「2022-2 기계학습 Term-Project 보고서
: Brain Tumor classification by CNN Model with Python」

201902218 통계학과
오승훈

1. 문제 정의

1.1) 도메인 소개

인체를 관장하는 뇌에서 발생하는 모든 문제는 인간의 건강에 심대한 영향을 미친다. 이 중에서도 치명적인 질환들이 존재하는데 뇌종양(Brain Tumor, encephaloma)이 그 중 하나이다. 뇌종양은 뇌와 뇌를 둘러싸고 있는 뇌척수막에서 생기는 불필요한 세포 덩어리(종양)를 의미하며 크게 뇌 자체에서 발생하는 원발성 종양과 신체의 다른 부분으로부터 유래된 이차성 종양으로 분류할 수 있다. 원발성 종양 중 신경교종(Glioma)과 수막종(Meningioma)은 초기 단계에서 발견하지 못하면 사람을 사망에 이르게 할 수 있는 치명적인 유형의 뇌종양이다.

일반적인 뇌종양의 증상으로는 종양의 부피 때문에 뇌압이 올라가고, 뇌 조직이 압박되면서 두통, 발작(경련), 의식 변화, 시력 저하 등의 신체 기능에 문제가 발생한다. 뇌종양은 보통 CT, MRI, PET 등의 영상 검사로 뇌와 신경의 상태를 확인하여 발견하게 되는데 수술로 조직을 일부 떼어서 현미경으로 확인하는 조직 검사로 종양의 악성도를 확인하고, 치료 계획을 세우게 된다.

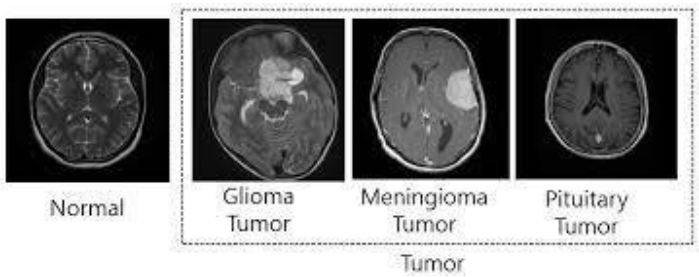
뇌종양은 특별한 예방법이 알려지지 않았고, 원인도 명확하지 않아서 위험 요인을 미리 알고 피하기도 어렵기 때문에 뇌종양을 조기에 진단하여 빠르게 치료를 시작하는 것이 가장 중요하다. 뇌 영상검사를 통해 얻은 데이터를 사람이 일일이 검토하는 것은 시간이 오래 걸리며 또한 부정확할 경우도 흔히 발생하기에 질적으로 우수한 많은 노동력을 요구한다. 그렇기에 딥러닝 기술을 활용하여 이미지 또는 영상 자료가 제공하는 데이터에서 의미 있는 특징 정보를 잘 추출할 수 있는 모델을 계속 만들어 발전시킨다면 이는 현재 의료 시스템을 도와 큰 효율과 예측 정확도 향상을 가져다주는 역할을 해낼 것으로 기대할 수 있다.

1.2) 데이터 둘러보기

데이터 셋은 다양한 각도에서 촬영한 뇌의 단면에 대한 MRI 이미지이다. 이미지의 크기는 동일하지 않으며 제각각이다. 총 7023개의 뇌 MRI 이미지 중 5,712개의 훈련 데이터 셋(Training)과 1311개의 테스트 데이터 셋(Testing)이 존재하며, 각 훈련 및 테스트 데이터 셋은 다시 4가지 클래스인 'no tumor(정상)', 'glioma tumor(신경교종 종양)', 'meningioma tumor(수막종 종양)', 'pituitary tumor(뇌하수체 종양)'으로 나뉜다. 각 클래스에 대한 데이터 개수를 정리하면 <표_1>과 같다.

<표_1>

클래스(class)	no tumor	glioma tumor	meningioma tumor	pituitary tumor
Training	1595	1321	1339	1457
Testing	405	300	306	300



1.3) 문제 해결을 위한 접근 방식

1.3.1) 데이터 분리

훈련 데이터 셋(Training)와 테스트 데이터 셋(Testing)은 이미 데이터가 잘 나누어져 있지만

검증 데이터 셋(Validation)은 따로 존재하지 않으므로 훈련 데이터 셋을 훈련:검증=8:2의 비율로 분리하여 검증 데이터 셋을 확보한다.

