Project Checkpoint #3

Kristen Vinh
kv1895@rit.edu

I. RECENT PROGRESS

A. Developing YOLO Cropping Models

I identified two YOLO models for extracting sweaters from images: Yolov8n-Seg and YOLOv8n-Pose. When put into use, both perform similar functions in terms of person identification and masking; however, YOLOv8n-Pose attempts to determine shoulder and hip points for more precise person identification. I then cropped to the torso and removed background, if possible, with both models. These performed similarly, although the YOLO pose occasionally yielded a more accurate crop, as shown in Fig. 1. In addition, when YOLOv8n-Pose failed to find shoulder and hip points, it used Yolov8n-Seg as a backup cropping method.

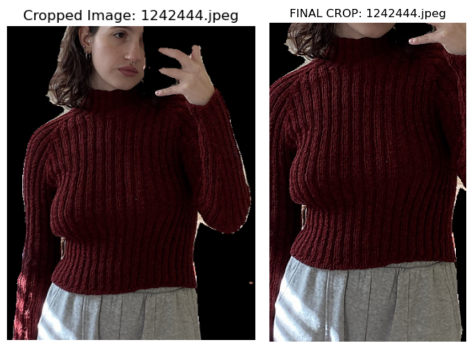


Fig. 1 Example of YOLO segment (left) vs. YOLO pose (right) on a Sample Sweater

If either YOLO model failed to obtain a crop or failed to mask out the background, it was designed to fall back to the original image for feature extraction. Some details were occasionally lost in either model, most frequently, sleeves and collars.

B. Feature Extraction Models for Evaluation

After completing YOLO cropping, I began testing additional feature extraction models beyond ResNet50. While the feature extraction code took a long time to run, the models themselves had similar code bases; only a few code changes were needed to switch between them, allowing me to test twelve final models as seen in Table I. All were then compiled using the same Approximate Nearest Neighbors (ANN) index, HNSWlib, as it had performed the best in previous tests.

TABLE I. FEATURE EXTRACTION MODEL AND CROPPING MODEL COMBINATIONS TESTED

Feature Model (A)	Cropping Method (B)
ResNet50	None
ResNet50	YOLO-seg
ResNet50	YOLO-pose
CLIP-base	None
CLIP-base	YOLO-seg
CLIP-base	YOLO-pose
CLIP-large	none
CLIP-large	YOLO-seg
CLIP-large	YOLO-pose
DINOv2	None
DINOv2	YOLO-Seg
DINOv2	YOLO-Pose

Each of these recommendation algorithms worked similarly, modeled after the code used in [1]:

1. An HNSWlib index is built with mean feature vectors extracted from nearly 10,000 sweaters from Ravelry.
    a. These features are extracted by first using the cropping method listed in column “Cropping Method (B)” in Table I, if applicable.
    b. It then uses the “Feature Model (A)” to extract features using the pre-trained library listed. If multiple sweater images are present for a given feature, it creates a mean feature vector across all photos as well as a Pickle file with the corresponding pattern IDs for each vector.
    c. Finally, it utilizes the HNSWlib library to construct a cosine space index, which occurs quickly.

2. To use the index for testing (and for later in the online model), the test image is processed through the same feature extraction and cropping pipeline to generate a single feature vector.
    a. That vector is then used to search the generated index and returns the top k vectors, which are the most similar to the input vector in the index.
    b. It then uses the Pickle file to find the corresponding pattern ID.

C. Initial Analysis of ANN Extraction Methods

I needed a better way to compare these twelve models and narrow them down to the best-performing ones, so I designed three sets of testing metrics:

1. Initial Build Metrics (Table II)
    i. Feature Extraction Time over 10,000 Sweaters (includes cropping time)
    ii. Index Build Time
    iii. Index Size on Disc
2. Accuracy Metrics (TABLE III)
    i. Index Recall: Internal accuracy of the HNSWLib Index
    ii. Retrieval Accuracy: Accuracy of the index when given “real-life” images, in this case, sample sweaters knit from patterns in the index
3. Query Time (Table IV): The time it takes a function to make 10 recommendations for one uploaded image.

The Index Recall is calculated as follows:

- A sample of 1,000 mean vectors is taken from the index. For each vector, a brute-force search is performed using sklearn.NearestNeighbors to find its 100% correct top-10 nearest neighbors. This list of nearest neighbors serves as the ground truth against which the index search is compared.
- Then, the same mean vectors are queried against the HNSWlib index (an approximate nearest neighbors search) to find the approximate nearest 10 neighbors.
- The final score is the average of the top 10 ground truth neighbors found in the HNSWlib results, averaged across all 1,000 mean vectors.

