

# 딥러닝 중간보고서

전체 Page의 배경과 캐릭터 일관성을 유지하는  
Manga Colorization 과 Translation

학번: 2024193011

이름: 신재완

2025년 11월 29일

## Contents

<b>1 SBERT를 통한 GUI Context Filtering 진행 과정</b>	<b>2</b>
1.1 서론 . . . . .	2
1.2 연구 방법 . . . . .	2
1.3 진행 과정 및 실험 결과 . . . . .	2
1.3.1 Model 설정 . . . . .	2
1.3.2 Weighting Factor 추가 . . . . .	2
1.3.3 Threshold와 LLM 주도 Query, displayableAppNames, topK 구현 . . . . .	3
1.4 연구 마무리와 주제 변경 . . . . .	4
<b>2 전체 Page의 배경과 캐릭터 일관성을 유지하는 Manga Colorization과 Translation</b>	<b>4</b>
2.1 서론 . . . . .	4
2.2 MangaDiT, MangaNinja . . . . .	5
2.3 Sequential Nano Banana Pro . . . . .	6
2.3.1 긴 연산 시간 . . . . .	8
2.3.2 페이지가 누적됨에 따른 일관성 붕괴 . . . . .	8
2.4 향후 계획 . . . . .	9

## 1. SBERT를 통한 GUI Context Filtering 진행 과정

### 1.1. 서론

YeYa macOS라는 OS 기반 Agent를 구현하던 중 심각한 병목 현상이 발생했다. AXUIElement를 통해 UI를 읽어오는 과정에서 전체 보이는 window에 대해 XML로 전환 시 약 5만 input token이 발생하여 비용이 많이 발생하고 속도도 느려지는 문제가 있었다. 이러한 문제를 해결하기 위해 user query에 따라 UI 요소를 filtering하는 시스템을 구축했다.

### 1.2. 연구 방법

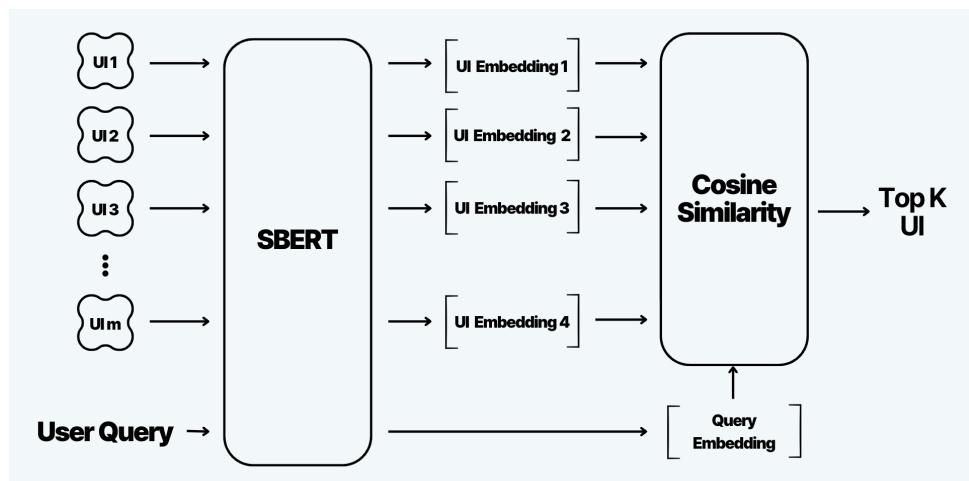


Figure 1: GUI Context Filtering 시스템 구조

Figure 1와 같이 시스템을 설계했다. 먼저 UI 요소들과 Query 각각을 SBERT를 통해 embedding으로 변환한다. 이후 Query embedding과 UI embedding 각각에 대해 cosine similarity 연산을 수행하여 Top K개만큼의 UI만 선별한다.

### 1.3. 진행 과정 및 실험 결과

#### 1.3.1. Model 설정

초기에는 all-MiniLM-L6 모델을 사용하여 embedding을 생성하려고 시도했으나 제한적인 언어만 지원하는 것을 확인했다. 그에 따라 distiluse-base-multilingual-cased-v1 모델로 변경했다. Huggingface에서 모델을 다운로드 받고 Swift에서 사용하기 위해 Core ML 구조로 변환했다.

#### 1.3.2. Weighting Factor 추가

중간 구축 이후 여러 테스트에서 과도하게 동일한 텍스트에만 집중하는 현상을 발견했다. 예를 들어 셔틀버스 예약 페이지에서 “셔틀 버스 예약해줘”라는 Query에 대해 정작 중요한 Button, Link에는 ‘셔틀’, ‘버스’, ‘예약’이라는 단어가 없어 Score가 낮고 아무 Action이 없는 텍스트만 Score가 높게 나와 Agent의 수행 능력이 떨어지는 문제를 발견했다.

