마스크 착용 여부 분류를 위한 ResNet-18 모델 구축

빅데이터 6기
이신영

목차

1. 서론

1.1 연구 배경 및 목적

사흘간 이미지 처리 이론을 집중적으로 학습하였다. 본 연구에서는 배운 내용을 바탕으로 OpenCV를 활용한 이미지 전처리와 CNN 모델을 활용한 이미지 분류, 하이퍼파라미터 튜닝을 통한 모델 성능 최적화를 진행한다. 본 연구의 핵심 목표는 다양한 실험 시도와 최적의 결과 도출이다.

2. 데이터셋

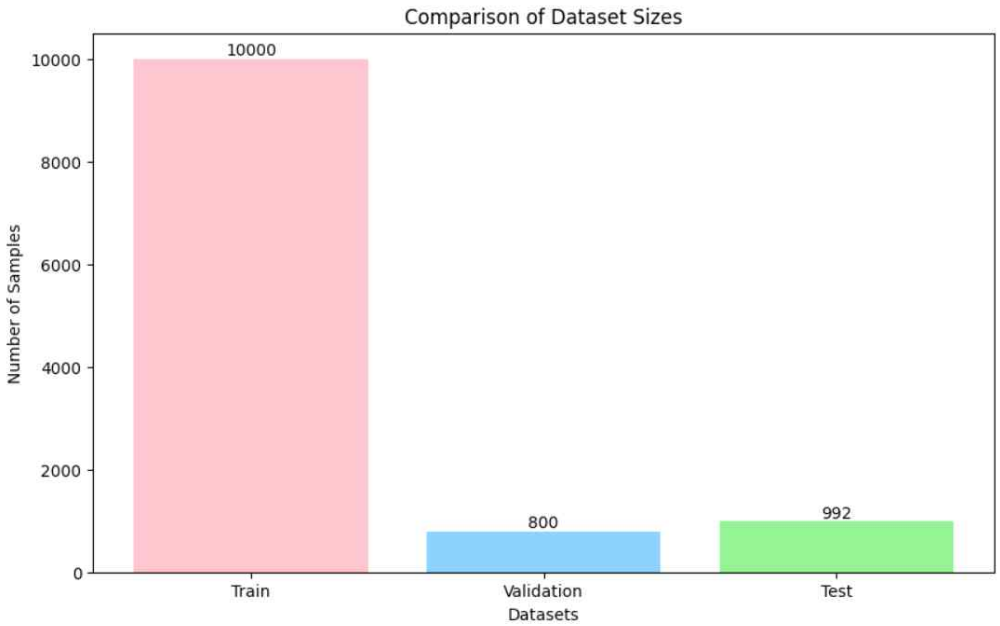
2.1 데이터셋 소개

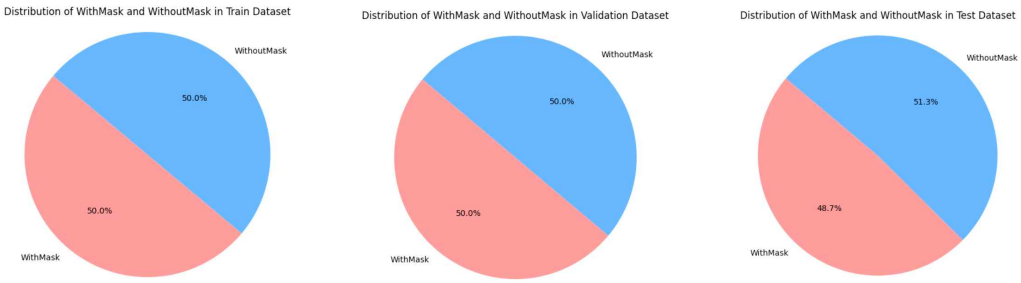
① 데이터셋 이름: Face Mask Detection ~12K Images Dataset

