Deep Learning Statistical Arbitrage

Oualid Missaoui

Agenda

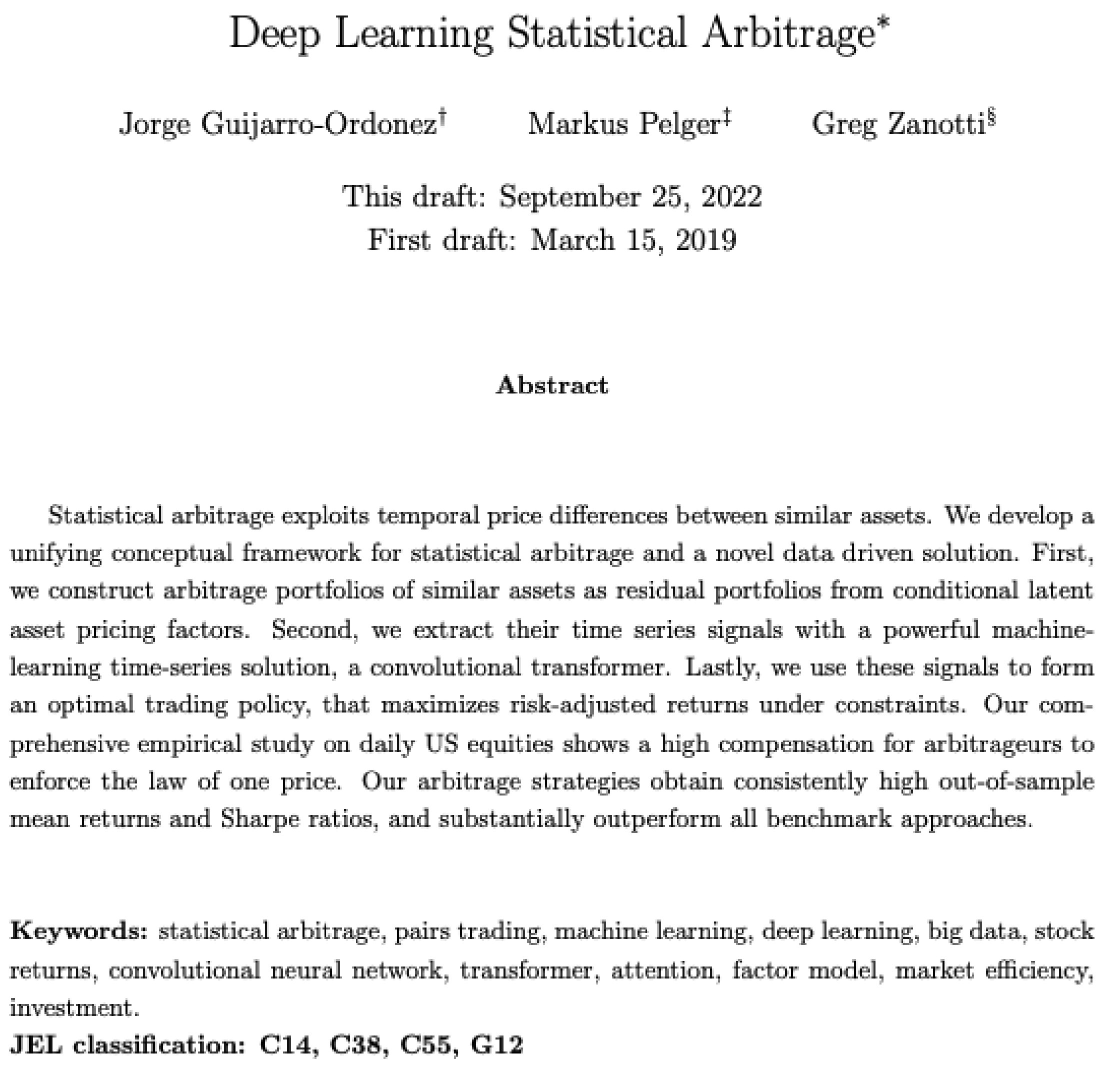
- Overview
- Architecture
- Empirical Results
- References

Overview

Linkedin

Stanford

Linkedin



Deep Learning Statistical Arbitrage*

Jorge Guijarro-Ordonez† Markus Pelger‡ Greg Zanotti§

This draft: September 25, 2022
First draft: March 15, 2019

Abstract

Statistical arbitrage exploits temporal price differences between similar assets. We develop a unifying conceptual framework for statistical arbitrage and a novel data driven solution. First, we construct arbitrage portfolios of similar assets as residual portfolios from conditional latent asset pricing factors. Second, we extract their time series signals with a powerful machine-learning time-series solution, a convolutional transformer. Lastly, we use these signals to form an optimal trading policy, that maximizes risk-adjusted returns under constraints. Our comprehensive empirical study on daily US equities shows a high compensation for arbitrageurs to enforce the law of one price. Our arbitrage strategies obtain consistently high out-of-sample mean returns and Sharpe ratios, and substantially outperform all benchmark approaches.

Keywords: statistical arbitrage, pairs trading, machine learning, deep learning, big data, stock returns, convolutional neural network, transformer, attention, factor model, market efficiency, investment.
JEL classification: C14, C38, C55, G12

Source

Overview

Take any big equity universe.

Strip out all risk premia with an asset-pricing model, so you’re left with residual ‘mispricing’ portfolios.

Then train a deep time-series model (CNN + Transformer) **not to predict returns**, but directly to **maximize Sharpe** (or mean–variance) of a long–short portfolio of those residuals, under realistic constraints.

Architecture

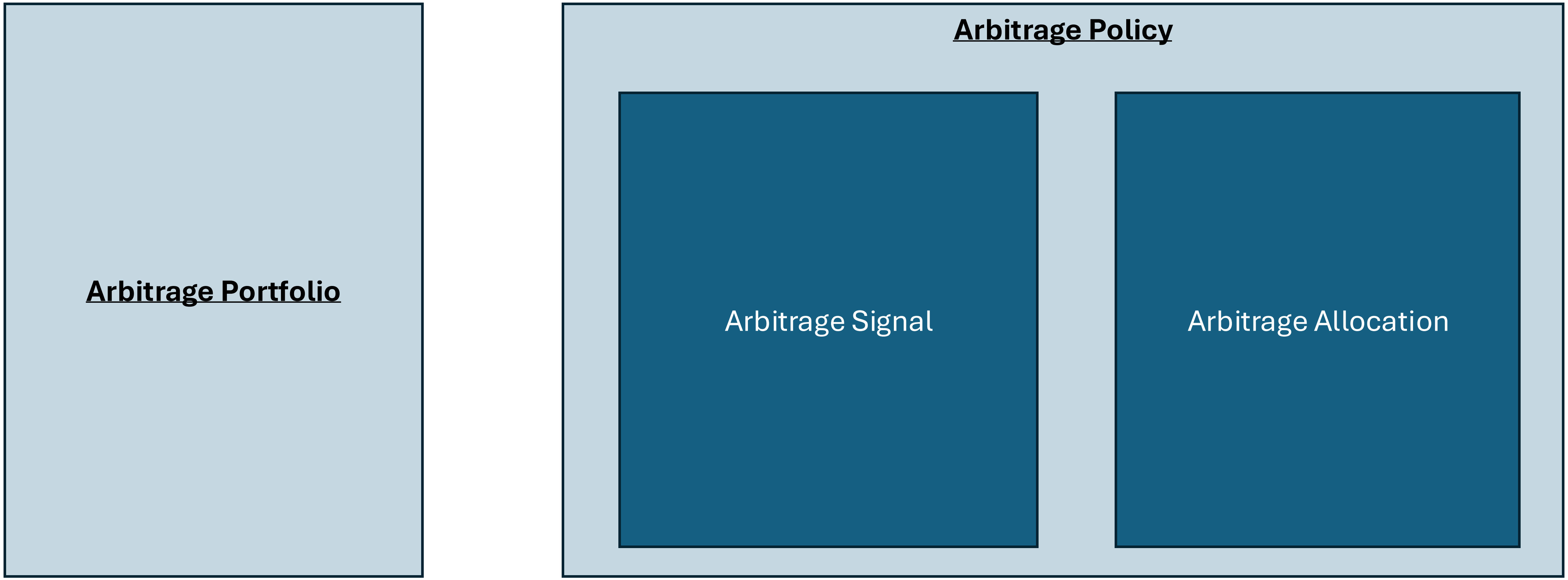
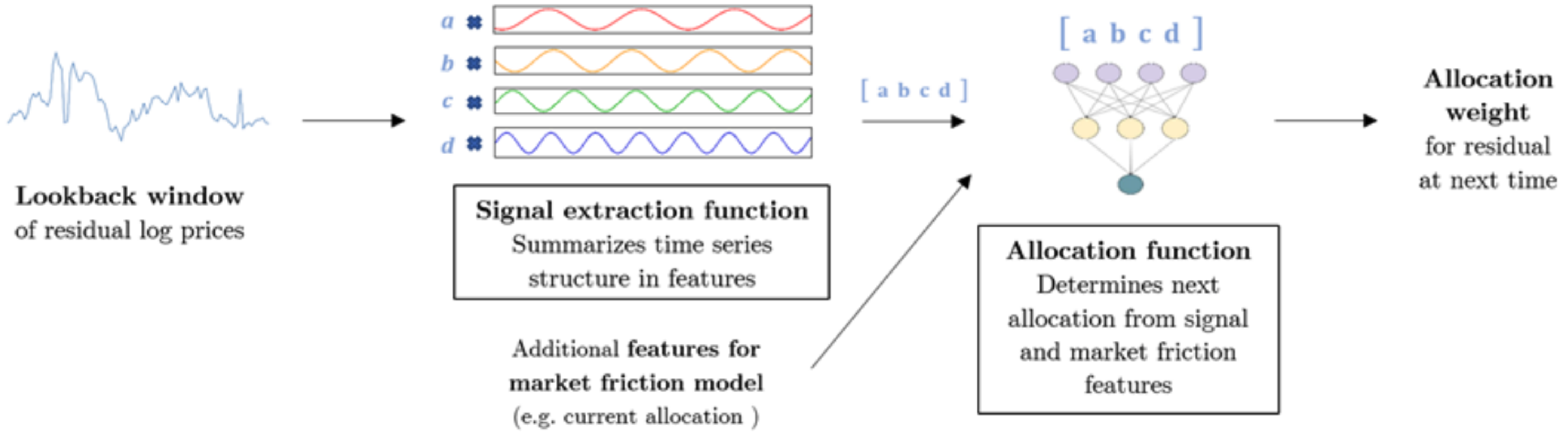


