|  |
| --- |
| **사람의 인상착의를 통한 성별 추정**  **요 약**  최근 데이터가 중요시 되고 있는 시점에서 정보 수집이 중요한 문제로 떠오르고 있다. 따라서 object detection 모델중 YOLOv3 이용하여 사람의 인상창의를 파악하고 그 정보를 바탕으로 성별을 추정하고자 한다. 10개 카테고리 10만장의 데이터로 학습했으며 결과로 나오는 카테고리를 조합하여 성별을 판단한다. |

**1. 서론**

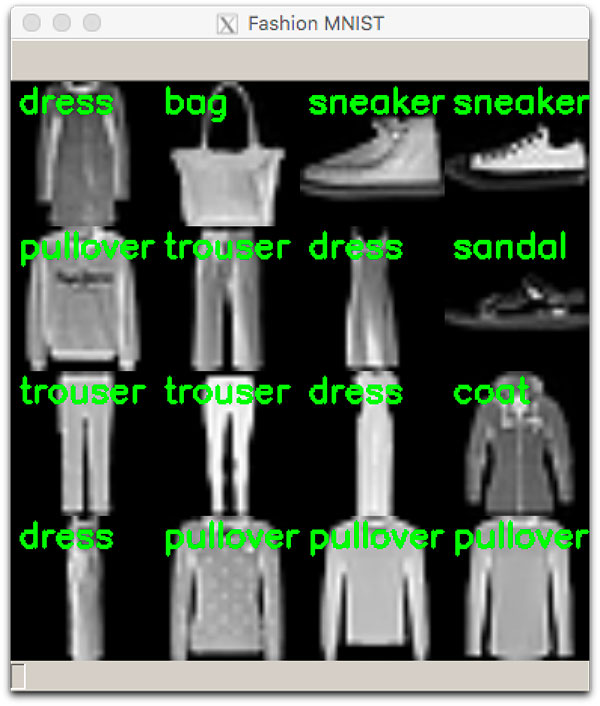
Demographic 데이터는 분석과 가공을 통해 마케팅, 건설, 부동산, 패션 등 사용될 수 있는 분야가 무궁무진하다. 따라서 행인들이 자주 다니는 길거리, 가게, 건물 등에 있는 CCTV나 안내 로봇에 설치되어있는 카메라를 통해 수집된 영상으로 행인들의 정보를 추정하고 데이터화한 데이터들의 수요가 존재할 것이다. 행인의 정보 추정에서 사용되는 지표 중 대표적인 것이 얼굴 인식을 통한 추정이다. 하지만 얼굴 인식을 통한 추정은 먼거리에 따른 정확도 저하, privacy에 따른 보안 문제 등으로 추정 방법으로의 사용이 제한적이다. 그러므로 얼굴 인식 외의 복장, 키, 체형 등을 인식하고 종합하여 행인의 정보를 추정하는 프로그램의 구현이 필요하다.

본 연구에서는 사람이 입고 있는 옷을 중점적으로 활용하여 행인의 정보를 추정해 나갈 것이다. 프로젝트 초기에는 selective search를 이용하여 object detection을 진행해 의류를 인식할 수 있는 인물의 사진을 input data로 받으면 인물의 의류 종류를 감지하고, 감지한 사진을 바탕으로 VGG16 모델을 사용해서 상의, 하의, 아우터를 포함한 50가지의 카테고리 그리고 28만개의 이미지 파일로 구성되어있는 패션 데이터셋인 DeepFashion의 데이터셋을 학습시켜 활용하여 구현하려였다. 하지만 프로젝트를 진행하며 selective search에는 한계가 존재해 이 모델로 학습을 시키면 제대로 의류의 종류를 감지해내지 못할 것이라 판단해 새로운 object detection 모델을 찾기로 결정하였고, 여러 object detection 모델을 비교한 후 속도와 성능에 모두 부합하는 YOLO모델을 선정해 사용하기로 결정하였고, 해당 모델로 프로젝트를 진행하였다.

**2. 관련연구**

**2.1. 의상 사진 통해 의상 종류 분류하기**

행인의 인상착의를 인식하고 분류하는 일 중 가장 첫번째로 할 수 있는 것은 복장 사진을 가지고 그 복장의 종류를 분류하는 작업이다. 이에 해당하는 작업은 텐서 플로우에서 예제로 제공해주고 있는데, 거기에서는 복장의 분류를 티셔츠와 바지부터, 아우터, 신발, 가방 등 10가지 종류로 분류하고 7만장의 데이터셋이 학습되어 있어 학습을 통한 분류를 진행한다. 복장의 종류 분류를 토대로 행인의 정보까지 추정하기 위해서는 10개의 종류 분류가 부족하기 때문에 여기서 옷을 조금 더 세분화하여 종류를 확장시키는 작업이 필요하고, 데이터셋 또한 복장 사진만 단독으로 나온 사진들을 데이터셋으로 사용했기 때문에 사람이 착용한 복장을 통해 옷의 종류를 분류할 수 있게 하는 작업이 필요하다. 그리고 인식률을 높이기 위해서 제공되는 데이터셋이 있다고 하더라도 해당 클래스의 의상이 제대로 나와있는 이미지 데이터인지 확인이 필요하다. 그리고 다양한 각도와 밝기의 데이터를 만들면 더욱 인식률이 높아진다.



**2.2. 사진을 통해 행인의 복장 분류하기**

앞서 나온 2.1에서의 모델은 복장이 단독으로 나온 사진을 통해 학습을 진행했기 때문에 복장을 인식하는 작업이 따로 필요하지가 않았다. 하지만 사진을 통해 행인의 복장을 각각 인식하고 분류하기 위해서는 Object detection 즉, 이미지 안에 있는 여러 레이블을 정확히 분류하는 동시에 레이블의 위치까지 파악하는 작업이 필요하다.

몇년 전 구글이 텐서플로우를 사용한 새로운 object detection API를 배포하였다. 그 중 tensorflow의 COCO 데이터셋에서 두번째로 느리지만 가장 정확한 모델인 Faster RCNN with Inception Resnet v2을 사용해 복장을 분류한 프로젝트가 존재한다. 이 프로젝트에서는 복장을 우선 하의 상의 전신 의상으로 크게 3가지로 구별하고, 하위 구별항목으로 50가지가 넘는 항목으로 분류를 한다. 프로젝트의 데이터셋은 홍콩 대학의 멀티미디어 연구실에서 모은 800,000장의 패션 사진을 사용해 학습시킨 결과를 바탕으로 의상을 분류한다.



이외에도 SSD(Single Shot MultiBox Detector)를 이용한 방법으로 사람이 입고 있는 의상을 탐지하는 프로젝트들도 다수 존재한다.