② 크기: 약 328.92MB

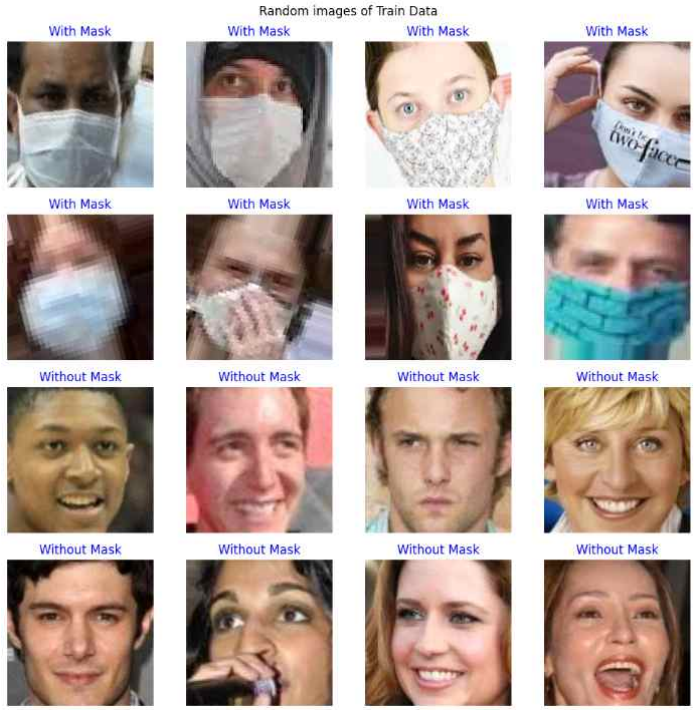
③ 데이터 구조

전체 데이터 (11,792개)	Train 데이터 (10,000개)	WithMask(5,000개)
		WithoutMask(5,000개)
	Validation 데이터 (800개)	WithMask(400개)
		WithoutMask(400개)
	Test 데이터 (992개)	WithMask(483개)
		WithoutMask(509개)





④ 데이터 예시



2.2 데이터 전처리

① Resize : CNN 모델은 특징을 추출하기 위해 input data의 형태에 맞추어 설계하기 때문에 모델마다 그 모델에 적합한 input size로 전체 이미지 사이즈를 resize하였다.

② ToTensor : Numpy 형태의 이미지를 torch의 (C, H, W) tensor 형태로 바꿔준다. 픽셀의 range가 0~1 값으로 스케일링 된다.

③ Normalize : local optima 문제를 예방하고 속도를 향상하기 위하여 정규화하였다. resize한 이미지를 학습할 것이기 때문에 resize한 데이터의 평균과 표준편차를 이용하였다.

3. 모델 설계

ResNet-18, VGG-16, 직접 구성한 CNN 모델을 실험하였다. 그 결과, 현실성, 학습 속도, 성능을 고려하였을 때 ResNet-18이 가장 우수하여 최종 모델로 선택한 뒤 하이퍼파라미터 튜닝을 진행하였다. 다른 모델의 학습 결과는 부록에 첨부하였다.

3.1 ResNet-18 모델 소개

ResNet-18은 ResNet 시리즈 중 18개의 레이어로 구성되어 있는 모델이다. ResNet은 Skip Connection을 도입하여 네트워크에서 발생하는 기울기 소실 문제를 완화하고, 더 깊은 네트워크를 효과적으로 학습할 수 있도록 했다는 특징이 있다.