Figure 1: Conceptual Arbitrage Model



a
b
c
d
[a b c d]
[a b c d]
Allocation weight for residual at next time
Lookback window of residual log prices
Signal extraction function Summarizes time series structure in features
Allocation function Determines next allocation from signal and market friction features
Additional features for market friction model (e.g. current allocation)

This figure illustrates the conceptual structure of our statistical arbitrage framework. The model takes as input the last L cumulative returns of a residual portfolio on a lookback window at a given time and outputs the predicted optimal allocation weight for that residual for the next time. The model is composed of a signal extraction function and an allocation function.

Arbitrage Portfolios via residuals

Factor Model and Estimation of Factor Returns

- We observe **excess returns** for N assets at time t:

$$R_t \in \mathbb{R}^N$$

- Assume a K-factor model:

$$R_t = \beta_{t-1} F_t + \varepsilon_t$$

  where:

  - $\beta_{t-1} \in \mathbb{R}^{N \times K}$: factor loadings (betas), known at $t-1$
  - $F_t \in \mathbb{R}^K$: factor returns at t
  - $\varepsilon_t \in \mathbb{R}^N$: residual (idiosyncratic) returns
- Given β_{t-1} and observed R_t, we can estimate factor returns by **OLS**:

$$\hat{F}_t = \arg\min_f \|R_t - \beta_{t-1} f\|_2^2 = \left(\beta_{t-1}^\top \beta_{t-1}\right)^{-1} \beta_{t-1}^\top R_t$$

Residuals in Terms of a Projection Matrix

- Define the **projection matrix onto the factor space**:

$$P_{\beta,t-1} = \beta_{t-1}\left(\beta_{t-1}^\top \beta_{t-1}\right)^{-1} \beta_{t-1}^\top \;\in \mathbb{R}^{N \times N}$$

- Plug $\hat{F}_t$ into the model to get residuals:

$$\varepsilon_t = R_t - \beta_{t-1}\hat{F}_t = R_t - \beta_{t-1}\left(\beta_{t-1}^\top \beta_{t-1}\right)^{-1} \beta_{t-1}^\top R_t$$

- Factor out R_t:

$$\varepsilon_t = \left(I_N - \beta_{t-1}\left(\beta_{t-1}^\top \beta_{t-1}\right)^{-1} \beta_{t-1}^\top\right) R_t = \Phi_{t-1} R_t$$

- So we define:

$$\boxed{\Phi_{t-1} := I_N - P_{\beta,t-1} = I_N - \beta_{t-1}\left(\beta_{t-1}^\top \beta_{t-1}\right)^{-1} \beta_{t-1}^\top}$$

Factor Neutrality of Residuals

We want to show:

$$\beta_{t-1}^{\top}\varepsilon_t = 0.$$

Start from:

$$\varepsilon_t = \Phi_{t-1}R_t.$$

Then:

$$\beta_{t-1}^{\top}\varepsilon_t = \beta_{t-1}^{\top}\Phi_{t-1}R_t = \beta_{t-1}^{\top}\Big(I_N - \beta_{t-1}\big(\beta_{t-1}^{\top}\beta_{t-1}\big)^{-1}\beta_{t-1}^{\top}\Big)R_t.$$

Distribute $\beta_{t-1}^{\top}$:

$$\beta_{t-1}^{\top}\varepsilon_t = \underbrace{\beta_{t-1}^{\top}R_t}_{(\mathrm{A})} - \underbrace{\beta_{t-1}^{\top}\beta_{t-1}\big(\beta_{t-1}^{\top}\beta_{t-1}\big)^{-1}\beta_{t-1}^{\top}R_t}_{(\mathrm{B})}.$$

Now simplify term (B):

1. $\beta_{t-1}^{\top}\beta_{t-1} \in \mathbb{R}^{K\times K}$ is invertible by assumption.
2. Therefore:

$$\beta_{t-1}^{\top}\beta_{t-1}\big(\beta_{t-1}^{\top}\beta_{t-1}\big)^{-1} = I_K.$$

So:

$$(\mathrm{B}) = I_K\,\beta_{t-1}^{\top}R_t = \beta_{t-1}^{\top}R_t.$$

Plug back into the previous expression:

$$\beta_{t-1}^{\top}\varepsilon_t = \beta_{t-1}^{\top}R_t - \beta_{t-1}^{\top}R_t = 0.$$

Thus:

$$\boxed{\beta_{t-1}^{\top}\varepsilon_t = 0}$$

> **Conclusion:** the **residual portfolios are orthogonal to the factors, i.e. have zero factor exposures**.

Interpretation as Arbitrage Portfolios

- Take the standard basis vector $e_n \in \mathbb{R}^N$ (1 on asset n, 0 elsewhere).
- Define the **arbitrage-portfolio weights** associated with residual n:

$$w^{(n)} := \Phi_{t-1}^\top e_n \in \mathbb{R}^N$$

- Its portfolio return is:

$$w^{(n)\top} R_t = e_n^\top \Phi_{t-1} R_t = (\Phi_{t-1} R_t)_n = \varepsilon_{n,t}.$$

Conclusion:

- $\varepsilon_{n,t}$ is the **return of a self-financing, factor-neutral long–short portfolio** $w^{(n)}$.
- After applying Φ_{t-1}, **each "asset" in residual space is an arbitrage portfolio over the original** N **stocks**, by construction **orthogonal to all factors**.

Factor Models Used to construct arbitrage portfolios

Fama–French Style Factor Models

- Cross-sectional factor structure:

$$R_t = \beta_{t-1} F_t + \varepsilon_t$$

- **Traded factors** (different choices of K):
  - $K = 1$: Market excess return
  - $K = 3$: Market, SMB (size), HML (value)
  - $K = 5$: Market, SMB, HML, profitability, investment
  - $K = 8$: Above 5 factors **plus**
    - Momentum
    - Short-term reversal
    - Long-term reversal

Latent PCA Factor Model

- Compute the **sample covariance matrix** of daily stock returns using a 252-day rolling window.
- Apply **Principal Component Analysis (PCA)**:
  - Extract the first K principal components (PCs) as latent factors.
  - For each date, regress stock returns R_t on these PCs to obtain betas β_{t-1}.
- Residuals:

$$\varepsilon_t = R_t - \beta_{t-1} F_t$$

give a **PCA-based residual factor model**.

IPCA (Instrumented PCA) Factor Model

- Follow Kelly, Pruitt & Su (2019) – **Instrumented PCA**:
  - Monthly **latent factors** whose loadings are **linear functions of firm characteristics**.
- For each firm (i) at time (t):

