Prepoznavanje lica koristeći tenzor-tenzor dekompoziciju

Tin Kranželić, Dora Parmać

Travanj 2021

Sadržaj

1 Uvod

Tenzor je višedimenzionalni niz. Tenzor prvog reda je vektor, tenzor drugog reda je matrica, a tenzor reda većeg od 3 je tenzor višeg reda. U mnogim slučajevima, bolje je pohraniti podatke u multidimenzionalnim nizovima nego u ekvivalentnoj matričnoj formi. Primjerice, promotrimo pohranjivanje slika različitih osoba, pod različitim osvjetljenjima, različitih ekspresija i sl. Korištenje tenzora višeg reda nam omogućuje pohranu ovakvih slika u tenzor reda 5, s modovima ljudi, osvijetljenja, ekspresija, pixela i gledišta. U HITS analizi, korištenje tenzora 3. reda omogućuje uključenje informacije o termu zajedno s tradicionalnom matricom susjedstva koja se sastoji od hubova i autoriteta. Općenito, tenzori višeg reda se mogu koristiti u raznim primjenama, poput emometrije, psihometrije, obradi slike i signala...

Jednom kada smo iskoristili tenzore višeg reda za pohranu multidimenzionalnih podataka, aplikacija će diktirati kakvu vrstu manipulacije podatcima moramo napraviti. Važan korak je kompresija podataka, čime se želi osigurati da takvi podatci zadrže određene značajke. Do danas, mnogo je različitih tipova dekompozicije tenzora predstavljeno.

Ne postoji jedna metoda dekompozicije tenzora koja udovoljava svim aplikacijama. Usporedno s matričnim slučajem, gdje se kompresija uglavnom obavlja putem SVD dekompozicije, za tenzore reda većeg od 2, čak je koncept ranga čudna stvar. Najčešće korištene tenzor dekompozicije su CANDECOMP/PARAFAC (CP) dekompozicija i Tucker dekompozicija.

Neka je $\boldsymbol{\mathcal{A}}$ tenzor reda $I \times J \times K$. Tada se njegova CP dekompozicija može zapisati kao suma vanjskog produkta vektora:

$$\boldsymbol{\mathcal{A}} = \sum_{l=1}^{R} u^{(l)} \circ v^{(l)} \circ w^{(l)} \qquad \Longrightarrow \qquad (\boldsymbol{\mathcal{A}})_{i,j,k} = \sum_{l=1}^{R} u_i^{(l)} v_j^{(l)} w_k^{(l)} \tag{1}$$

gdje su

$$u^{(l)} \in \mathbb{R}^I, v^{(l)} \in \mathbb{R}^J, w^{(l)} \in \mathbb{R}^K, zak = 1, 2, \dots R.$$

Tuckerova dekompozicija je:

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U} \times_2 \boldsymbol{V} \times_3 \boldsymbol{W} = \sum_{i=1}^{R_1} \sum_{j=1}^{R_2} \sum_{k=1}^{R_3} = \sigma_{ijk}(u^{(i)} \circ v^{(j)} \circ w^{(k)}) \qquad (2)$$

gdje je: $R_1 \leq I, R_2 \leq J, R_3 \leq K$ i za $i, j, k, i^{(i)} \in \mathbb{R}^I, v^{(l)} \in \mathbb{R}^J, w^{(l)} \in \mathbb{R}^K, \boldsymbol{\mathcal{G}} = \sigma_{ijk} \in \mathbb{R}^{R_1 \times R_2 \times R_3}, \boldsymbol{U} = [u^{(1)}, u^{(2)} \dots u^{(R_1)}], \boldsymbol{V} = [v^{(1)}, v^{(2)} \dots v^{(R_2)}]$,
$\boldsymbol{W} = [w^{(1)}, w^{(2)} \dots w^{(R_3)}]$.

Obje ove dekompozicije se mogu koristiti kao generalizacija matrične singularne dekompozicije višeg reda i analize glavnih komponenata. Međutim, za razliku od matričnog SVD-a čiji je rang broj ne-nul pojedinačnih vrijednosti, minimalna vrijednost R (tenzorski rang A) u CP-u ne može se uvijek izračunati. Za Tuckerovu dekompoziciju čak moramo definirati novi rang npr. n-rang da bi dalje mogli raditi.

U aplikacijama se često fiksira R i pronađe optimalna Frobeniusova norma između $\boldsymbol{\mathcal{A}}$ i zbroja vanjskih produkta ranga 1, sa ili bez dodatnih ograničenja na vektorske komponente. Iako ovaj pristup može biti dovoljan za mnoge primjene, u ovom radu predstavljamo potpuno drugačiji pristup te dobivamo faktorizaciju trećeg reda tenzora koje primjenjujemo na problem prepoznavanja lica. Sadašnji rad nije prvi koji razmatra upotrebu tenzora i faktorizacija tenzora u kontekstu prepoznavanja lica. Prvo djelo ove prirode za koje smo svjesni dogodilo se početkom 2000-ih i iznjedrilo je algoritam nazvan TensorFaces. TensorFaces izgrađen je oko SVD-a višeg reda i bio je prvi višedimenzionalni analog PCA za prepoznavanje lica. Ostali radovi na ovom području uključuju tensorsku analizu podsprostora, koja je slična pristupu koji ovdje zagovaramo po tome što se same slike smatraju dvodimenzionalnim objektima, a ne vektoriziranim (nalaze se u

tradicionalnom PCA temeljenom na matrici i u TensorFaces), ali se razlikuje da koristi teorijski pristup grafu (na temelju Laplacea).

2 Tradicionalni algoritmi

2.1 PCA algoritam

PCA ili Principal Component Analysis, izmislio je Karl Pearson 1901. godine, i od tada je klasična tehnika koja se koristi u prepoznavanju slika i kompresiji. Jednostavno rečeno, PCA se koristi kako bi se transformirao veliki broj moguće koreliranih varijabli u manji broj nekoreliranih varijabli poznatije kao glavne komponente. Prvih nekoliko glavnih komponenata objašnjava većinu varijacija originalnih podataka, dok preostali predstavljaju zanemariv doprinos. Stoga podatka možemo ekonomičnije opisati pomoću prvih nekoliko glavnih komponenata.

Neka je $\boldsymbol{X_1} \dots X_m$ kolekcija $l \times n$ slika. Definiramo matricu $\boldsymbol{A}$ tako da je j-ti stupac vec($\boldsymbol{X}_j$). Tada j-ti stupac predstavlja uzorak multivarijabilne random varijable. Srednja vrijednost se tada računa kao $\boldsymbol{M} = \frac{1}{m} \sum_{j=1}^{m} \boldsymbol{A}(:, j)$. Ako tada ažuriramo svaki stupac matrice $\boldsymbol{A}$ prema $\boldsymbol{A}(:, j) \leftarrow \boldsymbol{A}(:, j) - \boldsymbol{M}$, tada kažemo da je $\boldsymbol{A}$ u devijacijskoj formi srednje vrijednosti.