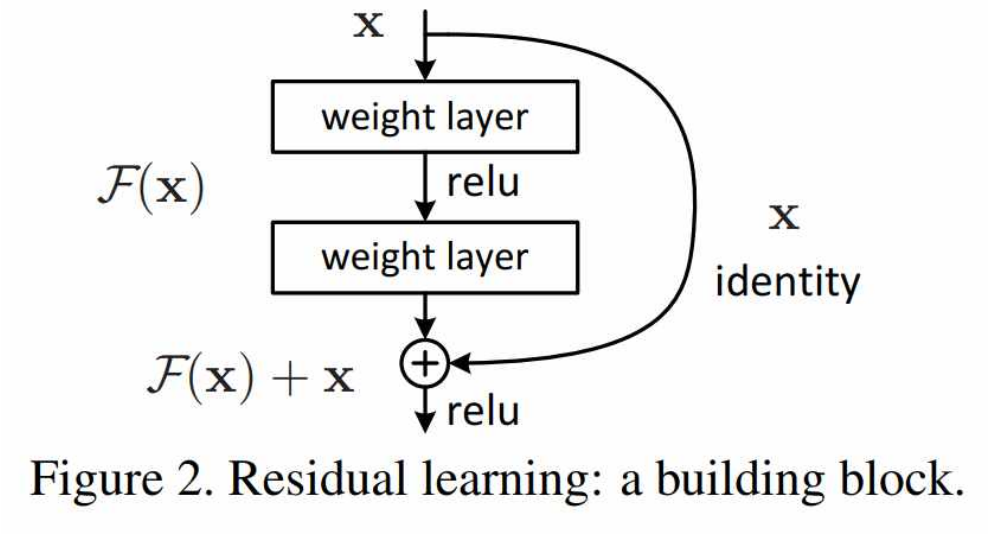


Figure 2. Residual learning: a building block.

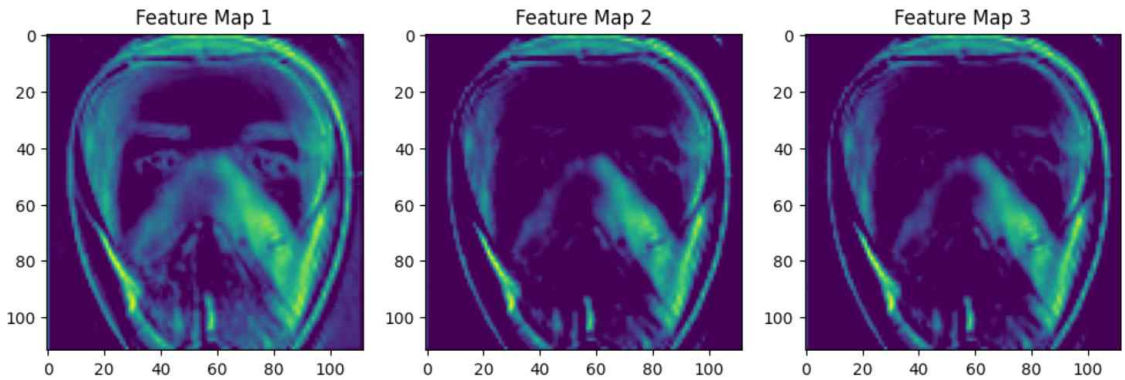
3.2 Resnet-18 모델 구현

① 이미지 리사이즈 : ResNet 모델은 ImageNet 데이터셋으로 사전 훈련된 모델들의 입력 크기와 일치하므로 입력 이미지의 크기는 주로 224X224 또는 229X229이다. 이 연구에서는 224X224로 리사이즈 하였다.

