인공신경망을 활용한 자폐증 조기 진단

-CNN, ResNet, DenseNet-

윤현서

목 차

Ⅰ. 서론

1. 연구의 필요성 및 목적

1) 연구의 필요성

(1) 자폐증

- 사회 기술, 언어, 의사소통 발달 등이 지연되거나 비정상적인 기능을 보이는 발달 장애[1)]
- 선천적 요인, 생화학적 요인, 유전적 요인, 뇌와 관련된 요인 등이 질병의 원인으로 추측되고 있으나 현재 명확한 원인은 발견되지 않음[2)]
- 자폐증의 68명 중 1명[3)]꼴로 나타나며 남자가 여자보다 4배 더 발생[4)]
- 행동치료, 언어치료[5)], 약물치료(향정신성의약품)[6)]로 증상을 효과적으로 완화할 수 있음
- 자폐증 환자 중 지능(IQ)이 70 이상이고 5~7세 수준의 언어 소통 능력을 보유한 경우, 최상의 예후를 보임[7)]

(2) 자폐증의 위험성

- 현재 자폐증을 완치하는 치료법은 없음
- 자폐증 아이에서 정신 지체가 75%로 빈번하고 경련성 질환도 높은 빈도로 발견됨[8)]
- 사회적 격리, 고용 문제, 가족 스트레스, 집단 따돌림, 자해[9)] 등 사회 문제와 연관됨

(3) 자폐증 진단 연구의 필요성

- 자폐증을 보인 사람의 4분의 1이 진단받지 않음[10)]
- 자폐증 진단 과정은 비용이 많이 들고 복잡하며 평균 3.5년이 걸림[11)]
- 초기에 자폐증 진단을 받도록 권유할 수 있는 접근성 높은 연구 필요성 대두

2) 연구 목적

(1) 기존 연구

- Predictive value of morphological features in patients with autism versus normal controls[12)]
  • 얼굴 특징과 자폐증은 연관성을 보임
  • 6가지 얼굴 특징으로 자폐아의 88%를 정확하게 진단하고 대조군의 22%을 잘못 분류
- Facial phenotypes in subgroups of prepubertal boys with autism spectrum disorders are correlated with clinical phenotypes[13)]
  • 자폐증이 있는 소년은 독특한 얼굴 구조를 보임
  • 자폐아와 부모의 12가지 신체 부위(키, 머리 모양, 귀, 코, 얼굴, 인중(코와 입 사이의 공간), 입 또는 입술, 치아, 손, 손가락 또는 엄지 손가락, 손톱 및 발)으로 평가

(2) 기존 연구의 한계 및 개선 방향

- Predictive value of morphological features in patients with autism versus normal controls
  • 논문은 의사결정트리를 활용한 통계적 접근만을 중점적으로 다루고 있음
  • CNN 기반 인공지능 기술을 바탕으로 분류하도록 프로젝트를 수행
- Facial phenotypes in subgroups of prepubertal boys with autism spectrum disorders are correlated with clinical phenotypes

• 소년만을 대상으로 하였고 안면 데이터 구축이 어려움
• 소년만이 아닌 전체 성별을 대상으로 얼굴 이미지를 사용하여 프로젝트를 수행

2. 연구주제

- 인공 신경망을 활용한 자폐증 조기 진단

3. 시사점

1) 얼굴 특성과 자폐증 진단을 다룬 기존 연구와 성능 비교
2) 다양한 인공 신경망 기술을 적용하여 최적 모델 발견
3) 얼굴 이미지만으로 간편한 자폐 진단 가능
4) 조기 자폐 진단을 통해 치료 예후 향상

Ⅱ. 이론적 배경

1. CNN

1) 등장 배경

(1) CNN은 1989년 LeCun이 발표한 “Backpropagation applied to handwritten zip code recognition[14]” 논문에서 필기체 인식과 함께 처음 소개됨

(2) 기존 Multi-Layer Neural Network에서의 변수의 개수, 네트워크 크기, 학습 시간 문제를 개선하기 위해 CNN이 등장

(3) 이미지의 공간 정보를 유지하면서 인접 이미지와의 특징을 효과적으로 인식하고 강조하는 방식을 사용

2) 모델 구조

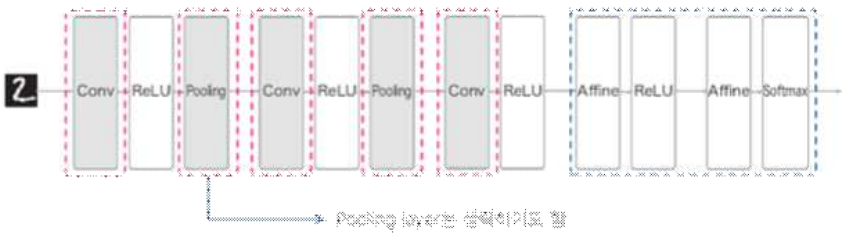

그림 1 CNN 모델의 구조

[출처:밑바닥부터 시작하는 딥러닝]