1.3.2) 데이터 전처리 개요

이미지 데이터에 대해서 더 안정적이고, 빠르게 학습이 될 수 있도록 파이썬 패키지를 이용하여 전처리를 진행한다. 데이터의 수는 해당 학부 수준의 프로젝트를 진행하기 위해서는 충분하다고 판단하여 'Data augmentation'은 별도로 진행하지 않았다. Image_Generator를 통해서 train, validation, test에 대한 전처리된 이미지를 생성할 것이다.

1.3.3) 모델 구축 및 평가 개요

일반적인 CNN 모델, VGGNet, ResNet을 활용한 모델을 구축한 후 그들의 성능을 비교 평가하기로 한다. 일반적인 CNN 모델을 만들 때는 컨볼루션 층의 개수, 커널(필터)의 크기와 개수 등이 모델 성능에 미치는 영향에 대한 논문을 참고하여 모델을 구축한다. VGGNet, ResNet은 현재 term-project를 통해 구현하려는 모델보다 훨씬 복잡한 이미지 데이터에 대한 분류를 진행하기 위해 개발된 모델이기에 비교적 단순한 데이터에 해당 모델(VGGNet과 ResNet)을 그대로 적용하게되면 층이 깊어지며 gradient가 소멸해버릴 가능성이나 모델이 수렴하지 않을 가능성이 높아질 뿐만 아니라 학습 시간이 매우 오래 걸릴 것으로 예상되었다. 따라서 VGGNet과 ResNet의 복잡도(층 개수, 일부 구조 등)는 줄인 대신 각 모델의 핵심 원리만 차용하는 'Mini-VGGNet', 'Mini-ResNet' 모델을 자체적으로 구축하도록 한다. 3가지 CNN 모델에 대한 성능 평가는 분류 문제에서 대표적인 평가 지표로 사용되는 혼동행렬(confusion matrix)기반의 'accuracy'를 사용한다.

1.3.4) 성능 튜닝 개요

아키텍처를 수정하거나 각 층의 하이퍼파라미터(필터(커널)의 크기, 개수, 보폭(stride))와 가중치 업데이트 최적화 알고리즘 등의 요소를 조절하여 가장 좋은 성능을 내는 모델을 찾는다.

1.4) 분석 환경 소개

1.4.1) 개발 환경

Jupyter Notebook은 오픈소스 (Open source) 기반의 웹 플랫폼으로, 파이썬을 비롯한 다양한 프로그래밍 언어로 코드 작성 및 실행하는 개발 환경이다. Jupyter Notebook을 사용한 이유는 웹브라우저상에서 셀(cell, chunk) 단위로 작성하여 실행할 수 있기에 큰 파이썬 파일도 셀 단위로 나누어 코드를 단계적으로 쉽게 실행하고, 시각적으로 빠르게 확인해볼 수 있다는 점 때문이다. 뿐만 아니라, 프로젝트의 결과를 'Github'에 업로드해야 하는데 이때 주피터 노트북의 결과 출력 방식 그대로를 Github에 올릴 수 있다는 장점도 있다.

1.4.2) 시스템(개인 컴퓨터 스펙) 환경 <표_2>

Element	specification
CPU	Intel(R) Core(TM) i5-8265U 1.80 GHz
Memory	RAM 8GB
GPU	Intel(R) UHD Graphics 620
Python version	3.11.1

2. 데이터 분석

2.1) 데이터 전처리(Preprocessing)

2.1.1) 이미지 데이터 정규화(normalization)

배치정규화(normalization)이 적용된 학습데이터의 problem space가 더 안정적으로 형성될 수 있기에 학습이 더 수월하게 진행된다는 장점이 있다. 일반적으로 모든 데이터들의 평균을 구한 후, 각 데이터에서 평균값을 빼준다. 데이터 분포의 중심을 0으로 옮기는 작업인 'Zero center'를 진행한 후 normalization을 적용하는데 이때 모든 데이터의 표준편차를 구한 후 이 값으로 각 데이터를 나눠준다. 위의 방식은 개별 이미지에 대해서가 아닌 batch 단위로 묶인 이미지들을 기반으로 하여 해당 평균 및 표준편차를 구한 뒤, normalization을 적용할 것이다.

