CNN 피부암 예측모델

소속 동아대 경영정보학과

작 성 자 1718285 이해강
1951959 김정수

주제 CNN을 활용한 피부암 예측모델 만들기

목 차

I. 서론

1. 연구필요성 및 목적

피부암은 전 세계적으로 흔한 악성 종양으로 잦은 야외활동으로 자외선 노출이 많을 경우 발생한다 건강보험심사평가원에 따르면, 우리나라 2020년 피부암 발생환자 수는 27,211명으로 2016년 19,236에 비해 5년 동안 41,5% 증가, 전체 암 발생의 0.5%를 차지하였고 인구 10만 명당 조발생률은 2.6명으로 보고되었다.(보건복지부 중암암등록본부 2012년 12월 발표 자료)

미국 피부과학 학회(American Academy of Dermatology)는 미국에서 신규 피부암 진단 건수를 540만 건을 초과했다 밝혔고 뉴질랜드의 경우 더마플러스의 피부암 보고서(Skin Cancer Index Reprot 2018)에서 매년 2500명 이상이 피부암 진단을 받고 2017년 480명의 사망자가 발생해 피부암으로 인한 사망률이 2020년 기준 세계에서 가장 높았던 것으로 나타났다 뉴질랜드 헤럴드 유럽과 호주 연구원들이 공동으로 연구 조사한 통계 자료에 따르면 뉴질랜드에 이어 호주, 서유럽, 북미, 북유럽 순으로 많은 피부암 사망자를 기록했다.

이처럼 야외활동이 많은 요즘 꾸준한 증가세를 보이는 피부암은 우리나라에선 대부분 고령층이 가장 많이 발생하고 있다.

피부암 환자 대부분이 고령층인 점, 거동이 불편한 점, 초기에 증상이 거의 없기 때문에 예방하기 어려운 점을 고려해 피부암 발생위험에 대한 선별적 검사를 시행하여 고위험군을 선별하고 본인의 병명을 조기에 알 수 있게 알려줌으로써 피부암의 초기 예방, 피부암으로 인한 사망을 줄이는 것이 중요하다.

본 연구의 목적은 딥러닝머신을 활용한 피부암 예측모델 개발이다. 실험데이터와 학습시킨 데이터와 비교하는 검증 작업을 실시하여 모델의 정확성을 확인하고자 한다,
추후 간단한 사진 입력만으로 환자가 어떤 종류의 피부암인지 확인할 수 있게끔 하여 환자의 피부암 조기 대처 및 예방, 딥러닝 기초자료 활용에 도움 되고자 본 연구를 실시하였다.

2. 연구내용 및 범위

피부암 예측모델은 합성곱 신경망(콘볼루션 신경망, Convolutional neural network, CNN)으로 층을 쌓은 후 데이터 모델을 토대로 학습을 시켰고 학습 정도를 확인하였다.

Kaggle 사이트의 Skin Cancer MNIST: HAM 10000에서 7종류의 피부암 사진 총 10015를 수집, 수집한 데이터에 학습을 원활하게 하기 위해 각 class에 코드를 부여하였다.(akiec:0, bcc:1, bkl:2, df:3, mel:4, nv:5, vasc:6) 수집 데이터에서 0.1은 테스트 데이터로 그 후 0.1은 검증 데이터로 나누어 훈련 데이터는 8111개 테스트 데이터는 1002 검증 데이터는 902로 나누어 모델층을 쌓고 8111개의 이미지 데이터 학습을 150번 가량 실시해 측정하였다.

학습 정도는 그래프화 시켜서 과적합(overfitting) 여부를 확인하였으며 부족한 피부암 이미지는 전이학습과 임의증식을 활용하여 비슷한 이미지를 찾아내 100번의 모델링을 학습시킨 후 결과를 확인하였다. 전이학습은 적은 학습 시간이 적으며 정확도가 높은 VGG16 모델을 사용하였고 임의증식은 이미지 데이터를 활용한 코딩작업을 실시하였다. 예측이 끝난 데이터는 혼동 행렬(Confusion matrix)을 통해 정확도를 측정였다.

II. 피부암 종류 및 현황

1. 피부암 종류

피부암은 주로 지나친 햇빛의 노출이 주된 원인으로 알려져 있으며, 편평상피세포암, 악성 흑색종, 기저세포암 등을 통틀어 피부암이라 부른다.

그림1. 편평상피세포암

편평상피세포암은 표피의 각질 형성 세포에서 유래한 악성 종양으로 우리나라의 경우 햇빛에 노출되는 부위인 얼굴에 과반수 발생하고 특히 입술과 뺨 등에 잘 생기며 반흔도 중요한 유발인자로 알려져있다. 우리나라에서 기저세포암과 함께 가장 많은 피부암의 하나이다. 피부 사마귀나 자궁경부암을 일으키는 사람유두종 바이러스가 이 암과 관련이 있다 외형상 비교적 크고 불균일한 모양의 붉은 피부가 부어올라 살덩어리가 부서진 것처럼 보이며, 만졌을 때 덩어리가 있는 경우 의심해 볼 수 있다.