Kovarijanca matrice $\boldsymbol{A}$ je dana s: $\boldsymbol{C} = \frac{1}{m-1} \boldsymbol{A}\boldsymbol{A}^T$. Očito je $\boldsymbol{C}$ simetrična, pozitivna, semidefinitna, ortogonalna, dijagonalna matrica, odnosno vrijedi: $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$, gdje je $\boldsymbol{D} = diag(d_{11}, d_{22}, \dots)$ dijagonalna matrica koja sadrži svojstvene vrijednosti od $\boldsymbol{C}$, pod pretpostavkom da je $d_{11} \geq d_{22} \geq \dots \geq 0$, a $\boldsymbol{U}$ je ortogonalna matrica. Stupci $\boldsymbol{u_i}$ matrice $\boldsymbol{U}$, svojstvene vrijednosti od $\boldsymbol{C}$ su zapravo glavne komponente, te $\boldsymbol{u_1}$ opisuje smjer u kojem podatci imaju maksimalnu varijancu.

Zbog numeričkih razloga, ne računa se $\boldsymbol{C}$ i njegova svojstvena dekompozicija. Umjesto toga, uz danu SVD dekompoziciju matrice $\boldsymbol{A}$, $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$ iz $\boldsymbol{C} = \frac{1}{m-1}\boldsymbol{A}\boldsymbol{A}^T = \frac{1}{m-1}\boldsymbol{U}\boldsymbol{S}^2\boldsymbol{V}^T$ slijedi da su ne-nul singularne vrijednosti matrice $\boldsymbol{A}$, označene s $\boldsymbol{s_{ii}}$ povezane s ne-nul svojstvenim vrijednostima od $\boldsymbol{C}$, odnosno vrijedi $\boldsymbol{s_{ii}} = \sqrt{(m-1)d_{ii}}$, a lijeva singularna matrica $\boldsymbol{U}$ sadrži glavne komponente od $\boldsymbol{C}$.

Glavna ideja je $\boldsymbol{A}(:,j) \approx \boldsymbol{U}_r\boldsymbol{U}_r^T\boldsymbol{A}(:,j)$ gdje je $\boldsymbol{U}_r = \boldsymbol{U}(:,1:r)$ za neki $r \ll m$, jer su singularne vrijednosti nakon r-te jako male usporedno s m što ukazuje na to da je velika većina varijance podataka dobro zabilježena u smjerovima koji odgovaraju najvećim r-singularnim vrijednostima. Matrica $\boldsymbol{U}_r\boldsymbol{U}_r^T$ je ortogonalni projektor na $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r$, prostor razapet s prvih r glavnih komponenata. Također, primijetimo:

$$\boldsymbol{A}(:,j) \approx \boldsymbol{U}_r\boldsymbol{U}_r^T\boldsymbol{A}(:,j) = \sum_{i=1}^{r} c_i\boldsymbol{u}_i \qquad c_i = \boldsymbol{u}_i^T\boldsymbol{A}(:,j) \tag{3}$$

što zapravo opisuje da je j-ta slika dobro aproksimirana kao linearna kombinacija ortogonalnih stupaca matrice $\boldsymbol{U}_r$. Ako $\boldsymbol{X}$ predstavlja novu sliku, tada možemo izračunati proširenje koeficijenata u ovom prostoru $\boldsymbol{U}_r^T vec(\boldsymbol{X})$ i odrediti sliku koja pripada osobi čiji je koeficijent proširenja najbliži koeficijentu slike za testiranje.

Algorithm 1: Traditional Matrix PCA Method

Input: Training: $\mathbf{I}_i$, $i = 1, 2, ..., N$, Testing: $\mathbf{J}$, Truncation index k
Output: Comparison of Test Image with Compressed Training Set

for $i = 1$ **to** N **do**
  $\mathbf{L}(:, i) \leftarrow$ vectorized $\mathbf{I}_i$
end
$\mathbf{M} \leftarrow$ mean image
$\mathbf{A} \leftarrow$ mean-deviation form of $\mathbf{L}$
$\mathbf{U} \leftarrow$ left singular vectors of $\mathbf{A}$
$\mathbf{G} \leftarrow \mathbf{U}(:, 1{:}k)^T\mathbf{A}$
$\mathbf{T} \leftarrow \mathbf{J} - \mathbf{M}$
$\mathbf{t} \leftarrow$ vectorized form of $\mathbf{T}$
$\mathbf{c} \leftarrow \mathbf{U}(:, 1{:}k)^T\mathbf{t}$
for $j = 1$ **to** N **do**
  Calculate $||c - \mathbf{G}(:, j)||_F$
end
Claim the training image whose coefficient is closest to that of the test image is the match.

Slika 1: PCA algoritam

2.2 TensorFaces

TensorFaces bio je prvi algoritam koji se koristio manipulacijom tenzora za prepoznavanje lica. U ovom algoritmu, podatci su predstavljeni tenzorom s različitim modovima za različite faktore. Jedan način na koji je algoritam konzistentan s tradicionalnim PCA je taj što su slike vektorizirane. Multidimenzionalnost se stoga dobiva samo korištenjem ostalih načina za predstavljanje ostalih značajki podataka. Aproksimacija tenzora obavlja se multidimenzionalnim SVD višeg reda - HOSVD-om, koji je Tuckerova reprezentacija.

Koristimo tenzor $\boldsymbol{\mathcal{L}}$ da bi specificirali ljude, točke gledišta, osvijetljenje, ekspresije i piksele, redom, a P, V, I, E su brojevi ljudi, točaka gledišta, osvijetljenja, ekspresije i piksela u podatcima za treniranje, dok su P', V', I', E' isti ti koeficijenti za podatke za testiranje. Pix je duljina svake vektorizirane slike. Tada možemo izračunati jezgreni tenzor:

$$\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{L}} \times_1 \boldsymbol{U}_p^T \times_2 \boldsymbol{U}_v^T \times_3 \boldsymbol{U}_i^T \times_4 \boldsymbol{U}_e^T \times_5 \boldsymbol{U}_{pix}^T \tag{4}$$

