Quiz 7 - *Neural Networks for Interpretable Time Series Forecasting*

* Indicates required question

1. Name *

2. Email *

The Transformer Architecture - Trading Signal Prediction

You are a quantitative analyst tasked with building a Transformer model to predict trading signals for financial markets. Your goal is to develop an automated trading system that can make informed Long/Short/Neutral position decisions.

Your Challenge:

- Analyze historical market data to predict future trading positions
- Make sequential decisions where each day's position may depend on previous decisions
- Handle multi-step forecasting with uncertainty quantification

Input Data:

- **Historical Features:** T_x = 20 days of past market data for each trading sequence
- **Feature Types:** Each day contains multiple indicators
- **Data Structure:** Each historical day is represented as a vector with D dimensions

Target Predictions (What You Want to Predict):

- **Trading Signals:** Position decisions for the next T_y = 5 trading days
- **Signal Types:** For each future day, decide between:
  - **Long** position
  - **Short** position
  - **Neutral:** No position
- **Sequential Nature:** Each day's decision may influence subsequent decisions

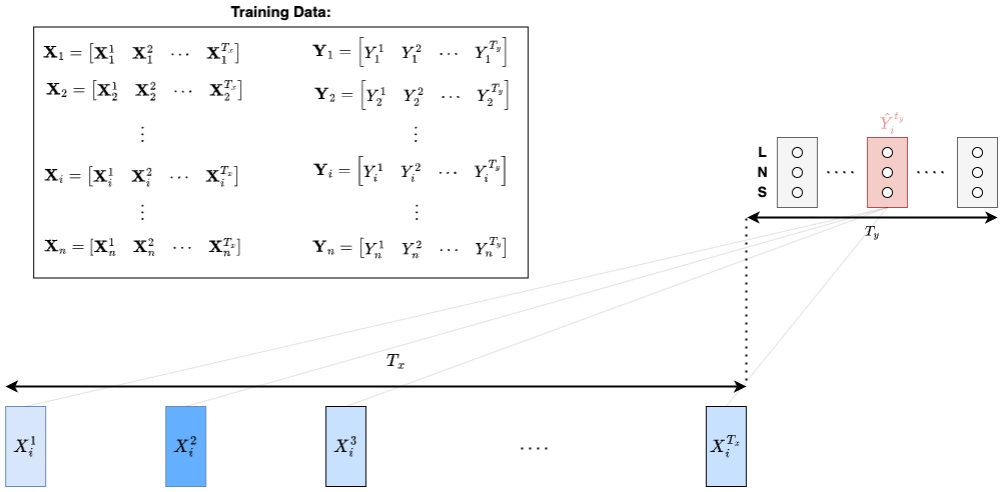
Traditional models struggle with this problem because:

- **Long-term Dependencies:** Today's decision might depend on patterns from weeks ago
- **Sequential Context:** Future decisions should consider previous position choices

- **Multi-step Prediction:** Need to predict multiple days ahead simultaneously
- **Attention Mechanism:** Different historical periods may be relevant for different prediction horizons

The Transformer architecture addresses these challenges through its encoder-decoder structure with attention mechanisms.

The Training Data:



3. In the training data, what does T_x represent? 1 point

Mark only one oval.

