de propietats diferenciadores en la detecció d'estat. (**Fig. *4.6***)

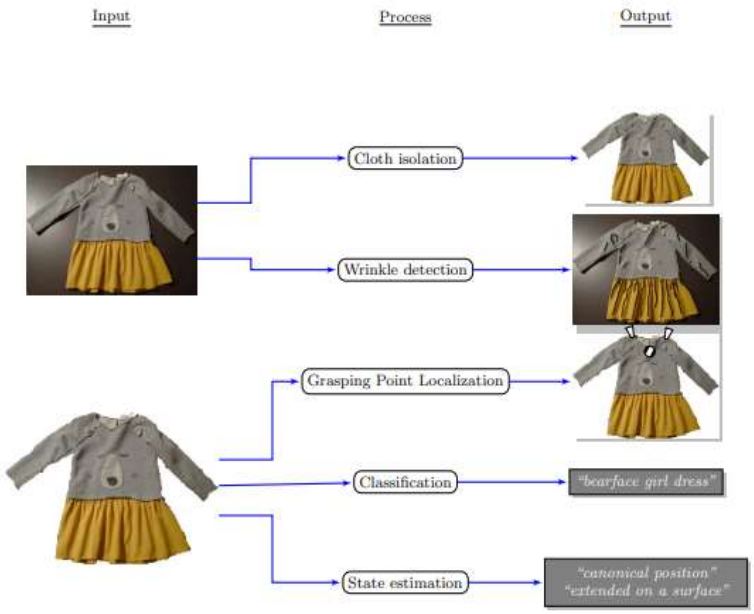


Fig. 4.6: Mètodes aplicables a la peça de roba, amb els resultats (dreta) a partir d'una entrada (esquerra) mitjançant un procés (centre), extret de L'article *"Perception of cloth in assistive robòtic manipulation tasks". (Font: (9), Fig. 1)*

Entre els mètodes proposats trobem la detecció de propietats a partir del tipus de roba, partint de patrons, per exemple determinar si és llana, gràcies a les diferents arrugues, o contorns, i després combinant totes aquestes mesures i fent ús d'una "*Support Vector Machine*" per a determinar i classificar la peça de roba. Un altre mètode és l'ús d'una "*Support Vector Machine*" amb dades tipus color, segments, "*Fast Point Feature Histogram*" [1], i altres característiques. Finalment, trobem mètodes

[1] "*Fast Point Feature Histogram"* es una reducció de complexitat del "*Point Feature Histogram",* en un conjunt pla amb normals dependents de cada punt. (40)

que duen a terme la detecció d'obertures, com ara l'entrada dels camals o l'obertura dels mitjons per a posar el peu a partir de la forma o textura de la peça

Partint d'aquest últim mètode, i el que potser és l'article rellevant més proper al problema presentat trobem *"Perception for the manipulation of socks"* (1), presenta un mètode per al reconeixement de la configuració d'un mitjó.

El mètode combina diferents procediments. Primerament, extreu les característiques de diferents propietats visibles, després s'entrena un classificador, es genera un model per a reconstruir la configuració del mitjó i finalment s'aplica una estratègia per a emparellar els mitjons a partir dels descriptors mencionats.

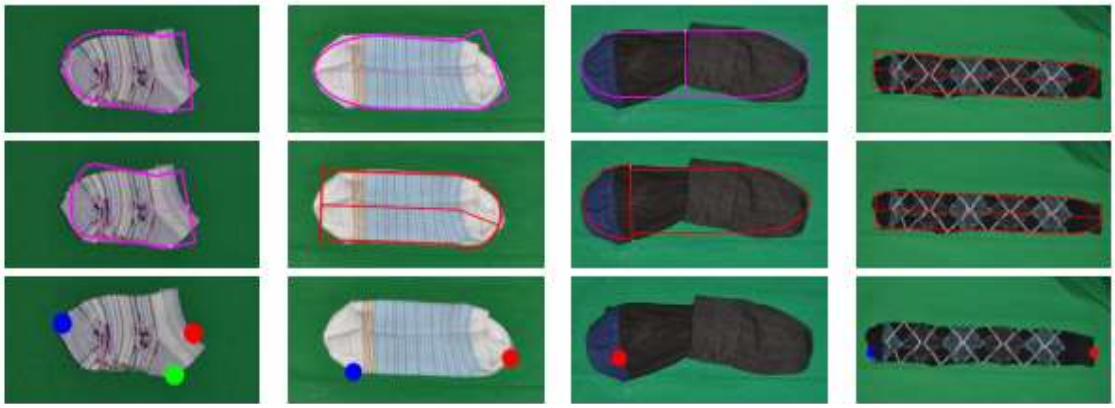


Fig. 4.7: Resultats correctes obtinguts a partir de "*Perception for the manipulation of socks". (Font: (1), Fig. 9.)*

Aquest a més a més, segueix els sistema de parametrització mencionat en articles anteriors, proporcionant informació per al plegament i emmagatzematge de les peces.

Tot i ser un treball enfocat molt específicament a la detecció d'un tipus molt concret de roba, els mitjons, es l'únic que tracta configuracions del revés. Es considera que els mitjons es pot trobar en quatre configuracions, de costat, taló cap amunt, taló cap avall i embolicat. A més a més el peu es pot trobar de dins cap a fora, o amb el costat correcte a l'exterior. Per tant, això implica un conjunt de vuit configuracions possibles a detectar.

5. Eines emprades

Quan es fa un treball que comporta tasques de programació, és molt important determinar quines parts s'han de desenvolupar des de zero i quines no. Ja que existeix un gran nombre de desenvolupadors treballant en múltiples sectors, és interessant fer una anàlisi de les diferents eines disponibles i de quines poden ser rellevants per aquest treball.

Utilitzant el conjunt d'eines adequades es pot minimitzar el cost de desenvolupament i augmentar la velocitat de producció, però és crucial determinar quines eines són realment necessàries, i la qualitat d'aquestes.

5.1. Python

Per a desenvolupar aquest treball, enfocat principalment al processament d'imatges per a detectar i classificar les diferents peces de roba, es requereix una base on poder desenvolupar les operacions i càlculs d'una manera automàtica, ràpida i senzilla.

És per això que és molt important determinar un llenguatge de programació eficient, fàcil d'utilitzar i que doni el màxim de llibertats a l'investigador pel que fa desplegar les diferents operacions.

Un dels llenguatges de programació més utilitzats dins del món del programari és C. Aquest és un llenguatge de programació de propòsit general, desenvolupat el 1969 per Dennis Ritchie, com a una evolució del llenguatge B. És un llenguatge de nivell mitjà, principalment utilitzant en el desenvolupament de sistemes operatius i programari de sistemes. (10) Derivat d'aquest també es troba C++ el qual segueix el paradigma de la programació orientada a objectes

```c
#include <stdio.h>
int main() {
    int n, i;
    unsigned long long fact = 1;
    printf("Enter an integer: ");
    scanf("%d", &n);

    // shows error if the user enters a negative integer
    if (n < 0)
        printf("Error! Factorial of a negative number doesn't exist.");
    else {
        for (i = 1; i <= n; ++i) {
            fact *= i;
        }
        printf("Factorial of %d = %llu", n, fact);
    }

    return 0;
}
```

Fig. 5.1: Exemple de codi escrit en C, per a trobar el factorial d'un nombre. *(Font: (42))*

Enfocat més directament al món de l'estadística trobem R. Desenvolupat per Ross Ihaka i Robert Gentleman l'any 1993, proporciona un entorn gratuït de programació enfocat a l'anàlisi de dades, capaç de treballar en un gran nombre de plataformes. (11)

```
# take input from the user
num = as.integer(readline(prompt="Enter a number: "))
factorial = 1
# check is the number is negative, positive or zero
if(num < 0) {
print("Sorry, factorial does not exist for negative numbers")
} else if(num == 0) {
print("The factorial of 0 is 1")
} else {
for(i in 1:num) {
factorial = factorial * i
}
print(paste("The factorial of", num ,"is",factorial))
}
```

Fig. 5.2: Exemple de codi escrit en R, per a trobar el factorial d'un nombre. *(Font: (41))*

Després trobem MatLab, un programari propietari desenvolupat per *MathWorks* a finals de 1970. Aquest es considera més enfocat a la computació numèrica i, per tant, permet manipulació de matrius, gràfiques de dades o funcions, implementació d'algoritmes, entre d'altres. (12)

```
f= factorial(n)
```

Fig. 5.3: Exemple de codi escrit en MatLab, per a trobar el factorial d'un nombre. *(Font: (43))*

Tal com es pot veure en la **Fig. *5.3***, MatLab conté moltes funcions matemàtiques predefinides que poden ser útils per a l'usuari, de manera que aquest no s'ha de preocupar d'escriure les implementacions.

Finalment Python, un llenguatge de programació d'alt nivell, gratuït, desenvolupat per Guido van Rossum i mantingut per *Python Software Foundation*. Sent d'ús general, es troba enfocat en la facilitat de lectura del codi per humans, a més a més de ser dinàmicament escrit i netejat. (13)

```
# Python program to find the factorial of a number provided by the user.

# change the value for a different result
num = 7

# To take input from the user
#num = int(input("Enter a number: "))

factorial = 1

# check if the number is negative, positive or zero
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   for i in range(1,num + 1):
       factorial = factorial*i
   print("The factorial of",num,"is",factorial)
```

Fig. 5.4: Exemple de codi escrit en Python, per a trobar el factorial d'un nombre. *(Font: (44))*

Inicialment, s'ha descartat C i R ja que existeixen alternatives més eficients pel problema al qual aquest treball es troba enfocat, amb menys complexitat d'ús i més facilitat de comprensió.

En una fase inicial es va utilitzar MatLab, per la funcionalitat que proporcionen les llibreries com ara "*Computer Vision Toolbox*" les quals permeten la creació de codis capaços de detectar, analitzar i en general, processar informació d'imatges, es va creure que seria una de les millors opcions.

Així i tot, avançant en el treball s'ha determinat que MatLab no compleix amb els requisits que aquesta investigació necessita. La finalitat d'aquest treball és poder avançar la investigació ja existent, millorar o aplicar algoritmes passats, i aportar tota aquella millora possible.

És molt important, doncs, que tota aquella feina que es dugui a terme sigui reutilitzable dins del món de la investigació, un fet que el programari de MatLab, al ser d'ús privat, no permet. En cas contrari requeriria tot aquell que fes lectura del treball tenir accés a una llicència MatLab, un fet que, a causa del preu, no pot ser complert per tots.

A més de la gratuïtat que Python aporta al treball, s'obtenen altres beneficis. En un programari d'accés obert, existeix una gran comunitat al seu voltant que proporciona solucions a problemes que un es pot trobar en el procés.

No només això, existeix un gran nombre de llibreries a les quals es pot accedir exclusivament desenvolupades en Python, entre les quals descobrim les més utilitzades en el sector de recerca.

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import joblib
import itertools

%matplotlib inline
from IPython.display import Image
from PIL import Image as ImagePL
from scipy.ndimage import zoom


# Libraries for TensorFlow
from tensorflow.keras import preprocessing
from tensorflow.keras.preprocessing import image
from tensorflow.keras import models, layers
from tensorflow.keras.models import clone_model
from tensorflow.keras.applications import resnet50
from tensorflow.keras import backend as K

from tensorflow.keras.applications.resnet50 import preprocess_input

import tensorflow as tf

from sklearn.metrics import precision_score, recall_score, confusion_matrix,
from sklearn.utils import class_weight
from sklearn.model_selection import KFold
```

Fig. 5.5: Secció de llibreries importades en un projecte de Python. (Font: Pròpia)

5.2. Anaconda

Una de les característiques que té Python és el fet que les llibreries que es desitgin utilitzar requereixen instal·lació a l'ordinador. Això implica que, cada cop que es vol crear un projecte nou, s'ha de verificar que no existeixi conflicte entre les llibreries. Per exemple, un es pot trobar en la situació de dues versions incompatibles entre elles, o llibreries directament inviables, com ara tenir *Tensorflow* i *PyTorch*, les quals tenen una utilitat similar.

La solució simple a aquest problema és desinstal·lar les llibreries que es troben en conflicte, i reinstal·lar aquelles que siguin necessàries. Això, però, introdueix noves dificultats en altres projectes els quals depenguin d'aquelles llibreries o versió.

No només això, sinó que en molts casos, la desinstal·lació de les llibreries no és suficient per tenir una completa neteja de les configuracions, variables temporals o canvis que hagin pogut influenciar les llibreries existents o futures.

És d'alt interès, doncs, aconseguir una solució a aquest problema, una solució que permeti la instal·lació de múltiples llibreries, en paral·lel a altres, i que a més a més permetin la fàcil i ràpida neteja en cas que tot el sistema col·lapsi.

És aquí on Anaconda apareix. Anaconda és la manera més fàcil per a treballar amb projectes de Python/R en una sola màquina (14). Aquesta és una distribució de Python i R per a la computació científica que s'enfoca en simplificar la gestió de llibreries i paquets i la seva distribució dins del mercat.

Fou creat el 2012, des de la companyia *Anaconda Inc*, i ens permet la creació d'entorns virtuals on els paquets poden ser instal·lats, eliminats i modificats sense influenciar els altres entorns.

A més a més, permet una connexió directa amb *Jupyter Notebook* la qual facilita encara més la separació de responsabilitats entre els diferents codis dins d'un propi projecte, connectant cada un d'aquests a un entorn virtual independent (15).

5.3. Jupyter Notebook

Python és un llenguatge de programació que permet la seva escriptura de codi de diverses maneres. Generalment, es pot fer ús del seu propi IDE (*Integrated Development Environment*) per a crear els diferents fitxers que ens permet la compilació i després l'execució del codi en qüestió.

Existeix però un problema, i és que, fent servir un simple IDE per a crear els diversos arxius, complica la feina per a visualitzar seccions, modificar parts del codi, o simplement executar certes parts per a observar que tot funcioni correctament, evitant, en cas necessari, seccions que requereixin molts recursos.

És per això que, per a fer aquest treball s'ha fet ús de Jupyter Notebook, una aplicació web per a crear i compartir documents computacionals. (16)

Jupyter Notebook, creada el 2015 com a una variant del projecte IPython per Fernando Pérez, permet l'escriptura de codi d'una manera seccionada, donant l'opció d'afegir comentaris, títols, descripcions per sobre i sota del codi. L'execució dels diferents blocs de codi de manera independent a tot el treball facilita la feina a l'hora de visualitzar seccions parcials del codi o resultats finals, sense haver d'executar tot el codi cada cop que es faci un canvi.

A més a més, fent ús d'una Jupyter Notebook es pot compartir d'una manera mes fàcil i clara, el codi creat, tot introduint conclusions i comentaris parcials de les diferents seccions. Les llibretes *Jupyter,* suporten altres tipus de llenguatge, no només Python, cosa que la fa una molt bona opció per a realitzar tot mena de codi d'investigació o recerca.

Un exemple d'aplicació es pot trobar en la pàgina web *Kaggle* (17), un repositori de Jupyter Notebooks i diversos *datasets* els quals permeten trobar, publicar, explorar o crear models penjats a la web per altres científics o participants d l'aprenentatge autònom.

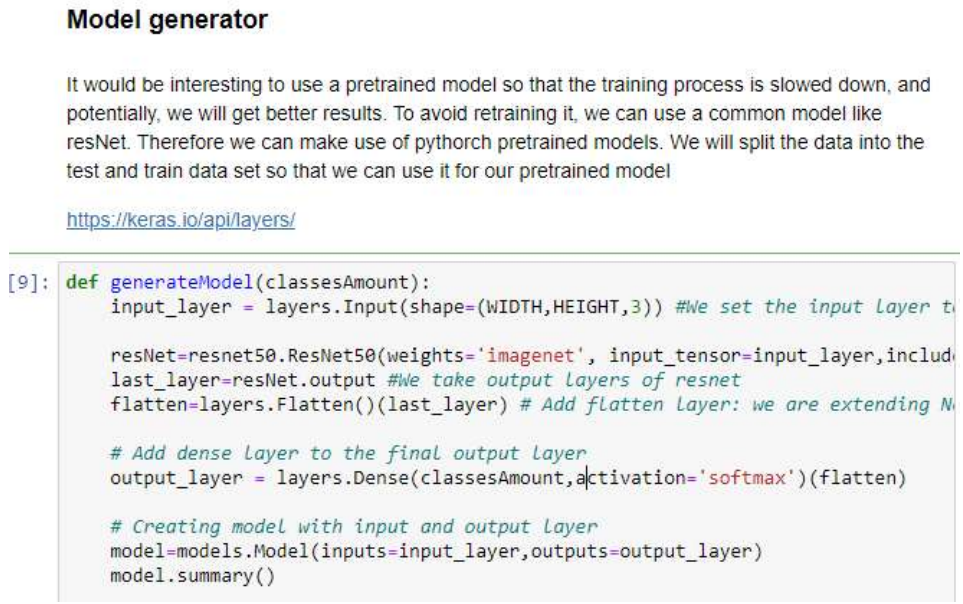


Fig. 5.6: Exemple de secció de Jupyter Notebook. (Font pròpia)

A l'hora de triar aquesta eina, no s'han trobat alternatives similars que tinguin característiques millors. A més a més, amb la col·laboració d'Anaconda, tot aquest projecte es troba aïllat d'altres projectes, independent a versions o conflictes amb anteriors treballs, i funcional de manera autònoma.

5.4. TensorFlow

TensorFlow, desenvolupada per Google Brain el 2015 (18), és una llibreria de codi obert enfocada principalment per a l'aprenentatge autònom. Això permet la detecció de patrons i correlacions d'una manera similar a l'aprenentatge humà.

Tal com es veurà més endavant, un mètode que s'ha intentat múltiples vegades és la detecció de les diferents possibles formes fent ús d'aprenentatge autònom, les quals no seria possible sense fer ús d'una llibreria de capacitats similars.

D'entre les alternatives trobem *Keras* (19) o *PyTorch* (20) les quals permeten obtenir resultats similars.

Tot i així, existeix un previ coneixement necessari per a fer-ne ús de manera còmode i avançada, un coneixement que es té de *TensorFlow* i no pas de les alternatives aquí mencionades.

Mitjançant *TensorFlow* es poden desenvolupar models d'aprenentatge autònom ja preestablerts o nous des de zero, fer ús de computacions numèriques complexes mitjançant dispositius com la GPU interna de l'ordinador, accelerant el procés general, i utilitzar models i *datasets* preentrenats preparats per a desenvolupar noves tecnologies.

Això fa que aquesta llibreria sigui una de les més utilitzades dins del món de l'aprenentatge autònom, considerant-se una de les eines més utilitzades pels investigadors del sector. (21)

En comparativa existeix *PyTorch*, una llibreria de capacitats similars desenvolupada per *Facebook AI Research Lab* el 2016 (22). Entre les diferències trobem l'ús principal de CUDA (23), una plataforma de computació en paral·lel feta per NVIDIA, i enfocada als dispositius d'aquesta companyia.

Tal com es pot observar en la ***Fig. 5.7***, *TensorFlow* ha estat durant molt de temps una de les eines més emprades dins del món de la recerca, hi ha estat recentment que ha agafat empenta *PyTorch* per posicionar-se molt a prop d'aquest.

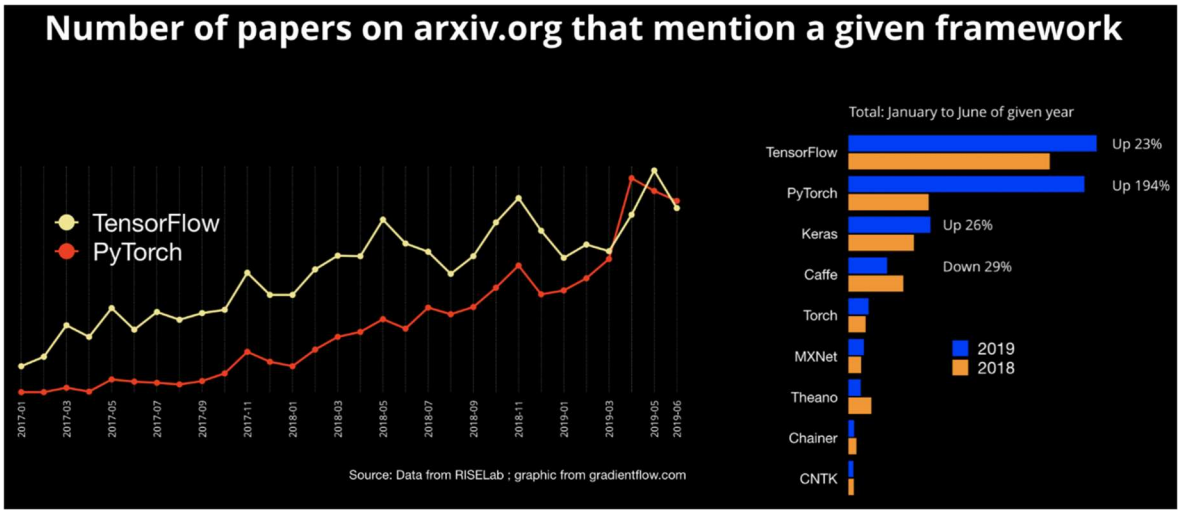


Fig. 5.7: Nombre d'articles de recerca penjats a arXiv.org que mencionen cada una de les tecnologies. (Font: RISELab, autor Ben Lorica) *(32)*

TensorFlow dona més oportunitats a desenvolupadors i organitzacions gràcies al gran nombre de possibilitats de llançament, mentre que *PyTorch* es troba més enfocada a investigadors i científics (20). Així i tot, a causa dels previs coneixements en *TensorFlow*, s'ha determinat que la millor decisió és treballar amb aquelles eines amb les quals s'ha treballat prèviament i no aprendre a utilitzar una nova.

5.5. Altres llibreries

Durant la realització del codi s'ha anat fent ús d'altres llibreries, de menor importància, però que requereixen ser mencionades.

Primerament, *OpenCV* (24) ofereix un gran nombre de funcions enfocades a la visió per computador que faciliten la feina per a processar i analitzar les diferents imatges de pantalons de les quals es fan ús.

Després trobem *Augmentor* (25) una llibre d'accés obert i gratuïta que s'ha fet servir principalment per a generar noves imatges a partir de les poques que s'han pres. D'aquesta manera es poden entrenar algoritmes d'aprenentatge autònom sense fer ús dels recursos que farien falta per a obtenir el nombre òptim d'imatges.

Finalment, *SkLearn* (26) es una llibreria especialitzada en l'anàlisi predictiva de dades, de fàcil accés i basada en *NumPy, matplotlib* i *SciPy.* Gràcies a aquest paquet, es poden realitzar les diferents gràfiques dels resultats d'aprenentatge autònom adjuntades en el projecte.

6. El procés iteratiu

A l'hora de fer un treball acadèmic, sigui de recerca o investigació, és molt important planejar i organitzar les diferents parts a treballar, juntament amb els objectius que es volen complir.

Aquest treball no ha estat una excepció, i, per tant, abans de començar amb la part pràctica ha estat d'alta importància el plantejament dels diferents processos que es volen dur a terme.

A més a més, sent aquest un treball el qual s'inicia quasi des de zero, amb poques referències a mètodes anteriors, és molt rellevant determinar un pla que permeti completar els diferents objectius sense perdre de vista el pas del temps i avenços en les diverses parts.

És per això que aquest treball s'ha realitzat d'una manera iterativa. En comptes de presentar una possible solució des de l'inici, realitzar-la i extreure conclusions, pot ser d'interès plantejar diversos mètodes més simples, fàcils de realitzar, i observar de manera iterativa com seguir la investigació a partir dels resultats obtinguts.

El procés iteratiu proporciona diversos beneficis respecte a altres estratègies d'organització. Aquesta estratègia permet a l'usuari refinar i millorar cada una de les parts que el componen d'una manera eficient i ràpida.

A més a més, en finalitzar cada una de les iteracions es fa un plantejament de la situació en la qual el treball es troba, s'analitza quins objectius s'han complert, quins requereixen millora i quins s'han de reiniciar i permet fer una reestructuració en conseqüència.

Existeixen altres estratègies com ara l'estratègia de cascada, la qual depèn de complir uns passos preestablerts per a arribar a un resultat. En aquesta, però, canvis externs o interns poden influenciar

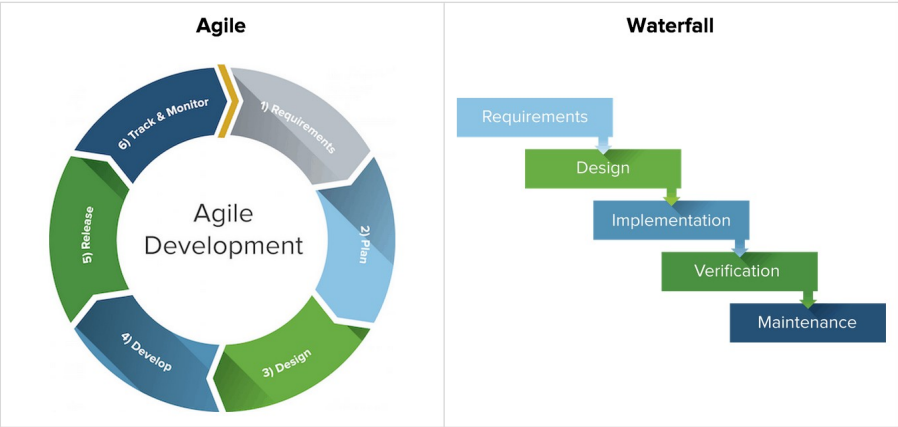


Fig. 6.1: Comparació dels passos en el sistema iteratiu (esquerra) en contrast al de cascada (dreta). *(31)*

la velocitat de producció. El procés iteratiu contempla aquests canvis i permet adaptar-se a les noves circumstàncies. (27)

A continuació es veuran aquelles iteracions on s'han obtingut resultats d'interès, aportant unes observacions i conclusions necessàries per a iniciar la següent etapa iterativa.

7. Intents previs

En aquest punt del treball es farà una revisió dels intents previs al mètode final, tot comentant els resultats i les conclusions obtingudes.

7.1. Determinació de l'estat del pantaló mitjançant una CNN

Partint de què se sap amb certesa que la imatge d'entrada és la d'un pantaló, es vol determinar si aquest es troba en una de les configuracions mencionades en els objectius, ***Reverse Up, Reverse Down, Right Up i Right Down***, utilitzant un mètode directe per realitzar aquesta classificació.

Idealment, si aquest intent funcionés, no caldria realitzar un pas previ per a detectar els pantalons en qüestió, ja que l'algoritme determinaria de forma automàtica que es tracta d'uns pantalons en un d'aquests quatre estats.

Per a l'entrenament d'un algoritme d'aprenentatge autònom es requereix un gran nombre de dades, com més elevat, més gran és la probabilitat d'obtenir un model més precís i profitós. Malauradament, no s'ha trobat una base de dades contenint classificacions semblants a les requerides.

Això implica que s'ha de crear la base de dades de manera manual.

Crear una base de dades d'una magnitud adient, però, requereix un gran nombre de fotografies, amb múltiples variacions, i la seva categorització final. Per tant, no és viable fer tot aquest conjunt d'accions de manera manual.

L'opció proposada és fer ús de "*data augmentation*" [1]per a incrementar la quantitat d'imatges diferents, a partir d'una base de dades més petita. Gràcies al generador que crearà les imatges de manera automàtica, es pot entrenar el model de manera eficient i ràpida.

[1] *Data Augmentation* és el procés d'augmentar la quantitat de dades de les que es disposa, sotmetent-les a algunes transformacions per obtenir-ne variacions sobre les dades originals. En el cas de les imatges, aquestes transformacions inclouen deformacions, filtres o altres.

Fig. 7.1: Imatge generada a partir de les imatges originals, utilitzada per a entrenar l'algoritme

A partir d'aquest generador, es realitzarà un model CNN (de l'anglès, *Convolutional Neural Network)* [1] enfocat a la predicció i classificació de les imatges en les categories desitjades.

Idealment, seria interessant la creació d'un model de xarxa neuronal des de zero, entrenar-lo durant suficient temps, obtenint els resultats més bons. Malauradament, el temps i recursos que això suposa sobrepassen els proporcionats per a la realització del treball. És, per tant, que s'han de buscar mètodes alternatius.

[1] Una CNN o Xarxa neuronal Convolucional, es una xarxa neuronal que treballa de manera més eficient amb imatge, audio o escrits. Es troba formada per múltiples xarxes neuronals de dimensions menors, apilades i entrenades en cadena per a obtenir millors resultats. (45)

L'opció la qual s'ha decantat és fer servir el mètode de "*transfer learning*" [1] per a fer un ús eficient dels recursos i reduir la quantitat d'iteracions, tot millorant els resultats que es podrien obtenir de crear un model amb informació reduïda.

Es determina que un bon model d'on partir el procés de transferència d'ensenyament és una *ResNet50*[2]*.* A partir d'aquesta xarxa neuronal pre-entrenada, s'han aplicat certes modificacions per intentar obtenir millors resultats un cop aplicada.

Primerament, per a fer un ús eficient del temps s'ha determinat que és interessant congelar les últimes capes de l'algoritme d'aprenentatge autònom. Aquestes són les que reaccionen a canvis petits, i, per tant, no cal reentrenar-les. La capa de sortida, però, caldrà modificar-la, juntament amb la d'entrada, per adaptar-se al tipus de dades amb les quals treballem.

La primera capa determina els paràmetres amb els quals el model podrà funcionar correctament, o més ben dit, la mida de la imatge que es fa servir per entrenar el model. Això vol dir que, si es fa ús d'una imatge de 540x540 amb colors RGB, la primera capa serà de 540x540x3. De la mateixa manera, l'última capa es troba determinada pels resultats de sortida que interessa obtenir. Ja que es pretén classificar la peça de roba en quatre categories diferents, l'última capa estarà formada per una capa densa de quatre unitats. Aquestes unitats determinen la mida del vector de sortida.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 540, 540, 3)]     0

 dense (Dense)               (None, 4)                 2367492
```

Fig. 7.2: Primera capa, de mida 540x540x3 (*input_1)* i última capa de 4 (*dense)*

[1] *Transfer Learning* es l'acció de fer servir models que s'han entrenat prèviament amb altres tipus de dades, modificant-lo i adaptant les capes superiors i inferiors, per a reentrenar aquestes amb noves dades. Permet la reutilització de models que d'altra manera requeririen de mesos per obtenir. (46)

[2] *ResNet50* és una variant del model ResNet format per 48 capes convolucionals preentrenada amb un nombre molt elevat de dades. Aquest model es fa servir molt sovint en l'aplicació de *Transfer Learning* a causa de la diversitat d'imatges que s'ha utilitzat en l'entrenament inicial. (47)

La capa densa es troba connectada a totes les neurones de la capa anterior, i permet, mitjançant un producte matricial entre la matriu anterior i aquesta, obtenir el conjunt de prediccions. Una imatge, doncs, aconseguirà quatre valors entre 0-1 a partir d'aquest producte, i, per tant, podem determinar a partir del nombre que sigui més gran, a quina categoria pertany la imatge.

Per a repartir el procés d'entrenament entre diferents dies, a més a més, s'ha implementat un sistema de *"checkpoints"* que permeten guardar l'entrenament parcial del model a l'ordinador i continuar amb aquest més endavant.

A partir del model entrenat, es pot determinar que aquest conté una precisió del **48.75%,** i en la **Fig. 7.3** es pot observar la diferent categorització.

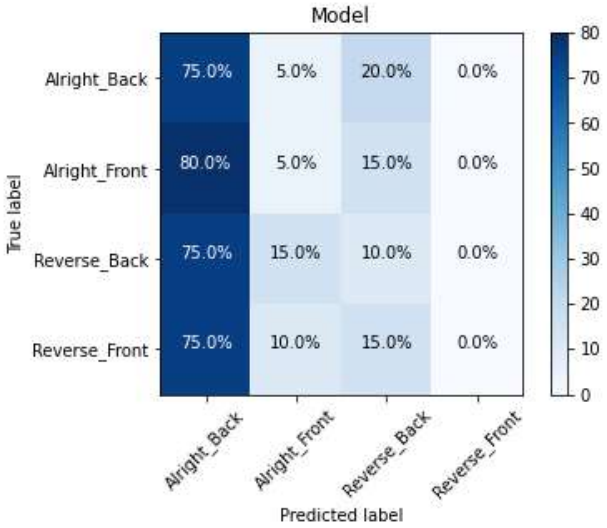


Fig. 7.3: Matriu de confusió del model entrenat a partir de la base de dades augmentada. Eix vertical l'etiqueta real, eix horitzontal l'etiqueta predita.

Idealment, i amb una precisió del 100%, hauria d'aparèixer una diagonal d'esquerra a dreta, des de dalt cap a baix, que contingues el màxim nombre d'objectes classificats correctament. Malauradament, però, aquesta categorització no és gaire precisa, la diagonal no és visible, i existeix un gran nombre d'objectes mal categoritzats.

Això pot provenir de molts problemes, entre els quals el més significatiu és el tamany reduït de la base de dades d'on s'inicia l'entrenament i el generador d'imatges. Amb una base de dades original més gran, probablement se n'obtindrien millors resultats.

Un cop vistos aquests resultats del primer entrenament s'han plantejat dos possibles camins a seguir. La primera opció és continuar i anar modificant els paràmetres fins a obtenir millors resultats. L'altra opció es descartar aquest intent i començar per un altre camí.

Els resultats són clarament pobres, i, tot i que pot ser interessant desenvolupar una implementació que funcioni correctament amb una xarxa neuronal, aquest camí només compliria un dels dos objectius planejats, la detecció de l'orientació, i per tant requeriria investigació extra o tornar a començar per a complir l'objectiu de classificació la peça com a pantaló.

Un altre motiu pel qual s'ha descartat és la lentitud en l'entrenament, en molts casos trigant aproximadament entre 3 i 4 hores a cada canvi. Teòricament, una modificació precisa dels paràmetres de la CNN, juntament amb l'ampliació de la base de dades donaria els resultats desitjats. Però tenint en compte els motius anteriors i el fet de tenir un temps limitat per a la realització del treball és més raonable considerar possibles alternatives.

7.2. Detecció i classificació de la peça de roba mitjançant CNN

Aquest intent, basat en el treball de "*Pose and category recognition of highly deformable objects using deep learning"* (6), presenta un mètode on es fa ús d'aprenentatge autònom per a detectar i classificar una peça de roba a la categoria de pantalons a partir d'una base de dades. La hipòtesi consisteix en, un cop detectada la peça en qüestió, i amb la determinació que aquesta és un pantaló, obtenir, a partir de les últimes capes de la xarxa neuronal, una màscara per a separar la peça de l'entorn.

Aquest intent consisteix en la realització de quatre models CNN (*Convolutional Neural Network)* amb diferents característiques per a observar els resultats obtinguts, amb la idea d'agafar aquell que doni millors resultats.

Per a preparar un model basat en una xarxa neuronal, és molt important obtenir una base de dades prou gran per a entrenar-la. Gràcies a la pàgina web de *Kaggle*, s'ha pogut descarregar un conjunt d'imatges, amb les etiquetes pertinents, que permeten la creació del model.

D'aquí es pot extreure 5.000 imatges de 20 categories diferents, d'entre les quals trobem "*T-Shirt", "Shoes", "Shorts"* i altres, visibles en la **Taula *1***.

Taula 1: Quantitat d'imatges existents per a cada etiqueta de la base de dades extreta de *Kaggle*

Etiqueta	Quantitat
T-Shirt	1011
Longsleeve	699
Pants	692
Shoes	431
Shirt	378
Dress	357

Outwear	312
Shorts	308
Not sure	228
Hat	171
Skirt	155
Polo	120
Undershirt	118
Blazer	109
Hoodie	100
Body	69
Other	67
Top	43
Blouse	23
Skip	12

Aquestes imatges contenen la peça generalment en una posició canònica, totalment estirada, i en un lloc quotidià. És, per tant, que els fons d'aquestes poden ser d'un terra de pedra, de fusta, penjades en els armaris o sobre un llit o tela, amb colors i textures diverses. A més a més, no conté pantalons o peces de roba que es trobin del revés o de dins cap a fora, i, per tant, només es pot fer servir per a detectar el tipus de roba.

Fig. 7.4: Exemples d'imatges del *data set* extret de *Kaggle.* A l'esquerra imatge del tipus *Hoodie*, a la dreta *Pants*.

Finalment, la mida i qualitat de la imatge és variable i, en conseqüència, es requereix un preprocessament per a ser utilitzades en la CNN, fixant la mida de la imatge a 250x250, la mida de la capa inicial del model.

Mitjançant una *ResNet50,* i seguint un procés similar al intent anterior, s'ha realitzat un model entrenat amb la base de dades completa, les 5000 fotografies amb 20 categories, i s'han observat els resultats següents després d'un entrenament de 20 cicles.

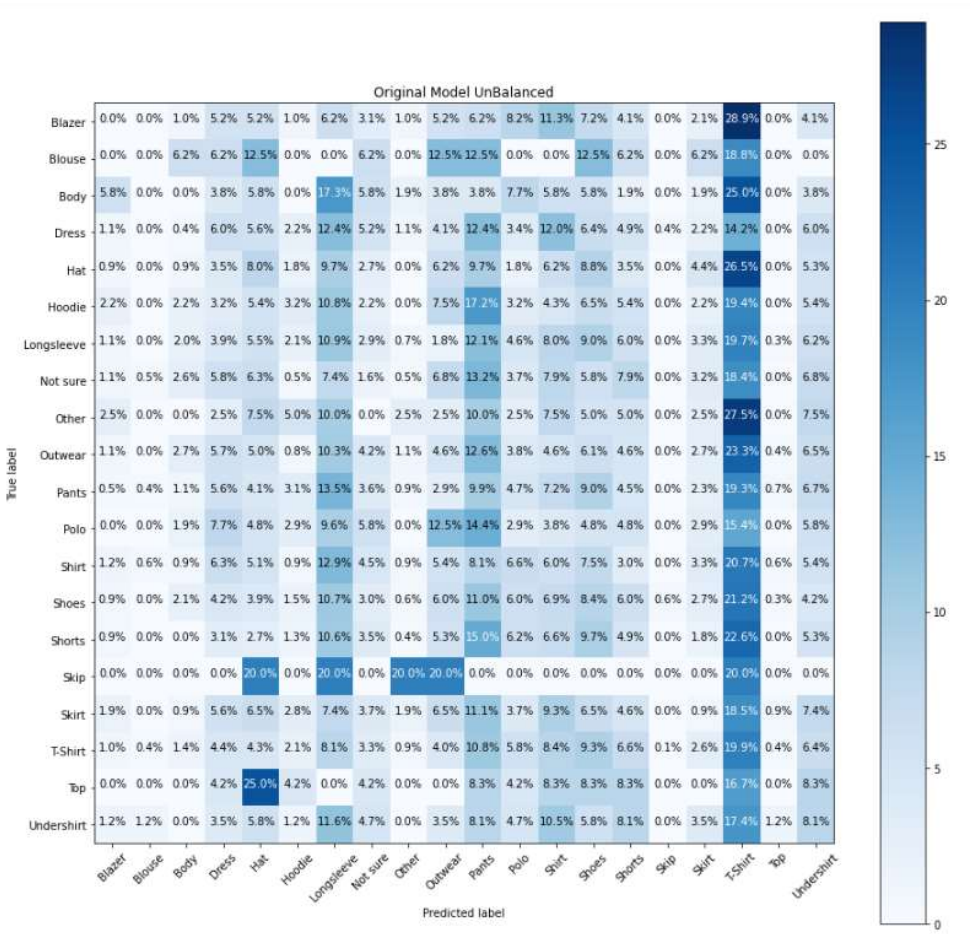


Fig. 7.5: Matriu de confusió del model entrenat a partir de la base de dades original. Eix vertical l'etiqueta real, eix horitzontal l'etiqueta predita. . Normalitzada en percentatge.

La precisió d'aquest model es troba al voltant del 71,03%, tot i que malauradament, aquest valor alt ve donat a partir de la gran quantitat d'imatges classificades a una categoria. Per tant, aquest mètode no resulta prou precís perquè sigui útil per classificar els pantalons.

Observant aquests resultats, s'ha determinat que seria interessant refer aquest model utilitzant uns pesos balancejats de les diferents categories, provocant que aquelles que apareixen més vegades continguin menys pes i, en conseqüència, influenciïn menys en la variació del model.

Balancejant aquestes dades veiem que la precisió ha augmentat a un 88,75%, però la classificació encara està bastant mal distribuïda.

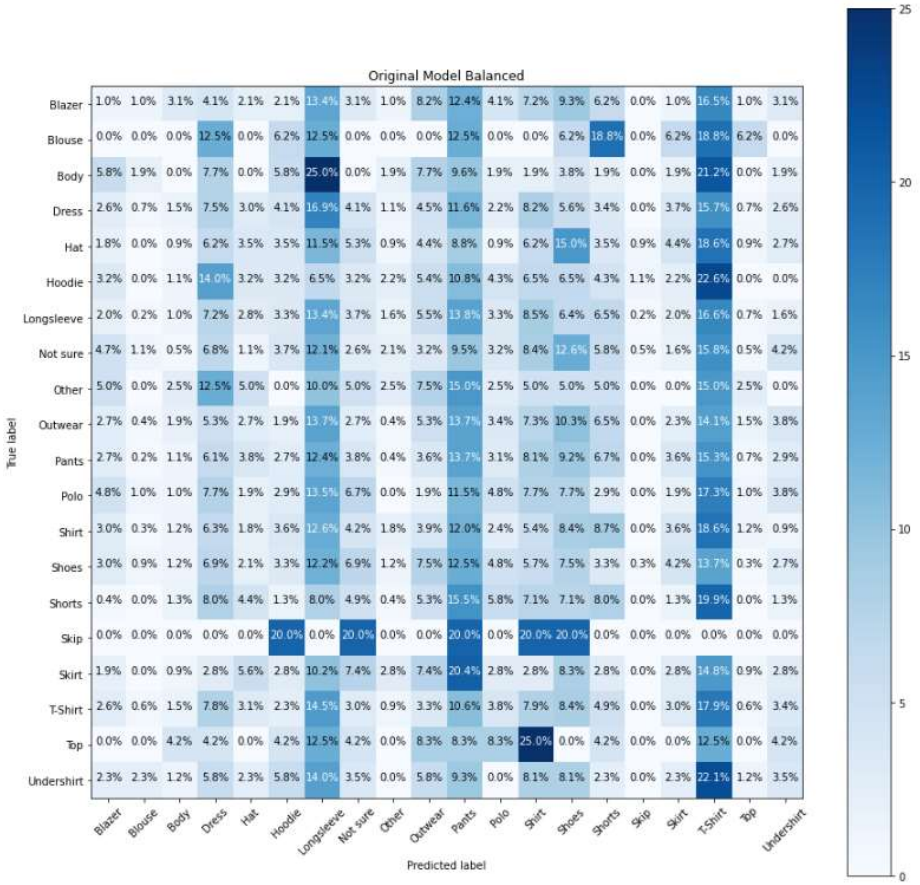


Fig. 7.6: Matriu de confusió del model entrenat a partir de la base de dades original balancejada. Eix vertical l'etiqueta real, eix horitzontal l'etiqueta predita. Normalitzada en percentatge.