gdje su $\boldsymbol{U}_p, \boldsymbol{U}_v, \boldsymbol{U}_i, \boldsymbol{U}_e, \boldsymbol{U}_{pix}$ lijeve matrice SVD-a matrica $\boldsymbol{L}_{(1)}, \boldsymbol{L}_{(2)}, \boldsymbol{L}_{(3)}, \boldsymbol{L}_{(4)}, \boldsymbol{L}_{(5)}$ tenzora $\boldsymbol{\mathcal{L}}$ duž svakog moda. Jezgreni tenzor $\boldsymbol{\mathcal{Z}}$ upravlja interakcijom između različitih faktora: $\boldsymbol{U}_p \in \mathbb{R}^{P \times P}$ obuhvaća prostor parametara ljudi, $\boldsymbol{U}_v \in \mathbb{R}^{V \times V}$ obuhvaća prostor parametara točaka gledišta, $\boldsymbol{U}_i \in \mathbb{R}^{I \times I}$ obuhvaća prostor parametara osvijetljenja, $\boldsymbol{U}_e \in \mathbb{R}^{E \times E}$ obuhvaća prostor parametara ekspresije, a, $\boldsymbol{U}_{pix} \in \mathbb{R}^{Pix \times (P \times V \times I \times E)}$ obuhvaća prostor slika. $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{Z}} \times_2 \boldsymbol{U}_v \times_3 \boldsymbol{U}_i \times_4 \boldsymbol{U}_e \times_5 \boldsymbol{U}_{pix}$ definira $V \times I \times E$ različite baze za svaku kombinaciju točaka gledišta, osvijetljenja i ekspresija, a sub tenzor $\boldsymbol{\mathcal{B}}_{v,i,e}$ veličine $P \times 1 \times 1 \times 1 \times Pix$ za bilo koji dani v, i, e. Za svaku danu sliku, koristi se spljoštena matrica $\boldsymbol{B}^{\dagger}_{v,i,e}$ tenzora $\boldsymbol{\mathcal{B}}_{v,i,e}$ za projiciranje u skup vektora koeficijenata $c_{v,i,e}$ za svaku v, i, e kombinaciju. Zatim onaj koji ima najmanji $c_{v,i,e} - \boldsymbol{U}_p(p,:)^T$ identificira nepoznatu sliku kao osobu p jer su vektor retci matrice $\boldsymbol{U}_p$ koeficijenti za svaku osobu p.

Za razliku od optimalnog smanjenja dimenzionalnosti u matričnom PCA koje se može dobiti skraćivanjem dekompozicije singularne vrijednosti, ne postoji trivijalni multilinearni analogon za smanjenje dimenzionalnosti Tuckerove faktorizacije. Jedan od prirodnih izbora za smanjenje dimenzionalnosti je skratiti matrice modova koje proizlaze iz SVD algoritma u modu N; recimo $\boldsymbol{U}_p$ je veličine $P \times P_1$, $\boldsymbol{U}_v$ je veličine $V \times V_1$, $\boldsymbol{U}_i$ je veličine $I \times I_1$, $\boldsymbol{U}_e$ je veličine $E \times E_1$, $\boldsymbol{U}_{pix}$ je veličine $Pix \times Pix_1$ gdje je $P_1 \leq P, V_1 \leq V, I_1 \leq I, E_1 \leq E, Pix_1 \leq (P \times V \times I \times E)$, te će tada dekompozicija reducirati na $P_1 \times V_1 \times I_1 \times E_1 \times Pix_1$ ali odgovarajuća aproksimacija uglavnom nije optimalna u smislu da bi odgovarajuća Tuckerova faktorizacija bila najbolja aproksimacija originalu u Frobeniusovoj normi. Osim toga, što se tiče prepoznavanja, još trebamo formirati bazu $\boldsymbol{\mathcal{Z}}$ koja je veličine $P_1 \times V \times I \times E \times Pix_1$.

Alternativno, možemo izračunati optimalni rang $(R_1, R_2, R_3, R_4, R_5)$ ortogonalne Tuckerove dekompozicije, ali optimizacija nije trivijalna i zahtijeva iterativni algoritam. Također, prikladni izbori za $R_i, i = 1, ..., 5$ se ne znaju a priori.

Algorithm 2: TensorFaces Method

Input: Training images $\mathbf{I}_{p,v,i,e}$, $p = 1, ..., P$, $v = 1, ..., V$, $i = 1, ..., I$, $e = 1, ..., E$;
Testing Image $\mathbf{J}$
Output: Coefficients of testing images with some tensor basis and comparison

for $p = 1, \ldots, P, v = 1, \ldots, V, i = 1, \ldots, I, e = 1, \ldots, E$ **do**
  $\mathcal{L}(p, v, i, e, :) \leftarrow$ vectorized $\mathbf{I}_{p,v,i,e}$
end
Do the HOSVD to get the core tensor $\mathcal{Z}$ and orthogonal factor matrices $\mathbf{U}_i$:
$\mathcal{L} \leftarrow \mathcal{Z} \times_1 \mathbf{U}_p \times_2 \mathbf{U}_v \times_3 \mathbf{U}_i \times_4 \mathbf{U}_e \times_5 \mathbf{U}_{pix}$
$\mathcal{B} \leftarrow \mathcal{Z} \times_2 \mathbf{U}_v \times_3 \mathbf{U}_i \times_4 \mathbf{U}_e \times_5 \mathbf{U}_{pix}$
$\mathcal{T}(1, 1, 1, 1, :) \leftarrow$ vectorized testing image
for $v = 1, \ldots, V, i = 1, \ldots, I, e = 1, \ldots, E$; **do**
  $c_{v,i,e} \leftarrow (\mathcal{B}^{\dagger}_{v,i,e})^T \mathcal{T}(1, 1, 1, 1, :)$
  for $p = 1, \ldots, P$ **do**
    Calculate $||c_{v,i,e} - \mathbf{U}_p(p, :)^T||_F$
  end
end
Claim the training image whose coefficient is closest to that of the test image is the match.

Slika 2: TensorFaces algoritam

3 Novi pristup - Tensor Framework

3.1 Tenzor-tenzor produkt

Definicija 3.1. *Neka je $\mathcal{A}$ tenzor veličine $l \times p \times n$ i $\mathcal{B}$ tenzor veličine $p \times m \times n$. Tada je tenzor-produkt $\mathcal{A} * \mathcal{B}$ $l \times m \times n$ tenzor:*

$$\mathcal{A} * \mathcal{B} = fold(bcirc(\mathcal{A}) \cdot unfold(\mathcal{B})) \tag{5}$$

Budući cirkularne matrice možemo dijagonalizirati normaliziranom diskretnom Fourierovom transformacijom (DFT), možemo blok-dijagonalizirati cirkularne matrice. Neka je $\mathcal{A} \in \mathbb{R}^{l \times m \times n}$ u $\boldsymbol{F}_n$ $n \times n$ DFT matrica. Sada je birc($\mathcal{A}$) blok matrica $n \times n$ koja ima $l \times m$ blokova. Neka $\boldsymbol{P}_1 i \boldsymbol{P}_2$ označavaju permutacije po

koracima tako da je:

$$\boldsymbol{P}_1 \cdot birc(\mathcal{A}) \cdot \boldsymbol{P}_2 = \begin{bmatrix} N_{11} & N_{12} & \dots N_{1m} \\ N_{21} & N_{22} & \dots N_{2m} \\ \vdots & \vdots & \ddots \\ N_{l1} & N_{l2} & \dots N_{lm} \end{bmatrix} \tag{6}$$