$$\beta_{i,t} = \Gamma^\top z_{i,t}$$

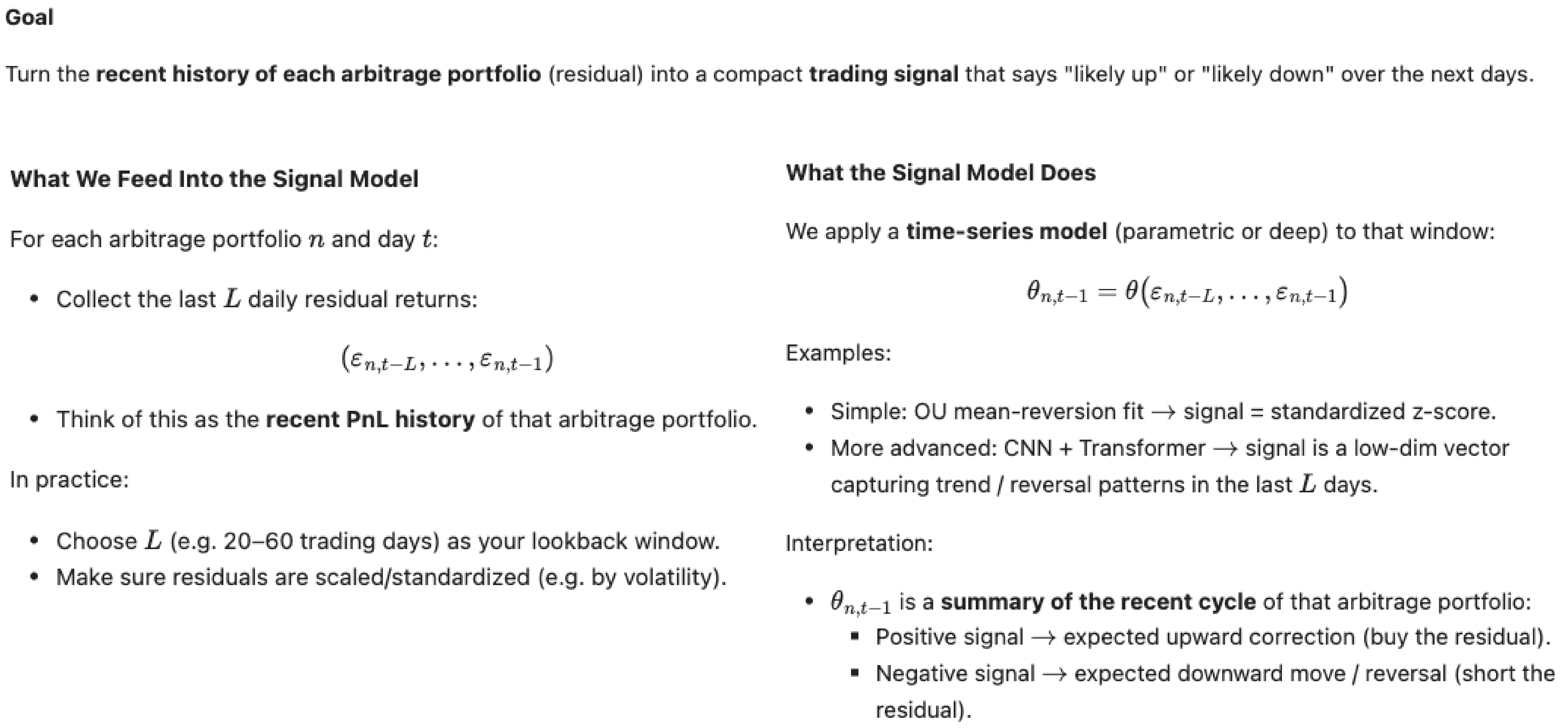
where:

  - $z_{i,t}$: vector of firm characteristics (instruments)
  - Γ: parameter matrix estimated by IPCA
- Characteristics (46 in total), broadly covering:
  - Value
  - Profitability
  - Investment
  - Intangibles
  - Trading frictions and liquidity, etc.
- **Estimation protocol**:
  - Use 20 years of **monthly** return and characteristic data to estimate Γ, β_t, and latent factors.
  - Re-estimate IPCA **once per year**.
- Daily application:
  - Use the most recent monthly β_{t-1} to model daily returns:

$$R_t = \beta_{t-1} F_t + \varepsilon_t$$

  - Residuals ε_t then define the **IPCA-based arbitrage portfolios**.

Arbitrage Signal (1)



Goal

Turn the **recent history of each arbitrage portfolio** (residual) into a compact **trading signal** that says "likely up" or "likely down" over the next days.

What We Feed Into the Signal Model

For each arbitrage portfolio n and day t:

- Collect the last L daily residual returns:

$$(\varepsilon_{n,t-L}, \ldots, \varepsilon_{n,t-1})$$

- Think of this as the **recent PnL history** of that arbitrage portfolio.

In practice:

- Choose L (e.g. 20–60 trading days) as your lookback window.
- Make sure residuals are scaled/standardized (e.g. by volatility).

What the Signal Model Does

We apply a **time-series model** (parametric or deep) to that window:

$$\theta_{n,t-1} = \theta\big(\varepsilon_{n,t-L}, \ldots, \varepsilon_{n,t-1}\big)$$

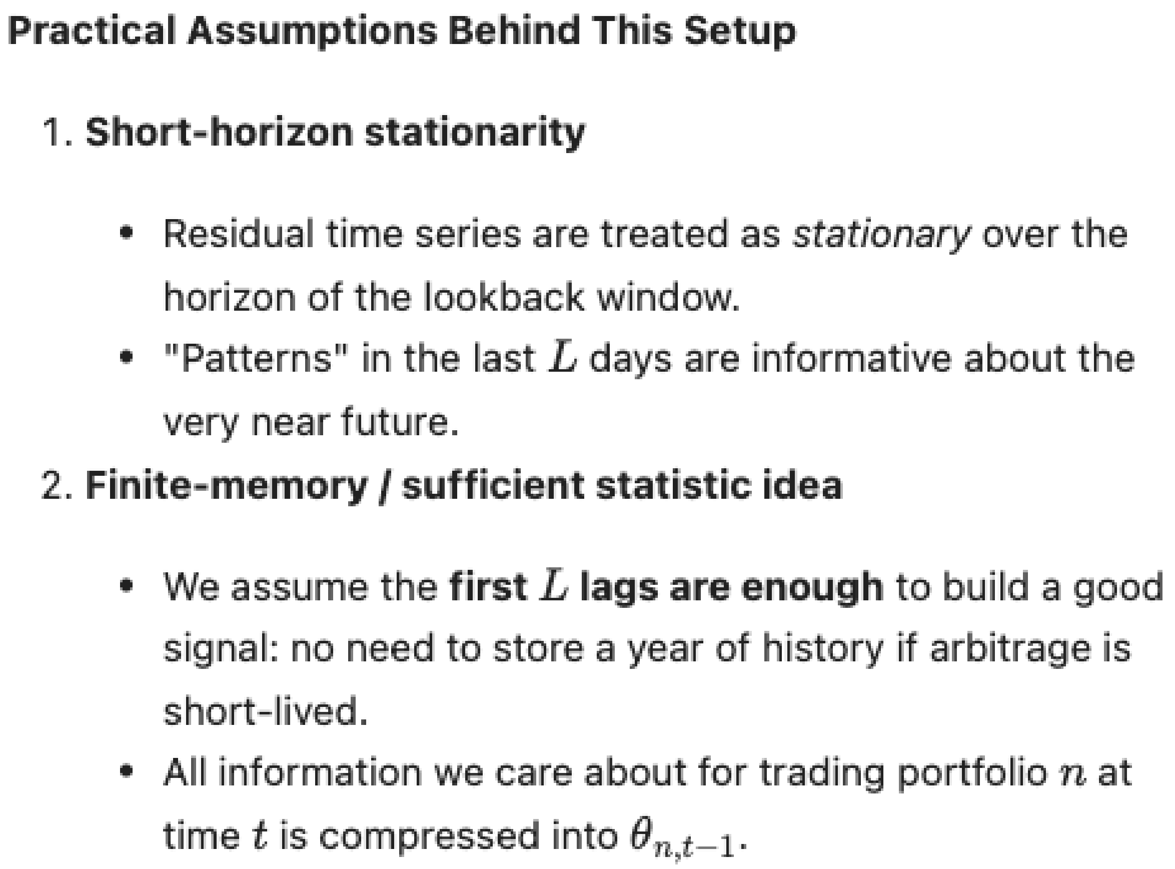
Examples:

- Simple: OU mean-reversion fit $\rightarrow$ signal = standardized z-score.
- More advanced: CNN + Transformer $\rightarrow$ signal is a low-dim vector capturing trend / reversal patterns in the last L days.

Interpretation:

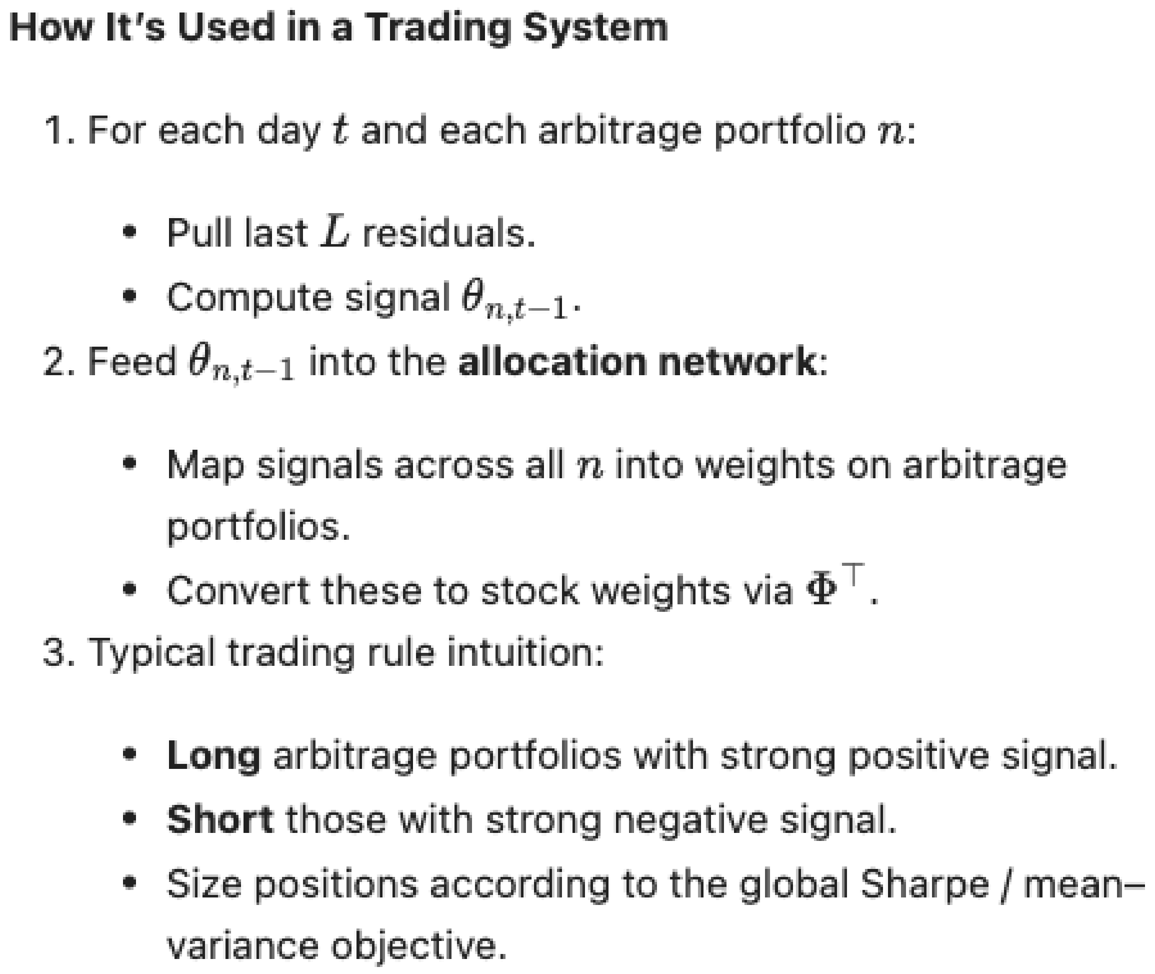
- $\theta_{n,t-1}$ is a **summary of the recent cycle** of that arbitrage portfolio:
  - Positive signal $\rightarrow$ expected upward correction (buy the residual).
  - Negative signal $\rightarrow$ expected downward move / reversal (short the residual).

Arbitrage Signal (2) - Practical Assumptions



Practical Assumptions Behind This Setup

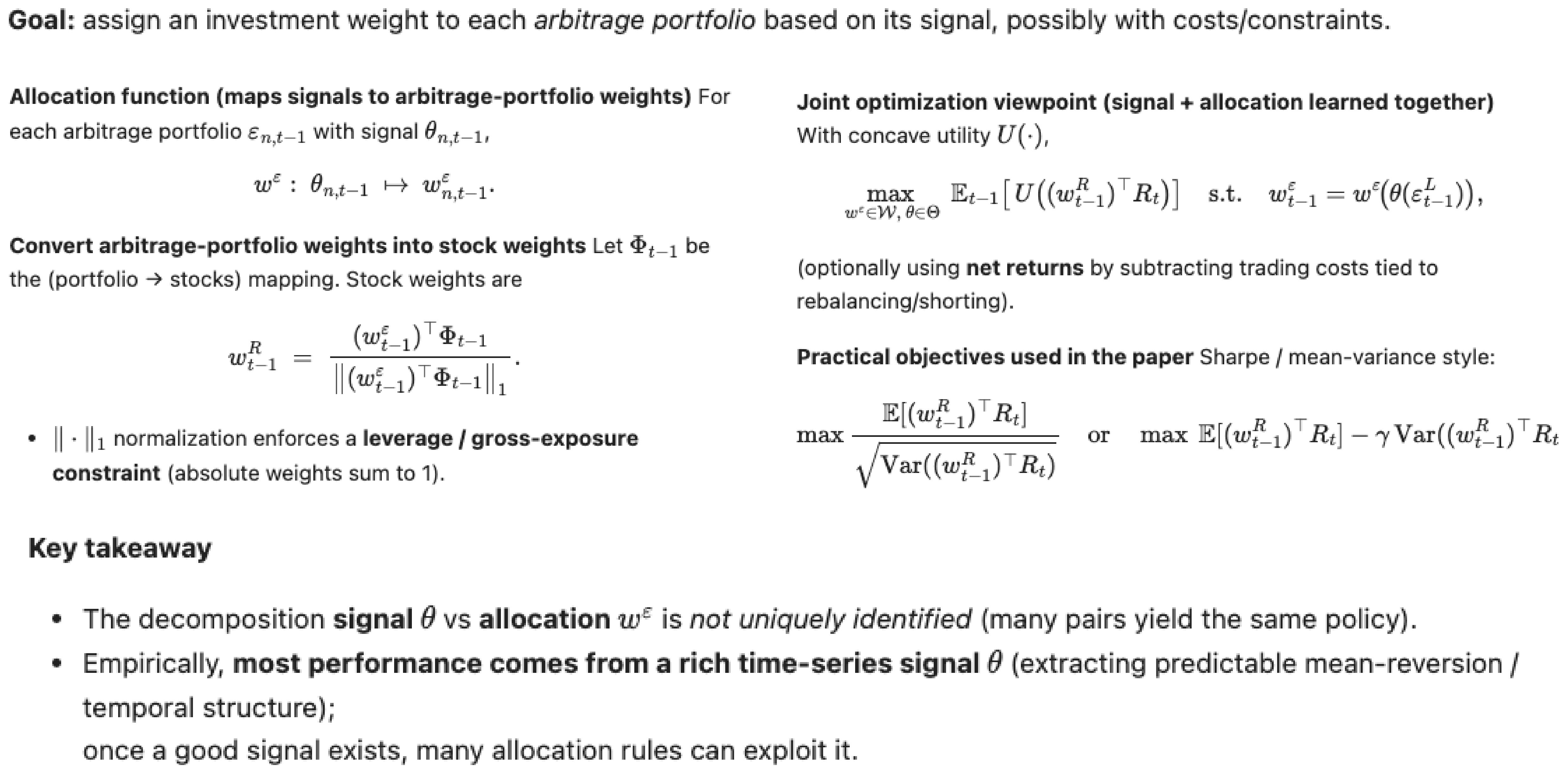
1. **Short-horizon stationarity**
   - Residual time series are treated as *stationary* over the horizon of the lookback window.
   - "Patterns" in the last L days are informative about the very near future.
2. **Finite-memory / sufficient statistic idea**
   - We assume the **first L lags are enough** to build a good signal: no need to store a year of history if arbitrage is short-lived.
   - All information we care about for trading portfolio n at time t is compressed into $\theta_{n,t-1}$.



How It's Used in a Trading System

1. For each day t and each arbitrage portfolio n:
   - Pull last L residuals.
   - Compute signal $\theta_{n,t-1}$.
2. Feed $\theta_{n,t-1}$ into the **allocation network**:
   - Map signals across all n into weights on arbitrage portfolios.
   - Convert these to stock weights via $\Phi^\top$.
3. Typical trading rule intuition:
   - **Long** arbitrage portfolios with strong positive signal.
   - **Short** those with strong negative signal.
   - Size positions according to the global Sharpe / mean–variance objective.

Arbitrage trading= signal + allocation + tradable weights



Goal: assign an investment weight to each *arbitrage portfolio* based on its signal, possibly with costs/constraints.

Allocation function (maps signals to arbitrage-portfolio weights) For each arbitrage portfolio $\varepsilon_{n,t-1}$ with signal $\theta_{n,t-1}$,

$$w^{\varepsilon} : \theta_{n,t-1} \mapsto w^{\varepsilon}_{n,t-1}.$$

Convert arbitrage-portfolio weights into stock weights Let Φ_{t-1} be the (portfolio → stocks) mapping. Stock weights are

$$w^{R}_{t-1} = \frac{(w^{\varepsilon}_{t-1})^{\top}\Phi_{t-1}}{\left\|(w^{\varepsilon}_{t-1})^{\top}\Phi_{t-1}\right\|_1}.$$

- $\|\cdot\|_1$ normalization enforces a **leverage / gross-exposure constraint** (absolute weights sum to 1).

Joint optimization viewpoint (signal + allocation learned together) With concave utility $U(\cdot)$,

$$\max_{w^{\varepsilon}\in\mathcal{W},\,\theta\in\Theta} \mathbb{E}_{t-1}\left[U\left((w^{R}_{t-1})^{\top}R_t\right)\right] \quad \text{s.t.} \quad w^{\varepsilon}_{t-1} = w^{\varepsilon}\left(\theta(\varepsilon^{L}_{t-1})\right),$$

(optionally using **net returns** by subtracting trading costs tied to rebalancing/shorting).

