Section 4: Deep Learning Project

심층 신경망을 이용한

질병 유무 탐지

AI 07 이기한

순서

배경 및 목적

프로젝트 배경

- '환부의 상태에 따라 어떤 질병인지 자동 분류할 수 있지 않을까?'

프로젝트 목적

- 비전문 인력의 질병 유무 탐색, 의료진의 피로도 감소
- 진단 의사 및 장비의 상태에 따른 정확도 감소 방지

데이터셋 선정 이유

- 비교적 환부의 유무가 명확한 흉부 X-ray 데이터셋 활용
- 현재 상황을 고려해 Covid-19 감염 데이터 제거

이미지 분류

모델 선정 이유(ResNet)

- 신경망의 깊이가 깊어질수록 여러 장점이 있지만 '기울기 소실/폭발'의 문제점이 발생하여 Error 값이 높아짐
- 입력값 x를 몇 layer 이후의 출력값에 더해주는 skip/shortcut connection으로 이를 해결
- bottleneck design으로 신경망의 복잡도를 감소시켜 연산량을 줄임
- 04. 한계에서 추가설명

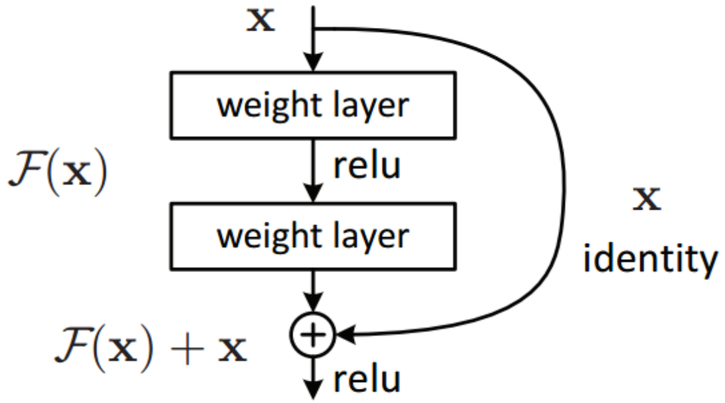


Figure 2. Residual learning: a building block.

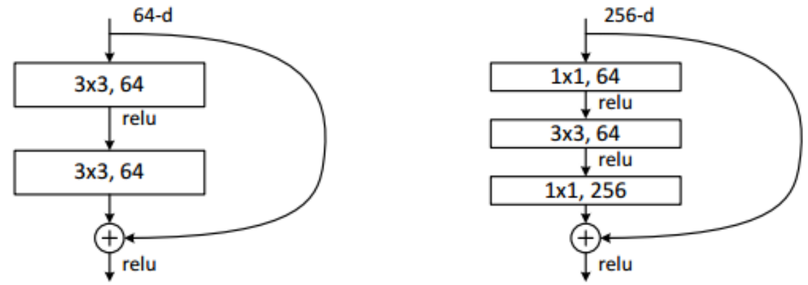


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

이미지 분류

전처리(Image Agmentation)

```
# Image preprocessing

train_data_gen = ImageDataGenerator(rescale=1./255,
                                    rotation_range=20,
                                    shear_range=0.1,
                                    horizontal_flip=True,
                                    width_shift_range=0.1,
                                    height_shift_range=0.1,
                                    brightness_range=[.2, .2])

train_generator = train_data_gen.flow_from_dataframe(train,
                                                     x_col='filename',
                                                     y_col='label',
                                                     target_size=(256, 256),
                                                     class_mode='categorical',
                                                     batch_size=16)

valid_data_gen = ImageDataGenerator(rescale=1./255)
                                    # rotation_range=20,
                                    # shear_range=0.1,
                                    # horizontal_flip=True,
                                    # width_shift_range=0.1,
                                    # height_shift_range=0.1,
                                    # brightness_range=[.2, .2])

valid_generator = valid_data_gen.flow_from_dataframe(valid,
                                                     x_col='filename',
                                                     y_col='label',
                                                     target_size=(256, 256),
                                                     class_mode='categorical',
                                                     batch_size=16)
```