For Retrieval Accuracy, I selected 50 random pullover patterns and 50 random cardigan patterns from the database of sweaters I had downloaded from Ravelry.

To then obtain “real-life” images, I downloaded images of sample knit projects from these patterns on Ravelry. I chose projects with minimal modifications (since some sweaters alter sleeve length, colorwork, etc., from the original pattern), but I tried to include a range of image types (on-body, off-body, etc.). I also aimed to capture clear, well-lit images, as the final web application will emphasize the importance of high-quality photos. The final metric is the percentage of the 100 patterns where the actual pattern was recommended when a picture of the sample project sweater was queried against the index.

TABLE II. BUILD METRICS

Feature Model (A)	Cropping Method (B)	Extraction Time (in seconds)	Build Time (in seconds)	Index Size
ResNet50	None	7,593.10	4.85	82.5 MB
ResNet50	YOLO-seg	14,550.22	4.85	82.5 MB
ResNet50	YOLO-pose	17,801.55	4.78	82.5 MB
CLIP-base	None	5,301.27	1	21.7 MB
CLIP-base	YOLO-seg	9,892.19	0.98	21.7 MB
CLIP-base	YOLO-pose	17,176.41	1.03	21.7 MB
CLIP-large	none	27,568.38	1.51	31.8 MB
CLIP-large	YOLO-seg	34,681.04	1.5	31.8 MB
CLIP-large	YOLO-pose	40,845.73	1.51	31.8 MB
DINO	None	10,445.79	1.58	31.8 MB
DINO	YOLO-Seg	16,668.57	1.84	31.8 MB
DINO	YOLO-Pose	21,166.62	1.41	31.8 MB

TABLE III. ACCURACY METRICS

Feature Model (A)	Cropping Method (B)	Recall @ 10 (existing indexes)	Average Recall (Sample Sweaters)
ResNet50	None	95.56%	7.00%
ResNet50	YOLO-seg	96.32%	8.00%
ResNet50	YOLO-pose	95.32%	6.00%
CLIP-base	None	98.94%	3.00%
CLIP-base	YOLO-seg	99.13%	3.00%
CLIP-base	YOLO-pose	99.12%	4.00%
CLIP-large	none	99.44%	9.00%
CLIP-large	YOLO-seg	99.38%	3.00%

CLIP-large	YOLO-pose	99.42%	4.00%
DINO	None	99.01%	5.00%
DINO	YOLO-Seg	99.57%	8.00%
DINO	YOLO-Pose	99.61%	10.00%

TABLE IV. QUERY TIME

Feature Model (A)	Cropping Method (B)	Average Query Time (in milliseconds)
ResNet50	None	10.71
ResNet50	YOLO-seg	9.73
ResNet50	YOLO-pose	10.39
CLIP-base	None	2.50
CLIP-base	YOLO-seg	2.55
CLIP-base	YOLO-pose	2.68
CLIP-large	none	2.40
CLIP-large	YOLO-seg	2.56
CLIP-large	YOLO-pose	2.67
DINO	None	3.46
DINO	YOLO-Seg	6.64
DINO	YOLO-Pose	6.70

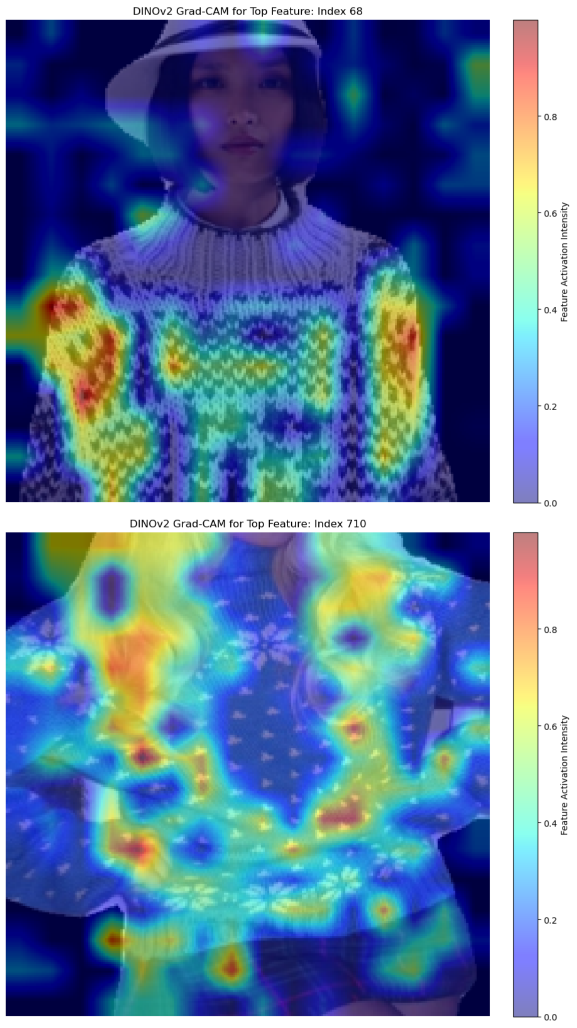
The Retrieval Accuracy ended up being relatively poor. My theory is that this is happening because many similar sweaters exist, and while some of the models can find relatively similar examples, they can't always find an exact match. I also theorized that the cardigan metric was consistently lower than pullovers because often, cardigans have shirts underneath that might skew the features extracted and, therefore, the recommendations.