$l \times m$ blok matrica sa $n \times n$ cirkularnih blokova N_{ij} gdje je prvi stupac sastavljen od svih i, j ulaza matrice $\boldsymbol{A}^k, k = 1, 2, \dots n$ i vrijedi:

$$\boldsymbol{P}_2^T \cdot (\boldsymbol{F}_{\boldsymbol{n}}^* \otimes \boldsymbol{I}_{\boldsymbol{n}}) \cdot \boldsymbol{P}_{\boldsymbol{2}} = \boldsymbol{I}_{\boldsymbol{m}} \otimes \boldsymbol{F}_{\boldsymbol{n}}^* \tag{7}$$

Primijetimo da je:

$$\boldsymbol{P}_{\boldsymbol{1}} \cdot (\boldsymbol{F}_{\boldsymbol{n}} \otimes \boldsymbol{I}_{\boldsymbol{l}}) \cdots {\boldsymbol{P}_{\boldsymbol{1}}}^T \cdot \boldsymbol{P}_{\boldsymbol{1}} \cdot bcirc(\mathcal{A}) \cdot \boldsymbol{P}_{\boldsymbol{2}} \cdot {\boldsymbol{P}_{\boldsymbol{2}}}^T \cdot (\boldsymbol{F}_{\boldsymbol{n}}^* \otimes \boldsymbol{I}_{\boldsymbol{m}}) \cdot \boldsymbol{P}_{\boldsymbol{2}} \tag{8}$$

jednako

$$(\boldsymbol{I}_{\boldsymbol{l}} \otimes \boldsymbol{F}_{\boldsymbol{n}}) \begin{bmatrix} N_{11} & N_{12} & \dots N_{1m} \\ N_{21} & N_{22} & \dots N_{2m} \\ \vdots & \vdots & \ddots \\ N_{l1} & N_{l2} & \dots N_{lm} \end{bmatrix} (\boldsymbol{I}_{\boldsymbol{m}} \otimes {\boldsymbol{F}_{\boldsymbol{n}}}^*) = \begin{bmatrix} D_{11} & D_{12} & \dots D_{1m} \\ D_{21} & D_{22} & \dots D_{2m} \\ \vdots & \vdots & \ddots \\ D_{l1} & D_{l2} & \dots D_{lm} \end{bmatrix} \tag{9}$$

gdje je svaka $\boldsymbol{D}_{ij}$ $n \times n$ dijagonalna matrica. Pomnožimo li lijevo stranu s $\boldsymbol{P}_1^T$ i desnu s $\boldsymbol{P}_1^T$ dobijemo:

$$(\boldsymbol{F}_{\boldsymbol{n}}^* \otimes \boldsymbol{I}_{\boldsymbol{l}}) \cdot birc(\mathcal{A}) \cdot (\boldsymbol{F}_{\boldsymbol{n}}^* \otimes \boldsymbol{I}_{\boldsymbol{m}}) = \boldsymbol{P}_1^T \begin{bmatrix} D_{11} & D_{12} & \dots D_{1m} \\ D_{21} & D_{22} & \dots D_{2m} \\ \vdots & \vdots & \ddots \\ D_{l1} & D_{l2} & \dots D_{lm} \end{bmatrix} \boldsymbol{P}_2^T = \begin{bmatrix} A^{(1)} & & \\ & \ddots & \\ & & A^{(n)} \end{bmatrix} \tag{10}$$

Primjena brze Fourierove transformacije duž trećeg moda $\mathcal{A}$ daje nam alternativno sredstvo za računanje $\hat{A}(i)$. Neka je $\hat{\mathcal{A}} = fft(\mathcal{A}, [], 3)$, te za svaki i $\hat{\boldsymbol{A}} = \hat{\mathcal{A}}(:, :, i)$. Kada su tenzori gusti, mogli bismo izračunati $\mathcal{A} * \mathcal{B}$ računanjem FFT-a duž svake niti $\mathcal{A}i\mathcal{B}$ kako bi dobili $\hat{\mathcal{A}}i\hat{\mathcal{B}}$, pomnožiti svaki par lica $\hat{\mathcal{A}}i\hat{\mathcal{B}}$ kako bi dobili $\hat{\mathcal{C}}$, te uzeti inverzni FFT duž niti kako bi dobili željeni rezultat. Kako s većinom aplikacija imamo $log(n) \leq m, l$ vremena, s ovim načinom dobijemo: $O(lpmn + (lp + pm)nlog(n)) \approx O(lpmn)$.

Algorithm 3: Tensor-Tensor Product, using Fourier Domain Computation

Input: $\mathcal{A} \in \mathbb{R}^{\ell \times p \times n}, \mathcal{B} \in \mathbb{R}^{p \times m \times n}$
Output: $\mathcal{C} := \mathcal{A} * \mathcal{B}$
$\hat{\mathcal{A}} \leftarrow \text{fft}(\mathcal{A}, [], 3)$; $\hat{\mathcal{B}} \leftarrow \text{fft}(\mathcal{B}, [], 3)$
for $i = 1$ **to** n **do**
  $\hat{\mathcal{C}}(:, :, i) \leftarrow \hat{\mathcal{A}}(:, :, i)\hat{\mathcal{B}}(:, :, i)$
end
$\mathcal{C} \leftarrow \text{ifft}(\hat{\mathcal{C}}, [], 3)$

Slika 3: Tenzor-tenzor produkt, koristeći FFT

Lemma 3.1. $\mathcal{A} * (\mathcal{B} * \mathcal{C}) = (\mathcal{A} * \mathcal{B}) * \mathcal{C}$

Definicija 3.2. *Ako je $\mathcal{A}$ $l \times m \times n$, tada je $\mathcal{A}^T$ $m \times l \times n$ tenzor nastao transponiranjem svakom frontalnog slicea, te mijenjanjem poretka transponiranih frontalnih sliceova od 2. do n-tog.*

Definicija 3.3. *Tenzor je f-dijagonalan ako je svaki frontalni slice dijagonalna matrica.*

Definicija 3.4. *$l \times l \times n$ tenzor $\mathcal{I}_{lln}$ je tenzor identitete čiji je prvi frontalni slice $l \times l$ matrica identiteta, te su ostali frontalni sliceovi 0. Slično, tenzor je f-gornjetrokutast ako je svaki frontalni slice gornjetrokutast.*

Definicija 3.5. *$l \times l \times n$ $\mathcal{Q}$ realni tenzor je ortogonalan ako vrijedi:*

$$\mathcal{Q}^T * \mathcal{Q} = \mathcal{Q} * \mathcal{Q}^T = \mathcal{I} \tag{11}$$

Primijetimo da to znači da je skup svih lateralni sliceova (koji su također ma-

trice) ortogonalni skup, u smislu da vrijedi:

$$\mathcal{Q}(:,I,:)^T * \mathcal{Q}(:,I,:) = \begin{cases} \boldsymbol{e_1} & \text{ako } i = j \\ 0 & \text{ako } i \neq j \end{cases}$$

gdje je svaki $\boldsymbol{e}_i$ tenzor čiji je prvi frontalni slice 1, a svi ostali frontalni sliceovi su 0.