2.2.1) '5-Conv + 2-FC CNN' (이하 일반 CNN) 모델 구축 및 모델 평가, 성능 튜닝

아래 그림_1과 같은 모델을 구축했다. CNN의 구성요소인 컨볼루션 층의 개수, 커널의 크기 및 개수 등의 차이에 따라 모델의 정확도와 학습 시간의 차이가 발생하므로 이를 고려하여 CNN을 구축해야 한다. 그림_2와 같이 컨볼루션 연산, 배치 정규화, 활성함수 적용, 풀링 과정으로 이루어진 특징 추출(feature extraction) 층을 컨볼루션 층(convolution layer)이라 설정하고, 이러한 컨볼루션 층을 5개를 거친 뒤에 해당 출력값을 1차원 배열로 flatten한 뒤 완전 연결층(Fully Connected layer)에 넣어서 class별로 분류하는 단계를 거친다. 이때 완전 연결층은 2개로 구성한다. (모든 과정이 나와있는 그림_3) 학습 과정에서 사용하는 손실함수는 다항분류일 때 사용하는 'categorical cross entropy'를 사용하였고, 신경망의 가중치 업데이트 방식을 결정하는 옵티마이저는 Adam을 사용한다.

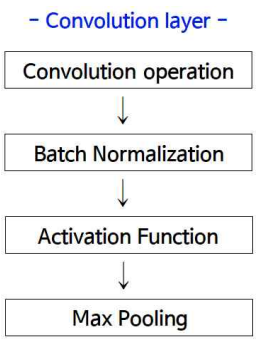


그림_1

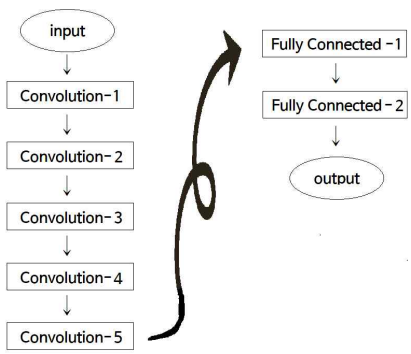


그림_2

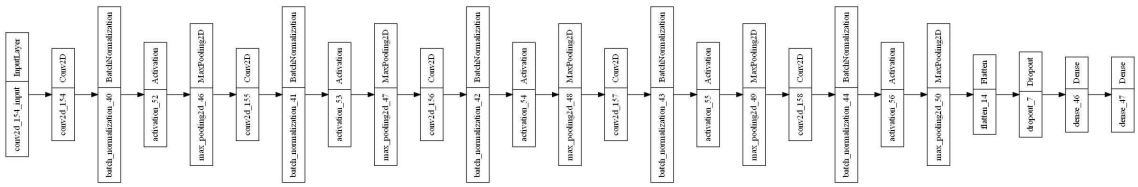


그림_3

▶ 일반 CNN 모델 평가

모델의 평가에 사용한 평가 척도는 'Accuracy'이다. 처음 구축한 모델의 Accuracy 값은 약 84%이다. training 과정에서 생성해놓은 validation data를 통해 먼저 중간 평가를 거친 후(그림_4) test data를 활용해 모델 평가를 진행했다. (그림_5) 각 과정에서 진행되는 누적 epoch 수에 따른 Accuracy와 Loss function 값을 시각적으로 나타내었다.