Therefore, I decided to prioritize the models with the highest "Retrieval Accuracy" (even though it was low) and the highest "Index Recall," while ensuring that I had one of each of the feature extraction models and one of each of the cropping models for real-life evaluation: ResNet50 with YOLO-SEG, CLIP Large with no crop, and DINO with YOLO-POSE. This ensured that I could see if a slightly less accurate feature extraction model (such as ResNet50), which had a shorter feature extraction time, could hold up to the more accurate but longer extraction-time models (CLIP Large, DINOv2).

D. *Explainable AI Model Development*

For each of the top three models, explainable AI scripts were written to highlight the features that each model extracted and used in the index. These scripts for generating Grad-CAM explanations were developed with the assistance of Google's Gemini, using the tf-keras-vis and pytorch-grad-cam libraries.

For each test sweater, a heatmap was generated to highlight the features. This does not mean these were the only features used, as models were relatively complex with many layers; they give an idea of features used in the index and provide an idea of where a model might be using pose or irrelevant features (hair, belts, background objects) that might lead to a decent or poor match, as seen in Figure 2.


DINOv2 Grad-CAM for Top Feature: Index 68
0.8
0.6
0.4
0.2
0.0
Feature Activation Intensity
DINOv2 Grad-CAM for Top Feature: Index 710
0.8
0.6
0.4
0.2
0.0
Feature Activation Intensity
Fig 2. Features heatmaps on two different test images

E. *Real-life Example Evaluation*

In total, I ended up with 75 real-life Reddit examples to test. I evaluated the criteria in Table V, including the use of the Grad-CAM heatmap. The Grad-CAM heatmap helps to explain when the model makes matches based on pose or other features, such as sweater features. It is therefore useful here as a metric to determine if a model is effectively extracting relevant features. This was tested on five recommendations for each test sweater. 63 samples downloaded from Reddit were determined to be viable, as sweaters were excluded if I could not determine their sweater type, neck type, sleeve features, or other additional features.

TABLE V. FINAL REAL-LIFE SWEATER EVALUATION METRICS

Metric	Description	Additional Details	Points
Sweater Type	Do the recommendations match the sweater type of the pattern (cardigan or pullover)?	Straightforward. Drop example if you can't tell, given the benefit of the doubt if it's a back shot that it's a cardigan.	5
Neck Type	Do the recommendations match the type of neck (v-neck, crew, shawl, etc.) of the pattern?	V-neck, standard neck (crew or yoke neck), wide neck or scoop neck, off shoulder, mock or turtle.	5
Sleeve Features	Do the recommendations have similar sleeve features (raglan, drop, bell) to the pattern?	Raglan, circular yoke, drop, set-in shoulder, /no sleeves.	5
Additional Features	Do the recommendations have similar features to the pattern, such as stockinette stitch, texture, type of colorwork, cablework, lacework, etc?		5
Explainable AI	Does the XAI Algorithm correctly extract features from the sample? 0 = no good features, 3= some good features, 5 = very good features		5

One downside of scoring based on features is that a sweater can score high (4 out of 5 matching a characteristic of the test sample), and still, a single good recommendation could be missing. Alternatively, the opposite could happen: a lot of the metrics could be poor, but the model could still succeed in giving at least one excellent recommendation. Both of these scenarios occurred; therefore, I added one additional metric to use alongside the scorecard: Is at least one sweater a "perfect" match, or close enough that I would personally recommend it? Ideally, I would have all these metrics scored by other people as well, but the average of my scores is in Table VI.

