Task 1: Functia primeste o matrice “photo” si un numar intreg “k” si realizeaza o aproximare a matricei “photo” utilizand SVD. Mai intai initializam matricea finala “new_X” cu dimensiunea corespunzatoare matricei de intrare “photo”. Se initializeaza matricea finala, apoi matricea “photo” este convertita la tipul “double”. Se aplica algoritmul SVD asupra matricei “photo”, care descompune matricea in trei componente principale: matricea de vectori proprii stanga (“U”), matricea diagonala a valorilor singulare (“S”) si matricea de vectori proprii dreapta (“V”). Matricele “U”, “S” si “V” sunt redimensionate la dimensiunile corespunzatoare numarului k. Aceasta inseamna ca se pastreaza primele k coloane din “U” si “V” si primele k valori de pe diagonala din “S”. Se calculeaza aproximarea matricei initiale “photo” prin inmultirea matricelor “U”, “S” si “V” transpusa (“V”’). Matricea rezultata “new_X” este convertita la tipul “unit8” pentru a fi o imagine valida.

Task 2: Functia primeste o matrice “photo” si un numar intreg pcs si realizeaza o aproximare a matricei “photo” utilizand analiza componentelor principale. Mai intai se initializeaza matricea finala “new_X” cu dimensiunea corespunzatoare matricei de intrare “photo”. Se initializeaza matricea finala, apoi matricea “photo” este convertita la tipul “double”. Se calculeaza media pentru fiecare vector linie al matricei “photo” si se stocheaza in vectorul “medie”. Se actualizeaza matricea “photo” prin scaderea vectorului “medie” din fiecare vector linie. Se construieste matricea “Z” prin transpunerea matricei actualizate “photo”. Se aplica algoritmul SVD asupra matricei Z, obtinand matricile “U”, “S” si “V”. Se construieste matricea “W” din primele pcs coloane ale matricei V. Se calculeaza matricea Y prin inmultirea transpusa a matricei W cu matricea actualizata photo. Se aproximeaza matricea initiala “photo” prin inmultirea matricelor “W” si “Y” si adaugarea vectorului “medie”. Matricea rezultata “new_X” este convertita la tipul “unit8” pentru a fi o imagine valida.

Task 3: Functia primeste o matrice “photo” si un numar intreg pcs si realizeaza o aproximare a matricei “photo” utilizand analiza componentelor principale bazata pe matricea de covarianta. Mai intai se initializeaza matricea finala “new_X” cu dimensiunea corespunzatoare matricei de intrare “photo”. Se initializeaza matricea finala, apoi matricea “photo” este convertita la tipul “double”. Se calculeaza media pentru fiecare vector linie al matricei “photo” si se stocheaza in vectorul “medie”. Se normalizeaza matricea initiala “photo” prin scaderea mediei fiecarui rand din matrice, rezultand matricea “normalized”. Se calculeaza matricea de covarianta “Z” utilizand formula (normalized * normalized’) / (n - 1), unde n este numarul de coloane al matricei “normalized”. Se calculeaza vectorii si valorile proprii ale matricei de covarianta “Z” utilizand functia eig(Z). In acest context, vectorii proprii reprezinta directiile principale ale variatiei datelor. Valorile proprii returnate de functia “eig” nu sunt ordonate asa ca pentru a le ordona in ordine descrescatoare, se utilizeaza functia fliplr(V) pentru a inversa ordinea coloanelor matricei “V”. Se selecteaza primele pcs coloane din matricea “V” ordonata si se stocheaza rezultatul in matricea “W”. Aceste coloane reprezinta primele pcs componente principale asociate celor mai mari valori proprii. Se obtine matricea “Y” prin inmultirea transpusa a matricei “W” cu

matricea “normalized”. Se aproximeaza matricea initiala “photo” prin inmultirea matricelor “W” si “Y”, si adaugarea vectorului “medie”. Rezultatul este stocat in matricea new_X, care este convertita la tipul “unit8” pentru a fi o imagine valida.

