ML FASHION ITEM RETRIEVAL

CS- 308 MACHINE LEARNING
UNDER THE GUIDANCE OF DR. JIGNESH BHATT

AMISHA LALWANI- 202251013

Citatition Detail of Paper: LEARNING FASHION COMPATIBILITY ACROSS APPAREL CATEGORIES FOR OUTFIT RECOMMENDATION Luisa F. Polanía, Satyajit Gupte: [2019 IEEE International Conference on Image Processing (ICIP)]

PROBLEM STATEMENT

Goal: Assist small fashion businesses with personalized recommendation features.

Given: A query image by the user.

Objective: Learn a model that recommends compatible outfits.

Constraints: Limited training - 2000 outfits and 5000 outfits only in 80:20 train:validate ratio.

Approach/Technology Used:

- Siamese Network with VGG-16 backbone for feature extraction and color histogram fusion for better compatiblity learning.
- Instead of simple concatenation used hadamard product between the image features extracted from 2 images and the two color histograms then concatenated the final results.
- Fully connected DeepFCMetricNetwork for learning the non linear compatiblity relation between fashion items image.

DATASET DESCRIPTION

Dataset used: Polyvore Outfits Dataset

Train: Used train.json from non- disjoint set, split into 80:20 for training and validation. First I used only 2000 outfits to train but due to not good enough results re-trained for 5000 outfits.

Test: Used different outfits from test.json from non-disjoint set to test.

Machine Specification: Used Kaggle notebook with GPUP100.

The dataset consists of two types of sets: **disjoint** and **non-disjoint**:

In the non-disjoint set, some items (but not complete outfits) appear in both training and test splits. This allows models to leverage previously seen items to predict outfit compatibility.

In the disjoint set, the training and test splits have no overlapping items. This ensures that models must generalize to unseen outfits without relying on previously encountered fashion items.

Dataset statistics:

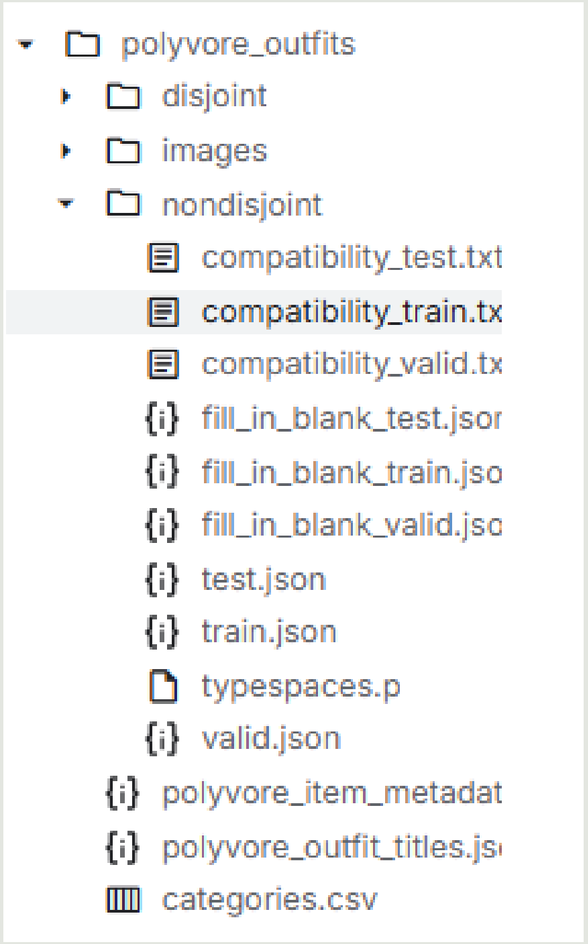
Non-disjoint set: 53,306 training, 5,000 validation and 10,000 testing outfits.

Disjoint set: 16,995 training outfits, 3,000 validation and 15,145 testing outfits.