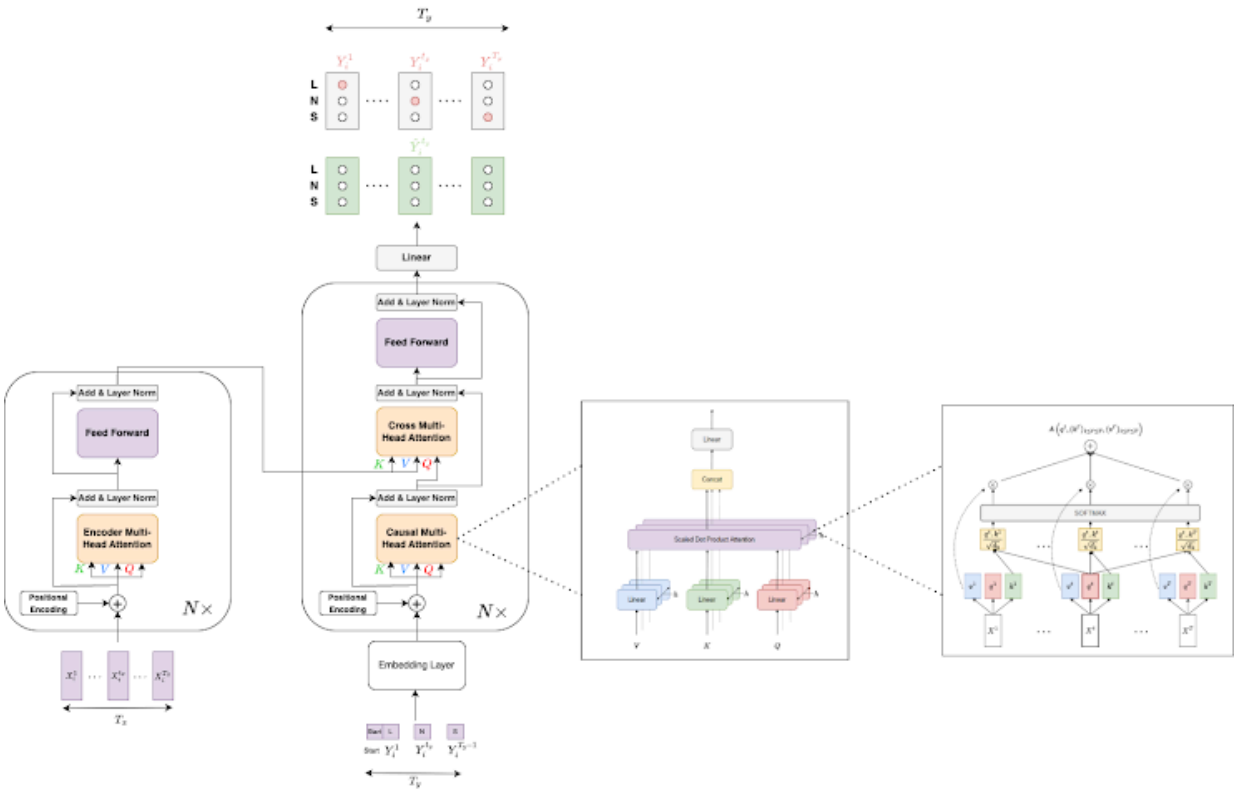
- The number of historical time steps (lookback window) used for prediction
- The number of future days to predict
- The number of trading sequences in the dataset

4. If you have N=1000 training sequences, with T_x=20 historical days, T_y=5 prediction days, and D=8 features per day, what are the shapes of your training tensors? 1 point

Mark only one oval.

- Input features: (1000, 20, 8), Target labels: (1000, 3)
- Input features: (1000, 20, 8), Target labels: (1000, 5)
- Input features: (1000, 28), Target labels: (1000, 5)

The Transformer Architecture: Teacher Forcing Approach



The Training Process: Using Teacher Forcing

Looking at the Transformer architecture diagram, we can see how teacher forcing works during the training process for our trading signal prediction problem.

Teacher Forcing is a training technique where the model is given the **true previous outputs** as input during training, rather than using its own predictions. This allows for efficient parallel training and stable learning.

The encoder processes the historical market data:

- **Input:** Historical feature sequence $[X_1^1, X_1^2, ..., X_1^{Tx}]$
- **Processing:** Through positional encoding → Multi-Head Attention → Feed Forward layers
- **Output:** Contextual representations of the historical data

Looking at the decoder input in the diagram, we see:

- **Decoder Input Sequence:** $[Start, Y_1^1, Y_1^t, ..., Y_1^{(Ty-1)}]$
- **Target Output Sequence:** $[Y_1^1, Y_1^t, ..., Y_1^{Ty}]$

How Teacher Forcing Works

Step 1: Input Preparation

- The decoder receives the **true previous trading positions** as input
- For example: [Start, Long, Short, Neutral, Long]
- These are the **actual correct labels** from the training data, not the model's predictions

Step 2: Embedding Layer

- Each discrete label (Start, Long, Short, Neutral) gets converted to an embedding vector
- The embedding layer maps: {Start, Long, Short, Neutral} → continuous vectors in R^d_model

Step 3: Parallel Processing

- All decoder positions process simultaneously during training
- Position 1 sees: [Start] → should predict Y_1^1
- Position 2 sees: [Start, Y_1^1] → should predict Y_1^2
- Position 3 sees: [Start, Y_1^1, Y_1^2] → should predict Y_1^3
- And so on...

Step 4: Causal Masking

- The "Causal Multi-Head Attention" ensures each position only sees previous positions
- This maintains the autoregressive property even during parallel training

Step 5: Cross-Attention

- Each decoder position attends to ALL encoder outputs (historical market data)
- This allows each trading decision to consider the full historical context

Step 6: Output Generation

- Linear layer converts decoder outputs to class probabilities
- Shape: (N,Ty, 3) for [Long, Short, Neutral] probabilities

5. What is the primary advantage of using teacher forcing during training of the Transformer decoder? 1 point

Mark only one oval.

- ◯ It reduces the model size and memory requirements
- ◯ It allows parallel processing of all decoder positions simultaneously, making training much faster than sequential generation
- ◯ It eliminates the need for attention mechanisms in the decoder

6. During training with teacher forcing on a batch of N_b sequences, the decoder processes inputs and produces output predictions. How is the loss function calculated across the entire batch? 1 point

Mark only one oval.

