

뇌출혈 조기 진단 프로젝트 - Version 4.0

Version 4.0 업데이트 내용

- v3.0: JWT 기반 로그인/회원가입 시스템 추가
- v4.0: AWS Lightsail 웹 배포 완료

목차

- [시스템 개요](#시스템-개요)
- [기술 스택](#기술-스택)
- [딥러닝 모델](#딥러닝-모델)
- [프로젝트 구조](#프로젝트-구조)
- [로컬 환경 설치 및 실행](#로컬-환경-설치-및-실행)
- [API 명세](#api-명세)
- [데이터베이스 설계](#데이터베이스-설계)
- [화면 흐름](#화면-흐름)
- [AWS Lightsail 배포 가이드](#aws-lightsail-배포-가이드)
- [보안 설정](#보안-설정)
- [주의사항](#주의사항)
- [버전 히스토리](#버전-히스토리)

시스템 개요

AI 기반 CT 영상 분석을 통한 뇌출혈 진단 보조 시스템입니다.

핵심 기능

기능	설명
뇌출혈 진단	ResNet50 기반 딥러닝 모델로 CT 이미지 분석
Grad-CAM 시각화	진단 근거를 히트맵으로 시각화
사용자 인증	JWT 기반 로그인/회원가입 (v3.0)
웹 인터페이스	Streamlit 기반 사용자 친화적 UI
웹 배포	AWS Lightsail + HTTPS (v4.0)

접속 URL

- 프론트엔드: <https://brain-hemorrhage.duckdns.org>
- 백엔드 API: <https://brain-hemorrhage.duckdns.org/api/>

기술 스택

Frontend (Streamlit)

기술	버전	용도
Python	3.12+	메인 언어
Streamlit	1.51.0	웹 UI 프레임워크
TensorFlow	2.20.0	딥러닝 모델
OpenCV	4.x	이미지 처리
Requests	2.x	API 통신

Backend (Spring Boot)

기술	버전	용도
Java	17	메인 언어
Spring Boot	3.2.0	웹 프레임워크
Spring Security	6.x	보안/인증
Spring Data JPA	3.x	ORM
MySQL	8.x	데이터베이스
JWT	0.12.3	JWT 토큰 처리
Lombok	1.18+	코드 간소화

배포 환경

기술	버전	용도
AWS Lightsail	Ubuntu 24.04 LTS	클라우드 서버
Nginx	1.24.0	리버스 프록시
Let's Encrypt	-	SSL 인증서
DuckDNS	-	무료 도메인

딥러닝 모델

개요

ResNet50 기반 딥러닝 모델을 사용하여 뇌 CT 영상에서 뇌출혈을 자동으로 감지하는 시스템입니다.

항목	내용
목표 정확도	90% 이상
모델	ResNet50 (ImageNet pretrained)
학습 전략	2단계 Fine-tuning
설명 가능성	Grad-CAM 기반 시각화

데이터셋

RSNA Intracranial Hemorrhage Detection 데이터셋 기반

구분	Hemorrhage	Normal	합계
Train	2,152장	3,371장	5,523장
Test	538장	843장	1,381장
비율	39%	61%	100%

클래스 불균형 처리: Class Weights 적용 (hemorrhage에 1.3배 가중치)

모델 아키텍처

Input (224x224x3)

v

+-----+-----+

ResNet50 (ImageNet pretrained)
- 50개 레이어

- Feature Extraction

```
+-----+  
|  
v  
GlobalAveragePooling2D  
v  
Dense(256, ReLU)  
v  
Dropout(0.5)  
v  
Dense(128, ReLU)  
v  
Dropout(0.3)  
v  
Dense(1, Sigmoid)  
v  
Output (0~1: 출혈 확률)  
'
```

2단계 학습 전략 (Fine-tuning)

Stage 1: Feature Extraction (5 epochs)