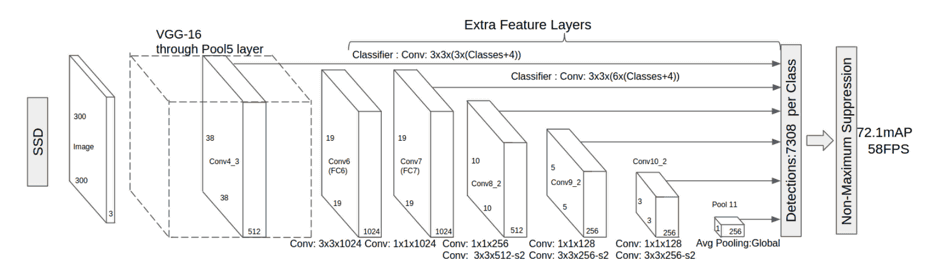


그림 1 SSD(Single Shot MultiBox Detector

**2.3. selective search**

selective search는 객체 인식이나 검출을 위한 가능한 후보 영역을 알아낼 수 있는 방법을 제공하는 것을 목표로 한다. selective search는 exhaustive search 방식과 segmentation 방식을 결합하여보다 뛰어난 후보 영역을 선정하는 것이다. Exhaustive search는 후보가 될만한 모든 영역을 샅샅이 조사하는 방식을 말하며 후보가 될만한 대상의 크기가 일정하지도 않고 가로, 세로 비율도 일정하지 않은 상황에서 모두 찾는 것이기 때문에 연산시간 관점에서는 수용이 불가하다.

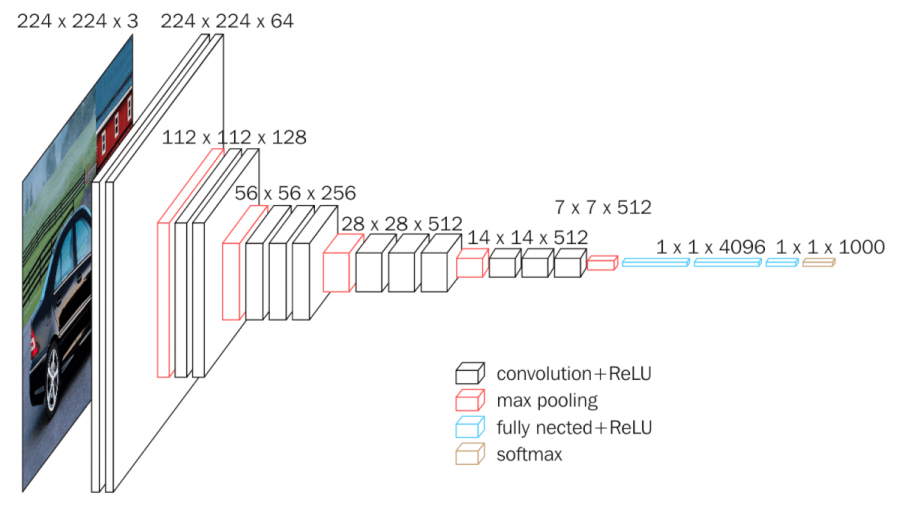
그래서 segmentation 방식을 이용하여 영상 데이터의 특성에 기반하는 방법을 결합한다. selective search 동작 방식은 일단 sub-segmentation을 수행한다. 각각의 객체가 1개의 영역에 할당이 될 수 있도록 많은 초기영역을 생성하고 작은 영역을 반복적으로 큰 영역으로 통합해 나가며 이때 알고리즘은 greedy 알고리즘을 사용한다. 우선 여러 영역으로부터 가장 비슷한 영역을 고르고, 이것들을 좀 더 큰 영역으로 통합하며 이 과정을 1개의 영역이 남을 때까지 반복한다. 마지막으로 통합된 영역들을 바탕으로 후보 영역을 만들어 낸다



**2.4. VGG16**

VGG 모델은 Visual Geometry Group의 약자로 머신 러닝 기반 컴퓨터 비전 모델의 시대를 열었던 AlexNet의 8-layers 모델보다 깊이가 2배 이상 깊은 네트워크의 학습에 성공했으며, 이를 통해 ImageNet Challenge에서 AlexNet의 오차율을 절반으로 줄였다. VGG모델이 16~19 레이어에 달하는 깊은 신경망을 학습할 수 있었던 것은 모든 합성곱 레이어에서 3x3 필터를 사용했기 때문이다. VGG 모델 이전에 Convolutional Network를 활용하여 이미지를 분류에서 좋은 성과를 보였던 모델들은 비교적 큰 Receptive Field를 갖는 11x11 필터나 7x7 필터를 포함한다.

하지만 VGG모델은 오직 3x3 크기의 작은 필터만 사용했음에도 이미지 부류 정확도를 비약적으로 개선시켰다. 3차례의 3x3 Convolution 필터링을 반복한 특징맵의 한 픽셀이 원본 이미지의 7x7 Receptive field의 효과를 볼 수 있다. 7x7 필터를 이용해 이미지에 대해 한번 Convolution을 수행한 것에 비해서 3x3 필터로 세 번 Convolution을 수행하면 비선형 함수가 적용되는 빈도가 높아지기 때문에 비선형성이 증가하여 특징 식별성이 증가된다



**2.4. darknet**

darknet은 YOLO의 개발자인 Joseph Redmon이 독자적으로 개발한 신경망 프레임워크로서 DNN(Deep neural network)들을 학습시키고 실행시킬 수 있는 framework이다. 그리고 yolo는 학습된 신경망 결과물 중 하나이다. darknet을 이용하면 yolo 뿐만 아니라 AlexNet, VGG-16, Resnet, Densenet 등 기존의 정통 주류의 DNN(Deep Neural Network)들도 돌려 볼 수 있다. Darknet과 yolo 코드는 모두에게 공개되어 있어 누구나 사용 가능하다. 여기서 DNN이란 ANN기법의 여러 문제가 해결되면서 모델 내 은닉층을 많이 늘려서 학습의 결과를 향상시키는 방법이 등장했고 이 방법을 지칭하는 말이다. DNN은 은닉층을 2개 이상 지닌 학습 방법을 뜻하며, 컴퓨터가 스스로 분류 레이블을 만들어 내고 공간을 왜곡하고 데이터를 구분 짓는 과정을 반복하여 최적의 구분선을 도출해낸다. 많은 데이터와 반복 학습, 사전 학습과 오류 역 전파 기법을 통해 현재 널리 사용되고 있습니다. 그리고 DNN을 응용한 알고리즘이 바로 CNN, RNN이다. ANN(Artificial Neural Network)는 딥러닝의 기초이며, 사람의 신경망 원리와 구조를 모방하여 만든 기계학습 알고리즘이다. 이것이 모든 딥러닝 모델들의 기반이라고 할 수 있다.