그림2. 악성흑생종

흑색종은 다른 피부암과 달리 굉장히 공격적이다 증식이 빠르고, 림프관을 따라 폐,뼈,간 등 다른 장기로 쉽게 전이되기 때문이다. 멜라닌을 만드는 세포인 멜라노사이트가 악성화된 종양으로 일반적으로 정상적인 피부에서 검은빛을 띤 색소성 반점이 새로 작게 성장하여 시작되며 신체의 특정 부위에서 주로 나타난다. 유전적 요인, 또는 자외선 등의 환경 요인에 의해서 발생하며 백인에게 많이 발생하는 '표재 확대형' 검은색 또는 진하거나 옅은 검은색이 섞인 딱딱한 응어리가 온몸의 모든 부위에 발생하는 것이 특징인 '결절형', 얼굴과 목, 손등 등 햇빛에 노출되는 부위에 갈색 또는 흑살

색의 멍이 생기는 '악성 흑자형'의 4종류로 크게 분류된다.

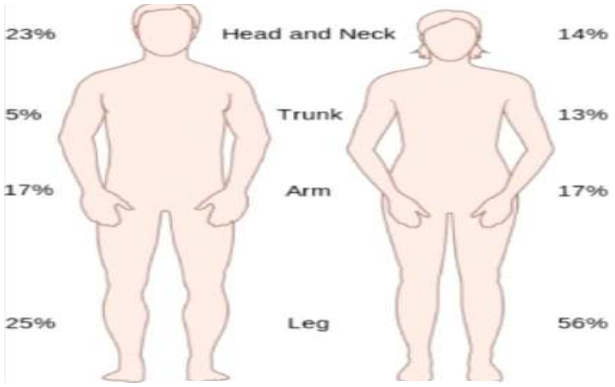

그림3. 흑색종 발생 부위 비율

남자의 경우 두경부 쪽에 23%, 하지에 25%, 팔 17%, 몸통에 5% 가량 나타난다. 여성의 경우 두경부 14%, 하지에 56% 정도 발생한다(그림3) 평범한 검은 반점으로 보여 간과하기 쉬우나, 반점의 모양이 대칭적이지 않거나, 반점의 경계가 불규칙하거나, 색깔이 변하거나 다양하거나, 크기가 클 때 의심해 볼 수 있다.

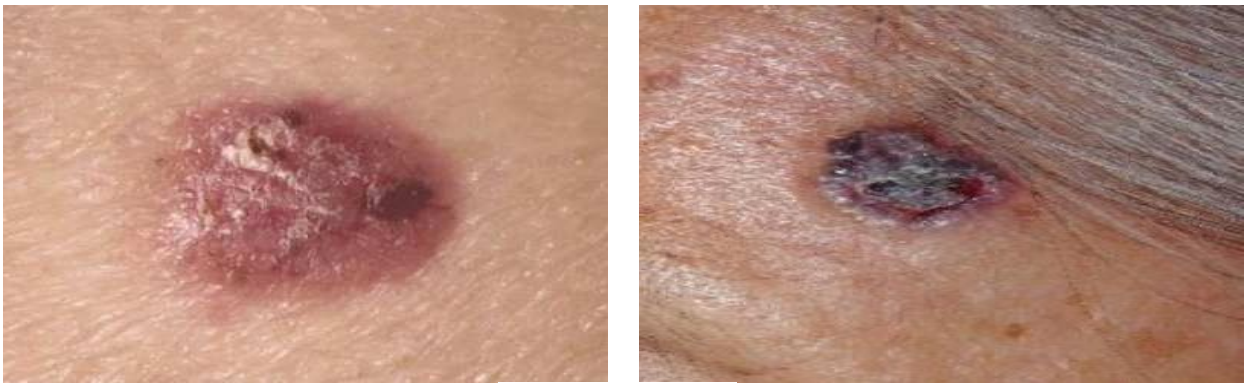
그림4. 기저세포암

기저세포암종은 가장 일반적인 유형의 피부암이다 미국의 경우 매년 4백만 명 이상의 사람에게서 발병하는데 자외선 노출 이력이 많으면서 피부색이 흰 사람에게 자주 발병한다 신체의 다른 부위로 확산되는 경우가 거의 없지만 조직에 발병하거나 드물게 눈, 귀, 뼈 등에 자라나는 경우 사망에 이를 수 있다 우리나라 사람에서는 점처럼 흑갈색의 작은 혹으로 생겨 커지는 경우가 존재한다.

