Pneumonia Detection Using Deep Learning

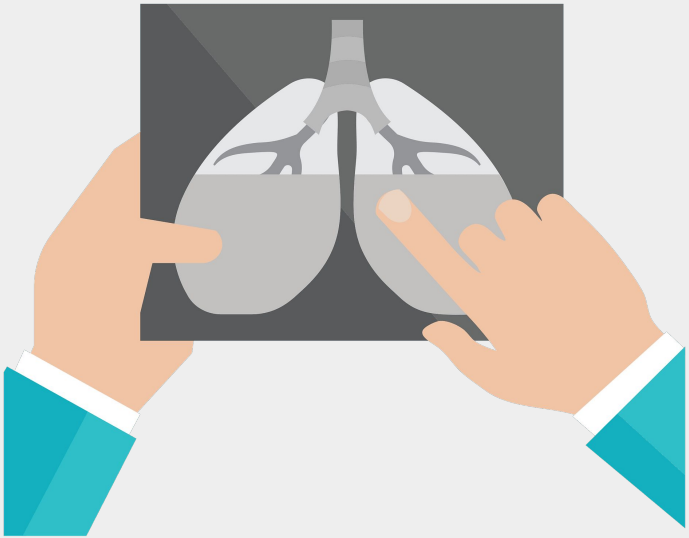
이재영, 지우근, 주한솔

목차

1. 프로젝트 주제 소개
2. 데이터 준비 과정
3. 데이터 시각화
4. 모델 생성
5. 모델훈련 및 성능검증
6. 모델성능 올리기

프로젝트 주제 소개

프로젝트 주제 소개: X-Ray 이미지를 통한 폐렴 진단



데이터 세트는 3개의 폴더(train, test, val)로 구성

5,863개의 X-Ray 이미지(JPEG)와 2개의 카테고리(폐렴/정상)

광저우 여성 아동 의료 센터의 1~5세 소아 환자의 데이터

두 명의 전문 의사가 등급을 매김

채점 오류를 설명하기 위해 세 번째 전문가도 평가 세트를 확인

X-Ray 이미지를 통한 폐렴 정보 설명

정상 흉부 X-Ray(왼쪽 패널)는 영상에서 비정상적인 혼탁이 없는 깨끗한 폐를 보여줍니다.

세균성 폐렴(가운데)은 전형적으로 국소적 엽성 경화를 나타내며, 이 경우 오른쪽 상엽(흰색 화살표)에 있는 반면,

바이러스성 폐렴(오른쪽)은 양쪽 폐에서 보다 확산된 “간질”패턴으로 나타냅니다.

데이터 시각화

데이터 시각화 - 데이터 개수

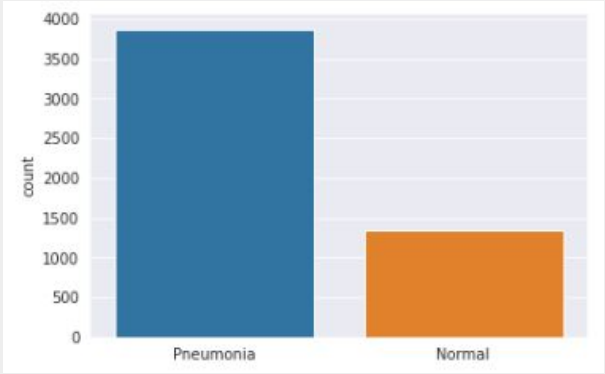
```
train = get_training_data('../input/chest-xray-pneumonia/chest_xray/chest_xray/train')
test = get_training_data('../input/chest-xray-pneumonia/chest_xray/chest_xray/test)
val = get_training_data('../input/chest-xray-pneumonia/chest_xray/chest_xray/val)
```

```
pirint('train:', len(train), '/', 'test:', len(test), '/', 'val:', len(val), '/', 'sum:', len(train) + len(test) + len(val)
```