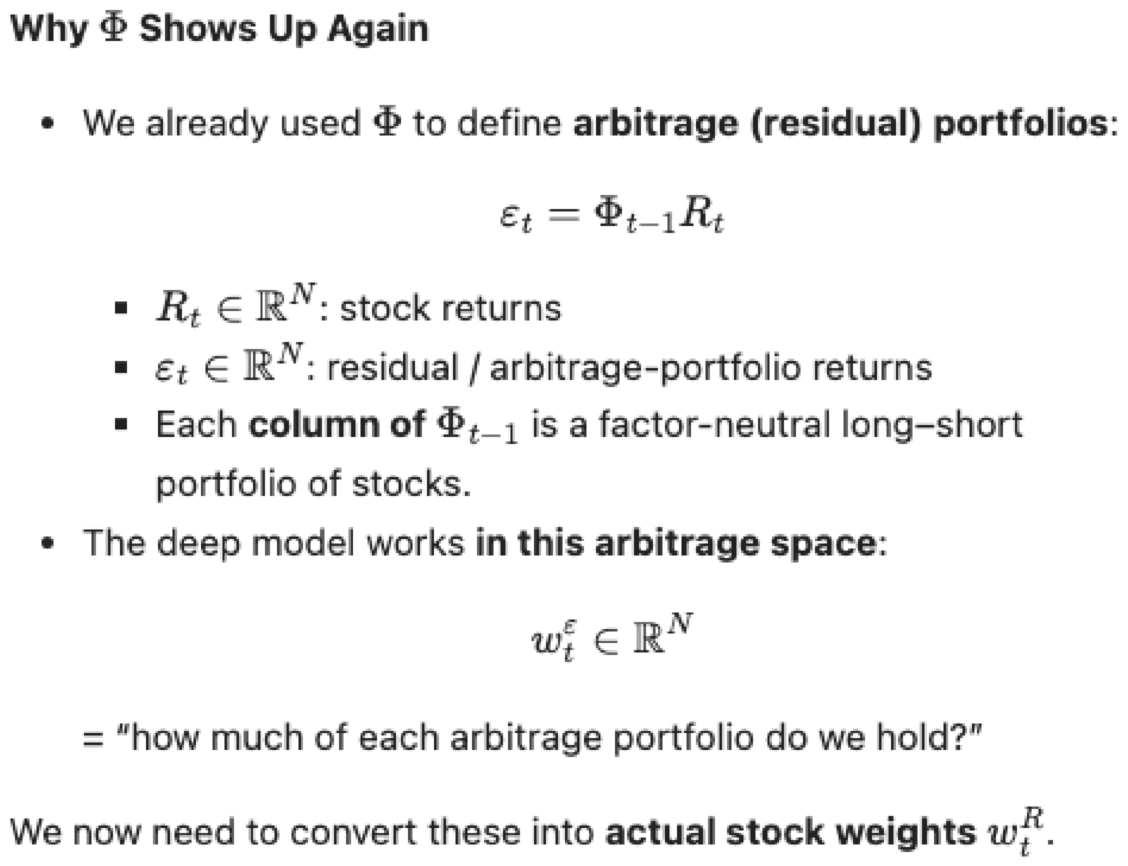
Practical objectives used in the paper Sharpe / mean-variance style:

$$\max \frac{\mathbb{E}[(w^{R}_{t-1})^{\top}R_t]}{\sqrt{\mathrm{Var}((w^{R}_{t-1})^{\top}R_t)}} \quad \text{or} \quad \max \mathbb{E}[(w^{R}_{t-1})^{\top}R_t] - \gamma\,\mathrm{Var}((w^{R}_{t-1})^{\top}R_t$$

Key takeaway

- The decomposition **signal** θ vs **allocation** w^{ε} is *not uniquely identified* (many pairs yield the same policy).
- Empirically, **most performance comes from a rich time-series signal** θ (extracting predictable mean-reversion / temporal structure);
  once a good signal exists, many allocation rules can exploit it.

From Arbitrage-Portfolio Weights to Stock Weights (1)



Why Φ Shows Up Again

- We already used Φ to define **arbitrage (residual) portfolios**:

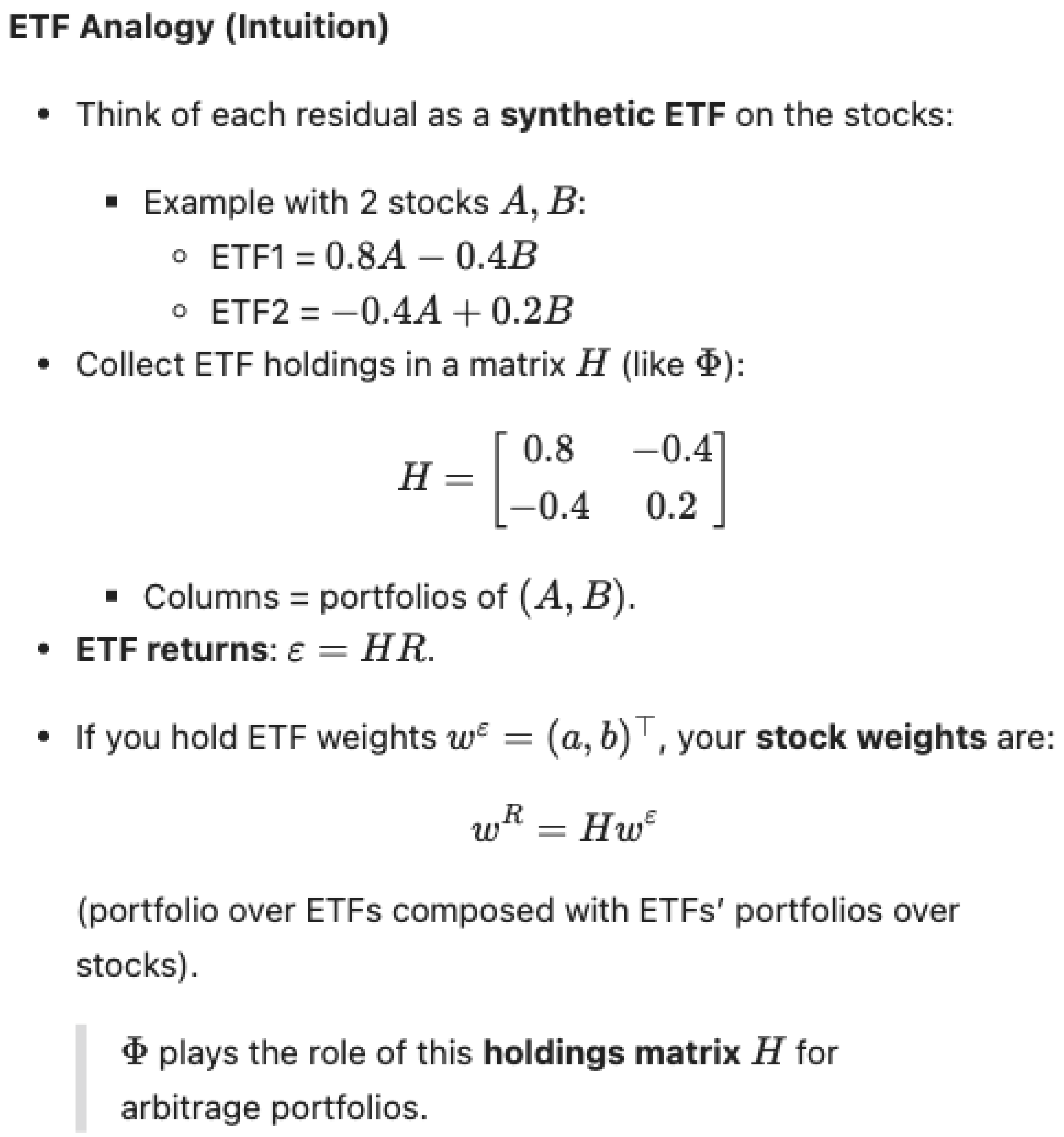
$$\varepsilon_t = \Phi_{t-1} R_t$$

  - $R_t \in \mathbb{R}^N$: stock returns
  - $\varepsilon_t \in \mathbb{R}^N$: residual / arbitrage-portfolio returns
  - Each **column of Φ_{t-1}** is a factor-neutral long–short portfolio of stocks.
- The deep model works **in this arbitrage space**:

$$w_t^\varepsilon \in \mathbb{R}^N$$

  = "how much of each arbitrage portfolio do we hold?"

We now need to convert these into **actual stock weights w_t^R**.



ETF Analogy (Intuition)

- Think of each residual as a **synthetic ETF** on the stocks:
  - Example with 2 stocks A, B:
    - ETF1 = $0.8A - 0.4B$
    - ETF2 = $-0.4A + 0.2B$
- Collect ETF holdings in a matrix H (like Φ):

$$H = \begin{bmatrix} 0.8 & -0.4 \\ -0.4 & 0.2 \end{bmatrix}$$

  - Columns = portfolios of (A, B).
- **ETF returns**: $\varepsilon = HR$.
- If you hold ETF weights $w^\varepsilon = (a, b)^\top$, your **stock weights** are:

$$w^R = Hw^\varepsilon$$

  (portfolio over ETFs composed with ETFs' portfolios over stocks).

> Φ plays the role of this **holdings matrix H** for arbitrage portfolios.