② 모델 : ResNet-18 모델을 불러와서 출력 레이어를 클래스 수 2에 맞게 조정

③ 손실 함수 : 이진 분류에 일반적으로 사용되는 크로스 엔트로피 손실 함수를 사용

④ 최적화 알고리즘 : SGD와 Adam을 실험한 뒤 결정

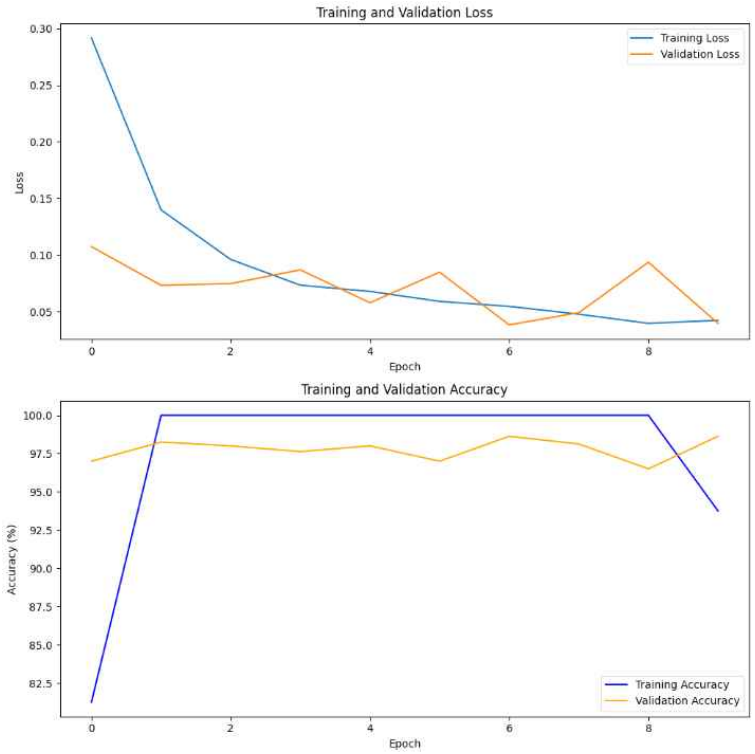


3.3 ResNet-18 모델 성능 평가 및 하이퍼파라미터 튜닝

본 연구에서는 최적화 알고리즘에 변화를 주어 하이퍼파라미터를 튜닝하였다.

3.3.1 optimizer = Adam

```
model = myResNet().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
```

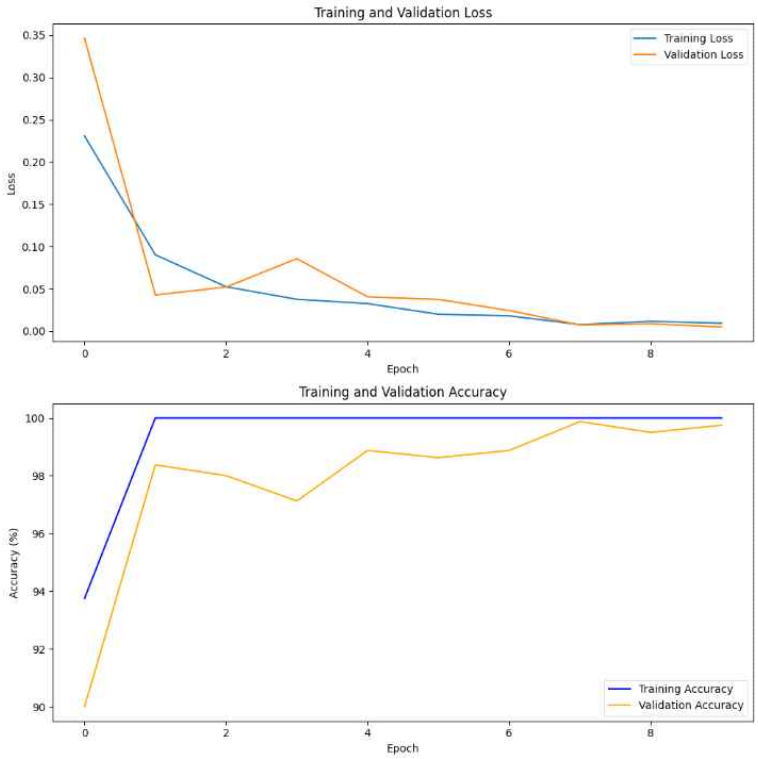



```
Epoch 1, Training loss: 0.2917
val loss: 0.1073, Accuracy: 97.00%
Epoch 2, Training loss: 0.1398
val loss: 0.0731, Accuracy: 98.25%
Epoch 3, Training loss: 0.0960
val loss: 0.0747, Accuracy: 98.00%
Epoch 4, Training loss: 0.0732
val loss: 0.0868, Accuracy: 97.62%
Epoch 5, Training loss: 0.0677
val loss: 0.0577, Accuracy: 98.00%
Epoch 6, Training loss: 0.0589
val loss: 0.0847, Accuracy: 97.00%
Epoch 7, Training loss: 0.0545
val loss: 0.0382, Accuracy: 98.62%
Epoch 8, Training loss: 0.0476
val loss: 0.0489, Accuracy: 98.12%
Epoch 9, Training loss: 0.0395
val loss: 0.0935, Accuracy: 96.50%
Epoch 10, Training loss: 0.0422
val loss: 0.0394, Accuracy: 98.62%
```

```
Test Loss: 0.050935367470130236
Test Accuracy: 0.9828629032258065
```