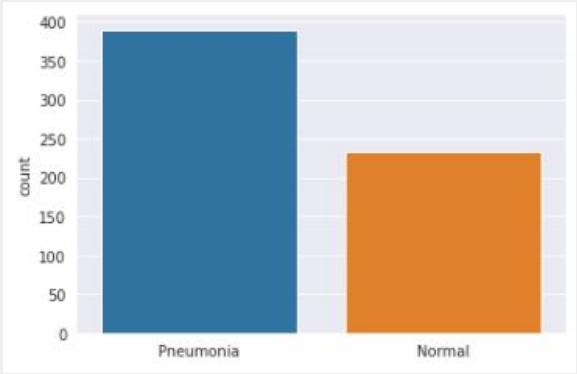
```
train: 5216 / test: 624 / val: 16 / sum: 5856
```

데이터 시각화 - count plot

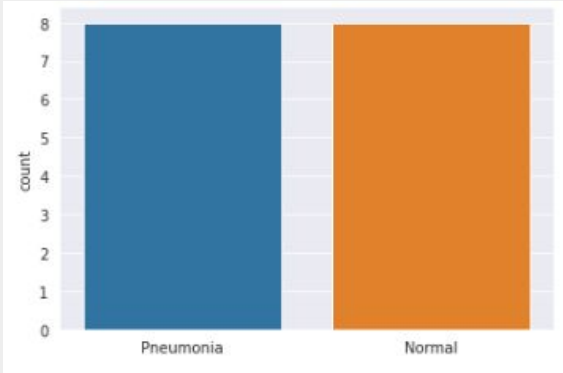
```
l = []
for i in train:
        if(i[l] == 0):
                l.append(“Pneumonia”)
        else:
                l.append(“Normal”)
sns.set_style(‘darkgrid’)
sns.countplot(l)
```




```
ll = []
for i in test:
        if(i[l] == 0):
                ll.append(“Pneumonia”)
        else:
                ll.append(“Normal”)
sns.set_style(‘darkgrid’)
sns.countplot(ll)
```




```
lll = []
for i in val:
        if(i[lll] == 0):
                lll.append(“Pneumonia”)
        else:
                lll.append(“Normal”)
sns.set_style(‘darkgrid’)
sns.countplot(lll)
```




```
print(l.count(“Pneumonia”), l.count(“Normal”))
```

```
3875 1341
```

```
print(ll.count(“Pneumonia”), ll.count(“Normal”))
```

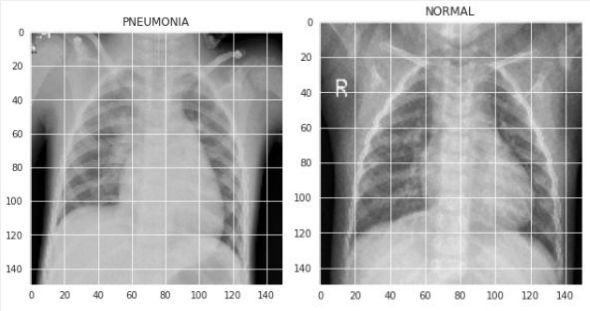
```
390 234
```

```
print(lll.count(“Pneumonia”), lll.count(“Normal”))
```