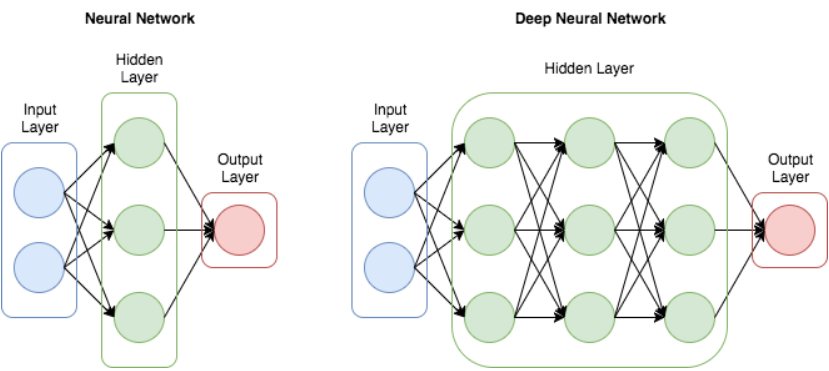


그림 2 DNN

**3. 프로젝트 내용**

**3.1. 중간과정**

**3.1.1 selective search + VGG16**

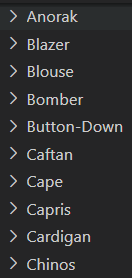
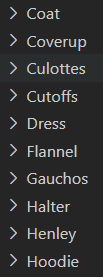
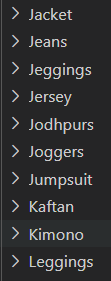
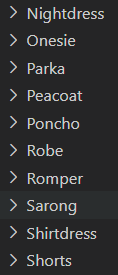
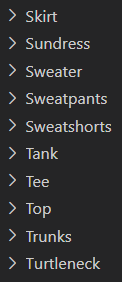
프로젝트 초기에는 selective search라는 알고리즘을 사용하여 object detection을 하고자 하였다. 여기서 selective search란 Bounding box들을 찾아주는 super pixel 기반으로 hierarchical grouping algorithm 방식을 사용한다. 가능한 모든 bounding box들을 찾고 box별로 유사하다고 판단되는 것들을 합쳐가면서 유사성이 없다고 판단되면 그 과정을 멈춘다. 유사성 판단은 color, texture,size, fill 4가지 요소들의 가중치 값을 계산하여 판단한다. 그렇게 해서 의상들을 넣고 테스트 해본 결과 필요로 하는 데이터 이외에 object들이 감지 되는 것을 알게 되었다 이러한 데이터들로는 저희가 원하는 결과를 내기 힘들겠다고 판단하여 다른 object detection & classification 모델을 찾고자 하였다.

**3.1.2 Dataset 구성**

Deepfashion의 이미지 데이터를 사용하였고 50개 카테고리 28만장의 데이터를 사용하였다. 이미지 데이터가 많은 만큼 GPU로 트레이닝을 시켰음에도 시간이 많이 걸렸다. 특히 selective search의 특성상 object로 인식되는 모든 곳을 box로 가져와 트레이닝 데이터셋으로 사용하기 때문에 전처리 과정에서 많은 시간이 소요되었다.

**3.1.3 Class 구성**

50개의 카테고리에는 Anorak, Blazer, Blouse, Dress, Coat, Hoodie, Jeans, Kimono, Skirt, Shorts 등등 옷의 여러 종류들이 있고 신발에 대한 데이터는 없었다. 각 카테고리당 약 5000 ~ 6000개의 데이터가 있다.

**3.1.4 한계점**

모델 자체의 한계점이 있었다. selective search는 우리가 필요로 하는 의상에 대한 인식 뿐만 아니라 너무 많은 수의 object들을 감지하였고 이는 threshold값을 변경해보아도 좋은 결과를 얻지 못하였다. 그리고 데이터셋의 카테고리에 맞지 않는 데이터들이 꽤 많았고 그 데이터들을 걸러내는 작업이 필요하였다. 아래 사진은 그 결과물이고 이런 이미지를 사용하기에는 프로젝트 진행 방향과 맞지 않는 것 같다는 판단을 하였다.

****

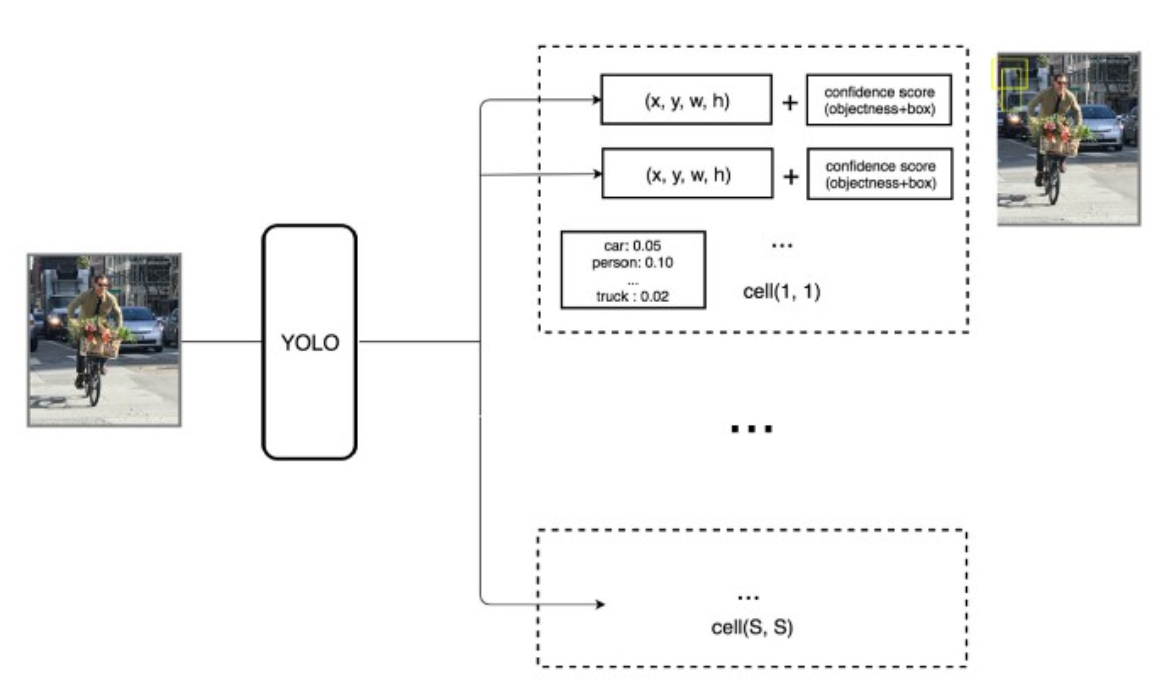
**3.1.5 진행 방향 재설정**

여러 시행착오 끝에 object detection & classification 모델을 바꿔야겠다는 결정을 하였다. 애초에 이번 프로젝트를 진행하면서 기초 지식이 거의 없는 상태인지라 구현되어 있는 코드들 중 구동 가능한 코드를 찾기에 바빴고 제대로 된 모델인지 판단을 할 수 없었다. 기초 지식 공부와 함께 object detection에 많은 방법들이 있고 그 중 우리가 사용할 딥러닝 모델은 yolov3로 정하였다. 실시간성으로 빠르게 데이터를 얻어야 되는 특성상 빠른 성능을 보이는 모델이 좋다고 판단했기 때문이다.