(1) CNN은 이미지 특징 추출 영역과 이미지 분류 영역으로 구성됨

(2) 특징 추출 영역은 컨볼루션 레이어와 풀링 레이어로 이루어짐

- 컨볼루션 레이어 : 필터로 공유 파라미터 수를 최소화하면서 이미지의 특징을 탐색
- 풀링 레이어 : 컨볼루션 레이어로부터 특징을 강화하고 모음

(3) 분류 영역은 CNN의 데이터 타입을 Fully Connected Neural Network의 형태로 변경하고 활성화 함수를 적용하여 분류 데이터를 출력

(4) 필터의 크기, 스트라이드, 패딩과 풀링 크기로 출력 데이터 크기를 조절하고, 필터의 개수로 출력 데이터의 채널을 결정

3) 주요 특징

(1) 컨볼루션 레이어에서는 3차원 데이터를 입력하고 3차원의 데이터로 출력하므로 데이터의 형상을 유지

(2) CNN는 같은 레이어 크기의 Fully Connected Neural Network와 비교해 볼 때, 더 작은 학습 파라미터로 더 높은 인식률을 제공

2. ResNet

1) 등장 배경

(1) ResNet은 2015년 ILSVRC 대회 (이미지넷 이미지 인식 대회)에서 우승한 마이크로소프트에서 개발한 알고리즘으로 2016년 “Deep Residual Learning for Image Recognition[15]“ 논문으로 처음 소개됨

(2) 기존 CNN 모델의 레이어가 깊어질수록 기울기 소실 · 폭주(gradient vanishing · exploding)가 발생하여 성능이 낮아지는 문제를 개선하기 위해 등장
(3) ResNet은 skip connection을 이용한 residual learning을 통해 기울기 소실 문제 해결

2) 모델 구조

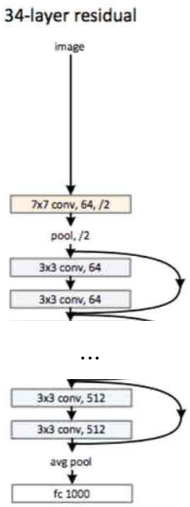

그림 3 ResNet
모델 구조
[출처:ResNet 논문]

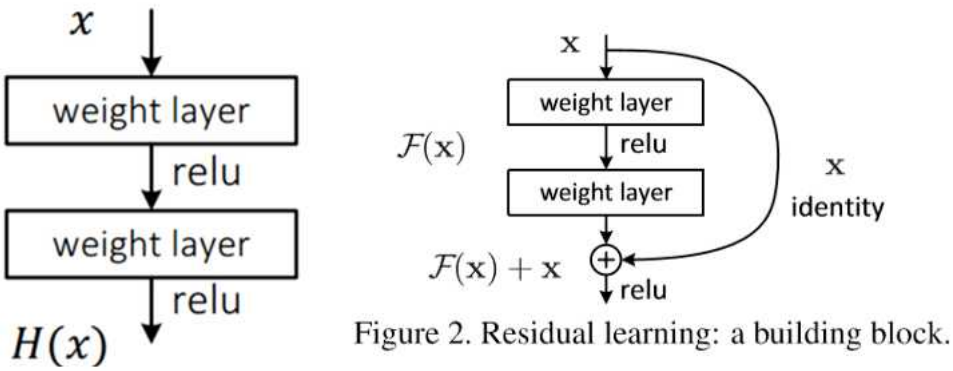

그림 4 Residual learning : identity block
[출처:Deep Residual Learning for Image Recognition]

(1) ResNet은 identity block과 convolution block으로 구성됨
(2) identity block : 네트워크의 출력 F(x)에 x를 더함
- 이미지에서는 H(x) = x, 네트워크의 출력 F(x)는 0이 되도록 학습
- F(x)+x=H(x)=x가 되도록 학습하면 F(x)+x의 미분값은 F'(x) + 1로 최소 1 이상
- 모든 레이어에서 기울기가 1+F'(x)이므로 기울기 소실 문제를 해결
(3) convolution block : 네트워크의 출력 F(x)에 1X1 컨볼루션 연산을 수행한 x를 더함

3) 주요 특징

(1) Residual learning을 사용하여 더 깊은 Network를 쉽게 최적화

(2) 기존 CNN 모델보다 레이어가 깊어질수록 정확도 향상

3. DenseNet

1) 등장 배경

(1) DenseNet은 비교적 최근인 2017년 “Densely Connected Convolutional Networks[16]” 논문에서 처음 소개됨