◯ Only the final predictions are compared to true labels:

$$\mathcal{L} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(- \sum_{k=1}^{3} Y_i^{T_y}[k] \log \hat{Y}_i^{T_y}[k] \right)$$

◯ Each prediction is compared to its corresponding true label:

$$\mathcal{L} = \frac{1}{N_b \cdot T_y} \sum_{i=1}^{N_b} \sum_{t=1}^{T_y} \left(- \sum_{k=1}^{3} Y_i^{t}[k] \log \hat{Y}_i^{t}[k] \right)$$

◯ Only the first predictions are used for loss calculation:

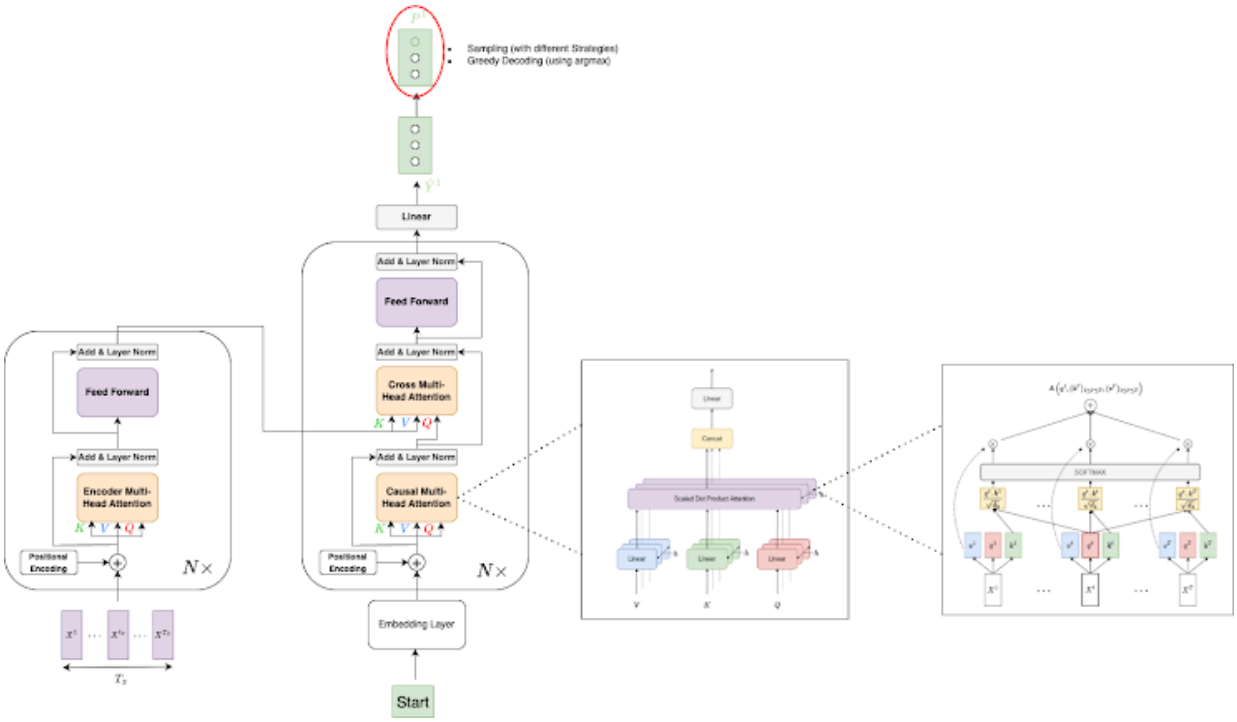
$$\mathcal{L} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(- \sum_{k=1}^{3} Y_i^{1}[k] \log \hat{Y}_i^{1}[k] \right)$$

7. During training, to predict output sequence [Y_1^1, Y_1^2, Y_1^3, Y_1^4, Y_1^5], the decoder uses input [Start, Y_1^1, Y_1^2, Y_1^3, Y_1^4]. When we have a new test input sequence and want to predict the output sequence, why is this training approach impossible during inference? 1 point

Mark only one oval.

- ◯ The test input sequence has different dimensions than the training data
- ◯ The attention mechanism works differently during inference than during training
- ◯ We don't know the true future trading positions Y_1^1, Y_1^2, Y_1^3, Y_1^4 that we need as decoder input (That's exactly what we're trying to predict!)

First Step of Inference - Diagram -



First Step of Inference in Transformer: From Historical Sequence to First Trading Prediction

Given a new test sequence of historical market data [X¹, X², ..., X^Tx], let's trace through the first step of inference to generate the first trading position Y¹.