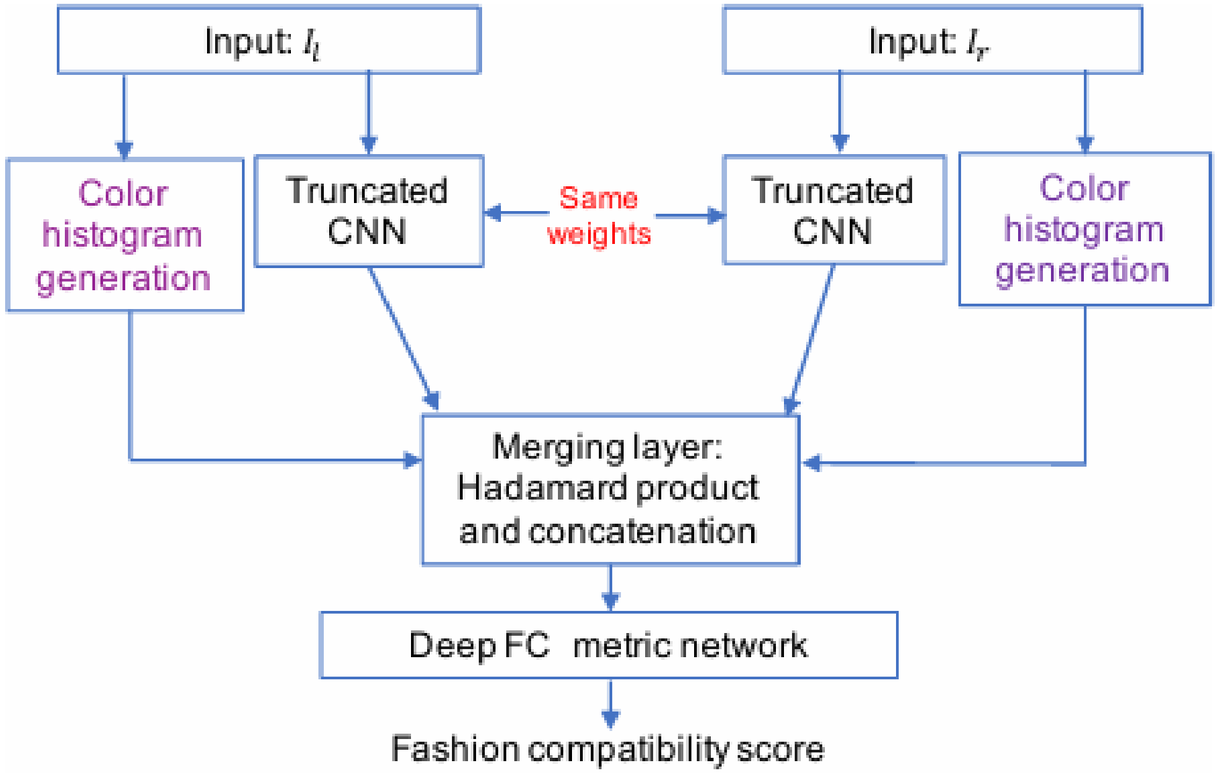
Each item has information in polyvore_item_metadata.json.