2. 피부암 현황

건강보험심사평가원의 통계에 따르면 2016년 우리나라의 암 발생 총 254,718건 중, 피부암은 1,344건으로 전체 암 발생의 0.5%를 차지하고 있으며 연령대별로는 60대가 19.1% 80대 이상이 27.0% 70대가 32,5%로 가장 많았다(그림5,5-1)

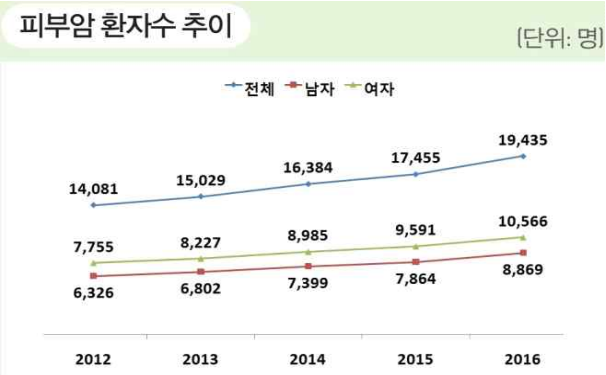

그림5. 2006년 성별 연령별 피부암 환자수 추이

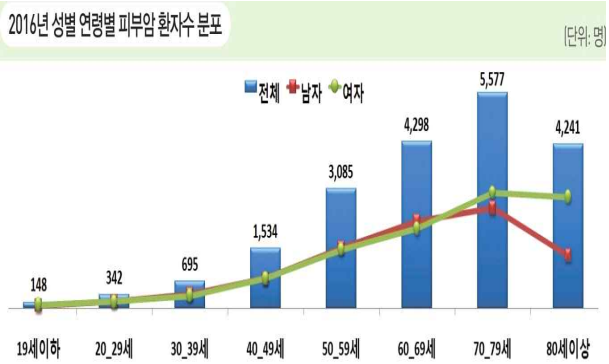

그림5-1. 2006년 성별 연령별 피부암 환자수 분포

보건의료빅데이터개방시스템을 이용하여 연도별 피부암 환자 수를 조회한 결과, 2010년 부터 2020년까지 흑색종과 기타 피부암 환자가 각각 2,343명에서 4,424명으로, 7,582명에서 19,093명으로 지속적으로 증가하는 양상을 보였다(그림6)

그림6. 연도별 피부암 및 노인성피부질환 환자 수

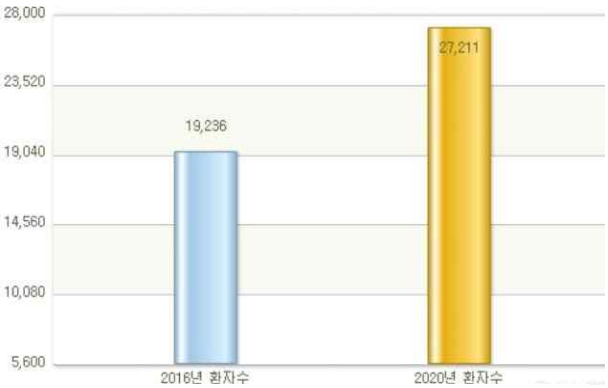

그림7. 2016년과 2020년 피부암 환자 수 비교

그림8. 악성흑색종 2010년 2021년 환자수 비교

피부암은 우리나라를 비롯한 동양인의 발병률이 낮아서 서양인의 질환으로 여겼으나, 국내 인구의 고령화, 자외선 등 피부 자극에 노출되는 환경이 증가하고 있다 특히, 피부암 종류 중 치료 결과가 가장 안좋은 악성흑색종도 증가 추세에 있어서 조기 발견이 중요하며 관심이 필요하다 건강보험심사평가원 자료를 보면 2011년 2576명이던 흑색종 진료 환자가 2021년 4734명으로 두 배 가까이 뛰었다(그림8)

III. 피부암 예측 알고리즘 분석

1. CNN(컨볼루션 신경망)

합성곱 신경망(CNN)은 인공신경망(ANN) 중 합성곱 연산을 사용하여 데이터를 3차원 데이터의 공간적인 정보를 유지하여 다음 레이어로 보낸다. 다차원 배열 데이터를 처리가 가능하기에 컬러 이미지 데이터를 처리하기에 용이하며, 일반적인 신경망은 이미지 데이터를 원본 그대로 처리를 하지만 CNN은 이미지내의 특징들을 추출하기 때문에 일반적인 신경망의 전역적 특징 학습의 보완하고 공간 정보를 유지 할 수 있다.(그림 9)

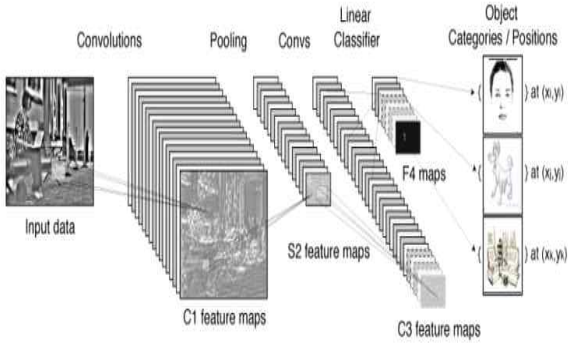

그림9. 컨볼루션 신경망(CNN)

컨볼루션 신경망은 크게 컨볼수션층(합성곱층)과 풀링층로 구성이 되어있다.