From Arbitrage-Portfolio Weights to Stock Weights (2)

- Residual (arbitrage) returns:

$$\varepsilon_t = \Phi_{t-1} R_t$$

- Portfolio PnL if we trade arbitrage portfolios:

$$\mathrm{PnL}_t = {w^{\varepsilon}_{t-1}}^{\top} \varepsilon_t = {w^{\varepsilon}_{t-1}}^{\top} \Phi_{t-1} R_t$$

- Use the identity $u^{\top} A v = (A^{\top} u)^{\top} v$:

$${w^{\varepsilon}_{t-1}}^{\top} \Phi_{t-1} R_t = (\Phi_{t-1}^{\top} w^{\varepsilon}_{t-1})^{\top} R_t$$

- **Define the implied stock weights**:

$$\boxed{w^{R}_{t-1} := \Phi_{t-1}^{\top} w^{\varepsilon}_{t-1}}$$

- Then:

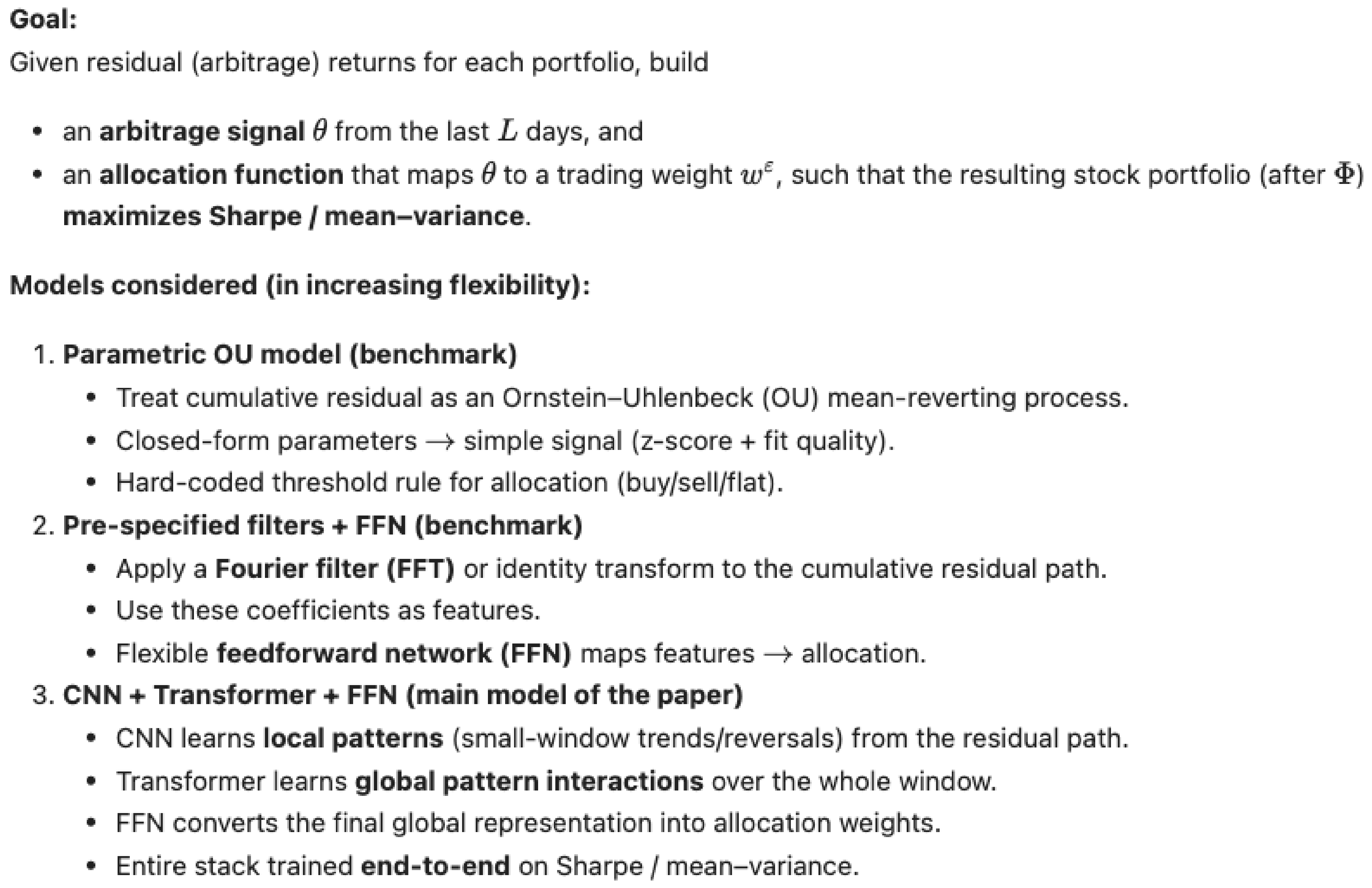
$$\mathrm{PnL}_t = {w^{R}_{t-1}}^{\top} R_t$$

So:

- Φ acts on **returns** to create residuals.
- Φ^T acts on **weights** to map "weights on arbitrage portfolios" to "weights on stocks".
- No inverse, no double-counting — just portfolio composition:

> **portfolio over arbitrage portfolios** $\rightarrow$ **portfolio over stocks** via Φ^T.

Goal & Menu of Signal/Allocation Models



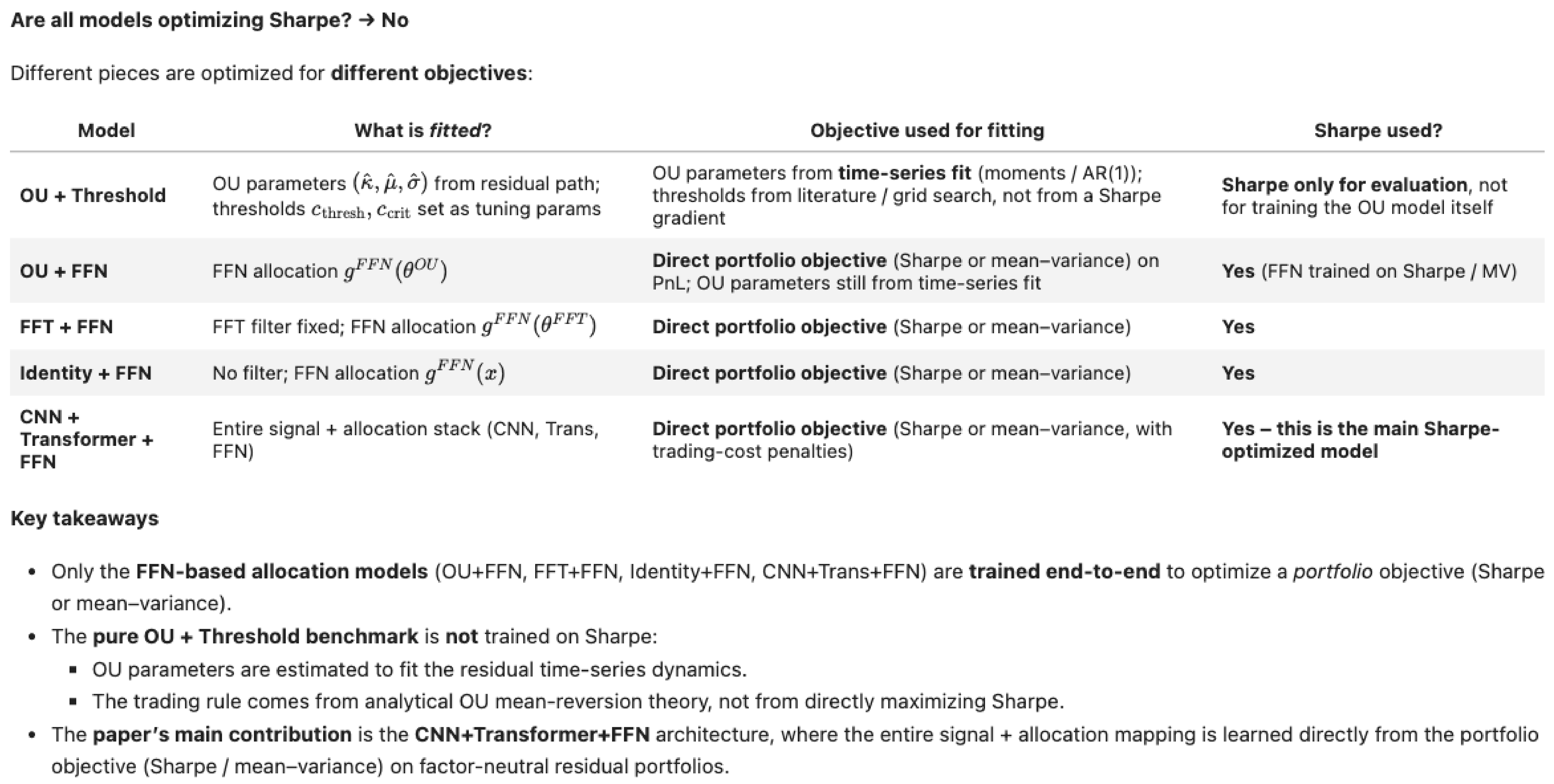
Goal:
Given residual (arbitrage) returns for each portfolio, build

- an **arbitrage signal** θ from the last L days, and
- an **allocation function** that maps θ to a trading weight w^{ϵ}, such that the resulting stock portfolio (after Φ) **maximizes Sharpe / mean–variance**.