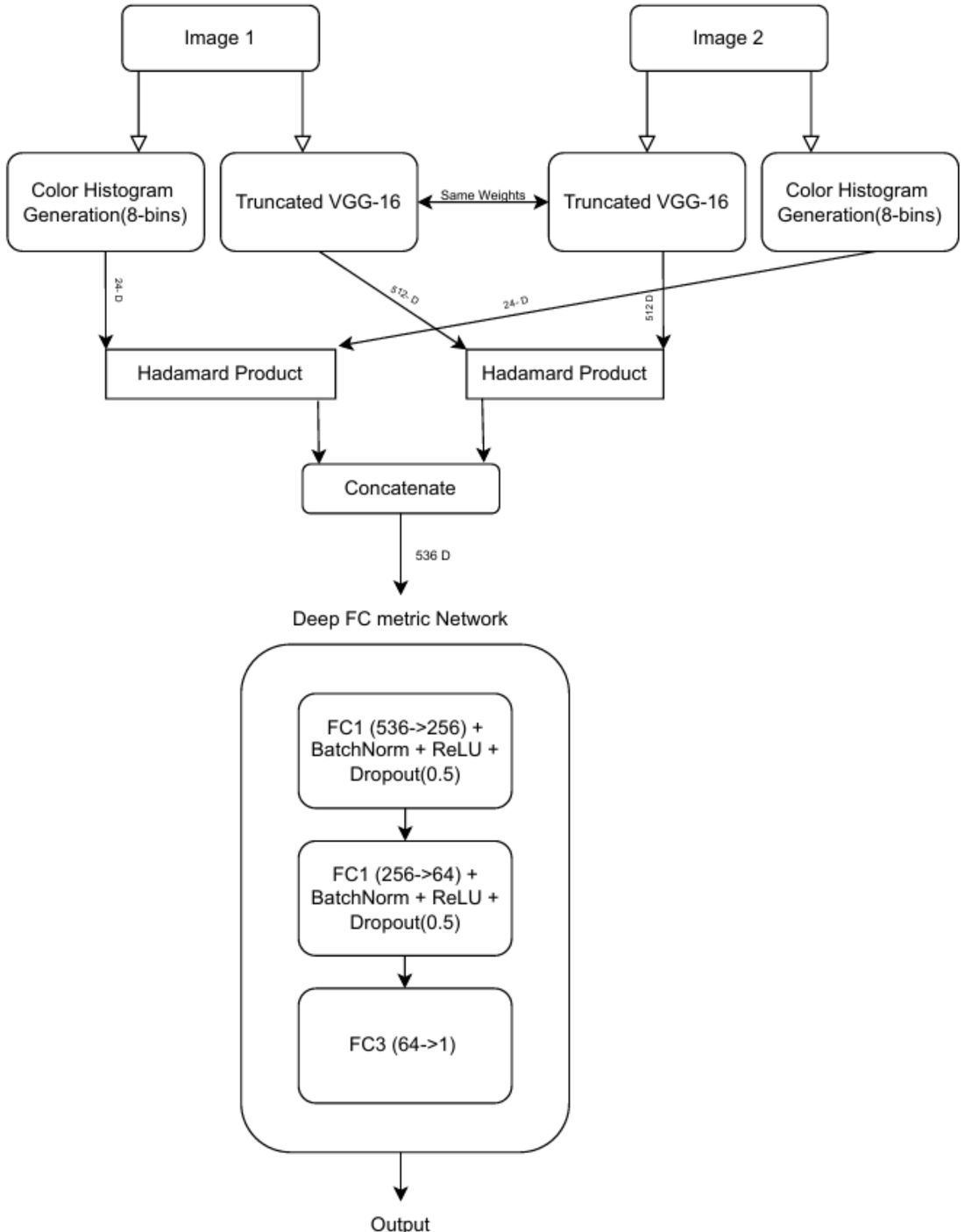
Category ids can be matched via categories.csv.