컨볼루션층은 이미지를 다차원 배열을 통해 이미지의 특징들을 추출하는 역할을 하기에 CNN의 중요한 기능을 하는 층이다. 컨볼루션층에 사용 되는 기능에는 컨볼루션 연산(합성곱연산), 패딩이 있으며,컨볼루션 연산은 컨볼루션층이 이미지의 특징을 추출 할 수 있도록 해주는 것이며 커널 또는 필터라는 n*m 크기의 행렬로 높이*너비 크기의 이미지를 처음부터 끝까지 겹치며 훑으면서 n*m 크기의 겹치는 부분을 각 이미지와 커널 원소를 곱하여 모두 더한 값을 출력을 한다.(그림 10)

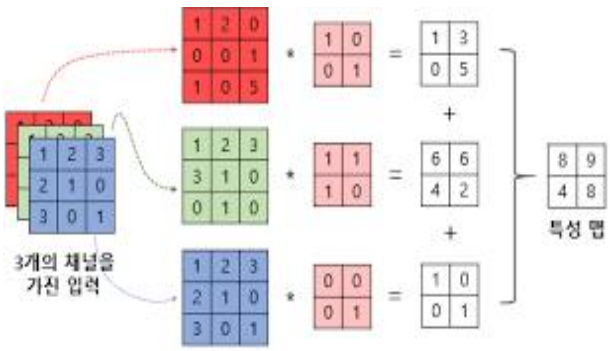

그림10. 컨볼루션층

패딩을 사용하게 되면 특징맵의 크기가 감소하지 않고, 입력 데이터의 형태와 동일한 형태의 출력값을 얻게 되어 주로 특징맵의 크기를 입력 데이터와 동일하게 얻기 위해 사용이 된다.(그림11)

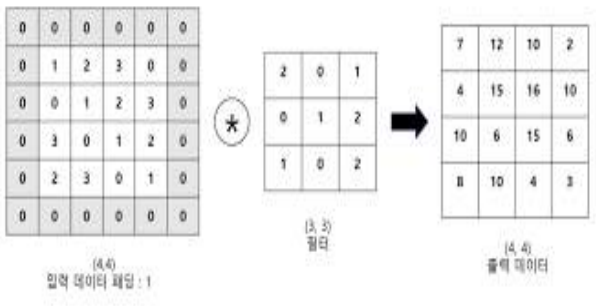

그림11. 컨볼루션층-패딩

풀링층의 풀링 연산은 모델이 중요한 특징을 학습할 수 있도록 함과 동시에 CNN이 이동 불변성 특성을 가지도록 하여 위치가 변동을 해도 물체를 인식 하도록 한다. 또한 모델 파라미터 수를 줄여주는 것으로 과대적합 문제 노출을 줄여 준다.(그림12)

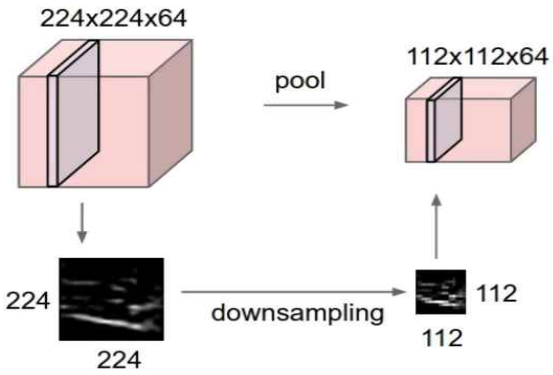

그림12. 풀링연산 이동 불변성

CNN의 개념을 확인을 한 후 저희가 진행하는 프로젝트와 적합한 알고리즘이라고 판단을 하여 CNN을 사용하여 예측 모델을 만들기로 결정했다.

2. Drop Out(드롭아웃)

과대적합은 모델이 예측 시 학습한 데이터에서는 성능이 좋지만 새로운 데이터에 대해서는 성능 낮은 현상을 말한다. 흔히 모델이 단순히 외웠다고 표현을 하며 그렇기에 모델이 문제를 일반화하지 못하여 예측모델로서는 적합하지 않게 된다. 그렇기에 과대적합을 방지하기 위해 드롭아웃을 이용하여 과대적합을 예방하기로 하였다.

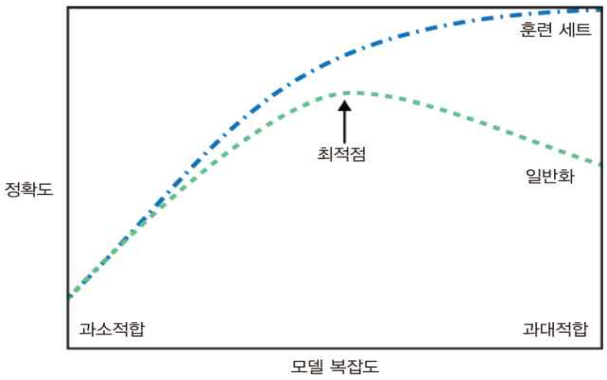

그림13. 과소적합 최적점 과대적합 비교

과대적합을 예방하기 위해서는 규제화 함수, 배치 정규화도 방법이긴 하다. 하지만 드롭아웃을 선택을 한 이유는 2개의 방식을 비교를 하였을 때 드롭아웃을 하는 것이 과대적합을 잘 예방을 하기에 드롭아웃을 선택을 했다. 드롭아웃은 학습이 진행되는 동안 신경망의 일부 유닛을 제외 하는 것으로 집중적으로 학습을 할 수 있도록 돕기 때문에 과대적합 방지와 더 나은 성능을 나오게 한다.