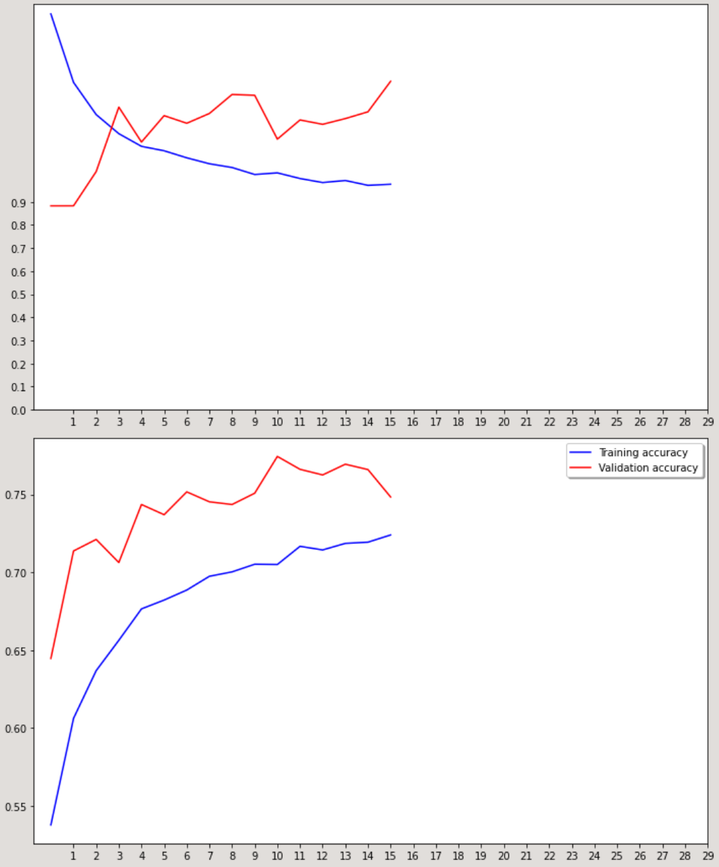
Fully Connected Layer

```
base_model = ResNet50(input_shape=(256, 256, 3), weights=None, include_top=False)

for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = AveragePooling2D()(x)
x = Flatten()(x)
x = Dense(128, activation="relu")(x)
x = Dropout(0.3)(x)
x = Dense(3, activation='softmax')(x)

model = Model(inputs = base_model.input, outputs = x)
```

이미지 분류

학습 결과

```
class_weight = {0: 10192/10192, 1: 10192/6012, 2: 10192/1345}

Epoch 1/30
728/728 [==============================] - 241s 314ms/step - loss: 1.7141 - accuracy: 0.5377 - val_loss: 0.8831 - val_accuracy: 0.6447
Epoch 2/30
728/728 [==============================] - 219s 301ms/step - loss: 1.4179 - accuracy: 0.6062 - val_loss: 0.8833 - val_accuracy: 0.7137
Epoch 3/30
728/728 [==============================] - 226s 311ms/step - loss: 1.2784 - accuracy: 0.6368 - val_loss: 1.0315 - val_accuracy: 0.7212
Epoch 4/30
728/728 [==============================] - 219s 300ms/step - loss: 1.1957 - accuracy: 0.6563 - val_loss: 1.3107 - val_accuracy: 0.7063
Epoch 5/30
728/728 [==============================] - 219s 301ms/step - loss: 1.1408 - accuracy: 0.6765 - val_loss: 1.1594 - val_accuracy: 0.7436
Epoch 6/30
728/728 [==============================] - 219s 300ms/step - loss: 1.1216 - accuracy: 0.6822 - val_loss: 1.2737 - val_accuracy: 0.7371
Epoch 7/30
728/728 [==============================] - 220s 302ms/step - loss: 1.0911 - accuracy: 0.6886 - val_loss: 1.2407 - val_accuracy: 0.7517
Epoch 8/30
728/728 [==============================] - 219s 301ms/step - loss: 1.0653 - accuracy: 0.6975 - val_loss: 1.2832 - val_accuracy: 0.7453
Epoch 9/30
728/728 [==============================] - 220s 302ms/step - loss: 1.0491 - accuracy: 0.7003 - val_loss: 1.3655 - val_accuracy: 0.7436
Epoch 10/30
728/728 [==============================] - 220s 302ms/step - loss: 1.0188 - accuracy: 0.7052 - val_loss: 1.3619 - val_accuracy: 0.7509
Epoch 11/30
728/728 [==============================] - 219s 301ms/step - loss: 1.0260 - accuracy: 0.7050 - val_loss: 1.1720 - val_accuracy: 0.7745
Epoch 12/30
728/728 [==============================] - 219s 301ms/step - loss: 1.0015 - accuracy: 0.7167 - val_loss: 1.2550 - val_accuracy: 0.7662
Epoch 13/30
728/728 [==============================] - 219s 301ms/step - loss: 0.9840 - accuracy: 0.7144 - val_loss: 1.2362 - val_accuracy: 0.7626
Epoch 14/30
728/728 [==============================] - 220s 302ms/step - loss: 0.9929 - accuracy: 0.7186 - val_loss: 1.2610 - val_accuracy: 0.7695
Epoch 15/30
728/728 [==============================] - 219s 301ms/step - loss: 0.9720 - accuracy: 0.7194 - val_loss: 1.2903 - val_accuracy: 0.7661
Epoch 16/30
728/728 [==============================] - 219s 301ms/step - loss: 0.9766 - accuracy: 0.7240 - val_loss: 1.4224 - val_accuracy: 0.7484
Epoch 16: early stopping
```