A partir d'aquí es va decidir reduir el nombre de categories, i, per tant, la quantitat d'imatges a processar. D'aquesta manera l'entrenament serà més àgil i a la vegada tindrà menys possibilitat d'error. Per a fer-ho s'ha netejat i simplificat el nombre de categories a tres. La categoria *Bottom_Long* conté aquelles que provenen de *Pants*, la categoria Top contindrà aquelles que provenen de *T-Shirt*, triada a causa de la quantitat d'imatges que proporciona i la seva diferència amb els pantalons, i finalment la categoria *Bottom_Short* conté totes aquelles que són de tipus *Shorts*, triada per a la seva similitud amb la categoria de pantalons, tot mantenint les diferencies de llargada dels camals.

Taula 2: Quantitat d'imatges existents per a cada etiqueta de la base de dades extreta de *Kaggle*

Etiqueta	**Quantitat**
Top	1054
Bottom_Long	692
Bottom_Short	308

A partir d'aquesta nova base de dades s'ha realitzat un entrenament sense balancejar els pesos observant una precisió alta, del 88,93%.

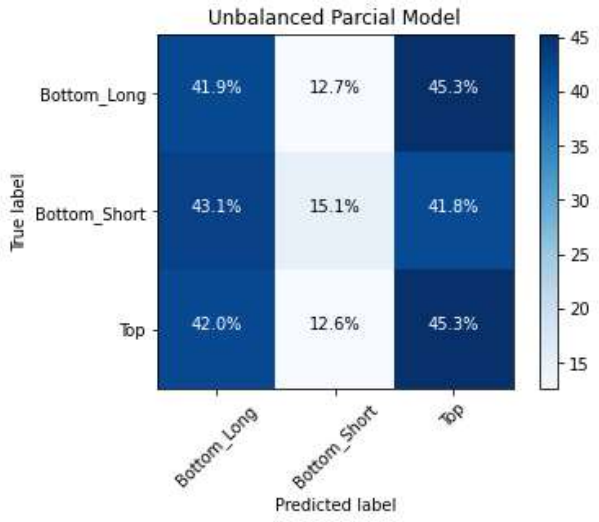


Fig. 7.7: Matriu de confusió de *"Unbalanced Parcial Model"*. Eix vertical l'etiqueta real, eix horitzontal l'etiqueta predita.

Malauradament, seguint el patró dels models anteriors es pot observar com existeixen moltes imatges classificades incorrectament com a *Bottom_Long* quan realment són de tipus Top. I de la mateixa manera, existeixen molt poques classificades correctament com a *Bottom_Short*. (***Fig. 7.7)***

Finalment, s'ha provat de realitzar la mateixa millora que en la primera part i balancejar el model, creant el "B*alanced Parcial Model*".

Els resultats són els següents, obtenint una precisió del 91,30%. Així i tot, es pot observar que els resultats aconseguits segueixen sense ser satisfactoris, arrossegant els mateixos problemes mencionats anteriorment. (**Fig. *7.8***)

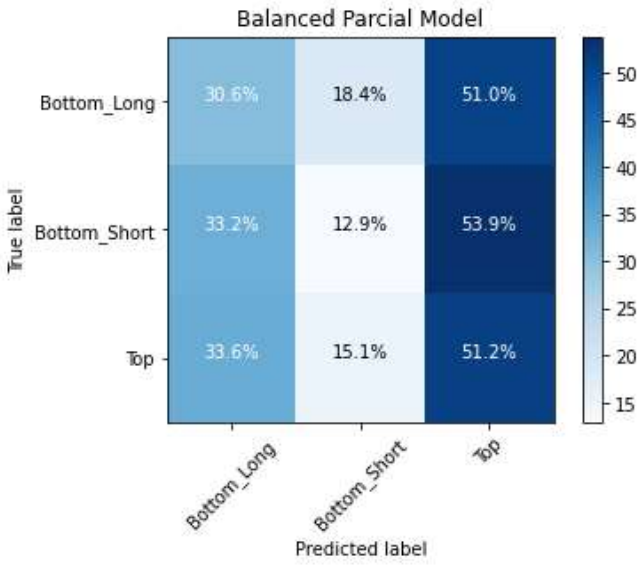


Fig. 7.8: Matriu de confusió de "*Balanced Parcial Model*". Eix vertical l'etiqueta real, eix horitzontal l'etiqueta predita.

En la següent taula s'observen els diferents resultats obtinguts de cada model, validats a partir de la precisió.

Taula 3: Resultats de precisió dels diferents models realitzats

Model	Precisió	Model	Precisió
Unbalanced Original Model	**71.03%**	*Unbalanced Parcial Model*	**88.93%**
Balanced Original Model	**88.75%**	*Balanced Parcial Model*	**91.30%**

Tal com es pot veure en la taula en qüestió, el model d'on es poden extreure els millors resultats és "*Balanced Parcial Model".*

A partir d'aquest punt en la iteració és important determinar els punts de millora i si val la pena continuar per aquest camí o buscar una alternativa.

Aquest intent pretenia fer ús de la CNN per a aïllar els pantalons del fons, utilitzant les últimes capes de la CNN. Així i tot, tal com es pot observar, no funciona correctament en molts dels casos, i, per tant, els resultats són pobres. Tal com s'ha comentat anteriorment, l'entrenament d'una CNN d'aquestes característiques involucra un gran temps de processament, el qual, sumant el fet que s'ha de retocar els paràmetres i reentrenar, provoquen una lentitud d'avanç, per un mètode que pot ser substituït per un de més eficient.

8. Mètode proposat

Un cop observats els diferents resultats de l'aplicació de xarxes neuronals directament a les imatges, s'ha determinat que un sotmetent-les primer a un preprocès, i utilitzant un altre tipus d'aprenentatge per ordinador pot donar millors resultats. És aquesta idea que dona lloc al següent mètode funcional.

El plantejament d'aquest ve directament lligat amb la realització per parts, dels diferents objectius plantejats, donant a pas a dos algoritmes que compleixen cada un dels objectius.

8.1. Objectiu 1: Extracció i classificació de la imatge en la categoria de pantaló

8.1.1. Extracció de la zona d'interès de la peça

El primer objectiu a complir és la detecció i classificació de diferents peces de roba per a seleccionar aquelles que pertanyin a la categoria de pantalons.

Aquest procés comença a partir d'una imatge (**Fig. *8.1***) que contingui una peça de roba. Tal com s'ha mencionat en les assumpcions, la imatge en qüestió ha de contenir únicament una peça de roba, sense altre tipus de teles visibles en aquesta (com ara mantes, coixins, o altres peces que puguin generar arrugues).

Fig. 8.1: Exemple d'imatge d'inici per l'algoritme d'extracció.

A causa de la naturalesa de les peces de roba, les seves formes i dimensions són altament variables, i poques característiques es mantenen iguals entre una peça i una altra. Una d'aquestes, però, és l'existència d'arrugues, les quals són visibles en les diferents peces de roba. Seguint el mètode proposat per Yamazaki K. I Inaba M. (5) es planteja, inicialment, l'aplicació d'un conjunt de filtres Gabor per aïllar la peça del seu entorn.

Els filtres Gabor reaccionen a les ones a partir de certa freqüència i direcció. La matriu aplicable pot tenir formes com les mostrades en la **Fig. *8.2***.

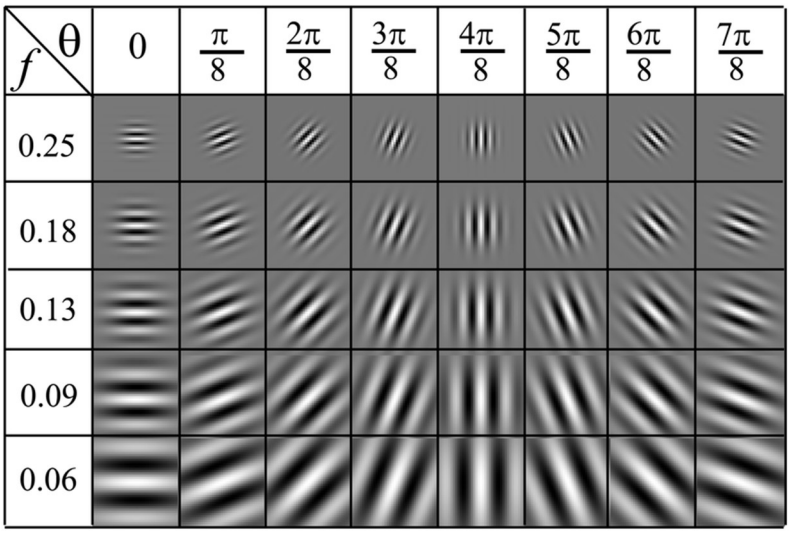


Fig. 8.2: Parts reals d'un banc de filtres *Gabor*, generats a partir de diferents combinacions d'angles i freqüències. *(48)*

En el cas d'aquest treball, s'ha fet una generació de 16 filtres variant l'angle de rotació per a obtenir un filtre aplicable en 360°. A més a més, s'ha fet servir el generador proporcionat per la llibreria *OpenCV* (24), fent ús de la funció *getGaborKernel* per a generar les diferents matrius i retornar-les en forma de llista.

Una representació matemàtica 2D d'un filtre *Gabor* pot ser donada per:

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = e^{a} * cos\left(2\pi \frac{x'}{\lambda} + \psi\right) \quad (8.1)$$

On:

$$a = -\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2} \quad (8.2)$$

$$x' = x \cos\theta + y \sin\theta \quad (8.3)$$

$$y' = -x \sin\theta + y \cos\theta \quad (8.4)$$

D'aquí podem observar els diferents paràmetres i determinar quina funció tenen cada un dins de la seva representació. x i y són les diferents coordenades del filtre, λ representa la longitud d'ona de la forma sinusoidal, i, per tant, com més gran, la distancia entre els diferents pics augmenta, θ representa l'orientació de les línies paral·leles del filtre, les quals reaccionaran diferent segons les ones en la imatge, ψ és l'òfset de fase, amb els valors petits generant resultats més nets, σ és la desviació estàndard la campana de Gauss, on valors grans generaran més cantonades, i finalment, γ és la ràtio d'aspecte que ens determina quant d'el·líptic serà el suport del filtre de Gabor.

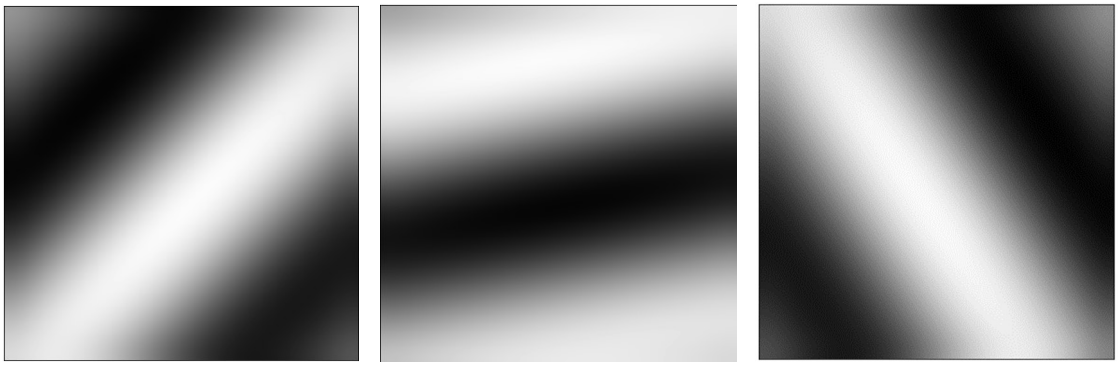


Fig. 8.4: Representació de tres dels setze filtres *Gabor* generats.

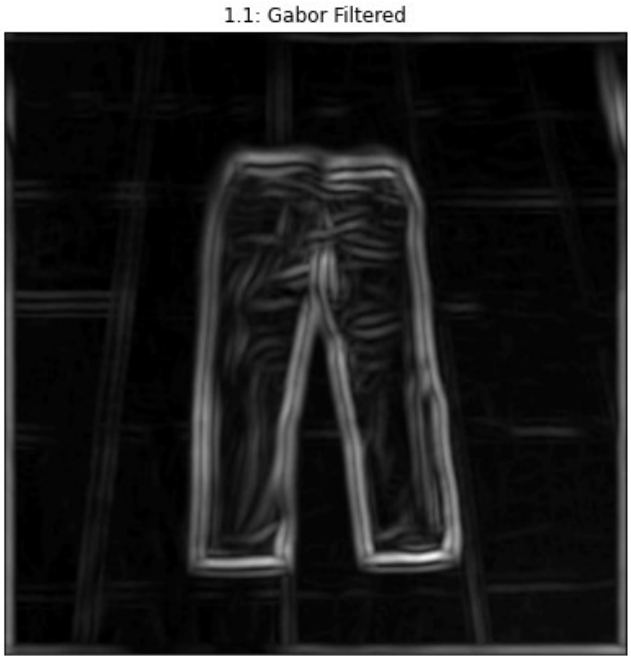


Fig. 8.3: Filtre Gabor aplicat a la imatge original en escala de grisos

A partir de diferents experiments de diverses imatges i analitzant els diferents resultats, s'ha determinat que les imatges reaccionen adequadament amb λ= 13.0, ψ = 0.5, σ = 8.0, γ = 0.5, amb una mida de 15x15 i amb la generació de 16 filtres variant l'angle d'aquests.

Després, aquestes matrius s'apliquen a la imatge original transformada a escala de grisos, per a ressaltar i detectar aquelles zones on apareixen arrugues, generant la **Fig. *8.3***.

A partir d'aquí, es fa un retall per a eliminar els costats superior, inferior, esquerra, i dreta de la imatge, i es binaritza la imatge en blanc i negre. Aquesta imatge binaritzada representa una zona ampliada del pantaló per a agafar aquells punts que potser s'han perdut en la detecció de les arrugues. A més a més, per poder obtenir una imatge neta, s'aplica una detecció de contorns i s'emplena la zona resseguida, donant a pas a la **Fig. *8.5***.

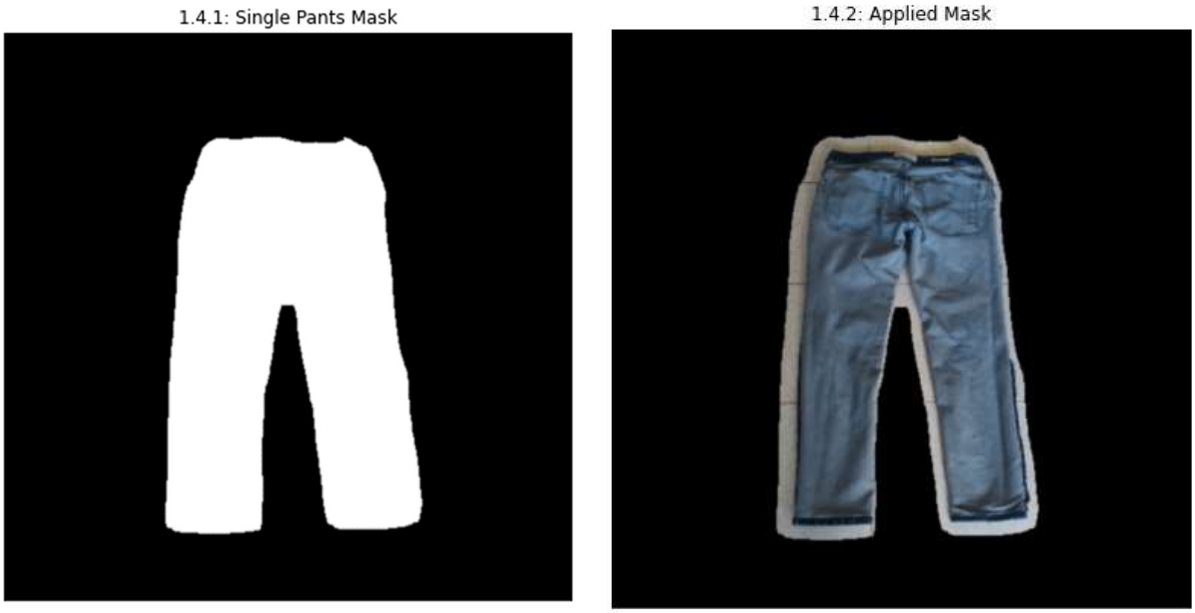


Fig. 8.5: A l'esquerra, mascara de la zona d'interès, a la dreta, mascara aplicada a la imatge original.

8.1.2. Extracció refinada de la peça

Aplicant la màscara mostrada en la **Fig. *8.5*** es pot observar com la imatge conté zones que no formen part del pantaló. Això pot provocar que la parametrització d'aquesta no es trobi prou propera al contorn dels pantalons, i que zones que no formen part del pantaló siguin confoses com a parts d'aquest.

És per això que es fa ús de l'algoritme de *GrabCut* (28) per a delimitar la peça d'una manera més precisa. Aquest es fa servir principalment per a l'extracció de fons de les imatges partint de la mínima interacció

humana. Hi ha dos mètodes proposats per *OpenCV*¸ els quals, mitjançant una màscara o un rectangle en la zona d'interès, generada per l'usuari, s'extreu el fons d'aquesta.

L'algoritme prové d'un problema d'optimització d'una funció de cost d'energia. Aquest se soluciona mitjançant la creació d'un model gràfic, on cada vèrtex i canto té cert pes. L'entrada de l'algoritme consisteix en la imatge i les seves etiquetes (definint-les com la zona de fons, la zona probablement de fons, la zona de primer pla i la zona de probablement primer pla), i aquest intenta trobar unes millors etiquetes que tinguin menys cost. El cost serà menor quan els veïns d'un conjunt de píxels amb certa etiqueta comparteixen el mateix color. Mitjançant una predeterminació, generalment manual, de les etiquetes, es pot obtenir un punt d'inici per a determinar la resta de píxels que comparteixen l'etiqueta.

Aquest treball, però, intenta buscar un mètode per a minimitzar la interacció humana, i, per tant, seria interessant evitar que l'usuari hagi de definir manualment la zona d'interès.

És per això que es farà ús de la màscara extreta anteriorment per a delimitar la zona d'interès. L'algoritme fa ús de quatre etiquetes, donades per un valor entre 0 i 3 de cada píxel. Quan el píxel té un valor de 0, es diu que aquest forma part del fons, i per tant, s'eliminarà en la imatge final, un valor de 1 ens diu que forma part de primer pla i existirà en la imatge final, 2 són aquelles zones on probablement hi haurà una zona del fons, i 3 és la zona on probablement hi haurà una zona de primer pla.

Transformant la imatge binària anterior al fet que els píxels blancs tinguin un valor de 3, mentre que els negres tinguin un valor de 0, podem introduir-la a l'algoritme i extreure la **Fig. *8.6***.

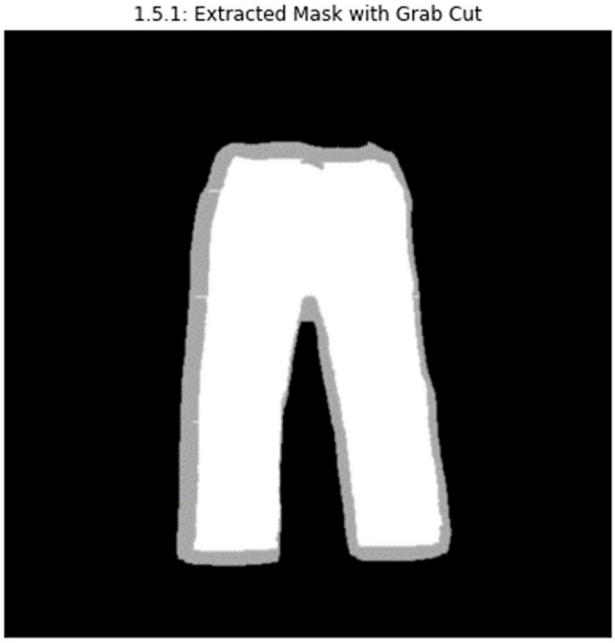


Fig. 8.6: Mascara generada per l'algoritme de GrabCut.

Aquí es poden observar clarament tres colors. El negre fa referència a tota la zona que és de fons, mentre que el blanc a la zona de primer pla. La zona grisa, en canvi, es troba en la categoria de probablement primer pla. Ja que la mascara inicial contenia només dos valors, fons i probablement primer pla, són aquests últims que es divideixen entre probablement primer pla, i primer pla. Es per tant que no trobarem píxels referents a probablement fons.

Mantenint únicament la zona de color blanc, transformant la resta a negre, i netejant la imatge, s'obté el resultat de la **Fig. *8.7***.

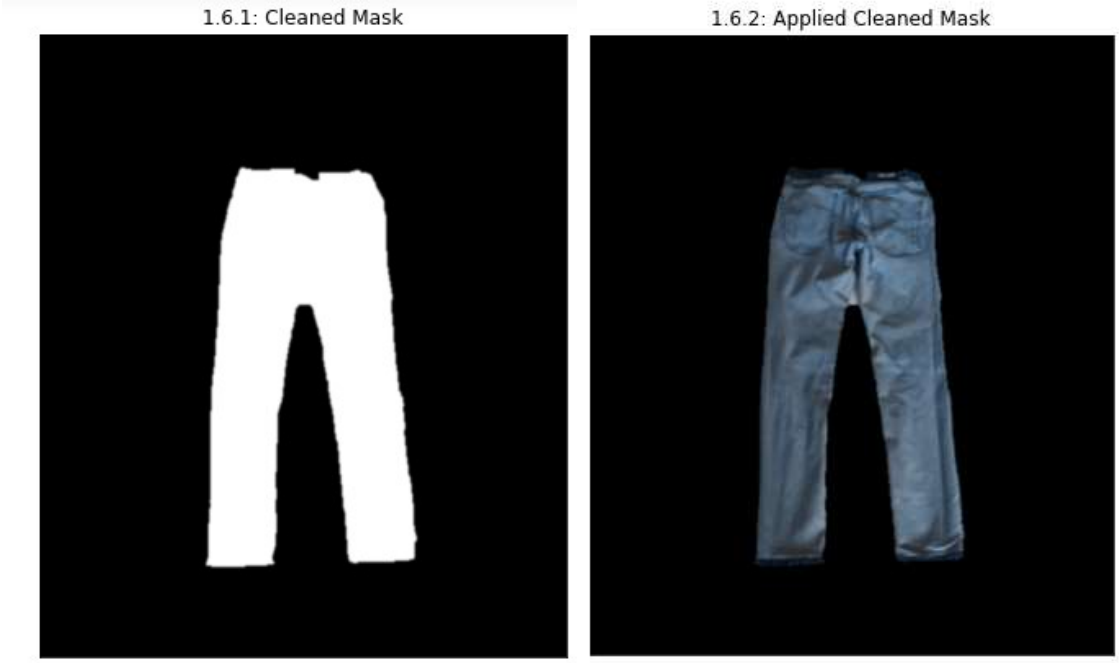


Fig. 8.7: A l'esquerra, mascara netejada de la peça, a la dreta imatge final aplicant la mascara netejada.

8.1.3. Parametrització de la peça

Amb la peça aïllada del fons, es pot passar a la parametrització d'aquesta, amb la intenció d'obtenir una parametrització semblant a aquella demostrada per Miller S. (7) L'objectiu d'aquest pas és determinar els diferents punts d'interès, com ara les cantonades i el contorn simplificat vistos en la **Fig. *8.8***, per a després utilitzar-les per a extreure informació de la peça.

Una de les característiques que es comparteix en moltes peces semblants és l'existència de diferents cantonades que delimiten l'estructura externa. Altres cantonades poden aparèixer si la peça presenta plecs inesperats, com ara en els camals. Així i tot, aquests entrarien dins de la categoria parcialment cap amunt, i parcialment cap avall, les quals no es contemplen en aquest treball.

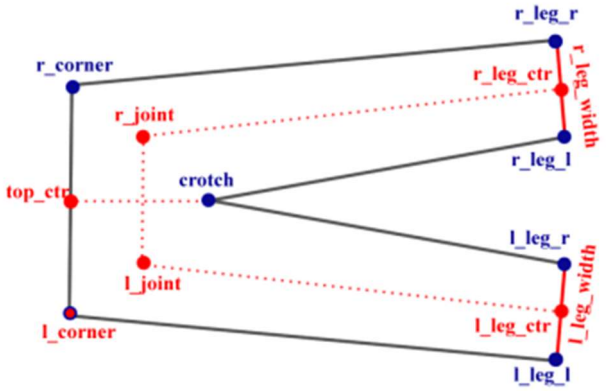


Fig. 8.8: Parametrització objectiu demostrada per Miller S. *(7)*

Per a detectar aquestes cantonades s'ha aplicat una detecció de cantonades Harris. Aquest algoritme serveix per a identificar les cantonades internes d'una imatge, definides com aquelles regions on existeix una variació del gradient en múltiples dimensions i direccions.

Fent ús de *OpenCV*, podem fer ús de la funció *cornerHarris*, la qual requereix de tres variables. Primerament trobem *blockSize*, el qual determina la mida dels veïns per a detectar les diferents cantonades; després trobem *ksize* referent al paràmetre d'obertura del filtre Sobel que s'utilitza i finalment k, el paràmetre lliure de l'equació *Harris*.

Fig. 8.9: Imatge resultat de l'aplicació del detector de cantonades Harris

Mitjançant l'experimentació, s'ha determinat que la millor combinació de paràmetres es troba composta per un *blockSize* de 23, una *ksize* de 9.0 i finalment una *k* de 0.1.

Aplicant directament l'algoritme a la màscara mostrada anteriorment, podem extreure la imatge mostrada en la **Fig. *8.9***. A més a més, s'aplicarà una dilatació en aquesta nova imatge per a ajuntar aquelles cantonades que es trobin massa juntes. Aquest pas és important en aquelles imatges on la zona central, on hi ha la bragueta, conte algun tipus de plec que provoca la detecció de dues o més cantonades. En aquests casos, la dilatació ajuntarà aquests punts perquè més endavant es puguin considerar com a un sol.

Binaritzant la imatge i aplicant una detecció d'objectes, podem aïllar i determinar cada un dels centres de cada cercle. Aquestes coordenades es faran servir més endavant, com a punts per a parametritzar la peça. Així i tot, a causa de la naturalesa de l'algoritme, els punts retornats no es troben ordenats, de manera que no es poden unir directament per a generar l'estructura.

Per a fer-ho, primerament es farà una detecció de vores, generant uns punts que es troben al voltant de tota la peça, però en aquest cas, es trobaran ordenats segons la seva continuïtat. **Fig. *8.10***

Fig. 8.10: Vores de la mascara aplicades a la imatge original

Partint d'aquests punts, es recorrerà cada un d'aquests i es determinarà la proximitat als centres de les cantonades, ordenant-les segons la proximitat relativa a l'ordre original de la vora.

Gràcies a aquest procés, podem connectar seqüencialment tots els punts, ara si, generant un contorn simplificat a partir de les diferents cantonades.

Tot i tenir ara les cantonades ordenades, és interessant observar com clarament existeixen punts extres que no corresponen a la parametrització esperada i, per tant, s'ha d'aplicar un sistema de simplificació per a reduir el nombre de punts. Agafant com a referència la **Fig. *8.11***, es pot observar com existeix un punt extra en el centre de la cintura, visualitzada per aquell lloc on la línia blava passa a ser grisa.

Fig. 8.11: Connexió dels diferents punts partint de les cantonades

El nombre ideal que delimita la peça és un conjunt de set punts i es poden com:

- Les dues cantonades superiors, esquerra i dreta.
- Les dues cantonades de cada camal
- El punt central, on es trobaria la bragueta.

Per tant, si un pantaló es pot parametritzar amb set punts, es considerarà que es tracta d'un pantaló. La simplificació realitzada en aquest cas correspondrà en l'eliminació dels punts a partir del producte escalar, com:

$$\frac{(\vec{x}_{i-1} - \vec{x}_{i-2})}{||\vec{x}_{i-1} - \vec{x}_{i-2}||} \cdot \frac{(\vec{x}_i - \vec{x}_{i-1})}{||\vec{x}_i - \vec{x}_{i-1}||} \leq t \qquad \forall i \geq 3 \qquad (8.5)$$

On *i é*s l'índex de cada punt existent en el conjunt de cantonades, *x é*s cada punt donat per l'índex *i,* i *t* una variable real, equivalent al màxim producte escalar acceptable. Si el producte escalar és superior a t, i, per tant, que les dues línies formades per $(\vec{x}_{i-1} - \vec{x}_{i-2})$ i $(\vec{x}_i - \vec{x}_{i-1})$ es trobin suficientment

alineades, s'eliminarà de la llista $\vec{x}_{i-1}$, transformant aquestes dues a una nova formada per $(\vec{x}_i - \vec{x}_{i-2})$.

A partir d'experimentació, s'ha determinat que el millor valor de *t* es troba en 0.7. Aplicant l'equació, es poden extreure les coordenades finals, visualitzades en la **Fig. *8.13***, on es pot veure com ara, en la cintura només existeix un segment.

Fig. 8.13: Cantonades finals del pantaló.

Fig. 8.12: Exemple addicional de la simplificació de cantonades

Un altre exemple on es veu més clarament aquesta simplificació es pot observar en la **Fig. *8.12***, on la pertorbació que apareix a l'esquerra a causa de la butxaca, queda completament eliminada i simplificada en la imatge de la dreta, un cop aplicada la simplificació.

A partir d'aquí, es determinarà si la peça en qüestió és o no un pantaló, segons el nombre retornat de punts.

8.1.4. Retall i transformació en la configuració final

Un cop assegurats que la peça en qüestió és la d'un pantaló, i que tenim un conjunt de set punts representant cada una de les cantonades més importants, podem passar al processament d'aquesta per a retallar i centrar la imatge en la zona de més interès, la zona superior sense camals.

Aquesta zona d'interès s'ha determinat a partir de l'observació de les diferències aparents entre cada tipus de pantaló.

Fig. 8.14: Quatre imatges dels camals referents als quatre estats que es volen determinar. D'esquerra a dreta, ReDo, ReUp, RiUp, RiDo.

Per exemple, la **Fig. *8.14*** mostra quatre camals diferents de quatre imatges corresponents a les quatre categories en les quals es vol classificar. Aquí es pot observar que s'hi pot extreure poca informació per a poder classificar amb precisió a una de les quatre categories d'interès. Es pot discernir, més o menys, entre aquell estat Revertit i Normal, pero no pas si es troben Cap per Amunt o Cap per Avall.

En canvi, en la **Fig. *8.15*** es poden observar diferents característiques que permeten la distinció clara entre els estats diferents, com ara les cremalleres, etiquetes, butxaques, textura de tela, i altres detalls. Per tant, l'algorisme de classificació serà més efectiu si es concentra en la part dels pantalons que realment aporta informació respecte a l'estat, i deixa de banda aquelles parts poc significatives.

Fig. 8.15: Quatre imatges de la zona central del pantaló referents als quatre estats que es volen determinar. D'esquerra a dreta, de dalt a avall, RiUp, RiDo, ReUp, ReDo

Per a obtenir aquestes imatges, s'ha determinat un nou nombre de punts a partir de les cantonades anteriors. Aquests nous punts queden reflectits en la **Fig. *8.16*** esquerra.

Els dos punts superiors de color rosa, fan referència als ja coneguts punts superiors del pantaló. Per a aconseguir aquests dos punts s'observaran les diferents línies que apareixen entre cada un dels punts ordenats. Les dues línies més llargues formaran part del contorn exterior dels camals, tal com es pot veure en la **Fig. *8.16*** centre.

La tercera i quarta línia segons longitud, formarà part del contorn interior dels camals, per tant, la cinquena serà aquella que delimiti el contorn de la cintura, on es troba la cremallera o botons, i

butxaques. Amb això es pot extreure els dos punts de color rosa a partir de la intersecció entre la línia de la cintura, i els dos dels camals.

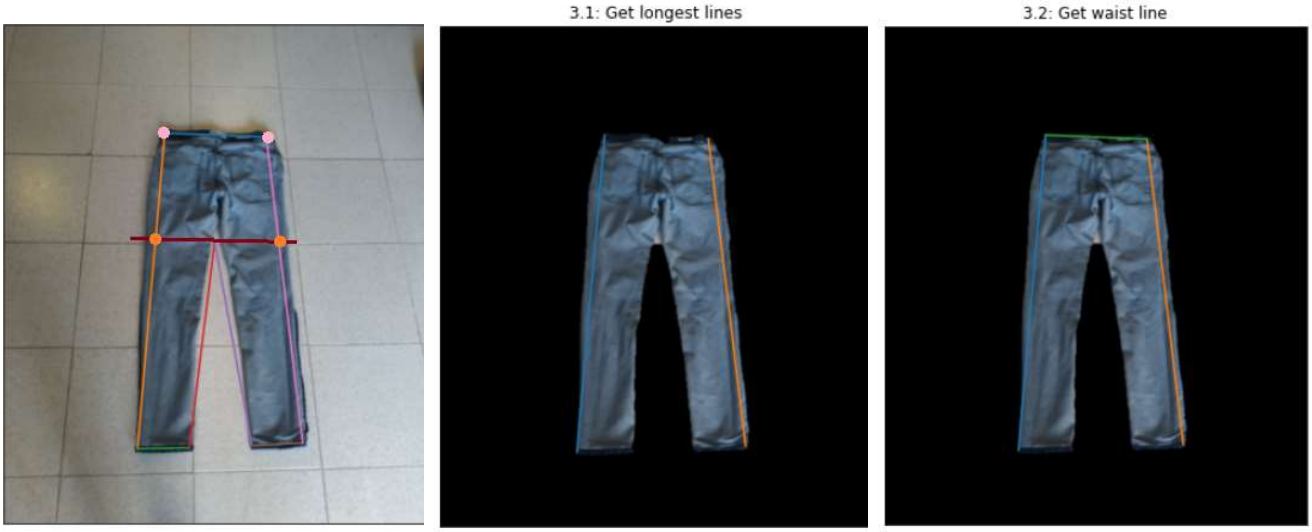


Fig. 8.16: D'esquerra a dreta, punts a trobar per a obtenir la nova estructura, línies mes llargues extretes a partir dels contorns de la imatge, línies mes llargues mes la línia de la cintura.

Trobats els punts rosa, es buscarà els punts taronges a partir de la intersecció d'una l'ínia paral·lela a la cintura, que passi pel punt de la bragueta o entrecuix, i les línies externes als camals. Unint aquests punts es pot trobar el requadre de la zona d'interès.

Fig. 8.17: Requadre final recorrent els punts d'interès

Finalment, s'aplica una deformació en la imatge original, a partir d'aquests nous quatre punts, transformant la imatge a una mida de 255x255, tal com es veu en la **Fig. *8.15***.

8.2. Objectiu 2: Classificació del pantaló en les quatre categories objectiu

Un cop detectada i classificada la peça en qüestió, i determinant que és un pantaló, es pot prosseguir amb la detecció i classificació dels pantalons en els quatre estats diferents, ***Reverse Up, Reverse Down, Right Up, Right Down.***

Aquesta tasca s'ha realitzat mitjançant l'entrenament d'una SVM (del anglés, *Support Vector Machine)*[1] a partir d'un model de bossa de paraules[2] creada dels descriptors extrets aplicant l'algoritme de *SIFT*[3]. (29)

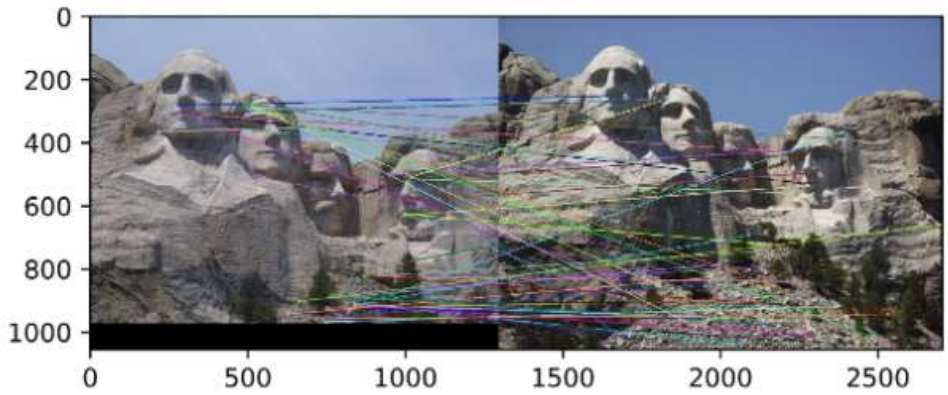


Fig. 8.18: Relació entre dues imatges a partir dels descriptors extrets mitjançant SIFT. *(50)*

[1] SVM o *Support Vector Machine* es un model d'aprenentatge autònom que intenta obtenir un pla de dimensió N que separi les diferents categories d'una manera clara. Per exemple, si es tinguéssim dos categories, una línia podria separar les dues categories de manera clara.

[2] Bossa de paraules, o *Bag of Words,* consisteix en un mètode el qual cada una de les imatges es troba representada sense importar l'estructura de la informació interna, en aquest cas, mitjançant els diferents descriptors que el componen.

[3] *SIFT,* o *Scale Invariant Feature Transform é*s un algoritme dissenyat per ***David G. Lowe*** en "*Distinctive Image Features from Scale-Invariant Keypoints"* (49) que permet l'extracció de diferents descriptors altament diferenciats per a relacionar imatges les unes entre les altres.

Gràcies a la similitud general que les imatges processades presenten, mantenint aquelles zones que aporten més informació en el centre de la fotografia, es pot fer ús d'aquest algoritme per a generar els diferents descriptors de cada categoria, i entrenar la SVM a partir d'aquí.

El primer problema que ens trobem, però és, tal com es va demostrar en el segon intent previ, la falta d'imatges per a entrenar la SVM. Un dels primers passos doncs és augmentar el nombre d'imatges existents, sigui de manera artificial o de manera real.

Fig. 8.19: A l'esquerra imatge original, a la dreta imatge generada amb *Augmentor*.

Mesclant els dos datasets realitzats en l'intent previ, i en el mètode actual, i processant les 160 imatges, s'extreuen 100 imatges retallades i transformades a la forma demostrada en la **Fig. *8.15***. A partir d'aquí, mitjançant la llibreria *d'Augmentor*, es genera un conjunt de 500 imatges les quals s'utilitzaran per a l'entrenament. Les prèvies 100 imatges originals, però, es faran servir per a les proves i visualització de resultats.

Per a crear el model de bossa de paraules, primer es trobaran els diferents descriptors mitjançant l'aplicació de l'algoritme SIFT, proporcionada per *OpenCV*. La quantitat de descriptors que apareixen és molt elevada per a poder fer l'entrenament a partir d'aquests, per tant, és important poder agrupar-los.

El mètode aplicat generalment és l'aplicació de "*k centers*" amb k sent el nombre de categories * 10, en aquest cas k = 40. Aquest algoritme de proximitat permet la selecció d'específics descriptors els quals minimitzin la distància entre els altres descriptors. En conseqüència, se seleccionaran els 40 millors descriptors, els quals gràficament, seran aquells centres de les diferents agrupacions que s'han format de manera natural.

A partir d'aquí es poden generar els diferents histogrames que representaran la imatge. Per a fer-ho, primer es trobaran cada un dels descriptors, i després es generarà un histograma de mida k. Aquest histograma serà omplert a partir de les prediccions del model "*k centers*" i normalitzat per a poder ser utilitzat en la SVM.

Un cop entrenada la SVM a partir del histograma i les etiquetes originals, es pot realitzar l'avaluació amb les imatges originals per a observar la precisió de l'algoritme de classificació.

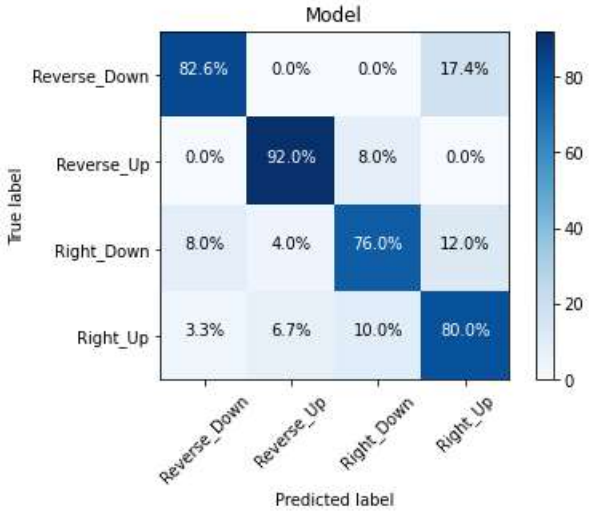


Fig. 8.20: Resultats de les prediccions de la SVM entrenada.

Tal i com es pot veure en la **Fig. *8.20***, aquest model pot classificar correctament les diferents imatges, amb una precisió del 81.5%. Aquests resultats són molt millors que aquells demostrats en els intents previs.

9. Anàlisis de resultats i propostes de millora

9.1. Objectiu 1

Un cop explicades les diferents parts de l'algoritme, és interessant observar els resultats obtinguts, els problemes que es poden trobar, i les diferents propostes de millora aplicables en els treballs futurs.

Durant el transcurs de la primera part, existeixen múltiples punts que provoquen la fallida en la detecció i classificació. Aquests es troben repartits entre els diferents algoritmes aplicats, augmentant acumulativament la probabilitat que es detecti incorrectament la peça de roba.

Per a comprovar el resultat de precisió aconseguida en aquesta primera part, s'ha realitzat l'execució del codi d'extracció dels pantalons en 80 imatges originals (vist en la **Taula *4*:** Percentatges de precisió a partir de les 80 imatges originals.) i en un conjunt de 500 imatges augmentades de les 80 anteriors (vist en la **Taula 5:** Percentatges de precisió a partir de 500 imatges generades partint de les originals.)

Aquesta precisió queda determinada segons el percentatge de peces que han quedat correctament separades del fons, sense comptar aquells casos on no han quedat correctament separades, ja sigui perquè no les ha classificat com a pantalons, o ha creat una parametrització resultant amb una imatge incorrecta.

Taula 4: Percentatges de precisió a partir de les 80 imatges originals.

Reverse Down	80%	***Reverse Up***	85%
Right Down	85%	***Right Up***	80%

TOTAL	82.5%

Taula 5: Percentatges de precisió a partir de 500 imatges generades partint de les originals.

Reverse Down	57.7%	***Reverse Up***	56.7%
Right Down	73.0%	***Right Up***	68.1%

TOTAL	63.4%

Es determina que l'algoritme té una precisió del 63.4%. Per tant, es considera un algoritme efectiu, amb molt de marge de millora aplicable mitjançant les diferents propostes que es veuran a continuació.