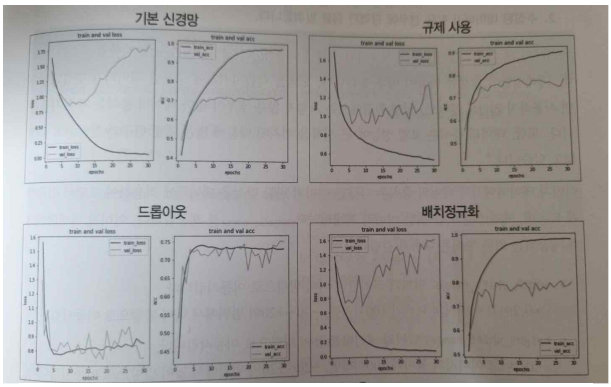

그림14. 과대적합 예방 방법 결과

3. 이미지 증식 및 전이학습

이미지 증식은 기존의 데이터에 변형을 추가하여 새로운 데이터를 만들어 일반화 문제를 예방할 수 있도록 도와준다. 또한 이미지 증식을 하여 다양한 데이터를 입력시키기에 모델을 더욱 견고하게 만들어 준다. 그렇기에 이미지가 적은 이 프로젝트에 적합하다고 판단을 하여 이미지 증식을 이용하게 되었으며 이미지 증식은 케라스에(keras)에서 제공하는 ImageGenerator를 통해 이미지 증식을 하였다. ImageGenerator에서 제공하는 변환 방식 중 아래의 변환 방식을 선택하고 설정했다.

- rescale: 이미지 픽셀값의 크기 조절 (설정값 1./255)
- horizontal_flip: 임으로 이미지를 수평 방향 뒤집기 (설정값 True)
- zoom_range: 임의의 비율만큼 이미지 확대/축소 (설정값 0.2)
- width_shift_range: 임의의 크기만큼 너비 방향 이동 (설정값 0.1)
- height_shift_range: 임의의 크기만큼 높이 방향 이동 (설정값 0.1)
- rotation_range: 이미지를 임의로 회전 (설정값: 30)

- fill_mode: 이미지 변환시 새로 생기는 픽셀을 채울방법 결정 (설정값 nearest)

전이학습은 하나의 작업을 위해 훈련된 모델을 유사 작업 수행 모델의 시작점을 활용하는 것으로 처음부터 새로 훈련하는 것 보다 전이학습을 통해 업데이트하고 재훈련하는 것이 더 빠르고 간편하기에 사용하기로 하였습니다. 케라스에서 제공되는 학습 모델 중 깊이가 깊지 않아 빠르면서도 정확도가 높은 VGG16모델을 사용하기로 하였습니다.

모델	용량	Top-1 정확도	Top-5 정확도	파라미터 수	깊이
Xception	88MB	0.790	0.945	22,910,480	126
VGG16	528MB	0.713	0.901	138,357,544	23
VGG19	549MB	0.713	0.900	143,667,240	26
ResNet50	98MB	0.749	0.921	25,636,712	–
ResNet101	171MB	0.764	0.928	44,707,176	–
ResNet152	232MB	0.766	0.931	60,419,944	–
ResNet50V2	98MB	0.760	0.930	25,613,800	–
ResNet101V2	171MB	0.772	0.938	44,675,560	–
ResNet152V2	232MB	0.780	0.942	60,380,648	–
InceptionV3	92MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215MB	0.803	0.953	55,873,736	572
MobileNet	16MB	0.704	0.895	4,253,864	88
MobileNetV2	14MB	0.713	0.901	3,538,984	88
DenseNet121	33MB	0.750	0.923	8,062,504	121
DenseNet169	57MB	0.762	0.932	14,307,880	169
DenseNet201	80MB	0.773	0.936	20,242,984	201
NASNetMobile	23MB	0.744	0.919	5,326,716	–
NASNetLarge	343MB	0.825	0.960	88,949,818	–

그림15. 전이학습 모델 비교

IV. 연구결과

모델의 층은 4가지의 방식으로 쌓았으며 아래의 결과물들은 4가지의 층의 모델에 대한 학습 그래프, 정확도와 혼동행렬의 결과물이다. 층을 쌓은 모습은 아래의 사진과 같다.(그림16)

```
model = Sequential()

model.add(Conv2D(filters = 128, kernel_size = 3, padding ='same',
                 activation = 'relu', input_shape = (150, 150, 3)))
model.add(Conv2D(filters = 128, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters =64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(64, activation = 'relu'))
model.add(Dense(7, activation = 'softmax'))
```

그림16-1. 모델1

```
model = Sequential()

model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu', input_shape = (150, 150, 3)))
model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 128, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 128, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
```