Models considered (in increasing flexibility):

1. **Parametric OU model (benchmark)**
   - Treat cumulative residual as an Ornstein–Uhlenbeck (OU) mean-reverting process.
   - Closed-form parameters $\rightarrow$ simple signal (z-score + fit quality).
   - Hard-coded threshold rule for allocation (buy/sell/flat).
2. **Pre-specified filters + FFN (benchmark)**
   - Apply a **Fourier filter (FFT)** or identity transform to the cumulative residual path.
   - Use these coefficients as features.
   - Flexible **feedforward network (FFN)** maps features $\rightarrow$ allocation.
3. **CNN + Transformer + FFN (main model of the paper)**
   - CNN learns **local patterns** (small-window trends/reversals) from the residual path.
   - Transformer learns **global pattern interactions** over the whole window.
   - FFN converts the final global representation into allocation weights.
   - Entire stack trained **end-to-end** on Sharpe / mean–variance.

Comparison of Signal/Allocation Models by Objective Function



Are all models optimizing Sharpe? → No

Different pieces are optimized for **different objectives**:

Model	What is *fitted*?	Objective used for fitting	Sharpe used?
OU + Threshold	OU parameters $(\hat{\kappa}, \hat{\mu}, \hat{\sigma})$ from residual path; thresholds $c_{\text{thresh}}, c_{\text{crit}}$ set as tuning params	OU parameters from **time-series fit** (moments / AR(1)); thresholds from literature / grid search, not from a Sharpe gradient	**Sharpe only for evaluation**, not for training the OU model itself
OU + FFN	FFN allocation $g^{FFN}(\theta^{OU})$	**Direct portfolio objective** (Sharpe or mean–variance) on PnL; OU parameters still from time-series fit	**Yes** (FFN trained on Sharpe / MV)
FFT + FFN	FFT filter fixed; FFN allocation $g^{FFN}(\theta^{FFT})$	**Direct portfolio objective** (Sharpe or mean–variance)	**Yes**
Identity + FFN	No filter; FFN allocation $g^{FFN}(x)$	**Direct portfolio objective** (Sharpe or mean–variance)	**Yes**
CNN + Transformer + FFN	Entire signal + allocation stack (CNN, Trans, FFN)	**Direct portfolio objective** (Sharpe or mean–variance, with trading-cost penalties)	**Yes – this is the main Sharpe-optimized model**

Key takeaways

- Only the **FFN-based allocation models** (OU+FFN, FFT+FFN, Identity+FFN, CNN+Trans+FFN) are **trained end-to-end** to optimize a *portfolio* objective (Sharpe or mean–variance).
- The **pure OU + Threshold benchmark** is **not** trained on Sharpe:
  - OU parameters are estimated to fit the residual time-series dynamics.
  - The trading rule comes from analytical OU mean-reversion theory, not from directly maximizing Sharpe.
- The **paper's main contribution** is the **CNN+Transformer+FFN** architecture, where the entire signal + allocation mapping is learned directly from the portfolio objective (Sharpe / mean–variance) on factor-neutral residual portfolios.

Parametric OU Signal & Threshold Allocation (Benchmark)

Idea: classical mean-reversion stat-arb (Avellaneda & Lee, Yeo & Papanicolaou).

1. **Input:** last L cumulative residuals for a given arbitrage portfolio:

$$x = (X_1, \ldots, X_L), \quad X_l = \sum_{j=1}^{l} \varepsilon_{t-L+j}.$$

2. **Model:** assume X_t follows an OU process

$$dX_t = \kappa(\mu - X_t)\,dt + \sigma\,dB_t.$$

   - Estimate $\hat{\kappa}, \hat{\mu}, \hat{\sigma}$ from discrete data (AR(1) fit).
   - Compute goodness-of-fit R^2.

3. **Signal (per residual):**

$$\theta^{OU} = (\hat{\kappa}, \hat{\mu}, \hat{\sigma}, X_L, R^2).$$

   - $\hat{\kappa}$: speed of mean reversion
   - $\hat{\mu}$: long-run "fair value"
   - X_L: current residual "price"
   - R^2: how reliable the OU fit is

4. **Allocation rule (OU + Threshold):**

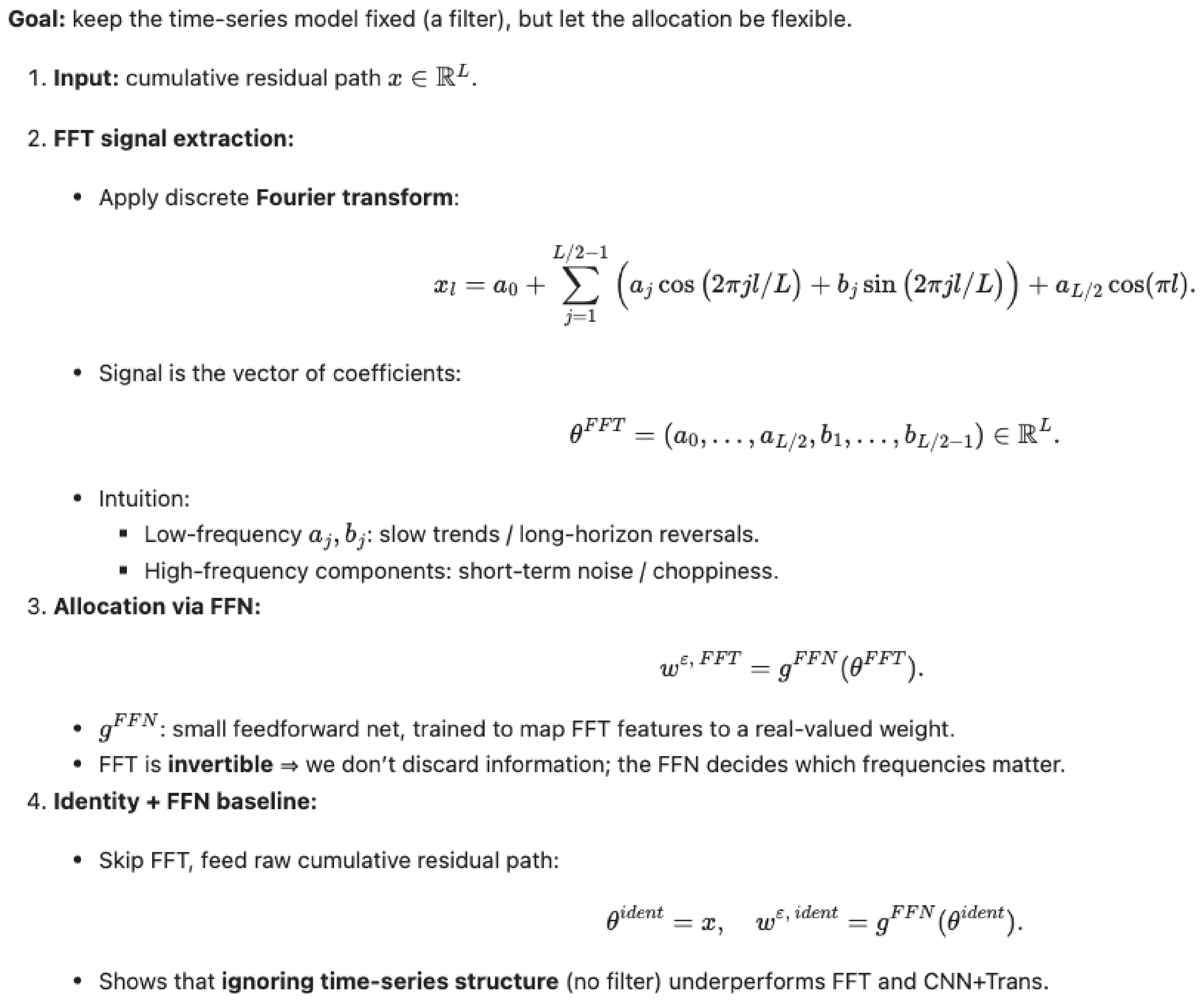
   - Compute standardized distance to mean:

$$z = \frac{X_L - \hat{\mu}}{\hat{\sigma}/\sqrt{2\hat{\kappa}}}.$$

   - If $z > c_{\text{thresh}}$ and $R^2 > c_{\text{crit}}$: **short** residual ($w^{\varepsilon} = -1$).
   - If $z < -c_{\text{thresh}}$ and $R^2 > c_{\text{crit}}$: **long** residual ($w^{\varepsilon} = +1$).
   - Else: **flat** ($w^{\varepsilon} = 0$).

OU + FFN variant:
Same OU signal θ^{OU}, but replace the hard threshold rule with a **small FFN** that learns a smoother mapping $\theta^{OU} \mapsto w^{\varepsilon}$.

Pre-specified Filters (FFT) + FFN Allocation (Benchmark)



Goal: keep the time-series model fixed (a filter), but let the allocation be flexible.