(2) 기존 모델의 레이어가 깊어질수록 발생하는 기울기 소실 문제를 개선하기 위해 등장

(3) 선행 레이어에서 후속 레이어로 향하는 shortpath를 만드는 ResNet, Stochastic depth, FractalNet 등의 접근 방식으로부터 simple connectivity pattern의 구조 제안

(4) DNN은 특징맵 크기가 동일한 모든 레이어가 직접 연결하여 네트워크의 레이어 간 정보 흐름을 극대화함

2) 모델 구조

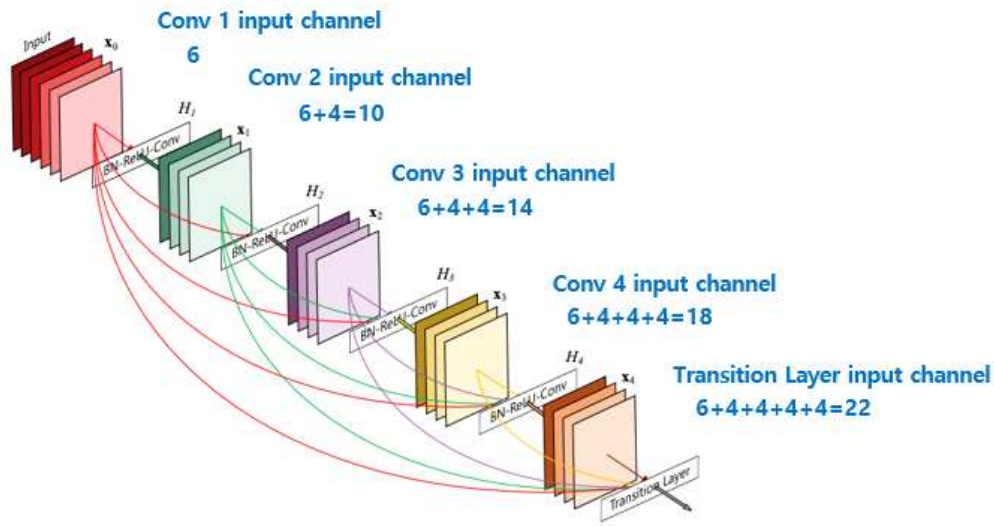

Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

그림 5 DenseNet 모델 구조

[출처:Densely Connected Convolutional Networks]

(1) 이전 레이어의 특징 맵을 계속해 다음 레이어의 입력과 연결하는 방식

(2) 연결은 특징 맵들의 더하기(ResNet의 방식)가 아닌 Concatenation을 사용

3) 주요 특징

(1) 네트워크 전체에서 개선된 정보와 데이터를 통해 기울기 소실 문제 완화

(2) 정규화 효과를 포함하므로, 더 작은 학습 데이터셋에 대한 과적합 감소

(3) 중복된 특징 맵은 다시 학습하지 않아 기존의 CNN보다 파라미터 개수 감소

Ⅲ. 연구 방법

1. 데이터셋

1) 데이터셋

- Kaggle의 자폐아 이미지 데이터셋[17)]

2) 데이터셋 구성

(1) jpg 형식의 이미지 2,940개

(2) 자폐아 이미지 1,470개와 비자폐아 이미지 1,470개

(2) 나이 : 2 ~ 14세

(3) 성별 비율 (남 : 여)

- 자폐아의 남녀 이미지 분포는 3 : 1
- 비자폐아의 남녀 이미지 분포는 1 : 1

(4) 인종 비율

- 백인과 유색 인종의 분포는 10 : 1

2. 세부 코드

1) 프로그램 환경 및 도구

- 플랫폼 : Google Colab[18)]
- 언어 : Python
- 파일 형식 : 주피터 노트북(.ipynb)

2) 세부 코드

* 코드가 방대하여 데이터 · 성능 시각화 및 기타 설정 코드는 첨부파일로 추가하였습니다.
* 아래 링크로 세부 코드를 볼 수도 있습니다.