그림16-2. 모델2

```
model = Sequential()

model.add(Conv2D(filters = 256, kernel_size = 3, padding ='same',
                 activation = 'relu', input_shape = (150, 150, 3)))
model.add(Conv2D(filters = 256, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 128, kernel_size = 3, padding ='same',
                 activation = 'relu', input_shape = (150, 150, 3)))
model.add(Conv2D(filters = 128, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 128, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters =128, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(64, activation = 'relu'))
model.add(Dense(7, activation = 'softmax'))
```

그림16-3. 모델3

```
model = Sequential()
model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu', input_shape = (150, 150, 3)))
model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 16, kernel_size = 3, padding ='same',
                 activation = 'relu', input_shape = (150, 150, 3)))
model.add(Conv2D(filters = 16, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 16, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters =16, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 64, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 16, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 16, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 16, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 16, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(Conv2D(filters = 32, kernel_size = 3, padding ='same',
                 activation = 'relu'))
model.add(MaxPool2D(pool_size = (2,2), strides = 2, padding='same'))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(64, activation = 'relu'))
model.add(Dense(7, activation = 'softmax'))

model.compile(optimizer = Adam(1e-4),
              loss = 'categorical_crossentropy',
              metrics=['acc'])
```

그림16-4. 모델4

학습을 할 때 할당 된 GPU에 넘치지 않도록 다양하게 쌓았으며 각 모델을 설명을 하면 모델1은 신경망을 128, 64개로 주고 합성곱층을 3개로 쌓았다. 모델2는 신경망 128, 64, 32개로 주고 합성곱층을 6개로 주었다. 모델3은 신경망을 256, 128, 64로 주었으며 합성곱층은 4개로 쌓았다. 모델4는 신경망을 64, 32, 16로 주었으며 합성곱층은 7개로 쌓았다. 아래의 결과들을 통해 적당한 층의 모델을 설명하겠다.(그림17)

1. 학습결과 그래프 비교

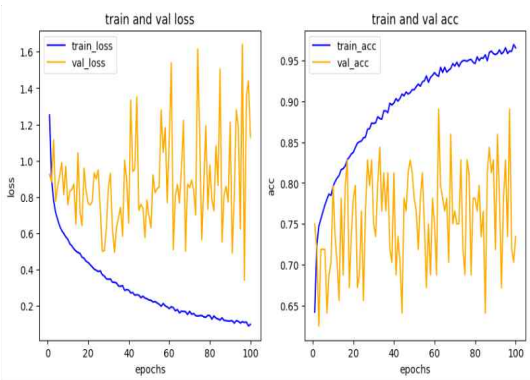

그림17-1. 모델1 학습결과

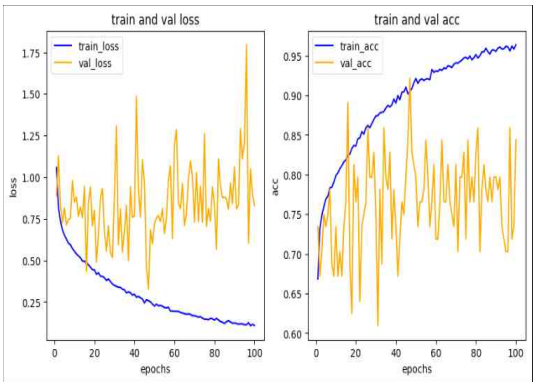

그림17-2. 모델2 학습결과

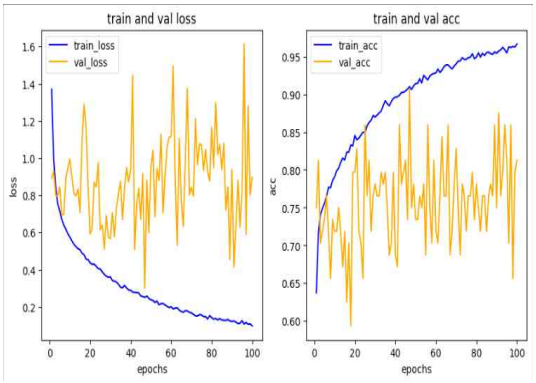

그림17-3. 모델3 학습결과

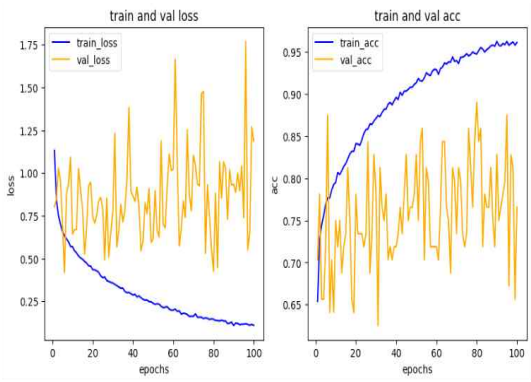

그림17-4. 모델4 학습결과

학습결과는 이미지 증식과 전이학습 후의 모델의 결과이다. epoch를 100번으로 한 것은 100번 후 부터는 정확도 1로 수치가 고정이 되어 결과로 보여지기에 부적합하다고 판단하였기 때문이다. 전체적인 학습결과를 비교를 하였을 때 과적합이 일어나지 않았으며 정확도가 몇 번 지점에서 가장 높은지에 대한 정보만 얻을 수 있을 뿐 어느 모델이 더 뛰어 난지에 대해서는 정확히 알 수는 없었다. 단, 모델1과 모델3은 전이학습 전에는 과적합이 일어나는 모습을 보여주었지만 전이학습을 한 후에는 과적합이 줄어든 모습을 볼 수 있다.