- 목적: Top layers 빠른 수렴
- 설정: Base model 전체 Freeze
- Learning Rate: 1e-3 (높은 LR)

```
'python  
for layer in base_model.layers:  
    layer.trainable = False  
'
```

Stage 2: Fine-tuning (25 epochs)

- 목적: 뇌 CT 특화 학습
- 설정: 마지막 30개 레이어 Unfreeze
- Learning Rate: 1e-4 (낮은 LR로 섬세하게)

```
'python  
for layer in base_model.layers[-30:]:  
    layer.trainable = True  
'
```

하이퍼파라미터 설정

파라미터	Stage 1	Stage 2
Learning Rate	1e-3	1e-4
Epochs	5	25
Batch Size	32	32
Optimizer	Adam	Adam
Loss	Binary Crossentropy	Binary Crossentropy

Class Weights (클래스 불균형 처리)

```
`python
class_weights = {
0: 1.0, # Normal
1: 1.3 # Hemorrhage (30% 가중치 강화)
}
`
```

Data Augmentation

학습 데이터의 다양성을 높여 과적합 방지

```
`python
ImageDataGenerator(
rescale=1./255,
rotation_range=20, # 회전: ±20°
width_shift_range=0.15, # 좌우 이동: 15%
height_shift_range=0.15, # 상하 이동: 15%
horizontal_flip=True, # 좌우 반전
zoom_range=0.2, # 확대/축소: 20%
brightness_range=[0.8, 1.2], # 밝기 조정
fill_mode='nearest'
)
`
```

Callbacks

```
`python
```

조기 종료

```
EarlyStopping(
monitor='val_loss',
patience=5,
restore_best_weights=True
)
```

학습률 감소

```
ReduceLROnPlateau(
monitor='val_loss',
factor=0.5, # LR을 절반으로
patience=3,
min_lr=1e-7
)
```

최적 모델 저장

```
ModelCheckpoint(
filepath='best_model.h5',
monitor='val_accuracy',
save_best_only=True
)
`
```

모델 성능 비교

모델	Test Accuracy	실제 예측	비고
CNN (직접 설계)	99.13%	실패	과적합
ResNet50 from scratch	84.50%	실패	학습 불충분
ResNet50 Transfer	80.45%	실패	Top layers만
ResNet50 Fine-tuning	87.83%	성공	2단계 학습

참고 성공 사례 (RSNA Kaggle Competition):

- ResNet50 Fine-tuning: 90%+ 달성
- Ensemble Deep Learning: 92% sensitivity

전처리 파이프라인

preprocessing_utils.py 모듈에서 제공

주요 기능

기능	설명
DICOM 지원	의료 영상 표준 형식 처리
Window Leveling	Brain Window (center=40, width=80)
자동 RGB 변환	그레이스케일 -> RGB
정규화	0-1 범위로 스케일링
TTA 지원	Test-Time Augmentation

사용 예시

```
'python
from preprocessing_utils import CTImagePreprocessor
preprocessor = CTImagePreprocessor(target_size=(224, 224))
image = preprocessor.preprocess('path/to/ct_image.jpg')
prediction = model.predict(image)
'
```

Grad-CAM (설명 가능한 AI)

gradcam_utils.py 모듈에서 제공

모델이 어느 부위를 보고 판단했는지 히트맵으로 시각화

작동 원리

- 마지막 Convolutional layer의 활성화 추출
- 예측 클래스에 대한 Gradient 계산
- Gradient-weighted activation map 생성
- 원본 이미지에 히트맵 오버레이

기능

- 진단 근거 시각화 (빨간색 = 높은 관심 영역)
- 뇌 영역 위치 추정 (9분면: 전두엽, 측두엽, 후두엽 등)
- 신뢰도 기반 자동 설명 텍스트 생성