A més a més, per a tenir una visió més clara dels resultats, s'ha passat l'algoritme per un conjunt d'imatges que no s'han fet servir en l'ajustament dels diferents paràmetres de l'algoritme. Ajustant deliberadament una part de l'algoritme, amb el que es millora la binarització de la imatge, i que s'explica més endavant amb més detall , s'obtenen els resultats de la **Taula *6***.

Taula 6: Percentatges de precisió a partir de 80 imatges alternatives.

Reverse Down	35%	***Reverse Up***	40%
Right Down	25%	***Right Up***	35%

TOTAL	33.75%

Es pot observar com aquests són molt més baixos, estan per sota del 50%, cosa que dona a qüestionar-se l'efectivitat de l'algoritme. Aquest problema i la seva possible millora es comentarà més endavant.

Finalment, s'ha vist el temps d'execució d'aquesta primera part. El temps mitjà és de 1.22s, sent la imatge processada més ràpida en 1.16s i la més lenta en 1.31s.

Taula 7: Temps d'execució de la part A

	Mitjà	Mínim	Màxim
Temps d'execució	1.22s	1.16s	1.31s

El temps mínim es dona quan la imatge pot ser classificada com a pantaló directament a partir del primer mètode d'extracció de fons, mentre que el temps màxim ve donat quan el procés s'ha de repetir amb un segon mètode.

A continuació s'observaran les diferents problemàtiques del codi que determinen aquestes precisions i els temps de processament, juntament amb l'aportació de possibles millores aplicables a l'algoritme en treballs futurs.

9.1.1. Extracció del fons

Per iniciar l'anàlisi de resultats s'ha dut a terme una observació dels diferents punts més restrictius en l'algoritme. El primer de tots, doncs, és l'extracció del fons, i com era d'esperar, ha demostrat ser una zona clau en la detecció del pantaló en si.

Es requereix la màscara extreta en aquest primer pas per a executar les diferents parts del codi, com ara la detecció de cantonades, i verificar més endavant si la peça és un pantaló, retallar segons els paràmetres trobats, i exportar-la per a poder classificar-la. Malauradament, però, a causa de la poca

robustesa de l'algoritme per a aconseguir les arrugues, i la diversitat de les imatges, es pot donar el cas que les peces no quedin correctament extretes, i, per tant, tots els passos següents fallin.

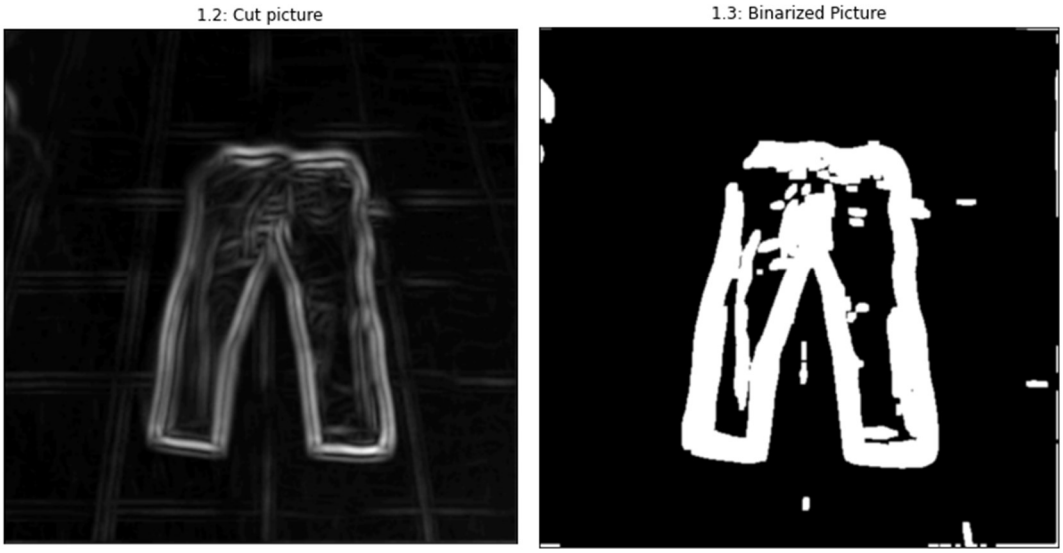


Fig. 9.2: Exemple de problemàtica en l'extracció del fons. A l'esquerra arrugues extretes de la imatge, a la dreta, binarització d'aquestes.

En la **Fig. *9.2*** es pot observar un cas on aquesta extracció del fons ha fallat. En aquesta imatge, la detecció de les diferents arrugues, tot i trobar correctament un gran nombre d'aquestes, no ha aconseguit reaccionar en alguns dels punts claus, més concretament en la cantonada esquerra alta, provocant que en la seva pròxima binarització, aquesta quedi oberta.

Fig. 9.1: Imatge original usada en la **Fig. 9.2**

Tal i com es pot veure en la imatge original (**Fig. *9.1***), la peça en questió conté un conjunt d'ombres, juntament amb una semblança amb el seu fons, en la cantonada esquerra superior, donada pel desplegament de la butxaca interna, provocant que el filtre aplicat no reaccioni apropiadament en comparació al seu entorn.

Un cop aplicada la binarització, es crea una obertura en aquella zona, la qual no permet la detecció de tot el contorn. Això crea una reacció en cadena, fent fallar els passos que venen a continuació.

Altres vegades es pot donar el cas on la peça en qüestió queda separada del fons, juntament amb una porció d'aquesta. Això provoca que la parametrització aplicada es trobi treballant amb parts que no formen part d'un pantaló, i que, per tant, crea la mateixa reacció en cadena.

Això es dona principalment a causa d'una barreja entre la binarització i l'aplicació de l'algorimte *grabCut*. Tal com es pot veure en la **Fig. *9.3***, la peça de roba es troba correctament en el centre de la imatge, i s'espera, un cop passat l'algoritme de *grabCut*, que aïlli completament el pantaló. Malauradament, això no es compleix, i, per tant, genera una imatge amb parts del fons incloses.

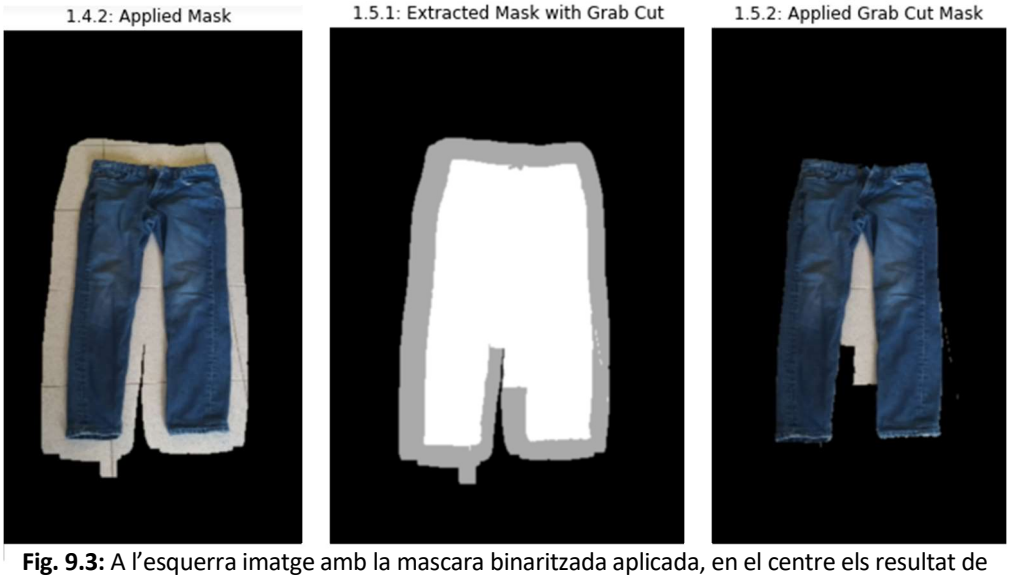


Fig. 9.3: A l'esquerra imatge amb la mascara binaritzada aplicada, en el centre els resultat de *grabCut*, a la dreta, imatge dels pantalons amb fons entre camals.

Aquesta mala delimitació de la peça, provoca una generació de cantonades on no hi hauria i per tant crea la parametrització i classificació de manera incorrecta.

Fixant-nos només en aquesta primera part de separació de la peça de roba del fons, es poden observar els següents resultats a la **Taula *8***. Aquests percentatges fan referència a la quantitat d'imatges que han estat correctament separades del seu fons, partint de les 80 imatges originals.

Taula 8: Percentatge de precisió d'imatges correctament separades a partir de les 80 imatges originals.

Reverse Down	90%	***Reverse Up***	95%
Right Down	95%	***Right Up***	85%

TOTAL	91.25%

És interessant, però, observar com reacciona en un conjunt més elevat d'imatges. És per això que s'ha fet servir de la llibreria *Augmentor* per a generar 500 imatges noves, aplicant l'algoritme a aquestes, i veient els resultats en la **Taula *9***.

Taula 9: Percentatge de precisió d'imatges correctament separades a partir de les 500 imatges augmentades.

Reverse Down	80%	***Reverse Up***	78.35%
Right Down	93.9%	***Right Up***	87.93%

TOTAL	84.6%

Es pot observar com la precisió ha disminuït, però es manté sent considerablement elevada, al voltant del 85%.

Vistos els resultats i aquells casos on l'algoritme falla, s'ha determinat que en moltes situacions es determina correctament la peça a partir de les arrugues, i que, és en el pas de l'aplicació de l'algoritme de *grabCut* que apareixen problemes.

Seria interessant doncs, buscar una implementació on l'aplicació del algoritme *grabCut* no doni fallades, o en cas contrari, buscar una alternativa que funcioni mitjançant un altre mètode.

Existeix un problema en l'aplicació d'aquest mètode. Un dels passos consisteix en la binarització de la imatge extreta mitjançant la detecció d'arrugues. Aquesta binarització, idealment separaria relativament els pantalons del seu fons, de manera que, mitjançant l'algoritme de *grabCut*, es pot obtenir una imatge més refinada.

Aquest pas, però, no separa de manera automàtica el fons de tota mena d'imatges. Un dels paràmetres de la binarització consisteix en l'especificació del llindar, el qual genera imatges molt diferents segons

el tipus de fons. Per posar com a exemple el conjunt d'imatges utilitzades, les 80 originals la qual s'ha fet servir com a referencia per a determinar els diferents mètodes, dona millors resultats amb un valor de llindar del 7.5, mentre que amb un conjunt d'imatges que no s'ha utilitzat fins aquest punt, les 80 alternatives, dona millors resultats amb un valor de 4.

Això el posiciona com un dels punts més importants per al funcionament de l'algoritme, sent molt rellevant per a millorar l'algoritme en treballs futurs.

Una de les propostes per a extreure el fons de la imatge és fer ús de tècniques com les demostrades per Inaba M. i Yamazaki K. (5). Aquest mètode nou consistiria en una variació del proposat mitjançant el filtre de Gabor, però inserint una SVM per a determinar la zona d'interès per l'algoritme de grabCut, proporcionant una extracció de la màscara més dinàmica i oberta a variacions que puguin aparèixer.

9.1.2. Detecció de cantonades

Després del pas explicat a l'apartat anterior, es troba la detecció de cantonades, simplificació de punts, i parametrització inicial, tot determinant si la peça és un pantaló o no.

És aquí on apareix la segona problemàtica més comuna. Moltes de les imatges correctament separades fallen en aquest punt a causa del mètode utilitzat per a la parametrització, o directament, en la detecció de les diferents cantonades.

Tal com es pot veure en la **Fig. *9.4***, la peça en qüestió conté els camals molt separats, juntament amb un plec a la part superior del camal de l'esquerra, provocant que es detectin dues cantonades, en comptes d'una.

Fig. 9.4: Exemple de problemàtica en la parametrització de la peça. A l'esquerra màscara aplicada a la imatge original, aïllant la peça, a la dreta, contorns generats a partir de les cantonades.

Ja que aquest plec és considerablement llarg, amb la cantonada formada per un angle gran, la simplificació de contorns no pot detectar aquest punt com un sense rellevància, i, per tant, no la pot eliminar del conjunt de cantonades trobades.

Altres vegades es dona el cas contrari, on no es detecten suficients cantonades per a realitzar la correcta parametrització, i provocant que no es pugui detectar correctament el pantaló.

Per exemple, en la **Fig. *9.5*** es dona el cas que les cantonades, inicialment, es troben correctament detectades. En un dels passos següents, però, s'aplica una simplificació dels punts aplicant una dilatació en la imatge i les cantonades del camal dret inferior queden enganxades, provocant que en la detecció dels centres es considerin com a una única cantonada.

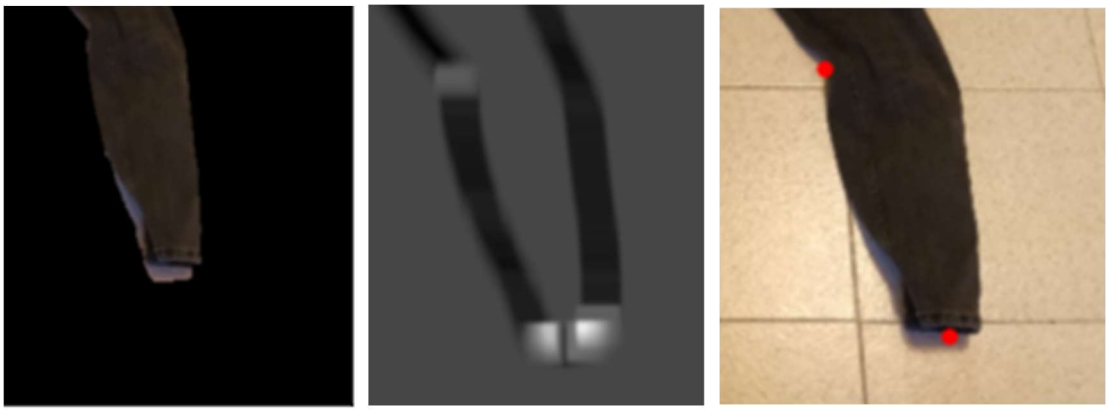


Fig. 9.5: Exemple de problemàtica en la detecció de cantonades. A l'esquerra mascara aplicada a la imatge original, al centre cantonades detectades, a la dreta, centre final de la cantonada.

Això fa doncs que no es trobin set cantonades i per tant que no es consideri un pantaló.

Taula 10: Percentatge de precisió dels pantalons correctament classificats com a tal a partir de les 500 imatges augmentades. Relatiu és el percentatge partint de les imatges ja correctament classificades com a pantalons, acumulat és tenint en compte els procediments anteriors.

	Relatiu	**Acumulat**
Reverse Down	72.2%	57.7%
Right Down	77.7%	73.0%

	Relatiu	**Acumulat**
Reverse Up	72.38%	56.7%
Right Up	77.4%	68.1%

TOTAL	74.92%	63.4%

Es pot observar a partir d'aquí com la precisió de l'algoritme disminueix segons es va avançant entre els diferents passos, i que la detecció de cantonades aplicada treballa pitjor que l'extracció de fons.

Per a millorar aquest sistema, s'hauria de realitzar un canvi en com es realitza la parametrització. És d'interès aplicar un algoritme similar al proposat per Miller S. (7), on mitjançant un sistema de pesos i un algoritme iteratiu, es va aproximant cada un dels punts a l'estructura del pantaló. Això permetria una millor modelització de la peça, sent més acurada, tot perdent velocitat en el procés.

9.1.3. Conseqüències dels problemes

Ja sigui perquè l'extracció del fons ha fallat, o un problema en la detecció de cantonades, en diversos casos poden aparèixer imatges detectades com a pantaló, tot i no contenir les característiques esperades. Això provoca que algunes de les imatges finals no continguin suficient informació per a poder ser utilitzada més endavant en la seva categorització.

Fig. 9.6: Tres exemples d'imatges classificades com a pantaló incorrectament.

Tal com es pot veure en la **Fig. *9.6***, algunes de les peces de roba s'han trobat classificades com a pantaló, tot contenint una quantitat molt limitada d'informació rellevant per a la seva classificació. Es veuen, a més a més, deformades en formes no habituals, tenint zones negres sense informació.

Aquestes han de ser manualment eliminades del conjunt d'imatges abans del seu entrenament per a assegurar uns resultats precisos. Un sistema de millora aplicable és aconseguir un tercer algoritme capaç de detectar aquests errors i executar algun tipus de variació per a arreglar o obtenir un resultat viable.

Partint de les 500 imatges generades prèviament, s'han trobat 2 d'aquestes incorrectament classificades, resultant en un 0.4% d'error amb falsos positius.

9.1.4. Conclusió de resultats

Vistos aquests resultats està clar com el sistema conté múltiples punts on es poden aplicar millores, ja mencionades en les seves relatives parts, i que, per tant, és aquest primer objectiu que conté el màxim marge de millora aplicable.

Els resultats de precisió observats són prou alts per a considerar un resultat satisfactori i que es pugui continuar amb l'objectiu 2. Malgrat tot, el sistema aplicat conté algunes parts arbitràries que seria interessant modificar.

Un d'aquests punts consisteix en el paràmetre mencionat en l'extracció del fons, aquell que ens permet binaritzar correctament la imatge, el qual ha de ser manualment modificat segons la base de dades d'imatges utilitzada.

A més, és important mencionar que en un gran nombre de referències, es fa servir un conjunt d'imatges amb un fons que conté un perfil RGB molt definit i conegut. En alguns casos, els pantalons en qüestió es troben en un fons completament verd o completament blau, que permet la fàcil extracció del pantaló a analitzar i d'aquesta manera prosseguir a la seva categorització.

Amb tot això comentat, es considera que aquest objectiu s'ha complert satisfactòriament donades les circumstàncies, tot i que, per a treballs futurs, és aquest el qual pot ser àmpliament millorat a partir d'altres mètodes.

9.1.5. Altres millores aplicables

Els canvis que s'han mencionat anteriorment són rellevants per a millorar l'algoritme actual. Tot i això, existeixen canvis que es poden aplicar, i altres camins que es poden seguir per a obtenir resultats millors, variant o completament refent l'algoritme proposat.

És d'especial interès, per a treballs futurs, reduir el nombre d'assumpcions i treballar sobre d'unes noves bases, augmentant les possibilitats de l'algoritme de ser aplicat en un espai real. La complexitat d'aquest augmentaria considerablement tot aportant uns resultats més genèrics.

Una primera proposta consisteix a fer ús *d'active sensing. Active sensing* consisteix en l'aplicació directa d'un robot o mecanisme controlat per a la manipulació de la peça directament per a transformar-la en una posició que faciliti la seva classificació.

Fent ús d'un mètode d'aquest tipus permetria ampliar el nombre de característiques que es poden obtenir d'una peça estàtica, tot augmentant la seva complexitat d'aplicació i recursos de

processament. A més a més, per a aplicar-ho es requerirà un robot capaç de dur a terme aquesta manipulació.

9.2. Objectiu 2

Aquest segon pas, centrat en la classificació dels diferents pantalons en una de les quatre categories especificades en els objectius conté poc marge de millora com a tal.

A causa de les assumpcions i dels resultats del pas anterior, aquest compleix l'objectiu de classificació amb resultats de precisió molt elevats, tal com es pot veure en la **Taula *11***.

Taula 11: Percentatge de precisió en la classificació del pantaló en cada una de les categories esperades.

Reverse Down	82.6%	***Reverse Up***	92%
Right Down	76%	***Right Up***	80%

TOTAL	81.5%

El temps de processament es troba principalment afectat per l'obtenció dels diferents descriptors de la imatge, donant com a resultat els temps de la **Taula *12***.

Taula 12: Temps d'execució de la classificació dels pantalons

	Mitjà	Mínim	Màxim
Temps d'execució	1.20s	0.01s	4.02s

9.2.1. Conclusió de resultats

A partir d'aquí es determina que els resultats obtinguts són òptims i eficients, amb marge de millora. Tot i això, aquest marge de millora apareix quan es varia el mètode aplicat en el segon objectiu, i, per tant, en cas d'intentar millorar, podria ser interessant investigar altres característiques del pantaló que puguin ser d'interès, i no només la zona seleccionada.

9.3. Combinació dels objectius

Un cop observats cada un dels passos de manera independent i determinant les diferents millores aplicables per a treballs futurs en cada un dels mètodes, pot ser interessant observar com reacciona si

s'ajunten els dos mètodes. Això va més enllà dels objectius formulats inicialment, però principal, però observar els resultats de manera conjunta dona una millor visió del funcionament de l'algoritme de manera conjunta.

En la **Fig. 9.7** es pot veure la matriu de confusió del sistema combinat, i, mitjançant els diferents percentatges, es pot determinar la precisió del sistema.

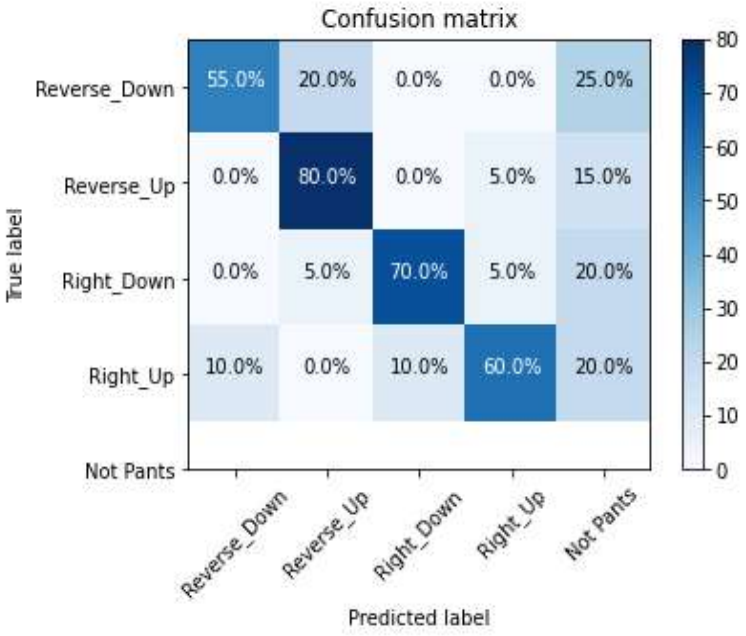


Fig. 9.7: Matriu de confusió del sistema final

En l'última columna es poden trobar aquelles imatges que no s'han classificat com a pantaló. En aquest conjunt d'imatges, totes són pantalons, i, per tant, aquelles classificades com a *Not Pants*, es troben incorrectament classificades.

Taula 13: Percentatge de precisió en la classificació del pantaló en cada una de les categories

Reverse Down	55%
Right Down	70%
No Pants	20%

Reverse Up	80%
Right Up	60%

TOTAL	66.25%

Taula 14: Temps d'execució de l'algoritme

	Mitjà	Mínim	Màxim
Temps d'execució	2.45s	1.08s	5.30s

Com era d'esperar, tot i mantenir-se baixos, els temps d'execució han augmentat equitativament a la quantitat de codi que s'està executant.

És interessant observar els resultats a partir de les imatges augmentades, visibles en la **Fig. *9.8***.

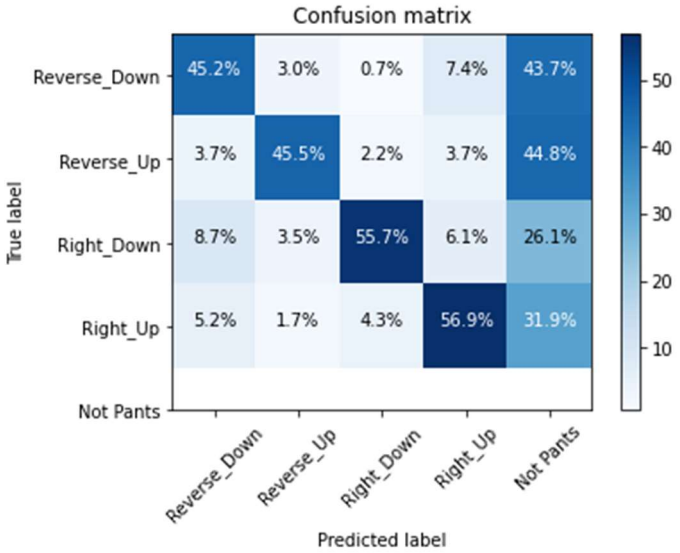


Fig. 9.8: Matriu de confusió a partir de les imatges augmentades

Taula 15: Percentatge de precisió en la classificació del conjunt augmentat de pantalons en cada una de les categories

Reverse Down	45.2%
Right Down	55.7%
No Pants	36.6%

Reverse Up	45.5%
Right Up	56.9%

TOTAL	50.82%

Taula 16: Temps d'execució de l'algoritme

	Mitjà	Mínim	Màxim
Temps d'execució	2.25s	1.06s	7.45s

Vist el percentatge de precisió de l'algoritme, de 50.82%, l'algoritme combinat no és gaire efectiu en determinar la peça i la seva categoria. En aquest cas el problema principal prové de l'existència de *No pants,* la qual fa baixar molt els percentatges de precisió.

Una opció ja coneguda per a millorar el sistema és fer ús dels canvis suggerits anteriorment en aquest mateix apartat. És important, però, mencionar que aquest sistema es troba incomplet en diferents sectors, que poden ser millorats en treballs futurs.

Una part crucial ja comentada anteriorment, és l'existència de les diferents assumpcions que es presenten en l'inici del treball. Aquests plantejaments fan que la implementació de l'algoritme dins del terreny real, mitjançant un robot domèstic, o simplement a partir d'una càmera remota, no sigui possible.

Seria interessant, doncs, en treballs futurs, fer ús dels coneixements obtinguts durant la realització d'aquest treball, per a crear un mètode amb reduïdes assumpcions, i que pugui detectar l'orientació de la peça, ja es consideri de manera completa, o parcialment de dins cap enfora.

Aquesta millora ja seria un pas molt rellevant per a poder fer ús de l'algoritme d'una manera més efectiva en terreny real. Altres objectius que es podrien plantejar són:

- Millora en el temps de processament d'imatge i predicció, per aconseguir un sistema a temps real
- Fer ús d'una combinació d'aprenentatge autònom i visió per computador per a millorar la detecció de la peça dins d'un entorn natural
- Augmentar les capacitats de l'algoritme per a treballar amb múltiples peces de roba a la vegada
- Augmentar la robustesa de les diferents seccions perquè s'adaptin a les peces de roba
- Augmentar el nombre de categories de peça de roba que són detectables per l'algoritme o
- Buscar un mètode general per a determinar l'orientació sense coneixement previ de la peça en qüestió.

L'article escrit per Ramisa A. (30) proposa una metodologia que permet la detecció de les peces de roba, juntament amb punts claus en elles, mitjançant una combinació de la detecció d'aparença d'una peça de roba considerant a més informació tridimensional per a detectar les parts que siguin d'interès per on agafar la peça.

Per a treballs futurs, podria ser interessant fer una anàlisi del mètode de l'article per a ampliar i aplicar les diferents deteccions que es duen a terme, al problema d'aquest treball, probablement millorant el sistema presentat en aquest treball per un de més robust, descartant algunes de les assumpcions d'aquest treball.

10. Anàlisi de l'impacte ambiental

A causa de la naturalesa d'aquest treball, l'impacte ambiental que pot aparèixer a conseqüència de tant el funcionament normal, com ara possibles avaries o accidents és mínim.

Ja que s'ha realitzat un codi, sense la seva aplicació al terreny mitjançant robots, el funcionament d'aquest quedarà limitat en un ordinador, sigui de sobretaula o portàtil, i, per tant, qualsevol mal funcionament provocarà com a màxim un alentiment de l'ordinador, i com a conseqüència un consum energètic més elevat.

Apareix doncs un impacte indirecte provocat pel funcionament d'un ordinador, ja es trobi el codi executant-se o no. A més a més, el codi en qüestió s'executa relativament de pressa i, per tant, l'impacte que apareix únicament durant l'execució del codi és mínim.

Finalment, la creació del robot domèstic per a dur a terme la tasca d'identificació, generaria un impacte de carboni molt més elevat que l'execució del codi, sent molt més rellevant en l'impacte ambiental que el mencionat anteriorment. Així i tot, aquests robots són generalment creats per a realitzar múltiples tasques, i per tant, aquesta petjada de carboni quedaria més repartida.

Conclusions

Durant el transcurs d'aquest treball s'ha presentat un algoritme de visió per computador per a la classificació de peces de roba a partir de les diferents tecnologies existents i els mètodes aplicats amb anterioritat. Aquest algoritme permet l'avanç de la investigació i recerca dins del món de la robòtica assistida, tot contemplant les diferents millores aplicables en treballs futurs, i el camí a recórrer per a ampliar la viabilitat del mètode en una aplicació real.

Els objectius proposats han estat complerts mitjançant l'aplicació de variacions de metodologies demostrades funcionals en la recerca prèvia, tot aportant diferents punts alternatius per a millorar la precisió de classificació i la velocitat de processament. Aquests objectius consistien en la detecció i classificació d'una peça de roba en la categoria de pantalons, i en la classificació d'aquests pantalons en quatre posicions possibles, *Reverse Up, Reverse Down, Right Up, i Right Down.*

A més a més, un cop mostrats els diferents resultats de la feina presentada, s'han localitzat les diferents parts del codi que tenen tendència a fallar, tot demostrant els diferents mètodes que es poden aplicar per a millorar el funcionament de l'algoritme. Addicionalment, es mencionen camins alternatius i bifurcacions aplicables en l'algoritme per a treballs futurs, els quals permetrien obtenir una solució aplicable mitjançant robots reals.

El primer objectiu s'ha complert amb un percentatge de satisfacció moderat, determinant si la peça en qüestió és un pantaló amb una certesa del 75%. Aquest és prou elevat perquè es pugui considerar satisfactori, però a causa dels problemes que s'han mencionat, conté marge de millora.

El segon objectiu s'ha complert completament amb un percentatge de precisió alt, del 81.5%, i, per tant, es considera que s'han obtingut uns resultats bons amb marge de millora per a treballs futurs.

Finalment però, s'ha observat com el mètode combinat no dona molt bons resultats, amb una precisió del 50.82%, i per tant es determina que combinar els dos mètodes no és una manera efectiva d'usar-lo.

La realització d'aquest treball ha requerit un aprenentatge constant de diferents codis, fórmules matemàtiques i algoritmes per a poder ser aplicats d'una manera coherent i efectiva. Això, juntament amb la limitació temporal que representa la realització d'un treball de fi de grau, fa que aquest treball es pugui considerar exitós i satisfactori.

Pressupost i/o Anàlisi Econòmica

A continuació es pot trobar, de manera desglossada, els diferents costos en la realització d'aquest treball.

Taula 17: Cost del treball desglossat en les diferents activitats

ACTIVITAT	DESGLOSSAMENT	HORES	COST	COST TOTAL
Aprenentatge Previ		**60h**	**20,00 €**	**1.200,00 €**
Adquisició de dades		**40h**	**15,00 €**	**600,00 €**
Intent Previ 1		**65h**	**-**	**812,50 €**
	Programació	40h	20,00 €	800,00 €
	Aprenentatge del model*	25h	0,50 €	12,50 €
Intent Previ 2		**90h**	**-**	**1.215,00 €**
	Programació	60h	20,00 €	1.200,00 €
	Aprenentatge del mode*l	30h	0,50 €	15,00 €
Objectiu A		**110h**	**-**	**2.200,00 €**
	Programació	110h	20,00 €	2.200,00 €
Objectiu B		**55h**	**-**	**612,50 €**
	Programació	30h	20,00 €	600,00 €
	Aprenentatge del model*	25h	0,50 €	12,50 €
Altres Intents		40h	20,00 €	800,00 €
Redacció de la memòria		142h	20,00 €	2.840,00 €
TOTAL		**602h**	**-**	**10.280,00 €**

*El cost de l'aprenentatge del model ve donat principalment pel cost energètic mitjà d'un ordinador de sobretaula.

Bibliografia

1. *Perception for the manipulation of socks.* **Chuan Wang, Ping, et al.** San Francisco, CA, USA : IEEE, 2011.

2. *Service robotics (the rise and bloom of service robots) [tc spotlight].* **Moradi, Hadi, et al.** 3, s.l. : IEEE, 2013, Vol. 20.

3. **Bloom, D.E. i Luca, D.L.** *Chapter 1 - The Global Demography of Aging: Facts, Explanations, Future,.* s.l. : Handbook of the Economics of Population Aging, 2016.

4. *Planning strategy for putting away laundry - isolating and unfolding task.* **Kakikura, Masayoshi i Kaneko, M.** Fukuoka, Japan : IEEE, 2002.

5. *A Cloth Detection Method based on Image Wrinkle Feature for Daily Assistive Robots.* **Yamazaki, Kimitoshi i Inaba, Masayuki.** Yokohama, Japan : Conference on Machine Vision Applications, 2009.

6. *Pose and category recognition of highly deformable objects using deep learning.* **Mariolis, Ioannis, et al.** Istanbul, Turkey : IEEE, 2015.

7. *Parametrized shape models for clothing.* **Miller, Stephen, et al.** Shanghai, China : IEEE, 2011.

8. *Garment perception and its folding using a dual-arm robot.* **Stria, Jan, et al.** Chicago, IL, USA : IEEE, 2014.

9. *Perception of cloth in assistive robotic manipulation tasks.* **Jiménez, Pablo i Torras, Carme.** s.l. : Natural Computing, 2020, Vol. 19.

10. **Formación, Deusto.** ¿Qué es el lenguaje C? *Dusto Formación.* [En línia] 23 / 08 / 2021. [Data: 26 / 04 / 2022.] https://www.deustoformacion.com/blog/programacion-diseno-web/que-es-lenguaje-c.

11. **R.** The R Project for Statistical Computing. *R.* [En línia] [Data: 26 / 04 / 2022.] https://www.r-project.org/.

12. **MathWorks.** ***Matemáticas. Gráficas. Programación.*** **[En línia] [Data: 26 / 04 / 2022.] https://es.mathworks.com/products/matlab.html.**

13. Foundation, Python Software. Python. ***Python.*** **[En línia] [Data: 26 / 04 / 2022.] https://www.python.org/.**

14. Anaconda. Anaconda. [En línia] [Data: 27 / 04 / 2022.] https://www.anaconda.com/.

15. Team, Anaconda. Getting Started with GPU Computing in Anaconda. *Anaconda.* [En línia] 30 / 09 / 2017. [Data: 27 / 04 / 2022.] https://www.anaconda.com/blog/getting-started-with-gpu-computing-in-anaconda.

16. Jupyter. Jupyter Notebook. *Jupyter.* [En línia] [Data: 27 / 04 / 2022.] https://jupyter.org/.

17. Front Page. *Kaggle.* [En línia] [Data: 27 / 04 / 2022.] https://www.kaggle.com/.

18. Brains, Google. TensorFlow. [En línia] [Data: 26 / 04 / 2022.] https://www.tensorflow.org/about.

19. Keras. *Keras.* [En línia] [Data: 26 / 04 / 2022.] https://keras.io/.

20. Dancuk, Milica. PyTorch vs TensorFlow: In-Depth Comparison. *PhoenixNap.* [En línia] 21 / 02 / 2021. [Data: 26 / 04 / 2022.] https://phoenixnap.com/blog/pytorch-vs-tensorflow.

21. svenkatsai123. Why TensorFlow is So Popular – Tensorflow Features. *GeeksforGeeks.* [En línia] 27 / 05 / 2021. [Data: 26 / 04 / 2022.] https://www.geeksforgeeks.org/why-tensorflow-is-so-popular-tensorflow-features/.

22. AI, Facebook. PyTorch. [En línia] [Data: 26 / 04 / 2022.] https://pytorch.org/.

23. Nvidia. Nvidia Developer. [En línia] [Data: 26 / 04 / 2022.] https://developer.nvidia.com/cuda-zone.

24. OpenCV. Front Page. *OpenCV.* [En línia] https://opencv.org/.

25. mdbloice. Augmentor. *GitHub.* [En línia] [Data: 27 / 04 / 2022.] https://github.com/mdbloice/Augmentor.

26. SkLearn. Front Page. *SkLearn.* [En línia] [Data: 27 / 04 / 2022.] https://scikit-learn.org/stable/index.html.

27. Team, Indeed Editorial. What Is the Iterative Process? (Definition and Steps). *Indeed.* [En línia] 10 / 06 / 2021. [Data: 26 / 04 / 2022.] https://www.indeed.com/career-advice/career-development/iterative-process.

28. OpenCV. Interactive Foreground Extraction using GrabCut Algorithm. *OpenCV.* [En línia] [Data: 21 / 05 / 2022.] https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html.

29. Pierre. Bag-of-words model with SIFT descriptors. *Kaggle.* [En línia] 2017. [Data: 24 / 05 / 2022.] https://www.kaggle.com/code/pierre54/bag-of-words-model-with-sift-descriptors/data.

30. *Learning RGB-D Descriptors of Garment Parts.* Alenyà Ribas, Guillem, et al. s.l. : Engineering Applications of Artificial Intelligence,, 2014.

31. A Complete Guide to the Waterfall Project Method. *SmartSheet.* [En línia] [Data: 28 / 04 / 2022.] https://www.smartsheet.com/content-center/best-practices/project-management/project-management-guide/waterfall-methodology.

32. Lorica, Ben. One simple graphic: Researchers love PyTorch and TensorFlow. *Oreilly.* [En línia] 25 / 07 / 2019. [Data: 26 / 04 / 2022.] https://www.oreilly.com/content/one-simple-graphic-researchers-love-pytorch-and-tensorflow/.

33. Kliou Inside-Out Stacked Women Flare Jeans 2020 High Waist Slim Pockets Blue Demin Pants Autumn Streetwear Female Long Trousers. *DHGate.* [En línia] [Data: 5 / 05 / 2022.] https://www.dhgate.com/product/kliou-inside-out-stacked-women-flare-jeans/594662870.html.

34. PANTALÓN TEJANO CON KEVLAR GM-2700. *Motocat.* [En línia] [Data: 05 / 05 / 2022.] https://www.motocat.net/pantalones-moto-tejanos/878-pantalon-tejano-con-kevlar-gm-2700.html.

35. Subirats, Francesc. Los primeros pantalones vaqueros. *sobrecolores.blogspot.* [En línia] 27 / 12 / 2011. [Data: 05 / 05 / 2022.] https://sobrecolores.blogspot.com/2011/12/blue-jeans-los-pantalones-tejanos.html.

36. Population ages 65 and above (% of total population). *The World Bank.* [En línia] [Data: 05 / 05 / 2022.] https://data.worldbank.org/indicator/sp.pop.65up.to.zs.

37. Shash, Anuj. Through The Eyes of Gabor Filter. *Medium.* [En línia] 17 / 06 / 2018. [Data: 05 / 05 / 2022.] https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97.

38. Prasad, Meghshyam G. *Graph Cuts for Image Segmentation.* Mumbai : Department of Computer Science and Engineering.

39. alokesh985. Introduction to Support Vector Machines (SVM). *GeeksForGeeks.* [En línia] 16 / 06 / 2020. [Data: 05 / 05 / 2022.] https://www.geeksforgeeks.org/introduction-to-support-vector-machines-svm/.

40. Fast Point Feature Histograms (FPFH) descriptors. *Point Cloud Library.* [En línia] [Data: 05 / 05 / 2022.] https://pcl.readthedocs.io/projects/tutorials/en/latest/fpfh_estimation.html.

41. R Program to Find the Factorial of a Number. *Datamentor.* [En línia] [Data: 06 / 05 / 2022.] https://www.datamentor.io/r-programming/examples/factorial/.

42. C Program to Find Factorial of a Number. *Programiz.* [En línia] [Data: 06 / 05 / 2022.] https://www.programiz.com/c-programming/examples/factorial.

43. Factorial in Matlab. *Educba.* [En línia] [Data: 06 / 05 / 2022.] https://www.educba.com/factorial-in-matlab/.

44. Python Program to Find the Factorial of a Number. *Programiz.* [En línia] [Data: 06 / 05 / 2022.] https://www.programiz.com/python-programming/examples/factorial.

45. IBM Cloud Education. Convolutional Neural Networks. *IBM.* [En línia] 20 / 10 / 2020. [Data: 06 / 05 / 2022.] https://www.ibm.com/cloud/learn/convolutional-neural-networks.

46. Donges, Niklas. What Is Transfer Learning? Exploring the Popular Deep Learning Approach. *Builtin.* [En línia] 19 / 06 / 2019. [Data: 06 / 05 / 2022.] https://builtin.com/data-science/transfer-learning.

47. Kaushik, Aakash. Understanding ResNet50 architecture. *Opengenus.* [En línia] [Data: 06 / 05 / 2022.] https://iq.opengenus.org/resnet50-architecture/.

48. Gate, Research. *Research Gate.* [En línia] 2 / 2016. [Data: 21 / 05 / 2022.] https://www.researchgate.net/figure/Real-parts-of-the-Gabor-filter-bank-Generated-for-different-combinations-of-th-in_fig5_292671765.

49. *Distinctive Image Features from Scale-Invariant Keypoints.* Lowe, David G. Canada : University of British Columbia, 2004.

50. Todd, Nathaniel. Local Feature Matching using SIFT Descriptors. *Medium.* [En línia] 25 / 9 / 2019. [Data: 24 / 05 / 2022.] https://medium.com/build-more/local-feature-matching-using-sift-descriptors-23305f5b89e7.