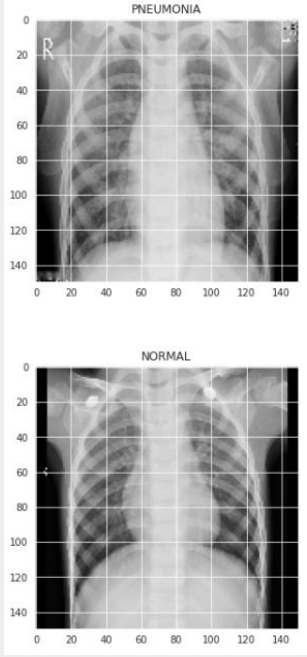
```
8 8
```

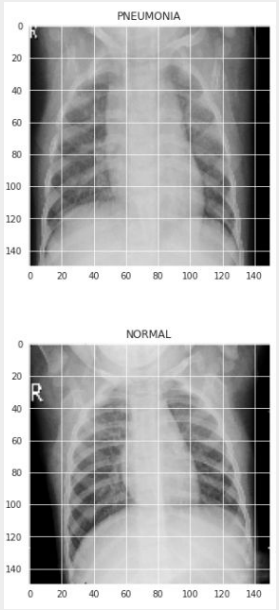
데이터 시각화

```
plt.figure(figsize = (5, 5))
plt.imshow(train[500][0],
cmap='gray')
plt.title(labels[train[500][1]])

plt.figure(figsize = (5, 5))
plt.imshow(train[-2][0],
cmap='gray')
plt.title(labels[train[-2]]1]])
```




```
plt.figure(figsize = (5, 5))
plt.imshow(test[50][0],
cmap='gray')
plt.title(labels[test[50][1]])

plt.figure(figsize = (5, 5))
plt.imshow(test[-2][0],
cmap='gray')
plt.title(labels[test[-2]]1]])
```






```
plt.figure(figsize = (5, 5))
plt.imshow(val[0][0],
cmap='gray')
plt.title(labels[val[0][1]])

plt.figure(figsize = (5, 5))
plt.imshow(val[-2][0],
cmap='gray')
plt.title(labels[val[-2]]1]])
```

데이터 전처리

데이터 전처리 - 정규화, Augmentation

```
# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255
x_test = np.array(x_test) / 255
```

```
# With data augmentation to prevent overfitting and handing the imbalance in dataset

datagen = ImageDataGenerator(rocation_range = 30,    # randomly rotate images in the range(degrees, 0 to 180)
                             zoom_range = 0.2,    # Randomly zoom image
                             width_shift_range = 0.1,    # randomly shift images horizontally (fraction of total width)
                             height_shift_range = 0.1,   # randomly shift images vertically (fraction of total height)
                             horizontal_flip = True,   # randomly flip images
                             vertical_flip = True)   # randomly flip images

datagen.fit(x_train)
```

모델 생성

모델 생성 - VGGNet, GoogleNet, Resnet

VGGNet

```
20/20 [==============================] - 1s 35ms/step - loss: 0.2997 - accuracy: 0.9119
Loss of the model is -  0.2997097671031952
20/20 [==============================] - 1s 33ms/step - loss: 0.2997 - accuracy: 0.9119
Accuracy of the model is -  91.18589758872986 %
```

GoogleNet

```
20/20 [==============================] - 1s 16ms/step - loss: 0.3029 - accuracy: 0.8798
Loss of the model is -  0.30293580889701843
20/20 [==============================] - 0s 16ms/step - loss: 0.3029 - accuracy: 0.8798
Accuracy of the model is -  87.9807710647583 %
```

Resnet

```
20/20 [==============================] - 62s 35ms/step - loss: 0.3466 - accuracy: 0.9054
Loss of the model is -  0.346641480922699
20/20 [==============================] - 1s 36ms/step - loss: 0.3466 - accuracy: 0.9054
Accuracy of the model is -  90.54487347602844 %
```

Resnet50 전이학습

```
base_model = ResNet50(weights=None, include_top=False, input_shape = (150,150,1))
base_model.summary()

model = Sequential()
model.add(base_model)
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer = 'adam' , loss = 'binary_crossentropy' , metrics = ['accuracy'])
```

실행결과

```
624/624 [==============================] - 1s 2ms/step
Loss of the model is -  0.3014889302162024
624/624 [==============================] - 1s 2ms/step
Accuracy of the model is -  90.06410241127014 %
```

캐글 성능 1위 모델

```
model = Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu' , input_shape = (150,150,1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(Flatten())
model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 1 , activation = 'sigmoid'))
model.compile(optimizer = "rmsprop" , loss = 'binary_crossentropy' , metrics = ['accuracy'])
model.summary()
```

모델 성능 개선

파라미터 조정 - 캐글 성능 1위 모델

rotation_range = 30 -> 60, vertical_flip = False -> True

vertical_flip = False -> True

zoom_range = 0.2 -> 0.1, width_shift_range = 0.1 -> 0.2, height_shift_range = 0.1 -> 0.2

vertical_flip = False -> True, model optimizer = rmsprop -> Adam

vertical_flip = False -> True, epochs = 12 -> 15

epochs = 25

epochs = 35

epochs = 30

랜덤 시드를 고정. Augmentation 과정은 어쩔 수 없음.

dropout (0.2, 0.2, 0.2 -> 0.1, 0.2, 0.3) 가중치의 개수에 따라 바꿈.