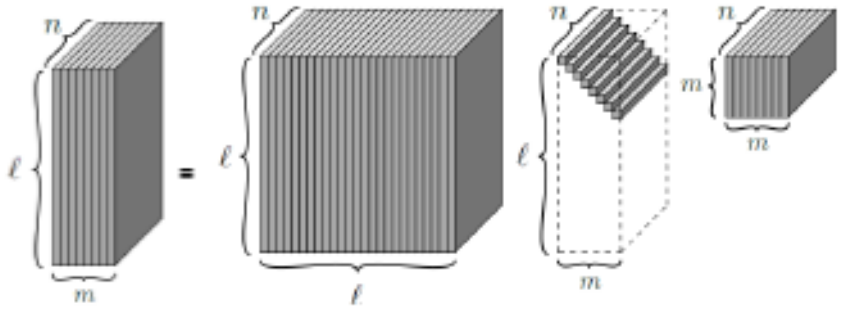
3.2 T-SVD

T-SVD je tenzor dekompozicija bazirana na već spomenutom tenzor-tenzor produktu, što je zapravo analogon matričnom SVD-u.

Teorem 3.2. *Neka je $\mathcal{A}$ realni $l \times m \times n$ tenzor. Tada se $\mathcal{A}$ može faktorizirati kao:*

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T \tag{12}$$

gdje je $\mathcal{U}$ ortogonalni $l \times l \times n$ tenzor, $\mathcal{V}$ ortogonalni $m \times m \times n$ tenzor, a $\mathcal{S}$ f-dijagonalni $l \times m \times n$ tenzor.



Slika 4: T-SVD

Jako lijepa karakteristika T-SVDa jest što daje način za pronalazak optimalne aproksimacije tenzora kao sumu $k < min(n_1, n_2)$ vanjskih produkta matrice analogno kako se matrični SVD može koristiti za definiranje optimalne aproksimacije matrice kao sume vanjskog produkta vektora:

Teorem 3.3. *Neka je $T - SVD$ tenzora $\mathcal{A} \in \mathbb{R}^{l \times m \times n}$ dan s $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$ i za $k < min(l, m)$ definiramo:*

$$\mathcal{A}_k = \sum_{i=1}^{k} \mathcal{U}(:, i, :) * \mathcal{S}(i, i, :) * \mathcal{V}(:, i, :)^T \tag{13}$$

Tada je $\mathcal{A}_k = arg \min_{\bar{\mathcal{A}} \in M} ||\mathcal{A} - \bar{\mathcal{A}}||_F$ gdje je $M = \{\mathcal{C} = \mathcal{X} * \mathcal{Y} | \mathcal{X} \in \mathbb{R}^{l \times k \times n}, \mathcal{Y} \in \mathbb{R}^{k \times m \times n}\}$. Primijetimo da, kada je 3.dimenzija tenzora jednaka 1, tada se ovaj teorem svodi na Eckart-Young teorem za matrice, budući t-produkt postane matrični produkt.

3.3 Novi pristup baziran na T-SVDu

Novi pristup se razlikuje od matričnog PCA u smislu da se ovdje podatci drže u tenzoru reda 3 i koristi se tenzor-tenzor faktorizacija na osnovu $*$ operacije. Prvi pristup je baziran specifično na spomennutom T-SVDu. Također, razlika s TensorFaces algoritmom je u tome što sada a) slike nisu pikselizirane, b) nema dodatnih dimenzija za osvijetljenje, poze i sl. c) bazirano je na T-SVD umjesto na HOSVDu. To znači da ovom metodom možemo proizvesti koeficijente direktno zahvaljujući ortogonalnosti baznih elemenata (tj. rekonstrukcija komprimiranih informacija odgovara ortogonalnoj projekciji na $*$ operaciji). To znači da suprotno TensorFacesu, ne moramo rješavati problem najmanjih kvadrata.

Ovdje ne vektoriziramo svaki uzorak i sve dekompozicije, i projekcije se izvode u tenzorskom prostoru. Budući ćemo koristiti matrice orijentirane 3. dimenziji, uvodimo novu operaciju: "squeeze" operacija tenzora $\mathcal{X} \in \mathbb{R}^{l \times 1 \times n}$ proizvodi

matricu:

$$\boldsymbol{X} := squeeze(\mathcal{X}) \longrightarrow \boldsymbol{X}(i,j) := \mathcal{X}(i,1,j) \tag{14}$$

dok je twist operacije inverz squeeze operacije:

$$twist(squeeze(\mathcal{X})) = \mathcal{X} \tag{15}$$

Neka je $\boldsymbol{M}$ srednja vrijednost slike, $\mathcal{A}(:,j,:) = twist(\boldsymbol{X}_j - \boldsymbol{M}), j = 1,\ldots,m$. Tada su uzorci reprezentirani vektor stupcima, duljine l, a ulazi su "tube" niti.

Analogno kovarijacijskoj matrici u tradicionalnom PCA, $\mathcal{K} := \frac{1}{m-1}\mathcal{A} * \mathcal{A}^T$ je simetrična, pozitivna, semidefinitna matrica, ortogonalno dijagonalna, tj. vrijedi $\mathcal{K} = \mathcal{U} * \mathcal{D} * \mathcal{U}^T$ gdje je $\mathcal{D}$ f-dijagonalni tenzor koji sadrži svojstvene vrijednosti od $\mathcal{K}$ na dijagonali. Stupce $\mathcal{U}(:,j,:)$ možemo zamisliti kao svojstvene matrice od $\mathcal{K}$. Kovarijacijski tenzor nam dopušta dobivanje maksimalne varijance zato što svako lice $\mathcal{K}(:,:,j)$ mjeri kovarijancu svih $j-ti$ stupaca slike.