- CNN 모델

• https://colab.research.google.com/drive/123P2jiyVlGO_q3tKRV-VlDQ2urVLReRv?usp=sharing

- ResNet 모델

• https://colab.research.google.com/drive/1azCECvPehMqgwtACrkElyddx41EG4jAf?usp=sharing

- DenseNet 모델

• https://colab.research.google.com/drive/1AtP2USbOEa5VNWLMSscYKJECNIvSvOCE?usp=sharing

(1) 환경 설정

- 사용 라이브러리

```
from tensorflow import keras
import tensorflow as  tf
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Dense, Activation,Dropout,Conv2D, MaxPooling2D,BatchNormalization
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, Sequential
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import os
from PIL import Image
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
```

(2) 데이터

- 파라미터 및 변수 설정

```
classes=os.listdir(train_dir)
batch_size=56 # 배치 크기
rand_seed=1220
start_epoch=0 # 시작 에폭
epochs=30 # 에폭
img_size=224 # 이미지 크기 : 224 X 224
lr=.001 # 학습률
```

- 데이터 크기에 따른 배치 설정 함수

```
# validation, test 배치 설정 함수
def get_bs(dir,b_max):
    length=0
    dir_list=os.listdir(dir)
    for d in dir_list:
        d_path=os.path.join (dir,d)
        length=length + len(os.listdir(d_path))
    batch_size=sorted([int(length/n) for n in range(1,length+1) if length % n ==0 and length/
n<=b_max],reverse=True)[0]
    return batch_size,int(length/batch_size)
```

- validation, test 데이터 배치 · 스텝 크기

```
valid_batch_size, valid_steps=get_bs(val_dir, 100)
test_batch_size, test_steps=get_bs(test_dir,100)
```

- 데이터 증강 : 가로 반전으로 데이터 2배 증가

```
# 이미지 인코딩 mobilenet 사용, 가로 반전
# 폴더로부터 이미지 크기 224 X 224, categorical(2D 원핫 인코딩된 라벨), rgb(3 개 채널)
train_gen=ImageDataGenerator(preprocessing_function=keras.applications.mobilenet.preprocess
_input, horizontal_flip=True).flow_from_directory(train_dir,  target_size=(img_size, img_size),
batch_size=batch_size, seed=rand_seed, class_mode='categorical', color_mode='rgb')
valid_gen=ImageDataGenerator(preprocessing_function=keras.applications.mobilenet.preprocess
_input) .flow_from_directory(val_dir, target_size=(img_size, img_size), batch_size=valid_batch_
size,class_mode='categorical',color_mode='rgb', shuffle=False)
test_gen=ImageDataGenerator(preprocessing_function=keras.applications.mobilenet.preprocess_i
nput).flow_from_directory(test_dir,target_size=(img_size, img_size), batch_size=test_batch_size,
class_mode='categorical',color_mode='rgb', shuffle=False )

test_file_names=test_gen.filenames  # 성능평가에 사용할 test 파일 이름
test_labels=test_gen.labels # 성능평가에 사용할 test 라벨
```

```
val_file_names=valid_gen.filenames  # 학습에 사용할 validation 파일 이름
val_labels=valid_gen.labels # 학습에 사용할 validation 라벨
```

(3) 모델

- CNN 모델

```
model = Sequential()
input_shape = (224,224,3)
model.add(Conv2D(64, (10, 10), input_shape=input_shape,activation='relu', padding='same', kernel_initializer='he_normal'))
model.add(Conv2D(64, (10, 10), activation='relu', padding='same', kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (10, 10),activation='relu',padding='same', kernel_initializer='he_normal'))
model.add(Conv2D(128, (10, 10),activation='relu',padding='same', kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (6, 6),activation='relu',padding='same', kernel_initializer='he_normal'))
model.add(Conv2D(64, (6, 6),activation='relu',padding='same', kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dense(2))
model.add(Activation('softmax'))
```

- ResNet[19] 모델

```
_model = tf.keras.applications.ResNet50V2( include_top=False, input_shape=(img_size, img_size,3), pooling='max', weights='imagenet')
_model.trainable=False
x=_model.layers[-1].output # 마지막 레이어 (Global Max Pooing 2D)
x=keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x) # 배치 정규화 추가
x=Dense(128, activation='relu')(x) # 활성화 함수 RELU
x=Dropout(rate=.4, seed = 123)(x) # Dropout 설정
predictions=Dense (len(classes), activation='softmax')(x) # 출력층 활성화 함수 Softmax
model = Model(inputs=_model.input, outputs=predictions)
for layer in model.layers:
    layer.trainable=True
```

- DenseNet[20] 모델

```
_model = tf.keras.applications.densenet.DenseNet201( include_top=False, input_shape=(img_size, img_size,3), pooling='max', weights='imagenet')
_model.trainable=False
x=_model.layers[-1].output # 마지막 레이어 (Global Max Pooing 2D)
x=keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x) # 배치 정규화 추가
x=Dense(128, activation='relu')(x) # 활성화 함수 RELU
```

```
x=Dropout(rate=.4, seed = 123)(x) # Dropout 설정
predictions=Dense (len(classes), activation='softmax')(x) # 출력층 활성화 함수 Softmax
model = Model(inputs=_model.input, outputs=predictions)
for layer in model.layers:
    layer.trainable=True
```