3.3.2 optimizer = SGD

```
model = myResNet().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01,momentum=0.9)
```




```
Epoch 1, Training loss: 0.2307
val loss: 0.3463, Accuracy: 90.00%
Epoch 2, Training loss: 0.0902
val loss: 0.0427, Accuracy: 98.38%
Epoch 3, Training loss: 0.0524
val loss: 0.0521, Accuracy: 98.00%
Epoch 4, Training loss: 0.0375
val loss: 0.0856, Accuracy: 97.12%
Epoch 5, Training loss: 0.0325
val loss: 0.0405, Accuracy: 98.88%
Epoch 6, Training loss: 0.0199
val loss: 0.0374, Accuracy: 98.62%
Epoch 7, Training loss: 0.0180
val loss: 0.0244, Accuracy: 98.88%
Epoch 8, Training loss: 0.0076
val loss: 0.0072, Accuracy: 99.88%
Epoch 9, Training loss: 0.0116
val loss: 0.0087, Accuracy: 99.50%
Epoch 10, Training loss: 0.0093
val loss: 0.0049, Accuracy: 99.75%
```

```
Test Loss: 0.008722839585032253
Test Accuracy: 0.9969758064516129
```

Training Loss와 Validation Loss를 비교한 결과 과적합이 일어나지 않았다고 판단하였다. 과적합 방지를 위해 dropout, L1/L2 정규화와 같은 처리를 시도하였으나 정확도가 떨어져서 배제하였다. 최종적으로 약 99.7%의 정확도를 보인 SGD 최적화 알고리즘 적용 ResNet-18을 모델로 선정하였다.

4. 결론

4.1 결과 요약

본 연구에서는 CNN 모델을 활용하여 마스크 착용/미착용 안면으로 이진 분류하는 작업을 수행하였다. 먼저 VGG-16, 직접 구성한 CNN 모델, ResNet-18 모델을 실험하였고, 성능과 속도 측면에서 우수한 ResNet-18 모델을 활용하기로 결정하였다.

모델 선정 이후 ResNet 모델에 맞추어 입력 이미지 크기를 224x224로, PyTorch의 ResNet-18 모델을 불러와서 출력 레이어를 2개의 클래스에 맞게 조정하였다. 손실 함수는 이진 분류에 적합한 크로스 엔트로피 손실 함수를 사용하였으며, 최적화 알고리즘은 SGD와 Adam을 실험하였다.

하이퍼파라미터 튜닝 실험 결과, Adam optimizer 적용 시 'Test Loss : 0.0509 / Test Accuracy : 약 98.29%', SGD optimizer 적용 시 'Test Loss : 0.0087 / Test Accuracy : 약 99.7%'의 결과가 도출되었다. 최종적으로 SGD optimizer를 적용한 ResNet-18 모델을 선택하였다. 과적합 방지를 위해 다양한 처리를 시도하였으나, 해당 데이터셋에서는 이러한 처리가 성능을 향상시키지 않았다.