Phase 1: Encoder Processing (Done Once)

Input: Historical sequence [X¹, X², ..., X^Tx]

- Each X^t represents market features for day t (prices, volume, technical indicators)
- Shape: (1, Tx, D) for a single test sequence

Encoder Flow:

1. **Positional Encoding:** Add position information to each time step
2. **Multi-Head Self-Attention:** Each historical day attends to all other historical days
3. **Feed Forward:** Process attended representations
4. **Output:** Encoder representations H = [h¹, h², ..., h^Tx]

Key Point: These encoder outputs H are computed once and will be reused for all T_y prediction steps.

Phase 2: First Decoder Step

Initial Decoder Input: Only the [Start] token

- The Start token gets converted to an embedding vector through the Embedding Layer
- Shape: (1, 1, d_model): single sequence, single token, d_model dimensions

Decoder Processing:

1. **Positional Encoding:** Add position encoding to the Start token embedding

2. **Causal Multi-Head Self-Attention:**

    - Input: Only the Start token representation
    - Output: Self-attended Start token (trivial since there's only one token)

3. **Cross Multi-Head Attention:**

    - **Queries (Q):** From the Start token representation
    - **Keys (K) and Values (V):** From ALL encoder outputs H = [h^1, h^2, ..., h^Tx]
    - **Process:** The Start token attends to the entire historical sequence
    - **Result:** Context-aware representation that considers all historical market data

4. **Feed Forward:** Further processing of the attended representation

5. **Linear Layer:** Projects the final representation to class logits

    - Input: (1, 1, d_model)
    - Output: (1, 1, 3) Probability Distribution for [Long, Short, Neutral]

6. **Decision Making:**

    - **Sampling:** Apply softmax and sample according to probabilities
    - **Greedy Decoding:** Take argmax to get the most likely class
    - **Result:** First trading position P^1

What Happens in This First Step
The Attention Mechanism ?

The model essentially asks: *"Given this entire market history, what should be my first trading position?"*

The cross-attention mechanism allows the Start token to:

- Weight the importance of different historical periods
- Combine information from multiple time steps
- Make an informed decision based on the full context

Result:

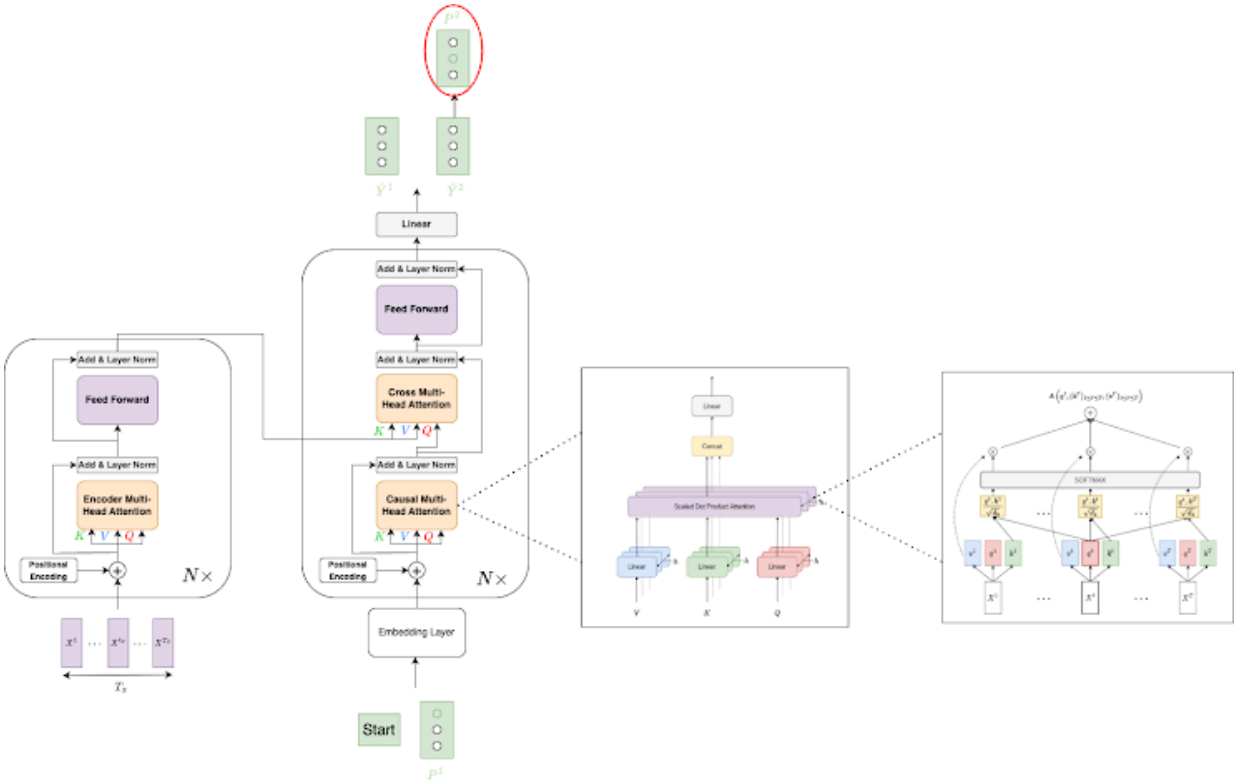
- **Prediction:** $P^1 \in$ {Long, Short, Neutral}
- **Next Input:** For step 2, the decoder will use [Start, P^1]