사용 예시

```
'python
from gradcam_utils import GradCAM, explain_with_gradcam
'
```

```
gradcam = GradCAM(model)
result = gradcam.explain_prediction(image_path, preprocessor)
print(result['predicted_class']) # 'hemorrhage' or 'normal'
print(result['confidence']) # 0.0 ~ 1.0
print(result['explanation']) # 자동 생성된 설명
```

학습 코드 실행 (Google Colab)

```
`python
```

1. Google Drive 마운트

```
from google.colab import drive
drive.mount('/content/drive')
```

2. 학습 코드 실행

```
%cd /content/drive/MyDrive/brain_ct
%run Model_code/ResNet50_Optimized_90percent.py
```

3. 결과 확인

- **resnet50_final_optimized.h5:** 최종 모델
- **training_curves.png:** 학습 곡선

프로젝트 구조

```
↳ 출혈/
└── ReadMe/
└── readme_ver4.md # 이 문서
```

로컬 환경 설치 및 실행

1. 사전 요구사항

- Java 17 이상
- Python 3.8 이상
- MySQL 8.x
- Spring Tool Suite (STS) 또는 IntelliJ

2. 데이터베이스 설정

MySQL에서 다음 명령 실행:

```
`sql
CREATE DATABASE brain_hemorrhage;
```

3. 백엔드 설정

1. application.yml 수정 (backend/src/main/resources/application.yml)

```
`yaml
spring:
```

```
datasource:  
url: jdbc:mysql://localhost:3306/brain_hemorrhage  
username: root  
password: [본인 비밀번호] # <- 수정 필요
```

2. STS에서 프로젝트 Import

- File -> Import -> Maven -> Existing Maven Projects
- Root Directory: D:\n\출혈\backend 선택
- Finish 클릭

3. 서버 실행

- BrainHemorrhageApplication.java 우클릭
- Run As -> Spring Boot App
- 콘솔에 "서버 시작 완료" 메시지 확인

4. 프론트엔드 실행

```
`bash  
cd D:\n\출혈\Streamlit  
pip install requests # 추가 패키지 설치  
streamlit run brain_ct_improved.py
```

5. 접속

- 프론트엔드: http://localhost:8501
- 백엔드 API: http://localhost:8080

API 명세

인증 API

회원가입

```
POST /api/auth/signup  
Content-Type: application/json  
Request:
```

```
{  
    "username": "testuser",  
    "password": "password123",  
    "name": "홍길동",  
    "email": "hong@example.com"  
}
```

Response (성공):

```
{  
    "success": true,  
    "message": "회원가입이 완료되었습니다."  
}
```

Response (실패):

```
{  
    "success": false,  
    "message": "이미 사용 중인 아이디입니다."
```

}

,

로그인

POST /api/auth/login

Content-Type: application/json

Request:

```
{  
  "username": "testuser",  
  "password": "password123"  
}
```

Response (성공):

```
{  
  "success": true,  
  "message": "로그인 성공",  
  "token": "eyJhbGciOiJIUzI1NiL...  
  "username": "testuser",  
  "name": "홍길동"  
}
```

Response (실패):

```
{  
  "success": false,  
  "message": "아이디 또는 비밀번호가 올바르지 않습니다."  
}
```

,

토큰 검증

GET /api/auth/validate

Authorization: Bearer {token}

Response:

```
{  
  "success": true,  
  "message": "토큰 유효",  
  "username": "testuser",  
  "name": "홍길동"  
}
```