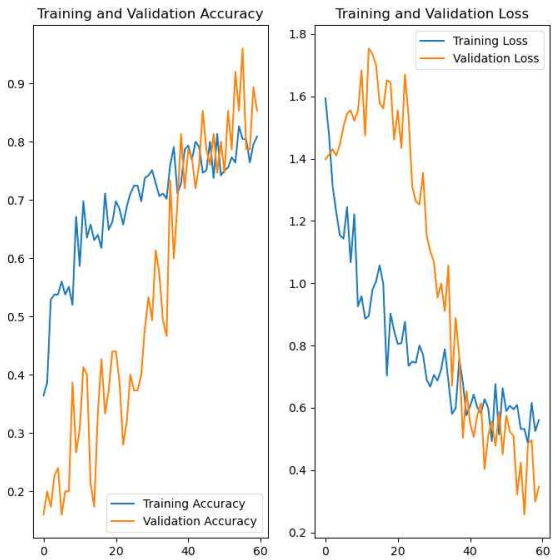


그림_4

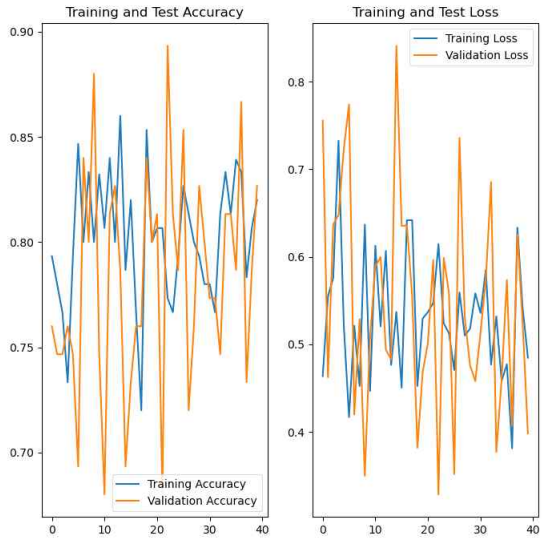


그림_5

▶ 일반 CNN 모델 성능 튜닝

처음 구축한 모델의 accuracy 값이 약 84%이므로 좀 더 분류 정확성을 올리기 위하여 하이퍼 파라미터를 튜닝하도록 한다. 모든 convolution layer의 필터(커널) 수를 2배씩 키웠고, 분류층의 dense 층을 하나 더 추가했다. 그 결과 accuracy 값이 약 94%으로 올라갔고, loss function은 0.12까지 떨어지면서 좀 더 성능이 좋은 모델이 되었음을 확인할 수 있다. (그림_6), (그림_7)

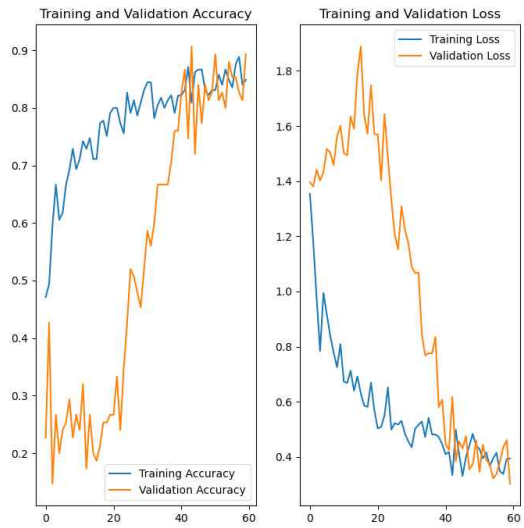


그림_6

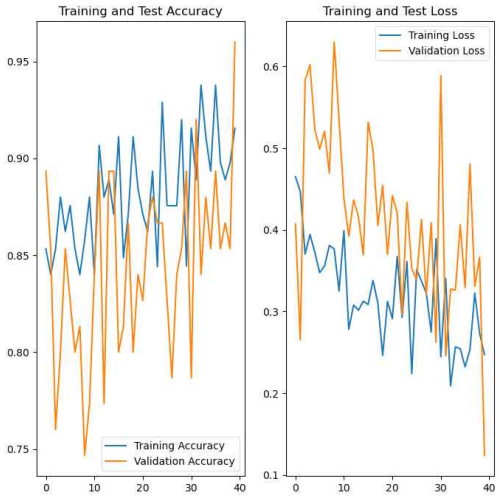


그림_7

▶ 일반 CNN 모델 시각화-1: feature map 추출

일반 모델에서 실제 어떻게 입력 데이터에 대한 특징 추출이 진행되고 있는지 시각적으로 파악하기 위해 특정 이미지를 선택한 후(그림_8) 5개의 Convolution layer를 거칠 때마다 생성되는 feature map을 모두 출력해본다. (그림_9) 층이 깊어질수록 원래 이미지 형태는 점점 없어지고, 이미지의 특징들만 표현되는 것을 확인할 수 있다.