Annex A: Objectius

A1. Objectiu Combinat

Libraries import

```
import os
import cv2
import csv
import time
import math
import imutils
import Augmentor
import numpy as np
from PIL import Image
from time import sleep
from tqdm.notebook import tqdm
from scipy import ndimage as nd
from scipy.ndimage import rotate
from matplotlib import pyplot as plt
```

```
%matplotlib inline
%run ./Image_Visualizer.ipynb #Delivers methods to handle images
%run ./List_Manager.ipynb #Delivers methods to control lists
```

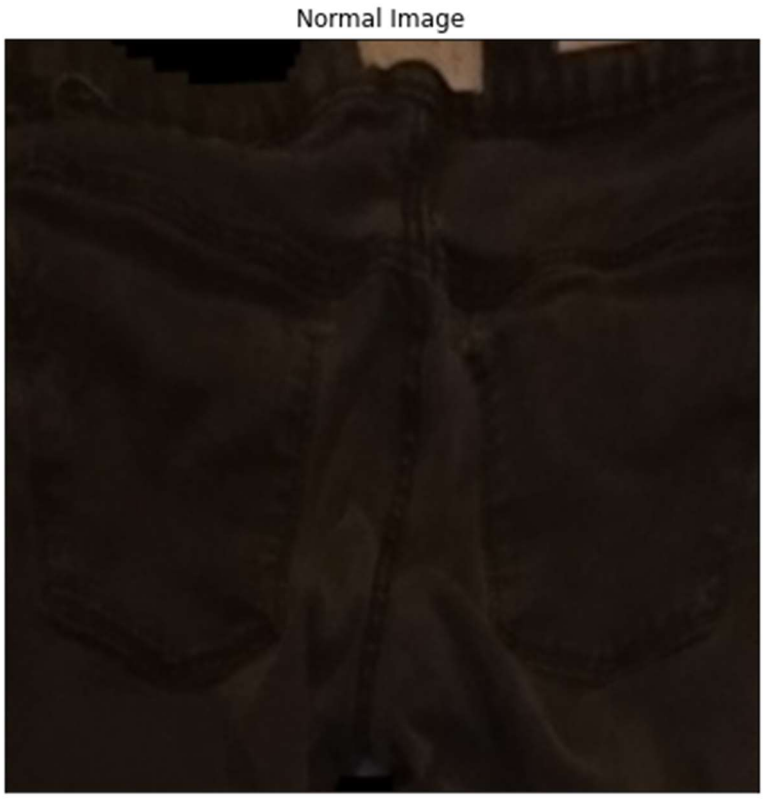
```
%run ./Objective_A.ipynb #Delivers methods to control lists
```

```
An exception has occurred, use %tb to see the full traceback.

SystemExit: Methods loaded


C:\Users\pauso\AppData\Roaming\Python\Python310\site-packages\IPython\co
re\interactiveshell.py:3369: UserWarning: To exit: use 'exit', 'quit', o
r Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)

An exception has occurred, use %tb to see the full traceback.

SystemExit: Methods loaded
```

```
%run ./Objective_B.ipynb #Delivers methods to control lists
```

```
Initialised with 94 image(s) found.
Output directory set to DataSets\ValidTrimmed\output.Operations: 5
      0: Flip (probability=0.5 top_bottom_left_right=LEFT_RIGHT )
      1: RotateRange (probability=0.5 max_left_rotation=-15 max_right_ro
tation=15 )
```

```
      2: Skew (probability=0.4 skew_type=RANDOM magnitude=0.5 )
      3: Zoom (probability=0.2 min_factor=1.1 max_factor=1.3 )
      4: Shear (probability=0.3 max_shear_left=2 max_shear_right=2 )
Images: 94
Classes: 4
      Class index: 0 Class label: Reverse_Down
      Class index: 1 Class label: Reverse_Up
      Class index: 2 Class label: Right_Down
      Class index: 3 Class label: Right_Up
Dimensions: 1
      Width: 255 Height: 255
Formats: 1
       JPEG

You can remove operations using the appropriate index and the remove_ope
ration(index) function.
```




```
An exception has occurred, use %tb to see the full traceback.

SystemExit: Methods loaded


An exception has occurred, use %tb to see the full traceback.

SystemExit: Methods loaded
```

Methods used

```
def PantsClassification(imageRoute, cutMatrix, var):
    image = cv2.imread(imageRoute)
    image = image[int(cutMatrix[2]):int(cutMatrix[3]), int(cutMatrix[0])
:int(cutMatrix[1])]
    imageMask = extractBKGGrab(image,var)
    corners, arePants = extractPantsOutline(image,imageMask, 4)
    if(arePants):
        pants = getTopPants(image, imageMask, corners)
        prediction = predictImage(pants, kmeans, mlp)
        return prediction
    else:
        return 4

"""
Function extracted from https://deeplizard.com/learn/video/km7pxKy4UHU t
```

```
o create a confusion matrix
"""
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    #plt.figure(figsize = [15,15])
    if normalize:
        cm = np.round((cm.astype('float') / cm.sum(axis=1)[:, np.newaxis
])*100, 1)
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    print(thresh)
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])
):
        text = cm[i, j]
        if normalize:
            text = str(cm[i, j])+"%"
        plt.text(j, i, text,
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    #plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Testing with a single image

```
imageRoute = r'DataSets\ValidDataset\Right_Down\1652615425192.jpg'

x1 = 700
x2 = 3250
y1 = 0
y2 = 50000
cutMatrix = [x1,x2,y1,y2]

PantsClassification(imageRoute, cutMatrix, 7.5)
```

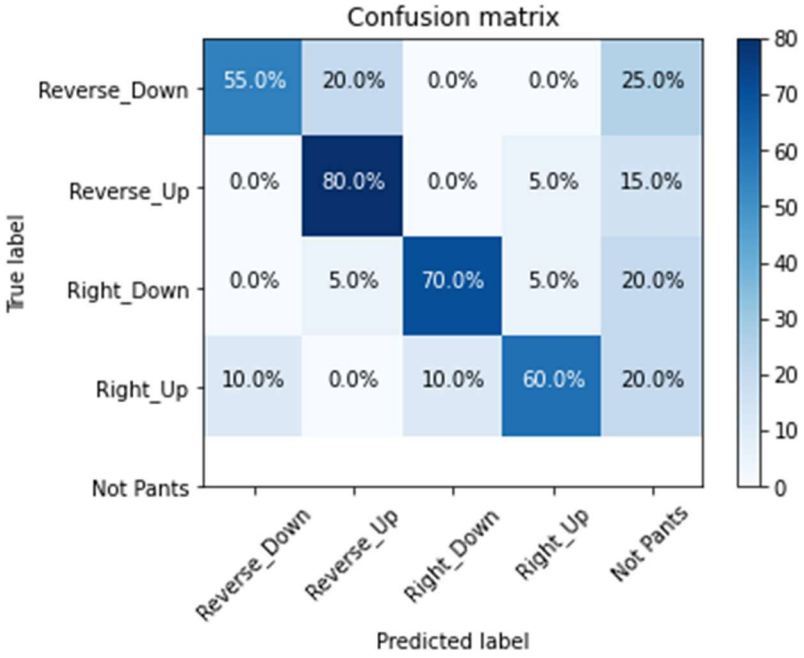
2

Testing with all images

We will execute the code with all the images, and see the results

```
ReverseDownPath = r'DataSets\ValidDataset\Reverse_Down'
ReverseUpPath = r'DataSets\ValidDataset\Reverse_Up'
RightDownPath = r'DataSets\ValidDataset\Right_Down'
RightUpPath = r'DataSets\ValidDataset\Right_Up'

AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]
labels = ["Reverse_Down","Reverse_Up","Right_Down", "Right_Up", "Not Pan
ts"]

def DetectPants(paths, cutMatrix):
    acom = 0
    maxTime = 0
    minTime = 10000
    cm = np.zeros((4,5))
    for iPath, path in enumerate(paths):
        if(iPath == 0):
            pathName = "Reverse_Down"
        elif(iPath == 1):
            pathName = "Reverse_Up"
        elif(iPath == 2):
            pathName = "Right_Down"
        else:
            pathName = "Right_Up"

        for files in os.listdir(path):
            iterStart = time.time()
            finalPath = os.path.join(path, files)
            result = PantsClassification(finalPath, cutMatrix, 7.5)
            cm[iPath][result] += 1
            # Calculate the diferent execution times
            iterEnd = time.time()
            duration = iterEnd-iterStart
            if(duration > maxTime):
                maxTime = duration
            if(duration < minTime):
                minTime = duration
            acom = acom +duration
    print("The average execution time was: "+ str(acom/80))
    print("The minimum execution time was: "+ str(minTime))
    print("The maximum execution time was: "+ str(maxTime))
    return cm

cm = DetectPants(AllPaths, cutMatrix)
```

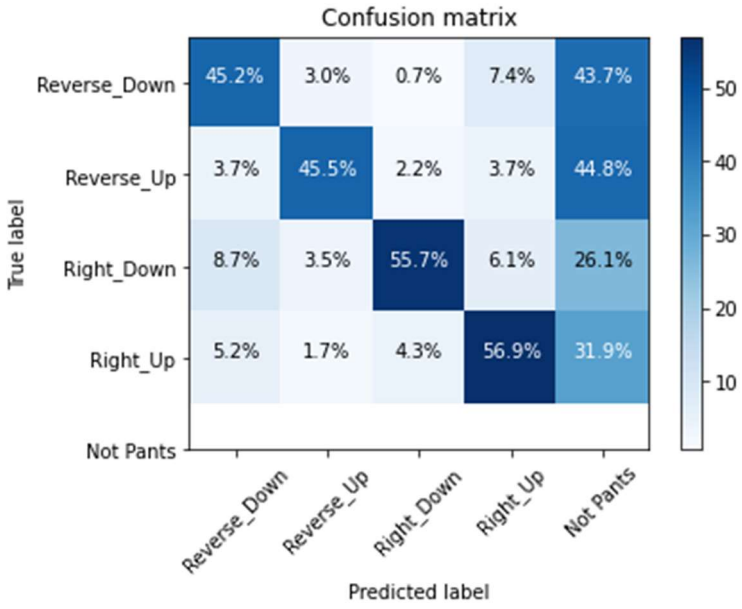
```
The average execution time was: 2.4519071131944656
The minimum execution time was: 1.088200330734253
The maximum execution time was: 5.305710792541504

plot_confusion_matrix(cm, labels, normalize=True)

Normalized confusion matrix
40.0
```



Testing with augmented images

```
ReverseDownPath = r'DataSets\ValidDataset\output\Reverse_Down'
ReverseUpPath = r'DataSets\ValidDataset\output\Reverse_Up'
RightDownPath = r'DataSets\ValidDataset\output\Right_Down'
RightUpPath = r'DataSets\ValidDataset\output\Right_Up'

AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]
labels = ["Reverse_Down","Reverse_Up","Right_Down", "Right_Up", "Not Pan
ts"]

cm = DetectPants(AllPaths, cutMatrix)

The average execution time was: 14.114530369639397
The minimum execution time was: 1.060016393661499
The maximum execution time was: 7.451405763626099

plot_confusion_matrix(cm, labels, normalize=True)
```

```
Normalized confusion matrix
28.45
```



A2. Objectiu A

Objective A: Detection and clasification of pants in a picture

This Jupyter notebook handles the completition of the first objective: Detection and clasification of pants in a picture for the TFG by Pau Solsona.

```
onlyMethods = False
```

Libraries and Initial Methods

First we will import all the required libraries that the notebook will use

```
import os
import cv2
import csv
import time
import math
import imutils
import Augmentor
import numpy as np
from PIL import Image
```

```
from time import sleep
from tqdm.notebook import tqdm
from scipy import ndimage as nd
from scipy.ndimage import rotate
from matplotlib import pyplot as plt
```

And execute some lines for a better code. This piece allows us to call different methods on other jupyter notebooks, allowing the user to easily handle images, and lists

```
%matplotlib inline
%run ./Image_Visualizer.ipynb #Delivers methods to handle images
%run ./List_Manager.ipynb #Delivers methods to control lists
```

Step 1: Background extraction

This first step will handle the extraction of the background in a picture, via the use of two diferent methods

```
def apply_filter(img, filters):
    #Normalize Image
    img = (img - img.mean()) / img.std()

    # This general function is designed to apply filters to our image
    # First create a numpy array the same size as our input image
    newimage = np.zeros_like(img)
    appliedFilters = []
    # Starting with a blank image, we loop through the images and apply
our Gabor Filter
    # On each iteration, we take the highest value (super impose), until
we have the max value across all filters
    # The final image is returned
    depth = -1 # remain depth same as original image

    for kern in filters:  # Loop through the kernels in our GaborFilter
        image_filter = np.sqrt(nd.convolve(img, np.real(kern), mode='wra
p')**2 +
                   nd.convolve(img, np.imag(kern), mode='wrap')**2)
        # Using Numpy.maximum to compare our filter and cumulative image
, taking the higher value (max)
        np.maximum(newimage, image_filter, newimage)
    #newimage = np.array(newimage, dtype='int')
    newimage = np.uint8(newimage)
    return newimage

def create_gaborfilter():
    #https://scikit-image.org/docs/0.11.x/auto_examples/plot_gabor.html
    # This function is designed to produce a set of GaborFilters
    # an even distribution of theta values equally distributed amongst p
i rad / 180 degree
```

```
    filters = []
    num_filters = 16
    ksize = 15  # The local area to evaluate
    sigma = 8.0 # Larger Values produce more edges
    lambd = 13.0 #distancia entre picos
    gamma = 0.5 #com d'eliptic
    psi = 0.5  # Offset value - lower generates cleaner results
    for theta in np.arange(0, np.pi, np.pi / num_filters):  # Theta is t
he orientation for edge detection
        kern = cv2.getGaborKernel((ksize, ksize), sigma, theta, lambd, g
amma, psi, ktype=cv2.CV_64F)
        kern /= 1.0 * kern.sum()  # Brightness normalization
        filters.append(kern)
    return filters

def getGaborImage(image_og, DEBUG = False):
    #Save shape
    processSize = [500,500]
    image_bg = cv2.cvtColor(image_og.copy(), cv2.COLOR_BGR2GRAY)

    #Resize to work with it
    image_bg_res = cv2.resize(image_bg.copy(), [350, 350], interpolation
= cv2.INTER_LINEAR)

    #Apply gaborFilter
    filters = create_gaborfilter()
    image_gb = apply_filter(image_bg_res, filters)
    if(DEBUG):
        showimage(image_gb, pltTitle = "1.1: Gabor Filtered")

    #Remove Corners
    image_gb = image_gb[int(5):int(image_gb.shape[1]-5), int(10):int(ima
ge_gb.shape[0]-5)]

    #Resize Both Pictures
    image_gb = cv2.resize(image_gb, processSize, interpolation= cv2.INTE
R_LINEAR)
    image_res = cv2.resize(image_og.copy(), processSize, interpolation=
cv2.INTER_LINEAR)
    if(DEBUG):
        showimage(image_gb, pltTitle= "1.2: Cut picture")
    return image_gb, image_res

def extractBKGGrab(image_og, var, DEBUG=False):
    originalSize = [image_og.shape[1], image_og.shape[0]]
    image_gb, image_res = getGaborImage(image_og, DEBUG)

    #Binarize Picture
    se=cv2.getStructuringElement(cv2.MORPH_RECT , (5,5))
    image_dil=cv2.morphologyEx(image_gb, cv2.MORPH_DILATE, se)
    (thresh, blackAndWhiteImage) = cv2.threshold(image_dil, var, 255, cv
```

```
2.THRESH_BINARY)
    if(DEBUG):
        showimage(blackAndWhiteImage, pltTitle = "1.3: Binarized Picture
")

    #Single object on screen
    imageMask = blackAndWhiteImage.copy()
    contours, hierarchy = cv2.findContours(imageMask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
    if(len(contours) <= 0):
        return imageMask
    max_contour = max(contours, key=len)

    imageMask = np.zeros_like(imageMask, dtype='uint8')
    imageMask = cv2.drawContours(imageMask, [max_contour], -1, (255), th
ickness=-1)
    se=cv2.getStructuringElement(cv2.MORPH_RECT , (10,10))
    imageMask = cv2.morphologyEx(imageMask, cv2.MORPH_ERODE, se)
    if(DEBUG):
        showimage(imageMask, pltTitle = "1.4.1: Single Pants Mask")
        showimage(cv2.bitwise_and(image_res, image_res, mask = imageMask
), pltTitle = "1.4.2: Applied Mask")

    #Transform Mask for Grab Cut
    mask = np.where((imageMask==255),3,0).astype('uint8')
    bgdModel = np.zeros((1,65),np.float64)
    fgdModel = np.zeros((1,65),np.float64)
    rect = [0,0,0,0]
    new_mask, fg, bg = cv2.grabCut(image_res,mask,rect,bgdModel,fgdModel
,5,cv2.GC_INIT_WITH_MASK)
    if(DEBUG):
        showimage(new_mask, pltTitle="1.5.1: Extracted Mask with Grab Cu
t")

    mask2 = np.where((new_mask==2)|(new_mask==0),0,1).astype('uint8')
    if(DEBUG):
        showimage(cv2.bitwise_and(image_res, image_res, mask = mask2), p
ltTitle = "1.5.2: Applied Grab Cut Mask")

    #Cleaning Mask
    se=cv2.getStructuringElement(cv2.MORPH_RECT , (5,5))
    mask2 = cv2.morphologyEx(mask2, cv2.MORPH_ERODE, se)
    contours, hierarchy = cv2.findContours(mask2, cv2.RETR_EXTERNAL, cv2
.CHAIN_APPROX_NONE)
    if(len(contours) <= 0):
        imageMask = cv2.resize(imageMask, originalSize, interpolation= c
v2.INTER_LINEAR)
        return imageMask
    max_contour = max(contours, key=len)

    #Applying mask
```

```
    imageMask = np.zeros_like(imageMask, dtype='uint8')
    imageMask = cv2.drawContours(imageMask, [max_contour], -1, (255), th
ickness=-1)
    imageMask = cv2.resize(imageMask, originalSize, interpolation= cv2.I
NTER_LINEAR)
    onlyPants = cv2.bitwise_and(image_og, image_og, mask = imageMask)
    if(DEBUG):
        showimage(imageMask, pltTitle = "1.6.1: Cleaned Mask")
        showimage(onlyPants, pltTitle = "1.6.2: Applied Cleaned Mask")
    return imageMask
```

Step 2: Pant detection

Once we know that the pants have been in theory extracted from the background, we will proceed to parametrize the object to find out if there is indeed pants on it

```
def checkAngle(p1,p2,p3, threshold, drawQuiver = False):
    p1 = p1.astype(float)
    p2 = p2.astype(float)
    p3 = p3.astype(float)

    line1 = p2-p1
    line2 = p3-p2
    line1mag = np.linalg.norm(line1)
    line2mag = np.linalg.norm(line2)
    if(line1mag <= 0.1 or line2mag <= 0.1):
        return False

    line1 = line1/line1mag
    line2 = line2/line2mag

    dot = np.dot(line1, line2)

    if(drawQuiver):
        print(dot)
        fig, ax = plt.subplots(figsize=[7,7])
        plt.quiver([0, 0], [0, 0], [line1[0], line2[0]], [line1[1], line
2[1]], angles='xy', scale_units='xy', scale=1)
        plt.xlim(-2, 2)
        plt.ylim(-2, 2)
        plt.show()
    if(dot <= threshold):
        return True
    else:
        return False

def extractPantsOutline(image_og, imageMask, minimumDistance, blockSize
= 23, ksize = 9, k = 0.1, DEBUG = False, quiv = False):
    targetSize = [500, 500]
    originalSize = [image_og.shape[1], image_og.shape[0]]
```

```
    imageMask = cv2.resize(imageMask, targetSize, interpolation= cv2.INT
ER_LINEAR)
    image_res = cv2.resize(image_og, targetSize, interpolation= cv2.INTE
R_LINEAR)

    #Preprocessing
    kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
    sharpened = cv2.filter2D(imageMask, -1, kernel) #Sharpen image
    kernelClose = np.ones((7,7),np.uint8)
    sharpened = cv2.morphologyEx(sharpened, cv2.MORPH_CLOSE, kernelClose
)
    if(DEBUG):
        showimage(sharpened, pltTitle = "2.1: Sharpen the image")

    #Get corners through Harris
    operatedImage = np.uint8(sharpened)
    dest = cv2.cornerHarris(operatedImage, blockSize, ksize, k) #Input p
arameters
    dest = cv2.dilate(dest, None) #Get corners
    if(DEBUG):
        showimage(dest, pltTitle = "2.2: Find corners")

    #Making Blobs out of those corners
    corners = np.zeros_like(dest)
    corners[dest>0.001*dest.max()] = [1.]
    kernelClose = np.ones((3,3),np.uint8)
    corners = cv2.morphologyEx(corners, cv2.MORPH_CLOSE, kernelClose)
    if(DEBUG):
        showimage(corners, pltTitle = "2.3: Get blobs of corners")

    #Transform corner type
    corners = corners.astype(np.uint8)

    #Getting centers of corners
    cornerContours, hierarchy = cv2.findContours(corners, cv2.RETR_EXTER
NAL, cv2.CHAIN_APPROX_NONE)
    centers = np.zeros_like(corners)
    cornerPoints = []
    for i in cornerContours:
        M = cv2.moments(i)
        if M['m00'] != 0:
            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])
            cv2.circle(centers, (cx, cy), 4, (1), -1)
            cornerPoints.append([cx,cy])
    if(DEBUG):
        imageWithCenters = image_res.copy()
        cornerCopy = np.array(centers.copy())
        imageWithCenters[centers>0.01*centers.max()] = [0,0,255]
        showimage(imageWithCenters, pltTitle = "2.4: Visualized Blobs")
```

```
    #Get Contours of image
    contours, hierarchy = cv2.findContours(imageMask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)
    if(len(contours) <= 0):
        return [], False
    pantsContour = max(contours, key=len)
    if(DEBUG):
        contourImage = cv2.drawContours(image_res.copy(), pantsContour,
-1, (0, 255, 0), 5)
        showimage(contourImage, pltTitle = "2.5: Contours")

    ##Possible millora aplicant minAreaRect

    #Sort Corners
    sortedCorners = []  #Non sorted
    for point in pantsContour:
        point = point[0]
        for corner in cornerPoints:
            distance = np.linalg.norm(corner-point) #gets length
            if distance < minimumDistance:
                sortedCorners.append(corner)
                cornerPoints.remove(corner)
                break
    if(DEBUG):
        drawLines(image_res, sortedCorners, pltTitle = "2.6: Initial Cor
ners")

    #Remove unnecessary corners
    for iteration in range(0, 3):
        finalCorners = []
        for index in range(0, len(sortedCorners)):
            p1Ind = index-2
            if(p1Ind < 0):
                p1Ind = len(sortedCorners)-2
            p2Ind = index-1
            if(p2Ind < 0):
                p2Ind = len(sortedCorners)-1

            p1 = np.array(sortedCorners[p1Ind])
            p2 = np.array(sortedCorners[p2Ind])
            p3 = np.array(sortedCorners[index])
            if(checkAngle(p1, p2, p3, 0.7, drawQuiver = quiv)):
                finalCorners.append(sortedCorners[p2Ind])
        sortedCorners = finalCorners
        if(len(sortedCorners) <= 7):
            break

    #Final result
    xDif = originalSize[0]/targetSize[0]
    yDif = originalSize[1]/targetSize[1]
    scaledCorners = []
```

```
    for index in range(0,len(finalCorners)):
        scaledCorners.append([int(finalCorners[index][0]*xDif),int(final
Corners[index][1]*yDif)])
    scaledCorners = np.array(scaledCorners)

    if(DEBUG):
        drawLines(image_og, scaledCorners, pltTitle = "2.7: Simplified C
ontours")

    corners = len(scaledCorners)
    if(corners == 7):
         return scaledCorners, True
    else:
         return scaledCorners, False
```

Step 3: Feature Extraction

Once pants have been found, we can proceed to generate a simplified version of these pants, containing the most important features inside of it

```
def findSharingPoint(Line1, Line2):
    if(np.array_equal(Line1[0],Line2[0])):
        return Line1[0]
    else:
        if(np.array_equal(Line1[1],Line2[0])):
            return Line1[1]
        else:
            return Line2[1]

def findIntersection(Line1, Line2, reesizeFactor = 0.2):
        A = Line1[0]*reesizeFactor
        B = Line1[1]*reesizeFactor

        C = Line2[0]*reesizeFactor
        D = Line2[1]*reesizeFactor
        a1 = B[1] - A[1];
        b1 = A[0] - B[0];
        c1 = a1 * (A[0]) + b1 * (A[1]);

        a2 = D[1] - C[1];
        b2 = C[0] - D[0];
        c2 = a2 * (C[0]) + b2 * (C[1]);

        determinant = a1 * b2 - a2 * b1;

        if determinant == 0:
            print("Parallel")
        else:
            x = int((b2 * c1 - b1 * c2) / determinant);
```

```
            y = int((a1 * c2 - a2 * c1) / determinant);
            return [x/reesizeFactor,y/(reesizeFactor)];

def getTopPants(image_og, imageMask, corners, DEBUG = False):
    pants = cv2.bitwise_and(image_og, image_og, mask = imageMask)
    #We get distances between all consecutive points
    distances = []
    for index1 in range(0, len(corners)):
        index0 = index1-1
        if(index0 < 0):
            index0 = len(corners)-1

        p1 = np.array(corners[index1])
        p0 = np.array(corners[index0])

        distance = np.linalg.norm(p0-p1)
        distances.append([distance, [corners[index0],corners[index1]]])

    #The list gets sorted
    quickSort(distances,0, len(distances)-1)

    #Get the two longest lines
    Line1 = distances[0][1]
    Line2 = distances[1][1]
    if(DEBUG):
        Lines = [Line1, Line2]
        drawSingleLines(pants, Lines, pltTitle = "3.1: Get longest lines
")

    #Get the waist line
    WaistPoints = distances[4][1]

    #We find the crotch point
    crotchPoint = findSharingPoint(distances[2][1], distances[3][1])
    WaistLine = np.array(WaistPoints[1])-np.array(WaistPoints[0])
    if(DEBUG):
        Lines = [Line1, Line2, WaistPoints]
        drawSingleLines(pants, Lines, pltTitle = "3.2: Get waist line")

    #Create crotch line in parallel to waistLine
    CrotchLine = [crotchPoint, crotchPoint + WaistLine]

    #Find intersection point between crotch line and longest one
    intersection1 = findIntersection(Line1, CrotchLine)
    intersection2 = findIntersection(Line2, CrotchLine)
    if(DEBUG):
        Lines = [[intersection1, crotchPoint], [intersection2, crotchPoi
nt]]
        drawSingleLines(pants, Lines, pltTitle = "3.3: Get intersection
lines with crotch")
```

```
    #New points with crotch and longest lines
    corner1 = findIntersection(Line1, WaistPoints)
    corner2 = findIntersection(Line2, WaistPoints)

    #Order all the points
    FinalPoints = np.float32([corner1, intersection1, intersection2, cor
ner2])
    if(DEBUG):
        drawLines(pants, FinalPoints, pltTitle = "3.4: Final set of poin
ts on original Image")

    #New size
    targetCoordinates = np.float32([[0,0], [0,255], [255,255], [255,0]])

    #Wrap image to fit
    matrix = cv2.getPerspectiveTransform(FinalPoints, targetCoordinates)
    result = cv2.warpPerspective(pants, matrix, (255, 255))
    if(DEBUG):
        showimage(result, pltTitle = "3.5: Warped image")
    return result

if(onlyMethods):
    raise SystemExit("Methods loaded")
```

Visualizing the process

Loading the image

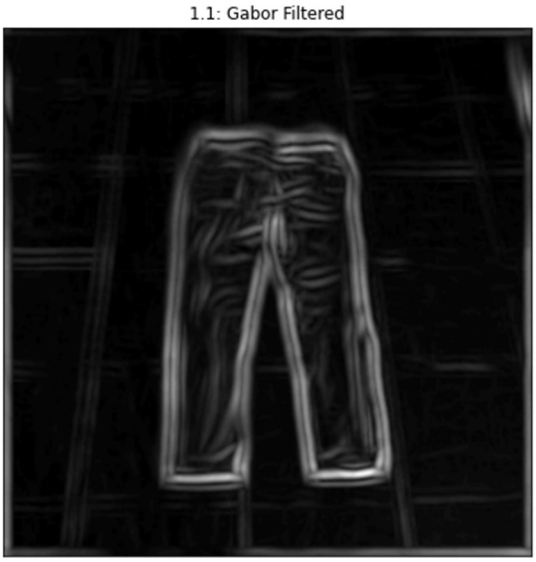
To visualize these pictures, we will execute in order the subsequent methods. First we load a picture and crop it so that the most important part of it are pants

```
PictureRoute = r'DataSets\ValidDataset\Reverse_Down\1652615425022.jpg'
x1 = 700
x2 = 3250
y1 = 0
y2 = 50000
cutMatrix = [x1,x2,y1,y2]

image = cv2.imread(PictureRoute)
image = image[int(cutMatrix[2]):int(cutMatrix[3]), int(cutMatrix[0]):int
(cutMatrix[1])]
showimage(image, pltTitle='Original')
```

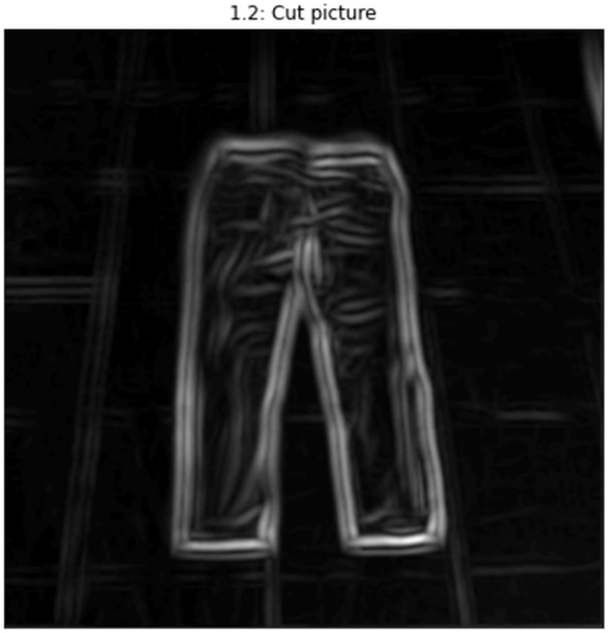
Extracting the Background

Once we have loaded the image, we can proceed to execute the different methods to extract the background from it

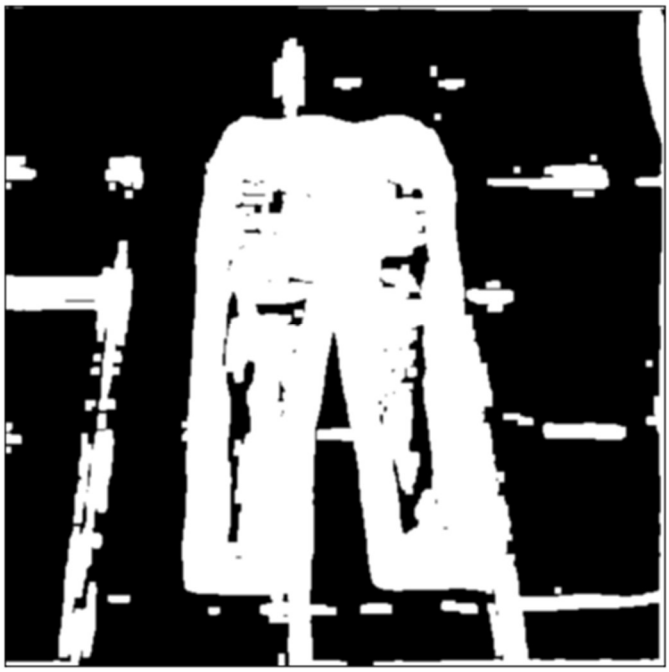
```
imageMask = extractBKGGrab(image,3,DEBUG = True)
```



1.2: Cut picture



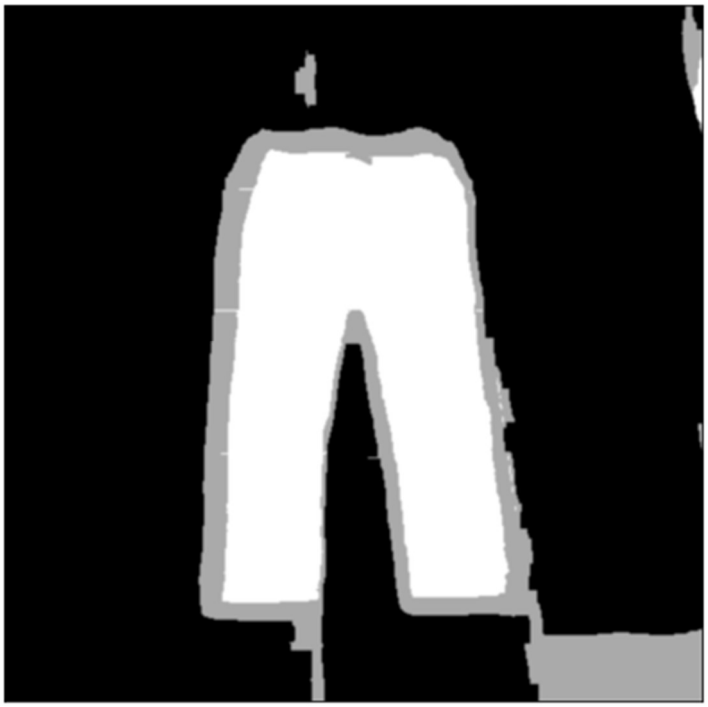
1.3: Binarized Picture



1.4.1: Single Pants Mask

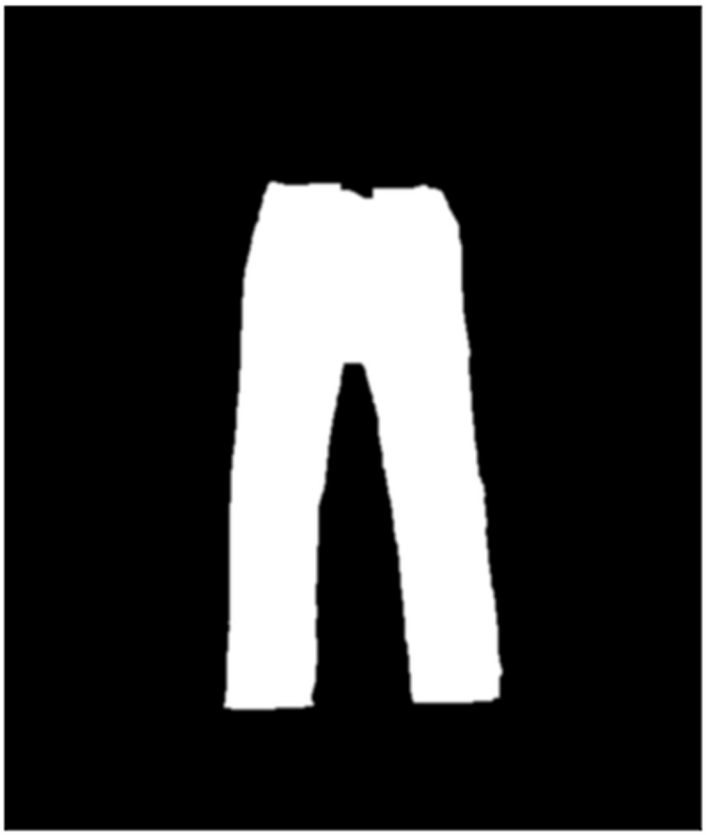
1.4.2: Applied Mask

1.5.1: Extracted Mask with Grab Cut



1.5.2: Applied Grab Cut Mask

1.6.1: Cleaned Mask

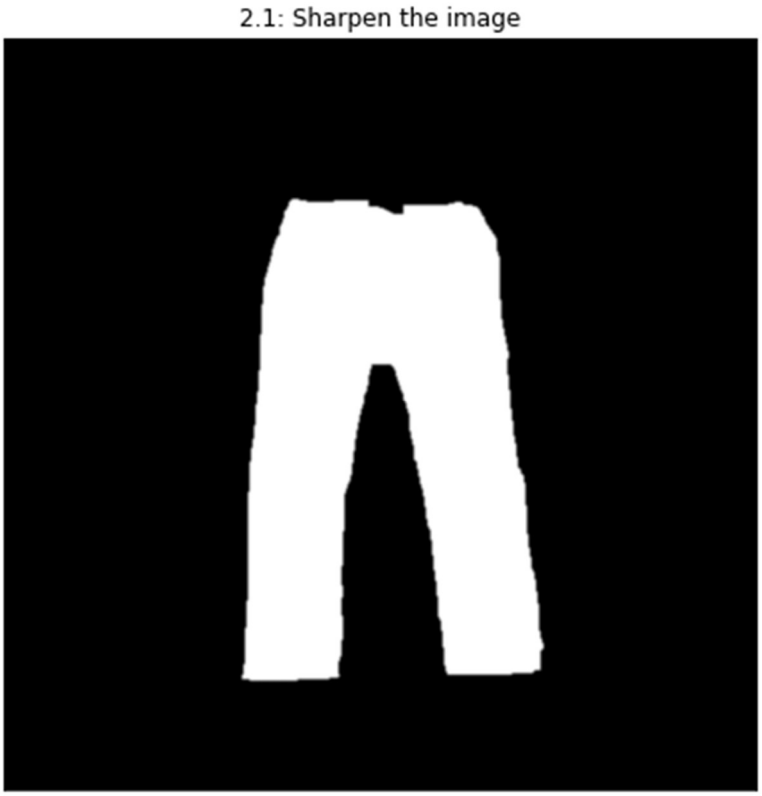


1.6.2: Applied Cleaned Mask

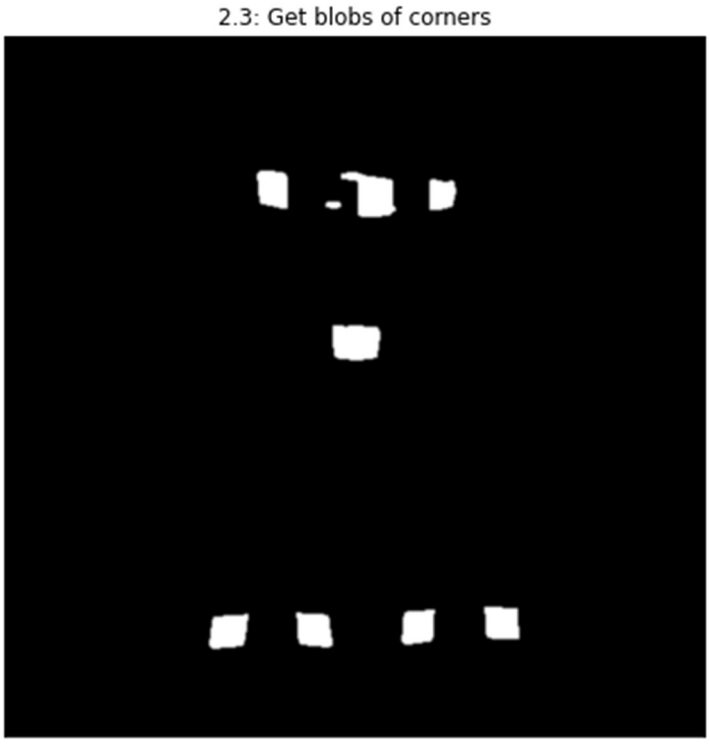
Extracting the pants

Without the background, and with the mask of the image containing only the pants, we can extract the diferent points from the parametrization

```
corners, arePants = extractPantsOutline(image,imageMask, 4, DEBUG = True
)
```




2.2: Find corners


2.3: Get blobs of corners

2.4: Visualized Blobs

2.5: Contours

2.6: Initial Corners

2.7: Simplified Contours

Getting the cropped version

Finally we can find the image with all the important features if they are pants

```
if(arePants):
    result = getTopPants(image, imageMask, corners, DEBUG = True)
```

3.2: Get waist line

3.3: Get intersection lines with crotch

3.4: Final set of points on original Image

3.5: Warped image

Testing the method

Once we verified that the methods work accordingly, we can proceed to test the method on multiple images, observing the different results on each category, and saving those images that have been generated on a specific folder

```
def pantsCheckViaGrab(image, var): #The first pipeline, checking while u
sing the grabcut method
    imageMask = extractBKGGrab(image, var)
    corners, arePants = extractPantsOutline(image,imageMask, 4)
    if(arePants):
        result = getTopPants(image, imageMask, corners)
        return result, True
    return image, False
```

Saving images

To save this pictures, it is needed to define the different routes

```
ReverseDownPath = r'DataSets\ValidDataset\Reverse_Down'
ReverseUpPath = r'DataSets\ValidDataset\Reverse_Up'
RightDownPath = r'DataSets\ValidDataset\Right_Down'
RightUpPath = r'DataSets\ValidDataset\Right_Up'

labels = ["Reverse_Down","Reverse_Up","Right_Down", "Right_Up"]
RowHeader = ['file','label']

Labels = ['file','label']
AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]

finalRoute = r'DataSets\ValidTrimmed'
finalCSVName = r'DataSets\ValidTrimmed\pants_dataset.csv'
filesCSV = []
```

And then we can execute the diferent pipelines and save the images, furthermore, we can see those cases that it breaks

```
def generateMultipleImages(paths, var, cutMatrix):
    acom = 0
    maxTime = 0
    minTime = 10000
    for iPath, path in enumerate(paths):
        if(iPath == 0):
            pathName = "Reverse_Down"
        elif(iPath == 1):
            pathName = "Reverse_Up"
        elif(iPath == 2):
            pathName = "Right_Down"
        else:
```

```
            pathName = "Right_Up"

        for files in os.listdir(path):
            iterStart = time.time()
            finalPath = os.path.join(path, files)
            image = cv2.imread(finalPath)
            image = image[int(cutMatrix[2]):int(cutMatrix[3]), int(cutMa
trix[0]):int(cutMatrix[1])]
            pantsPic, arePants = pantsCheckViaGrab(image, var)

            # Calculate the diferent execution times
            iterEnd = time.time()
            duration = iterEnd-iterStart
            if(duration > maxTime):
                maxTime = duration
            if(duration < minTime):
                minTime = duration
            acom = acom +duration

            #if(arePants):
                #print("Extracted via method of GrabCut")

            if(arePants != False):
                route = saveImage(pantsPic,finalRoute, pathName, iPath,
files)
                filesCSV.append([route,pathName])
            else:
                print("Issue with " + files)
    print("The average execution time was: "+ str(acom/80))
    print("The minimum execution time was: "+ str(minTime))
    print("The maximum execution time was: "+ str(maxTime))

x1 = 700
x2 = 3250
y1 = 0
y2 = 50000
cutMatrix = [x1,x2,y1,y2]
generateMultipleImages(AllPaths, 7.5, cutMatrix)
```

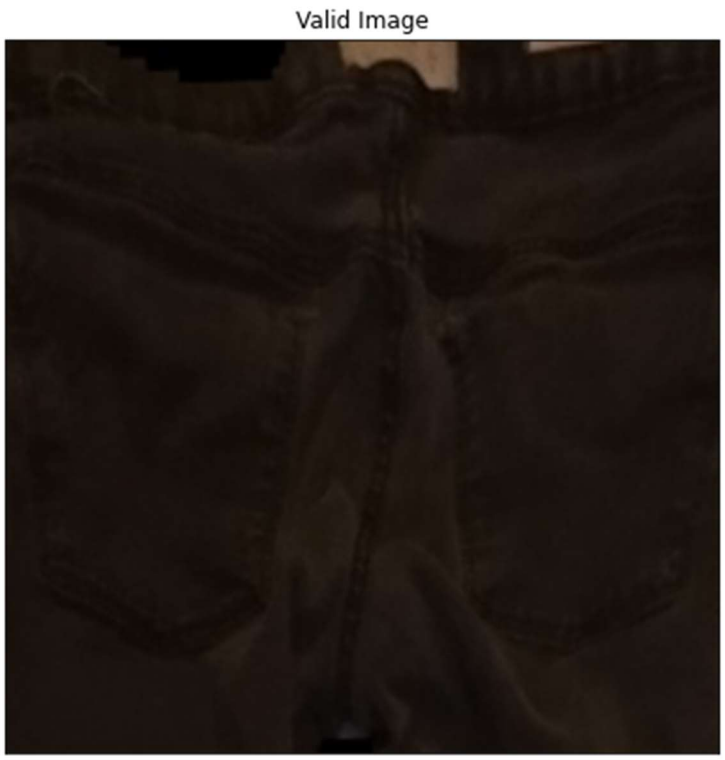
```
Issue with 1652555728229.jpg
Issue with 1652615425053.jpg
Issue with 1652615425080.jpg
Issue with 1652615425089.jpg
Issue with 1652555127365.jpg
Issue with 1652555127378.jpg
Issue with 1652615425158.jpg
Issue with 1652555127532.jpg
Issue with 1652555127568.jpg
Issue with 1652555127583.jpg
Issue with 1652555127590.jpg
Issue with 1652555127594.jpg
```