,

서버 상태 확인

GET /api/auth/health

Response:

```
{  
  "success": true,  
  "message": "서버 정상 동작 중"  
}
```

,

데이터베이스 설계

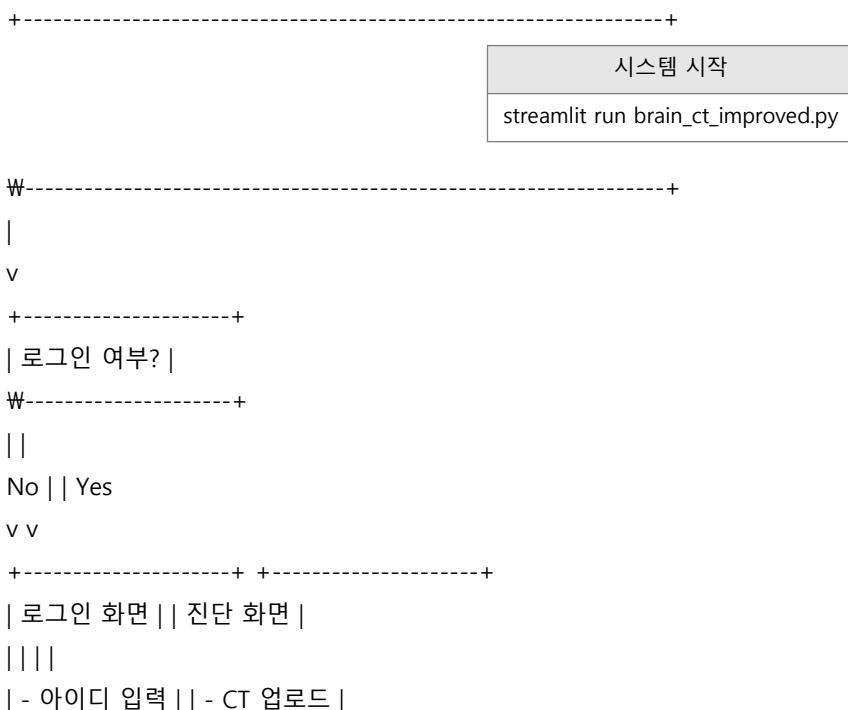
users 테이블

컬럼명	타입	제약조건	설명
id	BIGINT	PK, AUTO_INCREMENT	기본키
username	VARCHAR(50)	UNIQUE, NOT NULL	로그인 아이디
password	VARCHAR(255)	NOT NULL	암호화된 비밀번호
name	VARCHAR(100)	NOT NULL	사용자 이름
email	VARCHAR(100)	UNIQUE, NOT NULL	이메일
created_at	TIMESTAMP	DEFAULT NOW()	생성 시간
updated_at	TIMESTAMP	ON UPDATE NOW()	수정 시간

DDL

```
'sql
CREATE TABLE users (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);'
```

화면 흐름



- 비밀번호 입력		- 모델 선택
- 로그인 버튼		- Grad-CAM
- 회원가입 버튼		- 로그아웃 버튼

₩-----+ ₩-----+

|

회원가입 클릭

v

+-----+

| 회원가입 화면 |

||

| - 아이디 |

| - 비밀번호 |

| - 비밀번호 확인 |

| - 이름 |

| - 이메일 |

| - 회원가입 버튼 |

₩-----+

|

회원가입 성공 (2초 후 자동 이동)

|

v

+-----+

| 로그인 화면 |

₩-----+

,

AWS Lightsail 배포 가이드

시스템 아키텍처

+-----+

사용자 브라우저

₩-----+

|

v HTTPS (443)

+-----+

brain-hemorrhage.duckdns.org

(DuckDNS 도메인)

₩-----+

|

v

+-----+

AWS Lightsail (Ubuntu 24.04 LTS)				
\$24/월, 4GB RAM, 2 vCPU				

고정 IP: 15.165.254.0			
+-----+			
	Nginx (리버스 프록시)		
	Port: 80, 443		
	SSL: Let's Encrypt		
W-----+			
/api/ 요청		/ 요청	
v v			
+-----+ +-----+			
	Spring Boot	Streamlit	
	(백엔드 API)	(프론트엔드)	
	Port: 8080	Port: 8501	
	- JWT 인증	- CT 이미지 업로드	
	- 회원 관리	<--- - 모델 예측 결과	
	- 분석 기록 저장		- GradCAM 시각화
W-----+ W-----+			
v v			
+-----+ +-----+			
	MySQL	TensorFlow	
	Port: 3306	모델 파일	
	DB: brain_hemorrhage	(.h5 files)	
W-----+ W-----+			