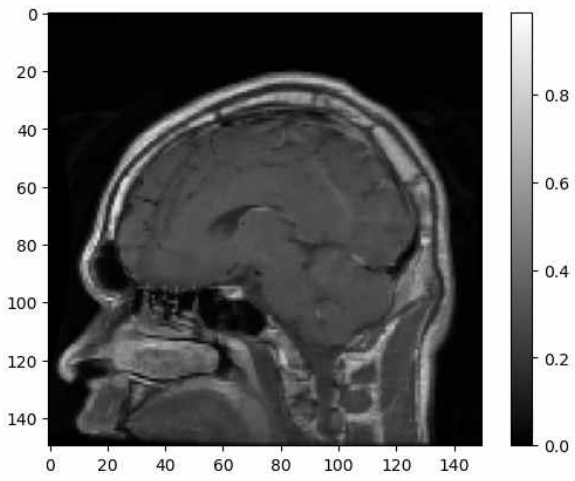


그림_8

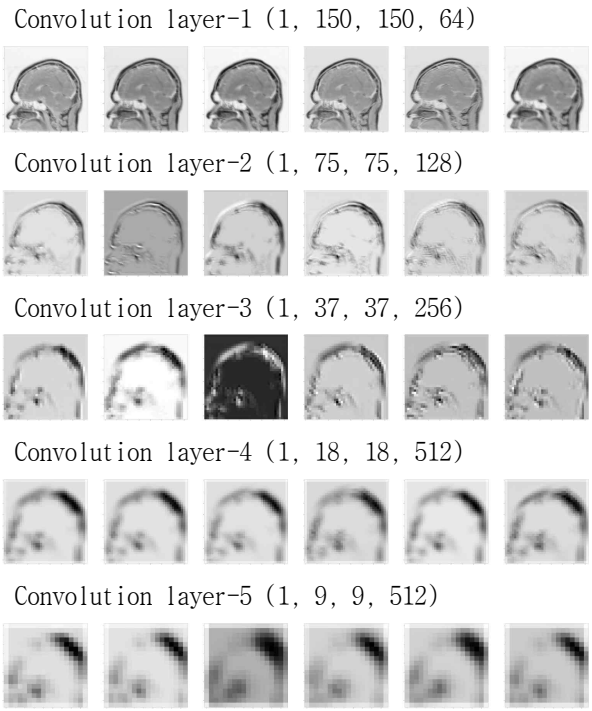


그림_9

▶ 일반 CNN 모델 시각화-2: 오분류 데이터 확인

구축한 모델이 실제 데이터가 속한 class와 다르게 예측한, 즉 오분류한 데이터는 어떤 것들이 있었는지 일부를 출력하여 알아본다. (그림_10)

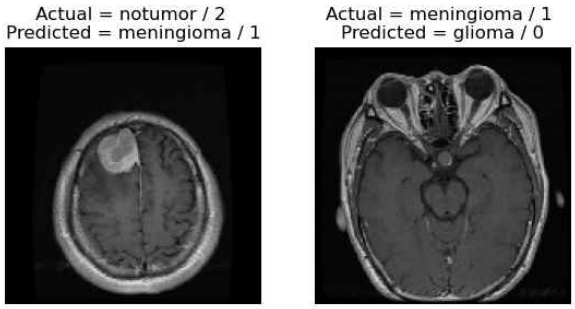


그림_10

2.2.2) 'Mini-VGGNet' 모델 구축 및 모델 평가

VGGNet은 네트워크의 깊이가 모델이 좋은 성능을 보이는 데 중요한 역할을 한다는 것을 보여주는 예시로 이 모델의 가장 큰 특징은 크기가 3x3이고, 1 strides인 작은 크기의 컨볼루션 필터(커널)로 깊은 layer(16～19)를 만들어 좋은 성능을 낸다는 것이다.

가장 대표적인 VGGNet인 VGGNet-16은 크게 5개의 블록으로 이루어져 있으며, 블록은 각각 2, 2, 3, 3, 3개의 convolution layer로 이루어져 있다. 이 convolution layer는 다시 64, 128, 256, 512, 512개의 필터수를 가지고 있다. 우리가 사용할 데이터는 VGGNet-16이 사용한 데이터에 비해 간단하고 크기가 작은 이미지 데이터이므로 VGGNet-16의 모델을 우리 데이터에 그대로 적용하기보다는 '작은 컨볼루션 필터와 깊은 구조'라는 특징은 살리되 convolution layer의 수를 줄이는 'Mini-VGGNet'을 만들기로 한다.

'Mini-VGGNet'은 크게 3개의 블록으로 이루어져 있으며, 블록은 각각 2개씩의 convolution layer로 구성했다. 이 convolution layer는 다시 64, 128, 256개의 필터 수를 가지고 있다. 각 블록의 마지막 부분에는 maxpooling을 적용했다. 이후에 512개의 hidden layer 두 층이 있고, 4개의 결과값을 softmax 활성함수를 거쳐 출력한다. (그림_11)

