בהחלט. אני מבין את הרצון שלך לאתגר את התוכנית ולוודא שאנחנו בוחרים בנתיב הרובסטי והעוצמתי ביותר. זו גישה נכונה לחלוטין. בוא ניקח את ההמלצה האחרונה שלנו ל\*\*"דרך ההיברידית" (Transformer + VAE)\*\*, שהיא אכן בחזית הטכנולוגיה, ונפרט אותה לתוכנית עבודה מלאה, מקיפה ומדויקת שתוכל להטמיע ישירות.

התוכנית הבאה מתארת את בניית המנוע מההתחלה ועד הסוף, תוך התייחסות למגבלות של כל רכיב והסבר כיצד השילוב ביניהם יוצר מערכת חזקה יותר.

### **תוכנית בנייה היברידית (Transformer + VAE): מנוע לזיהוי מצבי שוק**

**גרסה 2.0**

#### **1. הקדמה: פילוסופיה ארכיטקטונית - "הטוב משני העולמות"**

* 1.1. מטרה:  
  לבנות מנוע אוטונומי לזיהוי מצבי שוק, המייצר ייצוג וקטורי רציף, עשיר ואינפורמטיבי, על ידי שילוב העוצמות של שתי טכנולוגיות מובילות.
* 1.2. פילוסופיה:  
  אנו מאמצים ארכיטקטורה היברידית ה פועלת בשני שלבים מוכללים:
  1. **שלב הבנת הרצף (Transformer):** רכיב הטרנספורמר, המומחה העולמי בהבנת נתונים סדרתיים, "יקרא" את היסטוריית התנהגות השוק (שמיוצגת על ידי מאפייני MMD) ויזקק ממנה וקטור תמציתי של "תובנות".
  2. **שלב מיפוי המצב (VAE):** רכיב ה-VAE יקבל את וקטור התובנות המזוקק מהטרנספורמר וימקם אותו על "מפת שוק" הסתברותית ומאורגנת.
* **1.3. טיפול במגבלות באופן מובנה:**
  + **הטרנספורמר פותר את בעיית "איכות הקלט" של ה-VAE:** הוא פועל כמסנן רעשים ומזהה תבניות אוטומטי, ומספק ל-VAE קלט נקי ואיכותי.
  + **ה-VAE פותר את בעיית ה"קופסה השחורה" של הטרנספורמר:** הוא נותן מבנה וארגון לפלט העוצמתי אך הלא-מובנה של הטרנספורמר, ומאפשר לנו לנתח ולצייר את "מפת השוק" הנלמדת.

#### **2. חלק א' - תהליך ההכנה והאימון (Offline)**

שלב זה מתבצע פעם אחת כדי ליצור את המודל ההיברידי המאומן.