- 모델 컴파일

```
model.compile(Adamax(lr=lr), loss='categorical_crossentropy', metrics=['accuracy'])
```

- 조기 중단 및 최적 모델 저장 설정

```
es = EarlyStopping(monitor='val_loss', patience = 10, mode = 'min', restore_best_weights=Tru
e)
mc = ModelCheckpoint('/content/drive/MyDrive/best_densenet_model.h5', monitor='val_loss', m
ode='min', save_best_only=True)
```

- 모델 실행

```
results=model.fit(x=train_gen,  epochs=epochs, verbose=1, callbacks=[es, mc],  validation_data
=valid_gen,validation_steps=valid_steps,  shuffle=True,  initial_epoch=start_epoch)
```

(4) 성능 평가

- 모델 테스트셋 정확도

```
evl_test = model.evaluate(test_gen, batch_size=test_batch_size, verbose=1, steps=test_steps)
acc_test = evl_test[1]* 100
loss_test = evl_test[0]
print('정확도 : %5.2f' % (acc_test))
print('오차 : %5.2f' % (loss_test))
```

3. 모델 구조

1) CNN 모델

(1) 컨볼루션 레이어

- 필터 크기 : 10 X 10, 6 X 6

- Same Padding 사용 : 출력 이미지가 입력 이미지 크기와 같게 함

(2) Max Pooling

- 필터 크기 : 2 X 2

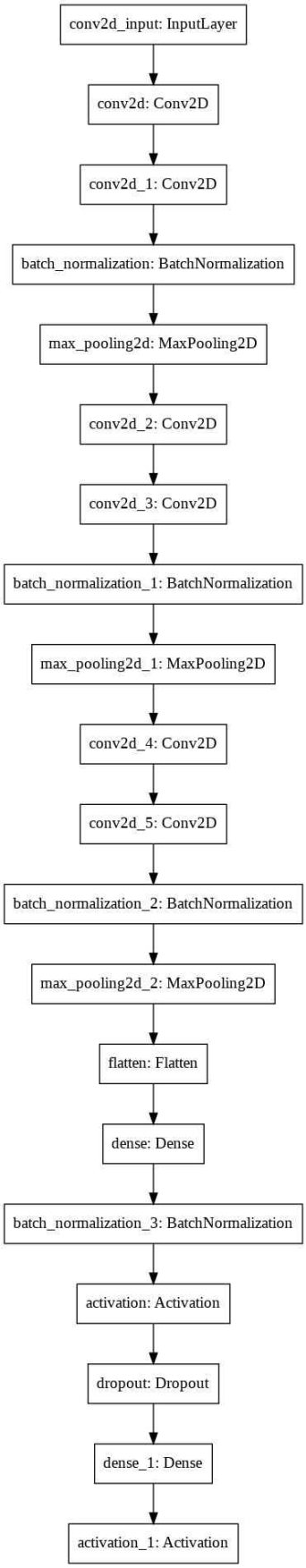
conv2d_input: InputLayer
conv2d: Conv2D
conv2d_1: Conv2D
batch_normalization: BatchNormalization
max_pooling2d: MaxPooling2D
conv2d_2: Conv2D
conv2d_3: Conv2D
batch_normalization_1: BatchNormalization
max_pooling2d_1: MaxPooling2D
conv2d_4: Conv2D
conv2d_5: Conv2D
batch_normalization_2: BatchNormalization
max_pooling2d_2: MaxPooling2D
flatten: Flatten
dense: Dense
batch_normalization_3: BatchNormalization
activation: Activation
dropout: Dropout
dense_1: Dense
activation_1: Activation

그림 6 CNN 모델 구조

2) ResNet 모델

* 모델 이미지 용량이 커서 한글 파일에 첨부되지 않아 첨부파일로 제출합니다.

- Deep Residual Learning for Image Recognition 논문의 ResNet 모델과 동일
- 마지막 층인 Global Max Pooing 2D 레이어에
• 배치 정규화
• 은닉층 : 활성화 함수 - RELU, 노드 - 128개
• 드롭아웃 : 0.4
• 출력층 : 활성화 함수 - Softmax, 노드 - 2개(클래스 개수) 추가함

3) DenseNet 모델

* 모델 이미지 용량이 커서 한글 파일에 첨부되지 않아 첨부파일로 제출합니다.