```
Issue with 1652615425323.jpg
Issue with 1652615425332.jpg
The average execution time was: 1.1474388062953949
The minimum execution time was: 1.0743358135223389
The maximum execution time was: 1.2122502326965332

generateCSV(finalRoute,finalCSVName)
```

Good and bad Case

We can visualize a good case, and a bad case. The bad ones need to be manually removed

```
ValidImageR = r'DataSets\ValidTrimmed\Reverse_Down\P_0_1652555127475.jpg
'
#InValidImageR = r'DataSets\ValidTrimmed\Reverse_Down\P_0_1652615425089.
jpg'
ValidImage = cv2.imread(ValidImageR)
#InValidImage = cv2.imread(InValidImageR)

showimage(ValidImage, pltTitle= "Valid Image")
#showimage(InValidImage, pltTitle= "Invalid Image")
```



Background extraction

We will extract only the background to see the results of accuracy

```
ReverseDownPath = r'DataSets\ValidDataset\Reverse_Down'
ReverseUpPath = r'DataSets\ValidDataset\Reverse_Up'
RightDownPath = r'DataSets\ValidDataset\Right_Down'
RightUpPath = r'DataSets\ValidDataset\Right_Up'

labels = ["Reverse_Down","Reverse_Up","Right_Down", "Right_Up"]
RowHeader = ['file','label']

Labels = ['file','label']
AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]

finalRoute = r'DataSets\ValidMask'
finalCSVName = r'DataSets\ValidMask\pants_dataset.csv'
filesCSV = []

def extractMask(paths, var, cutMatrix):
    acom = 0
    maxTime = 0
    minTime = 10000
    for iPath, path in enumerate(paths):
        if(iPath == 0):
            pathName = "Reverse_Down"
        elif(iPath == 1):
            pathName = "Reverse_Up"
        elif(iPath == 2):
            pathName = "Right_Down"
        else:
            pathName = "Right_Up"

        for files in os.listdir(path):
            iterStart = time.time()
            finalPath = os.path.join(path, files)
            image = cv2.imread(finalPath)
            image = image[int(cutMatrix[2]):int(cutMatrix[3]), int(cutMa
trix[0]):int(cutMatrix[1])]
            imageMask = extractBKGGrab(image, var)
            masked = cv2.bitwise_and(image, image, mask = imageMask)
            # Calculate the diferent execution times
            iterEnd = time.time()
            duration = iterEnd-iterStart
            if(duration > maxTime):
                maxTime = duration
            if(duration < minTime):
                minTime = duration
            acom = acom +duration
```

```
            route = saveImage(masked,finalRoute, pathName, iPath, files)
    print("The average execution time was: "+ str(acom/80))
    print("The minimum execution time was: "+ str(minTime))
    print("The maximum execution time was: "+ str(maxTime))

x1 = 700
x2 = 3250
y1 = 0
y2 = 50000
cutMatrix = [x1,x2,y1,y2]
extractMask(AllPaths, 7.5, cutMatrix)

The average execution time was: 1.1037381380796432
The minimum execution time was: 1.0753047466278076
The maximum execution time was: 1.1670854091644287

generateCSV(finalRoute,finalCSVName)
```

Unused Dataset

We can use the unused dataset to get more pictures for later use during the augmentation and training phase

```
ReverseDownPath = r'DataSets\UnusedDataset\Reverse_Down'
ReverseUpPath = r'DataSets\UnusedDataset\Reverse_Up'
RightDownPath = r'DataSets\UnusedDataset\Right_Down'
RightUpPath = r'DataSets\UnusedDataset\Right_Up'
Labels = ['file','label']
AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]

finalRoute = r'DataSets\ValidTrimmed'
filesCSV = []

x1 = 700
x2 = 3250
y1 = 0
y2 = 50000
cutMatrix = [x1,x2,y1,y2]
generateMultipleImages(AllPaths, 3.5, cutMatrix)

Issue with IMG_20220329_190239.jpg
Issue with IMG_20220329_190243.jpg
Issue with IMG_20220329_190245.jpg
Issue with IMG_20220329_190254.jpg
Issue with IMG_20220329_190256.jpg
Issue with IMG_20220329_190549.jpg
Issue with IMG_20220329_190552.jpg
Issue with IMG_20220329_190557.jpg
Issue with IMG_20220329_190600.jpg
Issue with IMG_20220329_190602.jpg
Issue with IMG_20220329_190605.jpg
```

```
Issue with IMG_20220329_190608.jpg
Issue with IMG_20220329_190610.jpg
Issue with IMG_20220329_190630.jpg
Issue with IMG_20220329_190148.jpg
Issue with IMG_20220329_190153.jpg
Issue with IMG_20220329_190202.jpg
Issue with IMG_20220329_190204.jpg
Issue with IMG_20220329_190207.jpg
Issue with IMG_20220329_190452.jpg
Issue with IMG_20220329_190456.jpg
Issue with IMG_20220329_190458.jpg
Issue with IMG_20220329_190504.jpg
Issue with IMG_20220329_190506.jpg
Issue with IMG_20220329_190510.jpg
Issue with IMG_20220329_190514.jpg
Issue with IMG_20220329_190516.jpg
Issue with IMG_20220329_190533.jpg
Issue with IMG_20220329_190045.jpg
Issue with IMG_20220329_190048.jpg
Issue with IMG_20220329_190102.jpg
Issue with IMG_20220329_190106.jpg
Issue with IMG_20220329_190108.jpg
Issue with IMG_20220329_190111.jpg
Issue with IMG_20220329_190115.jpg
Issue with IMG_20220329_190344.jpg
Issue with IMG_20220329_190346.jpg
Issue with IMG_20220329_190348.jpg
Issue with IMG_20220329_190353.jpg
Issue with IMG_20220329_190355.jpg
Issue with IMG_20220329_190358.jpg
Issue with IMG_20220329_190400.jpg
Issue with IMG_20220329_190402.jpg
Issue with IMG_20220329_190404.jpg
Issue with IMG_20220329_190406.jpg
Issue with IMG_20220329_185600.jpg
Issue with IMG_20220329_185604.jpg
Issue with IMG_20220329_185609.jpg
Issue with IMG_20220329_185614.jpg
Issue with IMG_20220329_185624.jpg
Issue with IMG_20220329_185651.jpg
Issue with IMG_20220329_185657.jpg
Issue with IMG_20220329_185802.jpg
Issue with IMG_20220329_185814.jpg
Issue with IMG_20220329_185849.jpg
Issue with IMG_20220329_185852.jpg
Issue with IMG_20220329_185856.jpg
Issue with IMG_20220329_185859.jpg
Issue with IMG_20220329_185902.jpg
The average execution time was: 1.343504649400711
The minimum execution time was: 1.1749951839447021
The maximum execution time was: 1.9383432865142822
```

```
generateCSV(finalRoute,finalCSVName)
```

Background extraction from unused dataset

```
ReverseDownPath = r'DataSets\UnusedDataset\Reverse_Down'
ReverseUpPath = r'DataSets\UnusedDataset\Reverse_Up'
RightDownPath = r'DataSets\UnusedDataset\Right_Down'
RightUpPath = r'DataSets\UnusedDataset\Right_Up'
Labels = ['file','label']
AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]

finalRoute = r'DataSets\UnusedMask'
filesCSV = []

x1 = 700
x2 = 3250
y1 = 0
y2 = 50000
cutMatrix = [x1,x2,y1,y2]
extractMask(AllPaths, 3.5, cutMatrix)
```

```
The average execution time was: 1.26330027282238
The minimum execution time was: 1.1561472415924072
The maximum execution time was: 1.7498860359191895
```

Extracting from generated images

We can test the methods on augmented data, to see how it performs

Generating images

```
baseTestPath = r'DataSets\ValidDataset'
baseTestCSV = r'DataSets\ValidDataset\pants_test.csv'

labels = ["Reverse_Down","Reverse_Up","Right_Down", "Right_Up"]
RowHeader = ['file','label']

generateCSV(baseTestPath,baseTestCSV)

#Farem serviro Augmentor per a generar imatges
p = Augmentor.Pipeline(baseTestPath)
p.flip_left_right(0.5)
p.rotate(probability=0.5, max_left_rotation=10, max_right_rotation=10)
p.skew(0.4, 0.2)
p.zoom(probability = 0.2, min_factor = 1.1, max_factor = 1.3)
p.shear(0.3, .5,.5)
p.status()
```

```
Initialised with 80 image(s) found.
Output directory set to DataSets\ValidDataset\output.Operations: 5
      0: Flip (probability=0.5 top_bottom_left_right=LEFT_RIGHT )
      1: RotateRange (probability=0.5 max_left_rotation=-10 max_right_ro
tation=10 )
      2: Skew (probability=0.4 skew_type=RANDOM magnitude=0.2 )
      3: Zoom (probability=0.2 min_factor=1.1 max_factor=1.3 )
      4: Shear (probability=0.3 max_shear_left=0.5 max_shear_right=0.5 )
Images: 80
Classes: 4
      Class index: 0 Class label: Reverse_Down
      Class index: 1 Class label: Reverse_Up
      Class index: 2 Class label: Right_Down
      Class index: 3 Class label: Right_Up
Dimensions: 1
      Width: 4032 Height: 3024
Formats: 1
       JPEG

You can remove operations using the appropriate index and the remove_ope
ration(index) function.
```

```
#p.sample(500)
```

Testing on generated

We can visualize the results on the folder, AugmentedTrimmed

```
ReverseDownPath = r'DataSets\ValidDataset\output\Reverse_Down'
ReverseUpPath = r'DataSets\ValidDataset\output\Reverse_Up'
RightDownPath = r'DataSets\ValidDataset\output\Right_Down'
RightUpPath = r'DataSets\ValidDataset\output\Right_Up'
Labels = ['file','label']
AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]

finalRoute = r'DataSets\AugmentedTrimmed'
filesCSV = []

#generateMultipleImages(AllPaths, 7.5, cutMatrix)
```

Background extraction from generation

We will extract the background from the generated images

```
ReverseDownPath = r'DataSets\ValidDataset\output\Reverse_Down'
ReverseUpPath = r'DataSets\ValidDataset\output\Reverse_Up'
RightDownPath = r'DataSets\ValidDataset\output\Right_Down'
RightUpPath = r'DataSets\ValidDataset\output\Right_Up'
Labels = ['file','label']
AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]
```

```
finalRoute = r'DataSets\AugmentedMask'
filesCSV = []

x1 = 700
x2 = 3250
y1 = 0
y2 = 50000
cutMatrix = [x1,x2,y1,y2]
extractMask(AllPaths, 7.5, cutMatrix)

The average execution time was: 0.0
The minimum execution time was: 10000
The maximum execution time was: 0
```

Pants extraction from the background ones

```
ReverseDownPath = r'DataSets\AugmentedMask\Reverse_Down'
ReverseUpPath = r'DataSets\AugmentedMask\Reverse_Up'
RightDownPath = r'DataSets\AugmentedMask\Right_Down'
RightUpPath = r'DataSets\AugmentedMask\Right_Up'
Labels = ['file','label']
AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]

finalRoute = r'DataSets\AugmentedMaskTrimmed'
filesCSV = []

def getPants(paths, cutMatrix):
    acom = 0
    maxTime = 0
    minTime = 10000
    for iPath, path in enumerate(paths):
        if(iPath == 0):
            pathName = "Reverse_Down"
        elif(iPath == 1):
            pathName = "Reverse_Up"
        elif(iPath == 2):
            pathName = "Right_Down"
        else:
            pathName = "Right_Up"

        for files in os.listdir(path):
            iterStart = time.time()
            finalPath = os.path.join(path, files)
            image = cv2.imread(finalPath)
            image = image[int(cutMatrix[2]):int(cutMatrix[3]), int(cutMa
trix[0]):int(cutMatrix[1])]
            imageMask = cv2.cvtColor(image.copy(), cv2.COLOR_BGR2GRAY)
            (thresh, imageMask) = cv2.threshold(imageMask, 0, 255, cv2.T
HRESH_BINARY)
            corners, arePants = extractPantsOutline(image,imageMask, 4)
```

```
            if(arePants):
                result = getTopPants(image, imageMask, corners)
            # Calculate the diferent execution times
            iterEnd = time.time()
            duration = iterEnd-iterStart
            if(duration > maxTime):
                maxTime = duration
            if(duration < minTime):
                minTime = duration
            acom = acom +duration

            if(arePants):
                route = saveImage(result,finalRoute, pathName, iPath, fi
les)
    print("The average execution time was: "+ str(acom/80))
    print("The minimum execution time was: "+ str(minTime))
    print("The maximum execution time was: "+ str(maxTime))

x1 = 700
x2 = 3250
y1 = 0
y2 = 50000
cutMatrix = [x1,x2,y1,y2]
#getPants(AllPaths, cutMatrix)
```

A3. Objectiu B

Objective B: Classification of pants into four categories

This Jupyter notebook handles the completition of the second objective: Classification of pants into four categories for the TFG by Pau Solsona.

```
onlyMethods = False
```

Libraries

```
import os
import cv2
import csv
import math
import asyncio
import imutils
import Augmentor
import numpy as np
import pandas as pd
import multiprocess
```

```
import pickle
from joblib import dump, load
from multiprocess import Process, Queue
from tqdm.notebook import tqdm
from sklearn import metrics
from matplotlib import pyplot as plt
from sklearn.cluster import MiniBatchKMeans
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import precision_score, recall_score, confusion_mat
rix, classification_report, accuracy_score, f1_score
import itertools

%matplotlib inline
%run ./Image_Visualizer.ipynb #Delivers methods to handle images
```

Load data

```
RowHeader = ['file','label']

def generateCSV(rootPath, CSV):
    labels = ["Reverse_Down","Reverse_Up","Right_Down", "Right_Up"]
    filesCSV = []
    for iPath, path in enumerate(labels):
        pathName = labels[iPath]
        folderPath = os.path.join(rootPath, path)
        for files in os.listdir(folderPath):
            finalPath = os.path.join(folderPath, files)
            filesCSV.append([finalPath,pathName])

    with open(CSV, 'w', newline='') as csvfile:
        csvwriter = csv.writer(csvfile)
        csvwriter.writerow(RowHeader)
        for row in filesCSV:
            csvwriter.writerow(row)
```

CSV for test images

We load the CSV that will be used to test the model, via it's CSV

```
baseTestPath = r'DataSets\ValidTrimmed'
baseTestCSV = r'DataSets\ValidTrimmed\pants_dataset.csv'

generateCSV(baseTestPath,baseTestCSV)
```

CSV for train images

We generate new pictures with the Augmentor generator

```
baseTrainPath = r'DataSets\ValidTrimmed\output'
```

```
#Farem serviro Augmentor per a generar imatges
p = Augmentor.Pipeline(baseTestPath)
p.flip_left_right(0.5)
p.rotate(probability=0.5, max_left_rotation=15, max_right_rotation=15)
p.skew(0.4, 0.5)
p.zoom(probability = 0.2, min_factor = 1.1, max_factor = 1.3)
p.shear(0.3, 2,2)
p.status()
```

```
Initialised with 94 image(s) found.
Output directory set to DataSets\ValidTrimmed\output.Operations: 5
      0: Flip (probability=0.5 top_bottom_left_right=LEFT_RIGHT )
      1: RotateRange (probability=0.5 max_left_rotation=-15 max_right_ro
tation=15 )
      2: Skew (probability=0.4 skew_type=RANDOM magnitude=0.5 )
      3: Zoom (probability=0.2 min_factor=1.1 max_factor=1.3 )
      4: Shear (probability=0.3 max_shear_left=2 max_shear_right=2 )
Images: 94
Classes: 4
      Class index: 0 Class label: Reverse_Down
      Class index: 1 Class label: Reverse_Up
      Class index: 2 Class label: Right_Down
      Class index: 3 Class label: Right_Up
Dimensions: 1
      Width: 255 Height: 255
Formats: 1
       JPEG

You can remove operations using the appropriate index and the remove_ope
ration(index) function.
```

```
if(onlyMethods == False):
    p.sample(500)
```

```
Processing <PIL.Image.Image image mode=RGB size=255x255 at 0x253CE685C30
>: 100%|| 500/500 [00:01<00:00, 341.60 Samples
```

```
baseTrainCSV = r'DataSets\ValidTrimmed\output\pants_train.csv'
```

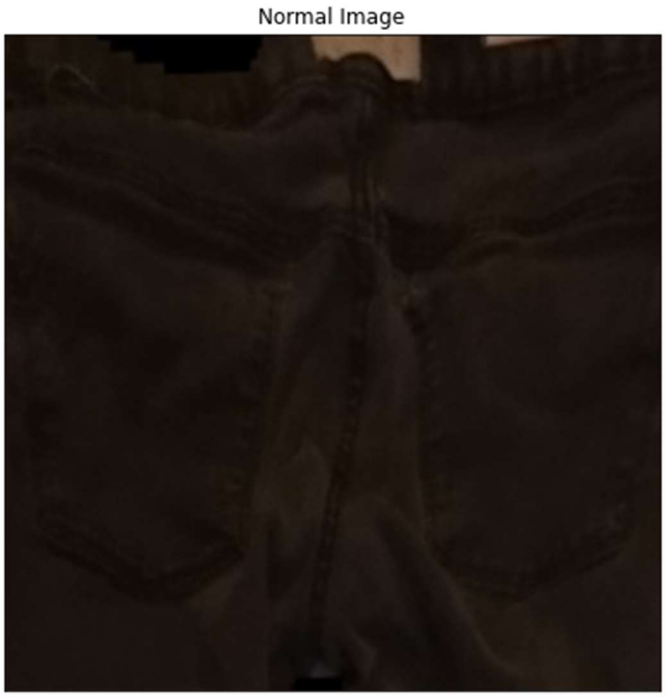
```
generateCSV(baseTrainPath,baseTrainCSV)
```

Visualize Generated Images

```
NormalImageR = r'DataSets\ValidTrimmed\Reverse_Down\P_0_1652555127475.jp
g'
AugmentedImageR = r'DataSets\ValidTrimmed\output\Reverse_Down_original_P
_0_1652555127475.jpg_f24e2bbb-64ab-4602-b983-518319a3fd97.jpg'
```

```
NormalImage = cv2.imread(NormalImageR)
AugmentedImage = cv2.imread(AugmentedImageR)
```

```
showimage(NormalImage, pltTitle= "Normal Image")
showimage(AugmentedImage, pltTitle= "Augmented Image")
```



Training the SVM

```
def generateDescriptors(CSV):
    imagesCSV = pd.read_csv(CSV)
    #https://www.kaggle.com/code/pierre54/bag-of-words-model-with-sift-d
escriptors/notebook
    #Creating array of descriptors
    descriptorsArray = []
    sift = cv2.SIFT_create()
    NoneType = type(None)
    for index in tqdm(range(0, len(imagesCSV.file))):
        image = imagesCSV.file[index]
        img = cv2.imread(image)
        kp, des = sift.detectAndCompute(img, None)
        if(type(des) != NoneType):
            for d in des:
                descriptorsArray.append(d)
    return descriptorsArray

def clusterDescriptors(descriptorsArray, CSV, imagesPath):
    imagesCSV = pd.read_csv(CSV)
    labels = imagesCSV.label.sort_values().unique()
    #Cluster the descriptors
    k = np.size(labels) * 10
    batch_size = np.size(os.listdir(imagesPath)) * 3
    kmeans = MiniBatchKMeans(n_clusters=k, batch_size=batch_size, verbos
e=1).fit(descriptorsArray)
    return kmeans

def generateHistograms(CSV, kmeans):
    #Generate Histograms
    trainCSV = pd.read_csv(CSV)
    labels = trainCSV.label.sort_values().unique()
    kmeans.verbose = False
    histo_list = []
    sift = cv2.SIFT_create()
    NoneType = type(None)
    k = np.size(labels) * 10
    for index in tqdm(range(0, len(trainCSV.file))):
        imagePath = trainCSV.file[index]
        img = cv2.imread(imagePath)
        kp, des = sift.detectAndCompute(img, None)
        histo = np.zeros(k)
        nkp = np.size(kp)
        if(type(des) != NoneType):
            for d in des:
                idx = kmeans.predict([d])
                histo[idx] += 1/nkp # Because we need normalized histogr
ams, I prefere to add 1/nkp directly
```

```
            histo_list.append(histo)
    return histo_list

def trainSVM(histo_list, CSV):
    trainCSV = pd.read_csv(CSV)
    labels = trainCSV.label.sort_values().unique()

    #Train the SVM
    X = np.array(histo_list)
    Y = []

    # It's a way to convert species name into an integer
    for s in trainCSV.label:
        Y.append(np.min(np.nonzero(labels == s)))
    mlp = MLPClassifier(verbose=True, max_iter=600000)
    mlp.fit(X, Y)
    return mlp
```

Generating the model

```
def generateModel(CSV, trainPath):
    descriptors = generateDescriptors(CSV)
    kmeans = clusterDescriptors(descriptors, CSV, trainPath)
    dump(kmeans, r'Models\kmeans.joblib')
    histo_list = generateHistograms(CSV, kmeans)
    mlp = trainSVM(histo_list,CSV)
    dump(mlp, r'Models\mlp.joblib')

if(os.path.exists(r'Models\mlp.joblib') and os.path.exists(r'Models\kmea
ns.joblib')):
    mlp = load(r'Models\mlp.joblib')
    kmeans = load(r'Models\kmeans.joblib')
else:
    generateModel(baseTrainCSV, baseTrainPath)
```

Predict a single class

We can use this method to predict the image into it's diferent classes

```
def predictImage(image, kmeans, mlp):
    NoneType = type(None)
    labels = ["Reverse_Down","Reverse_Up","Right_Down", "Right_Up"]
    k = np.size(labels) * 10
    sift = cv2.SIFT_create()
    kp, des = sift.detectAndCompute(image, None)
    histo = np.zeros(k)
    nkp = np.size(kp)
    if(type(des) != NoneType):
        for d in des:
            idx = kmeans.predict([d])
            histo[idx] += 1/nkp # Because we need normalized histograms,
```

```
I prefere to add 1/nkp directly
    predictions = mlp.predict_proba([histo])[0]
    index = np.where(predictions == np.amax(predictions))[0][0]
    return index

if(onlyMethods):
    raise SystemExit("Methods loaded")
```

Observing results

```
testCSV = pd.read_csv(baseTestCSV)

histo_list_test = []
sift = cv2.SIFT_create()
trainCSV = pd.read_csv(baseTestCSV)
labels = trainCSV.label.sort_values().unique()
k = np.size(labels) * 10
NoneType = type(None)
for index in tqdm(range(0, len(testCSV.file))):
    imagePath = testCSV.file[index]
    img = cv2.imread(imagePath)
    kp, des = sift.detectAndCompute(img, None)
    histo = np.zeros(k)
    nkp = np.size(kp)
    if(type(des) != NoneType):
        for d in des:
            idx = kmeans.predict([d])
            histo[idx] += 1/nkp # Because we need normalized histograms,
I prefere to add 1/nkp directly

    histo_list_test.append(histo)

X = np.array(histo_list_test)
Y = []

# It's a way to convert species name into an integer
for s in testCSV.label:
    Y.append(np.min(np.nonzero(labels == s)))

{"model_id":"9b2f376c6caa48a9abd0350961b155f7","version_major":2,"versio
n_minor":0}

"""
Function extracted from https://deeplizard.com/learn/video/km7pxKy4UHU t
o create a confusion matrix
"""
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
```

```
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    print(cm)
    #plt.figure(figsize = [15,15])
    if normalize:
        cm = np.round((cm.astype('float') / cm.sum(axis=1)[:, np.newaxis
])*100, 1)
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])
):
        text = cm[i, j]
        if normalize:
            text = str(cm[i, j])+"%"
        plt.text(j, i, text,
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    #plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Results

Once the models have been trained, we can test and observe the diferent results

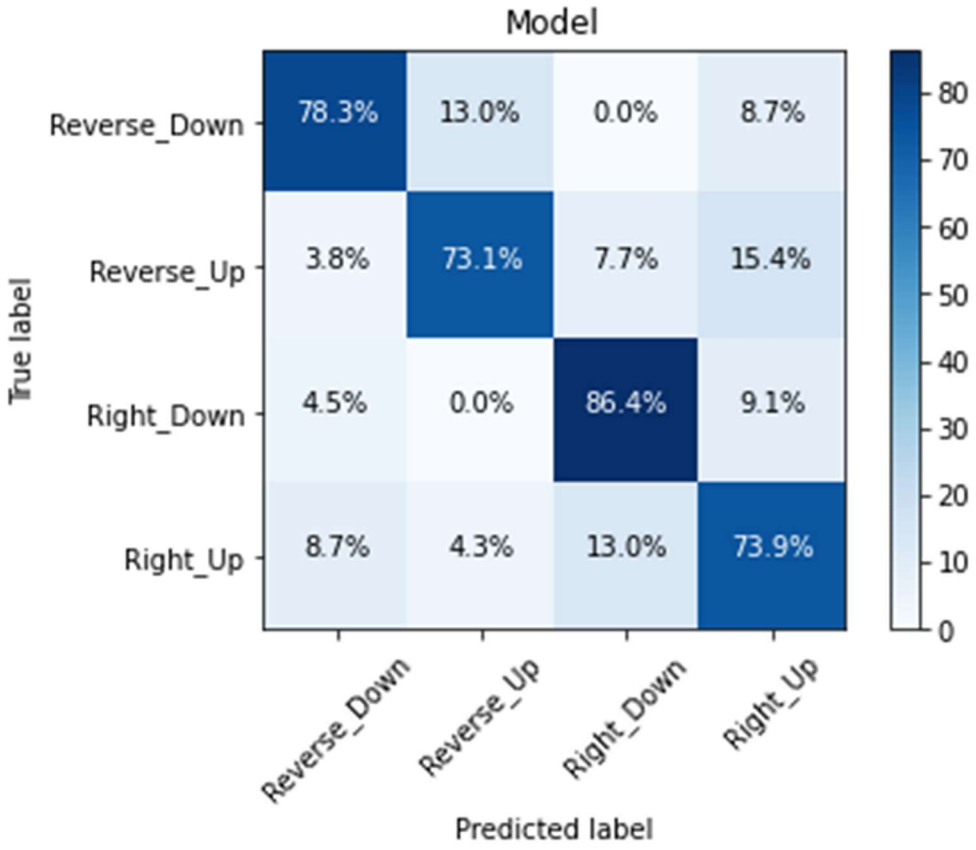
```
y_pred = mlp.predict(X)
print ("Acc. score:", accuracy_score(Y, y_pred))
```

```
Acc. score: 0.776595744680851
```

```
cm = confusion_matrix(y_true=Y, y_pred=y_pred)
plot_confusion_matrix(cm=cm, normalize = True,classes=labels, title='Mod
el')
```

```
[[18  3  0  2]
 [ 1 19  2  4]
 [ 1  0 19  2]
```

```
 [ 2  1  3 17]]
Normalized confusion matrix
[[78.3 13.   0.   8.7]
 [ 3.8 73.1  7.7 15.4]
 [ 4.5  0.  86.4  9.1]
 [ 8.7  4.3 13.  73.9]]
```



Annex B: Intents Previs

B1. Intent Previ 1

Detecting straight from dataset

Libraries

```
import os
import cv2
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image as Image
import Augmentor
import itertools

%matplotlib inline

# Libraries for TensorFlow
from tensorflow.keras.preprocessing import image as imagePrep
from tensorflow.keras import models, layers
from tensorflow.keras.models import clone_model
from tensorflow.keras.applications import resnet50
from tensorflow.keras import backend as K

from tensorflow.keras.applications.resnet50 import preprocess_input

import tensorflow as tf

from sklearn.metrics import precision_score, recall_score, confusion_mat
rix, classification_report, accuracy_score, f1_score
from sklearn.utils import class_weight
from sklearn.model_selection import KFold
from tensorflow.keras import backend as K

print("Num GPUs Available: ", tf.config.list_physical_devices())
K.image_data_format()
```

```
Num GPUs Available:  [PhysicalDevice(name='/physical_device:CPU:0', devi
ce_type='CPU'), PhysicalDevice(name='/physical_device:GPU:0', device_typ
e='GPU')]
```

```
'channels_last'
```

```
Labels = ['file','label']
ReverseDownPath = r'Data\OriginalPantsDataset\Reverse_Down'
```

```
ReverseUpPath = r'Data\OriginalPantsDataset\Reverse_Up'
RightDownPath = r'Data\OriginalPantsDataset\Right_Down'
RightUpPath = r'Data\OriginalPantsDataset\Right_Up'
AllPaths = [ReverseDownPath,ReverseUpPath,RightDownPath,RightUpPath]
originalFilename = r'Data\OriginalPantsDataset\pants_dataset.csv'
```

Creating CSV Images classification

```
allFiles = []
for iPath, path in enumerate(AllPaths):
    if(iPath == 0):
        pathName = "Reverse_Back"
    elif(iPath == 1):
        pathName = "Reverse_Front"
    elif(iPath == 2):
        pathName = "Alright_Back"
    elif(iPath == 3):
        pathName = "Alright_Front"

    for files in os.listdir(path):
        finalPath = os.path.join(path, files)
        allFiles.append([finalPath,pathName])

with open(originalFilename, 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    csvwriter.writerow(Labels)
    for row in allFiles:
        csvwriter.writerow(row)
```

Data augmentation

Let's create more data from the existing one. We want to load images, compress them, make 100 variations of each, and save them

```
BATCH_SIZE = 8
EPOCHS = 25
WIDTH = 540
HEIGHT = 540
STEPS_EPOCH = 50
augmentedDataSetPath = r'PantsDataset'

def saveImage(image, folderName, iPath, iRange, iFile):
    name = 'P_'+str(iPath)+str(iFile)+str(iRange)+'.jpg'
    if not os.path.exists(augmentedDataSetPath):
        os.mkdir(augmentedDataSetPath)
    finalDest = os.path.join(augmentedDataSetPath,folderName)
    if not os.path.exists(finalDest):
        os.mkdir(finalDest)
    finalDest = os.path.join(finalDest,name)
```

```
    image.save(finalDest, "JPEG", optimize = True)#, quality = 10)

allNewFiles = []
for iPath, path in enumerate(AllPaths):
    if(iPath == 0):
        pathName = "Reverse_Back"
    elif(iPath == 1):
        pathName = "Reverse_Front"
    elif(iPath == 2):
        pathName = "Alright_Back"
    elif(iPath == 3):
        pathName = "Alright_Front"

    for iFile, file in enumerate(os.listdir(path)):
        fullPath = os.path.join(path, file)
        image = Image.open(fullPath)
        image = image.resize((WIDTH, HEIGHT))
        saveImage(image, pathName,iPath, 0, iFile)
        print('Compressed ' + file)
```

```
Compressed IMG_20220329_190235.jpg
Compressed IMG_20220329_190237.jpg
Compressed IMG_20220329_190239.jpg
Compressed IMG_20220329_190241.jpg
Compressed IMG_20220329_190243.jpg
Compressed IMG_20220329_190245.jpg
Compressed IMG_20220329_190248.jpg
Compressed IMG_20220329_190251.jpg
Compressed IMG_20220329_190254.jpg
Compressed IMG_20220329_190256.jpg
Compressed IMG_20220329_190543.jpg
Compressed IMG_20220329_190549.jpg
Compressed IMG_20220329_190552.jpg
Compressed IMG_20220329_190557.jpg
Compressed IMG_20220329_190600.jpg
Compressed IMG_20220329_190602.jpg
Compressed IMG_20220329_190605.jpg
Compressed IMG_20220329_190608.jpg
Compressed IMG_20220329_190610.jpg
Compressed IMG_20220329_190630.jpg
Compressed IMG_20220329_190142.jpg
Compressed IMG_20220329_190145.jpg
Compressed IMG_20220329_190148.jpg
Compressed IMG_20220329_190150.jpg
Compressed IMG_20220329_190153.jpg
Compressed IMG_20220329_190159.jpg
Compressed IMG_20220329_190202.jpg
Compressed IMG_20220329_190204.jpg
Compressed IMG_20220329_190207.jpg
Compressed IMG_20220329_190211.jpg
```

```
Compressed IMG_20220329_190450.jpg
Compressed IMG_20220329_190452.jpg
Compressed IMG_20220329_190456.jpg
Compressed IMG_20220329_190458.jpg
Compressed IMG_20220329_190504.jpg
Compressed IMG_20220329_190506.jpg
Compressed IMG_20220329_190510.jpg
Compressed IMG_20220329_190514.jpg
Compressed IMG_20220329_190516.jpg
Compressed IMG_20220329_190533.jpg
Compressed IMG_20220329_190041.jpg
Compressed IMG_20220329_190045.jpg
Compressed IMG_20220329_190048.jpg
Compressed IMG_20220329_190052.jpg
Compressed IMG_20220329_190102.jpg
Compressed IMG_20220329_190106.jpg
Compressed IMG_20220329_190108.jpg
Compressed IMG_20220329_190111.jpg
Compressed IMG_20220329_190115.jpg
Compressed IMG_20220329_190119.jpg
Compressed IMG_20220329_190344.jpg
Compressed IMG_20220329_190346.jpg
Compressed IMG_20220329_190348.jpg
Compressed IMG_20220329_190353.jpg
Compressed IMG_20220329_190355.jpg
Compressed IMG_20220329_190358.jpg
Compressed IMG_20220329_190400.jpg
Compressed IMG_20220329_190402.jpg
Compressed IMG_20220329_190404.jpg
Compressed IMG_20220329_190406.jpg
Compressed IMG_20220329_185600.jpg
Compressed IMG_20220329_185604.jpg
Compressed IMG_20220329_185609.jpg
Compressed IMG_20220329_185614.jpg
Compressed IMG_20220329_185624.jpg
Compressed IMG_20220329_185635.jpg
Compressed IMG_20220329_185647.jpg
Compressed IMG_20220329_185651.jpg
Compressed IMG_20220329_185657.jpg
Compressed IMG_20220329_185705.jpg
Compressed IMG_20220329_185759.jpg
Compressed IMG_20220329_185802.jpg
Compressed IMG_20220329_185805.jpg
Compressed IMG_20220329_185808.jpg
Compressed IMG_20220329_185814.jpg
Compressed IMG_20220329_185849.jpg
Compressed IMG_20220329_185852.jpg
Compressed IMG_20220329_185856.jpg
Compressed IMG_20220329_185859.jpg
Compressed IMG_20220329_185902.jpg
```

Making the generators

```
p = Augmentor.Pipeline(augmentedDataSetPath)
p.resize(1, WIDTH, HEIGHT)
p.flip_left_right(0.5)
p.rotate(probability=0.3, max_left_rotation=5, max_right_rotation=5)
p.skew(0.4, 0.5)#p.zoom(probability = 0.2, min_factor = 1.1, max_factor
= 1.5)
p.status()
```

```
Initialised with 80 image(s) found.
Output directory set to PantsDataset\output.Operations: 4
      0: Resize (probability=1 width=540 height=540 resample_filter=BICU
BIC )
      1: Flip (probability=0.5 top_bottom_left_right=LEFT_RIGHT )
      2: RotateRange (probability=0.3 max_left_rotation=-5 max_right_rot
ation=5 )
      3: Skew (probability=0.4 skew_type=RANDOM magnitude=0.5 )
Images: 80
Classes: 4
      Class index: 0 Class label: Alright_Back
      Class index: 1 Class label: Alright_Front
      Class index: 2 Class label: Reverse_Back
      Class index: 3 Class label: Reverse_Front
Dimensions: 1
      Width: 540 Height: 540
Formats: 1
       JPEG

You can remove operations using the appropriate index and the remove_ope
ration(index) function.
```

```
p.sample(1)
```

```
Processing <PIL.Image.Image image mode=RGB size=540x540 at 0x220BA23B670
>: 100%|███| 1/1 [00:00<00:00, 16.39 Samples/s]
```

```
train_set = p.keras_generator(batch_size=BATCH_SIZE)
```

```
csvFile = pd.read_csv('Data\OriginalPantsDataset\pants_dataset.csv')
csvFile.head()
generator = imagePrep.ImageDataGenerator()
test_set = generator.flow_from_dataframe(
    dataframe=csvFile,
    x_col="file",
    y_col="label",
    target_size=(WIDTH, HEIGHT),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training',
```

```
    shuffle=True
)
```

```
Found 80 validated image filenames belonging to 4 classes.
```

```
def checkPointCreation(checkpoint_path):
    print("saving checkpoint")
    checkpoint_dir = os.path.dirname(checkpoint_path)

    # Create a callback that saves the model's weights
    cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint
_path,
                                                     save_weights_only=True,
                                                     verbose=1)
    return cp_callback

def generateWeights(labels):
    class_weights = class_weight.compute_class_weight('balanced',classes
= np.unique(labels),y=labels)
    leng = max(labels)
    class_weight_dict = {i : class_weights[i] for i in range(leng+1)}
    print(class_weight_dict)
    return class_weight_dict

def generateModel(classesAmount):
    input_layer = layers.Input(shape=(WIDTH,HEIGHT,3)) #We set the input
layer to the data format we desire

    resNet=resnet50.ResNet50(weights='imagenet', input_tensor=input_laye
r,include_top=False) #We get the resnet model
    last_layer=resNet.output #We take output layers of resnet
    flatten=layers.Flatten()(last_layer) # Add flatten layer: we are ext
ending Neural Network by adding flattn layer

    # Add dense layer to the final output layer
    output_layer = layers.Dense(classesAmount,activation='softmax')(flat
ten)

    # Creating model with input and output layer
    model=models.Model(inputs=input_layer,outputs=output_layer)
    model.summary()

    for layer in model.layers[:-1]: #Freezing lower layers (in this case
only lower layer)
        layer.trainable=False
    model.summary()
    return clone_model(model)

def load_weights(cp_path):
    print("Getting weights")
    current_Epoch = 0
    filepath = ''
```

```
    if os.path.exists(cp_path):
        for filename in os.listdir(cp_path):
            f = os.path.join(cp_path, filename)
            # checking if it is a file
            if os.path.isfile(f):
                try:
                    lsp = filename.split('-')[1]
                    num = int(lsp.split('.')[0])
                    if(current_Epoch <= num):
                        current_Epoch = num
                        file = filename.split('.index')[0]
                        filepath = os.path.join(cp_path, file)
                        print(filepath)
                except:
                    print("not this")
        print("Starting from epoch " + str(current_Epoch))
    else:
        os.mkdir(cp_path)
    return current_Epoch, filepath

"""
Function extracted from https://deeplizard.com/learn/video/km7pxKy4UHU t
o create a confusion matrix
"""
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    #plt.figure(figsize = [15,15])
    if normalize:
        cm = np.round((cm.astype('float') / cm.sum(axis=1)[:, np.newaxis
])*100, 1)
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])
```

```
):
        text = cm[i, j]
        if normalize:
            text = str(cm[i, j])+"%"
        plt.text(j, i, text,
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    #plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

First Test

```
cp_path = "checkPoints/model/cp-{epoch:04d}.ckpt"
cp_callback = checkPointCreation(cp_path)
```

```
saving checkpoint
```

```
model = generateModel(len(p.class_labels))
model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=
['accuracy'])
```

```
Model: "model"
_________________________________________________________________________
_________________________
 Layer (type)                Output Shape              Param #     Connect
ed to
=========================================================================
=========================
 input_1 (InputLayer)        [(None, 540, 540, 3  0           []
                             )]

 conv1_pad (ZeroPadding2D)   (None, 546, 546, 3)  0           ['input
_1[0][0]']

 conv1_conv (Conv2D)         (None, 270, 270, 64  9472        ['conv1
_pad[0][0]']
                             )

 conv1_bn (BatchNormalization)  (None, 270, 270, 64  256         ['conv1
_conv[0][0]']
                             )

 conv1_relu (Activation)     (None, 270, 270, 64  0           ['conv1
_bn[0][0]']
                             )

 pool1_pad (ZeroPadding2D)   (None, 272, 272, 64  0           ['conv1
_relu[0][0]']
```

```
                                )

 pool1_pool (MaxPooling2D)      (None, 135, 135, 64  0          ['pool1
_pad[0][0]']
                                )

 conv2_block1_1_conv (Conv2D)   (None, 135, 135, 64  4160       ['pool1
_pool[0][0]']
                                )

 conv2_block1_1_bn (BatchNormal  (None, 135, 135, 64  256       ['conv2
_block1_1_conv[0][0]']
 ization)                       )

 conv2_block1_1_relu (Activatio  (None, 135, 135, 64  0         ['conv2
_block1_1_bn[0][0]']
 n)                             )

 conv2_block1_2_conv (Conv2D)   (None, 135, 135, 64  36928      ['conv2
_block1_1_relu[0][0]']
                                )

 conv2_block1_2_bn (BatchNormal  (None, 135, 135, 64  256       ['conv2
_block1_2_conv[0][0]']
 ization)                       )

 conv2_block1_2_relu (Activatio  (None, 135, 135, 64  0         ['conv2
_block1_2_bn[0][0]']
 n)                             )

 conv2_block1_0_conv (Conv2D)   (None, 135, 135, 25  16640      ['pool1
_pool[0][0]']
                                6)

 conv2_block1_3_conv (Conv2D)   (None, 135, 135, 25  16640      ['conv2
_block1_2_relu[0][0]']
                                6)

 conv2_block1_0_bn (BatchNormal  (None, 135, 135, 25  1024      ['conv2
_block1_0_conv[0][0]']
 ization)                       6)

 conv2_block1_3_bn (BatchNormal  (None, 135, 135, 25  1024      ['conv2
_block1_3_conv[0][0]']
 ization)                       6)

 conv2_block1_add (Add)         (None, 135, 135, 25  0          ['conv2
_block1_0_bn[0][0]',
                                6)                               'conv2
_block1_3_bn[0][0]']
```