**3.2. 모델 선정**

**3.2.1 Object Detection Model(1) – YOLOv3**

Object detection 분야에서는 Faster\_RCNN, MobileNet, SSD 등의 모델이 있지만 이들보다 이름이 더 알려진 YOLOv3 모델을 사용하였다. YOLOv3은 YOLOv2에 비해 느리다는 단점이 있지만 성능면에서는 더 우수하다고 알려져있다.



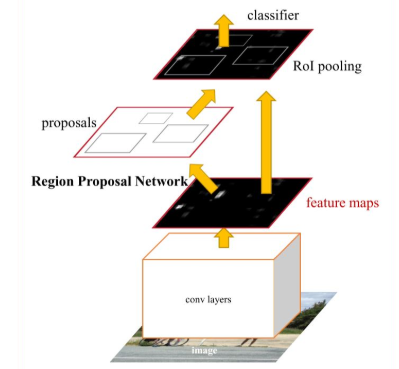
B개의 boundary boxes를 예측하고 각 box는 하나의 box confidence score를 가진다. 예측된 box 수에 관계없이 단 하나의 객체만 감지한다. 그 후 C개의 conditional class probabilities를 예측한다. 그래서 SxS Grid cells로 나누고 각 cell마다 하나의 객체를 예측한다. 그리고 미리 설정된 개수의 boundary boxes를 통해 객체의 위치와 크기를 파악한다. 이때 각 cell마다 하나의 객체만을 예측할수 있기 때문에 여러 객체가 겹쳐있으면 몇몇의 객체는 탐지를 못 하게 될 수 있다.

boundary box는 객체의 위치(x,y), 객체의 크기(w,h), box confidence score로 구성되어 총 5개의 인자를 가지고 있다. box confidence score는 box가 객체를 포함하고 있을 가능성과 boundary box가 얼마나 정확한지를 반영한다.

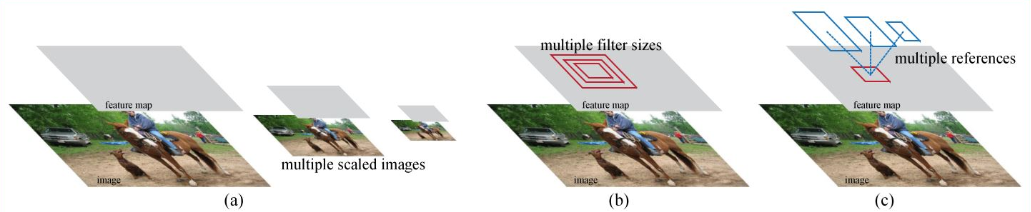


**3.2.2 Object Detection Model(2) – Faster\_RCNN**

Fast\_RCNN은 R-CNN의 복잡한 training, test pipeline을 통합함으로써 속도와 정확도면에서 눈의 띄는 성능 향상을 가져왔지만 Real-time object detector에 한 발짝 더 다가가기에는 여전히 속도면에서 아쉬운 부분이 있다. Fast\_RCNN에서 가장 큰 계산 부하를 차지하는 region proposal 생성을 새로운 방식으로 대체하고 이를 모델 내부로 통합시켰다.



Region Proposal Networks는 이미지를 입력 받아 사각형 형태의 Object Proposal과 Objectness Score를 출력해주는 역할을 한다. 이는 Fully convolutional network의 형태이며, Fast\_RCNN과 convolutional layers를 공유하게끔 디자인 되어있다., Region Proposal Networks에는 다음과 같은 특징을 가지는데, Anchor box는 sliding window의 각 위치에서 Bounding box의 후보로 사용되는 상자이다. 이는 기존에 두루 사용되던 Image/Feature pyramids와 Multiple-scaled sliding window와 차이를 보인다. 동일한 크기의 sliding window를 이동시키며 window의 위치를 중심으로 사전에 정의된 다양한 비율/크기의 anchor box들을 적용하여 feature를 추출하는 것이다. 이는 image/feature pyramids처럼 image 크기를 조정할 필요가 없으며, multiple-scaled sliding window처럼 filter 크기를 변경할 필요도 없으므로 계산 효율이 높은 방식이라 할 수 있다. 하지만 앞에서 말했던 것처럼 실시간성에서 속도 효율이 좋지 못하므로 현 프로젝트에 적용하기에는 제약 사항이 있다고 판단하였다.

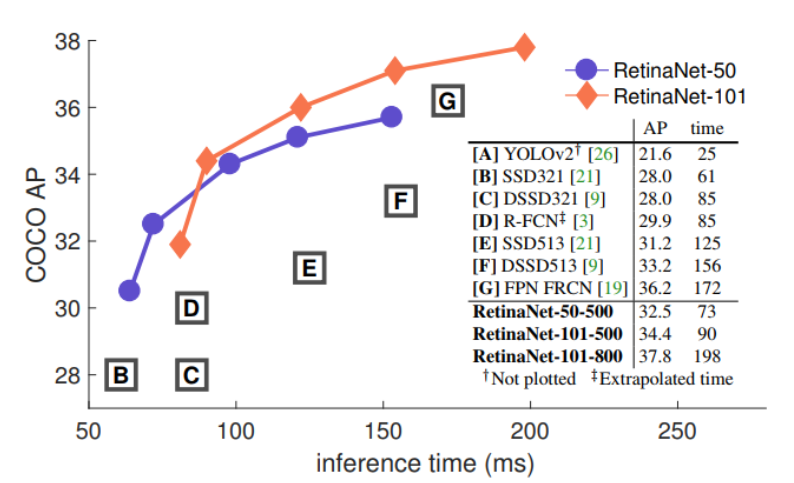


**3.2.3 Object Detection Model(3) – RetinaNet**

RetinaNet은 Focal Loss가 효과적임을 증명하기 위해 고안되었고 Feature Pyramid 구조를 사용해서 bounding box를 찾아낸다. 여러 층의 feature map에서 여러 anchor를 사용하는 방식이다. 네트워크 구조는 사진과 같다.