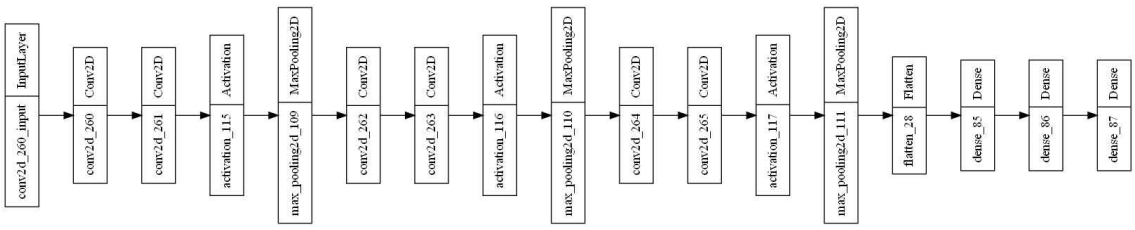


그림_11

▶ Mini-VGGNet 모델 구축 모델 평가

모델의 평가에 사용한 평가 척도는 'accuracy'이다. 구축한 모델의 accuracy 값은 약 89%이다.

training 과정에서 생성해놓은 validation data를 통해 먼저 중간 평가를 거친 후(그림_12) test data를 활용해 모델 평가를 진행했다. (그림_13) 각 과정에서 진행되는 누적 epoch 수에 따른 Accuracy와 Loss function 값을 시각적으로 나타내었다.

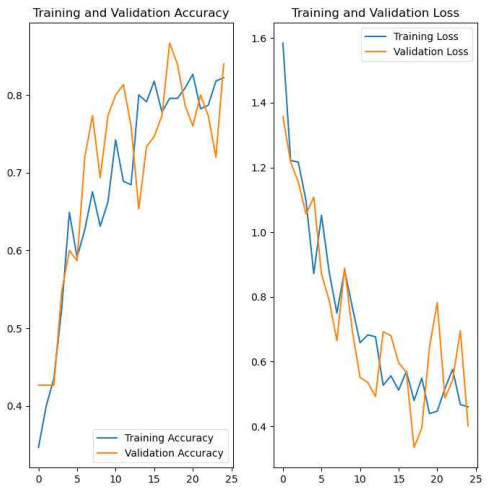


그림_12

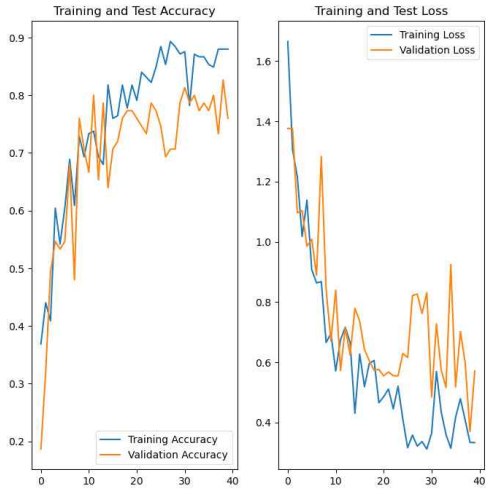


그림_13

2.2.3) 'Mini-ResNet' 모델 구축 및 모델 평가

ResNet은 residual 개념을 고안한 모델로써 깊은 신경망을 효과적으로 학습하기 위한 방법을 제시한다. 신경망의 깊이가 깊어질수록 성능은 좋아지다가 특정 시점부터 반대로 성능이 나빠지게 되는데 이런 문제 해결을 위해 'residual block'이라는 개념을 도입했다. residual block은 그레디언트가 잘 전파될 수 있도록 하는 일종의 shortcut(skip connection)을 만들어 준다. 물론 이러한 구조에서는 신경망의 깊이가 깊어질수록 파라미터가 무한대로 커지기 때문에 이를 해결하기 위해 병목 블록(bottleneck block)을 추가로 둔다. ResNet 뒤에는 사용되는 레이어 수에 대응하는 숫자가 붙는데 ResNet50이하의 ResNet은 병목 블럭(bottleneck block)을 사용하지 않고 기본 블럭(basic block)만을 사용한다.

ResNet50의 convolution layer를 보면 병목 블럭과 기본 블럭의 차이를 알 수 있는데 병목 블럭에는 3×3 convolution layer 앞뒤로 1×1 convolution layer가 붙어 있는데, 1×1 convolution layer의 채널 수를 조절하며 차원을 줄였다 늘릴 수 있기에 병목 블럭이 기본 블록보다 파라미터 수가 적을 수 있는 원리이기도 하다.