The DINO Pose model performed significantly better at recommending sweaters to the Reddit real-life samples across all five metrics. The results for DINO in Table VI can be interpreted as: more than 4 out of 5 recommendations, on average, matched sweater type; more than 3 out of 5 recommendations matched the neck type; roughly half matched the sleeve features, and more than 3 out of 5 matched other features like texture, cables, and colorwork type. In addition, an average score of 3 for explainable AI meant that most models had at least some relevant features identified using the Grad-CAM model.

I also identified 30 out of 63 recommendations as having at least a “perfect” match as produced by the DINOv2. In addition, many sweaters were close but did not have an exact match, as shown in Fig. 3, and were therefore excluded from this metric.

TABLE VI. QUALITATIVE SCORING AVERAGES (OUT OF 5)

Model	Sweater Type	Neck Type	Sleeve Features	Additional Features	Explainable AI
Resnet	3.97	2.86	2.11	2.16	2.90
DINO	4.60	3.11	2.49	3.27	3.25
CLIP	3.63	2.51	1.95	1.85	3.06

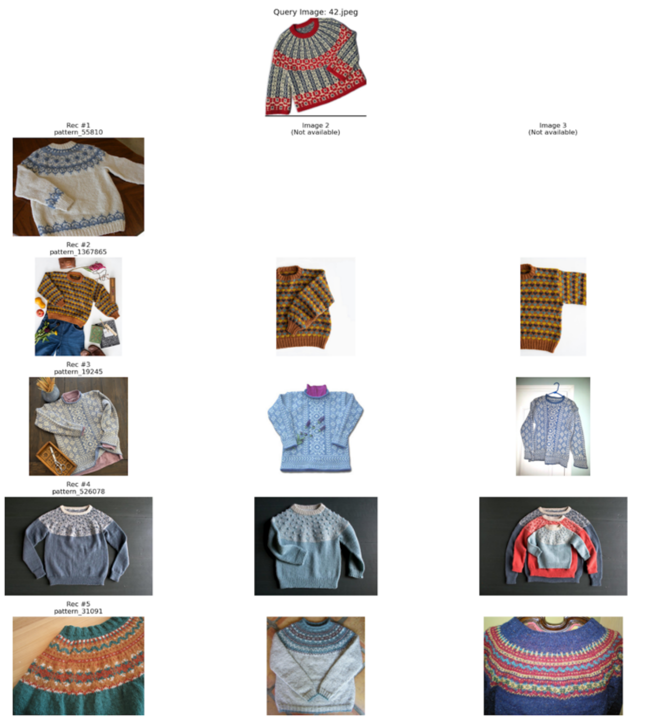


Fig. 3. Sweater recommendations with similar colorwork styles, but no sweater is a close enough match

Given DINOv2 and YOLO-Pose-Crop’s outperformance on both the sweater sample exact matching and the real-life test, I decided to build my final front-end model with the DINOv2 feature extraction, YOLO-Pose-Crop, and an HNSWlib index.

F. *Front-End Building*

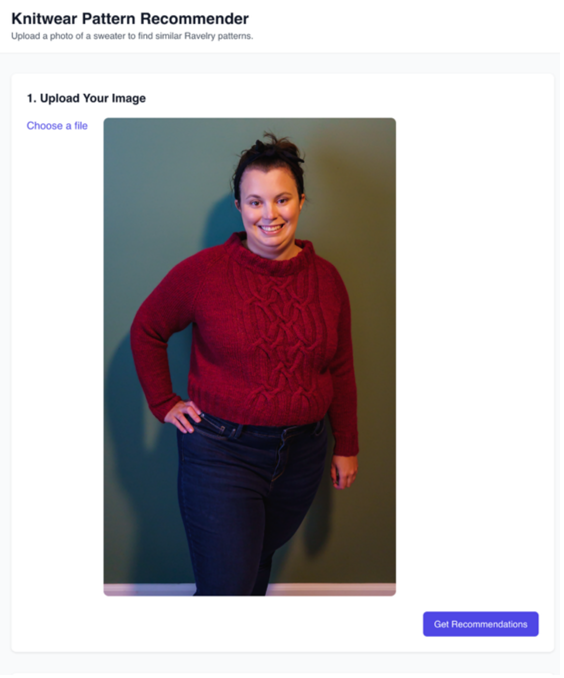
With the assistance of Google's Gemini AI, a front-end application was built using FastAPI. The script loads all necessary files:

- The HNSWlib (Hierarchical Navigable Small World) index, generated with the features from nearly 10,000 of the most popular sweaters on Ravelry
- The Dinov2 model (from Hugging Face) features an extracting script to extract features from the test image (and was used to extract features for the index)
- YOLOv8 scripts for segmentation, pose-based cropping, and background removal.
- An explainable AI script using GradCAM to display features extracted
- A Pickle file mapping the index's integer IDs to their corresponding Ravelry pattern IDs

When a user uploads an image, the following actions happen:

- The image is first processed by the YOLO crop script, which attempts to identify a human torso in the image and crops the photo to help extract just the sweater. This helps improve the accuracy of feature extraction because faces, pants, and backgrounds are removed.
- It then extracts features using the DINOv2 model and stores them as a vector. The vectors in the index were also extracted similarly, although they were created using a mean vector of all features from all available images of a sweater pattern in Ravelry.
- It then runs a similarity search using the HNSWlib index generated earlier and returns the top ten similar sweaters, retrieving their pattern ID from the Pickle file.
- Then it queries the Ravelry API to retrieve details for each pattern ID, including a photo, the pattern link, and the pattern name.
- Finally, it generates a Grad-CAM heatmap that visualizes the parts of the uploaded image the model relied on most for its features.

An example of the front-end interface is shown in Figs. 4 and 5. The app correctly displays the index matching the sweater pattern in recommendation #1.

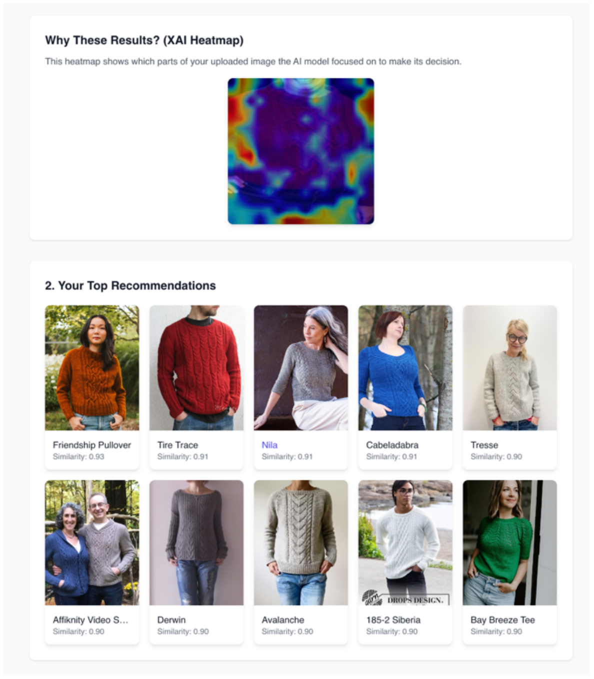


Knitwear Pattern Recommender

Upload a photo of a sweater to find similar Ravelry patterns.

1. Upload Your Image

Fig. 4 Upload Section of Front-End App



Why These Results? (XAI Heatmap)

This heatmap shows which parts of your uploaded image the AI model focused on to make its decision.

2. Your Top Recommendations

Friendship Pullover
Similarity: 0.93

Tire Trace
Similarity: 0.91

Nila
Similarity: 0.91

Cabeladabra
Similarity: 0.91

Tresse
Similarity: 0.90

Affiknity Video S...
Similarity: 0.90

Derwin
Similarity: 0.90

Avalanche
Similarity: 0.90

185-2 Siberia
Similarity: 0.90

Bay Breeze Tee
Similarity: 0.90

Fig. 5 XAI and Recommendation Portion of Front-End App

Gemini also assisted in developing front-end HTML and JavaScript to display these results in a front-end application, creating a webpage where users can easily upload an image and obtain results.

II. QUESTIONS/CONCERNS

- Is there a preferred way to cite Gemini code generation in my final project report?

- Do I need to have a fully functional front-end app available to others, or is a locally running version fine?

III. NEXT STEPS

1. Finish Front End App
    1. Improve look
    2. Add text further explaining the app's processes
3. Upload to web servers if necessary

REFERENCES

[1] Sridevi, M., Manikya Arun, N., Martha, Sheshikala, & Sudarshan, Dr. (2020). Personalized fashion recommender system with image based neural networks. IOP Conference Series: Materials Science and Engineering. 981. 022073. 10.1088/1757-899X/981/2/022073.