* **2.1. שלב 1: הנדסת מאפייני MMD (יצירת הקלט הגולמי)**
  + תהליך זה נשאר זהה לתוכנית הקודמת. התוצר הוא סט נתונים היסטורי, כאשר לכל נקודת זמן (כל 30 דקות) יש MMD\_Feature\_Vector עשיר המחושב על בסיס חלון של N נרות קודמים.
* **2.2. שלב 2: הגדרת הארכיטקטורה ההיברידית (PyTorch)**
  + נבנה מודל אחד, HybridRegimeEncoder, המכיל את שני הרכיבים.
  + **2.2.1. רכיב הטרנספורמר (Transformer Encoder Block):**
    - **קלט:** מטריצה בגודל N x F (למשל, 100 x 10), כאשר N הוא אורך הרצף ו-F הוא גודל ה-MMD\_Feature\_Vector.
    - **ארכיטקטורה:**
      * שכבת PositionalEncoding כדי להוסיף מידע על מיקום ברצף.
      * 1-2 שכבות של nn.TransformerEncoderLayer, המכילות כל אחת Multi-Head Self-Attention ורשת Feed-Forward.
      * **היפר-פרמטרים מומלצים להתחלה:** nhead=2 (מספר ראשי קשב), dim\_feedforward=256.
    - **פלט:** וקטור תמצית אחד בגודל D\_model (למשל, 64), המייצג את ההבנה של הטרנספורמר את כל הרצף.
  + **2.2.2. ראש ה-VAE (VAE Head):**
    - **קלט:** וקטור התמצית בגודל D\_model (64) מהטרנספורמר.
    - **ארכיטקטורה:**
      * שתי שכבות ליניאריות נפרדות: fc\_mu = nn.Linear(64, 8) ו-fc\_log\_var = nn.Linear(64, 8). (בדוגמה זו, גודל המרחב הסמוי הוא 8).
      * לוגיקת דגימה (reparameterization trick) ליצירת וקטור המצב הסופי z.
    - **פלט:** קואורדינטה במרחב הסמוי, Regime\_Vector בגודל 8.
* **2.3. שלב 3: אימון המודל ההיברידי**
  + המודל כולו מאומן **מקצה לקצה (End-to-End)** באופן **בלתי-מפוקח**.
  + **תהליך:**
    1. לוקחים אצווה (batch) של רצפים היסטוריים (מטריצות N x F).
    2. מעבירים כל רצף דרך הטרנספורמר ואז דרך ראש ה-VAE לקבלת Regime\_Vector (z).
    3. מעבירים את z דרך מפענח (Decoder) שמנסה לשחזר את הקלט המקורי (או את וקטור התמצית מהטרנספורמר).
    4. **פונקציית ההפסד (Loss):** מחשבים את ההפסד המשולב של VAE (שגיאת שחזור + KL Divergence).
    5. מבצעים backpropagation ומעדכנים את המשקולות **של הטרנספורמר ושל ה-VAE יחד**.
  + **התוצר הסופי:** קובץ מודל מאומן אחד, **hybrid\_regime\_engine.pth**, המכיל את כל המשקולות של המערכת ההיברידית.

#### **3. חלק ב' - תהליך תפעולי בזמן אמת (Online)**

זהו התהליך הרזה והמהיר שרץ כל 30 דקות.

* **3.1. שלב 1: אתחול המערכת**
  + טוענים את המודל המאומן hybrid\_regime\_engine.pth לזיכרון ומעבירים אותו למצב הערכה (model.eval()).
* **3.2. שלב 2: זרימה חיה**
  1. **טריגר:** אירוע NEW\_30MIN\_BAR.
  2. **חישוב והרכבת מטריצה:** המערכת מחשבת את ה-MMD\_Feature\_Vector העדכני ומרכיבה את מטריצת ה-N x F המלאה.
  3. **הסקה מהירה (Inference):**
     + המטריצה כולה מוזנת למודל ההיברידי הטעון.
     + המודל מבצע מעבר קדימה (forward pass) דרך הטרנספורמר ודרך ראש ה-VAE.
     + **השלב הזה יואץ באמצעות TensorRT**.
  4. **הפלט הסופי:** המודל מוציא וקטור רציף אחד: **Regime\_Vector** (למשל, בגודל 8). זוהי קואורדינטת מצב השוק הנוכחית.
* **3.3. שלב 3: אינטגרציה עם מערכת ה-MARL הראשית**
  + ה-Regime\_Vector הרציף (בגודל 8) מועבר לרכיב הראשי.
  + הוא **משרשר** אותו לווקטורים מה-LSTM של סוכני 30 ו-5 דקות: Final\_State = concatenate([Vector\_30m, Vector\_5m, Regime\_Vector])
  + ה-Final\_State המאוחד והעשיר הזה הוא הקלט הסופי למדיניות המשותפת של מערכת ה-MARL הראשית.

תוכנית זו מייצגת את הגישה המקיפה והעמידה ביותר שגיבשנו. היא ממנפת את הטכנולוגיות המתקדמות ביותר לעיבוד רצפים (Transformer) וללמידת ייצוג הסתברותי (VAE), תוך שהיא מטפלת באופן מובנה במגבלות של כל אחת מהן. זהו מתווה שלם שניתן לקחת ישירות לשלב המימוש.