- Densely Connected Convolutional Networks 논문의 DenseNet 모델과 동일
- 마지막 층인 Global Max Pooing 2D[21] 레이어에
• 배치 정규화
• 은닉층 : 활성화 함수 - RELU, 노드 - 128개
• 드롭아웃 : 0.4
• 출력층 : 활성화 함수 - Softmax, 노드 - 2개(클래스 개수) 추가함

4. 모델 파라미터

1) 배치 크기 : 56 / 100 / 100 (훈련 / 검증 / 평가)
2) 에폭 : 30
3) 학습률 : 0.001
4) 최적화 함수 : AdaMax[22]
5) 오차(loss) 함수 : Categorical Crossentropy
5) Early Stoping : 오차가 10 에폭 이상 감소하지 않으면 중단
6) 모델 하단 구조
(1) 배치 정규화
(2) 은닉층 : 활성화 함수 - RELU, 노드 - 128개
(3) 드롭아웃 : 0.4
(4) 출력층 : 활성화 함수 - Softmax, 노드 - 2개

Ⅳ. 연구 결과

1. CNN 모델

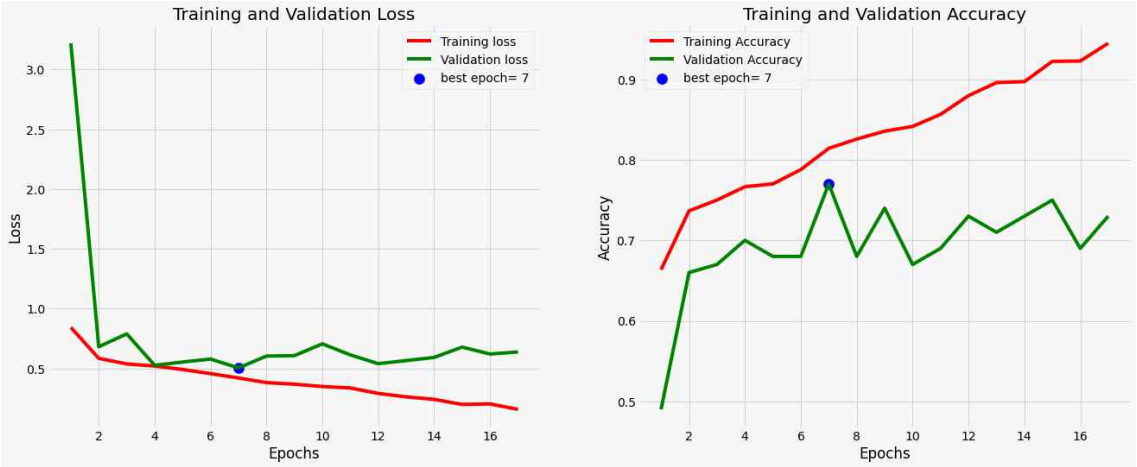

그림 7 CNN 모델 학습 과정 성능

- 최적 학습 모델에서의 에폭 : 7
- 검증 데이터 정확도 : 79.33%
- 검증 데이터 오차 : 0.47
- 학습 시간 : 약 42분

2. ResNet 모델

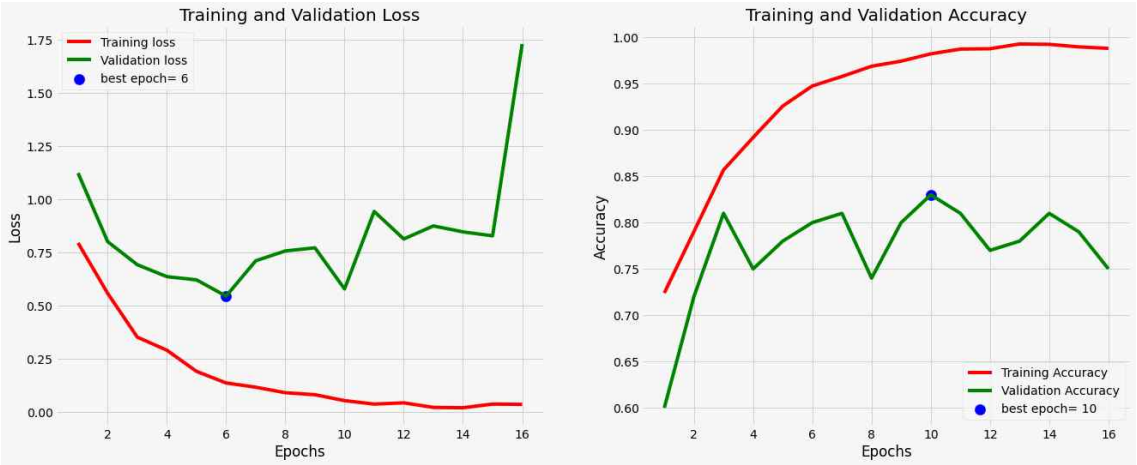

그림 8 ResNet 모델 학습 과정 성능

- 최적 학습 모델에서의 에폭 : 6
- 검증 데이터 정확도 : 84.00%
- 검증 데이터 오차 : 0.43
- 학습 시간 : 약 17분

3. DenseNet 모델

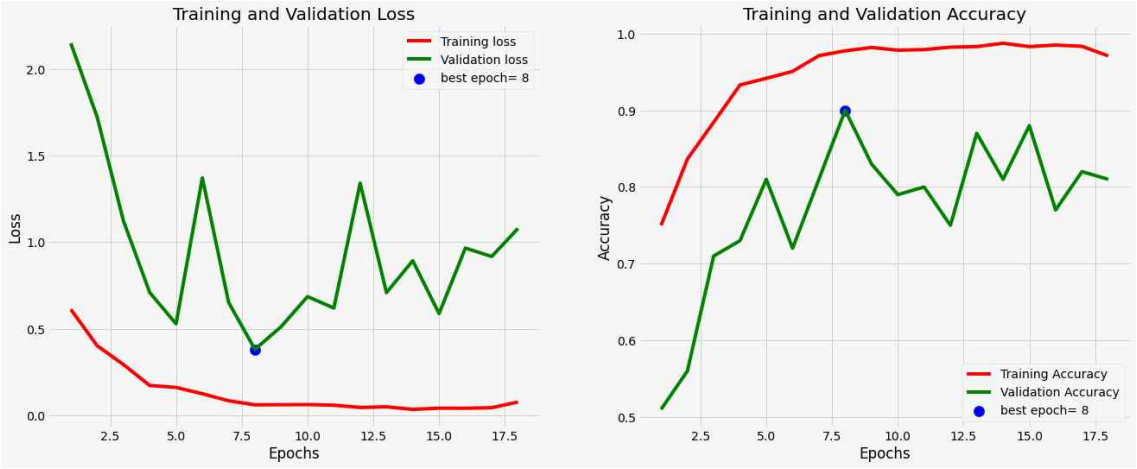

그림 9 DenseNet 모델 학습 과정 성능

- 최적 학습 모델에서의 에폭 : 8
