

GitHub 백업 가이드

캐글 노트북 코드를 로컬 및 **GitHub**에 백업하기

작업 일자: 2024년 11월 11일

프로젝트: 당뇨병망막병증 분류 AI

Repository: <https://github.com/parkahjin/diabetic-retinopathy-classification>

목차

1. [백업이 필요한 이유](#)
 2. [로컬 백업 과정](#)
 3. [Git 버전 관리 시작](#)
 4. [GitHub 원격 백업](#)
 5. [최종 백업 현황](#)
-

1. 백업이 필요한 이유

문제 상황

- 캐글 노트북: "Save Version" 기능 오류 발생
- **Your Work** 미저장: 캐글 서버에 제대로 저장되지 않음
- 세션 종료 시 손실 위험: 임시 세션으로만 존재할 가능성

백업 전략

1. 로컬 다운로드 (1차 백업)
2. **Git** 버전 관리 (변경 이력 추적)
3. **GitHub** 원격 저장소 (클라우드 백업 + 포트폴리오)

2. 로컬 백업 과정

2.1 캐글에서 다운로드한 파일

노트북 파일: - `mobilenetv2-model-77.18%.ipynb` (320KB) - 최종 코드 - `MobileNetV2 model - 76%.ipynb` - Phase 1 코드

데이터 파일: - `xy_train.npy` - 전처리된 학습 데이터 - `xy_val.npy` - 전처리된 검증 데이터 - `y_train.npy` - 학습 레이블 - `y_val.npy` - 검증 레이블

모델 파일: - `best_model_improved.keras` - Phase 1 모델 (76.64%) - `best_model_improved.pkl` - 모델 pickle 파일 - `best_model_improved.weights.h5` - 모델 가중치

문서 파일: - `당뇨망막병증_프로젝트완전정리_20241111.md` (Markdown) - `당뇨망막병증_프로젝트완전정리_20241111.pdf` (PDF, 160KB)

2.2 로컬 저장 위치

```
C:\Users\Myj-B0G-12M\Desktop\딥러닝 파일\  
├─ MobileNetV2 model - 76% (폴더)  
├─ 첩자라 (폴더)  
├─ mobilenetv2-model-77.18%.ipynb  
├─ 당뇨망막병증_프로젝트완전정리_20241111.md  
└─ 당뇨망막병증_프로젝트완전정리_20241111.pdf
```

3. Git 버전 관리 시작

3.1 Git 초기화

작업 디렉토리로 이동:

```
cd "C:\Users\Myj-B0G-12M\Desktop\딥러닝 파일"
```

Git 저장소 초기화:

```
git init
```

결과:

```
Initialized empty Git repository in C:/Users/Myj-B0G-12M/Desktop/딥러닝 파일/.git/
```

3.2 Git 사용자 정보 설정

사용자 이름 설정:

```
git config --global user.name "parkahjin"
```

이메일 설정:

```
git config --global user.email "park266692@gmail.com"
```

3.3 .gitignore 파일 생성

대용량 파일을 GitHub에 올리지 않기 위한 설정:

```
echo *.npy > .gitignore  
echo *.keras >> .gitignore
```

```
echo *.h5 >> .gitignore
echo *.pkl >> .gitignore
```

.gitignore 내용:

```
*.npy
*.keras
*.h5
*.pkl
```

제외 이유: - **GitHub** 파일 크기 제한: 100MB - 데이터 파일은 재생성 가능: 전체
리 코드가 있음 - 모델 파일은 재학습 가능: 학습 코드가 있음 - 용량 절약: 불필요
한 대용량 파일 제외

3.4 파일 추가 및 커밋

모든 파일 스테이징:

```
git add .
```

첫 커밋:

```
git commit -m "Initial commit: 당뇨망막병증 분류 프로젝트 (Phase 1: 76.64%, 최종: 77.18%)"
```

커밋 결과:

```
[master (root-commit) 076f56f] Initial commit: 당뇨망막병증 분류 프로젝트 (Phase 1: 76.64%, 최
5 files changed, 74+ insertions(+)
create mode 100644 .gitignore
create mode 100644 MobileNetV2 model - 76%/mobilenetv2-model-76.ipynb
create mode 100644 mobilenetv2-model-77.18%.ipynb
create mode 100644 당뇨망막병증_프로젝트완전정리_20241111.md
create mode 100644 당뇨망막병증_프로젝트완전정리_20241111.pdf
```

4. GitHub 원격 백업

4.1 GitHub Repository 생성

Repository 정보: - 이름: `diabetic-retinopathy-classification` - 설명: 당뇨병 망막병증 분류 AI (MobileNetV2, 77.18% accuracy) - 공개 설정: Public (포트폴리오용) - **URL:** `https://github.com/parkahjin/diabetic-retinopathy-classification`

생성 방법: 1. GitHub 로그인 2. 우측 상단 `+` → `New repository` 3. Repository name 입력 4. Public 선택 5. **"Add a README file"** 체크 해제 (로컬 파일 사용) 6. `Create repository` 클릭

4.2 원격 저장소 연결

GitHub 저장소 추가:

```
git remote add origin https://github.com/parkahjin/diabetic-retinopathy-classification.git
```

기본 브랜치 이름 설정:

```
git branch -M main
```

4.3 코드 푸시

원격 저장소에 푸시:

```
git push -u origin main
```

인증: - **Username:** `parkahjin` - **Password:** Personal Access Token (GitHub Settings에서 생성)

Personal Access Token 생성 방법: 1. GitHub → Settings → Developer settings 2. Personal access tokens → Tokens (classic) 3. **Generate new token (classic)** 클릭 4. Note: **딥러닝 프로젝트** 5. **repo** 체크 6. **Generate token** 클릭 7. 토큰 복사 (다시 볼 수 없음!)