8. During the first step of inference, the decoder starts with only the [Start] token to predict the first trading position P^1. In the cross-attention layer, what serves as the Queries (Q), Keys (K), and Values (V)? 1 point

   Mark only one oval.

   - ◯ Q = Start token representation, K = V = previous predictions
   - ◯ Q = Start token representation, K = V = encoder outputs from historical sequence [X^1, X^2, ..., X^Tx]
   - ◯ Q = encoder outputs, K = V = Start token

Second Step of Inference - Diagram -



Second Step of Inference in Transformer From First Prediction to Second Trading Decision

After the first step of inference generated P^1, we now move to the second step to predict P^2. The diagram shows how the decoder input has grown from [Start] to [Start, P^1].

What We Have:

- **Encoder Outputs:** H = [h^1, h^2, ..., h^Tx] (unchanged from step 1)
- **First Prediction:** P^1 (e.g., "Long")
- **New Decoder Input:** [Start, P^1]

Goal: Predict the second trading position P^2

Growing Context:

- **Step 2 Input:** [Start, P^1]
- Each token gets converted to embeddings through the Embedding Layer
- Shape: (1, 2, d_model):single sequence, two tokens, d_model dimensions

Decoder Processing

1. Positional Encoding:

- Add position encodings to both Start and P^1 embeddings
- Position 1: Start + PE^1
- Position 2: P^1 + PE^2

2. Causal Multi-Head Self-Attention:

- **Position 1 (Start):** Can only attend to itself
- **Position 2 (P^1):** Can attend to both Start and P^1

3. Cross Multi-Head Attention:

- **Queries (Q):** From both decoder positions [Start, P^1]
- **Keys (K) and Values (V):** From ALL encoder outputs H = [h^1, h^2, ..., h^Tx]
- **Process:** Both positions attend to the historical market data
- **Key Insight:** The P^1 position can now make decisions based on:
    - The historical market context (through cross-attention)
    - The previous trading decision (through self-attention)

4. Feed Forward Processing:

- Apply feed-forward networks to both positions

5. Output Generation:

- **Linear Layer:** Projects representations to class logits
- **Focus:** Only the LAST position (position 2, corresponding to P^1) produces the next prediction
- **Output:** Probability Distribution for $P^2 \in$ {Long, Short, Neutral}

What's Different in Step 2 ?

Unlike step 1, the model now has:

- **Sequential Context:** Knowledge of the first trading decision P^1
- **Historical Context:** Still has access to all market history

- **Combined Decision Making:** Can make P^2 based on both market trends AND the previous position

Self-Attention Impact ?

The causal self-attention allows the model to ask:
"Given that I decided to go Long yesterday (P^1), and considering all the market history, what should I do today (P^2)?"

Key Insights
Growing Sequence Length

- **Step 1:** Input length = 1 [Start]
- **Step 2:** Input length = 2 [Start, P^1]
- **Step 3:** Input length = 3 [Start, P^1, P^2]
- **Pattern:** Each step adds one more token to the context

Autoregressive Nature

Each prediction depends on ALL previous predictions:

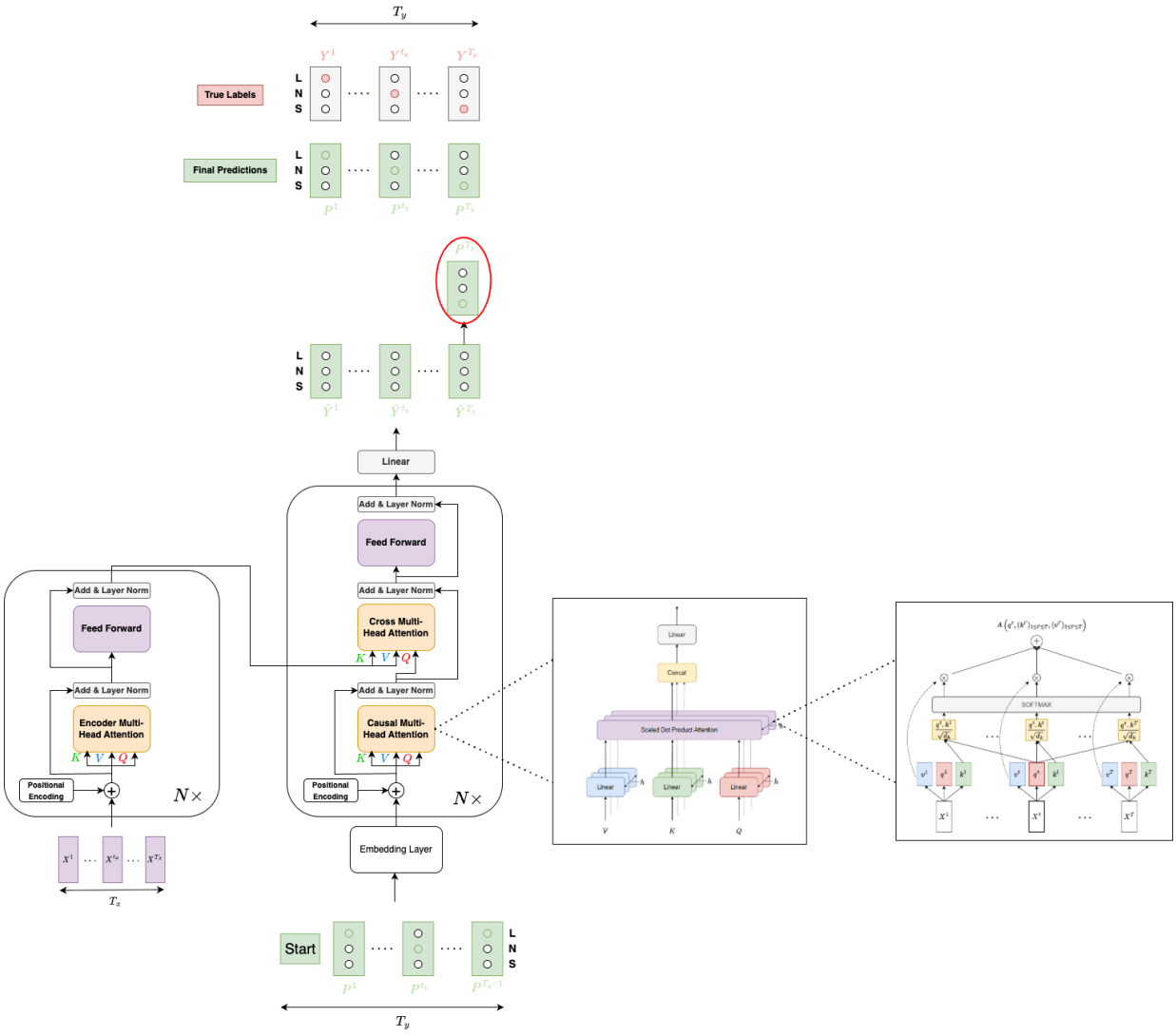
- P^2 depends on P^1
- P^3 will depend on P^1 AND P^2
- This creates coherent trading sequences

Result:

- **Prediction:** $P^2 \in$ {Long, Short, Neutral}
- **Next Input:** For step 3, decoder will use [Start, P^1, P^2]

This autoregressive process continues until all T_y positions are predicted, with each step building upon the previous trading decisions while maintaining access to the full historical market context.

Final Output of the Inference Process - Diagram -



Final Output of the Inference Process Complete Sequence Generation

After T_y autoregressive steps, the inference process generates the complete trading sequence:

Final Prediction Sequence: [P^1, P^2, P^3, ..., P^T_y]

Last Step (T_y):

- **Decoder Input:** [Start, P^1, P^2, ..., P^(T_y-1)]
- **Output:** Final trading position P^T_y
- **Result:** Complete 5-day trading strategy

Model Evaluation

Once inference is complete, we compare the model's predictions against the true labels:

Model Predictions: [P^1, P^2, P^3, P^4, P^5]
True Labels: [Y^1, Y^2, Y^3, Y^4, Y^5]

Evaluation Metrics:

- **Accuracy:** How many positions match exactly (P^t = Y^t)
- **Sequence Accuracy:** Does the entire sequence match perfectly
- **Position-wise Analysis:** Performance by prediction horizon (day 1 vs day 5)

Key Insight: The autoregressive nature means that early prediction errors (e.g., wrong P^1) can compound and affect all subsequent predictions, making sequence-level evaluation crucial for assessing real trading performance.

9. When predicting the fourth trading position P^4, what information does the model have access to during inference? 1 point

Mark only one oval.