Each set has information in polyvore_outfit_titles.json.



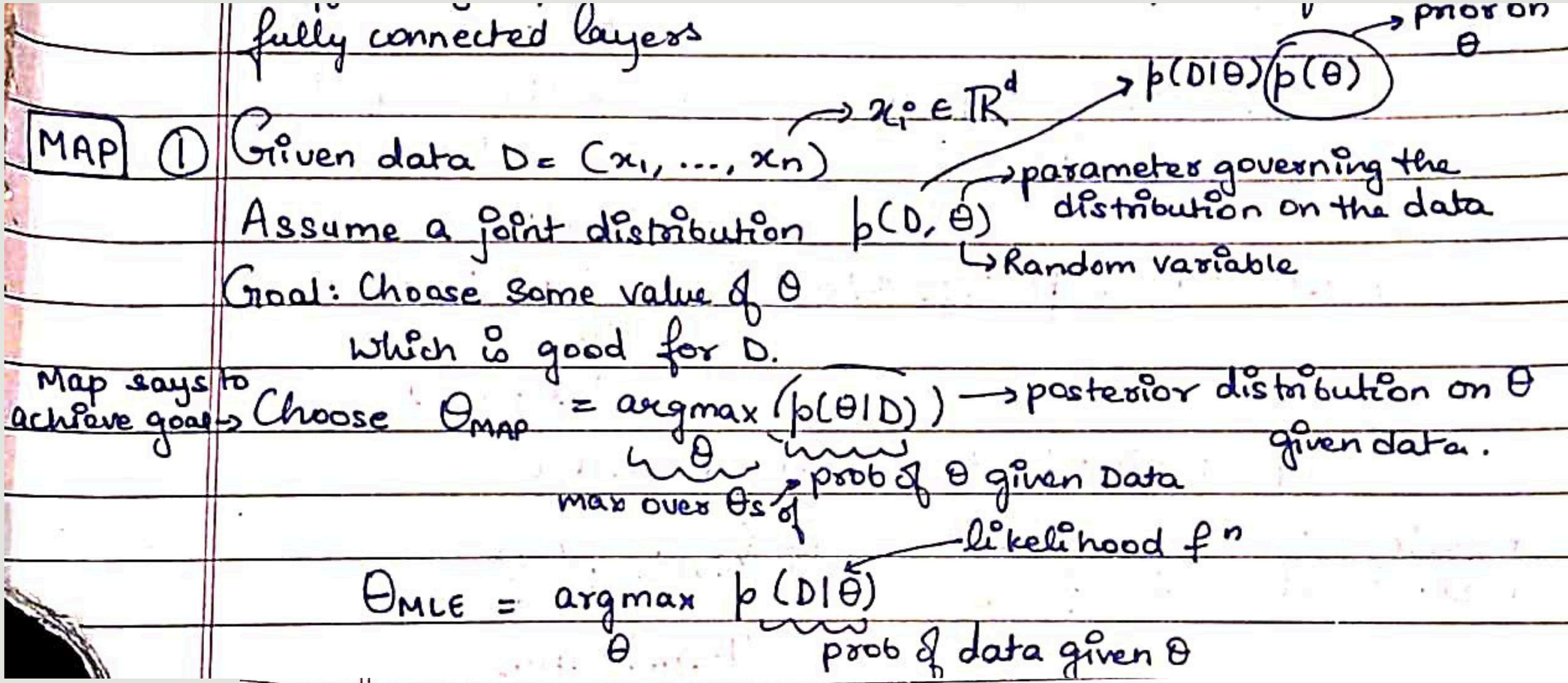
NEURAL NETWORK ARCHITECTURE





KEY TERMS TO KNOW

MAP- MAXIMUM APOSTERIORI PROBLEM



fully connected layers

MAP ① Given data $D = (x_1, \ldots, x_n)$ → $x_i \in \mathbb{R}^d$

Assume a joint distribution $p(D, \theta)$ → $p(D|\theta)$ $p(\theta)$ → prior on θ

parameter governing the distribution on the data

↳ Random variable

Goal: Choose some value of θ which is good for D.

Map says to achieve goal → Choose $\theta_{MAP} = \underset{\theta}{\text{argmax}} \, p(\theta|D)$ → posterior distribution on θ given data.

max over θs of → prob of θ given Data

likelihood fn

$\theta_{MLE} = \underset{\theta}{\text{argmax}} \, p(D|\theta)$

prob of data given θ

④ Hadamard product: Element-wise product

$$A = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 7 & 8 \\ 9 & 10 \end{bmatrix} \quad A \odot B = \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix} \odot \begin{bmatrix} 7 & 8 \\ 9 & 10 \end{bmatrix}$$

- Captures interaction btw corresponding dimensions. (like matching features) $= \begin{bmatrix} 7 & 16 \\ 36 & 50 \end{bmatrix}$

↳ helps to see what features overlap the most for compatibility

TRAINING PROCESS

N training I/P image pairs $I = \{(I_\sigma, I_\ell)_i\}_{i=1}^{N}$

I_ℓ : I/P to left branches of siamese sub-network.

I_r : I/P to right

$Y = \{y_i\}_{i=1}^{N}$ denoted binary labels

$y_i = 1$ if I/P pair (I_r, I_ℓ) → fashion compatible else 0.

u_i - O/P of last FC layer of metric sub-network when pair (I_r, I_ℓ) - i/p. Readout fn $J(\cdot)$ is

$$\hat{y}_i = J(x_i) = w^T x_i + \epsilon$$

w → Wts. of readout fn

ϵ → Noise has logistic distribution → standard logistic distribution.

Batch Normalization applied after every FC layer of metric sub network

Siamese sub net 2 set of wts θ_l, θ_r - but mirrored $\therefore \theta = \theta_l = \theta_r$

$\theta_s = \{\theta_t\}_{t=1}^{S}$ → subset of θ that correspond to S filters selected for fine tuning.

$W = \{W_j \in \mathbb{R}^{P_j \times Q_j}\}_{j=1}^{M}$ be the set of wts of FC metric sub network.

↳ Modeled with a Matrix-variate normal distribution of

$\therefore$ 0 mean, i.e. $W_j \sim MN(0, \Lambda_j, \Upsilon_j^2 I)$ $\forall j$ $0 \in \mathbb{R}^{P_j \times Q_j}$ is 0 matrix &

$\Upsilon_j^2 I \in \mathbb{R}^{P_j \times P_j}$ row covariance diag matrix.

The positive semi-definate col^n covariance matrix $\leftarrow \Lambda_j \in \mathbb{R}^{Q_j \times Q_j}$ - can be learned in order to capture correlation btw layer i/p units.