4.4 푸시 성공 확인

예상 출력:






```
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 1.23 MiB | 2.34 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/parkahjin/diabetic-retinopathy-classification.git
* [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```





5. 최종 백업 현황

5.1 3중 백업 체계 구축 완료

백업 위치	상태	내용	목적
로컬 PC	✅ 완료	모든 파일 (코드 + 데이터 + 모델)	즉시 접근 가능
Git 저장소	✅ 완료	버전 관리 (.git 폴더)	변경 이력 추적
GitHub	✅ 완료	코드 + 문서 (대용량 파일 제외)	클라우드 백업 + 포트폴리오

5.2 GitHub에 저장된 파일

포함된 파일: -  mobilenetv2-model-77.18%.ipynb (320KB) - 
MobileNetV2 model - 76%/mobilenetv2-model-76.ipynb -  당뇨망막병증_프로젝트
완전정리_20241111.md (22KB) -  당뇨망막병증_프로젝트완전정리_20241111.pdf
(160KB) -  .gitignore

제외된 파일 (로컬에만 존재): -  *.npy (데이터 파일) - 대용량 -  *.keras
(모델 파일) - 대용량 -  *.h5 (가중치 파일) - 대용량 -  *.pkl (pickle 파일) - 대용량

5.3 백업 완료 후 이점

1. 안전성 - 로컬 하드 손상 시에도 GitHub에서 복구 가능 - 실수로 삭제해도 Git 히스토리에서 복원 가능
2. 포트폴리오 활용 - Public Repository로 채용 담당자에게 공유 가능 - README 추가하여 프로젝트 설명 보강 가능 - Jupyter Notebook이 GitHub에서 바로 렌더링됨
3. 협업 가능 - 다른 개발자와 코드 공유 용이 - Issue 트래킹 가능 - Pull Request로 코드 리뷰 가능
4. 버전 관리 - 코드 변경 이력 추적 - 필요 시 이전 버전으로 롤백 - 브랜치로 실험적 변경 시도 가능

6. 향후 업데이트 방법

파일 수정 후 **GitHub** 업데이트

1. 수정된 파일 확인:

```
git status
```

2. 변경사항 추가:

```
git add .
```

3. 커밋 메시지와 함께 저장:

```
git commit -m "Update: TTA 최적화 코드 추가"
```

4. **GitHub**에 푸시:

```
git push
```

새 파일 추가 시

```
git add 새파일이름.ipynb  
git commit -m "Add: Grad-CAM 시각화 코드"  
git push
```

README.md 추가 권장

프로젝트 설명 파일 생성:

```
# 당뇨망막병증 분류 AI 프로젝트  
  
## 프로젝트 개요  
망막 이미지에서 당뇨망막병증을 탐지하는 이진 분류 AI 모델  
  
## 주요 성과  
- Phase 1 정확도: 76.64%  
- 최종 정확도 (TTA + 임계값 최적화): 77.18%  
  
## 기술 스택  
- TensorFlow/Keras  
- MobileNetV2  
- CLAHE 전처리  
- Focal Loss
```


파일 구조

- `mobilenetv2-model-77.18%.ipynb`: 최종 코드
- `당뇨망막병증_프로젝트완전정리_20241111.pdf`: 프로젝트 문서

7. 정리

완료된 작업

- ✓ 로컬 백업 - 캐글에서 모든 파일 다운로드 - 안전한 로컬 저장소 확보
- ✓ **Git** 버전 관리 - Git 저장소 초기화 - .gitignore로 대용량 파일 제외 - 첫 커밋 완료
- ✓ **GitHub** 원격 백업 - Public Repository 생성 - 코드 및 문서 푸시 완료 - 포트폴리오 활용 가능

백업 완료 체크리스트

- [x] 로컬에 모든 파일 다운로드
- [x] Git 초기화 및 설정
- [x] .gitignore 설정
- [x] 첫 커밋 완료
- [x] GitHub Repository 생성
- [x] 원격 저장소에 푸시 완료
- [x] Public 설정으로 포트폴리오 준비

백업 완료일: 2024년 11월 11일

Repository URL: <https://github.com/parkahjin/diabetic-retinopathy-classification>

작업 시간: 약 10분

부록: 유용한 **Git** 명령어

상태 확인

```
git status          # 현재 상태 확인
git log             # 커밋 히스토리 보기
git log --oneline   # 간단한 히스토리
```

변경사항 확인

```
git diff           # 수정 내용 확인
git diff --staged  # 스테이징된 변경사항 확인
```

브랜치 관리

```
git branch          # 브랜치 목록
git branch new-feature # 새 브랜치 생성
git checkout main   # main 브랜치로 이동
```

원격 저장소

```
git remote -v      # 원격 저장소 확인
git pull           # GitHub에서 최신 코드 가져오기
git push           # GitHub에 변경사항 푸시
```

문서 끝