그에 따라 UI 종류에 따른 Weighting Factor를 추가하여 Button, Link, Input 등 Actionable한 UI의 Score를 높게 설정했다. 이를 통해 보다 높은 Action 능력을 확인할 수 있었다.

Listing 1: UI Type별 Weighting Factor 설정

```
private let uiTypeWeights: [String: Double] = [
    "app": 2.0,           // Always show apps
    "application": 1.5,
    "button": 1.5,        // Interactive elements are important
    "radio button": 1.3,
    "checkbox": 1.3,
    "link": 1.5,
    "text field": 2.0,
    "text": 0.8,          // Static text is less important
    "static text": 0.8,
    "image": 0.7,
    "group": 0.6,
    "scroll area": 0.6
]
```

### 1.3.3. Threshold와 LLM 주도 Query, displayableAppNames, topK 구현

이후 시도에서 항상 일관적인 topK를 설정하니 상황에 따라 UI 요소가 과도하게 많이 들어가거나 요소가 부족한 상황이 발생했다. 예를 들어 전체 화면을 분석해야 하는 경우 큰 Top K가 필요한데 검색 버튼을 눌러야 하는 경우에는 적은 Top K이어도 충분하다. 이런 경우를 유동적으로 반영할 수 없었다.

또한 점수가 매우 낮아 무의미한 내용을 filter하기 위해 Score에 threshold 0.1을 설정했다. 그리고 경우에 따라 읽어야 하는 App이 다를 수 있다. 전체 화면을 분석하려면 모든 App을 읽어야 하지만 일부 App만 조작한다면 해당 App만 전달하면 된다.

최종적으로 LLM이 주도적으로 searchQuery, displayableAppNames, topK를 설정하도록 구현했다. 이를 통해 평균적으로 2천 input token으로 줄일 수 있었다. Input token이 줄어들고 관련된 UI만 LLM에게 전달하여 보다 빠르고 정확한 Action을 수행할 수 있었다.

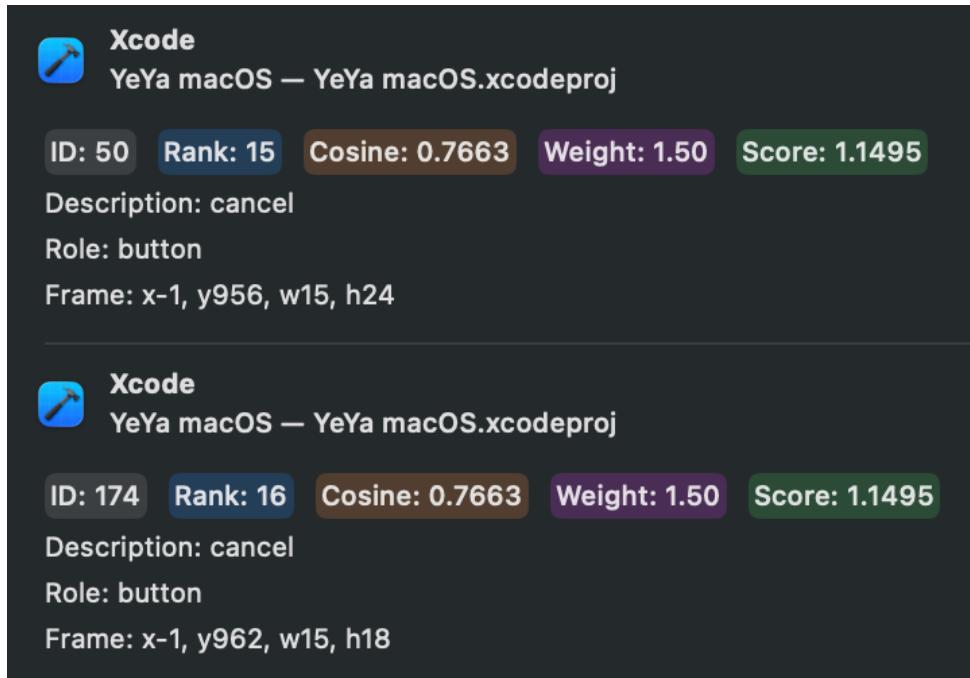


Figure 2: Score 기반 Filtering 결과

#### 1.4. 연구 마무리와 주제 변경

해당 방법을 통해 충분히 잘 작동하는 시스템을 구축했다. 그러나 학습을 시키거나 성능 지표를 정하기에는 딥러닝을 통한 방법론을 심화시키기 어렵다고 판단하여 여기까지 진행하고 새로운 주제를 연구하기로 결정했다.