$\Lambda = \{\Lambda_j\}_{j=1}^{M}$ - set of col cov matrix

Network is trained by solving the following MAP prob^m

$$\hat{\theta}_s, \hat{W}, \hat{\Lambda}, \hat{w} = \underset{\theta_s, W, \Lambda, w}{\operatorname{argmax}} \; p(\theta_s, W, \Lambda, w | I, Y)$$

$$= \underset{\theta_s, W, \Lambda, w}{\operatorname{argmax}} \; p(Y | I, \theta_s, W, w) \times p(W|\Lambda) \times p(\theta_s) \times p(W) \quad (3)$$

likelihood probability — prior probabilities

noise (ϵ) - logistic
prob distrib^n of y_i given x_i - Bernoulli ∴ likelihood

$$p(Y|I, \theta_s, W, w) \propto \prod_{i=1}^{N} p(y_i=1|x_i)^{y_i} (1 - p(y_i=1|x_i))^{(1-y_i)} \quad (4)$$

Sigmoid fn

$$p(y_i=1|x_i)^{y_i} = \sigma(w^T x_i) = \frac{1}{(1 + \exp\{-w^T x_i\})}$$

Each entry of the vectors in the set θ_s & each entry of w is modeled with laplacian distribution of zero mean & variance σ_p^2, in case of θ_i & variance σ_w^2 in case of w

Motivation is to promote sparsity & ∴ reduce redundacy in the wts.

θ_s	Selected CNN filters	Laplacian $(0, \sigma_p^2)$	Promotes sparse filters
w	Readout weights	Laplacian $(0, \sigma_w^2)$	Promotes sparse decision logic

W_i - modeled by matrix-variate normal distribution with 0 mean, the prior prob $p(W|\Lambda)$ takes the form

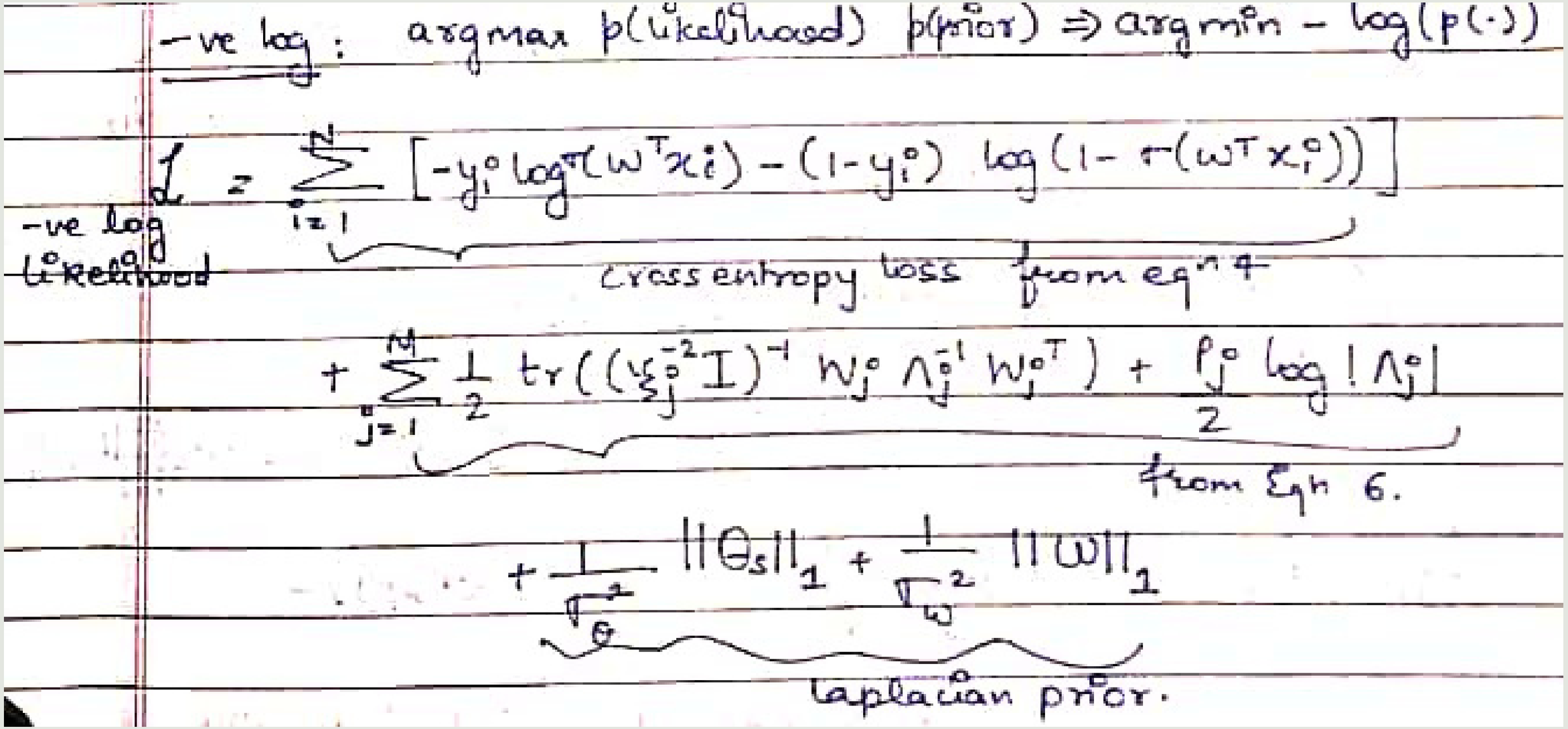
$$p(W|\Lambda) = \prod_{j=1}^{M} p(W_j|\Lambda_j) \quad (5)$$