- Only the historical market data [X^1, X^2, ..., X^Tx] through the encoder outputs
- Only the previous predictions [Start, P^1, P^2, P^3] through the decoder self-attention
- Both the complete historical market data [X^1, X^2, ..., X^Tx] (via cross-attention) AND the sequence of previous predictions [Start, P^1, P^2, P^3] (via causal self-attention)

10. During inference step 3, the decoder input is [Start, P^1, P^2] and we want to predict P^3. What does the causal self-attention mask look like, and why is this masking necessary? 1 point

Mark only one oval.

- Attention mask allows all positions to see each other
- Causal mask prevents future information leakage
- No masking is needed during inference since we're not training

Volatility Forecasting using the Temporal Fusion Transformer

The forecasting problem involves predicting future realized volatility for 31 financial stock indices. We need to understand the different types of information available and how they contribute to making accurate predictions.

Input Features: Three Types

The TFT model processes three distinct categories of input features, each providing different types of information:

1. Static Features

- **Definition**: Characteristics that never change for each financial index
- **Properties**: These remain constant throughout the entire time period
- **Examples**:
    - Index category (developed markets vs. emerging markets)
    - Geographic region (North America, Europe, Asia)
    - Market size classification
    - Base currency of the index
- **Role**: Provide context about the fundamental nature of each index

2. Time-Varying Known Features

- **Definition**: Information that changes over time but is known in advance
- **Properties**: We can observe these values for both past and future time periods
- **Examples**:
    - Calendar information (day of the week, month, holidays)
    - Scheduled economic announcements
    - Predetermined policy meetings
    - Known seasonal patterns

- **Role**: Help the model understand predictable patterns and upcoming events

3. Time-Varying Unknown Features

- **Definition**: Information that changes over time but is only available for historical periods
- **Properties**: We can only observe these values up to the current time
- **Examples**:
    - Past realized volatility values
    - Historical trading volumes
    - Market sentiment from previous days
    - Unexpected market events or news shocks

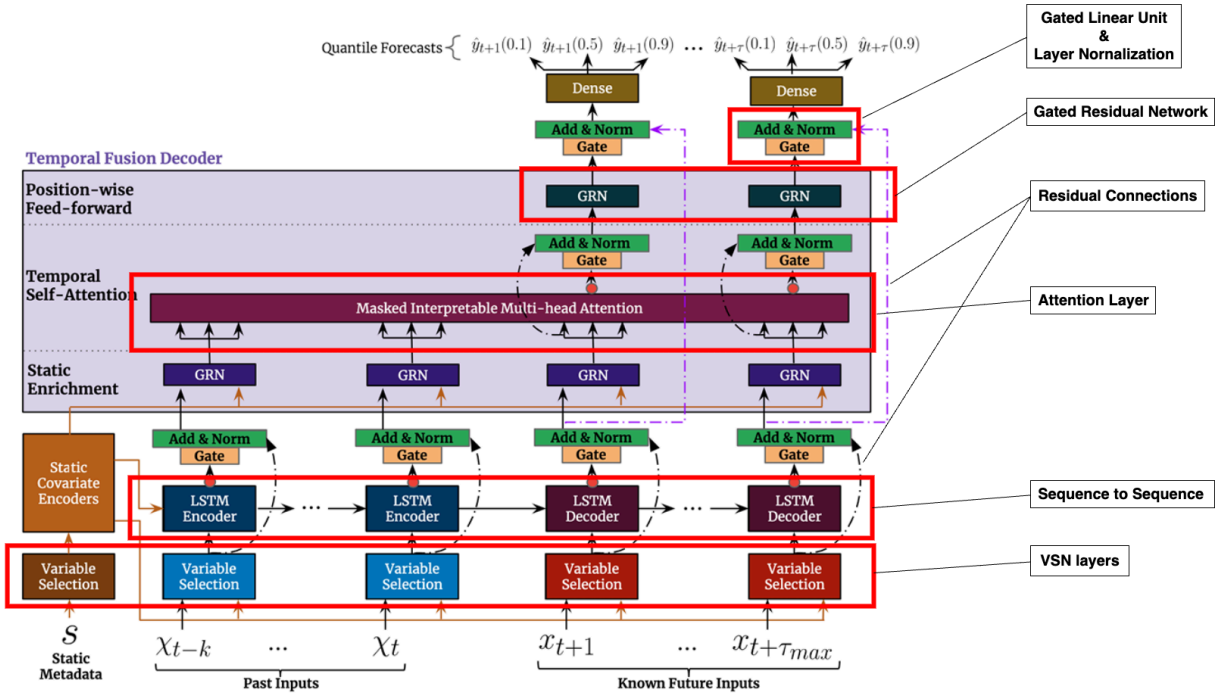
- **Role**: Capture the actual market dynamics and patterns from historical data

