Deep Learning Exam - Bilet 3 Task Solutions

# Task 1: Recommendation System - Matrix Factorization

In this task, we implement a matrix factorization-based collaborative filtering method for building a recommendation system.   
By decomposing the user-item interaction matrix into two smaller matrices, we are able to generate predictions for missing values, which represent recommendations for the users.

Code:

import numpy as np  
from sklearn.decomposition import TruncatedSVD  
  
# Example user-item interaction matrix (e.g., users rating music tracks)  
R = np.array([  
 [5, 0, 3, 0],  
 [4, 0, 0, 2],  
 [1, 1, 0, 5],  
 [0, 0, 5, 4],  
 [2, 3, 0, 0]  
])  
  
# Matrix factorization using Singular Value Decomposition (SVD)  
svd = TruncatedSVD(n\_components=2) # Reduce matrix to 2 latent features  
U = svd.fit\_transform(R) # User matrix  
sigma = svd.singular\_values\_ # Singular values (diagonal matrix)  
Vt = svd.components\_ # Transposed item matrix  
  
# Reconstruct the original matrix (approximation)  
R\_approx = np.dot(U, np.dot(np.diag(sigma), Vt))  
  
# Output the approximated matrix, which includes predicted values  
print("Original Matrix (with missing values):")  
print(R)  
print("Approximated Matrix (predicted values):")  
print(np.round(R\_approx, 2))

# Task 2: Object Tracking - SORT Algorithm

In this task, we implement the Simple Online and Realtime Tracking (SORT) algorithm for tracking multiple objects in a video sequence.   
SORT uses a Kalman filter to predict object locations and the Hungarian algorithm for data association, making it suitable for tracking in real-time applications.

Code:

import numpy as np  
import cv2  
from sort import Sort # Make sure to have SORT implementation available  
  
# Initialize video capture (using a sample video file)  
cap = cv2.VideoCapture("video.mp4")  
  
# Initialize SORT tracker  
tracker = Sort()  
  
# Loop over video frames  
while True:  
 ret, frame = cap.read()  
 if not ret:  
 break  
  
 # Example: Detecting objects in the frame (replace with actual object detection)  
 # Let's assume we have a list of detections [x1, y1, x2, y2, confidence]  
 detections = np.array([  
 [100, 100, 200, 200, 0.9],  
 [150, 150, 250, 250, 0.8]  
 ])  
  
 # Update SORT tracker with detections  
 tracked\_objects = tracker.update(detections)  
  
 # Draw the tracked objects  
 for obj in tracked\_objects:  
 x1, y1, x2, y2, obj\_id = [int(v) for v in obj[:5]]  
 cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)  
 cv2.putText(frame, f'ID: {int(obj\_id)}', (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)  
  
 # Display the resulting frame  
 cv2.imshow("Frame", frame)  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
  
cap.release()  
cv2.destroyAllWindows()

# Task 3: Speech Recognition - RNN with MFCC Features

In this task, we build a simple speech-to-text system using Mel Frequency Cepstral Coefficients (MFCC) features.   
We extract MFCC features from an audio file and feed them into a Recurrent Neural Network (RNN) to convert the spoken language into text.

Code:

import numpy as np  
import librosa  
import tensorflow as tf  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import LSTM, Dense, Dropout  
  
# Load a sample audio file (replace with your own .wav file)  
audio\_path = 'audio\_sample.wav'  
y, sr = librosa.load(audio\_path, sr=16000)  
  
# Extract MFCC features  
mfcc = librosa.feature.mfcc(y=y, sr=sr, n\_mfcc=13)  
  
# Reshape MFCCs for the RNN input (samples, timesteps, features)  
mfcc = np.expand\_dims(mfcc.T, axis=0) # Add batch dimension  
  
# Build a simple RNN for speech-to-text  
model = Sequential()  
model.add(LSTM(128, return\_sequences=True, input\_shape=(mfcc.shape[1], mfcc.shape[2])))  
model.add(Dropout(0.5))  
model.add(LSTM(128))  
model.add(Dense(10, activation='softmax')) # Assuming 10 output classes for simplicity  
  
model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  
  
# Train the model on a speech dataset (placeholder, use actual speech data)  
# model.fit(mfcc, target\_labels, epochs=10)  
  
print("Model Summary:")  
model.summary()