## 2. 전체 Page의 배경과 캐릭터 일관성을 유지하는 Manga Colorization과 Translation

### 2.1. 서론

Manga, cartoon, manhwa와 같은 작품을 제작할 때 채색을 하지 않은 line drawing이나 monochrome manga에 비해 full colored manga는 시간이 2배 이상 오래 걸린다. 그에 따라 현재까지도 많은 작품들이 monochrome manga로 출판되고 있다.



Figure 3: Manga에서의 의성어, 의태어, 말풍선 예시

또한 manga에서 다국어로 번역을 하는 것도 시간이 오래 걸리는 작업이다. 단순 텍스트 기반으로는 처리할 수 없는데, Figure 3과 같이 의성어, 의태어, 말풍선이 그림과 하나가 되어 있기 때문이다. 일반적으로는 텍스트를 번역하는 역본 작업과 번역된 텍스트를 그림으로 삽입하거나 그림을 수정하는 식질 작업을 거치게 된다.

최근 여러 생성형 AI 모델이 등장하면서 이 두 가지 문제를 한 번에 해결할 수 있게 되었다. 하지만 세부적으로는 해결해야 하는 문제가 많이 남아있다.

## 2.2. MangaDiT, MangaNinja

가장 먼저 시도해보려 한 것은 현재 가장 좋은 모델을 실행시켜보는 것이었다. 가장 최근에 나온 Line Art Colorization SOTA 모델로는 MangaDiT [1]와 MangaNinja [2]가 있다.

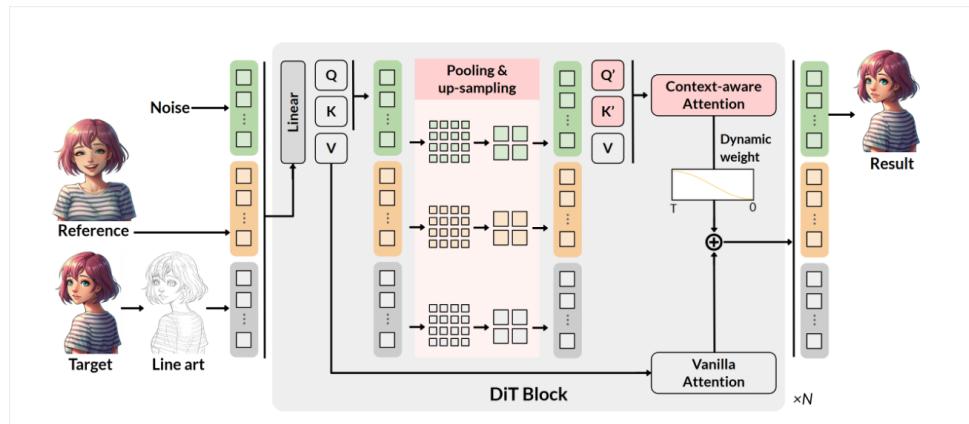


Figure 4: MangaDiT 모델 구조

MangaDiT는 FLUX1.dev를 기반으로 transformer를 사용했으며 NVIDIA A100 80GB를 통해 학습되었다. RTX 3070 8GB로는 실행이 불가능해 보여 MangaNinja를 설치해봤다.

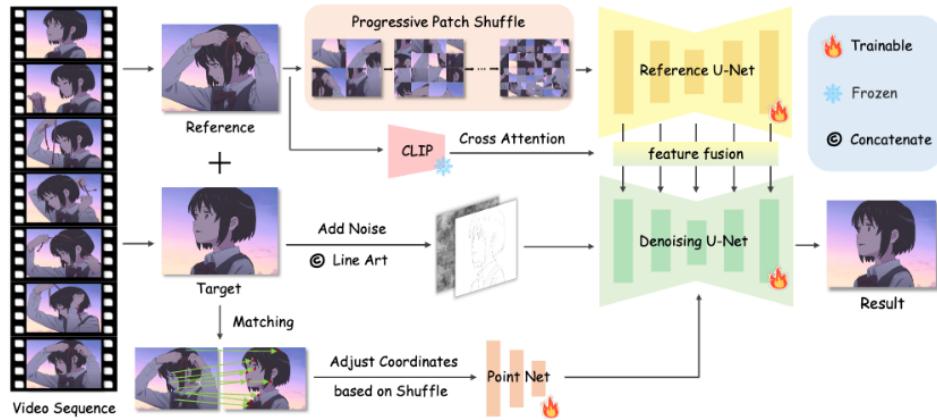


Figure 5: MangaNinja 모델 구조

MangaNinja는 NVIDIA A100 80GB 8장으로 하루 동안 학습되었다고 하지만 6GB VRAM으로 inference는 가능하다고 하여 설치해봤다. Stable Diffusion 1.5 기반이라 약 30GB 크기의 모델을 설치하고 inference 해보니 한 장에 30분이 소요되었다.

