Skeletal Solution for training and prediction phases:

Data Collection: Gather a diverse dataset of images or video frames containing falls. Include examples of falls specifically occurring at staircases, escalators, steps, etc.

Data PreProcessing:

We preprocess the data by resizing videos, extracting frames.

Semantic Embeddings:

Create semantic embeddings or attributes that describe falls.This can be done from domain knowledge, expert input, or learned directly from data using techniques like attribute learning.

Training Phase:

Initialize the chosen model and load the preprocessed data. Train the model using the collected fall dataset. Adjust the model architecture and hyperparameters as necessary.

Prediction Phase:

The chosen model to perform fall detection by leveraging semantic embeddings. Our input data is processed, and feature representations are extracted. These features are then matched with semantic embeddings or attributes associated with falls using similarity measures or other techniques.

Using similarity scores that are obtained,we apply thresholding techniques to determine whether a fall has occurred based on these scores . If the similarity score exceeds threshold, the input data is classified as a fall . Additional factors, like presence of obstacles or environmental factors, can be considered to improve detection accuracy.

Performance is evaluated based on validation set containing fall-related data, and the model is fine-tuned and hyperparameters adjusted to optimize detection accuracy and minimize false positives.

Code Template Using YOLO and Pose estimation algorithms:

This solution assumes that you have already collected and annotated a dataset of videos containing instances of people falling

```
import cv2
import numpy as np

# Load YOLO model and classes
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]

# Load Pose Estimation model
# (Code to load and initialize Pose Estimation model goes here)

# Function to perform fall detection
def detect_falls(frame):
    # Perform object detection using YOLO
    # (Code to perform object detection using YOLO goes here)
```

```
    # Perform pose estimation
    # (Code to perform pose estimation goes here)

    # Analyze detected objects and pose keypoints to identify falls
    # (Code to analyze detected objects and pose keypoints goes here)

    # Return True if fall detected, False otherwise
    # (Code to determine if fall detected goes here)

# Read input video
video_capture = cv2.VideoCapture("input_video.mp4")

# Process video frames
while True:
    ret, frame = video_capture.read()
    if not ret:
        break

    # Detect falls in the frame
    if detect_falls(frame):
        print("Fall detected!")

    # Display output frame
    cv2.imshow("Fall Detection", frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

# Release video capture and close windows
video_capture.release()
cv2.destroyAllWindows()
```