

Anime Character Identifier - Project Report

Project Overview

The **Anime Character Identifier** is a hybrid deep learning-based image recognition system designed to identify anime characters from uploaded images. It retrieves character details, anime appearances, and streaming platforms where the anime is available. It combines multiple models to improve accuracy and handles both known and unknown characters using fallback mechanisms.

Project Workflow (End-to-End)

Phase 1: Project Initialization

- **Tech Stack:** Python, TensorFlow (DeepDanbooru), Streamlit (UI)
- Setup DeepDanbooru (TF model) and created a minimal Streamlit UI.
- **Functionality:**
 - Upload anime image
 - Process image using DeepDanbooru
 - Recognize the most probable character tag
 - Return and display character name

Phase 2: Model Upgrade - DeepDanbooru Integration

- **DeepDanbooru:** A tag-based classification model trained on the Danbooru image dataset.
 - Recognizes tags like character:naruto, series:naruto, hair_color:black.
 - Pretrained model loaded via `deepdanbooru.project.load_model_from_project()`.
 - Tags loaded from tags.txt

How DeepDanbooru Works:

1. **Preprocessing:** Image resized to 512×512 with white background padding.
2. **Normalization:** Pixel values scaled between 0–1.
3. **Model Inference:** Image passed through a ResNet-based CNN model.
4. **Tag Output:** Predicts probabilities for all possible tags.
5. **Character Extraction:** Filters tags starting with character: and returns the highest confidence name.

Phase 3: Character Metadata with Jikan API (MyAnimeList)

- Integrated **Jikan REST API** to fetch metadata.
- Input: Character name → Jikan API → Output:
 - Official image
 - Bio/About section
 - List of anime titles featuring the character

Phase 4: Hybrid Recognition with CLIP

- **Problem:** DeepDanbooru fails for rare or unlisted characters.

- **Solution:** Use **CLIP** to match visual similarity with a custom character image database.

How CLIP Works (Contrastive Language–Image Pretraining):

1. **Joint Embedding:** Learns to embed both images and text into the same vector space.
2. **Visual-Semantic Matching:** High cosine similarity between image and correct description.
3. **Used here:** Matches uploaded image embeddings with precomputed ones from a known set.

CLIP Integration Steps:

- Prepare a folder `character_db/` containing labeled face images.
- Generate image embeddings using `open_clip_torch`.
- Save embeddings to a `character_embeddings.pkl` file.
- If DeepDanbooru fails:
 - Generate embedding of the uploaded image
 - Compare with database using cosine similarity
 - Return the closest match

Related Function:

- `match_with_clip(image)` → Returns best matching character and confidence.

Phase 5: Streaming Info with AniList GraphQL API

- Query the **AniList API** to show where the anime is available to stream.
- Based on character → identify anime → run GraphQL query.
- Returns streaming platform links (Netflix, Crunchyroll, etc.)

Phase 6: Multi-Character Detection with YOLOv8 (Upcoming)

- Use **YOLOv8**, a state-of-the-art object detection model, to detect multiple characters.

How YOLOv8 Works:

1. **Image Input:** YOLOv8 divides the image into a grid.
2. **Object Prediction:** Predicts bounding boxes and classes for each grid cell.
3. **NMS (Non-Max Suppression):** Filters overlapping boxes with low confidence.
4. **Output:** List of bounding boxes with labels and confidence scores.
5. **In Our Case:**
 - Detect multiple anime faces
 - Crop regions and run each through DeepDanbooru + CLIP fallback

YOLO Implementation (Planned):

```
from ultralytics import YOLO
model = YOLO("yolov8n.pt")
results = model.predict(image)
```

Recognition Pipeline Summary

Input Image → [DeepDanbooru]

↓ If confident

Character Name



Fetch Bio (Jikan)



Show Anime → Fetch Streaming



[CLIP fallback]



Match from embeddings



Fetch Bio → Show Info

File Breakdown

File	Purpose
app.py	Streamlit frontend UI logic
recognizer.py	DeepDanbooru + CLIP-based recognition
fetch_bio.py	Calls Jikan API for bio and anime appearances
streaming_info.py	Uses AniList API to get streaming links
fallback_matcher.py	CLIP fallback matcher implementation
character_gallery.py	Fetches character images from API or Google
models/	Contains DeepDanbooru model files
character_db/	Character images used for embedding comparison
character_embeddings.pkl	Precomputed embedding vectors

How to Run






Step 1: Clone and install

```
pip install -r requirements.txt
```

Step 2: Run the app

```
streamlit run app.py
```

Future Work

-  Add YOLOv8 for multi-character detection
-  Show bounding boxes in UI
-  Enable user-submitted image corrections to retrain the model
-  Add FAISS indexing for faster embedding search
-  Implement real-time feedback improvement for predictions



Authors & Credits

- Developed by **Team 18**
- Powered by **DeepDanbooru**, **CLIP**, **AniList API**, **Jikan API**, and **Streamlit**



This report serves as comprehensive documentation for contributors, researchers, or developers exploring anime-based image recognition systems.