- 데이터 로그 변환 (clear)
- loss 함수 설정 -> Binary Cross Entropy, Categorical Cross Entropy, Sparse Categorical Cross

파라미터 조정 - epoch, learning rate 조절

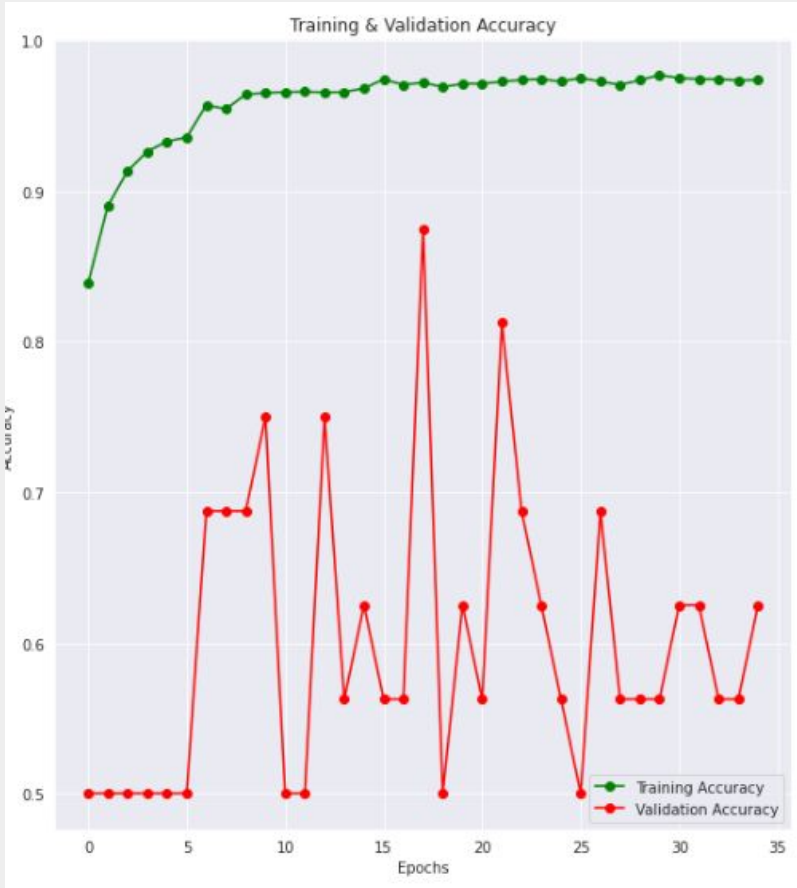
```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', patience = 4, verbose=1, factor=0.2, min_lr=0.00000001)
```

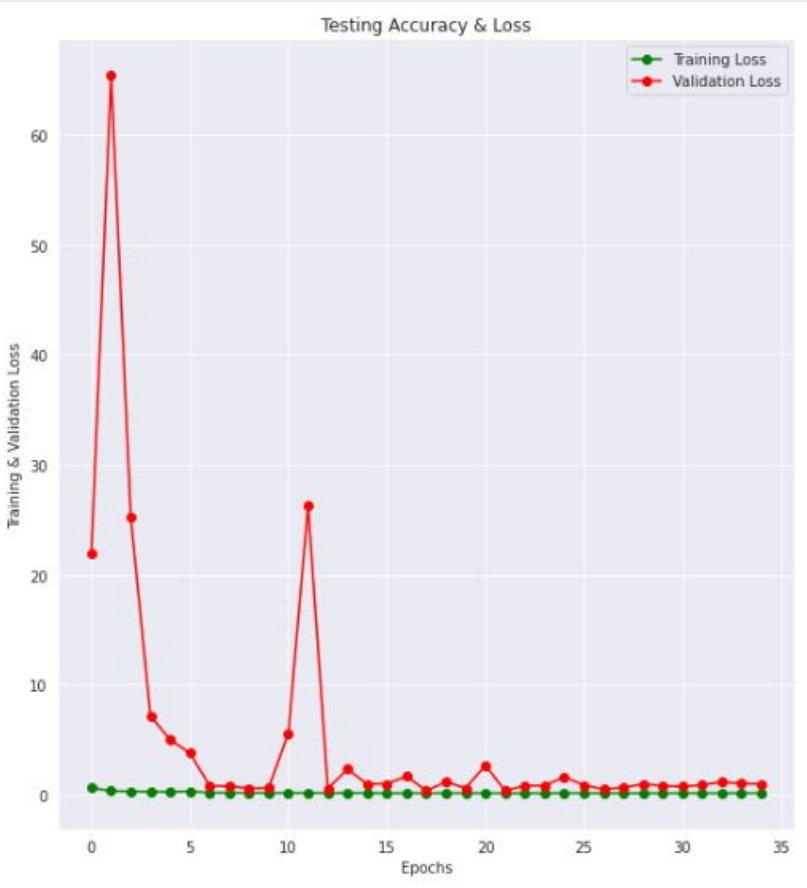
```
history = model.fit(datagen.flow(x_train,y_train, batch_size = 32) ,epochs = 35 , validation_data = datagen.flow(x_val, y_val) ,callbacks = [learning_rate_reduction])
```

```
print("Loss of the model is - " , model.evaluate(x_test,y_test)[0])
print("Accuracy of the model is - " , model.evaluate(x_test,y_test)[1]*100 , "%")
```