W-----+ |

포트 구성

포트	서비스	접근
22	SSH	관리자만
80	HTTP -> HTTPS 리다이렉트	외부
443	HTTPS (Nginx)	외부
3306	MySQL	내부만
8080	Spring Boot	내부만
8501	Streamlit	내부만

배포 과정

1단계: 인스턴스 생성

플랫폼: Linux/Unix

OS: Ubuntu 24.04 LTS

플랜: \$24/월 (4GB RAM, 2 vCPU, 80GB SSD)

리전: 서울 (ap-northeast-2)

2단계: 네트워크 설정

```
'bash
```

고정 IP 할당

방화벽 포트 오픈: 22, 80, 443, 8080, 8501

3단계: 시스템 초기화

```
'bash
```

```
sudo apt update && sudo apt upgrade -y  
sudo apt install openjdk-17-jdk -y  
sudo apt install mysql-server -y  
sudo apt install python3-pip python3-venv nginx git maven -y
```

4단계: MySQL 설정

```
'bash
```

```
sudo mysql_secure_installation  
sudo mysql
```

```
'sql
```

```
CREATE DATABASE brain_hemorrhage CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;  
CREATE USER 'brainuser'@'localhost' IDENTIFIED BY '1234';  
GRANT ALL PRIVILEGES ON brain_hemorrhage.* TO 'brainuser'@'localhost';  
FLUSH PRIVILEGES;
```

5단계: 프로젝트 배포

```
'bash
```

```
sudo apt install git-lfs -y  
git lfs install  
git clone https://github.com/parkahjin/brain-hemorrhage.git brain_project
```

6단계: Spring Boot 빌드

```
'bash
```

application.yml 수정 (DB 연결 정보)

```
cd ~/brain_project/backend
```

```
mvn clean package -DskipTests
```

7단계: Streamlit 환경 구성

```
'bash
```

```
python3 -m venv ~/streamlit_env  
source ~/streamlit_env/bin/activate  
pip install -r ~/brain_project/requirements.txt
```

8단계: Streamlit 설정 (WebSocket 문제 해결)

```
'bash
```

```
mkdir -p ~/.streamlit
```

```
nano ~/.streamlit/config.toml
`toml
[server]
enableCORS = false
enableXsrfProtection = false
enableWebSocketCompression = false
`
```

9단계: Nginx 리버스 프록시

```
`nginx
server {
listen 80;
server_name brain-hemorrhage.duckdns.org;
return 301 https://$host$request_uri;
}
server {
listen 443 ssl;
server_name brain-hemorrhage.duckdns.org;
ssl_certificate /etc/letsencrypt/live/brain-hemorrhage.duckdns.org/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/brain-hemorrhage.duckdns.org/privkey.pem;
location /api/ {
proxy_pass http://127.0.0.1:8080;
proxy_http_version 1.1;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
}
location /_stcore/stream {
proxy_pass http://127.0.0.1:8501/_stcore/stream;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_read_timeout 86400;
}
location / {
proxy_pass http://127.0.0.1:8501;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
proxy_set_header Host $host;
proxy_read_timeout 86400;
}
}`
```

10단계: SSL 인증서 (Let's Encrypt)

```
'bash
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d brain-hemorrhage.duckdns.org
'
```