Zbog razloga sličnih kao kod PCA matrične metode, ne računamo svojstveno dekompoziciju od $\mathcal{K}$, već koristimo tenzor verziju SVDa: $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$. Iz $\mathcal{K} := \frac{1}{m-1}\mathcal{A} * \mathcal{A}^T = \frac{1}{m-1}\mathcal{U} * \mathcal{S} * \mathcal{S}^T * \mathcal{U}^T$ da su ne-nul svojstvene vrijednosti od $\mathcal{K}$ jvadrati pripadnih singularnih vrijednosti od $\mathcal{A}$, a lijeva ortogonalna matrica $\mathcal{U}$ sadrži glavne komponente $\mathcal{K}$.

Budući je $\mathcal{U}$ ortogonalna, $\mathcal{U}(:,1:k,:)$ je ortogonalni skup lateralnih sliceova $\mathcal{U}(:,j,:) \in \mathbb{R}^{l\times 1\times n}$ i $\mathcal{U}(:,1:k,:) * \mathcal{U}(:,1:k,:)^T$ definira ortogonalni projektor koji projicira centriranu sliku na prostor manje dimenzije. Ukoliko koristimo prvih nekoliko lijevih singularnih lateralnih sliceova od $\mathcal{U}$ kao novu bazu, tada:

$$\mathcal{A}(:,j,:) \approx \mathcal{U}(:,1:k,:) * \mathcal{U}(:,1:k,:)^T * \mathcal{A}(:,j,:) = \sum_{t=1}^{k} \mathcal{U}(:,t,:) * \mathcal{C}(t,j,:) \tag{16}$$

gdje je $\mathcal{C}(:,j,:) = \mathcal{U}(:,1:k,:)^T * \mathcal{A}(:,j,:)$ i svaka nit $\mathcal{C}(t,j,:) = \mathcal{U}(:,t,:)^T * \mathcal{A}(:,j,:), t = 1,2,\ldots,k$. Tada se svaka centrirana slika $\mathcal{A}(:,j,:)$ može zamisliti kao

t-linearna kombinacija ortogonalnih elemenata baze $\mathcal{U}(:, j+t, :), t = 1, 2, \ldots, k$ s koeficijentima $\mathcal{C}(t, j, :)$. Reduciramo problem prepoznavanja na usporedbu koeficijenata test i trening slika. Primijetimo da ono što treba biti spremljeno su $k, l \times n$ elementi baze (koji se koriste za projekciju ulaznih slika) i $k\times$ (broj trening slika) $\times n$ tenzor $\mathcal{C}$ svih koeficijenata za trening slike.

Algorithm 4: T-SVD Method I

Input: Training: $\mathbf{I}_i$, $i = 1, 2, ..., N$; Test image $\mathbf{J}$, Truncation index k
Output: Match of Test image against Compressed Representation of Training Set

for $i = 1$ **to** N **do**
  $\mathcal{L}(:, i, :) \leftarrow \mathbf{I}_i$
end
$\mathcal{M} \leftarrow$ mean image
$\mathcal{A} \leftarrow$ mean-deviation form of $\mathcal{L}$
$\mathcal{U} \leftarrow$ left singular vectors of tensor $\mathcal{A}$
$\mathcal{C} \leftarrow \mathcal{U}(:, 1:k, :)^T * \mathcal{A}$
$\mathcal{T}(:, 1, :) \leftarrow \text{twist}(\mathbf{J} - \mathcal{M})$
$\mathcal{B} \leftarrow \mathcal{U}(:, 1:k, :)^T * \mathcal{T}$
for $j = 1$ **to** N **do**
  Calculate $||\mathcal{B} - \mathcal{C}(:, j, :)||_F$
end
Claim the training image whose coefficient is closest to that of the test image is the match.

Slika 5: T-SVD algoritam 1

Lemma 3.4. *Kada su slike jednodimenzionalni signali ($n = 1$), tenzorski SVD je tada zapravo PCA.*

U našoj analizi, koristno je zapisati 16 u matričnoj formi. Neka je $\boldsymbol{Y}_j := squeeze(\mathcal{A}(:, j, :))$.Primijetimo da je iz Algoritma 2, $\boldsymbol{Y}_j$ samo $j - ta$ trening slika (s oduzetom srednjom vrijednosti). Neka je $circ(v)$ cirkularna matrica čiji je prvi stupac v. Iz 16, definicije $*$ i malo manipulacije, vrijedi:

$$\boldsymbol{Y}_j \approx \sum_{t=1}^{k} \boldsymbol{W}_t circ(\boldsymbol{c_{t,j}}) \tag{17}$$

gdje je $\boldsymbol{W}_t := squeeze(\mathcal{U}(:, t, :))$ matrica $l \times n$ a $c_{t,j}^T := squeeze(\mathcal{C}(t, j, :)^T)$ vektor redak duljine n.

Teorem 3.5. *Neka je* $\boldsymbol{F}$ $n \times n$ *DFT matrica. S* $\hat{\boldsymbol{Y}}_j := \boldsymbol{Y}_j\boldsymbol{F}$, $\hat{\boldsymbol{W}}_j := \boldsymbol{W}_j\boldsymbol{F}$ *i* $\hat{\boldsymbol{c}_{t,j}}$ *Fourierovi koeficijenti vektor stupca* $\boldsymbol{c}_{t,j}$ *duljine* n*,*

$$\hat{\boldsymbol{Y}}_j \approx \left[\sum_{t=1}^{k} \hat{\boldsymbol{c}_{t,j}}^{(1)} \hat{\boldsymbol{U}}^{(1)}(:,t), \ldots, \sum_{t=1}^{k} \hat{\boldsymbol{c}_{t,j}}^{(n)} \hat{\boldsymbol{U}}^{(n)}(:,t) \right] \tag{18}$$

gdje $\hat{\boldsymbol{c}_{t,j}}^{(i)}$ *označava* $i-$*tu komponentu tog vektora, a* $\hat{\boldsymbol{W}}_t(:,i) = \hat{\boldsymbol{U}^{(i)}}(:,t)$ *gdje je* $\hat{\boldsymbol{U}}^{(i)}$ *unitarna matrica. Slijedi da je Algoritam 2 ekvivalentan somultanoj (i nezavisnoj) PCA metodi na svakom stupcu transformiranje slike, kada oduzmemo srednju vrijednost, u Fourierovom prostoru, uzimajući isto skraćenje indeksa* k *na svakom stupcu.*

Glavna posljedica ovog teorema je što ilustrira potencijal za daljnju kompresiju podataka u bazi podataka, pod uvjetom da smo spremni raditi izravno u Fourierovom prostoru. Prvo primijetimo da nema gubitka u postupku prepoznavanja ako radimo u Fourierovoj domena (pretpostavit ćemo da je n paran radi jednostavnosti):

1. Usporedba koeficijenata proširenja testne slike $\boldsymbol{J}$ s trening koeficijentom tenzora može se napraviti u Fourierovoj domeni. Primijetimo da se produkt $\mathcal{Z} := \mathcal{U}(:,1:k,:)^T * twist(\boldsymbol{J} - \boldsymbol{M}))$ može izračunati pomoću Algoritma 3, gdje akumuliramo $k \times 1 \times n$ objekt $\hat{\mathcal{Z}}$ u Fourierovoj domeni svakako.