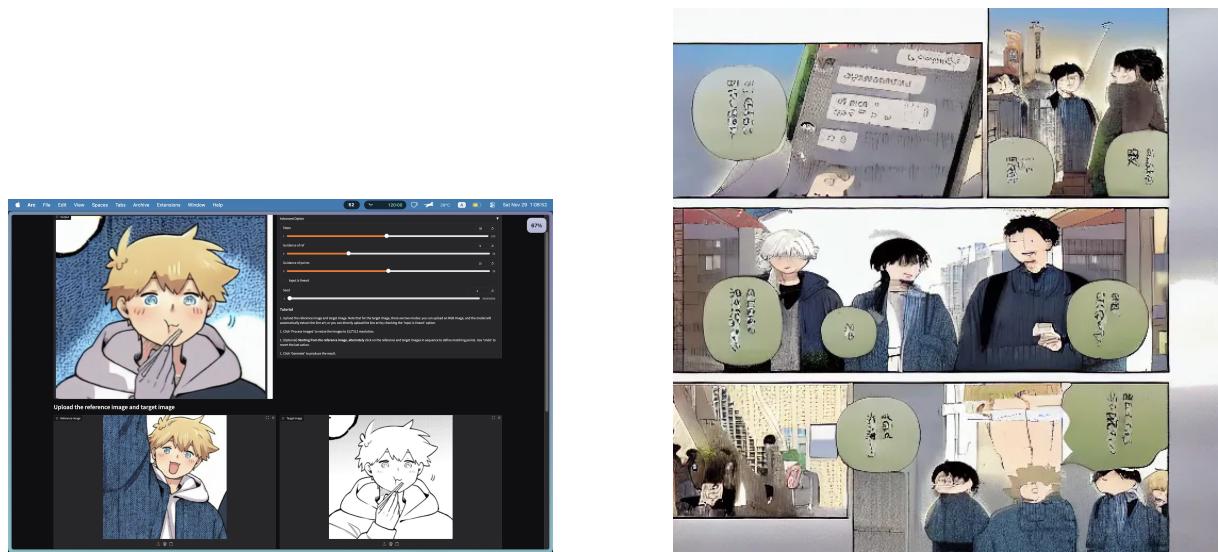


Figure 6: MangaNinja 실험 결과

또한 단일 캐릭터 한 명으로는 어느 정도 동작하는데 전체 panel을 한 번에 넣었을 때는 모든 것들이 뭉개지는 것을 확인할 수 있었다.

### 2.3. Sequential Nano Banana Pro

Open source 모델로는 학습을 하거나 inference 하는 것 자체가 현실적으로 어려워 보여 API를 통한 시스템 구축을 해보기로 했다. 2025년 11월 20일에 출시한 Nano Banana Pro의 image edit이 일관성을 유지하며 높은 수준의 편집 기능을 제공하여 사용해봤다.

실험에 사용한 manga는 蒼川なな(Aokawa Nana) 작가의 「合コンに行ったら女がいなかった話」(How I Attended an All-Guy's Mixer) [6] 시리즈이다. 해당 작품은 pixiv에서 연재되고 있는

창작 만화로, 총 18페이지로 구성된 에피소드를 실험에 활용했다.



Figure 7: Nano Banana Pro 채색 및 번역 결과 비교

Figure 7와 같은 결과가 나왔다. 채색도 거의 완벽하고 의성어, 의태어, 말풍선의 번역도 잘 되는 것을 확인할 수 있다. 사용한 프롬프트는 다음과 같다.

#### Listing 2: 채색용 프롬프트

```
Colorize all the manga pages. Maintain consistency of character.  
Maintain speech balloons and texts. Do not modify speech balloons  
and texts. Maintain the grid.
```

#### Listing 3: 번역용 프롬프트

```
Translate this Japanese manga page into Korean. Transform all the  
Japanese character into Korean character. Maintain the shape of  
speech balloons and other figures and background and grids. Also  
translate all the onomatopoeia into Korean. You have to translate  
only speech balloons, onomatopoeia into Korean.
```

하지만 2가지 문제가 발견되었다.