$$= \prod_{j=1}^{M} \frac{\exp\left(-\frac{1}{2} tr\left(\left(\gamma_j^2 I\right)^{-1} W_j \Lambda_j W_j^T\right)\right)}{(2\pi)^{\frac{P_j Q_j}{2}} |\gamma_j^2 I|^{\frac{P_j}{2}} |\Lambda_j|^{\frac{Q_j}{2}}} \quad (6)$$

$tr(\cdot)$ - trace of matrix

$|\cdot|$ - determinant

Replacing 4,6 in 3 and taking the negative logarithm leads to the following optimization problem



LOSS FUNCTION

Term-1 : BCE LOSS-

Helps the model distinguish between compatible and incompatible pairs.

Term-2: Matric-Normal-Prior-

Regularizes Weight matric W penalizes high variance in weights promotes structured regularization guided by Λ_j - encorages weights to learn coherent embeddings

Term-3: CNN Sparsity Prior

Promotes sparsity in CNN filter weights, L1 norm makes many small weights to be 0 → prevents overfitting, improves generalization smaller inference time.

$$\hat{\mathbf{\Theta}}_s, \hat{\mathbf{W}}, \hat{\mathbf{\Lambda}}, \hat{w} = \underset{\mathbf{\Theta}_s, \mathbf{W}, \mathbf{\Lambda}, w}{\arg\min} - \sum_{i=1}^{N} [y_i \ln(r(w^T x_i)) + \quad (8)$$

$$(1 - y_i)\ln(1 - r(w^T x_i))] + \lambda_1 \sum_{j=1}^{M} \mathrm{tr}(W_j \Lambda_j^{-1} W_j{}^T) +$$

$$\lambda_2 \sum_{t=1}^{S} \|\theta_t\|_1 + \lambda_3 \|w\|_1 \ \text{ s. t. } \ \Lambda_j \succeq 0, \mathrm{tr}|\Lambda_j| = 1, \forall j,$$

where $\lambda_1, \lambda_2, \lambda_3$ are regularization parameters, which are incorporated to tune the strength of the regularization terms, and are estimated using grid search. The constraint $\Lambda_j \succeq 0$ comes from the positive semi-definite property that covariance matrices need to satisfy. An alternating optimization procedure

Term-4: Classifier Sparsity Prior

Forces feature selection by suppressing irrelevant features-> makes decision boundary rely only on key features.

RESULTS- QUALITATIVE AND QUANTITATIVE

To measure result quantitatively I used Precision@K and Lift@K as explained in the paper.

Let $\{\psi_n\}_{n=1}^{N_t}$ be the set of testing Polyvore outfits, where ψ_n is formed by the query item (first item in the outfit), denoted as q_n, and C_n complementary items, denoted as $\{o_n^{(c)}\}_{c=1}^{C_n}$, which belong to C_n different apparel categories. Let $R_n^c(K)$ denote the top K recommendations generated by the proposed network for the complementary item in category c, given the query q_n. To generate $R_n^c(K)$, pairs are first formed between the query item and all the rest of the items in the Polyvore dataset which belong to category c, then the items with the top K fashion compatibility scores are selected to form $R_n^c(K)$. The precision@K for outfit ψ_n is

$$\text{precision@}K(\psi_n) = \frac{1}{C_n}\sum_{c=1}^{C_n}\mathbf{1}[o_n^{(c)} \in R_n^c(K)], \quad (10)$$

where $\mathbf{1}[\cdot]$ denotes the indicator function. The average of the precision@K across the N_t testing outfits is referred to as the average precision@K. The recommendation performance of a model is evaluated using the lift of average precision@K, which is defined as