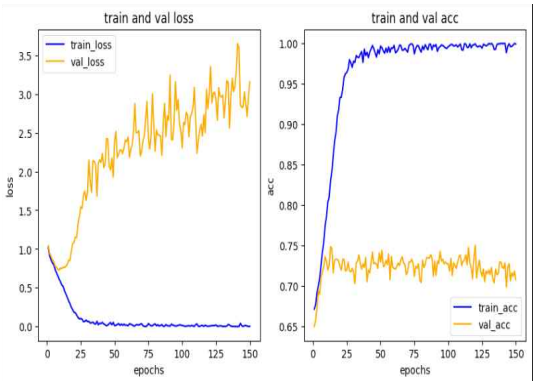

그림18-1. 모델1 이미지증식 전이 학습 전

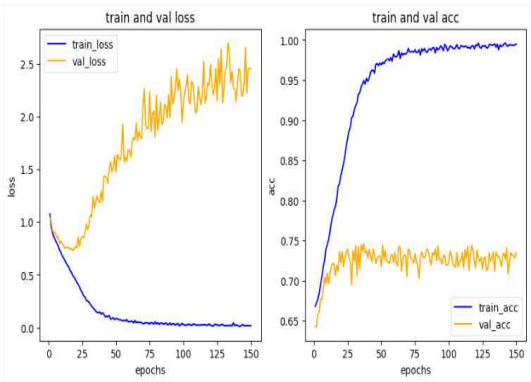

그림18-2. 모델2 이미지증식 전이 학습 전

2. 모델 정확도 비교

정확도(Accuracy)는 전체 데이터 중 실제 데이터의 정답과 모델이 예측한 정답이 같은 비율을 보여주는 것이기에 일반적으로 모델의 성능을 보여주기에는 괜찮은 수치이다. 각 모델의 정확도를 보게 되면 모델1 0.9205, 모델2 0.9052, 모델3 0.9100 모델4 0.7768로 “모델1의 정확도가 가장 높고 모델4의 정확도가 가장 낮기에 모델1이 가장 좋다.” 라는 결과를 도출 할 수 있지만 kaggle에서 수집한 자료는 불균형을 이루고 있는 상태이기에 정확도만 가지고는 모델 판별이 제대로 되지 않는다.(그림19)

그렇기에 혼동행렬을 통해서 정밀도(Precision)와 재현율(Recall)을 비교하여 좀 더 정확한 비교를 하여 모델을 선택을 하고자 한다.(그림20)

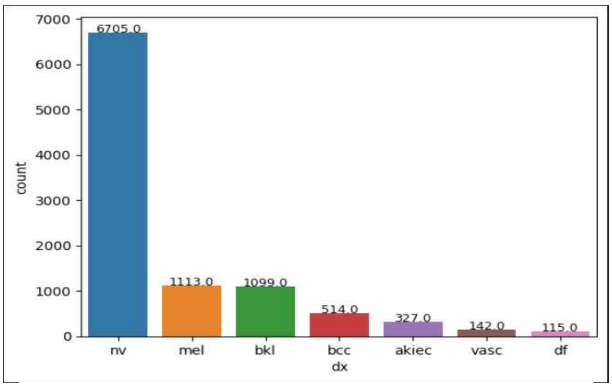

그림19. 클래스별 이미지 개수

3. 모델예측 검증 비교

	precision	recall	f1-score	support
0	0.79	0.43	0.56	44
1	0.69	0.68	0.69	60
2	0.61	0.57	0.59	109
3	0.56	0.50	0.53	10
4	0.50	0.40	0.45	99
5	0.85	0.93	0.89	663
6	1.00	0.29	0.45	17
accuracy			0.79	1002
macro avg	0.72	0.54	0.59	1002
weighted avg	0.78	0.79	0.78	1002

그림20-1. 모델1 혼동행렬

	precision	recall	f1-score	support
0	0.75	0.20	0.32	44
1	0.84	0.52	0.64	60
2	0.60	0.52	0.56	109
3	0.60	0.30	0.40	10
4	0.40	0.62	0.49	99
5	0.88	0.91	0.89	663
6	0.91	0.59	0.71	17
accuracy			0.77	1002
macro avg	0.71	0.52	0.57	1002
weighted avg	0.79	0.77	0.77	1002

그림20-2. 모델2 혼동행렬

	precision	recall	f1-score	support
0	0.77	0.39	0.52	44
1	0.68	0.63	0.66	60
2	0.65	0.54	0.59	109
3	0.38	0.50	0.43	10
4	0.39	0.53	0.45	99
5	0.88	0.89	0.88	663
6	0.91	0.59	0.71	17
accuracy			0.77	1002
macro avg	0.67	0.58	0.61	1002
weighted avg	0.78	0.77	0.77	1002