- 검증 데이터 정확도 : 88.33%
- 검증 데이터 오차 : 0.41
- 학습 시간 : 약 13분

Ⅴ. 요약 및 결론

1. 결론

1) 전체 모델 성능 비교

	정확도	오차	학습 시간
CNN	79.33	0.47	33
ResNet	84	0.43	17
DenseNet	**88.33**	**0.41**	**13**

표 3 모델 성능 비교

(1) DenseNet 모델이 CNN, ResNet 모델보다 정확도, 오차, 학습 시간 면에서 모두 우수한 성능을 보임

(2) DenseNet 모델과 ResNet 모델에서의 정확도는 큰 차이를 보이지 않지만, 오차와 학습 시간을 감소시켰다는 점에서 더 좋은 모델로 평가됨

2) 기존 연구와 비교

(1) Predictive value of morphological features in patients with autism versus normal controls

- 해당 논문은 의사결정트리를 활용하여 정확도 88% 구현 (448명 백인 대상)
- 프로젝트는 DenseNet 모델을 활용하여 정확도 88.33% 구현 (2,940명 모든 인종 대상)

• 기존 연구보다 모든 인종을 대상으로 분류하였음에도 불구하고 높은 정확도를 보임

• 기존 연구보다 분류를 위해 데이터 수집 과정의 비용 절약

• 통계적 기술을 활용한 기존 연구보다 분류 정확도가 낮아 정확도 향상을 위해 파라미터 조정 필요

(2) Facial phenotypes in subgroups of prepubertal boys with autism spectrum disorders are correlated with clinical phenotypes

- 해당 논문은 t-test를 활용하여 연관성 발견 (448명 남아 대상, 3D 안면 데이터)
- 프로젝트는 DenseNet 모델을 활용 (2,940명 모든 성별 대상, 얼굴 이미지 데이터)