```
 conv2_block1_out (Activation)  (None, 135, 135, 25  0           ['conv2
_block1_add[0][0]']
                                6)

 conv2_block2_1_conv (Conv2D)   (None, 135, 135, 64  16448       ['conv2
_block1_out[0][0]']
                                )

 conv2_block2_1_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
_block2_1_conv[0][0]']
 ization)                       )

 conv2_block2_1_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block2_1_bn[0][0]']
 n)                             )

 conv2_block2_2_conv (Conv2D)   (None, 135, 135, 64  36928       ['conv2
_block2_1_relu[0][0]']
                                )

 conv2_block2_2_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
_block2_2_conv[0][0]']
 ization)                       )

 conv2_block2_2_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block2_2_bn[0][0]']
 n)                             )

 conv2_block2_3_conv (Conv2D)   (None, 135, 135, 25  16640       ['conv2
_block2_2_relu[0][0]']
                                6)

 conv2_block2_3_bn (BatchNormal  (None, 135, 135, 25  1024       ['conv2
_block2_3_conv[0][0]']
 ization)                       6)


 conv2_block2_add (Add)         (None, 135, 135, 25  0           ['conv2
_block1_out[0][0]',
                                6)                               'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 135, 135, 25  0           ['conv2
_block2_add[0][0]']
                                6)

 conv2_block3_1_conv (Conv2D)   (None, 135, 135, 64  16448       ['conv2
_block2_out[0][0]']
                                )

 conv2_block3_1_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
```

```
_block3_1_conv[0][0]']
 ization)                       )

 conv2_block3_1_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block3_1_bn[0][0]']
 n)                             )

 conv2_block3_2_conv (Conv2D)   (None, 135, 135, 64  36928       ['conv2
_block3_1_relu[0][0]']
                                )

 conv2_block3_2_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
_block3_2_conv[0][0]']
 ization)                       )

 conv2_block3_2_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block3_2_bn[0][0]']
 n)                             )

 conv2_block3_3_conv (Conv2D)   (None, 135, 135, 25  16640       ['conv2
_block3_2_relu[0][0]']
                                6)

 conv2_block3_3_bn (BatchNormal  (None, 135, 135, 25  1024       ['conv2
_block3_3_conv[0][0]']
 ization)                       6)

 conv2_block3_add (Add)         (None, 135, 135, 25  0           ['conv2
_block2_out[0][0]',
                                6)                               'conv2
_block3_3_bn[0][0]']

 conv2_block3_out (Activation)  (None, 135, 135, 25  0           ['conv2
_block3_add[0][0]']
                                6)

 conv3_block1_1_conv (Conv2D)   (None, 68, 68, 128)  32896       ['conv2
_block3_out[0][0]']

 conv3_block1_1_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 68, 68, 128)  147584      ['conv3
_block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
```

```
_block1_2_conv[0][0]']
 ization)

 conv3_block1_2_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block1_2_bn[0][0]']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 68, 68, 512)  131584      ['conv2
_block3_out[0][0]']

 conv3_block1_3_conv (Conv2D)   (None, 68, 68, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 68, 68, 512)  0          ['conv3
_block1_0_bn[0][0]',
                                                                 'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 68, 68, 512)  0          ['conv3
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 68, 68, 128)  65664       ['conv3
_block1_out[0][0]']

 conv3_block2_1_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 68, 68, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block2_2_bn[0][0]']
 n)
```

```
 conv3_block2_3_conv (Conv2D)   (None, 68, 68, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 68, 68, 512)  0           ['conv3
_block1_out[0][0]',
                                                                  'conv3
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 68, 68, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 68, 68, 128)  65664       ['conv3
_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block3_1_conv[0][0]']
 ization)

 conv3_block3_1_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 68, 68, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 68, 68, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 68, 68, 512)  0           ['conv3
_block2_out[0][0]',
                                                                  'conv3
_block3_3_bn[0][0]']

 conv3_block3_out (Activation)  (None, 68, 68, 512)  0           ['conv3
_block3_add[0][0]']
```

```
 conv3_block4_1_conv (Conv2D)   (None, 68, 68, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 68, 68, 128)  147584      ['conv3
_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 68, 68, 512)  66048       ['conv3
_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block4_3_conv[0][0]']
 ization)

 conv3_block4_add (Add)         (None, 68, 68, 512)  0          ['conv3
_block3_out[0][0]',
                                                                 'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 68, 68, 512)  0          ['conv3
_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 34, 34, 256)  131328      ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block1_1_relu[0][0]']
```

```
 conv4_block1_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block1_2_bn[0][0]']
 n)

 conv4_block1_0_conv (Conv2D)   (None, 34, 34, 1024  525312      ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block1_2_relu[0][0]']
                                )


 conv4_block1_0_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 34, 34, 1024  0          ['conv4
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 34, 34, 1024  0          ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 34, 34, 256)  262400      ['conv4
_block1_out[0][0]']

 conv4_block2_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block2_2_conv[0][0]']
```

```
 ization)

 conv4_block2_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block2_2_relu[0][0]']
                                )

 conv4_block2_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )

 conv4_block2_add (Add)         (None, 34, 34, 1024  0          ['conv4
_block1_out[0][0]',
                                )                                 'conv4
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 34, 34, 1024  0          ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 34, 34, 256)  262400      ['conv4
_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block3_3_conv[0][0]']
```

```
 ization)                       )

 conv4_block3_add (Add)         (None, 34, 34, 1024  0           ['conv4
_block2_out[0][0]',
                                )                                 'conv4
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 34, 34, 1024  0           ['conv4
_block3_add[0][0]']
                                )

 conv4_block4_1_conv (Conv2D)   (None, 34, 34, 256)  262400      ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block4_2_bn[0][0]']

 n)

 conv4_block4_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 34, 34, 1024  0           ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 34, 34, 1024  0           ['conv4
_block4_add[0][0]']
                                )
```

```
 conv4_block5_1_conv (Conv2D)   (None, 34, 34, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block5_1_bn[0][0]']
 n)

 conv4_block5_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block5_2_conv[0][0]']
 ization)

 conv4_block5_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block5_3_conv[0][0]']
 ization)                       )

 conv4_block5_add (Add)         (None, 34, 34, 1024  0          ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 34, 34, 1024  0          ['conv4
_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 34, 34, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
```

```
_block6_1_relu[0][0']

 conv4_block6_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block6_2_conv[0][0']
 ization)

 conv4_block6_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block6_2_bn[0][0']
 n)

 conv4_block6_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block6_2_relu[0][0']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block6_3_conv[0][0']
 ization)                       )

 conv4_block6_add (Add)         (None, 34, 34, 1024  0          ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0']

 conv4_block6_out (Activation)  (None, 34, 34, 1024  0          ['conv4
_block6_add[0][0']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 17, 17, 512)  524800     ['conv4
_block6_out[0][0']

 conv5_block1_1_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block1_1_conv[0][0']
 ization)

 conv5_block1_1_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block1_1_bn[0][0']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 17, 17, 512)  2359808    ['conv5
_block1_1_relu[0][0']

 conv5_block1_2_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block1_2_conv[0][0']
 ization)


 conv5_block1_2_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block1_2_bn[0][0']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 17, 17, 2048  2099200    ['conv4
```

```
_block6_out[0][0]']
                                )

 conv5_block1_3_conv (Conv2D)   (None, 17, 17, 2048  1050624     ['conv5
_block1_2_relu[0][0]']
                                )

 conv5_block1_0_bn (BatchNormal  (None, 17, 17, 2048  8192       ['conv5
_block1_0_conv[0][0]']
 ization)                       )

 conv5_block1_3_bn (BatchNormal  (None, 17, 17, 2048  8192       ['conv5
_block1_3_conv[0][0]']
 ization)                       )

 conv5_block1_add (Add)         (None, 17, 17, 2048  0           ['conv5
_block1_0_bn[0][0]',
                                )                                 'conv5
_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 17, 17, 2048  0           ['conv5
_block1_add[0][0]']
                                )

 conv5_block2_1_conv (Conv2D)   (None, 17, 17, 512)  1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 17, 17, 512)  2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block2_2_bn[0][0]']
 n)

 conv5_block2_3_conv (Conv2D)   (None, 17, 17, 2048  1050624     ['conv5
_block2_2_relu[0][0]']
                                )

 conv5_block2_3_bn (BatchNormal  (None, 17, 17, 2048  8192       ['conv5
```

```
_block2_3_conv[0][0]']
 ization)                       )

 conv5_block2_add (Add)         (None, 17, 17, 2048  0           ['conv5
_block1_out[0][0]',
                                )                                 'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 17, 17, 2048  0           ['conv5
_block2_add[0][0]']
                                )

 conv5_block3_1_conv (Conv2D)   (None, 17, 17, 512)  1049088     ['conv5
_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 17, 17, 512)  2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 17, 17, 2048  1050624     ['conv5
_block3_2_relu[0][0]']
                                )

 conv5_block3_3_bn (BatchNormal  (None, 17, 17, 2048  8192       ['conv5
_block3_3_conv[0][0]']
 ization)                       )

 conv5_block3_add (Add)         (None, 17, 17, 2048  0           ['conv5
_block2_out[0][0]',
                                )                                 'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 17, 17, 2048  0           ['conv5
_block3_add[0][0]']
                                )
```

```
 flatten (Flatten)              (None, 591872)       0           ['conv5
_block3_out[0][0]']

 dense (Dense)                  (None, 4)            2367492     ['flatt
en[0][0]']

==================================================================================
==========================
Total params: 25,955,204
Trainable params: 25,902,084
Non-trainable params: 53,120

__________________________________________________________________________________
__________________________
Model: "model"
__________________________________________________________________________________
__________________________
 Layer (type)                   Output Shape         Param #     Connect
ed to
==================================================================================
==========================
 input_1 (InputLayer)           [(None, 540, 540, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 546, 546, 3)  0           ['input
_1[0][0]']

 conv1_conv (Conv2D)            (None, 270, 270, 64  9472        ['conv1
_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 270, 270, 64  256         ['conv1
_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 270, 270, 64  0           ['conv1
_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 272, 272, 64  0           ['conv1
_relu[0][0]']
                                )

 pool1_pool (MaxPooling2D)      (None, 135, 135, 64  0           ['pool1
_pad[0][0]']
                                )

 conv2_block1_1_conv (Conv2D)   (None, 135, 135, 64  4160        ['pool1
_pool[0][0]']
                                )
```

```
 conv2_block1_1_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
_block1_1_conv[0][0]']
 ization)                       )

 conv2_block1_1_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block1_1_bn[0][0]']
 n)                             )

 conv2_block1_2_conv (Conv2D)   (None, 135, 135, 64  36928       ['conv2
_block1_1_relu[0][0]']
                                )

 conv2_block1_2_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
_block1_2_conv[0][0]']
 ization)                       )

 conv2_block1_2_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block1_2_bn[0][0]']
 n)                             )

 conv2_block1_0_conv (Conv2D)   (None, 135, 135, 25  16640       ['pool1
_pool[0][0]']
                                6)

 conv2_block1_3_conv (Conv2D)   (None, 135, 135, 25  16640       ['conv2
_block1_2_relu[0][0]']
                                6)

 conv2_block1_0_bn (BatchNormal  (None, 135, 135, 25  1024       ['conv2
_block1_0_conv[0][0]']
 ization)                       6)

 conv2_block1_3_bn (BatchNormal  (None, 135, 135, 25  1024       ['conv2
_block1_3_conv[0][0]']
 ization)                       6)

 conv2_block1_add (Add)         (None, 135, 135, 25  0           ['conv2
_block1_0_bn[0][0]',
                                6)                               'conv2
_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 135, 135, 25  0           ['conv2
_block1_add[0][0]']
                                6)

 conv2_block2_1_conv (Conv2D)   (None, 135, 135, 64  16448       ['conv2
_block1_out[0][0]']
                                )

 conv2_block2_1_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
_block2_1_conv[0][0]']
```

```
 ization)                       )

 conv2_block2_1_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block2_1_bn[0][0]']
 n)                             )

 conv2_block2_2_conv (Conv2D)   (None, 135, 135, 64  36928       ['conv2
_block2_1_relu[0][0]']
                                )

 conv2_block2_2_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
_block2_2_conv[0][0]']
 ization)                       )

 conv2_block2_2_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block2_2_bn[0][0]']
 n)                             )

 conv2_block2_3_conv (Conv2D)   (None, 135, 135, 25  16640       ['conv2
_block2_2_relu[0][0]']
                                6)

 conv2_block2_3_bn (BatchNormal  (None, 135, 135, 25  1024       ['conv2
_block2_3_conv[0][0]']
 ization)                       6)


 conv2_block2_add (Add)         (None, 135, 135, 25  0           ['conv2
_block1_out[0][0]',
                                6)                               'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 135, 135, 25  0           ['conv2
_block2_add[0][0]']
                                6)

 conv2_block3_1_conv (Conv2D)   (None, 135, 135, 64  16448       ['conv2
_block2_out[0][0]']
                                )

 conv2_block3_1_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
_block3_1_conv[0][0]']
 ization)                       )

 conv2_block3_1_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block3_1_bn[0][0]']
 n)                             )

 conv2_block3_2_conv (Conv2D)   (None, 135, 135, 64  36928       ['conv2
_block3_1_relu[0][0]']
                                )
```

```
 conv2_block3_2_bn (BatchNormal  (None, 135, 135, 64  256        ['conv2
_block3_2_conv[0][0]']
 ization)                       )

 conv2_block3_2_relu (Activatio  (None, 135, 135, 64  0          ['conv2
_block3_2_bn[0][0]']
 n)                             )

 conv2_block3_3_conv (Conv2D)   (None, 135, 135, 25  16640       ['conv2
_block3_2_relu[0][0]']
                                6)

 conv2_block3_3_bn (BatchNormal  (None, 135, 135, 25  1024       ['conv2
_block3_3_conv[0][0]']
 ization)                       6)

 conv2_block3_add (Add)         (None, 135, 135, 25  0           ['conv2
_block2_out[0][0]',
                                6)                                'conv2
_block3_3_bn[0][0]']

 conv2_block3_out (Activation)  (None, 135, 135, 25  0           ['conv2
_block3_add[0][0]']
                                6)

 conv3_block1_1_conv (Conv2D)   (None, 68, 68, 128)  32896       ['conv2
_block3_out[0][0]']

 conv3_block1_1_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 68, 68, 128)  147584      ['conv3
_block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block1_2_conv[0][0]']
 ization)

 conv3_block1_2_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block1_2_bn[0][0]']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 68, 68, 512)  131584      ['conv2
_block3_out[0][0]']
```

```
 conv3_block1_3_conv (Conv2D)   (None, 68, 68, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 68, 68, 512)  0           ['conv3
_block1_0_bn[0][0]',
                                                                  'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 68, 68, 512)  0           ['conv3
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 68, 68, 128)  65664       ['conv3
_block1_out[0][0]']

 conv3_block2_1_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 68, 68, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block2_2_bn[0][0]']
 n)


 conv3_block2_3_conv (Conv2D)   (None, 68, 68, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 68, 68, 512)  0           ['conv3
_block1_out[0][0]',
```

```
                                                                  'conv3
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 68, 68, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 68, 68, 128)  65664       ['conv3
_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block3_1_conv[0][0]']
 ization)

 conv3_block3_1_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 68, 68, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 68, 68, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 68, 68, 512)  0           ['conv3
_block2_out[0][0]',
                                                                  'conv3
_block3_3_bn[0][0]']

 conv3_block3_out (Activation)  (None, 68, 68, 512)  0           ['conv3
_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 68, 68, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
```

```
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 68, 68, 128)  147584      ['conv3
_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 68, 68, 128)  512        ['conv3
_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 68, 68, 128)  0          ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 68, 68, 512)  66048       ['conv3
_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 68, 68, 512)  2048       ['conv3
_block4_3_conv[0][0]']
 ization)

 conv3_block4_add (Add)         (None, 68, 68, 512)  0          ['conv3
_block3_out[0][0]',
                                                                 'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 68, 68, 512)  0          ['conv3
_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 34, 34, 256)  131328      ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block1_2_bn[0][0]']
 n)
```

```
 conv4_block1_0_conv (Conv2D)   (None, 34, 34, 1024  525312      ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block1_2_relu[0][0]']

                                )

 conv4_block1_0_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 34, 34, 1024  0           ['conv4
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 34, 34, 1024  0           ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 34, 34, 256)  262400      ['conv4
_block1_out[0][0]']

 conv4_block2_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block2_2_relu[0][0]']
                                )
```

```
 conv4_block2_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )

 conv4_block2_add (Add)         (None, 34, 34, 1024  0          ['conv4
_block1_out[0][0]',
                                )                                 'conv4
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 34, 34, 1024  0          ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 34, 34, 256)  262400     ['conv4
_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 34, 34, 256)  590080     ['conv4
_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 34, 34, 1024  263168     ['conv4
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 34, 34, 1024  0          ['conv4
_block2_out[0][0]',
                                )                                 'conv4
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 34, 34, 1024  0          ['conv4
_block3_add[0][0]']
```

```
                                )

 conv4_block4_1_conv (Conv2D)   (None, 34, 34, 256)  262400      ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block4_2_conv[0][0]']
 ization)


 conv4_block4_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 34, 34, 1024  0          ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 34, 34, 1024  0          ['conv4
_block4_add[0][0]']
                                )

 conv4_block5_1_conv (Conv2D)   (None, 34, 34, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block5_1_bn[0][0]']
```

```
 n)

 conv4_block5_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block5_2_conv[0][0]']
 ization)

 conv4_block5_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block5_3_conv[0][0]']
 ization)                       )

 conv4_block5_add (Add)         (None, 34, 34, 1024  0          ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 34, 34, 1024  0          ['conv4
_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 34, 34, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 34, 34, 256)  590080      ['conv4
_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 34, 34, 256)  1024       ['conv4
_block6_2_conv[0][0]']
 ization)

 conv4_block6_2_relu (Activatio  (None, 34, 34, 256)  0          ['conv4
_block6_2_bn[0][0]']
 n)
```

```
 conv4_block6_3_conv (Conv2D)   (None, 34, 34, 1024  263168      ['conv4
_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 34, 34, 1024  4096       ['conv4
_block6_3_conv[0][0]']
 ization)                       )

 conv4_block6_add (Add)         (None, 34, 34, 1024  0           ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 34, 34, 1024  0           ['conv4
_block6_add[0][0]']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 17, 17, 512)  524800      ['conv4
_block6_out[0][0]']

 conv5_block1_1_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block1_1_bn[0][0]']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 17, 17, 512)  2359808     ['conv5
_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block1_2_conv[0][0]']
 ization)


 conv5_block1_2_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 17, 17, 2048  2099200     ['conv4
_block6_out[0][0]']
                                )

 conv5_block1_3_conv (Conv2D)   (None, 17, 17, 2048  1050624     ['conv5
_block1_2_relu[0][0]']
                                )

 conv5_block1_0_bn (BatchNormal  (None, 17, 17, 2048  8192       ['conv5
_block1_0_conv[0][0]']
```

```
 ization)                       )

 conv5_block1_3_bn (BatchNormal  (None, 17, 17, 2048  8192       ['conv5
_block1_3_conv[0][0]']
 ization)                       )

 conv5_block1_add (Add)         (None, 17, 17, 2048  0           ['conv5
_block1_0_bn[0][0]',
                                )                                 'conv5
_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 17, 17, 2048  0           ['conv5
_block1_add[0][0]']
                                )

 conv5_block2_1_conv (Conv2D)   (None, 17, 17, 512)  1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 17, 17, 512)  2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block2_2_bn[0][0]']
 n)

 conv5_block2_3_conv (Conv2D)   (None, 17, 17, 2048  1050624     ['conv5
_block2_2_relu[0][0]']
                                )

 conv5_block2_3_bn (BatchNormal  (None, 17, 17, 2048  8192       ['conv5
_block2_3_conv[0][0]']
 ization)                       )

 conv5_block2_add (Add)         (None, 17, 17, 2048  0           ['conv5
_block1_out[0][0]',
                                )                                 'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 17, 17, 2048  0           ['conv5
```

```
_block2_add[0][0]']
                                )

 conv5_block3_1_conv (Conv2D)   (None, 17, 17, 512)  1049088     ['conv5
_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 17, 17, 512)  2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 17, 17, 512)  2048       ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 17, 17, 512)  0          ['conv5
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 17, 17, 2048  1050624     ['conv5
_block3_2_relu[0][0]']
                                )

 conv5_block3_3_bn (BatchNormal  (None, 17, 17, 2048  8192       ['conv5
_block3_3_conv[0][0]']
 ization)                       )

 conv5_block3_add (Add)         (None, 17, 17, 2048  0           ['conv5
_block2_out[0][0]',
                                )                                 'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 17, 17, 2048  0           ['conv5
_block3_add[0][0]']
                                )

 flatten (Flatten)              (None, 591872)       0           ['conv5
_block3_out[0][0]']

 dense (Dense)                  (None, 4)            2367492     ['flatt
en[0][0]']

==================================================================
==========================
```

```
Total params: 25,955,204
Trainable params: 2,367,492
Non-trainable params: 23,587,712
_________________________________________________________________
_____________________________
```

```
currentEpoch, file = load_weights("checkPoints/model")
if(currentEpoch != 0):
    model.load_weights(file)
```

```
Getting weights
not this
checkPoints/model\cp-0001.ckpt.data-00000-of-00001
checkPoints/model\cp-0001.ckpt
checkPoints/model\cp-0002.ckpt.data-00000-of-00001
checkPoints/model\cp-0002.ckpt
checkPoints/model\cp-0003.ckpt.data-00000-of-00001
checkPoints/model\cp-0003.ckpt
checkPoints/model\cp-0004.ckpt.data-00000-of-00001
checkPoints/model\cp-0004.ckpt
checkPoints/model\cp-0005.ckpt.data-00000-of-00001
checkPoints/model\cp-0005.ckpt
checkPoints/model\cp-0006.ckpt.data-00000-of-00001
checkPoints/model\cp-0006.ckpt
checkPoints/model\cp-0007.ckpt.data-00000-of-00001
checkPoints/model\cp-0007.ckpt
checkPoints/model\cp-0008.ckpt.data-00000-of-00001
checkPoints/model\cp-0008.ckpt
checkPoints/model\cp-0009.ckpt.data-00000-of-00001
checkPoints/model\cp-0009.ckpt
checkPoints/model\cp-0010.ckpt.data-00000-of-00001
checkPoints/model\cp-0010.ckpt
checkPoints/model\cp-0011.ckpt.data-00000-of-00001
checkPoints/model\cp-0011.ckpt
checkPoints/model\cp-0012.ckpt.data-00000-of-00001
checkPoints/model\cp-0012.ckpt
checkPoints/model\cp-0013.ckpt.data-00000-of-00001
checkPoints/model\cp-0013.ckpt
checkPoints/model\cp-0014.ckpt.data-00000-of-00001
checkPoints/model\cp-0014.ckpt
checkPoints/model\cp-0015.ckpt.data-00000-of-00001
checkPoints/model\cp-0015.ckpt
checkPoints/model\cp-0016.ckpt.data-00000-of-00001
checkPoints/model\cp-0016.ckpt
checkPoints/model\cp-0017.ckpt.data-00000-of-00001
checkPoints/model\cp-0017.ckpt
checkPoints/model\cp-0018.ckpt.data-00000-of-00001
checkPoints/model\cp-0018.ckpt
checkPoints/model\cp-0019.ckpt.data-00000-of-00001
checkPoints/model\cp-0019.ckpt
checkPoints/model\cp-0020.ckpt.data-00000-of-00001
```

```
checkPoints/model\cp-0020.ckpt
checkPoints/model\cp-0021.ckpt.data-00000-of-00001
checkPoints/model\cp-0021.ckpt
checkPoints/model\cp-0022.ckpt.data-00000-of-00001
checkPoints/model\cp-0022.ckpt
checkPoints/model\cp-0023.ckpt.data-00000-of-00001
checkPoints/model\cp-0023.ckpt
checkPoints/model\cp-0024.ckpt.data-00000-of-00001
checkPoints/model\cp-0024.ckpt
checkPoints/model\cp-0025.ckpt.data-00000-of-00001
checkPoints/model\cp-0025.ckpt
checkPoints/model\cp-0026.ckpt.data-00000-of-00001
checkPoints/model\cp-0026.ckpt
checkPoints/model\cp-0027.ckpt.data-00000-of-00001
checkPoints/model\cp-0027.ckpt
Starting from epoch 27
```

```
model.fit(train_set,steps_per_epoch=STEPS_EPOCH,epochs=EPOCHS,initial_ep
och = currentEpoch,verbose=True, validation_data = test_set,callbacks=[c
p_callback])
```

```
<keras.callbacks.History at 0x221f5f13fa0>
```

```
if(currentEpoch < EPOCHS):
    model.save('saved_model/model')
```

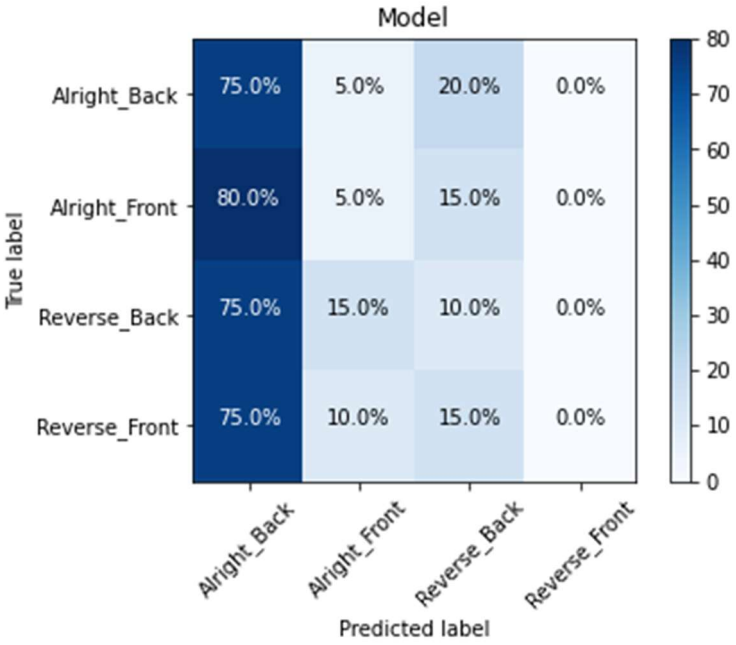
```
model = models.load_model('saved_model/model')
predictions = model.predict(test_set)
rounded_predictions = np.argmax(predictions, axis=-1)
model.evaluate(test_set)
```

```
10/10 [==============================] - 36s 3s/step - loss: 562.1454 -
accuracy: 0.4875
```

```
[562.1453857421875, 0.48750001192092896]
```

```
cm = confusion_matrix(y_true=test_set.labels, y_pred=rounded_predictions
)
plot_confusion_matrix(cm=cm, normalize = True,classes=test_set.class_ind
ices, title='Model')
```

```
Normalized confusion matrix
[[75.  5. 20.  0.]
 [80.  5. 15.  0.]
 [75. 15. 10.  0.]
 [75. 10. 15.  0.]]
```



B2. Intent Previ 2

Machine Learning to detect pants

In this notebook, we will be training our pants model to use on predicitions and therefore, be able to detect the required pants

Library Imports

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import joblib
import itertools

%matplotlib inline
from IPython.display import Image
from PIL import Image as ImagePL
from scipy.ndimage import zoom


# Libraries for TensorFlow
```

```
from tensorflow.keras import preprocessing
from tensorflow.keras.preprocessing import image
from tensorflow.keras import models, layers
from tensorflow.keras.models import clone_model
from tensorflow.keras.applications import resnet50
from tensorflow.keras import backend as K

from tensorflow.keras.applications.resnet50 import preprocess_input

import tensorflow as tf

from sklearn.metrics import precision_score, recall_score, confusion_mat
rix, classification_report, accuracy_score, f1_score
from sklearn.utils import class_weight
from sklearn.model_selection import KFold
```

Load data

Dataset extracted from: https://www.kaggle.com/agrigorev/clothing-dataset-full

References from

https://www.kaggle.com/pintu161/transfer-learning-in-pytorch-using-resnet18/notebook

https://www.kaggle.com/marissafernandes/clothes-image-classifier/notebook

https://www.kaggle.com/viratkothari/image-classification-transfer-learning-resnet50

```
originalCSV = pd.read_csv('Data/images.csv')
originalCSV.head()
```

	image	sender_id	label	kids
0	4285fab0-751a-4b74-8e9b-43af05deee22	124	Not sure	False
1	ea7b6656-3f84-4eb3-9099-23e623fc1018	148	T-Shirt	False
2	00627a3f-0477-401c-95eb-92642cbe078d	94	Not sure	False
3	ea2ffd4d-9b25-4ca8-9dc2-bd27f1cc59fa	43	T-Shirt	False
4	3b86d877-2b9e-4c8b-a6a2-1d87513309d0	189	Shoes	False

```
originalCSV.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5403 entries, 0 to 5402
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   image      5403 non-null   object
 1   sender_id  5403 non-null   int64
 2   label      5403 non-null   object
```

```
 3   kids       5403 non-null   bool
dtypes: bool(1), int64(1), object(2)
memory usage: 132.0+ KB
```

Deleting unnecessary or wrong images

```
def imageCleaning(path):
    extensions = []
    bad_list = []
    for filee in os.listdir(path):
        file_path = os.path.join(path, filee)
        print('** Path: {}  **'.format(file_path), end="\r", flush=True)
        try:
            im = ImagePL.open(file_path)
            rgb_im = im.convert('RGB')
            if filee.split('.')[1] not in extensions:
                extensions.append(filee.split('.')[1])
        except:
            print("Error with " + file_path)
            bad_list.append(file_path)

#imageCleaning('Data\images_compressed')

typeOfData = originalCSV["label"].value_counts()
typeOfData
```

```
T-Shirt       1011
Longsleeve     699
Pants          692
Shoes          431
Shirt          378
Dress          357
Outwear        312
Shorts         308
Not sure       228
Hat            171
Skirt          155
Polo           120
Undershirt     118
Blazer         109
Hoodie         100
Body            69
Other           67
Top             43
Blouse          23
Skip            12
Name: label, dtype: int64
```

Loading Data

```
VALIDATION_SPLIT = 0.2
BATCH_SIZE = 32
WIDTH = 224
HEIGHT = 224
EPOCHS = 20
```

Data set generators

```
def createTrainSetAndValidationSet(data):
    generator = image.ImageDataGenerator(
        rescale=1./255,
        validation_split=VALIDATION_SPLIT
    )
    train_set = generator.flow_from_dataframe(
        dataframe=data,
        x_col="image",
        y_col="label",
        target_size=(WIDTH, HEIGHT),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        subset='training',
        shuffle=True
    )
    val_set = generator.flow_from_dataframe(
        dataframe=data,
        x_col="image",
        y_col="label",
        target_size=(WIDTH, HEIGHT),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        subset='validation',
        shuffle=False
    )
    num_classes = len(train_set.class_indices)
    print("Total classes:", num_classes)
    return train_set, val_set, num_classes
```

Checkpoint generator

```
def checkPointCreation(checkpoint_path):
    checkpoint_dir = os.path.dirname(checkpoint_path)

    # Create a callback that saves the model's weights
    cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint
_path,
                                                     save_weights_only=True,
```

```
                                                    verbose=1)
    return cp_callback
```

Weights generator

```
def generateWeights(labels):
    class_weights = class_weight.compute_class_weight('balanced',classes
= np.unique(labels),y=labels)
    leng = max(labels)
    class_weight_dict = {i : class_weights[i] for i in range(leng+1)}
    print(class_weight_dict)
    return class_weight_dict
```

Model generator

It would be interesting to use a pretrained model so that the training process is slowed down, and potentially, we will get better results. To avoid retraining it, we can use a common model like resNet. Therefore we can make use of pythorch pretrained models. We will split the data into the test and train data set so that we can use it for our pretrained model

https://keras.io/api/layers/

```
def generateModel(classesAmount):
    input_layer = layers.Input(shape=(WIDTH,HEIGHT,3)) #We set the input
layer to the data format we desire

    resNet=resnet50.ResNet50(weights='imagenet', input_tensor=input_laye
r,include_top=False) #We get the resnet model
    last_layer=resNet.output #We take output layers of resnet
    flatten=layers.Flatten()(last_layer) # Add flatten layer: we are ext
ending Neural Network by adding flattn layer

    # Add dense layer to the final output layer
    output_layer = layers.Dense(classesAmount,activation='softmax')(flat
ten)

    # Creating model with input and output layer
    model=models.Model(inputs=input_layer,outputs=output_layer)
    model.summary()

    #for layer in model.layers[:-1]: #Freezing lower layers (in this cas
e only lower layer)
    #    layer.trainable=False
    model.summary()
    return clone_model(model)
```

Once the model is prepared we can train it

Getting weights

Sometimes it's good to retrive the existing weights

```
def load_weights(cp_path):
    print("Getting weights")
    current_Epoch = 0
    filepath = ''
    if os.path.exists(cp_path):
        for filename in os.listdir(cp_path):
            f = os.path.join(cp_path, filename)
            # checking if it is a file
            if os.path.isfile(f):
                try:
                    lsp = filename.split('-')[1]
                    num = int(lsp.split('.')[0])
                    if(current_Epoch <= num):
                        current_Epoch = num
                        file = filename.split('.index')[0]
                        filepath = os.path.join(cp_path, file)
                        print(filepath)
                except:
                    print("not this")
        print("Starting from epoch " + str(current_Epoch))
    return current_Epoch, filepath
```

Confusion matrix visualizer

```
"""
Function extracted from https://deeplizard.com/learn/video/km7pxKy4UHU t
o create a confusion matrix
"""
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    #plt.figure(figsize = [15,15])
    if normalize:
        cm = np.round((cm.astype('float') / cm.sum(axis=1)[:, np.newaxis
])*100, 1)
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
```

```
    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])
):
        text = cm[i, j]
        if normalize:
            text = str(cm[i, j])+"%"
        plt.text(j, i, text,
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    #plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

Frist training pass

We are going to first, try out with the original data, as intended and see the results from there

Original data

```
imagesPath = 'Data/images_compressed'
newCSV = originalCSV.copy()
newCSV['image'] = imagesPath + '/' + originalCSV['image'] +'.jpg' #We ad
just the data so that it is well defined
newCSV = newCSV[['image', 'label']]

train_set, val_set, num_classes = createTrainSetAndValidationSet(newCSV)
```

```
C:\Users\pauso\anaconda3\envs\TFG_Project\lib\site-packages\keras_prepro
cessing\image\dataframe_iterator.py:279: UserWarning: Found 5 invalid im
age filename(s) in x_col="image". These filename(s) will be ignored.
  warnings.warn(

Found 4319 validated image filenames belonging to 20 classes.
Found 1079 validated image filenames belonging to 20 classes.
Total classes: 20

C:\Users\pauso\anaconda3\envs\TFG_Project\lib\site-packages\keras_prepro
cessing\image\dataframe_iterator.py:279: UserWarning: Found 5 invalid im
```

```
age filename(s) in x_col="image". These filename(s) will be ignored.
  warnings.warn(

cp_path = "checkPoints/modelTraining_original_unbalanced/cp-{epoch:04d}.
ckpt"
cp_callback_original_unbalanced= checkPointCreation(cp_path)

originalModel_unbalanced = generateModel(num_classes)
originalModel_unbalanced.compile(loss='categorical_crossentropy', optimi
zer='adam',metrics=['accuracy'])

Model: "model"
_________________________________________________________________________
_________________________
 Layer (type)                Output Shape              Param #     Connect
ed to
=========================================================================
=========================
 input_1 (InputLayer)        [(None, 224, 224, 3  0           []
                             )]

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)  0           ['input
_1[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64  9472        ['conv1
_pad[0][0]']
                             )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256      ['conv1
_conv[0][0]']
                             )

 conv1_relu (Activation)     (None, 112, 112, 64  0           ['conv1
_bn[0][0]']
                             )

 pool1_pad (ZeroPadding2D)   (None, 114, 114, 64  0           ['conv1
_relu[0][0]']
                             )

 pool1_pool (MaxPooling2D)   (None, 56, 56, 64)   0           ['pool1
_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160     ['pool1
_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256      ['conv2
_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0        ['conv2
```

```
_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1
_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_0_bn[0][0]',
                                                                  'conv2
_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_1_bn[0][0]']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block2_1_relu[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
```

```
_block2_2_conv[0][0]']
 ization)

 conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_2_bn[0][0]']
 n)

 conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_out[0][0]',
                                                                  'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block2_add[0][0]']

 conv2_block3_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_1_conv[0][0]']
 ization)


 conv2_block3_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_1_bn[0][0]']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block3_1_relu[0][0]']

 conv2_block3_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_2_conv[0][0]']
 ization)

 conv2_block3_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_2_bn[0][0]']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block3_2_relu[0][0]']

 conv2_block3_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block3_3_conv[0][0]']
 ization)
```

```
 conv2_block3_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block2_out[0][0]',
                                                                  'conv2
_block3_3_bn[0][0]']

 conv2_block3_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block3_add[0][0]']

 conv3_block1_1_conv (Conv2D)   (None, 28, 28, 128)  32896       ['conv2
_block3_out[0][0]']

 conv3_block1_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_2_conv[0][0]']
 ization)

 conv3_block1_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_2_bn[0][0]']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 28, 28, 512)  131584      ['conv2
_block3_out[0][0]']

 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_0_bn[0][0]',
                                                                  'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0           ['conv3
```

```
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block1_out[0][0]']

 conv3_block2_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_2_bn[0][0]']
 n)

 conv3_block2_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_out[0][0]',
                                                                  'conv3
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_1_conv[0][0]']
 ization)


 conv3_block3_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_1_bn[0][0]']
 n)
```

```
 conv3_block3_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block2_out[0][0]',
                                                                 'conv3
_block3_3_bn[0][0]']

 conv3_block3_out (Activation)  (None, 28, 28, 512)  0          ['conv3
_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block4_2_relu[0][0]']
```

```
 conv3_block4_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block4_3_conv[0][0]']
 ization)

 conv3_block4_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block3_out[0][0]',
                                                                 'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 28, 28, 512)  0          ['conv3
_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 14, 14, 256)  131328     ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 14, 14, 256)  590080     ['conv4
_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_2_bn[0][0]']
 n)

 conv4_block1_0_conv (Conv2D)   (None, 14, 14, 1024  525312     ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 14, 14, 1024  263168     ['conv4
_block1_2_relu[0][0]']
                                )

 conv4_block1_0_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 14, 14, 1024  0          ['conv4
```

```
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block1_out[0][0]']


 conv4_block2_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block2_2_relu[0][0]']
                                )

 conv4_block2_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )

 conv4_block2_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block1_out[0][0]',
                                )                                 'conv4
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block2_out[0][0]']
```

```
 conv4_block3_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 14, 14, 256)  590080     ['conv4
_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 14, 14, 1024  263168     ['conv4
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block2_out[0][0]',
                                )                                 'conv4
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block3_add[0][0]']
                                )

 conv4_block4_1_conv (Conv2D)   (None, 14, 14, 256)  262400     ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 14, 14, 256)  590080     ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
```

```
_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block4_add[0][0]']
                                )


 conv4_block5_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_1_bn[0][0]']
 n)

 conv4_block5_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_2_conv[0][0]']
 ization)

 conv4_block5_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )
```

```
 conv4_block5_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block5_3_conv[0][0]']
 ization)                       )

 conv4_block5_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_2_conv[0][0]']
 ization)

 conv4_block6_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_2_bn[0][0]']
 n)

 conv4_block6_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block6_3_conv[0][0]']
 ization)                       )

 conv4_block6_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block6_add[0][0]']
                                )
```

```
 conv5_block1_1_conv (Conv2D)   (None, 7, 7, 512)    524800      ['conv4
_block6_out[0][0]']

 conv5_block1_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_1_bn[0][0]']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_2_conv[0][0]']
 ization)

 conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200     ['conv4
_block6_out[0][0]']

 conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block1_2_relu[0][0]']

 conv5_block1_0_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_0_conv[0][0]']
 ization)

 conv5_block1_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_3_conv[0][0]']
 ization)

 conv5_block1_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_0_bn[0][0]',

                                                                  'conv5
_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block1_add[0][0]']

 conv5_block2_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_1_conv[0][0]']
```

```
 ization)

 conv5_block2_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_2_bn[0][0]']
 n)

 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block2_2_relu[0][0]']

 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_out[0][0]',
                                                                  'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
```

```
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block2_out[0][0]',
                                                                  'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block3_add[0][0]']

 flatten (Flatten)              (None, 100352)       0           ['conv5
_block3_out[0][0]']

 dense (Dense)                  (None, 20)           2007060     ['flatt
en[0][0]']

==================================================================
==========================
Total params: 25,594,772
Trainable params: 25,541,652
Non-trainable params: 53,120
__________________________________________________________________
_________________________
Model: "model"
__________________________________________________________________
_________________________
 Layer (type)                Output Shape         Param #     Connect
ed to
==================================================================
==========================
 input_1 (InputLayer)        [(None, 224, 224, 3  0           []
                             )]

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)  0           ['input
_1[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64  9472        ['conv1
_pad[0][0]']
                             )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256      ['conv1
_conv[0][0]']
                             )
```

```
 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1
_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1
_relu[0][0]']
                                )


 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1
_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1
_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1
_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_0_bn[0][0]',
```

```
                                                                  'conv2
_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_1_bn[0][0]']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block2_1_relu[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_2_conv[0][0]']
 ization)

 conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_2_bn[0][0]']
 n)

 conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_out[0][0]',
                                                                  'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block2_add[0][0]']

 conv2_block3_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_1_conv[0][0]']
 ization)

 conv2_block3_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
```

```
_block3_1_bn[0][0']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block3_1_relu[0][0']

 conv2_block3_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_2_conv[0][0']
 ization)

 conv2_block3_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_2_bn[0][0']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block3_2_relu[0][0']

 conv2_block3_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block3_3_conv[0][0']
 ization)

 conv2_block3_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block2_out[0][0',
                                                                 'conv2
_block3_3_bn[0][0']


 conv2_block3_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block3_add[0][0']

 conv3_block1_1_conv (Conv2D)   (None, 28, 28, 128)  32896       ['conv2
_block3_out[0][0']

 conv3_block1_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_1_conv[0][0']
 ization)

 conv3_block1_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_1_bn[0][0']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block1_1_relu[0][0']

 conv3_block1_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_2_conv[0][0']
 ization)

 conv3_block1_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_2_bn[0][0']
 n)
```

```
 conv3_block1_0_conv (Conv2D)   (None, 28, 28, 512)  131584      ['conv2
_block3_out[0][0]']

 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_0_bn[0][0]',
                                                                  'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block1_out[0][0]']

 conv3_block2_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_2_bn[0][0]']
 n)

 conv3_block2_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block2_3_conv[0][0]']
 ization)
```

```
 conv3_block2_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_out[0][0]',
                                                                  'conv3
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_1_conv[0][0]']
 ization)

 conv3_block3_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block2_out[0][0]',
                                                                  'conv3
_block3_3_bn[0][0]']


 conv3_block3_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
```

```
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block4_3_conv[0][0]']
 ization)

 conv3_block4_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block3_out[0][0]',
                                                                 'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 28, 28, 512)  0          ['conv3
_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 14, 14, 256)  131328      ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_2_conv[0][0]']
 ization)
```