- train accuracy : 0.7050
- validation accuracy : 0.7745
- 문제 해결을 위해 낮은 accuracy

이상치 탐지

모델 선정 이유(AnoGAN)

- 의료 데이터 특성상 anormal한 데이터를 확보하기 어려움
- anormal한 데이터를 확보해도 labeling에 굉장한 인력, 시간, 자금이 소요
- 비지도 학습(Unsupervised learning) 필요

모델 학습 과정

1. normal data만 DCGAN을 통해 학습시킴
2. Pz에서 z 값을 random samplig
3. z1 계수에서 normal data와 유사하게 나오도록 업데이트 지속
4. normal data의 latent space로 적절하게 mapping되어 있는지 확인하여 normal/abnormal인지 판단

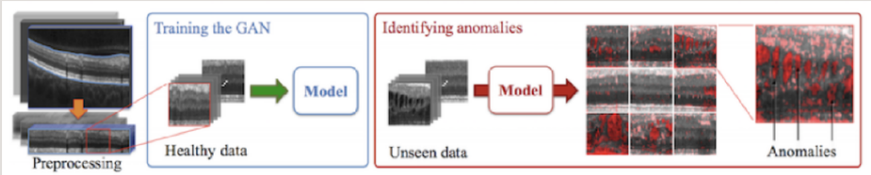


Fig. 1. Anomaly detection framework. The preprocessing step includes extraction and flattening of the retinal area, patch extraction and intensity normalization. Generative adversarial training is performed on healthy data and testing is performed on both, unseen healthy cases and anomalous data.

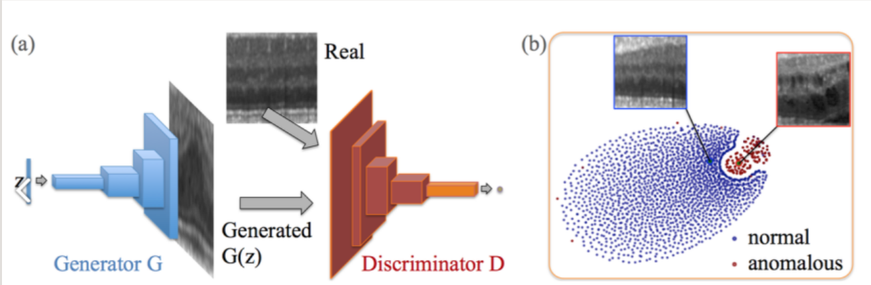
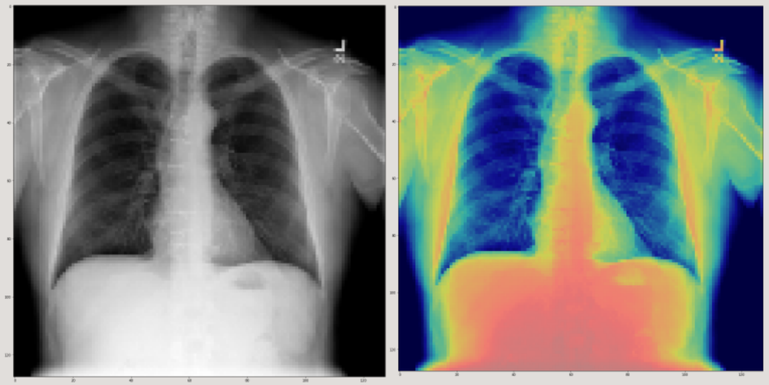