2. Zbog konjugirane simetrije, samo se polovica produkata matrice mora izvršiti da bi izračunali $\hat{\mathcal{Z}}$. To znači da se samo prvih $\frac{n}{2}+1$ frontalnih sliceova od $\hat{\mathcal{U}}(:,1:k,:)$ moraju spremiti.

3. Ako želimo usporediti $\mathcal{Z}$ s lateralnim sliceovima od $\mathcal{C}$, možemo usporediti $\hat{\mathcal{Z}}(:,1,1:\frac{n}{2}+1)$ s $\hat{\mathcal{C}}(:,j,1:\frac{n}{2}+1)$ za $j = 1, \ldots,$ broj trening slika.

Spremanje prvih $\frac{n}{2}+1$ frontalnih sliceova zahtijeva istu količinu prostora za pohranu kao i spremanje $\mathcal{U}, \mathcal{C}$. To je zato što, osim DC pojma i $\frac{n}{2}+1$ pojmova

moramo pohraniti složene brojeve dvostruke preciznosti. Ali, budući da teorem kaže da radimo neovisni PCA na svakom od stupaca transformirane, slike u Fourierovom prostoru, odlučujemo da ima više smisla mijenjati k u stupcima. Konkretno, zamijenili bismo 16 sa:

$$\hat{\boldsymbol{Y}}_j \approx \left[\sum_{t=1}^{k_1} \hat{\boldsymbol{c}_{t,j}}^{(1)} \hat{\boldsymbol{U}}^{(1)}(:,t), \ldots, \sum_{t=1}^{k_n} \hat{\boldsymbol{c}_{t,j}}^{(n)} \hat{\boldsymbol{U}}^{(n)}(:,t) \right] \tag{19}$$

gdje smo sigurni da je zbog očuvanja simetrije, indeks skraćivanja za stupac j, $2 \leq j \leq \frac{n}{2}$ isti kao i za stupac $n - j + 2$.

Zbog konjugirane simetrije i potrebe za računanjem samo prve polovice koeficijenata za svaku sliku, moramo eksplicitno pohraniti posljednjih $n - \frac{n}{2} - 1$ koeficijenata. Dakle, zahtjeve za pohranom možemo opisati na slijedeći način:

1. Za danu $l \times n$ sliku u trening skupu, i $\frac{n}{2}+1$ indeksa skraćivanja $k_1, \ldots, k_{\frac{n}{2}+1}$, tako da je $k_1 \geq k_i$, moramo pohraniti:

   $$K = k_1 + 2k_2 + \ldots + 2k_{\frac{n}{2}} + 2k_{\frac{n}{2}+1} \tag{20}$$

   realnih (double precision) brojeva koji predstavljaju neskraćene koeficijente u Fourierovom prostoru.

2. Iz svake $\hat{\boldsymbol{U}}^i, i = 1, 2, \ldots, \frac{n}{2} + 1$ moramo spremiti samo k_i stupaca. Ovo zahtijeva pohranu lK double precision brojeva.

3. Za m = broj trening slika, moramo spremiti $lK + mK$ double precision brojeva ukupno.

Preostaje nam vidjeti kako identificirati parametre $k_i, i = 1, \ldots, \frac{n}{2} + 1$. Iz teorema 3.2 imamo da $\hat{\boldsymbol{U}}^i$ sadrži lijeve singularne vektore i-tog frontalnog slicea od $\hat{\mathcal{A}}$, a $\hat{\boldsymbol{S}}^i$ sadrži pripadne singularne vrijednosti $\hat{\boldsymbol{s}}^i_j, j = 1, 2, \ldots, minl, m$ i-tog frontalnog slicea. Nadalje, primijetimo da:

$$||\hat{\mathcal{A}}(:,:,1:\frac{n}{2}+1)||_F^2 = \sum_{i=1}^{\frac{n}{2}+1} ||\hat{\mathcal{A}}(:,:,i)||_F^2 = \sum_{i=1}^{\frac{n}{2}+1} \sum_{j=1}^{min(l,m)} (\hat{\boldsymbol{s}}_j^i)^2 \tag{21}$$

Dakle, da bi optimalno definirali indekse skraćivanja, promotrimo mjerenje relativne energije:

$$\frac{\sum_{i=1}^{\frac{n}{2}+1} \sum_{j=1}^{k_i} (\hat{\boldsymbol{s}}_j^i)^2}{||\hat{\mathcal{A}}(:,:,1:\frac{n}{2}+1)||_F^2} \tag{22}$$

Očito bi trebali zadržati najveće (uzimajući u obzir oba indeksa i i j) singularne vrijednosti da bi održali relativnu energiju blizu 1. Stoga, odlučimo koja je željena vrijednost relativno energije (npr. 0.9), poredamo $\hat{\boldsymbol{s}}_j^i$) u vektoru $\boldsymbol{q}$ od najvećeg do najmanjeg, pronađemo najmanji indeks t tako da je: $\sum_{i=1}^{\frac{n}{2}+1} \sum_{j=1}^{k_i} \{(\hat{\boldsymbol{s}}_j^i)^2 : (\hat{\boldsymbol{s}}_j^i)^2 > \boldsymbol{q}(t)^2\} > ||\hat{\mathcal{A}}(:,:,1:\frac{n}{2}+1)||_F^2 \times .9$. Prisjetimo se da za svaki fiksirani i, $\hat{\boldsymbol{s}}_1^i \geq \hat{\boldsymbol{s}}_2^i \geq \ldots \geq \hat{\boldsymbol{s}}_{min\{l,m\}}^i$. Dakle, k_i su definirani tako da je $\hat{\boldsymbol{s}}_{k_t}^i > \boldsymbol{q}(t)$ ali $\hat{\boldsymbol{s}}_{k_{t+1}}^i \leq \boldsymbol{q}(t)$. Nova metoda bazirana na ovoj strategiji opisana je Algoritmom 5.