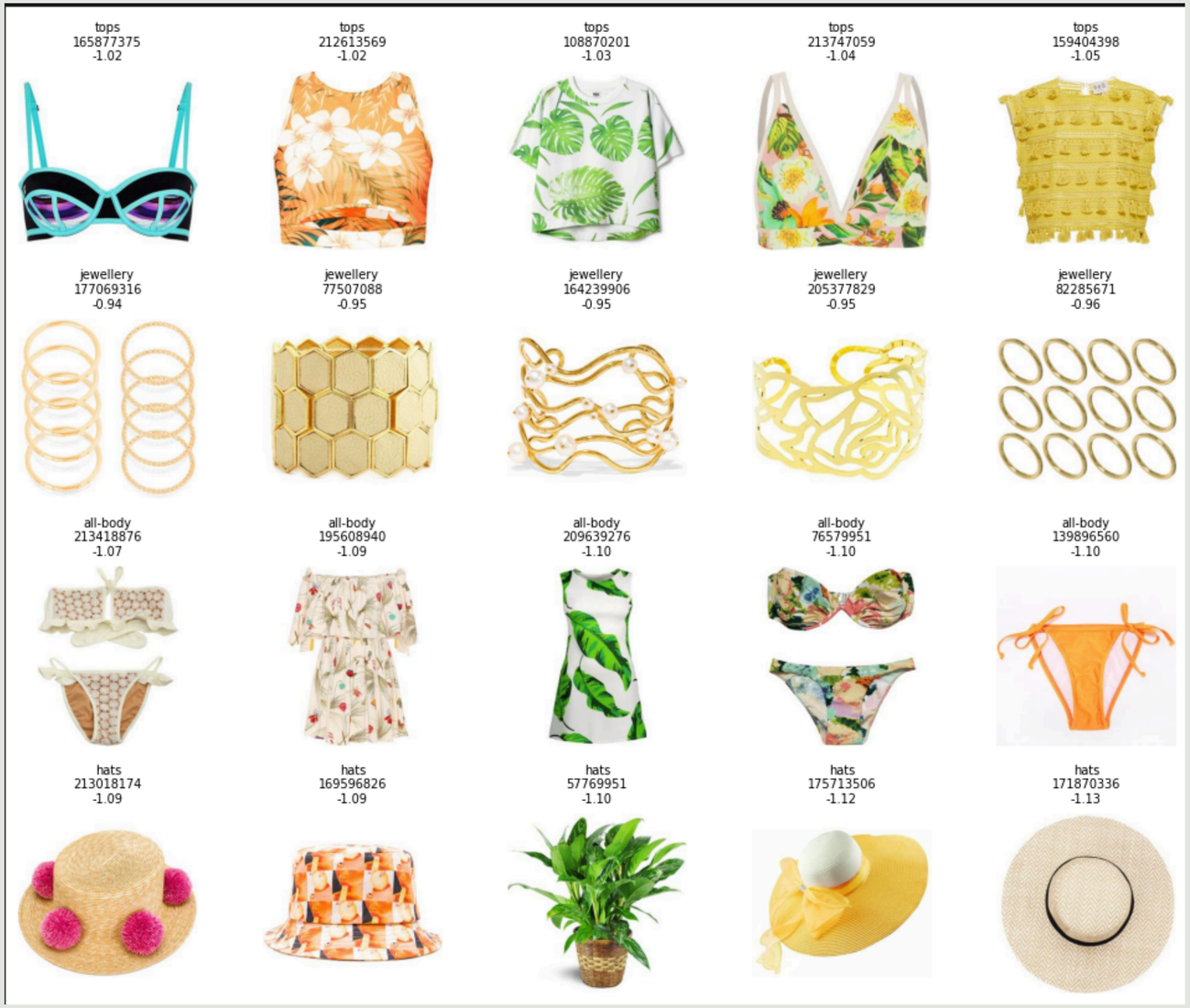
$$\text{Lift@K} = \frac{\text{average precision@}K\text{(model)}}{\text{average precision@}K\text{(random)}}, \quad (11)$$

where average precision@K (random) is that of a recommender that would select items at random for each of the apparel categories of interest.

Precision@K just represents how many outfits from the real outfit set of query get recommended by my model.