• 기존 연구보다 데이터 수집 과정에서 상당한 비용 절약

3) 개선 방향

(1) 높은 성능을 보여준 DenseNet 모델에 집중하여 최적 하이퍼 파라미터 탐색

(2) 추가적인 자폐증 이미지 데이터셋 확보

2. 요약

1) 조기 자폐증 진단을 받도록 권유할 수 있는 접근성 높은 연구 필요함

2) 자폐아 이미지 데이터로 자폐아를 분류하기 위해 CNN, ResNet, DenseNet 모델을 사용함

3) DenseNet 모델이 짧은 학습 시간 동안 정확도 88.33%로 CNN, ResNet 모델보다 높은 성능을 보임

4) 향후 하이퍼 파라미터 탐색을 통해 DenseNet 모델 성능 개선

참고 문헌

1) Landa RJ (2008). “Diagnosis of autism spectrum disorders in the first 3 years of life”. 《Nat Clin Pract Neurol》 4 (3): 138–147. doi:10.1038/ncpneuro0731. PMID 18253102
2) “삼성서울병원,“ 자폐증, 2014년 2월 5일 수정, 2020년 12월 10일 접속, http://www.samsunghospital.com/home/healthInfo/content/contenView.do?CONT_SRC_ID=09a4727a8000f31d&CONT_SRC=CMS&CONT_ID=3215&CONT_CLS_CD=001020001001.
3) “자폐 범주성 장애,“ MSD 메뉴얼 일반인용, 2018년 8월 6일 수정, 2020년 12월 10일 접속, https://www.msdmanuals.com/ko/홈/아동의-건강-문제/학습과-발달-장애/자폐-범주성-장애.
4) Jung, H., Park, H., Choi, Y., Kang, H., Lee, E., Kweon, H., ... & Rhim, I. (2018). Sexually dimorphic behavior, neuronal activity, and gene expression in Chd8-mutant mice. Nature neuroscience, 21(9), 1218-1228.
5) Myers SM, Johnson CP (November 2007). “Management of children with autism spectrum disorders”. 《Pediatrics》 120 (5): 1162–1182. doi:10.1542/peds.2007-2362. PMID 17967921.
6) Sukhodolsky DG, Bloch MH, Panza KE, Reichow B (November 2013). “Cognitive-behavioral therapy for anxiety in children with high-functioning autism: a meta-analysis”. 《Pediatrics》 132 (5): e1341–50. doi:10.1542/peds.2013-1193. PMC 3813396. PMID 24167175.
7) “삼성서울병원,“ 자폐증, 2014년 2월 5일 수정, 2020년 12월 10일 접속, http://www.samsunghospital.com/home/healthInfo/content/contenView.do?CONT_SRC_ID=09a4727a8000f31d&CONT_SRC=CMS&CONT_ID=3215&CONT_CLS_CD=001020001001.
8) Szymanski LS & Kaplan LC (1997): Mental Retardation, in Textbook of Child & Adolescent Psychiatry. edited by Wiener JM. pp. 183-218. American Psychiatric Press.
9) “Autism spectrum disorder - Symptoms and causes”. 《Mayo Clinic》, 2019년 7월 13일 접속.
10) “One-fourth of children with autism are undiagnosed,“ ScienceDaily, 2020년 1월 9일 수정, 2020년 12월 10일 접속, https://www.sciencedaily.com/releases/2020/01/200109130218.htm.
11) Crane, Laura, James W Chester, Lorna Goddard, Lucy A Henry, and Elisabeth Hill. “Experiences of Autism Diagnosis: A Survey of over 1000 Parents in the United Kingdom.” Autism 20, no. 2 (February 2016): 153–62. https://doi.org/10.1177/1362361315573636.
12) Ozgen H, Hellemann GS, de Jonge MV, Beemer FA, van Engeland H. Predictive value of morphological features in patients with autism versus normal controls. J Autism Dev Disord. 2013 Jan;43(1):147-55. doi: 10.1007/s10803-012-1554-4. PMID: 22669539; PMCID: PMC3536966.
13) Aldridge K, George ID, Cole KK, Austin JR, Takahashi TN, Duan Y, Miles JH. Facial phenotypes in subgroups of prepubertal boys with autism spectrum disorders are correlated with clinical phenotypes. Mol Autism. 2011 Oct 14;2(1):15. doi: 10.1186/2040-2392-2-15. PMID: 21999758; PMCID: PMC3212884.
14) LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.
15) He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
16) Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
17) “Detect Autism from a facial image,“ Kaggle, 2020년 4월 19일 수정, 2020년 12월 14일 접속, https://www.kaggle.com/gpiosenka/autistic-children-data-set-traintestvalidate/version/5.
18) “Google Colab,“ Google Colab, n.d. 수정, 2020년 12월 15일 접속, https://colab.research.google.com/.
19) “tf.keras.applications.ResNet50V2,“ TensorFlow, 2020년 12월 14일 수정, 2020년 12월 20일 접속, https://www.tensorflow.org/api_docs/python/tf/keras/applications/ResNet50V2.
20) “tf.keras.applications.DenseNet201,“ TensorFlow, 2020년 12월 14일 수정, 2020년 12월 20일 접속, https://www.tensorflow.org/api_docs/python/tf/keras/applications/DenseNet201.
21) Lin, M., Chen, Q., & Yan, S. (2013). Network in network. arXiv preprint arXiv:1312.4400.
22) Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.