### 2.3.1. 긴 연산 시간

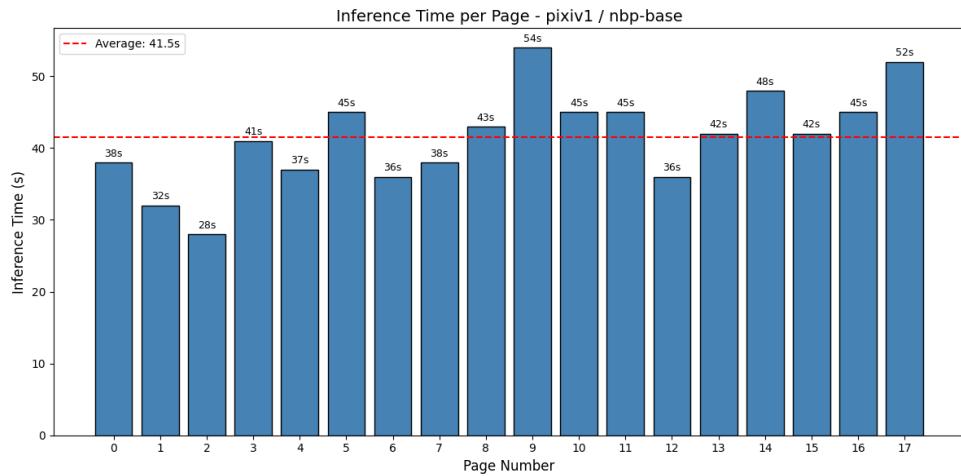


Figure 8: Nano Banana Pro 페이지별 연산 시간

현재 가장 간단하게 구현한 방식으로는 chat session에서 지속적으로 image를 누적시키면서 수정하도록 했다. 이를 통해 전체 18 page에서 캐릭터의 일관성을 유지시킬 수 있었다. 하지만 문제는 sequential하게 처리해야 하기 때문에 page 수에 선형적으로 비례하여 전체 채색 시간이 늘어나게 된다. 한 페이지 연산에 평균적으로 41.5초가 걸리는데 18쪽이면 약 12분이 소요되는 것이다.

### 2.3.2. 페이지가 누적됨에 따른 일관성 붕괴



Figure 9: 일관성 붕괴 사례 1: 캐릭터 외형 변화

페이지가 누적됨에 따라 여러 부분에서 캐릭터의 일관성이 무너진다. Figure 9에서는 3명의 인물의 머리 색깔이나 옷이 계속해서 바뀌는 것을 확인할 수 있다.



Figure 10: 일관성 봉괴 사례 2: 피부색 변화

Figure 10에서는 같은 캐릭터의 피부색이 계속 어두워지다가 갑자기 밝아지는 현상도 있었다.



Figure 11: 일관성 봉괴 사례 3: 말풍선 손실 및 장면 변형

Figure 11에서는 빨간색 강조 표시 부분처럼 말풍선이 아예 사라지거나 초록색 강조 표시처럼 아예 완전히 다른 장면이 삽입되기도 했다. 또한 많은 경우에서 파란색 강조 표시처럼 표정이 조금씩 바뀌는 경우도 빈번했다.

#### 2.4. 향후 계획

본 연구의 향후 계획은 다음과 같다.

- Batch 처리를 통해 전체 page를 훨씬 빠르게 채색하는 시스템을 구축할 계획이다.
- 현재는 코드 없이 모델 테스트 페이지에서 일일이 직접 응답을 보내고 받는 방식으로 테스트 했지만 모든 과정을 자동화하여 end-to-end 시스템을 구축할 계획이다.
- 여러 언어에 대한 번역 이미지도 자동적으로 생성되는 시스템을 구축할 계획이다.
- 일관성이 보다 잘 유지되기 위한 프롬프트 개선 및 시스템을 구축할 계획이다.

## References

- [1] MangaDiT Authors, *MangaDiT: Interactive Manga Colorization via Diffusion Transformer*, arXiv preprint, 2025.
- [2] MangaNinja Authors, *MangaNinja: Line Art Colorization with Precise Reference Following*, arXiv preprint, 2025.
- [3] Reimers, N., & Gurevych, I., *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019.
- [4] Black Forest Labs, *FLUX.1: A Rectified Flow Transformer for Image Generation*, 2024.
- [5] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B., *High-Resolution Image Synthesis with Latent Diffusion Models*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.
- [6] 蒼川なな (Aokawa Nana), 「合コンに行ったら女がいなかった話」(How I Attended an All-Guy's Mixer), pixiv, 2025. <https://www.pixiv.net/en/artworks/136886858>