4.2 느낀 점 및 향후 방향

CNN 모델 기반 이미지 처리 이론을 바탕으로 진행한 본 연구는 다양한 CNN 모델의 구조와 하이퍼파라미터 튜닝에 관한 지식 습득과 인사이트 도출의 기회가 되었다. 본 연구를 통해 먼저, 최적화 알고리즘이 성능에 미치는 영향을 확인하였다. 사전 조사 결과 SGD가 Adam보다 낮은 성능을 보인다는 실험 내용이 많았지만, 실제로 본 연구 결과에서는 SGD가 Adam보다 더 우수한 성능을 보였다. 이를 통해 하이퍼파라미터 튜닝에 정답은 없음을 경험하였고 실제 문제에 적합한 CNN 모델을 만들기 위해서는 많은 실험과 경험이 필요함을 알 수 있었다.

연구 과정에서는 실험 환경으로 인해 더 많은 실험을 해보지 못한 아쉬움이 있었다. 모델의 구조를 단순화하거나 CPU 대신 GPU를 사용하는 등의 방법을 활용하였음에도 모델 학습 시간이 많이 소요되어 다양한 CNN 모델이나 하이퍼파라미터 조정을 더 시도하지 못한 것이 본 연구의 제약이 되었다. 따라서 향후 머신러닝을 진행한다면 실험 환경을 고려하여 모델의 구조에 변화를 주는 데에도 신경을 써야겠다고 생각했다.

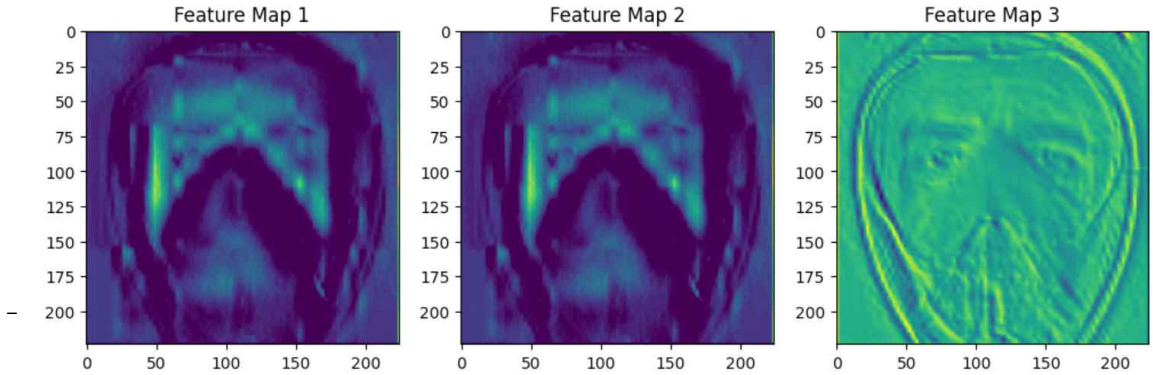
본 연구를 통해 많은 실험을 통해 모델의 성능을 개선하고, 다양한 하이퍼파라미터를 조정하여 최적의 설정을 찾는 것이 중요하다고 느꼈다. 또한, 실험 과정에서 발생하는 다양한 문제들을 해결하며 모델 개발에 대한 실무적인 경험을 쌓을 수 있었다. 앞으로는 실험을 더욱 철저히 계획하고 진행하여 모델의 성능을 높이고 실제 문제에 대한 효과적인 해결책을 찾아나가고자 한다.

부록

코드

https://github.com/2shin0/FaceMaskDetection-ResNet18/blob/main/README.md

VGG-16



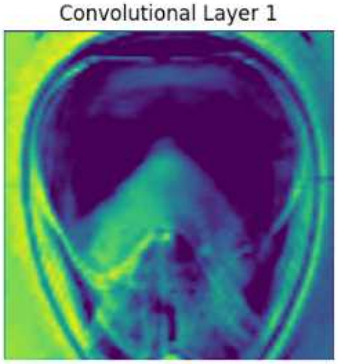

```
Epoch 1, Training loss: 871142677.6560
val loss: 0.6933, Accuracy: 50.00%
Epoch 2, Training loss: 0.6998
val loss: 0.6936, Accuracy: 50.00%
Epoch 3, Training loss: 0.6969
val loss: 0.6933, Accuracy: 50.00%
Epoch 4, Training loss: 0.6954
val loss: 0.6932, Accuracy: 50.00%
Epoch 5, Training loss: 0.6942
val loss: 0.6933, Accuracy: 50.00%
```