11단계: systemd 서비스 등록

```
Spring Boot 서비스 (/etc/systemd/system/springboot.service)
`ini
[Unit]
Description=Spring Boot Brain Hemorrhage Backend
After=network.target mysql.service
Requires=mysql.service
[Service]
Type=simple
User=ubuntu
WorkingDirectory=/home/ubuntu/brain_project/backend
ExecStart=/usr/bin/java -jar /home/ubuntu/brain_project/backend/target/hemorrhage-1.0.0.jar
Restart=always
RestartSec=5
[Install]
WantedBy=multi-user.target
'
```

```
Streamlit 서비스 (/etc/systemd/system/streamlit.service)
```

```
`ini
[Unit]
Description=Streamlit Brain Hemorrhage Frontend
After=network.target
[Service]
Type=simple
User=ubuntu
WorkingDirectory=/home/ubuntu/brain_project/Streamlit
Environment="PATH=/home/ubuntu/streamlit_env/bin:/usr/local/bin:/usr/bin:/bin"
ExecStart=/home/ubuntu/streamlit_env/bin/streamlit run brain_ct_improved.py --server.port 8501 --server.address 0.0.0.0
Restart=always
RestartSec=5
[Install]
WantedBy=multi-user.target
'
```

서비스 활성화

```
'bash
sudo systemctl daemon-reload
sudo systemctl start springboot streamlit
sudo systemctl enable springboot streamlit
'
```

유지보수 명령어

서비스 관리

`bash

상태 확인

```
sudo systemctl status springboot streamlit nginx mysql
```

재시작

```
sudo systemctl restart springboot
```

```
sudo systemctl restart streamlit
```

```
sudo systemctl restart nginx
```

로그 확인

```
sudo journalctl -u springboot -f
```

```
sudo journalctl -u streamlit -f
```

```
sudo tail -f /var/log/nginx/error.log
```

코드 업데이트

`bash

```
cd ~/brain_project
```

```
git pull origin main
```

Spring Boot 재빌드

```
cd backend
```

```
mvn clean package -DskipTests
```

```
sudo systemctl restart springboot
```

Streamlit 재시작

```
sudo systemctl restart streamlit
```

시스템 모니터링

`bash

```
free -h # 메모리 확인
```

```
df -h # 디스크 확인
```

```
htop # CPU/프로세스 확인
```

트러블슈팅

502 Bad Gateway

`bash

```
sudo systemctl status springboot streamlit
```

```
sudo systemctl restart springboot streamlit nginx
```

WebSocket 연결 실패

- ~/.streamlit/config.toml 설정 확인

- Nginx에서 /_stcore/stream 경로 설정 확인

메모리 부족

`bash

스왑 메모리 추가

```
sudo fallocate -l 4G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

보안 설정

JWT 토큰

- 알고리즘: HS256
- 유효기간: 1시간 (3600000ms)
- 비밀키: application.yml에서 설정

비밀번호 암호화

- 알고리즘: BCrypt
- 특징: Salt 자동 생성, 단방향 해시

CORS 설정

- 허용 Origin: localhost:8501, localhost:8502, brain-hemorrhage.duckdns.org
- 허용 Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
- 허용 Headers: Authorization, Content-Type 등

서버 보안 정보

항목	값
MySQL DB	brain_hemorrhage
MySQL User	brainuser
SSL 만료일	2026-03-05 (자동 갱신)

주의사항

의료 면책 조항

- 본 시스템은 연구 및 교육 목적으로 개발되었습니다.
- 실제 의료 진단을 대체할 수 없습니다.
- 최종 진단은 반드시 전문의가 수행해야 합니다.

개발 환경 주의

- application.yml의 비밀번호는 본인 환경에 맞게 수정하세요.
- JWT 비밀키는 운영 환경에서 반드시 변경하세요.
- 디버그 로그는 운영 환경에서 비활성화하세요.

버전 히스토리

버전	날짜	주요 변경사항
1.0	2025-11	초기 버전 (CNN 모델)
2.0	2025-11	ResNet50 Fine-tuning, Grad-CAM 추가
3.0	2025-12	로그인/회원가입 시스템 추가
4.0	2025-12	AWS Lightsail 웹 배포 완료

문의

- GitHub: <https://github.com/parkahjin/brain-hemorrhage>
- 프로젝트 관련 문의사항은 Issue를 통해 남겨주세요.

작성자: 박아진

마지막 업데이트: 2025-12-05