그림20-3. 모델3 혼동행렬

	precision	recall	f1-score	support
0	0.58	0.34	0.43	44
1	0.75	0.40	0.52	60
2	0.50	0.62	0.55	109
3	0.36	0.50	0.42	10
4	0.53	0.41	0.46	99
5	0.86	0.92	0.89	663
6	0.90	0.53	0.67	17
accuracy			0.77	1002
macro avg	0.64	0.53	0.56	1002
weighted avg	0.77	0.77	0.76	1002

그림20-1. 모델1 혼동행렬

(akiec:0, bcc:1, bkl:2, df:3, mel:4, nv:5, vasc:6)

정밀도(Precison)는 모델이 True라고 예측한 정답 중 실제 True인 비율, 재현율은 실제 데이터가 True인 것 중 모델이 True라고 예측한 비율을 나타낸다. 특히 클래스가 많은 모델일수록 각 클래스의 정답률을 확인 하는 것이 중요하다고 판단이 되어 각 모델의 정밀도와 재현율을 비교 하였다. 비교 결과 한 모델이 정밀도가 높다고 해서 다른 모델 보다 재현율이 높지는 않다는 것을 파악을 하였다. 정밀도와 재현율이 모두 높은 모델을 선택 하기는 힘들기에 프로젝트 주제 상 예측한 정답 중 실제 True 비율이 높아야 한다고 판단을 하여 정밀도가 다른 모델보다 준수한 모델1을 적합한 모델이라고 판단했다.

Ⅴ 결론

본 프로젝트를 간략히 요약하면 합성곱 신경망(콘볼루션 신경망, Convolutional neural network)을 이용하여 피부암 예측모델을 개발하는 것이다. 모델의 층은 4가지의 방식으로 쌓았으며, 개발한 피부암 예측모델을 이용한 성능 확인을 위해 각 모델의 정확도를 분석해 보았을 때 모델1 0.9205, 모델2 0.9052, 모델3 0.9100 모델4 0.7768로 모델1의 정확도가 가장 높음을 알 수 있었다. 모델1은 신경망을 128, 64개로 주고 합성곱층을 3개로 쌓았다. 또한 전체적인 학습결과를 비교를 하였을 때 정확도가 몇 번 지점에서 가장 높은지에 대한 정보만 얻을 수 있을 뿐 어느 모델이 더 뛰어난지에 대해서는 정확히 알아낼 수 없었다. 단, 모델1과 모델3은 전이학습 전에는 과적합이 일어나는 모습을 보여주었지만 전이학습을 한 후에는 과적합이 줄어든 모습을 볼 수 있었다. 성공적인 결과를 얻음으로써 본 논문의 연구가 이론적인 내용만을 다루는 것이 아니라 실제 적용이 가능함을 증명하였다 볼 수 있다. 다만 이미지를 끌어온 KAGGLE는 자료의 불균형을 이루고 있는 상태라 위 결과가 완벽하다 단정지을 순 없다 이후 연구에서는 개발한 프로그램을 더 다양한 방법과 자료를 토대로 보완해 나가는 추가 연구가 필요할 것이다.

본 프로젝트가 완성이 된다면, 아래와 같은 기대효과를 얻을 수 있다.

- 병원에 가지 않아도 증상을 파악이 가능하여 고령층에 도움이 됨
- 자가진단을 통한 자기 몸 관리가 가능
- 비대면 진단 시 빠른 진단과 조치가 가능

참고문헌

1. 피부과 전무의가 바라본 피부질환 진료환경의 변화. 조성진
2. 생활 속 질병통계 100선. 건강보험심사평가원
3. Nature & Life 암/ 피부암. 편평상피세포암. 2022.10.08
4. MSD 매뉴얼. 피부암 개요
5. "뉴질랜드, 2020년 피부암 사망률 1위...호주, 서유럽, 북미". news THEONE. 2022.04.01
6. ONECHURCH. "뉴질랜드 피부암". onechurch.nz/news_nz/62812 2022.10.08
7. 서울대학교암병원. "피부암 암종별 의학정보"
8. 강북삼성병원. "고령화로 증가하는 피부암&악성흑색종"
9. 휘경우리들내과의원. "피부암, 조기 발견이 쉬운 학성 흑색종"
10. 고려대학교안암병원. "질병정보. 피부암“
11. ABODY “악성 흑색종 증상 및 치료”
12. 백견불여일타 딥러닝 입문 “2020년 6월 8일 조휘용 지음, 로드북”
13. 실전 텐서플로2를 활용한 딥러닝 컴퓨터 비전 “2020년 6월 5일 벤자민 플랜치, 앨리엇 안드레스 지음/ 김정인 옮김, 위키북스”
14. 딥 러닝 기반의 악성흑색종 분류를 위한 컴퓨터 보조진단 알고리즘 “저자 임 상 헌・이 명 숙”