ResNet의 구조에서 가장 핵심이 되는 부분은 바로 identity mapping 부분이다. 기본 블록과 병목 블록에서 모두에서 사용하는 identity mapping은 입력 데이터 x가 어떠한 함수를 통과하더라도 다시 x의 형태로 출력될 수 있게 하는 것이다.

병목 블록을 사용하지 않는 가장 간단한 ResNet 모델인 ResNet18조차도 앞선 'Mini-VGGNet' 모델을 구축할 때와 마찬가지의 이유로 우리 데이터에 그대로 적용하기 보다는 'shortcut(skip connection)과 identity mapping'이라는 특징은 살리되 디테일한 요소를 줄여서 만든 'Mini-ResNet'을 구축하기로 한다. 'Mini-ResNet'은 ResNet18의 아키텍쳐 중 25%에 해당하는

convolution layer을 구성할 것이다. identity mapping이 있는 identity block을 시행하기 전에 일반적인 convolution layer 1층을 거친다. 그 후 identity mapping이 2회 이루어지는 identity block을 구축한다. 앞선 두 모델과 다르게 분류층에 들어가기 전 'Flatten' 대신에, 'GlobalAveragePooling'을 진행하고, 노드 수를 많게 하는 F.C층을 거치지 않고 곧바로 softmax로 분류한 뒤 해당하는 클래스를 출력하는 모델이 된다. (그림_14)

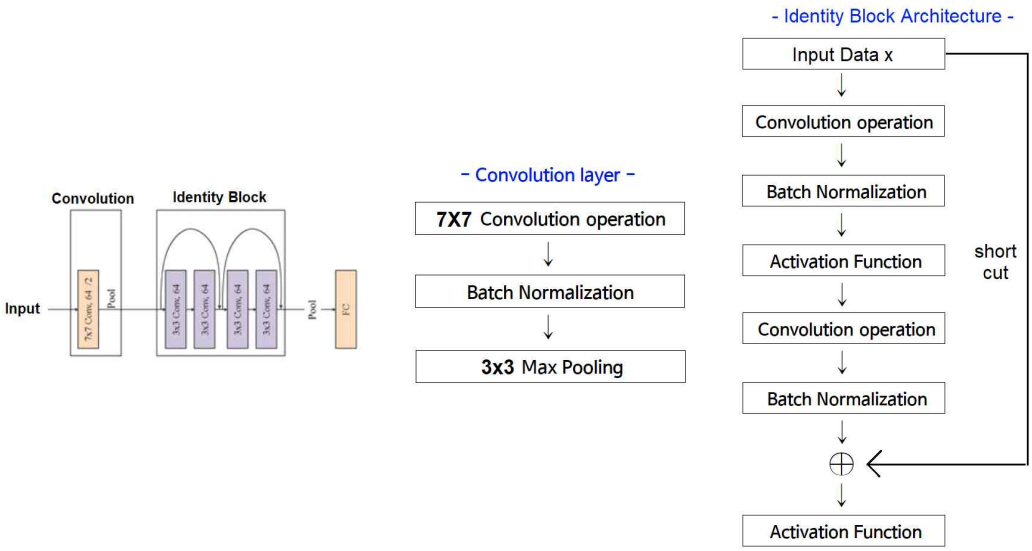


그림_14

▶ **Mini-ResNet 모델 구축 모델 평가**

모델의 평가에 사용한 평가 척도는 'accuracy'이다. 구축한 모델의 accuracy 값은 약 93%이다. ResNet의 경우 validation data를 사용하여 중간 평가를 진행하지 않았는데 그 이유는 비교적 간단한 데이터에 대해서 입력 데이터를 합하여 다음 단계로 출력하는 identity mapping을 사용했을 때 쉽게 잘 맞추기 때문에 굳이 성능을 미리 향상 시킬 필요가 없었다.

3. 마무리: 비교 평가 및 참고문헌 소개

3.1) 비교 평가

비교 평가는 크게 '자체 비교'와 '외부 비교'로 나누어 진행한다. '자체 비교'란 개인적으로 구축한 ① 일반 CNN 모델_'5-Conv + 3-F.C CNN', ② VGGNet의 원리를 차용한 'Mini-VGGNet', ③ ResNet의 원리를 차용한 'Mini-ResNet', 이 3가지 모델의 accuracy를 서로 비교하는 과정이다. '외부 비교'는 해당 데이터를 가져온 'Kaggle'에서 동일 데이터 셋을 통해 CNN 모델을 구현한 다른 kaggle 유저들의 결과와 내가 구축한 모델의 accuracy를 비교하는 과정이다.