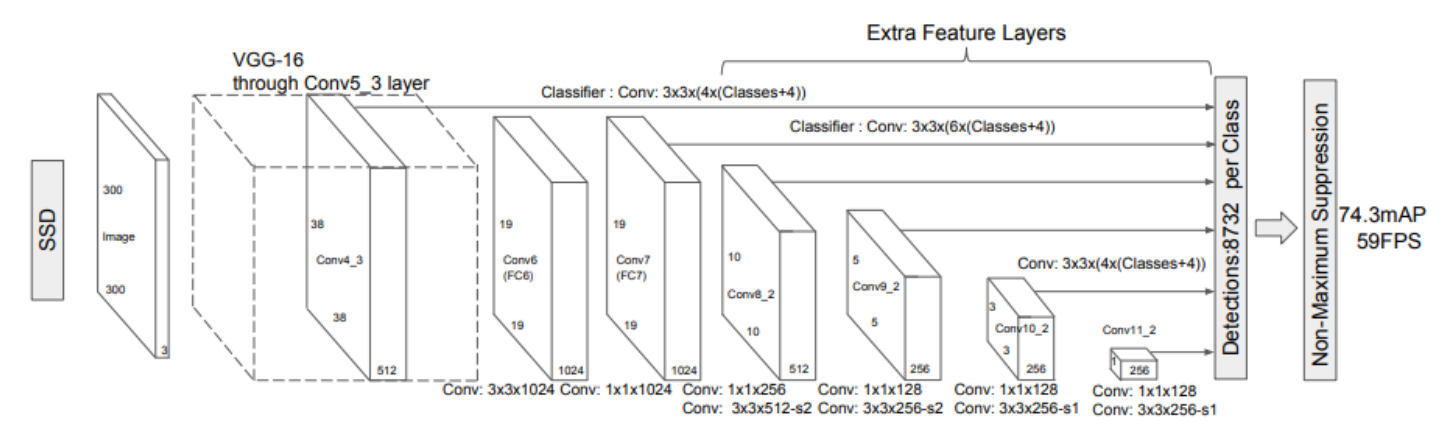


여기서 Focal Loss를 알기 위해서는 우선 Easy Negative sample이라는 것을 알아야 한다. Easy Negative sample이란 쉽게 특정 클래스가 아님을 분류할 수 있는 object를 말하는 것으로 R-CNN 계열의 모델들이 RPN에서 2000개에 달하는 RoI를 뽑아내 분류하는 과정에서 생성된다. Object Proposal들이 생성될 때, 배경 이미지를 포함하고 있거나 찾아내기 적당한 크기의 object들이 대부분의 bounding box 내에 들어 있게 된다. 하지만 object가 너무 크거나 작은 경우, 또는 잘려 있는 경우에는 그 물체의 bounding box와 클래스를 구하기 힘들다. 이렇게 쉬운 샘플과 어려운 샘플의 개수 비율이 너무 크게 차이나기 때문에 자연스럽게 Loss function이 쉬운 샘플들을 더 잘 찾게 된다. 하지만 성능을 판단하는 척도는 어려운 샘플을 얼마나 잘 찾느냐에 있기 때문에 쉬운 샘플들이 Loss에 미치는 영향을 약화시키고 어려운 샘플들이 Loss에 미치는 영향을 크게 하는 것이 핵심 아이디어이다.



**3.2.4 Object Detection Model(4) – SSD(Single Shot Multibox Detector)**

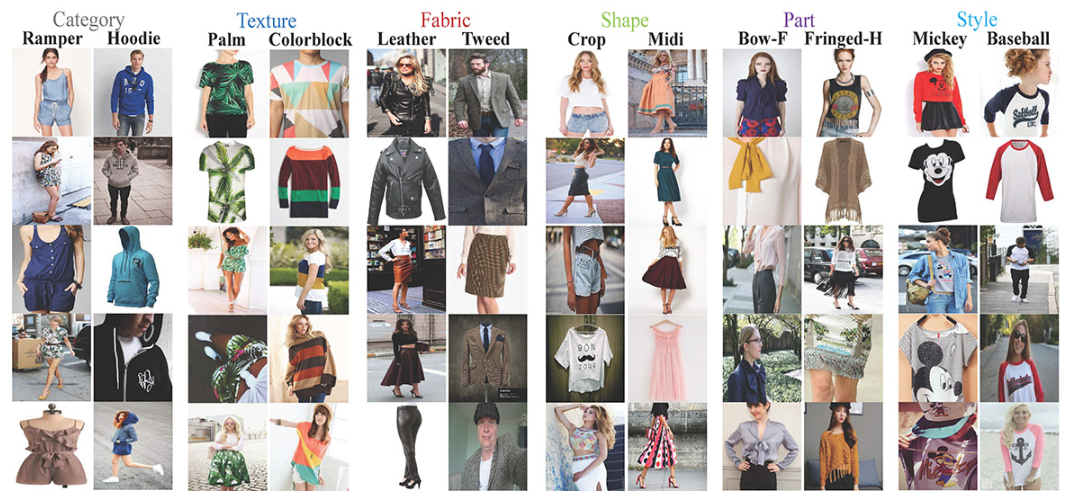
SSD의 알고리즘은 output을 만드는 공가능ㄹ 나누고 각 피쳐맵에서 다른 비율과 스케일로 default box를 생성하고 모델을 통해 계산된 좌표와 클래스 값에 default box를 활용해 최종 bounding box를 생성한다. yolo처럼 네트워크 하나만 사용하여 오브젝트의 경계박스를 찾고 클래스를 인식한다. VGG-16을 기본으로 사용하는데 약간 수정하여 이미지 특징 추출기로 사용한다. yolo는 최종 특징맵에만 경계 박스와 클래스 정보가 있는데 비해 SSD는 여러 히든 레이어에 정보가 분산되엉 있다. conv4\_3, conv7, conv8\_2, conv9\_2, conv10\_2, conv11\_2을 입력으로 convolution하여 생성된 6개의 특징맵 안에는 경계 박스와 클래스 정보가 담겨있다. yolo보다 빠르고 faster-RCNN보다 정확하다고 한다. 정확도를 높이기 위해서 여러 크기의 다른 feature map들로부터 여러 크기의 predict를 수행하고 비율 또한 다르게 적용한다. end-to-end 학습을 할 수 있게 구축했으며 저해상도 이미지에서도 높은 정확도를 가진다.



**3.2.5 비교 및 선정**

여러 detection 모델이 있지만 실시간성을 위해 감지 속도가 빠르게 성능이 보장되면서 detectino뿐만 아니라 classification까지 같이 해줄 수 있는 yolov3를 사용하는 것으로 결정하였다. yolov3에 대한 예제 코드나 설명 또한 인터넷 상에 많이 있는 만큼 다른 모델들에 비해 수월하게 진행 할 수 있을 것으로 예상했다.