Forecasting Goal

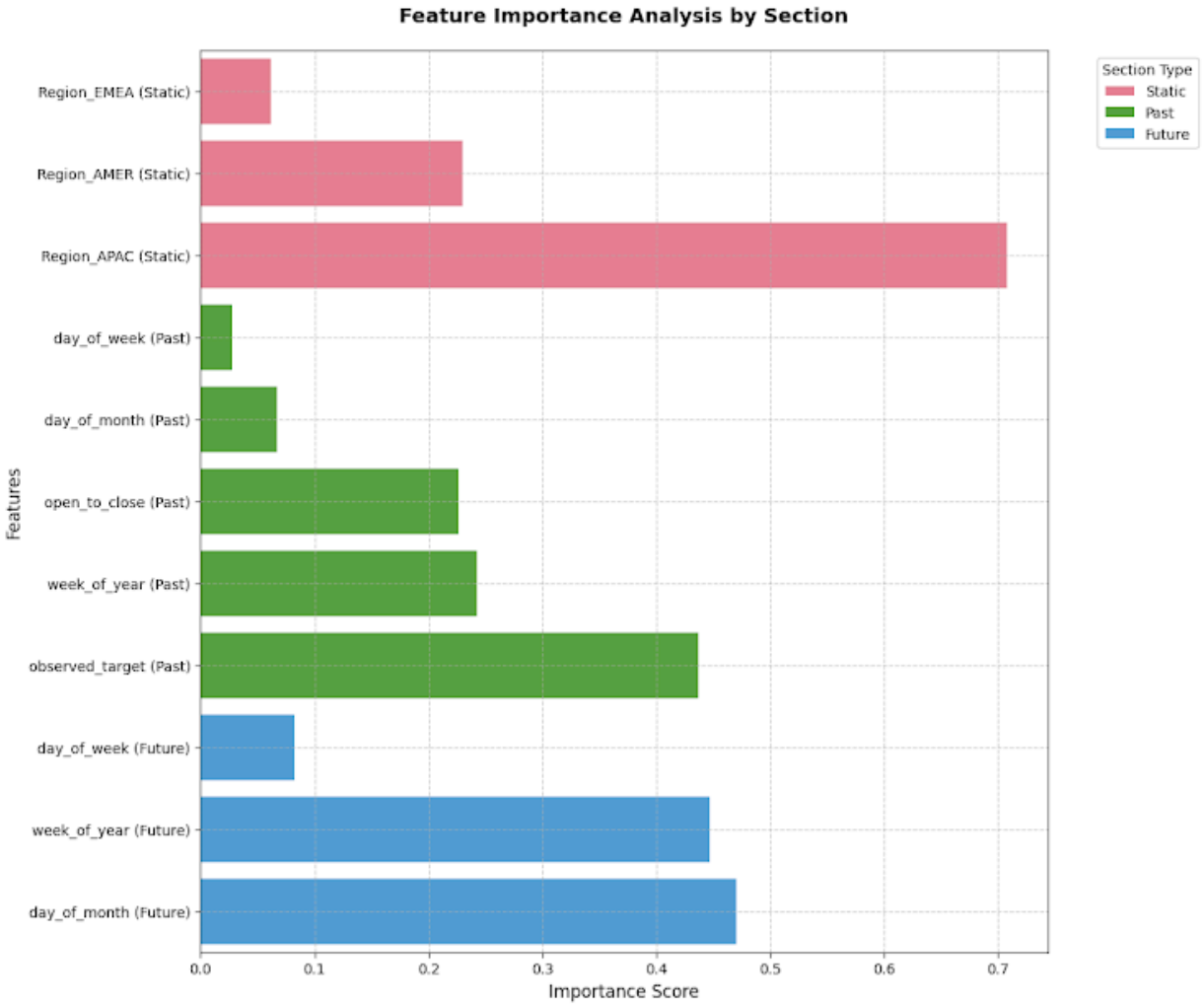
Primary Objective: Use all available information to predict the daily realized volatility for each of the 31 financial indices over multiple future time periods.

What the Model Does:
The TFT model takes the three types of features and learns how to combine them optimally to generate accurate forecasts. It considers the unique characteristics of each index (static features), incorporates known future information (known features), and learns from historical market patterns (unknown features).

The TFT Architecture:



11. What is the primary purpose of Variable Selection Networks (VSN) in the TFT architecture? 1 point



Mark only one oval.

- ◯ To generate future predictions
- ◯ To dynamically select the most relevant features from static, time-varying known, and time-varying unknown features
- ◯ To visualize attention weights

12. Which of the following is NOT mentioned as a component of the TFT architecture? 1 point

Mark only one oval.

- ◯ Variable Selection Networks
- ◯ Masked Multi-Head Attention
- ◯ Convolutional Neural Networks (CNN)

Comments - Questions ?

13.