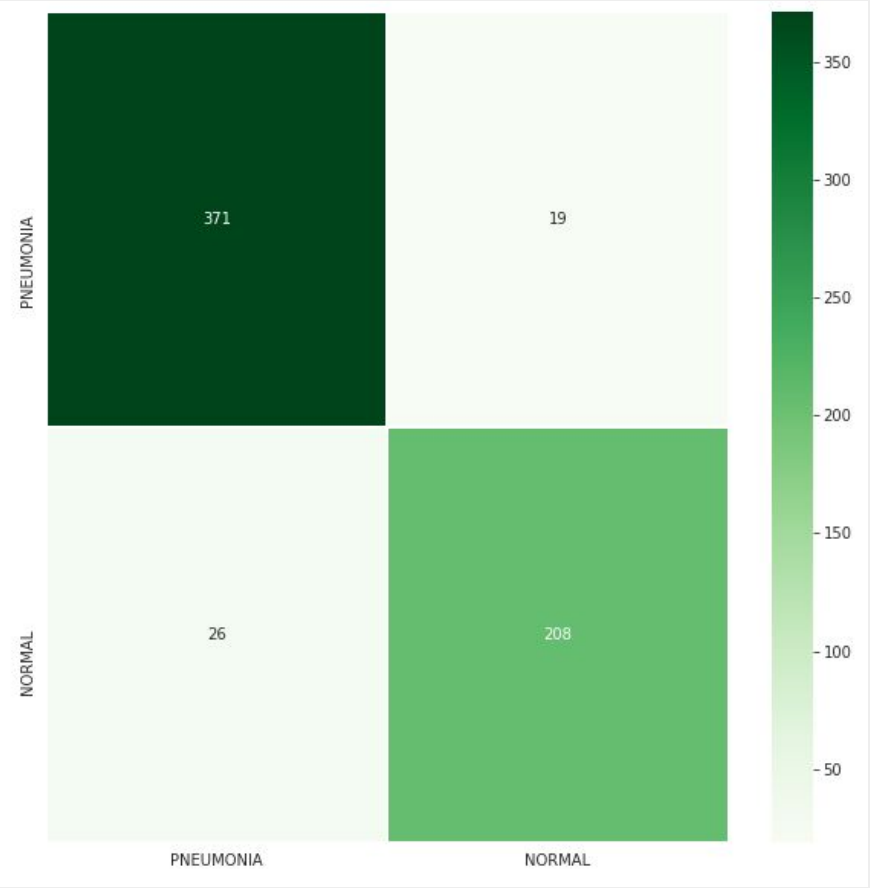
```
624/624 [==============================] - 0s 458us/step
Loss of the model is -  0.2585679814219475
624/624 [==============================] - 0s 450us/step
Accuracy of the model is -  92.78846383094788 %
```

모델 훈련 결과 시각화


Training & Validation Accuracy
Accuracy
Epochs
Training Accuracy
Validation Accuracy


Testing Accuracy & Loss
Training & Validation Loss
Epochs
Training Loss
Validation Loss

모델 훈련 결과 시각화



	precision	recall	f1-score	support
Pneumonia (Class 0)	0.93	0.95	0.94	390
Normal (Class 1)	0.92	0.89	0.90	234
accuracy			0.93	624
macro avg	0.93	0.92	0.92	624
weighted avg	0.93	0.93	0.93	624

Thank you