RESULTS- QUALITATIVE AND QUANTITATIVE

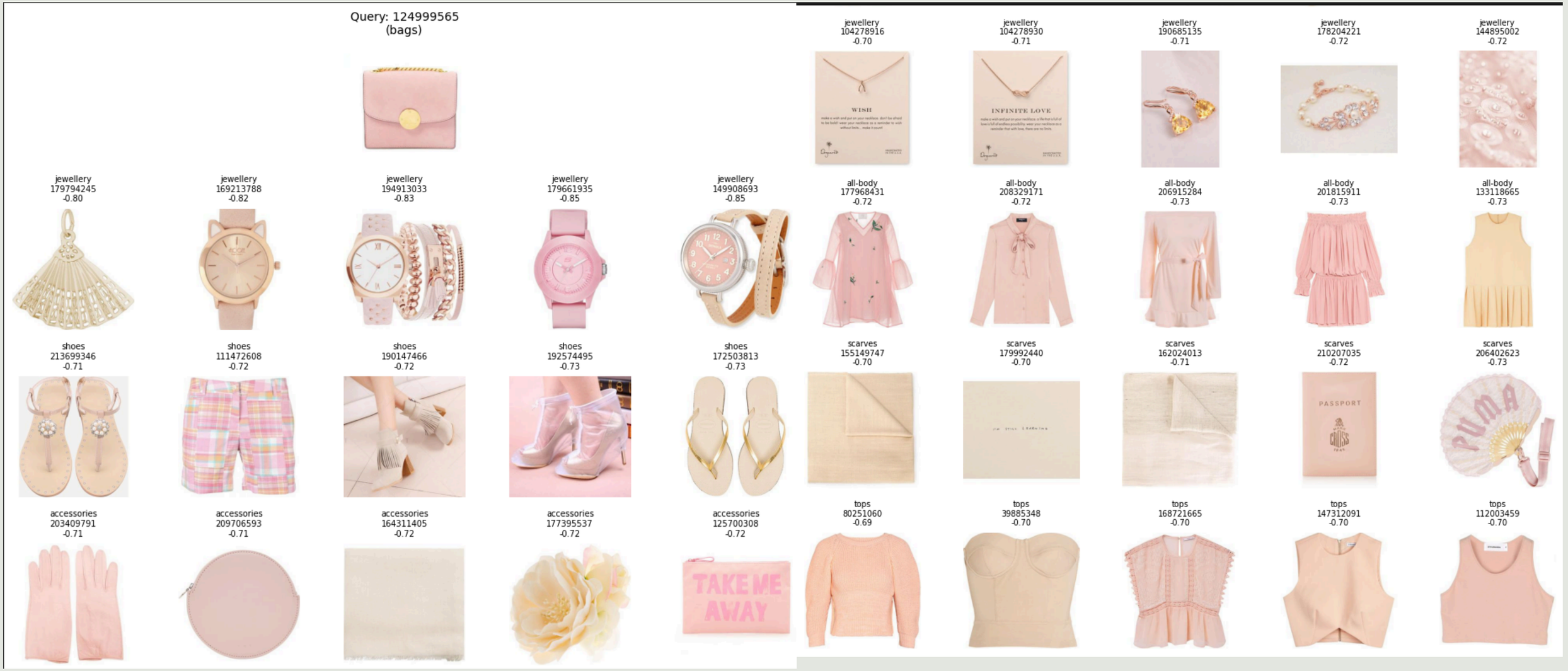
When model was trained for 1600 outfits from train.json and validated for 400 outfits from train.json. One sample result is shown:-


tops
165877375
-1.02
tops
212613569
-1.02
tops
108870201
-1.03
tops
213747059
-1.04
tops
159404398
-1.05
jewellery
177069316
-0.94
jewellery
77507088
-0.95
jewellery
164239906
-0.95
jewellery
205377829
-0.95
jewellery
82285671
-0.96
all-body
213418876
-1.07
all-body
195608940
-1.09
all-body
209639276
-1.10
all-body
76579951
-1.10
all-body
139896560
-1.10
hats
213018174
-1.09
hats
169596826
-1.09
hats
57769951
-1.10
hats
175713506
-1.12
hats
171870336
-1.13


accessories
213043315
-1.10
accessories
10986897
-1.12
accessories
206991249
-1.13
accessories
96983131
-1.15
accessories
51761450
-1.16

Precision@5 came out to be zero i.e. none of the recommended items belonged to the original outfit set of the query, in paper they used 10650, 1902, 1395 for training, validation and testing so I had used only 1600 outfits for training and 400 for validation and 5 for testing so I increased training to 4000 outfits validation to 1000 outfits.

When model was trained for 4000 outfits from train.json and validated for 1000 outfits from train.json. One result for same query is shown we can see the improvements in the recommended items :-



But Precision@K still was 0. So I increased K=5 to K=100 then also Precision@K= 0

Cause of low precision

This low precision@K might be due to the fact that I am not taking Lambda as defined in the paper, I used Λj=||Wj||^2 then in the loss function trace term becomes=1 constant meaning all input features are equally important and uncorrelated while acc to the closed form defination of lambda it is that some features are correlated- shrink the directions that are not relevant. Also the training set is also small I need to change update_lambda defination as well as train on more data.

Thank You

For your attention