Algorithm 5: T-SVD Method II

Input: Training: $\mathbf{I}_i$, $i = 1, 2, ..., N$; Test image $\mathbf{J}$, Truncation index k
Output: Fourier Domain Comparison of Test image against Compressed Representation of Training Set

for $i = 1$ **to** N **do**
  $\mathcal{L}(:, i, :) \leftarrow \mathbf{I}_i$
end
$\mathcal{M} \leftarrow$ mean image
$\mathcal{A} \leftarrow$ mean-deviation form of $\mathcal{L}$
Frontal face SVDs of $\hat{\mathbf{A}}^{(i)} \leftarrow \hat{\mathcal{A}}(:, :, i)$, $i = 1, \ldots \frac{n}{2} + 1$
Employ drop strategy; keep only $\hat{\mathbf{U}}^{(i)}(:, 1 : k_i)$
$\mathcal{T}(:, 1, :) \leftarrow$ twist$(\mathbf{J} - \mathcal{M})$
for $i = 1 : \frac{n}{2} + 1$, $t = 1 : k_i$, $j = 1 : N$ **do**
  $\hat{\mathbf{c}}_{t,j}^{(i)} \leftarrow (\hat{\mathbf{U}}^{(i)}(:, t))^H \hat{\mathbf{A}}^{(i)}(:, j)$
end
for $i = 1 : \frac{n}{2} + 1$, $t = 1 : k_i$, **do**
  $\hat{\mathbf{b}}_t^{(i)} \leftarrow (\hat{\mathbf{U}}^{(i)}(:, t))^H \hat{\mathbf{T}}^{(i)}(:, 1)$
end
for $j = 1 : N$ **do**
  Calculate $||\hat{\mathbf{b}}_t^{(i)} - \hat{\mathbf{c}}_{t,j}^{(i)}||_F$
end
Claim the training image whose coefficient is closest to that of the test image is the match.

Slika 6: T-SVD algoritam 2

3.4 Alternativa - tenzor OR pivotiranje

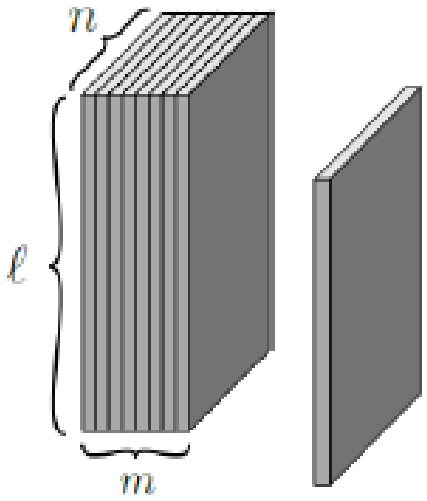
U ovom odijeljku predstavljamo jednu moguću alternativu već spomenutom T-SVD: T-PQR (tenzor OR pivotiranje) faktorizaciju. Razlog je potencijalna prednost updatea i downdate baze podataka. Mana ove metode jest da optimizacija neće biti optimalna. Dodavanje ili brisanje novog lateralnog slicea $\mathcal{L}$ baze podataka $\mathcal{A}$ veličine $l \times m \times n$ ilustrirana je slijedećom slikom:

Prvo, definirajmo permutacijski tenzor. Permutacijski tenzor $\mathcal{P} \in \mathbb{R}^{m \times m \times n}$ je tenzor čiji se ulazi sastoje samo od 0 i 1, i za koji vrijedi $\mathcal{P}^T * \mathcal{P} = \mathcal{P} * \mathcal{P}^T = \mathcal{I}$.

Teorem 3.6. *Neka je $\mathcal{A}$ $l \times m \times n$ realni tenzor. Tada se $\mathcal{A}$ može faktorizirati na slijedeći način:*

$$\mathcal{A} * \mathcal{P} = \mathcal{Q} * \mathcal{R} \tag{23}$$

gdje je $\mathcal{Q}$ $l \times l \times n$ ortogonalni tenzor, $\mathcal{R}$ je $l \times m \times n$ f-gornjetrokutasti tenzor,



Slika 7: Updating i downdating

a $\mathcal{P}$ je permutacijski tenzor.

Prednost korištenja permutacijskog tenzora je u permutiranju lateralni sliceova tako da je više energije sadržano u prvih nekoliko lateralni sliceova ortogonalnog $\mathcal{Q}$ tenzora. Ovo je slično korištenju permutacije stupaca kod matričnog QR algoritma da bi se postigla bolja rang svojstva. Analogno s Algoritmom 4, dolazimo do Algoritma 6.

Kada pričamo o updateu, pretpostavljamo da će se najčešće raditi o ažuriranju/updateu baze, što povećava lateralnu dimenziju za 1, tako da $l \times m \times n$ baza postaje $l \times (m+1) \times n$ baza. Dodavanje slike, implicitno znači da imamo još jedan lateralni slice originalnog tenzora. Kako se faktorizacija radi u Fourierovoj domeni, prvo moramo napraviti dodatni FFT duž "tube" niti slicea nove slike kako bi ju premjestili u Fourierovu domenu. Jednom kada je u Fourierovoj domeni, novi

Algorithm 6: Tensor QR Method

Input: Training: $\mathbf{I}_i$, $i = 1, 2, ..., N$, Testing: $\mathbf{J}$, Truncation index k
Output: Coefficients of testing images with a basis of reduced dimension

for $i = 1$ **to** N **do**
  $\mathcal{L}(:, i, :) \leftarrow \mathbf{I}_i$
end
$\mathcal{M} \leftarrow$ mean image
$\mathcal{A} \leftarrow$ mean-deviation form of $\mathcal{L}$
$[\mathcal{Q}, \mathcal{R}] \leftarrow$ tensor (pivoted) QR decomposition of $\mathcal{A}$
$\mathcal{C} \leftarrow \mathcal{Q}(:, 1:k, :)^T * \mathcal{A}$
$\mathcal{T}(:, 1, :) \leftarrow \text{twist}(\mathbf{J} - \mathcal{M})$
$\mathcal{B} \leftarrow \mathcal{Q}(:, 1:k, :)^T * \mathcal{T}$
for $j = 1$ **to** N **do**
  Calculate $||\mathcal{B} - \mathcal{C}(:, j, :)||_F$
end
Claim the training image whose coefficient is closest to that of the test image is the match.

Slika 8: Tensor QR metoda

stupac je dodan svakom $\hat{\boldsymbol{R}}^{(i)}$ i standardno matrično QR pivotiranje se odvija da se ažurira $\boldsymbol{P}^{(1)}, \hat{\boldsymbol{Q}}^{(1)}, \hat{\boldsymbol{R}}^{(i)}$. Ažurirana permutacija se dalje primjenjuje na prethodno uvećanu $\hat{\boldsymbol{R}}^{(i)}$, te se također računaju ažuriranja $\hat{\boldsymbol{Q}}^{(i)}, \hat{\boldsymbol{R}}^{(i)}$. Drugim riječima, prednost korištenja matričnog QR pivotiranja na matrici $\hat{\boldsymbol{A}}^{(i)}$ u Fourierovoj domeni naspram SVD na svakoj od ovih matrica jest to što je mnogo jeftinije ažurirati pivotiranu QR faktorizaciju nego SVD.