Task 4: 1) visualise_image : Functia primeste o matrice de antrenament “train_mat” si un numar “number” si returneaza o imagine reprezentata de matricea “im”. Se initializeaza matricea “im” cu dimensiunea 28 x 28 cu valori de zero. Apoi se citeste linia cu numarul “number” din matricea de antrenament “train_mat” si se stocheaza in variabila “line”. Transformam linia citita “line” intr-o matrice 28 x 28 utilizand functia “reshape”, astfel incat linia sa fie reprezentata ca o imagine cu dimensiunea 28 x 28, matricea rezultata fiind transpusa. Matricea rezultata “im” este convertita la tipul “unit8” pentru a fi o imagine valida. 2) prepare_data: Functia primeste numele unui fisier de date si numarul de imagini de antrenament dorite si returneaza matricea “train_mat” si vectorul “train_val”. Se initializeaza variabila “n” cu valoarea 784, care reprezinta numarul total de pixeli ai unei imagini de dimensiune 28 x 28, matricea “train_mat” cu dimensiunea “no_train_images” x n, avand toate elementele initializate cu zero si vectorul “train_val” cu dimensiunea 1 x no_train_images, avand toate elementele initializate cu zero. Incarcam datele din tabelul specificat in fisierul cu numele “name” utilizand functia “load”. Datele sunt stocate in variabila “data”. Se copiaza primele “no_train_images” linii din tabelul de imagini de antrenament “data.trainX” in matricea “train_mat”. Se copiaza primele “no_train_images” valori ale vectorului de etichete “data.trainY” in vectorul train_val. 3) magic_with_pca: Functia primeste o matrice de antrenament “train_mat” si numarul de componente principale dorite pcs, si returneaza matricea “train” (aproximatia matricii initiale), vectorul medie “miu”, matricea “Y” (matricea obtinuta prin schimbarea bazei matricii initiale), si matricea de transformare “Vk” (primele pcs coloane din matricea de vectori proprii). Mai intai se obtin dimensiunile matricei de antrenament train_mat si se stocheaza in variabilele m si n. Apoi, se voi initializa “train” cu dimensiunea m x n, avand toate elementele zero, “miu” cu dimensiunea 1 x n, avand toate elementele zero , matricea “Y” cu dimensiunea m x pcs, avand toate elementele zero si se mai initializeaza matricea “Vk” cu dimensiunea n x pcs, avand toate elementele initializate cu zero. Pe urma, se converteste matricea de antrenament “train_mat” la tipul “double”. Se calculeaza media fiecarei coloane a matricei de antrenament “train_mat” si se stocheaza in vectorul miu. Se scade media “miu” din matricea de antrenament “train_mat”. Se calculeaza matricea de covarianta “cov_mat” a matricei de antrenament si apoi vectorii si valorile proprii ale acestei matrici. Se sorteaza valorile proprii in ordine descrescatoare si se obtine ordinea de sortare in variabila “aux”. Se reordoneaza matricea de vectori proprii “V” in functie de ordinea de sortare “aux”, astfel incat coloana cu cea mai mare valoare proprie sa fie prima coloana, si asa mai departe. Se selecteaza primele pcs coloane din matricea de vectori proprii “V” si se stocheaza in matricea “Vk”. Se creeaza matricea “Y” prin inmultirea matricei de antrenament normalizata cu matricea “Vk”, urmand sa se calculeze matricea “train” ca

produsul matricei “Y” cu transpusa matricei “Vk”. 4) prepare_photo: Functia primeste o imagine “im” si returneaza un sir de pixeli reprezentand imaginea ca un vector linie. Se initializeaza sirul “sir” cu dimensiunea 1 x 784, avand toate elementele initializate cu zero. Se inverseaza pixelii imaginii “im” prin scaderea valorii fiecarui pixel de la 255. Pe urma, se transpune imaginea “im”, astfel incat randurile devin coloane si coloanele devin randuri. Se transforma imaginea transpusa intr-un vector linie utilizand functia “reshape”. Parametrul [] specifica ca functia trebuie sa determine automat dimensiunile vectorului linie. Se stocheaza vectorul linie rezultat in variabila sir. 5) K-Nearest Neightbours: Functia realizeaza o predictie pe baza unei etichete “test” si unui set de etichete “Y” asociate unor vectori. Aceasta functie returneaza o predictie bazata pe cele mai apropiate k etichete din setul “Y”. Se initializeaza variabila “prediction” cu -1, urmand sa initializam vectorul “distance” cu dimensiunea m x 1 toate elementele acestuia fiind setate pe zero. De asemenea, se obtine dimensiunea matricei “Y” utilizand functia “size” si se stocheaza numarul de randuri in variabila “m” si numarul de coloane in variabila “n”.Pe urma, se calculeaza distanta euclidiana intre fiecare rand din matricea “Y” si vectorul “test”, si se stocheaza rezultatele in vectorul “distance”. Acest lucru se realizeaza prin parcurgerea randurilor matricei “Y” intr-un loop “for” si utilizarea functiei “norm” pentru a calcula norma euclidiana a diferentei intre randul curent si vectorul “test”. Se ordoneaza crescator distantele calculate si se stocheaza indicii ordonarii in variabila “indices”. Acest lucru se realizeaza utilizand functia “sort”. Se selecteaza primele k etichete din setul “labels” corespunzatoare celor mai apropiate k vectori din matricea Y. Acest lucru se realizeaza prin indexarea vectorului labels cu indicii primelor k elemente din vectorul indices.La final se calculeaza predictia ca mediana a celor k etichete selectate. Acest lucru se realizeaza utilizand functia “median”, stocandu-se predictia in variabila prediction. 6) classifyImage: Functia clasifica o imagine “im” pe baza unui set de date de antrenament “train_mat” si etichetelor asociate “train_val”, utilizand algoritmul PCA si algoritmul K-Nearest Neighbors. Functia returneaza o predictie pentru imaginea im. Se initializeaza variabila “prediction” cu -1. Se converteste imaginea “im” la tipul double. Se aplica functia 'magic_with_pca" asupra setului de date “train_mat”, utilizand parametrul pcs (Se calculeaza matricea de antrenament transformata prin PCA utilizand functia “magic_with_pca”, si se stocheaza rezultatul in variabila “train_mat_pca”). Ulterior, scadem din vectorul imagine media fiecarei coloane din train_mat si schimbam baza imaginii “im” prin inmultirea acesteiea cu matricea “Vk”. La final, se calculeaza predictia pentru imaginea “im” folosind algoritmul K-Nearest Neighbors implementat anterior cu k = 5. Se utilizeaza setul de etichete train_val si matricea train_mat_pca, predictia (rezultatul) stocandu-se in variabila “predicion”.