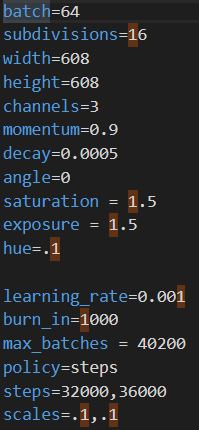
**3.3.1 Dataset, Class 구성**



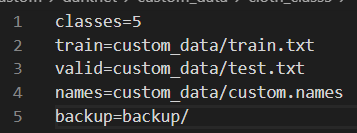
Deepfashion의 28만장의 데이터셋 중에서 초기에는 5개 카테고리 약 5만개의 이미지 데이터를 이용하여 트레이닝 시켰다. 청바지, 레깅스, 티셔츠, 스커트, 블레이져로 클래스를 나누어 진행하였고 특정 이미지에 대해서는 만족할만한 성과를 내어주었다. 그래서 카테고리를 10개, 약 10만개의 이미지 데이터를 사용하여 진행하였다. 드레스, 숏팬츠, 자켓, 가디건, 블라우스를 추가 하였고 카테고리가 많아질수록 성능은 이전보다 조금은 하락하였지만 성별 및 나이대를 추정하기 위해서는 더 많은 카테고리가 필요했기 때문에 그대로 진행하였다.

**3.3.2 5\_class**

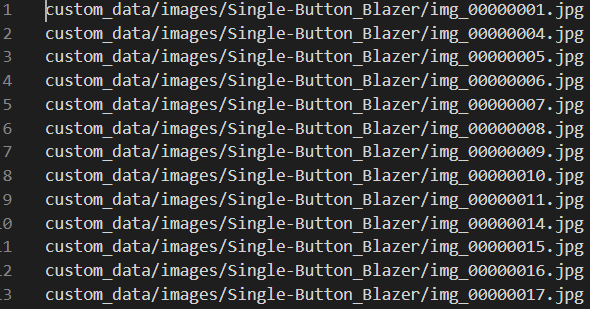
처음에는 청바지, 레깅스, 티셔츠, 스커트, 블레이저 5가지 카테고리 5만개의 이미지로 트레이닝을 하였다. max\_batchsize는 20200으로 하였고 filter는 30으로 설정하였다. 그리고 class값을 5로 바꿔주고 나머지 값들은 그대로 놔두고 진행하였다. 보통 CPU, GPU 성능이 안 좋을 때에는 batch와 subdivisions값을 1로 두고 한다고 하지만 성능이 좋은 GPU를 사용하였기 때문에 batch를 64, subdivisions를 16으로 두고 트레이닝 하였다. 테스트 때에는 가끔 에러가 나는 경우가 있었는데 batch와 subdivisions를 1로 해주면 그런 에러를 잡을 수 있다. height와 width는 608로 지정하였는데 트레이닝 속도는 줄어들 수 있지만 성능이 좋아지기 때문에 그대로 두고 진행하였다.



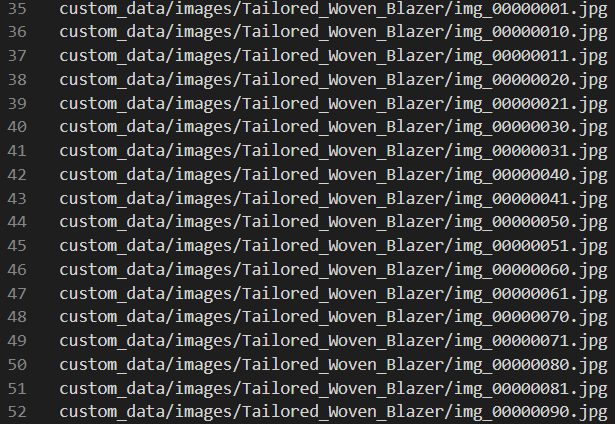
yolov3를 구동하기 위해서는 .cfg 설정파일 말고도 트레이닝 셋과 클래스 네임, weight 파일들이 저장될 위치를 정의해주는 설정 파일도 필요하다. 이 정보들은 .data 파일에 저장되어 있고 아래 사진과 같이 구성되어 있다.



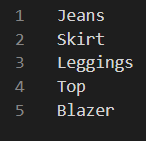
train.txt는 트레이닝 시킬 이미지의 상대주소를 가지고 있는 파일이다.



test.txt는 트레이닝 후 test를 위한 이미지 데이터 셋들의 상대주소를 가지고 있는 파일이다.

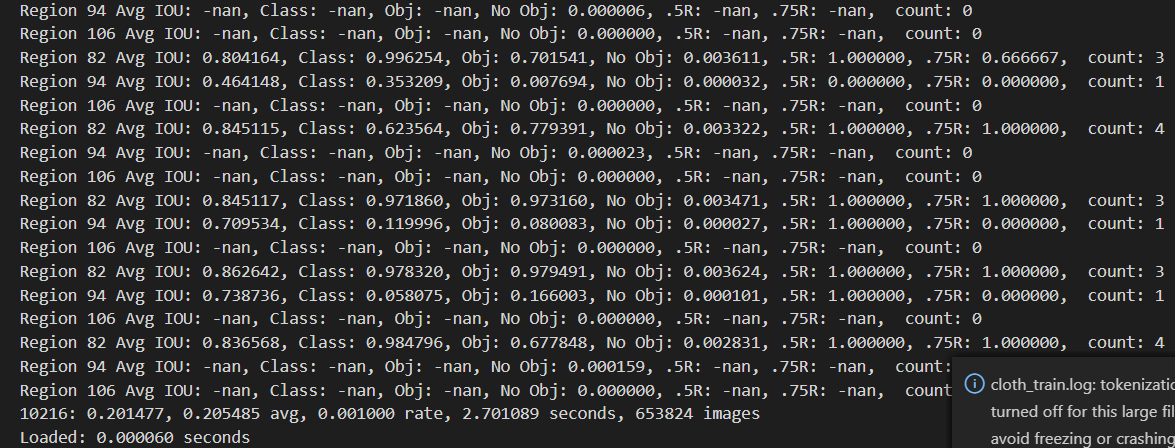


.names 파일은 클래스 종류를 가지고 있는 파일이다.

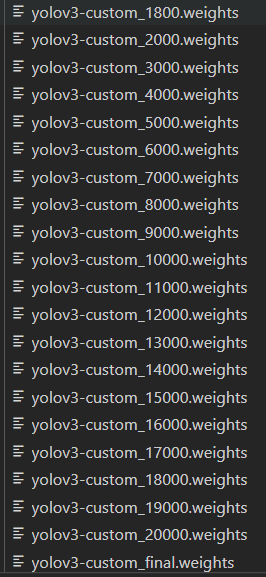


5개 카테고리이기 때문에 5개의 클래스를 가진다.

그리고 트레이닝을 시킨 결과를 .log 파일로 저장 해 두었다.