직접 만든 CNN 모델

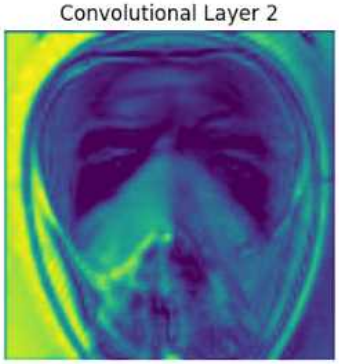
```python
class MyCNN(nn.Module):
    def __init__(self):
        super(MyCNN, self).__init__()
        # Convolutional layers
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3, stride=1, padding=1)
        self.conv3 = nn.Conv2d(in_channels=32, out_channels=32, kernel_size=3, stride=1, padding=1)
        # Max pooling layers
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        # Dropout layer
        self.dropout = nn.Dropout(0.5)
        # Fully connected layer
        self.fc = nn.Linear(32 * 4 * 4, 2)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = F.relu(self.conv3(x))
        x = self.pool(x)
        x = x.view(-1, 32 * 4 * 4)
        x = self.dropout(x)
        x = self.fc(x)
        return x
```

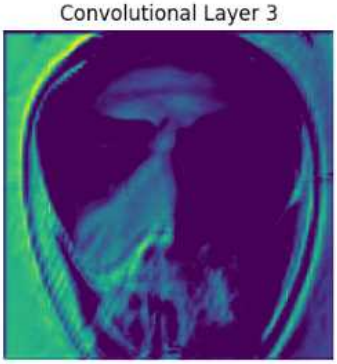
Convolutional Layer 1



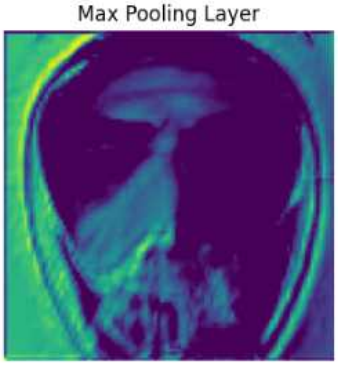
Convolutional Layer 2



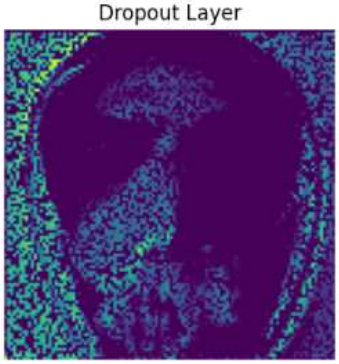
Convolutional Layer 3



Max Pooling Layer



Dropout Layer



Fully Connected Layer

```
Epoch 1, Training loss: 0.2665
val loss: 0.2006, Accuracy: 93.38%
Epoch 2, Training loss: 0.2072
val loss: 0.1017, Accuracy: 96.75%
Epoch 3, Training loss: 0.1881
val loss: 0.1543, Accuracy: 93.50%
Epoch 4, Training loss: 0.1706
val loss: 0.1449, Accuracy: 93.00%
Epoch 5, Training loss: 0.1741
val loss: 0.1584, Accuracy: 94.62%
Epoch 6, Training loss: 0.2099
val loss: 0.1593, Accuracy: 93.75%
Epoch 7, Training loss: 0.1817
val loss: 0.1125, Accuracy: 96.00%
Epoch 8, Training loss: 0.1790
val loss: 0.1265, Accuracy: 94.12%
Epoch 9, Training loss: 0.1827
val loss: 0.1081, Accuracy: 95.12%
Epoch 10, Training loss: 0.2176
val loss: 0.1406, Accuracy: 94.62%
```

참고문헌

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).

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the International Conference on Learning Representations (ICLR).