3.1.1) 자체 비교

자체 비교에서는 앞서 구축한 3가지 CNN 모델에 대해 그 성능을 'accuracy'로 단순 비교해보

고 그에 따른 간단한 생각을 정리하도록 한다.

▶ 'accuracy' 높은 모델 순서: ③ > ② > ①
▶ 자체적인 아이디어로 만든 모델 ①의 경우 하이퍼 파라미터 튜닝 및 아키텍처 변경과 같이 성능 향상을 위한 단계까지 거쳐가며 성능을 높였고, 모델 ②와 ③은 복잡한 모델의 비교적 핵심만 담고 있는 Mini 모델로 괜찮은 성능을 보여준 것 같다. 특히 ResNet의 경우 깊은 신경망에 대해서 큰 힘을 발휘하는데 현재 가지고 있는 데이터 크기와 복잡도에 비해 깊은 ResNet은 과한 모델이기에 과감히 축소하여 모델을 구축하였다. input data의 크기가 아주 큰 편은 아니라 Pooling을 많이 진행하기에는 어려움이 있었고, 컨볼루션 연산을 할 때 필터(커널) 수를 많이 할수록 연산에 걸리는 시간이 배로 증가하여 batch size와 epochs를 조절했음에도 불구하고 현실적인 어려움이 있었다.

3.1.2) 외부 비교

Kaggle 내의 해당 데이터에 대하여 다양한 유저가 제각기 다른 모델로 실험을 했기에 일반적인 CNN을 사용한 케이스, VGGNet을 사용한 케이스, ResNet을 사용한 케이스를 하나씩 임의로 선정하여 비교하도록 한다. <표_3>을 보면 동일 데이터에 대한 다른 Kaggle 유저들의 결과값을 모두 비교할 순 없었으나 임의로 선택한 유저들의 결과값과 비교했을 때 비슷하거나 더 높은 성능을 보였음을 알 수 있다.

<표_3>

Model	Architecture		accuracy(%)	
	mine	Other's	mine	Other's
① 일반적인 CNN	5-Conv + 3-F.C	6-Conv + 3-F.C	93.8	76.74
② VGGNet	5-Convolution block(Conv*2) + 3-F.C	VGG16	89	92.87
③ ResNet	Conv-1 + identity block(Conv*2)-2	ResNet50	93	77

※ ① 타 유저 결과 URL: https://www.kaggle.com/code/mohamedeldakrory8/brain-tumor-mri-classification-graduation
※ ② 타 유저 결과 URL: https://www.kaggle.com/code/ucrkmveri/brain-tumour-detection-using-vgg16
※ ③ 타 유저 결과 URL: **https://www.kaggle.com/code/shiblinomani/brain-tumor-mri-transferlr-vgg19-and-racenet50/notebook**

3.2) 참고문헌 소개

다음과 같은 논문 및 구글링 자료(URL)를 활용해 모델 구성 및 파이썬 코드 작성에 참고하였다. 구축하고자 하는 모델을 먼저 설정한 후 해당 모델을 구현하기 위해 필요한 파이썬 코드를 찾는데 주로 도움을 받았고, Kaggle에 있는 파이썬 코드는 전혀 참고하지 않았다. 논문의 경우 해당 주제에 대한 연구가 어느 정도 이루어져 있는지에 대해 확인하는 용도로 참고하였다.

- 논문 -

① 강재용, 곽정환,「Deep Learning-Based Brain Tumor Classification in MRI images using Ensemble of Deep Features」, 한국 컴퓨터정보학회, 2021.07, page 37~44
② 박세진 외 4명,「딥러닝을 이용한 뇌 자기공명영상의 정량 분석기법, 정보과학회지, 2017,05」
③ 공준배, 장민석, 「CNN의 컨볼루션 레이어, 커널과 정확도의 연관관계 분석, 한국전자통신학회 논문지 2019」

- 구글링 자료 -

① 사용 데이터 출처: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/code
② 훈련과정 시각화 관련: https://codetorial.net/tensorflow/visualize_training_history.html
③ Image_generator 관련: https://chancoding.tistory.com/93
④ VGGNet 관련: https://aistudy9314.tistory.com/25
⑤ Resnet 관련-1: https://deep-learning-study.tistory.com/510
⑥ Resnet 관련-2: https://inhovation97.tistory.com/39