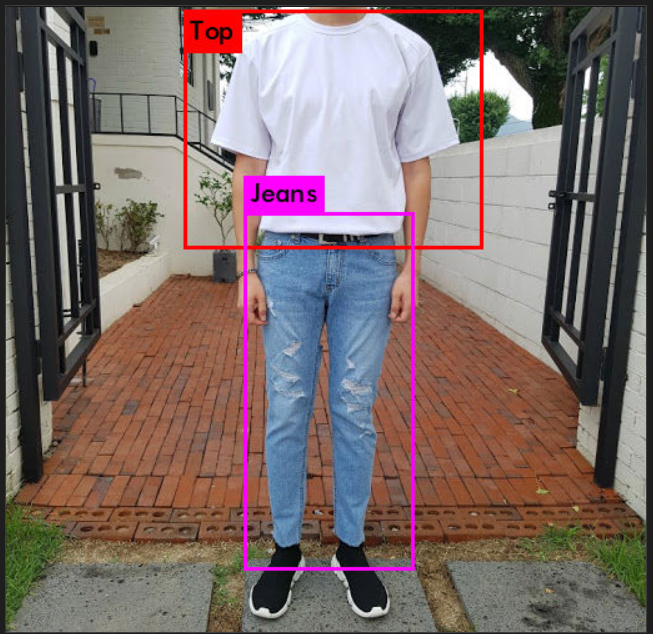


위 데이터들을 이용하여 트레이닝하여 1000batch가 돌 때마다 weight를 저장하였고 아래와 같이 저장되었다.



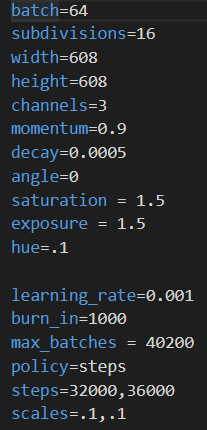
그리고 ./darknet detector test path/to/.data path/to/.cfg path/to/darknet53.conv.73 이라는 명령어를 사용하여 트레이닝을 시작한다. 여기서 darknet53.conv.73은 초기에 정해져있는 weight값이고 batch를 돌면서 설정해 두었던 횟수만큼 진행되면 .data파일에 정의된 backup의 위치에 weight파일들이 저장되게 된다. 트레이닝이 끝난 후 결과 값으로는 아래와 같다.





**3.3.3 10\_class**

5개의 클래스에서의 트레이닝은 나름 성공적인 결과를 얻었다. 그래서 5개의 카테고리로는 성별을 특정 짓기 힘들기 때문에 카테고리를 10개로 늘렸다. 클래스를 늘리게 되면 그만큼 데이터가 많이 필요하게 되고 그렇기 때문에 트레이닝 시간도 늘어나게 된다. 하지만 프로젝트를 올바른 방향으로 진행하기 위해서는 더 많은 클래스를 정의해야 하지만 우선을 10개의 클래스로 진행하였다. max\_batch size는 40200으로 하였고 filter는 (10class + 5) \* 3 = 45으로 설정하였다. class값은 10으로 바꿔주고 batch와 subdivisions는 64, 16으로 고정시키고 진행하였다. 5개의 클래스일 때와는 달리 테스트시 batch와 subdivisions 를 1,1로 해주지 않으면 아예 돌아가지 않았다. 성능을 위해 height, width는 608로 해주었다.



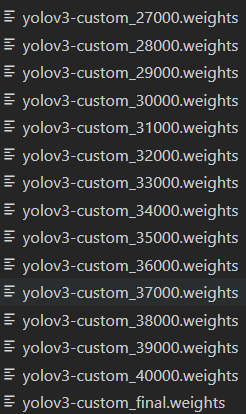
.cfg파일을 제외한 대부분의 파일들은 5개의 클래스를 진행할 때와 비슷하고 .names의 클래스 이름들만 바꾸어주면 되었다.

****

10개의 클래스를 트레이닝 시킨 결과 역시 .log 파일에 저장해두었다.

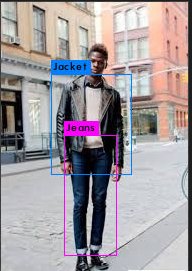
****

그 결과 나온 weight들은 다음과 같다.

****

max\_batch size가 40200이므로 40000번째 weight까지가 존재하게 되고 5개 클래스 때와 마찬가지로 ./darknet detector test path/to/.data path/to/.cfg path/to/darknet53.conv.73

명령어를 이용하여 트레이닝을 하였다. 그 결과 5개의 클래스 때보다 추가된 라벨을 가지는 결과가 나오게 되었다.

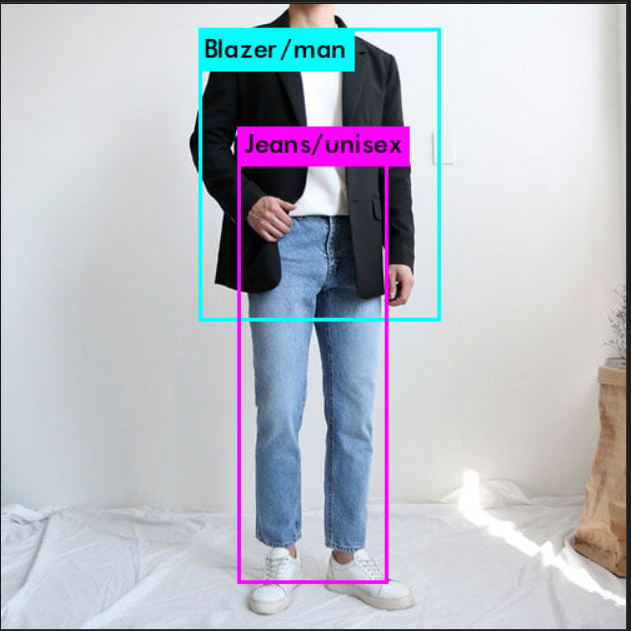
****

**3.3.4 gender classification**

이제 옷의 특징으로 성별을 분류하기 위해서는 다시 클래스를 정의해야 한다. 예를 들어 청바지 사진이 있으면 남자 청바지는 man\_jeans라고 한다면 여자 청바지는 woman\_jeans라고 라벨링을 하여 다시 트레이닝 시켜야 한다. 이렇게 여자, 남자 구분없이 입을 수 있는 옷들은 두 성별 모두 라벨링하고 원피스, 스커트 등 여자 옷들은 woman\_dress, woman\_skirt 등으로 라벨링 할 수 있다. 하지만 옷으로만 성별을 나눈 것은 정확하지 않은 기준일 수도 있다. 그래서 원래는 다른 얼굴과 같은 다른 특징들과 더불어서 특징을 나누는 기준으로 사용하면 성능을 더욱 높일 수 있다. 앞에 설명과 같이 트레이닝 시키면 어느정도 성능을 가지는 weight 값을 가질 수 있겠지만 현재 가지고 있는 데이터셋이 여자 비율이 아주 많은 상태이다. 이런 상태에서 트레이닝을 하게 되면 데이터 불균형으로 인해 원하는 성능을 기대하기는 힘들다. 그래서 이번 프로젝트에서는 옷을 classification을 한 결과의 조합으로 성별을 특정 짓기로 하였다.