```
 conv4_block1_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_2_bn[0][0]']
 n)

 conv4_block1_0_conv (Conv2D)   (None, 14, 14, 1024  525312      ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block1_2_relu[0][0]']
                                )

 conv4_block1_0_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block1_out[0][0]']

 conv4_block2_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)
```

```
 conv4_block2_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block2_2_relu[0][0]']
                                )

 conv4_block2_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )


 conv4_block2_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block1_out[0][0]',
                                )                                 'conv4
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block2_out[0][0]',
                                )                                 'conv4
```

```
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block3_add[0][0]']
                                )

 conv4_block4_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block4_add[0][0]']
                                )

 conv4_block5_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)
```

```
 conv4_block5_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_1_bn[0][0]']
 n)

 conv4_block5_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_2_conv[0][0]']
 ization)

 conv4_block5_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block5_3_conv[0][0]']

 ization)                       )

 conv4_block5_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_2_conv[0][0]']
 ization)
```

```
 conv4_block6_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_2_bn[0][0]']
 n)

 conv4_block6_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block6_3_conv[0][0]']
 ization)                       )

 conv4_block6_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block6_add[0][0]']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 7, 7, 512)    524800     ['conv4
_block6_out[0][0]']

 conv5_block1_1_bn (BatchNormal  (None, 7, 7, 512)   2048       ['conv5
_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 7, 7, 512)   0          ['conv5
_block1_1_bn[0][0]']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 7, 7, 512)    2359808    ['conv5
_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 7, 7, 512)   2048       ['conv5
_block1_2_conv[0][0]']
 ization)

 conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0          ['conv5
_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200    ['conv4
_block6_out[0][0]']

 conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624    ['conv5
_block1_2_relu[0][0]']

 conv5_block1_0_bn (BatchNormal  (None, 7, 7, 2048)  8192       ['conv5
```

```
_block1_0_conv[0][0]']
 ization)

 conv5_block1_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_3_conv[0][0]']
 ization)

 conv5_block1_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_0_bn[0][0]',
                                                                  'conv5
_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block1_add[0][0]']

 conv5_block2_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_2_bn[0][0]']
 n)


 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block2_2_relu[0][0]']

 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_out[0][0]',
                                                                  'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
```

```
_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block2_out[0][0]',
                                                                  'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block3_add[0][0]']

 flatten (Flatten)              (None, 100352)       0           ['conv5
_block3_out[0][0]']

 dense (Dense)                  (None, 20)           2007060     ['flatt
en[0][0]']

==================================================================
==========================
Total params: 25,594,772
Trainable params: 25,541,652
Non-trainable params: 53,120
```

```
_________________________________________________________________
_____________________________

currentEpoch, file = load_weights("checkPoints/modelTraining_original_un
balanced")
if(currentEpoch != 0):
    originalModel_unbalanced.load_weights(file)
originalModel_unbalanced.fit(train_set,epochs=EPOCHS,initial_epoch = cur
rentEpoch,verbose=True,validation_data=val_set, callbacks=[cp_callback_o
riginal_unbalanced])

Getting weights
not this
checkPoints/modelTraining_original_unbalanced\cp-0001.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0001.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0002.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0002.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0003.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0003.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0004.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0004.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0005.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0005.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0006.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0006.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0007.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0007.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0008.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0008.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0009.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0009.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0010.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0010.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0011.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0011.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0012.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0012.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0013.ckpt.data-00000-of
-00001
```

```
checkPoints/modelTraining_original_unbalanced\cp-0013.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0014.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0014.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0015.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0015.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0016.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0016.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0017.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0017.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0018.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0018.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0019.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0019.ckpt
checkPoints/modelTraining_original_unbalanced\cp-0020.ckpt.data-00000-of
-00001
checkPoints/modelTraining_original_unbalanced\cp-0020.ckpt
Starting from epoch 20
```

```
<keras.callbacks.History at 0x2c468530af0>
```

```
if(currentEpoch < EPOCHS):
    originalModel_unbalanced.save('saved_model/pantsModel_original_unbal
anced')
```

Visualizing

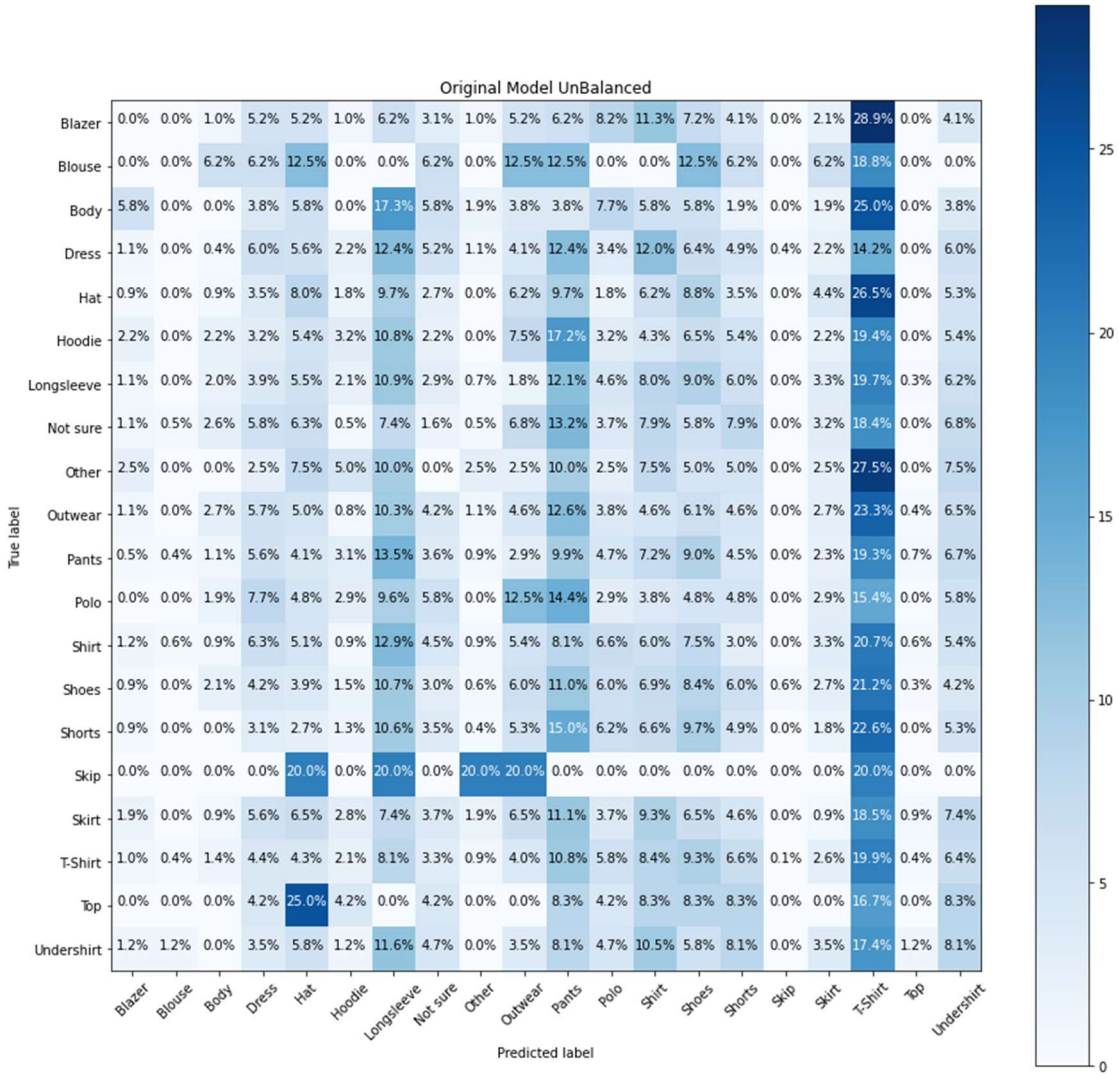
```
originalModel_unbalanced = models.load_model('saved_model/pantsModel_ori
ginal_unbalanced')
print("Model loaded")
predictions = originalModel_unbalanced.predict(train_set)
rounded_predictions = np.argmax(predictions, axis=-1)
originalModel_unbalanced.evaluate(train_set)
```

```
Model loaded
135/135 [==============================] - 20s 144ms/step - loss: 2.1622
- accuracy: 0.7103
```

```
[2.162170648574829, 0.7103496193885803]
```

```
cm = confusion_matrix(y_true=train_set.labels, y_pred=rounded_prediction
s)
plot_confusion_matrix(cm=cm,normalize = True, classes=train_set.class_in
dices, title='Original Model UnBalanced')
```

```
Normalized confusion matrix
[[ 0.   0.   1.   5.2  5.2  1.   6.2  3.1  1.   5.2  6.2  8.2 11.3  7.2
   4.1  0.   2.1 28.9  0.   4.1]
 [ 0.   0.   6.2  6.2 12.5  0.   0.   6.2  0.  12.5 12.5  0.   0.  12.5
   6.2  0.   6.2 18.8  0.   0. ]
 [ 5.8  0.   0.   3.8  5.8  0.  17.3  5.8  1.9  3.8  3.8  7.7  5.8  5.8
   1.9  0.   1.9 25.   0.   3.8]
 [ 1.1  0.   0.4  6.   5.6  2.2 12.4  5.2  1.1  4.1 12.4  3.4 12.   6.4
   4.9  0.4  2.2 14.2  0.   6. ]
 [ 0.9  0.   0.9  3.5  8.   1.8  9.7  2.7  0.   6.2  9.7  1.8  6.2  8.8
   3.5  0.   4.4 26.5  0.   5.3]
 [ 2.2  0.   2.2  3.2  5.4  3.2 10.8  2.2  0.   7.5 17.2  3.2  4.3  6.5
   5.4  0.   2.2 19.4  0.   5.4]
 [ 1.1  0.   2.   3.9  5.5  2.1 10.9  2.9  0.7  1.8 12.1  4.6  8.   9.
   6.   0.   3.3 19.7  0.3  6.2]
 [ 1.1  0.5  2.6  5.8  6.3  0.5  7.4  1.6  0.5  6.8 13.2  3.7  7.9  5.8
   7.9  0.   3.2 18.4  0.   6.8]
 [ 2.5  0.   0.   2.5  7.5  5.  10.   0.   2.5  2.5 10.   2.5  7.5  5.
   5.   0.   2.5 27.5  0.   7.5]
 [ 1.1  0.   2.7  5.7  5.   0.8 10.3  4.2  1.1  4.6 12.6  3.8  4.6  6.1
   4.6  0.   2.7 23.3  0.4  6.5]
 [ 0.5  0.4  1.1  5.6  4.1  3.1 13.5  3.6  0.9  2.9  9.9  4.7  7.2  9.
   4.5  0.   2.3 19.3  0.7  6.7]
 [ 0.   0.   1.9  7.7  4.8  2.9  9.6  5.8  0.  12.5 14.4  2.9  3.8  4.8
   4.8  0.   2.9 15.4  0.   5.8]
 [ 1.2  0.6  0.9  6.3  5.1  0.9 12.9  4.5  0.9  5.4  8.1  6.6  6.   7.5
   3.   0.   3.3 20.7  0.6  5.4]
 [ 0.9  0.   2.1  4.2  3.9  1.5 10.7  3.   0.6  6.  11.   6.   6.9  8.4
   6.   0.6  2.7 21.2  0.3  4.2]
 [ 0.9  0.   0.   3.1  2.7  1.3 10.6  3.5  0.4  5.3 15.   6.2  6.6  9.7
   4.9  0.   1.8 22.6  0.   5.3]
 [ 0.   0.   0.   0.  20.   0.  20.   0.  20.  20.   0.   0.   0.   0.
   0.   0.   0.  20.   0.   0. ]
 [ 1.9  0.   0.9  5.6  6.5  2.8  7.4  3.7  1.9  6.5 11.1  3.7  9.3  6.5
   4.6  0.   0.9 18.5  0.9  7.4]
 [ 1.   0.4  1.4  4.4  4.3  2.1  8.1  3.3  0.9  4.  10.8  5.8  8.4  9.3
   6.6  0.1  2.6 19.9  0.4  6.4]
 [ 0.   0.   0.   4.2 25.   4.2  0.   4.2  0.   0.   8.3  4.2  8.3  8.3
   8.3  0.   0.  16.7  0.   8.3]
 [ 1.2  1.2  0.   3.5  5.8  1.2 11.6  4.7  0.   3.5  8.1  4.7 10.5  5.8
   8.1  0.   3.5 17.4  1.2  8.1]]
```



Original data Balanced

```
cp_path = "checkPoints/modelTraining_original_balanced"
cp_file = "/cp-{epoch:04d}.ckpt"
cp_callback_original_balanced= checkPointCreation(cp_path+cp_file)
class_weight_dict_org = generateWeights(train_set.labels)
```

```
{0: 2.2262886597938145, 1: 13.496875, 2: 4.1528846153846155, 3: 0.808801
4981273408, 4: 1.9110619469026549, 5: 2.3220430107526884, 6: 0.351710097
7198697, 7: 1.1365789473684211, 8: 5.39875, 9: 0.8242366412213741, 10: 0
.3890990990990991, 11: 2.0764423076923078, 12: 0.6484984984984985, 13: 0
.6446268656716417, 14: 0.9555309734513274, 15: 43.19, 16: 1.999537037037
0371, 17: 0.2702753441802253, 18: 8.997916666666667, 19: 2.5110465116279
07}
```

```
originalModel_balanced = generateModel(num_classes)
originalModel_balanced.compile(loss='categorical_crossentropy', optimize
r='adam',metrics=['accuracy'])
```

```
Model: "model_1"
__________________________________________________________________________________________
 Layer (type)                Output Shape              Param #     Connected to
==========================================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0           []

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)       0           ['input_2[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64)      9472        ['conv1_pad[0][0]']

 conv1_bn (BatchNormalization)  (None, 112, 112, 64)   256         ['conv1_conv[0][0]']

 conv1_relu (Activation)     (None, 112, 112, 64)      0           ['conv1_bn[0][0]']

 pool1_pad (ZeroPadding2D)   (None, 114, 114, 64)      0           ['conv1_relu[0][0]']

 pool1_pool (MaxPooling2D)   (None, 56, 56, 64)        0           ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)     4160        ['pool1_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)    256         ['conv2_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)    0           ['conv2_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)     36928       ['conv2_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)    256         ['conv2_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)    0           ['conv2
```

```
_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1
_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_0_bn[0][0]',
                                                                  'conv2
_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_1_bn[0][0]']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block2_1_relu[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_2_conv[0][0]']
 ization)

 conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_2_bn[0][0]']
 n)

 conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
```

```
_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_out[0][0]',
                                                                  'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block2_add[0][0]']

 conv2_block3_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_1_conv[0][0]']
 ization)


 conv2_block3_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_1_bn[0][0]']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block3_1_relu[0][0]']

 conv2_block3_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_2_conv[0][0]']
 ization)

 conv2_block3_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_2_bn[0][0]']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block3_2_relu[0][0]']

 conv2_block3_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block3_3_conv[0][0]']
 ization)

 conv2_block3_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block2_out[0][0]',
                                                                  'conv2
_block3_3_bn[0][0]']

 conv2_block3_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block3_add[0][0]']

 conv3_block1_1_conv (Conv2D)   (None, 28, 28, 128)  32896       ['conv2
_block3_out[0][0]']
```

```
 conv3_block1_1_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block1_2_conv[0][0]']
 ization)

 conv3_block1_2_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block1_2_bn[0][0]']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 28, 28, 512)  131584      ['conv2
_block3_out[0][0]']

 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048        ['conv3
_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048        ['conv3
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_0_bn[0][0]',
                                                                  'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block1_out[0][0]']

 conv3_block2_1_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block2_1_bn[0][0]']
```

```
 n)

 conv3_block2_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_2_bn[0][0]']
 n)

 conv3_block2_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_out[0][0]',
                                                                  'conv3
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_1_conv[0][0]']
 ization)


 conv3_block3_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_2_bn[0][0]']
 n)
```

```
 conv3_block3_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block2_out[0][0]',
                                                                  'conv3
_block3_3_bn[0][0]']

 conv3_block3_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block4_3_conv[0][0]']
 ization)

 conv3_block4_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block3_out[0][0]',
                                                                  'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block4_add[0][0]']
```

```
 conv4_block1_1_conv (Conv2D)   (None, 14, 14, 256)  131328      ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_2_bn[0][0]']
 n)

 conv4_block1_0_conv (Conv2D)   (None, 14, 14, 1024  525312      ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block1_2_relu[0][0]']
                                )

 conv4_block1_0_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block1_out[0][0]']
```

```
 conv4_block2_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 14, 14, 256)  590080     ['conv4
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 14, 14, 1024  263168     ['conv4
_block2_2_relu[0][0]']
                                )

 conv4_block2_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )

 conv4_block2_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block1_out[0][0]',
                                )                                 'conv4
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 14, 14, 256)  262400     ['conv4
_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 14, 14, 256)  590080     ['conv4
_block3_1_relu[0][0]']
```

```
 conv4_block3_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block2_out[0][0]',
                                )                                 'conv4
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block3_add[0][0]']
                                )

 conv4_block4_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )
```

```
 conv4_block4_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block4_add[0][0]']
                                )


 conv4_block5_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_1_bn[0][0]']
 n)

 conv4_block5_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_2_conv[0][0]']
 ization)

 conv4_block5_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block5_3_conv[0][0]']
 ization)                       )

 conv4_block5_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block5_add[0][0]']
```

```
                                )

 conv4_block6_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_2_conv[0][0]']
 ization)

 conv4_block6_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_2_bn[0][0]']
 n)

 conv4_block6_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block6_3_conv[0][0]']
 ization)                       )

 conv4_block6_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block6_add[0][0]']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 7, 7, 512)    524800      ['conv4
_block6_out[0][0]']

 conv5_block1_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_1_bn[0][0]']
 n)
```

```
 conv5_block1_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_2_conv[0][0]']
 ization)

 conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200     ['conv4
_block6_out[0][0]']

 conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block1_2_relu[0][0]']

 conv5_block1_0_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_0_conv[0][0]']
 ization)

 conv5_block1_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_3_conv[0][0]']
 ization)

 conv5_block1_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_0_bn[0][0]',

                                                                  'conv5
_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block1_add[0][0]']

 conv5_block2_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_2_conv[0][0]']
```

```
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_2_bn[0][0]']
 n)

 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block2_2_relu[0][0]']

 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_out[0][0]',
                                                                  'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5
```

```
_block2_out[0][0]',
                                                                 'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block3_add[0][0]']

 flatten_1 (Flatten)            (None, 100352)       0           ['conv5
_block3_out[0][0]']

 dense_1 (Dense)                (None, 20)           2007060     ['flatt
en_1[0][0]']

==================================================================================
==========================
Total params: 25,594,772
Trainable params: 25,541,652
Non-trainable params: 53,120
__________________________________________________________________________________
_________________________
Model: "model_1"
__________________________________________________________________________________
_________________________
 Layer (type)                   Output Shape         Param #     Connect
ed to
==================================================================================
==========================
 input_2 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input
_2[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1
_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1
_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1
_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1
_relu[0][0]']
                                )
```

```
 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1
_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1
_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1
_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_0_bn[0][0]',
                                                                  'conv2
_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
```

```
_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_1_bn[0][0]']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block2_1_relu[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_2_conv[0][0]']
 ization)

 conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_2_bn[0][0]']
 n)

 conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_out[0][0]',
                                                                  'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block2_add[0][0]']

 conv2_block3_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_1_conv[0][0]']
 ization)

 conv2_block3_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_1_bn[0][0]']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block3_1_relu[0][0]']

 conv2_block3_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_2_conv[0][0]']
 ization)
```

```
 conv2_block3_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_2_bn[0][0]']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block3_2_relu[0][0]']

 conv2_block3_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block3_3_conv[0][0]']
 ization)

 conv2_block3_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block2_out[0][0]',
                                                                  'conv2
_block3_3_bn[0][0]']


 conv2_block3_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block3_add[0][0]']

 conv3_block1_1_conv (Conv2D)   (None, 28, 28, 128)  32896       ['conv2
_block3_out[0][0]']

 conv3_block1_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_2_conv[0][0]']
 ization)

 conv3_block1_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_2_bn[0][0]']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 28, 28, 512)  131584      ['conv2
_block3_out[0][0]']

 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_0_conv[0][0]']
 ization)
```

```
 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_0_bn[0][0]',
                                                                  'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block1_out[0][0]']

 conv3_block2_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_2_bn[0][0]']
 n)

 conv3_block2_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_out[0][0]',
                                                                  'conv3
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
```

```
_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_1_conv[0][0]']
 ization)

 conv3_block3_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block2_out[0][0]',
                                                                 'conv3
_block3_3_bn[0][0]']


 conv3_block3_out (Activation)  (None, 28, 28, 512)  0          ['conv3
_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block4_1_relu[0][0]']
```

```
 conv3_block4_2_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 28, 28, 512)  2048        ['conv3
_block4_3_conv[0][0]']
 ization)

 conv3_block4_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block3_out[0][0]',
                                                                  'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 14, 14, 256)  131328      ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 14, 14, 256)  1024        ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 14, 14, 256)  0           ['conv4
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 14, 14, 256)  1024        ['conv4
_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 14, 14, 256)  0           ['conv4
_block1_2_bn[0][0]']
 n)

 conv4_block1_0_conv (Conv2D)   (None, 14, 14, 1024  525312      ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block1_2_relu[0][0]']
```

```
                                )

 conv4_block1_0_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 14, 14, 256)  262400     ['conv4
_block1_out[0][0]']

 conv4_block2_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 14, 14, 256)  590080     ['conv4
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 14, 14, 1024  263168     ['conv4
_block2_2_relu[0][0]']
                                )

 conv4_block2_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )


 conv4_block2_add (Add)         (None, 14, 14, 1024  0          ['conv4
```

```
_block1_out[0][0]',
                                )                                 'conv4
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 14, 14, 256)  262400     ['conv4
_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 14, 14, 256)  1024      ['conv4
_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 14, 14, 256)  0         ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 14, 14, 256)  590080     ['conv4
_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 14, 14, 256)  1024      ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 14, 14, 256)  0         ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 14, 14, 1024  263168     ['conv4
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 14, 14, 1024  4096      ['conv4
_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block2_out[0][0]',
                                )                                 'conv4
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block3_add[0][0]']
                                )

 conv4_block4_1_conv (Conv2D)   (None, 14, 14, 256)  262400     ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 14, 14, 256)  1024      ['conv4
```

```
_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block4_add[0][0]']
                                )

 conv4_block5_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_1_bn[0][0]']
 n)

 conv4_block5_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_2_conv[0][0]']
```

```
 ization)

 conv4_block5_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block5_3_conv[0][0]']

 ization)                       )

 conv4_block5_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_2_conv[0][0]']
 ization)

 conv4_block6_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_2_bn[0][0]']
 n)

 conv4_block6_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
```

```
_block6_3_conv[0][0]']
 ization)                       )

 conv4_block6_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block6_add[0][0]']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 7, 7, 512)    524800      ['conv4
_block6_out[0][0]']

 conv5_block1_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_1_bn[0][0]']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_2_conv[0][0]']
 ization)

 conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200     ['conv4
_block6_out[0][0]']

 conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block1_2_relu[0][0]']

 conv5_block1_0_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_0_conv[0][0]']
 ization)

 conv5_block1_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_3_conv[0][0]']
 ization)

 conv5_block1_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_0_bn[0][0]',
                                                                  'conv5
```

```
_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block1_add[0][0]']

 conv5_block2_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_2_bn[0][0]']
 n)


 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block2_2_relu[0][0]']

 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_out[0][0]',
                                                                 'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
```

```
_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block2_out[0][0]',
                                                                  'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block3_add[0][0]']

 flatten_1 (Flatten)            (None, 100352)       0           ['conv5
_block3_out[0][0]']

 dense_1 (Dense)                (None, 20)           2007060     ['flatt
en_1[0][0]']

==================================================================
==========================
Total params: 25,594,772
Trainable params: 25,541,652
Non-trainable params: 53,120
__________________________________________________________________
_____________________________
```

```
currentEpoch, file = load_weights("checkPoints/modelTraining_original_ba
lanced")
if(currentEpoch != 0):
    originalModel_balanced.load_weights(file)
originalModel_balanced.fit(train_set,epochs=EPOCHS, initial_epoch = curr
entEpoch,verbose=True,validation_data=val_set, callbacks=[cp_callback_or
iginal_balanced], class_weight = class_weight_dict_org)
```

```
Getting weights
not this
checkPoints/modelTraining_original_balanced\cp-0001.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0001.ckpt
checkPoints/modelTraining_original_balanced\cp-0002.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0002.ckpt
checkPoints/modelTraining_original_balanced\cp-0003.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0003.ckpt
checkPoints/modelTraining_original_balanced\cp-0004.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0004.ckpt
checkPoints/modelTraining_original_balanced\cp-0005.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0005.ckpt
checkPoints/modelTraining_original_balanced\cp-0006.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0006.ckpt
checkPoints/modelTraining_original_balanced\cp-0007.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0007.ckpt
checkPoints/modelTraining_original_balanced\cp-0008.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0008.ckpt
checkPoints/modelTraining_original_balanced\cp-0009.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0009.ckpt
checkPoints/modelTraining_original_balanced\cp-0010.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0010.ckpt
checkPoints/modelTraining_original_balanced\cp-0011.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0011.ckpt
checkPoints/modelTraining_original_balanced\cp-0012.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0012.ckpt
checkPoints/modelTraining_original_balanced\cp-0013.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0013.ckpt
checkPoints/modelTraining_original_balanced\cp-0014.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0014.ckpt
checkPoints/modelTraining_original_balanced\cp-0015.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0015.ckpt
checkPoints/modelTraining_original_balanced\cp-0016.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0016.ckpt
```

```
checkPoints/modelTraining_original_balanced\cp-0017.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0017.ckpt
checkPoints/modelTraining_original_balanced\cp-0018.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0018.ckpt
checkPoints/modelTraining_original_balanced\cp-0019.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0019.ckpt
checkPoints/modelTraining_original_balanced\cp-0020.ckpt.data-00000-of-0
0001
checkPoints/modelTraining_original_balanced\cp-0020.ckpt
Starting from epoch 20
```

```
<keras.callbacks.History at 0x2c5f3af4c40>
```

```
if(currentEpoch < EPOCHS):
    originalModel_balanced.save('saved_model/pantsModel_original_balance
d')
```

Visualizing

```
originalModel_balanced = models.load_model('saved_model/pantsModel_origi
nal_balanced')
predictions = originalModel_balanced.predict(train_set)
rounded_predictions = np.argmax(predictions, axis=-1)
originalModel_balanced.evaluate(train_set)
```

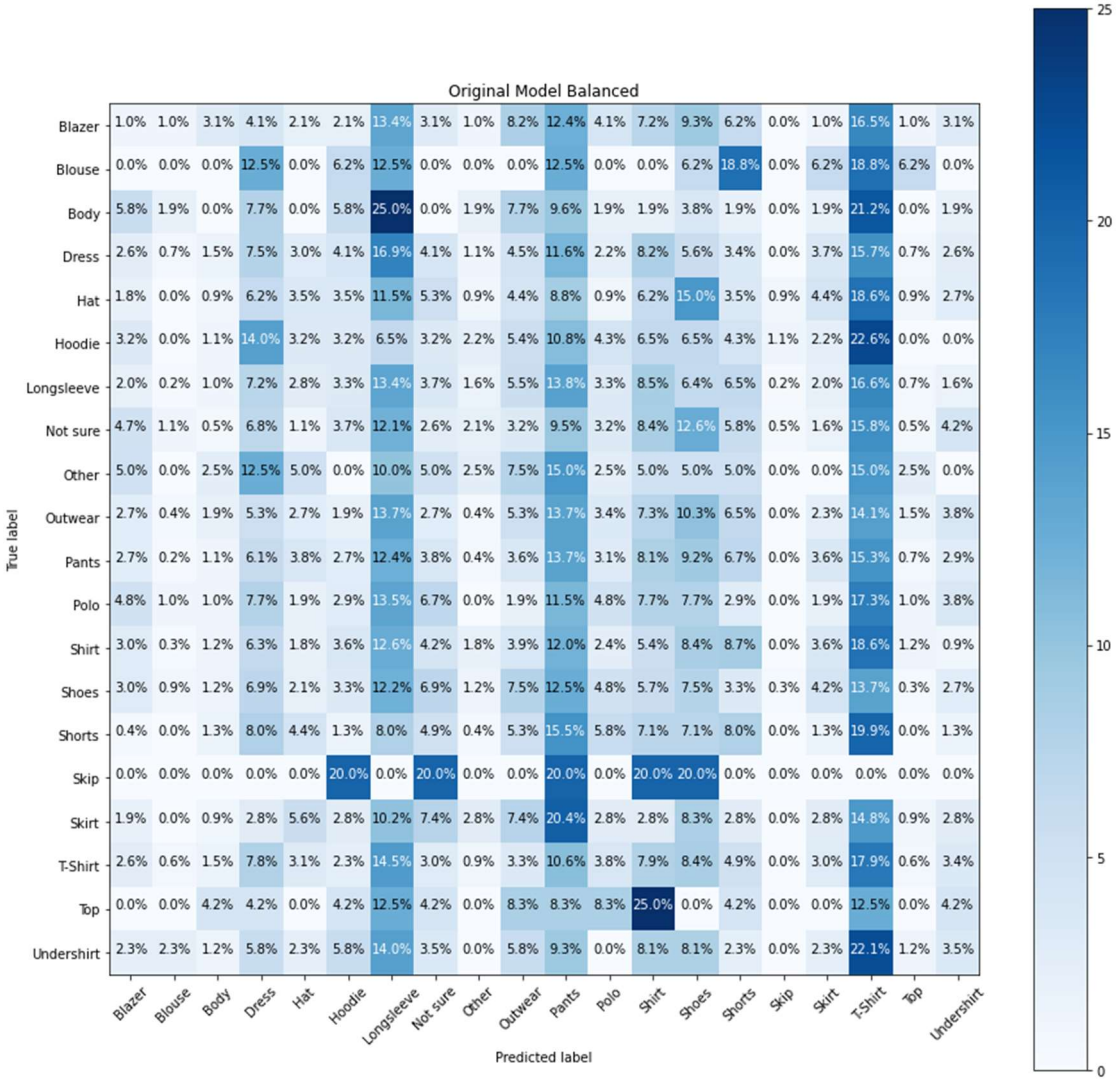
```
135/135 [==============================] - 22s 149ms/step - loss: 0.4169
- accuracy: 0.8875
```

```
[0.41690242290496826, 0.8874739408493042]
```

```
cm = confusion_matrix(y_true=train_set.labels, y_pred=rounded_prediction
s)
plot_confusion_matrix(cm=cm,normalize = True, classes=train_set.class_in
dices, title='Original Model Balanced')
```

```
Normalized confusion matrix
[[ 1.   1.   3.1  4.1  2.1  2.1 13.4  3.1  1.   8.2 12.4  4.1  7.2  9.3
   6.2  0.   1.  16.5  1.   3.1]
 [ 0.   0.   0.  12.5  0.   6.2 12.5  0.   0.   0.  12.5  0.   0.   6.2
  18.8  0.   6.2 18.8  6.2  0. ]
 [ 5.8  1.9  0.   7.7  0.   5.8 25.   0.   1.9  7.7  9.6  1.9  1.9  3.8
   1.9  0.   1.9 21.2  0.   1.9]
 [ 2.6  0.7  1.5  7.5  3.   4.1 16.9  4.1  1.1  4.5 11.6  2.2  8.2  5.6
   3.4  0.   3.7 15.7  0.7  2.6]
 [ 1.8  0.   0.9  6.2  3.5  3.5 11.5  5.3  0.9  4.4  8.8  0.9  6.2 15.
   3.5  0.9  4.4 18.6  0.9  2.7]
 [ 3.2  0.   1.1 14.   3.2  3.2  6.5  3.2  2.2  5.4 10.8  4.3  6.5  6.5
   4.3  1.1  2.2 22.6  0.   0. ]
 [ 2.   0.2  1.   7.2  2.8  3.3 13.4  3.7  1.6  5.5 13.8  3.3  8.5  6.4
```

```
  6.5  0.2  2.  16.6  0.7  1.6]
[ 4.7  1.1  0.5  6.8  1.1  3.7 12.1  2.6  2.1  3.2  9.5  3.2  8.4 12.6
  5.8  0.5  1.6 15.8  0.5  4.2]
[ 5.   0.   2.5 12.5  5.   0.  10.   5.   2.5  7.5 15.   2.5  5.   5.
  5.   0.   0.  15.   2.5  0. ]
[ 2.7  0.4  1.9  5.3  2.7  1.9 13.7  2.7  0.4  5.3 13.7  3.4  7.3 10.3
  6.5  0.   2.3 14.1  1.5  3.8]
[ 2.7  0.2  1.1  6.1  3.8  2.7 12.4  3.8  0.4  3.6 13.7  3.1  8.1  9.2
  6.7  0.   3.6 15.3  0.7  2.9]
[ 4.8  1.   1.   7.7  1.9  2.9 13.5  6.7  0.   1.9 11.5  4.8  7.7  7.7
  2.9  0.   1.9 17.3  1.   3.8]
[ 3.   0.3  1.2  6.3  1.8  3.6 12.6  4.2  1.8  3.9 12.   2.4  5.4  8.4
  8.7  0.   3.6 18.6  1.2  0.9]
[ 3.   0.9  1.2  6.9  2.1  3.3 12.2  6.9  1.2  7.5 12.5  4.8  5.7  7.5
  3.3  0.3  4.2 13.7  0.3  2.7]
[ 0.4  0.   1.3  8.   4.4  1.3  8.   4.9  0.4  5.3 15.5  5.8  7.1  7.1
  8.   0.   1.3 19.9  0.   1.3]
[ 0.   0.   0.   0.   0.  20.   0.  20.   0.   0.  20.   0.  20.  20.
  0.   0.   0.   0.   0.   0. ]
[ 1.9  0.   0.9  2.8  5.6  2.8 10.2  7.4  2.8  7.4 20.4  2.8  2.8  8.3
  2.8  0.   2.8 14.8  0.9  2.8]
[ 2.6  0.6  1.5  7.8  3.1  2.3 14.5  3.   0.9  3.3 10.6  3.8  7.9  8.4
  4.9  0.   3.  17.9  0.6  3.4]
[ 0.   0.   4.2  4.2  0.   4.2 12.5  4.2  0.   8.3  8.3  8.3 25.   0.
  4.2  0.   0.  12.5  0.   4.2]
[ 2.3  2.3  1.2  5.8  2.3  5.8 14.   3.5  0.   5.8  9.3  0.   8.1  8.1
  2.3  0.   2.3 22.1  1.2  3.5]]
```



Simplifying Dataset

Data cleaning

Since most of the data has been categorized, and we are mostly interested on some of the labels, and not all, to simplify and improve the training process, we are going to remove some of the categories. And transform them into a single type. We will transform those labels into, Top, Shorts, Pants, and remove some

The label Top will contain: T-Shirt

The label Bottom will contain: Pants

The other Shorts will contain: Shorts

The removed ones will be: Skirt Dress OutWear Hat Blazer Hoodie Skip Blouse Top Polo Undershirt LongSleeve Shirt

```
imagesPath = 'Data/images_compressed'
dataCSV = originalCSV.copy()
dataCSV['image'] = imagesPath + '/' + originalCSV['image'] +'.jpg' #We a
djust the data so that it is well defined
dataCSV.dropna()

dataCSV.loc[dataCSV['label'] == 'T-Shirt','label'] = 'Top'

dataCSV.loc[dataCSV['label'] == 'Pants','label'] = 'Bottom_Long'

dataCSV.loc[dataCSV['label'] == 'Shorts','label'] = 'Bottom_Short'

newColumns = ['Top','Bottom_Long', 'Bottom_Short']
finalData = dataCSV.loc[dataCSV["label"].str.contains("Top|Bottom_Long|B
ottom_Short")]
data_df = finalData[['image', 'label']]

typeOfData = data_df["label"].value_counts()
typeOfData
```

```
Top             1054
Bottom_Long      692
Bottom_Short     308
Name: label, dtype: int64
```

Now we can check our new data

```
typeOfData = data_df["label"].value_counts()
data_df.head()
```

```
                                                image          label
1  Data/images_compressed/ea7b6656-3f84-4eb3-9099...            Top
3  Data/images_compressed/ea2ffd4d-9b25-4ca8-9dc2...            Top
5  Data/images_compressed/5d3a1404-697f-479f-9090...   Bottom_Short
7  Data/images_compressed/4c8f245e-a039-46fd-a6b9...            Top
8  Data/images_compressed/c995c900-693d-4dd6-8995...    Bottom_Long
```

```
train_set, val_set, num_classes = createTrainSetAndValidationSet(data_df
)
```

```
Found 1644 validated image filenames belonging to 3 classes.
Found 410 validated image filenames belonging to 3 classes.
Total classes: 3
```

Unbalanced data

We will first try with an unbalanced weighting on the data

```
cp_path = "checkPoints/modelTraining_unbalanced/cp-{epoch:04d}.ckpt"
cp_callback_unbalanced= checkPointCreation(cp_path)

unbalancedModel = generateModel(num_classes)
unbalancedModel.compile(loss='categorical_crossentropy', optimizer='adam
',metrics=['accuracy'])

Model: "model_2"
_________________________________________________________________________
_________________________
 Layer (type)                Output Shape              Param #     Connect
ed to
=========================================================================
=========================
 input_3 (InputLayer)        [(None, 224, 224, 3  0           []
                             )]

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)  0           ['input
_3[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64  9472        ['conv1
_pad[0][0]']
                             )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1
_conv[0][0]']
                             )

 conv1_relu (Activation)     (None, 112, 112, 64  0           ['conv1
_bn[0][0]']
                             )

 pool1_pad (ZeroPadding2D)   (None, 114, 114, 64  0           ['conv1
_relu[0][0]']
                             )

 pool1_pool (MaxPooling2D)   (None, 56, 56, 64)   0           ['pool1
_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1
_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
```

```
_block1_1_relu[0][0']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_2_conv[0][0']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_2_bn[0][0']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1
_pool[0][0']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block1_2_relu[0][0']

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_0_conv[0][0']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_3_conv[0][0']
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_0_bn[0][0',
                                                                 'conv2
_block1_3_bn[0][0']

 conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block1_add[0][0']

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block1_out[0][0']

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_1_conv[0][0']
 ization)

 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_1_bn[0][0']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block2_1_relu[0][0']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_2_conv[0][0']
 ization)

 conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
```

```
_block2_2_bn[0][0]']
 n)

 conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_out[0][0]',
                                                                  'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block2_add[0][0]']

 conv2_block3_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_1_conv[0][0]']
 ization)


 conv2_block3_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_1_bn[0][0]']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block3_1_relu[0][0]']

 conv2_block3_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_2_conv[0][0]']
 ization)

 conv2_block3_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_2_bn[0][0]']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block3_2_relu[0][0]']

 conv2_block3_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block3_3_conv[0][0]']
 ization)

 conv2_block3_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block2_out[0][0]',
                                                                  'conv2
```

```
_block3_3_bn[0][0']

 conv2_block3_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block3_add[0][0']

 conv3_block1_1_conv (Conv2D)   (None, 28, 28, 128)  32896       ['conv2
_block3_out[0][0']

 conv3_block1_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_1_conv[0][0']
 ization)

 conv3_block1_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_1_bn[0][0']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block1_1_relu[0][0']

 conv3_block1_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_2_conv[0][0']
 ization)

 conv3_block1_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_2_bn[0][0']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 28, 28, 512)  131584      ['conv2
_block3_out[0][0']

 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block1_2_relu[0][0']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_0_conv[0][0']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_3_conv[0][0']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_0_bn[0][0',
                                                                 'conv3
_block1_3_bn[0][0']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block1_add[0][0']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block1_out[0][0']
```

```
 conv3_block2_1_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block2_2_bn[0][0]']
 n)

 conv3_block2_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 28, 28, 512)  2048        ['conv3
_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_out[0][0]',
                                                                  'conv3
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block3_1_conv[0][0]']
 ization)


 conv3_block3_1_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
```

```
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block2_out[0][0]',
                                                                 'conv3
_block3_3_bn[0][0]']

 conv3_block3_out (Activation)  (None, 28, 28, 512)  0          ['conv3
_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 28, 28, 128)  512       ['conv3
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 28, 28, 128)  512       ['conv3
_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block4_3_conv[0][0]']
 ization)
```

```
 conv3_block4_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block3_out[0][0]',
                                                                  'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 14, 14, 256)  131328      ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_2_bn[0][0]']
 n)

 conv4_block1_0_conv (Conv2D)   (None, 14, 14, 1024  525312      ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block1_2_relu[0][0]']
                                )

 conv4_block1_0_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']
```

```
 conv4_block1_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block1_out[0][0]']


 conv4_block2_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block2_2_relu[0][0]']
                                )

 conv4_block2_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )

 conv4_block2_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block1_out[0][0]',
                                )                                 'conv4
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_1_conv[0][0]']
 ization)
```

```
 conv4_block3_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block2_out[0][0]',
                                )                                 'conv4
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block3_add[0][0]']
                                )

 conv4_block4_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
```

```
_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block4_add[0][0]']
                                )


 conv4_block5_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_1_bn[0][0]']
 n)

 conv4_block5_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_2_conv[0][0]']
 ization)

 conv4_block5_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block5_3_conv[0][0]']
 ization)                       )
```

```
 conv4_block5_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_2_conv[0][0]']
 ization)

 conv4_block6_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_2_bn[0][0]']
 n)

 conv4_block6_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block6_3_conv[0][0]']
 ization)                       )

 conv4_block6_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block6_add[0][0]']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 7, 7, 512)    524800      ['conv4
_block6_out[0][0]']
```

```
 conv5_block1_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_1_bn[0][0]']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_2_conv[0][0]']
 ization)

 conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200     ['conv4
_block6_out[0][0]']

 conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block1_2_relu[0][0]']

 conv5_block1_0_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_0_conv[0][0]']
 ization)

 conv5_block1_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_3_conv[0][0]']
 ization)

 conv5_block1_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_0_bn[0][0]',

                                                                  'conv5
_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block1_add[0][0]']

 conv5_block2_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_1_bn[0][0]']
```

```
 n)

 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_2_bn[0][0]']
 n)

 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block2_2_relu[0][0]']

 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_out[0][0]',
                                                                  'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
```

```
_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block2_out[0][0]',
                                                                  'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block3_add[0][0]']

 flatten_2 (Flatten)            (None, 100352)       0           ['conv5
_block3_out[0][0]']

 dense_2 (Dense)                (None, 3)            301059      ['flatt
en_2[0][0]']

