Hugging Face Hub

Module 4 of 7

Presenter: Vương Cường — Research Assistant, BAI Lab

Introduction

What is Hugging Face Hub?

Hugging Face Hub = **Centralized platform for ML datasets and models**

Purpose

- **Version control for large files** - Git LFS backend for datasets/models
- **Optimized for ML workloads** - Efficient storage and transfer
- **Community sharing and collaboration** - Open science principles
- **Professional data management** - Reproducible research workflows

Why Essential for Research?

- **Proper dataset versioning** - Track data changes systematically
- **Model artifact storage** - Save and share trained models
- **Collaboration enablement** - Team access to shared resources
- **Reproducibility support** - Fixed dataset/model versions for papers

Comparison with Traditional Storage

Storage Method	File Size Limit	Version Control	ML Optimization	Collaboration	Cost
GitHub	100MB per file	Yes	No	Limited	Free/Paid
Google Drive	5TB	No	No	Basic	Free/Paid
Dropbox	No limit	No	No	Basic	Paid
AWS S3	5TB per object	No	No	Programmatic	Paid
HF Hub	**Very Large**	**Yes**	**Yes**	**Excellent**	**Free/Paid**

Key Advantages

- **Git LFS integration** - Seamless version control for large files
- **Dataset viewer** - Browse data without downloading
- **Model cards** - Standardized documentation
- **Direct integration** - Works with transformers, datasets libraries
- **Free hosting** - Generous limits for research use

Purpose-built for ML research workflows - Unlike general storage solutions

Essential Operations (80/20 Principle)

Authentication and Setup

```
# Install Hugging Face CLI
uv pip install huggingface_hub

# Login with your token (one-time setup)
huggingface-cli login
# Enter your token from: https://huggingface.co/settings/tokens
```

Core Upload/Download Operations

```python
# Most common operations for research
from huggingface_hub import login, upload_folder, download_folder, upload_file

# Upload dataset folder
upload_folder(
    folder_path="dataset/processed/my_research_data",
    repo_id="bailab/research-dataset-v1",
    repo_type="dataset"
)

# Upload trained model
upload_folder(
    folder_path="models/checkpoints/best_model",
    repo_id="bailab/transformer-model-v1",
    repo_type="model"
)

# Download for use in training
download_folder(
    repo_id="bailab/research-dataset-v1",
    local_dir="./data"
)
```

Integration with Popular Libraries

```
# Direct integration with datasets library
from datasets import load_dataset, Dataset

# Load from HF Hub
dataset = load_dataset("bailab/research-dataset-v1")

# Save to HF Hub
dataset.push_to_hub("bailab/processed-dataset-v2")
```

Lab Demo with base-research-repo

Step 1: Setup HF Hub Authentication

```
cd base-research-repo

# Login to Hugging Face
huggingface-cli login
# Paste your token from: https://huggingface.co/settings/tokens

# Verify authentication
huggingface-cli whoami
```

Step 2: Upload Dataset

```
# Create example dataset structure
mkdir -p dataset/processed/bailab_demo_data
echo '{"text": "Sample research data", "label": 1}' > dataset/processed/bailab_demo_data/train.jsonl
echo '{"text": "Test research data", "label": 0}' > dataset/processed/bailab_demo_data/test.jsonl

# Upload to HF Hub
from huggingface_hub import upload_folder

upload_folder(
    folder_path="dataset/processed/bailab_demo_data",
    repo_id="bailab/demo-research-dataset",
    repo_type="dataset",
    commit_message="Initial upload of demo research dataset"
)
```

Step 3: Download and Use Dataset

```python
# In your training script
from datasets import load_dataset

# Load dataset from HF Hub
dataset = load_dataset("bailab/demo-research-dataset")

# Access train/test splits
train_data = dataset['train']
test_data = dataset['test']

print(f"Training samples: {len(train_data)}")
print(f"Test samples: {len(test_data)}")
```

Step 4: Upload Trained Model

```python
# After training, upload model artifacts
from transformers import AutoTokenizer, AutoModel

# Save model locally first
model.save_pretrained("./trained_model")
tokenizer.save_pretrained("./trained_model")

# Upload to HF Hub
upload_folder(
    folder_path="./trained_model",
    repo_id="bailab/demo-trained-model",
    repo_type="model"
)
```

Important Considerations

File Size and Storage Limits

GitHub vs HF Hub Guidelines

- **Never commit large files to GitHub** (>25MB)
- **Use HF Hub for datasets** (>25MB)
- **Use HF Hub for model checkpoints** (always large)
- **Keep code in GitHub** - HF Hub for data/models only

Storage Quotas and Limits

```python
# Check your storage usage
from huggingface_hub import get_repo_discussions, repo_info

info = repo_info("bailab/your-dataset-repo", repo_type="dataset")
print(f"Repository size: {info.cardData.size if info.cardData else 'Unknown'}")
```

Private vs Public Repositories

- **Public repositories** - Open science, community access
- **Private repositories** - Sensitive data, work-in-progress
- **Organization repositories** - Team collaboration

Token Management

```
# Create tokens with minimal required permissions
# Read: Download models/datasets
# Write: Upload new versions
# Delete: Remove repositories (use carefully)

# Store tokens securely
export HUGGINGFACE_TOKEN="your_token_here"
```

Data Management Best Practices

Version Control Strategy

- **Semantic versioning** for datasets (v1.0, v1.1, etc.)
- **Descriptive commit messages** for changes
- **Model cards and dataset cards** for documentation
- **Tag releases** for paper submissions

Repository Organization

```
bailab/research-dataset-v1/
├── train.parquet
├── validation.parquet
├── test.parquet
├── README.md              # Dataset card
└── dataset_info.json      # Metadata
```

Summary: Hugging Face Hub

What We Covered

✅ **HF Hub fundamentals** - ML-optimized storage platform
✅ **Advantages over traditional storage** - Version control + ML features
✅ **Essential operations** - Upload, download, and integration
✅ **Lab-specific workflow** - Dataset and model management
✅ **Best practices** - Security, versioning, and organization

Key Takeaways

1. **HF Hub is purpose-built for ML research** - Not just file storage
2. **Seamless version control** for large datasets and models
3. **Direct library integration** - Works with transformers, datasets
4. **Proper separation of concerns** - Code in GitHub, data in HF Hub
5. **Professional data management** enables reproducible research

Impact on Research Workflow

- **Eliminates data storage headaches** - No more email attachments
- **Enables true reproducibility** - Fixed dataset/model versions
- **Facilitates collaboration** - Team access to shared resources
- **Supports open science** - Easy sharing with research community

Next Steps

➡️ **Module 5: Weights & Biases Tracking** - Experiment management
➡️ Create your first dataset repository on HF Hub
➡️ Upload a model checkpoint from current project