위와 같이 분류해보았고 남녀 구분없이 입을 수 있는 옷은 unisex 남자가 주로 입는 옷은 man 여자가 주로 입는 옷은 woman으로 정의하였다. 그 결과 다음과 같은 이미지들을 얻게 되었다.



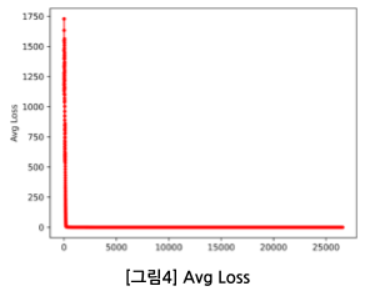
**4. 프로젝트 결과**

**4.1. 연구 결과**

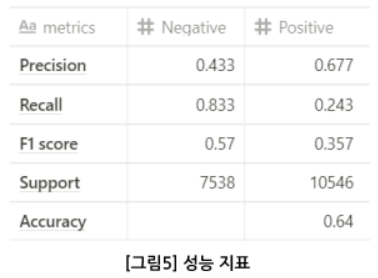
프로젝트를 통해서 input data로 복장을 입은 인물의 사진을 받으면 인물의 인상착의를 미리 설정해둔 10가지 카테고리 내에서 구별해서, 현재 입고 있는 옷의 종류를 추출한다. 그리고 옷의 종류를 추출한 결과값을 바탕으로 미리 카테고리별로 설정해둔 성별값을 통해 인물 사진에 나온 인물의 성별을 예측한다. 카테고리 별로 남성이 자주 입는 의류는 남성값을 여성이 많이 입는 의류는 여성값으로 설정하고, 남녀가 모두 즐겨입는 의류는 구별이 불가능하기 때문에 혼용이라는 항목으로 설정해 놓았다. 따라서 예측 결과값은 인물의 의류 종류에 따라 남성, 여성, 혼용으로 나타난다.

**4.2. 모델 성능평가**

max\_batch 사이즈를 50200으로 설정하여 트레이닝하는 동안 평균 loss값은 처음에는 조금 치솟았지만 이후 꾸준히 하락하여 0.2~0.37 사이를 유지 하였다. Avg IOU도 계속 상승하여 0.8 후반대를 유지하였다. 실제 테스트 결과 21000번째 weight가 잘 동작하는 것을 알아냈고 여러 이미지에 대하여 테스트 해보았다.

****

성능 평가를 위한 테스트 데이터를 넣은 결과를 분류 성능 평가 지표를 이용하여 평가하였다. precision, recall, F1 score, accuracy 등의 중요 요소들이 있는데 아래 표와 같은 결과 도출되었다. 높은 인식률을 보이지 못하기 때문에 더 정제된 데이터셋이 필요할 것으로 예상된다.



**4.3 한계점**

현재 사용한 Deepfashion 데이터셋은 임의의 사진 200장을 선정한 결과 약 10장의 사진을 제외한 모든 이미지가 여성의 사진으로 구성되어 있다. 원래 목표 중 하나였던 의상으로 성별 추정을 하기위해 총 10만장의 사진에 성별 label을 설정하여 모델을 학습시켜 성별을 추정하려 하였으나 편중된 데이터셋으로 인해 여자, 남자 클래스를 나누지는 못하였고, 다른 패션 데이터셋을 구해 남성의 사진을 보강하여 학습시키려 하였으나 새로운 데이터셋을 구해 새로 모델을 학습시키기에 프로젝트 마감 기간이 얼마 남지 않아 측정에 정확성을 보장해주지 않지만, 결과로 나오는 의상의 종류를 통해 성별을 판단하는 한계점을 가지고 프로젝트를 종료했다. 그리고 애초에 옷을 가지고 성별과 나이대를 구별하기는 매우 힘든 일이었다. 그래서 다른 특징들이 더 필요하고 예를 들면 얼굴, 헤어스타일, 체형 등이 있을 것이다. 이러한 특징들을 기준으로 하여 성별과 나이대를 분류한다면 더 좋은 결과를 얻을 수 있을 것이다.

**5. 결론**

**5.1. 기대효과**

인물의 특징을 추정하는 방법으로 실제값과 근사하게 예측할 수 있는 얼굴인식을 사용하지 못했을 때의 추정 방법을 프로젝트로 진행한 것이기 때문에 비록 측정값은 얼굴인식에 비해 낮을 수는 있으나, 본 프로젝트와 얼굴인식 방법을 모두 이용하면 측정값의 정확도는 더 오를 것이고, 얼굴인식을 사용하지 못하는 경우는 본 프로젝트의 추정값이 유일한 인물의 성별을 추정하는 결과값이 될 수 있기 때문에 프로젝트를 더 확장하면 인구통계학적 데이터를 효과적으로 수집 및 사용할 수 있을 것이다.

**5.2. 추후 연구 방향**

사람의 의상만으로는 성별과 나이대를 파악하는 것은 아주 어려운 과제이다. 요즘에는 남자, 여자 관계 없이 입는 옷들이 많고 특정 나이대로 대표되는 옷이 정해져 있지 않기 때문에 의상뿐만이 아닌 헤어 스타일, 얼굴 등의 특징들을 같이 이용하여 성별과 나이대를 추정하는 것이 더 정확한 방법이 될 것이다. 그래서 현재에는 어떤 의상을 입었는지 뿐만 아니라 헤어 스타일 등 다른 특징들을 파악하여 사람의 성별과 나이대를 파악하면 더 좋은 성능을 낼 수 있을 것이다.

**6. 참고문헌**

[1] 분류성능평가지표(<https://sumniya.tistory.com/m/26>)

[2] yolov3(<https://www.learnopencv.com/training-yolov3-deep-learning-based-custom-object-detector/>)

[3] yolov3(<https://wdprogrammer.tistory.com/50>)

[4] selective search(<https://donghwa-kim.github.io/SelectiveSearch.html>)

[5] VGG구조(<https://bskyvision.com/504>)

[6] Faster R-CNN(<https://curt-park.github.io/2017-03-17/faster-rcnn/>)

[7] Single Shot MultiBox Detector(<https://m.blog.naver.com/sogangori/221007697796>)

[8] RetinaNet(<https://github.com/MagmaTart/Paper-Reading/blob/master/summarys/Focal-Loss-for-Dense-Object-Detection.md>)

[9] Single Shot MultiBox Detector2(<https://cocopambag.tistory.com/15>)