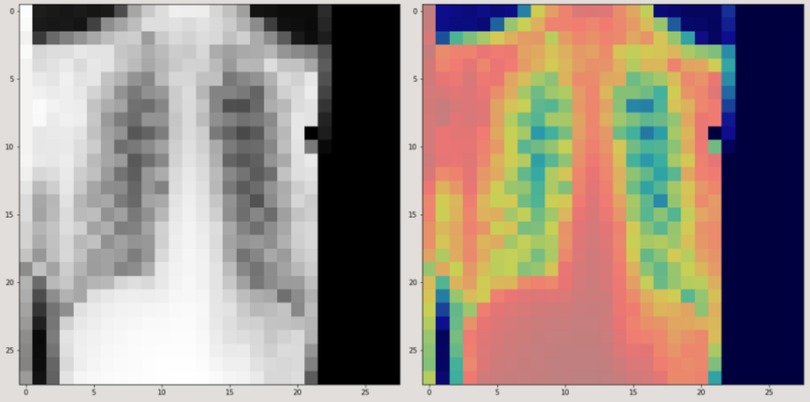
Fig. 2. (a) Deep convolutional generative adversarial network. (b) t-SNE embedding of normal (blue) and anomalous (red) images on the feature representation of the last convolution layer (orange in (a)) of the discriminator.

이상치 탐지

학습 결과



anomaly score : 12482.14



anomaly score : 1061.78

프로젝트의 한계점 및 향후 보완점

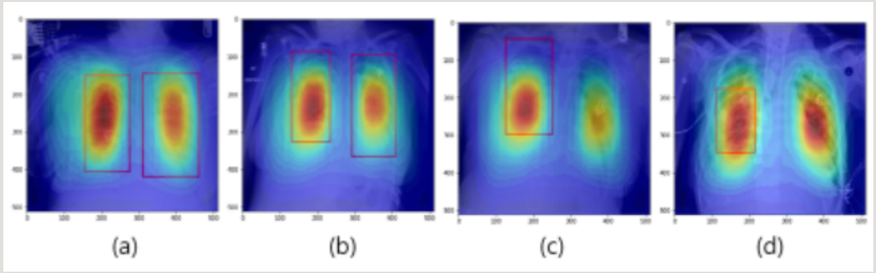
01 ResNet을 이용한 이미지 분류 모델이 데이터의 특성에 부적합한 정확도를 도출

02 AnoGAN 학습 결과 매우 높은 anormaly score 도출

03 CLAHE(Contrast Limited Adaptive Histogram equalization)와 같은 histogram equalization를 통해 학습 데이터 강화

04 Image Segmentation(U-Net) 활용

05 RetinaNet을 이용한 Object Detection으로 질병의 정밀 탐지



참고 자료

모델 학습 데이터셋

- https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

Unsupervised Anormaly Detection with Generative Adversarial Networks to Guide Marker Discovery

- https://arxiv.org/abs/1703.05921

[AnoGAN] Unsupervised Anormaly Detection with GAN 정리

- https://sensibilityit.tistory.com/506

AnoGAN 참고 Github repo

- https://github.com/yjucho1/anoGAN

RetinaNet Object Detection 참고 자료

- https://yhu0409.tistory.com/category/1.%20%EB%AA%A8%EB%8D%B8%20%EB%B6%84%EC%84%9D/Object%20Detection

감 사 합 니 다 .

AI 07 이기한