==================================================================================
==========================
Total params: 23,888,771
Trainable params: 23,835,651
Non-trainable params: 53,120
__________________________________________________________________________________
_________________________
Model: "model_2"
__________________________________________________________________________________
_________________________
 Layer (type)                   Output Shape         Param #     Connect
ed to
==================================================================================
==========================
 input_3 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input
_3[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1
_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1
_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1
_bn[0][0]']
                                )
```

```
 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1
_relu[0][0]']
                                )

 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1
_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1
_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1
_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_0_bn[0][0]',
                                                                  'conv2
_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2
```

```
_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_1_bn[0][0]']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block2_1_relu[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_2_conv[0][0]']
 ization)

 conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_2_bn[0][0]']
 n)

 conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_out[0][0]',
                                                                  'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block2_add[0][0]']

 conv2_block3_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_1_conv[0][0]']
 ization)

 conv2_block3_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_1_bn[0][0]']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
```

```
_block3_1_relu[0][0]']

 conv2_block3_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_2_conv[0][0]']
 ization)

 conv2_block3_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_2_bn[0][0]']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block3_2_relu[0][0]']

 conv2_block3_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block3_3_conv[0][0]']
 ization)

 conv2_block3_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block2_out[0][0]',
                                                                  'conv2
_block3_3_bn[0][0]']


 conv2_block3_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block3_add[0][0]']

 conv3_block1_1_conv (Conv2D)   (None, 28, 28, 128)  32896       ['conv2
_block3_out[0][0]']

 conv3_block1_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_2_conv[0][0]']
 ization)

 conv3_block1_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_2_bn[0][0]']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 28, 28, 512)  131584      ['conv2
_block3_out[0][0]']
```

```
 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block1_0_bn[0][0]',
                                                                 'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0          ['conv3
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block1_out[0][0]']

 conv3_block2_1_bn (BatchNormal  (None, 28, 28, 128)  512       ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 28, 28, 128)  0         ['conv3
_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 28, 28, 128)  512       ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 28, 28, 128)  0         ['conv3
_block2_2_bn[0][0]']
 n)

 conv3_block2_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block1_out[0][0]',
                                                                 'conv3
```

```
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_1_conv[0][0]']
 ization)

 conv3_block3_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block2_out[0][0]',
                                                                  'conv3
_block3_3_bn[0][0]']


 conv3_block3_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
```

```
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block4_3_conv[0][0]']
 ization)

 conv3_block4_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block3_out[0][0]',
                                                                 'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 28, 28, 512)  0          ['conv3
_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 14, 14, 256)  131328      ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_2_bn[0][0]']
 n)
```

```
 conv4_block1_0_conv (Conv2D)   (None, 14, 14, 1024  525312      ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block1_2_relu[0][0]']
                                )

 conv4_block1_0_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block1_out[0][0]']

 conv4_block2_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block2_2_relu[0][0]']
                                )
```

```
 conv4_block2_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )


 conv4_block2_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block1_out[0][0]',
                                )                                 'conv4
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block2_out[0][0]',
                                )                                 'conv4
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block3_add[0][0]']
```

```
                                )

 conv4_block4_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block4_add[0][0]']
                                )

 conv4_block5_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_1_bn[0][0]']
 n)
```

```
 conv4_block5_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_2_conv[0][0]']
 ization)

 conv4_block5_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block5_3_conv[0][0]']

 ization)                       )

 conv4_block5_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_2_conv[0][0]']
 ization)

 conv4_block6_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_2_bn[0][0]']
 n)
```

```
 conv4_block6_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block6_3_conv[0][0]']
 ization)                       )

 conv4_block6_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block6_add[0][0]']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 7, 7, 512)    524800      ['conv4
_block6_out[0][0]']

 conv5_block1_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_1_bn[0][0]']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_2_conv[0][0]']
 ization)

 conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200     ['conv4
_block6_out[0][0]']

 conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block1_2_relu[0][0]']

 conv5_block1_0_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_0_conv[0][0]']
 ization)

 conv5_block1_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
```

```
_block1_3_conv[0][0]']
 ization)

 conv5_block1_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_0_bn[0][0]',
                                                                  'conv5
_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block1_add[0][0]']

 conv5_block2_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_2_bn[0][0]']
 n)


 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block2_2_relu[0][0]']

 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_out[0][0]',
                                                                  'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block2_out[0][0]']
```

```
 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block2_out[0][0]',
                                                                  'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block3_add[0][0]']

 flatten_2 (Flatten)            (None, 100352)       0           ['conv5
_block3_out[0][0]']

 dense_2 (Dense)                (None, 3)            301059      ['flatt
en_2[0][0]']

==================================================================================
Total params: 23,888,771
Trainable params: 23,835,651
Non-trainable params: 53,120
__________________________________________________________________________________
currentEpoch, file = load_weights("checkPoints/modelTraining_unbalanced"
)
```

```
if(currentEpoch != 0):
    unbalancedModel.load_weights(file)
unbalancedModel.fit(train_set,epochs=EPOCHS, initial_epoch = currentEpoc
h, verbose=True,validation_data=val_set, callbacks=[cp_callback_unbalanc
ed])
```

```
Getting weights
not this
checkPoints/modelTraining_unbalanced\cp-0001.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0001.ckpt
checkPoints/modelTraining_unbalanced\cp-0002.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0002.ckpt
checkPoints/modelTraining_unbalanced\cp-0003.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0003.ckpt
checkPoints/modelTraining_unbalanced\cp-0004.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0004.ckpt
checkPoints/modelTraining_unbalanced\cp-0005.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0005.ckpt
checkPoints/modelTraining_unbalanced\cp-0006.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0006.ckpt
checkPoints/modelTraining_unbalanced\cp-0007.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0007.ckpt
checkPoints/modelTraining_unbalanced\cp-0008.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0008.ckpt
checkPoints/modelTraining_unbalanced\cp-0009.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0009.ckpt
checkPoints/modelTraining_unbalanced\cp-0010.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0010.ckpt
checkPoints/modelTraining_unbalanced\cp-0011.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0011.ckpt
checkPoints/modelTraining_unbalanced\cp-0012.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0012.ckpt
checkPoints/modelTraining_unbalanced\cp-0013.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0013.ckpt
checkPoints/modelTraining_unbalanced\cp-0014.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0014.ckpt
checkPoints/modelTraining_unbalanced\cp-0015.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0015.ckpt
checkPoints/modelTraining_unbalanced\cp-0016.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0016.ckpt
checkPoints/modelTraining_unbalanced\cp-0017.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0017.ckpt
checkPoints/modelTraining_unbalanced\cp-0018.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0018.ckpt
checkPoints/modelTraining_unbalanced\cp-0019.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0019.ckpt
checkPoints/modelTraining_unbalanced\cp-0020.ckpt.data-00000-of-00001
checkPoints/modelTraining_unbalanced\cp-0020.ckpt
Starting from epoch 20
```

```
<keras.callbacks.History at 0x2c610cbd1e0>
```

```
if(currentEpoch < EPOCHS):
    unbalancedModel.save('saved_model/pantsModel_unbalanced')
```

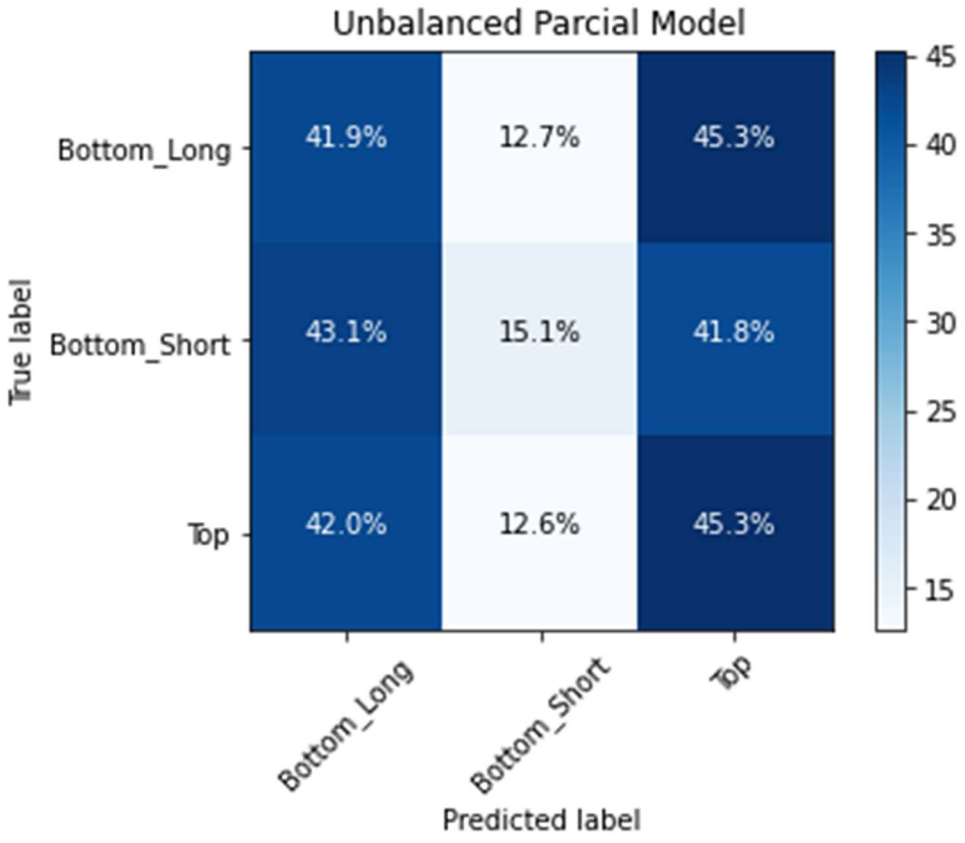
Visualizing

```
unbalancedModel = models.load_model('saved_model/pantsModel_unbalanced')
predictions = unbalancedModel.predict(train_set)
rounded_predictions = np.argmax(predictions, axis=-1)
unbalancedModel.evaluate(train_set)
```

```
52/52 [==============================] - 9s 147ms/step - loss: 15.1489 -
accuracy: 0.8893

[15.148859977722168, 0.8892943859100342]
```

```
cm = confusion_matrix(y_true=train_set.labels, y_pred=rounded_prediction
s)
plot_confusion_matrix(cm=cm, normalize = True,classes=train_set.class_in
dices, title='Unbalanced Parcial Model')
```

```
Normalized confusion matrix
[[41.9 12.7 45.3]
 [43.1 15.1 41.8]
 [42.  12.6 45.3]]
```



Conclusions

In this model we can see that Top is being classified accordingly, however some of those are set to bottom_long. The amount of bottom long classified as top and the other way around is too big to aknowladge as good results, altought the accuracy in theoru is high, up to 93% I dont understand why

Balanced Model

We will also train a blanaced Model, having some balanced weights to use

```
cp_path = "checkPoints/modelTraining_balanced/cp-{epoch:04d}.ckpt"
cp_callback_balanced= checkPointCreation(cp_path)
class_weight_dict = generateWeights(train_set.labels)
```

```
{0: 0.9699115044247788, 1: 2.3620689655172415, 2: 0.6469893742621016}
```

```
balancedModel = generateModel(num_classes)
balancedModel.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
Connected to
==================================================================
 input_4 (InputLayer)        [(None, 224, 224, 3)]     0         []

 conv1_pad (ZeroPadding2D)   (None, 230, 230, 3)       0         ['input_4[0][0]']

 conv1_conv (Conv2D)         (None, 112, 112, 64)      9472      ['conv1_pad[0][0]']

 conv1_bn (BatchNormalization)  (None, 112, 112, 64)   256       ['conv1_conv[0][0]']

 conv1_relu (Activation)     (None, 112, 112, 64)      0         ['conv1_bn[0][0]']

 pool1_pad (ZeroPadding2D)   (None, 114, 114, 64)      0         ['conv1_relu[0][0]']

 pool1_pool (MaxPooling2D)   (None, 56, 56, 64)        0         ['pool1_pad[0][0]']
```

```
 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1
_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1
_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_0_bn[0][0]',
                                                                  'conv2
_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_1_conv[0][0]']
 ization)
```

```
 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_1_bn[0][0]']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block2_1_relu[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block2_2_conv[0][0]']
 ization)

 conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block2_2_bn[0][0]']
 n)

 conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block1_out[0][0]',
                                                                  'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block2_add[0][0]']

 conv2_block3_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_1_conv[0][0]']
 ization)


 conv2_block3_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_1_bn[0][0]']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block3_1_relu[0][0]']

 conv2_block3_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_2_conv[0][0]']
 ization)

 conv2_block3_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
```

```
_block3_2_bn[0][0]']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block3_2_relu[0][0]']

 conv2_block3_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block3_3_conv[0][0]']
 ization)

 conv2_block3_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block2_out[0][0]',
                                                                  'conv2
_block3_3_bn[0][0]']

 conv2_block3_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block3_add[0][0]']

 conv3_block1_1_conv (Conv2D)   (None, 28, 28, 128)  32896       ['conv2
_block3_out[0][0]']

 conv3_block1_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_2_conv[0][0]']
 ization)

 conv3_block1_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_2_bn[0][0]']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 28, 28, 512)  131584      ['conv2
_block3_out[0][0]']

 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
```

```
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_0_bn[0][0]',
                                                                  'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block1_out[0][0]']

 conv3_block2_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_2_bn[0][0]']
 n)

 conv3_block2_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_out[0][0]',
                                                                  'conv3
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block2_out[0][0]']
```

```
 conv3_block3_1_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block3_1_conv[0][0]']
 ization)


 conv3_block3_1_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 28, 28, 512)  2048        ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block2_out[0][0]',
                                                                  'conv3
_block3_3_bn[0][0]']

 conv3_block3_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 28, 28, 128)  0           ['conv3
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 28, 28, 128)  512         ['conv3
_block4_2_conv[0][0]']
```

```
 ization)

 conv3_block4_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block4_3_conv[0][0]']
 ization)

 conv3_block4_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block3_out[0][0]',
                                                                 'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 28, 28, 512)  0          ['conv3
_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 14, 14, 256)  131328      ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_2_bn[0][0]']
 n)

 conv4_block1_0_conv (Conv2D)   (None, 14, 14, 1024  525312      ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block1_2_relu[0][0]']
                                )
```

```
 conv4_block1_0_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block1_out[0][0]']


 conv4_block2_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block2_2_relu[0][0]']
                                )

 conv4_block2_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )

 conv4_block2_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block1_out[0][0]',
                                )                                 'conv4
```

```
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block2_out[0][0]',
                                )                                 'conv4
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block3_add[0][0]']
                                )

 conv4_block4_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_1_conv[0][0]']
 ization)
```

```
 conv4_block4_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']

 conv4_block4_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block4_add[0][0]']
                                )


 conv4_block5_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_1_bn[0][0]']
 n)

 conv4_block5_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_2_conv[0][0]']
 ization)
```

```
 conv4_block5_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block5_3_conv[0][0]']
 ization)                       )

 conv4_block5_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)

 conv4_block6_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_2_conv[0][0]']
 ization)

 conv4_block6_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_2_bn[0][0]']
 n)

 conv4_block6_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block6_3_conv[0][0]']
 ization)                       )
```

```
 conv4_block6_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block6_add[0][0]']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 7, 7, 512)    524800      ['conv4
_block6_out[0][0]']

 conv5_block1_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_1_bn[0][0]']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_2_conv[0][0]']
 ization)

 conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200     ['conv4
_block6_out[0][0]']

 conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block1_2_relu[0][0]']

 conv5_block1_0_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_0_conv[0][0]']
 ization)

 conv5_block1_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_3_conv[0][0]']
 ization)

 conv5_block1_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_0_bn[0][0]',

                                                                  'conv5
_block1_3_bn[0][0]']
```

```
 conv5_block1_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block1_add[0][0]']

 conv5_block2_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_2_bn[0][0]']
 n)

 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block2_2_relu[0][0]']

 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_out[0][0]',
                                                                  'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_1_bn[0][0]']
 n)
```

```
 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block2_out[0][0]',
                                                                  'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block3_add[0][0]']

 flatten_3 (Flatten)            (None, 100352)       0           ['conv5
_block3_out[0][0]']

 dense_3 (Dense)                (None, 3)            301059      ['flatt
en_3[0][0]']

==================================================================
==========================
Total params: 23,888,771
Trainable params: 23,835,651
Non-trainable params: 53,120
__________________________________________________________________
_____________________________
Model: "model_3"
__________________________________________________________________
_____________________________
 Layer (type)                   Output Shape         Param #     Connect
ed to
==================================================================
==========================
 input_4 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input
```

```
_4[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1
_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1
_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1
_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1
_relu[0][0]']
                                )


 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1
_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1
_pool[0][0]']

 conv2_block1_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_1_conv[0][0]']
 ization)

 conv2_block1_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_1_bn[0][0]']
 n)

 conv2_block1_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block1_1_relu[0][0]']

 conv2_block1_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block1_2_conv[0][0]']
 ization)

 conv2_block1_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block1_2_bn[0][0]']
 n)

 conv2_block1_0_conv (Conv2D)   (None, 56, 56, 256)  16640       ['pool1
_pool[0][0]']

 conv2_block1_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block1_2_relu[0][0]']

 conv2_block1_0_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
```

```
_block1_0_conv[0][0]']
 ization)

 conv2_block1_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block1_3_conv[0][0]']
 ization)

 conv2_block1_add (Add)         (None, 56, 56, 256)  0          ['conv2
_block1_0_bn[0][0]',
                                                                 'conv2
_block1_3_bn[0][0]']

 conv2_block1_out (Activation)  (None, 56, 56, 256)  0          ['conv2
_block1_add[0][0]']

 conv2_block2_1_conv (Conv2D)   (None, 56, 56, 64)   16448      ['conv2
_block1_out[0][0]']

 conv2_block2_1_bn (BatchNormal  (None, 56, 56, 64)  256        ['conv2
_block2_1_conv[0][0]']
 ization)

 conv2_block2_1_relu (Activatio  (None, 56, 56, 64)  0          ['conv2
_block2_1_bn[0][0]']
 n)

 conv2_block2_2_conv (Conv2D)   (None, 56, 56, 64)   36928      ['conv2
_block2_1_relu[0][0]']

 conv2_block2_2_bn (BatchNormal  (None, 56, 56, 64)  256        ['conv2
_block2_2_conv[0][0]']
 ization)

 conv2_block2_2_relu (Activatio  (None, 56, 56, 64)  0          ['conv2
_block2_2_bn[0][0]']
 n)

 conv2_block2_3_conv (Conv2D)   (None, 56, 56, 256)  16640      ['conv2
_block2_2_relu[0][0]']

 conv2_block2_3_bn (BatchNormal  (None, 56, 56, 256)  1024      ['conv2
_block2_3_conv[0][0]']
 ization)

 conv2_block2_add (Add)         (None, 56, 56, 256)  0          ['conv2
_block1_out[0][0]',
                                                                 'conv2
_block2_3_bn[0][0]']

 conv2_block2_out (Activation)  (None, 56, 56, 256)  0          ['conv2
_block2_add[0][0]']
```

```
 conv2_block3_1_conv (Conv2D)   (None, 56, 56, 64)   16448       ['conv2
_block2_out[0][0]']

 conv2_block3_1_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_1_conv[0][0]']
 ization)

 conv2_block3_1_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_1_bn[0][0]']
 n)

 conv2_block3_2_conv (Conv2D)   (None, 56, 56, 64)   36928       ['conv2
_block3_1_relu[0][0]']

 conv2_block3_2_bn (BatchNormal  (None, 56, 56, 64)  256         ['conv2
_block3_2_conv[0][0]']
 ization)

 conv2_block3_2_relu (Activatio  (None, 56, 56, 64)  0           ['conv2
_block3_2_bn[0][0]']
 n)

 conv2_block3_3_conv (Conv2D)   (None, 56, 56, 256)  16640       ['conv2
_block3_2_relu[0][0]']

 conv2_block3_3_bn (BatchNormal  (None, 56, 56, 256)  1024       ['conv2
_block3_3_conv[0][0]']
 ization)

 conv2_block3_add (Add)         (None, 56, 56, 256)  0           ['conv2
_block2_out[0][0]',
                                                                  'conv2
_block3_3_bn[0][0]']


 conv2_block3_out (Activation)  (None, 56, 56, 256)  0           ['conv2
_block3_add[0][0]']

 conv3_block1_1_conv (Conv2D)   (None, 28, 28, 128)  32896       ['conv2
_block3_out[0][0]']

 conv3_block1_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_1_conv[0][0]']
 ization)

 conv3_block1_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_1_bn[0][0]']
 n)

 conv3_block1_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
```

```
_block1_1_relu[0][0]']

 conv3_block1_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block1_2_conv[0][0]']
 ization)

 conv3_block1_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block1_2_bn[0][0]']
 n)

 conv3_block1_0_conv (Conv2D)   (None, 28, 28, 512)  131584      ['conv2
_block3_out[0][0]']

 conv3_block1_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block1_2_relu[0][0]']

 conv3_block1_0_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_0_conv[0][0]']
 ization)

 conv3_block1_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block1_3_conv[0][0]']
 ization)

 conv3_block1_add (Add)         (None, 28, 28, 512)  0          ['conv3
_block1_0_bn[0][0]',
                                                                 'conv3
_block1_3_bn[0][0]']

 conv3_block1_out (Activation)  (None, 28, 28, 512)  0          ['conv3
_block1_add[0][0]']

 conv3_block2_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block1_out[0][0]']

 conv3_block2_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_1_conv[0][0]']
 ization)

 conv3_block2_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block2_1_bn[0][0]']
 n)

 conv3_block2_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block2_1_relu[0][0]']

 conv3_block2_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block2_2_conv[0][0]']
 ization)

 conv3_block2_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
```

```
_block2_2_bn[0][0]']
 n)

 conv3_block2_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block2_2_relu[0][0]']

 conv3_block2_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block2_3_conv[0][0]']
 ization)

 conv3_block2_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block1_out[0][0]',
                                                                  'conv3
_block2_3_bn[0][0]']

 conv3_block2_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block2_add[0][0]']

 conv3_block3_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block2_out[0][0]']

 conv3_block3_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_1_conv[0][0]']
 ization)

 conv3_block3_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_1_bn[0][0]']
 n)

 conv3_block3_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block3_1_relu[0][0]']

 conv3_block3_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block3_2_conv[0][0]']
 ization)

 conv3_block3_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block3_2_bn[0][0]']
 n)

 conv3_block3_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block3_2_relu[0][0]']

 conv3_block3_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block3_3_conv[0][0]']
 ization)

 conv3_block3_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block2_out[0][0]',
                                                                  'conv3
```

```
_block3_3_bn[0][0]']


 conv3_block3_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block3_add[0][0]']

 conv3_block4_1_conv (Conv2D)   (None, 28, 28, 128)  65664       ['conv3
_block3_out[0][0]']

 conv3_block4_1_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_1_conv[0][0]']
 ization)

 conv3_block4_1_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_1_bn[0][0]']
 n)

 conv3_block4_2_conv (Conv2D)   (None, 28, 28, 128)  147584      ['conv3
_block4_1_relu[0][0]']

 conv3_block4_2_bn (BatchNormal  (None, 28, 28, 128)  512        ['conv3
_block4_2_conv[0][0]']
 ization)

 conv3_block4_2_relu (Activatio  (None, 28, 28, 128)  0          ['conv3
_block4_2_bn[0][0]']
 n)

 conv3_block4_3_conv (Conv2D)   (None, 28, 28, 512)  66048       ['conv3
_block4_2_relu[0][0]']

 conv3_block4_3_bn (BatchNormal  (None, 28, 28, 512)  2048       ['conv3
_block4_3_conv[0][0]']
 ization)

 conv3_block4_add (Add)         (None, 28, 28, 512)  0           ['conv3
_block3_out[0][0]',
                                                                  'conv3
_block4_3_bn[0][0]']

 conv3_block4_out (Activation)  (None, 28, 28, 512)  0           ['conv3
_block4_add[0][0]']

 conv4_block1_1_conv (Conv2D)   (None, 14, 14, 256)  131328      ['conv3
_block4_out[0][0]']

 conv4_block1_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_1_conv[0][0]']
 ization)

 conv4_block1_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
```

```
_block1_1_bn[0][0]']
 n)

 conv4_block1_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block1_1_relu[0][0]']

 conv4_block1_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block1_2_conv[0][0]']
 ization)

 conv4_block1_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block1_2_bn[0][0]']
 n)

 conv4_block1_0_conv (Conv2D)   (None, 14, 14, 1024  525312      ['conv3
_block4_out[0][0]']
                                )

 conv4_block1_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block1_2_relu[0][0]']
                                )

 conv4_block1_0_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_0_conv[0][0]']
 ization)                       )

 conv4_block1_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block1_3_conv[0][0]']
 ization)                       )

 conv4_block1_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block1_0_bn[0][0]',
                                )                                 'conv4
_block1_3_bn[0][0]']

 conv4_block1_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block1_add[0][0]']
                                )

 conv4_block2_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block1_out[0][0]']

 conv4_block2_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_1_conv[0][0]']
 ization)

 conv4_block2_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_1_bn[0][0]']
 n)

 conv4_block2_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
```

```
_block2_1_relu[0][0]']

 conv4_block2_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block2_2_conv[0][0]']
 ization)

 conv4_block2_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block2_2_bn[0][0]']
 n)

 conv4_block2_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block2_2_relu[0][0]']
                                )

 conv4_block2_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block2_3_conv[0][0]']
 ization)                       )


 conv4_block2_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block1_out[0][0]',
                                )                                 'conv4
_block2_3_bn[0][0]']

 conv4_block2_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block2_add[0][0]']
                                )

 conv4_block3_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block2_out[0][0]']

 conv4_block3_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_1_conv[0][0]']
 ization)

 conv4_block3_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_1_bn[0][0]']
 n)

 conv4_block3_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block3_1_relu[0][0]']

 conv4_block3_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block3_2_conv[0][0]']
 ization)

 conv4_block3_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block3_2_bn[0][0]']
 n)

 conv4_block3_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
```

```
_block3_2_relu[0][0]']
                                )

 conv4_block3_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block3_3_conv[0][0]']
 ization)                       )

 conv4_block3_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block2_out[0][0]',
                                )                                 'conv4
_block3_3_bn[0][0]']

 conv4_block3_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block3_add[0][0]']
                                )

 conv4_block4_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block3_out[0][0]']

 conv4_block4_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_1_conv[0][0]']
 ization)

 conv4_block4_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_1_bn[0][0]']
 n)

 conv4_block4_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block4_1_relu[0][0]']

 conv4_block4_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block4_2_conv[0][0]']
 ization)

 conv4_block4_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block4_2_bn[0][0]']
 n)

 conv4_block4_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block4_2_relu[0][0]']
                                )

 conv4_block4_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block4_3_conv[0][0]']
 ization)                       )

 conv4_block4_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block3_out[0][0]',
                                )                                 'conv4
_block4_3_bn[0][0]']
```

```
 conv4_block4_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block4_add[0][0]']
                                )

 conv4_block5_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block4_out[0][0]']

 conv4_block5_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_1_conv[0][0]']
 ization)

 conv4_block5_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_1_bn[0][0]']
 n)

 conv4_block5_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block5_1_relu[0][0]']

 conv4_block5_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block5_2_conv[0][0]']
 ization)

 conv4_block5_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block5_2_bn[0][0]']
 n)

 conv4_block5_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block5_2_relu[0][0]']
                                )

 conv4_block5_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block5_3_conv[0][0]']

 ization)                       )

 conv4_block5_add (Add)         (None, 14, 14, 1024  0           ['conv4
_block4_out[0][0]',
                                )                                 'conv4
_block5_3_bn[0][0]']

 conv4_block5_out (Activation)  (None, 14, 14, 1024  0           ['conv4
_block5_add[0][0]']
                                )

 conv4_block6_1_conv (Conv2D)   (None, 14, 14, 256)  262400      ['conv4
_block5_out[0][0]']

 conv4_block6_1_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_1_conv[0][0]']
 ization)
```

```
 conv4_block6_1_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_1_bn[0][0]']
 n)

 conv4_block6_2_conv (Conv2D)   (None, 14, 14, 256)  590080      ['conv4
_block6_1_relu[0][0]']

 conv4_block6_2_bn (BatchNormal  (None, 14, 14, 256)  1024       ['conv4
_block6_2_conv[0][0]']
 ization)

 conv4_block6_2_relu (Activatio  (None, 14, 14, 256)  0          ['conv4
_block6_2_bn[0][0]']
 n)

 conv4_block6_3_conv (Conv2D)   (None, 14, 14, 1024  263168      ['conv4
_block6_2_relu[0][0]']
                                )

 conv4_block6_3_bn (BatchNormal  (None, 14, 14, 1024  4096       ['conv4
_block6_3_conv[0][0]']
 ization)                       )

 conv4_block6_add (Add)         (None, 14, 14, 1024  0          ['conv4
_block5_out[0][0]',
                                )                                 'conv4
_block6_3_bn[0][0]']

 conv4_block6_out (Activation)  (None, 14, 14, 1024  0          ['conv4
_block6_add[0][0]']
                                )

 conv5_block1_1_conv (Conv2D)   (None, 7, 7, 512)    524800      ['conv4
_block6_out[0][0]']

 conv5_block1_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_1_conv[0][0]']
 ization)

 conv5_block1_1_relu (Activatio  (None, 7, 7, 512)   0          ['conv5
_block1_1_bn[0][0]']
 n)

 conv5_block1_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block1_1_relu[0][0]']

 conv5_block1_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block1_2_conv[0][0]']
 ization)

 conv5_block1_2_relu (Activatio  (None, 7, 7, 512)   0          ['conv5
```

```
_block1_2_bn[0][0]']
 n)

 conv5_block1_0_conv (Conv2D)   (None, 7, 7, 2048)   2099200     ['conv4
_block6_out[0][0]']

 conv5_block1_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block1_2_relu[0][0]']

 conv5_block1_0_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_0_conv[0][0]']
 ization)

 conv5_block1_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block1_3_conv[0][0]']
 ization)

 conv5_block1_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_0_bn[0][0]',
                                                                  'conv5
_block1_3_bn[0][0]']

 conv5_block1_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block1_add[0][0]']

 conv5_block2_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block1_out[0][0]']

 conv5_block2_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_1_conv[0][0]']
 ization)

 conv5_block2_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_1_bn[0][0]']
 n)

 conv5_block2_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block2_1_relu[0][0]']

 conv5_block2_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block2_2_conv[0][0]']
 ization)

 conv5_block2_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block2_2_bn[0][0]']
 n)


 conv5_block2_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block2_2_relu[0][0]']
```

```
 conv5_block2_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block2_3_conv[0][0]']
 ization)

 conv5_block2_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block1_out[0][0]',
                                                                  'conv5
_block2_3_bn[0][0]']

 conv5_block2_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block2_add[0][0]']

 conv5_block3_1_conv (Conv2D)   (None, 7, 7, 512)    1049088     ['conv5
_block2_out[0][0]']

 conv5_block3_1_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_1_conv[0][0]']
 ization)

 conv5_block3_1_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_1_bn[0][0]']
 n)

 conv5_block3_2_conv (Conv2D)   (None, 7, 7, 512)    2359808     ['conv5
_block3_1_relu[0][0]']

 conv5_block3_2_bn (BatchNormal  (None, 7, 7, 512)   2048        ['conv5
_block3_2_conv[0][0]']
 ization)

 conv5_block3_2_relu (Activatio  (None, 7, 7, 512)   0           ['conv5
_block3_2_bn[0][0]']
 n)

 conv5_block3_3_conv (Conv2D)   (None, 7, 7, 2048)   1050624     ['conv5
_block3_2_relu[0][0]']

 conv5_block3_3_bn (BatchNormal  (None, 7, 7, 2048)  8192        ['conv5
_block3_3_conv[0][0]']
 ization)

 conv5_block3_add (Add)         (None, 7, 7, 2048)   0           ['conv5
_block2_out[0][0]',
                                                                  'conv5
_block3_3_bn[0][0]']

 conv5_block3_out (Activation)  (None, 7, 7, 2048)   0           ['conv5
_block3_add[0][0]']

 flatten_3 (Flatten)            (None, 100352)       0           ['conv5
_block3_out[0][0]']
```

```
 dense_3 (Dense)                (None, 3)            301059      ['flatt
en_3[0][0]']

==================================================================================
==========================
Total params: 23,888,771
Trainable params: 23,835,651
Non-trainable params: 53,120
__________________________________________________________________________________
_____________________________
```

```
currentEpoch, file = load_weights("checkPoints/modelTraining_balanced")
if(currentEpoch != 0):
    balancedModel.load_weights(file)
balancedModel.fit(train_set,epochs=EPOCHS,initial_epoch = currentEpoch,
verbose=True,validation_data=val_set, callbacks=[cp_callback_balanced],
class_weight = class_weight_dict)
```

```
Getting weights
not this
checkPoints/modelTraining_balanced\cp-0001.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0001.ckpt
checkPoints/modelTraining_balanced\cp-0002.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0002.ckpt
checkPoints/modelTraining_balanced\cp-0003.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0003.ckpt
checkPoints/modelTraining_balanced\cp-0004.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0004.ckpt
checkPoints/modelTraining_balanced\cp-0005.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0005.ckpt
checkPoints/modelTraining_balanced\cp-0006.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0006.ckpt
checkPoints/modelTraining_balanced\cp-0007.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0007.ckpt
checkPoints/modelTraining_balanced\cp-0008.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0008.ckpt
checkPoints/modelTraining_balanced\cp-0009.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0009.ckpt
checkPoints/modelTraining_balanced\cp-0010.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0010.ckpt
checkPoints/modelTraining_balanced\cp-0011.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0011.ckpt
checkPoints/modelTraining_balanced\cp-0012.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0012.ckpt
checkPoints/modelTraining_balanced\cp-0013.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0013.ckpt
checkPoints/modelTraining_balanced\cp-0014.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0014.ckpt
checkPoints/modelTraining_balanced\cp-0015.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0015.ckpt
```

```
checkPoints/modelTraining_balanced\cp-0016.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0016.ckpt
checkPoints/modelTraining_balanced\cp-0017.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0017.ckpt
checkPoints/modelTraining_balanced\cp-0018.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0018.ckpt
checkPoints/modelTraining_balanced\cp-0019.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0019.ckpt
checkPoints/modelTraining_balanced\cp-0020.ckpt.data-00000-of-00001
checkPoints/modelTraining_balanced\cp-0020.ckpt
Starting from epoch 20
```

```
<keras.callbacks.History at 0x2c6294b78b0>
```

```
if(currentEpoch < EPOCHS):
    balancedModel.save('saved_model/pantsModel_balanced')
```

Visualizing

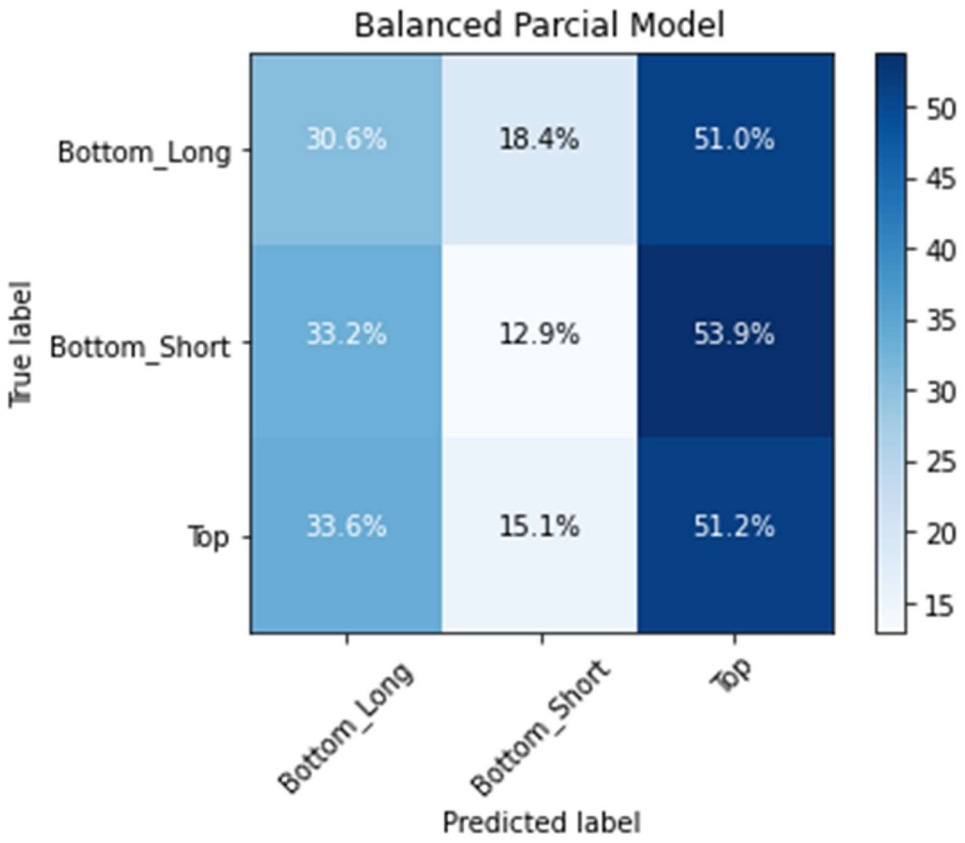
```
balancedModel = models.load_model('saved_model/pantsModel_balanced')
predictions = balancedModel.predict(train_set)
rounded_predictions = np.argmax(predictions, axis=-1)
balancedModel.evaluate(train_set)
```

```
52/52 [==============================] - 9s 142ms/step - loss: 0.6190 -
accuracy: 0.9130
```

```
[0.6190418601036072, 0.9130170345306396]
```

```
cm = confusion_matrix(y_true=train_set.labels, y_pred=rounded_prediction
s)
plot_confusion_matrix(cm=cm,normalize = True, classes=train_set.class_in
dices, title='Balanced Parcial Model')
```

```
Normalized confusion matrix
[[30.6 18.4 51. ]
 [33.2 12.9 53.9]
 [33.6 15.1 51.2]]
```



Conclusions

When using a balanced model, we can see that the accuracy goes lower, and the errors increase. However we can still see how there are almost no objects classified correctly as bottom_short, bottom_short

First conclusions

It is clear that by just using transfer learning as show before, the results are bad, therefore i shall find a better way. Possible options are using a differential learning rate.

Instead of unfreezing specific layers, it's probably a better idea to use a differential learning rate, where the learning rate is determined on a per-layer basis. The bottom layers will then have a very low learning rate, as these generalise quite well, responding principally to edges, blobs and other trivial geometries, whereas the layers responding to more complex features will have a larger learning rate. In the past, the 2:4:6 rule (negative powers of 10) has worked quite well for me — using a learning rate of 10^-6 for the bottommost few layers, 10^-4 for the other transfer layers and 10^-2 for any additional layers we added. I have also heard others using 2:3:4:5:6 or 2:3:4 with different architectures. For ResNets and their derivatives, I have always felt more comfortable with 2:4:6 than 2:3:4, but I have

absolutely no empirical evidence to back this up with. (https://medium.com/starschema-blog/transfer-learning-the-dos-and-donts-165729d66625)

Also, redoing it from scratch,

When transferring to a task with a relatively large data set, you might want to train the network from scratch (which would make it not transfer learning at all). At the same time — given that such a network would be initialised with random values, you have nothing to lose by using the pretrained weights! Unfreeze the entire network, remove the output layer and replace it with one matching the number of destination task classes, and fine-tune the whole network.

Same reference

Annex C: Funcions d'ajuda

C1. Visualitzador d'imatges

It holds methods to be used when trying to visualize an image, of diferent type

```
def showimage(myimage, figsize=[7,7], pltTitle = "", figax =[-1,-1]):
    if (myimage.ndim>2):  #This only applies to RGB or RGBA images (e.g.
not to Black and White images)
        myimage = myimage[:,:,::-1] #OpenCV follows BGR order, while mat
plotlib likely follows RGB order
    if(figax[0] == -1):
        fig, ax = plt.subplots(figsize=figsize)
    else:
        fig = figax[0]
        ax = figax[1]
    ax.imshow(myimage, cmap = 'gray', interpolation = 'bicubic')
    plt.xticks([]), plt.yticks([])  # to hide tick values on X and Y axi
s
    plt.title(pltTitle)
    plt.show()

#https://www.freedomvc.com/index.php/2021/10/16/gabor-filter-in-edge-det
ection/

def drawLines(image, arrayPoints, figsize=[7,7], pltTitle = ""):
    fig, ax = plt.subplots(figsize=figsize)
    for index in range(0, len(arrayPoints)):
        p1Index = index-1
        if(p1Index < 0):
            p1Index = len(arrayPoints)-1
        p1 = arrayPoints[p1Index]
        p2 = arrayPoints[index]
        plt.plot([p1[0], p2[0]], [p1[1], p2[1]])
    showimage(image, figax = [fig, ax], pltTitle = pltTitle)

def drawSingleLines(image, Lines, figsize=[7,7], pltTitle = ""):
    fig, ax = plt.subplots(figsize=figsize)

    for index in range(0, len(Lines)):
        p1 = Lines[index][0]
        p2 = Lines[index][1]
        plt.plot([p1[0], p2[0]], [p1[1], p2[1]])
    showimage(image, figax = [fig, ax], pltTitle = pltTitle)

def saveImage(image, baseRoot, folderName, iPath, fileName):
    name = 'P_'+str(iPath)+"_" + fileName
    if not os.path.exists(baseRoot):
        os.mkdir(baseRoot)
```

```
    finalDest = os.path.join(baseRoot,folderName)
    if not os.path.exists(finalDest):
        os.mkdir(finalDest)
    finalDest = os.path.join(finalDest,name)
    cv2.imwrite(finalDest, image)
    return finalDest

def generateCSV(rootPath, CSV):
    filesCSV = []
    for iPath, path in enumerate(labels):
        pathName = labels[iPath]
        folderPath = os.path.join(rootPath, path)
        for files in os.listdir(folderPath):
            finalPath = os.path.join(folderPath, files)
            filesCSV.append([finalPath,pathName])

    with open(CSV, 'w', newline='') as csvfile:
        csvwriter = csv.writer(csvfile)
        csvwriter.writerow(RowHeader)
        for row in filesCSV:
            csvwriter.writerow(row)
```

C2. Controlador de Llistes

Helpers to handle lists

```
def quickSort(arr, low, high):
    if len(arr) == 1:
        return arr
    if low < high:

        # pi is partitioning index, arr[p] is now
        # at right place
        pi = partition(arr, low, high)

        # Separately sort elements before
        # partition and after partition
        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)

def partition(arr, low, high):
    i = (low-1)         # index of smaller element
    pivot = arr[high]     # pivot

    for j in range(low, high):

        # If current element is smaller than or
        # equal to pivot
```

```
    if arr[j] >= pivot:

        # increment index of smaller element
        i = i+1
        arr[i], arr[j] = arr[j], arr[i]

arr[i+1], arr[high] = arr[high], arr[i+1]
return (i+1)
```