1. **Input:** cumulative residual path $x \in \mathbb{R}^L$.
2. **FFT signal extraction:**
   - Apply discrete **Fourier transform**:

   $$x_l = a_0 + \sum_{j=1}^{L/2-1} \Big(a_j \cos\left(2\pi j l / L\right) + b_j \sin\left(2\pi j l / L\right) \Big) + a_{L/2} \cos(\pi l).$$

   - Signal is the vector of coefficients:

   $$\theta^{FFT} = (a_0, \ldots, a_{L/2}, b_1, \ldots, b_{L/2-1}) \in \mathbb{R}^L.$$

   - Intuition:
     - Low-frequency a_j, b_j: slow trends / long-horizon reversals.
     - High-frequency components: short-term noise / choppiness.
3. **Allocation via FFN:**

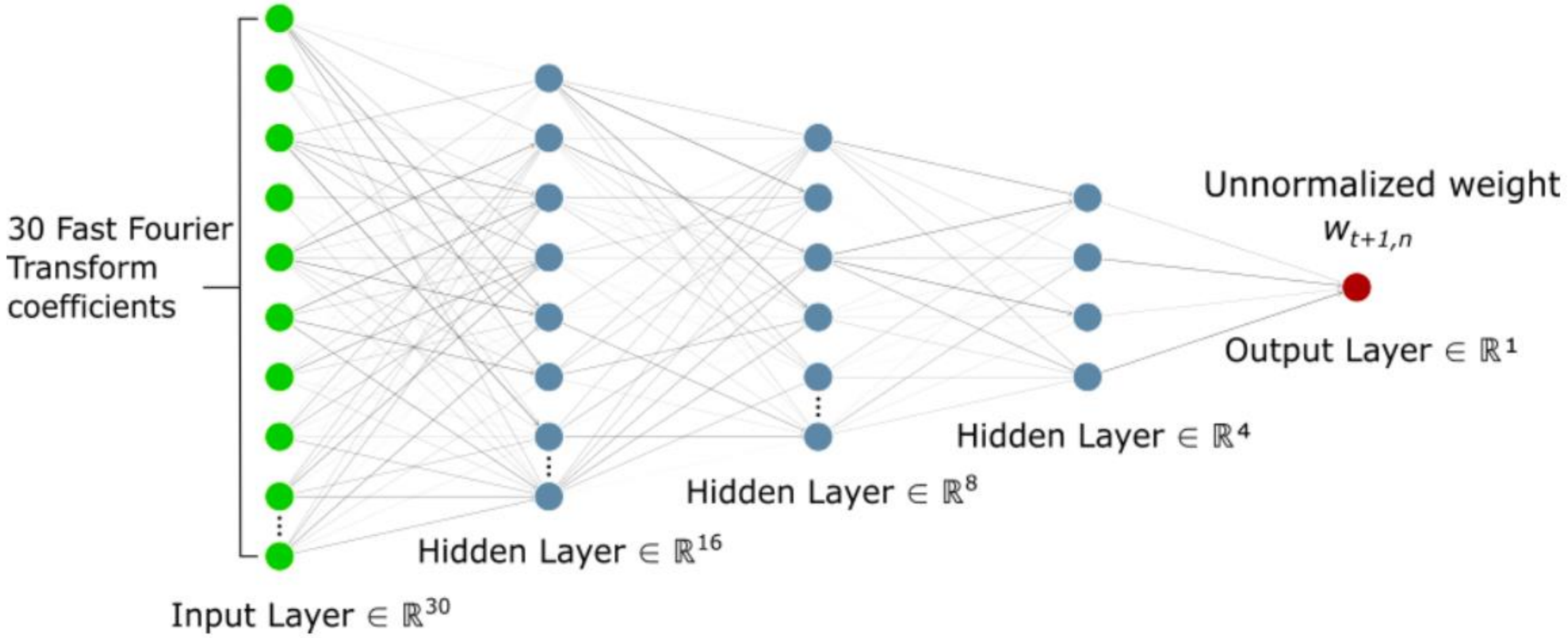
   $$w^{\varepsilon,\, FFT} = g^{FFN}(\theta^{FFT}).$$

   - g^{FFN}: small feedforward net, trained to map FFT features to a real-valued weight.
   - FFT is **invertible** $\Rightarrow$ we don't discard information; the FFN decides which frequencies matter.
4. **Identity + FFN baseline:**
   - Skip FFT, feed raw cumulative residual path:

   $$\theta^{ident} = x, \quad w^{\varepsilon,\, ident} = g^{FFN}(\theta^{ident}).$$

   - Shows that **ignoring time-series structure** (no filter) underperforms FFT and CNN+Trans.

Figure A.1. Feedforward Network Architecture



CNN + Transformer: High-Level Intuition (Main Model)

This is the **main contribution** of the paper: a data-driven time-series model for the arbitrage signal.

Key idea:
Break the problem into **local patterns** and **global dependencies**:

1. **CNN (local filters)**
    - Slides small filters over the cumulative residual path.
    - Learns patterns like "local up move", "local down move", "U-shape", "mini-reversal".
    - Output: a matrix $\tilde{x} \in \mathbb{R}^{L \times D}$:
        - rows = time steps,
        - columns = activations of D learned local patterns.
2. **Transformer (global pattern projection)**
    - Looks at the whole $\tilde{x}$ and computes attention-based combinations.
    - Each attention head ≈ one **global pattern** (e.g., slow uptrend, fast crash + rebound).
    - Output: a low-dimensional representation h_L^{proj} summarizing the last L days.
3. **FFN allocation**
    - Final signal:

$$\theta_{t-1}^{\text{CNN+Trans}} = h_L^{\text{proj}}$$

    - Allocation:

$$w_t^{\varepsilon} = g^{FFN}\left(\theta_{t-1}^{\text{CNN+Trans}}\right).$$

The whole stack (CNN + Transformer + FFN) is trained **end-to-end** to maximize portfolio Sharpe / mean–variance.

CNN: Learning Local Residual Patterns

Input: cumulative residual path for one arbitrage portfolio:

$$x = (X_1, \dots, X_L) \in \mathbb{R}^L.$$

2-layer 1D CNN:

1. **First convolutional layer**
   - Filter size $D_{\text{size}} = 2$, D filters.
   - For each filter d and position ℓ:

$$y^{(0)}_{\ell,d} = \sum_{m=1}^{D_{\text{size}}} W^{(0)}_{d,m}\, x_{\ell-m+1}, \quad x^{(1)}_{\ell,d} = \mathrm{ReLU}\big(y^{(0)}_{\ell,d}\big).$$

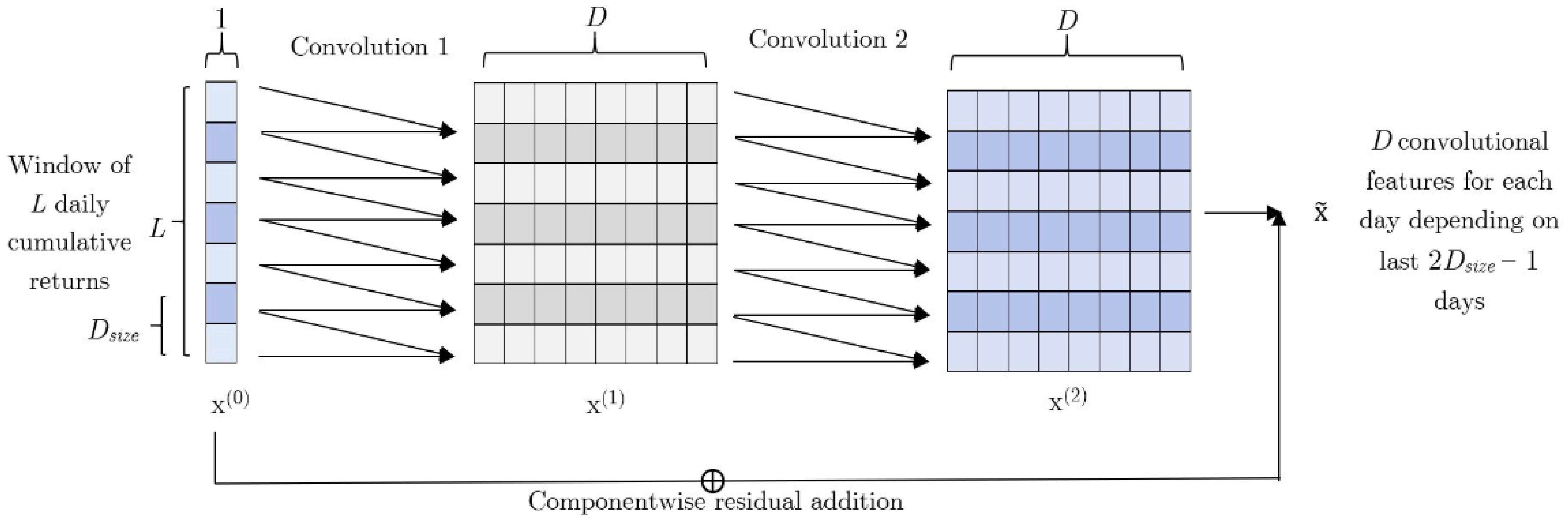
   - Captures **very local** slopes and tiny reversals.
2. **Second convolutional layer**
   - Same idea, applied to $x^{(1)}$:

$$y^{(1)}_{\ell,d} = \sum_{m=1}^{D_{\text{size}}} \sum_{j=1}^{D} W^{(1)}_{d,j,m}\, x^{(1)}_{\ell-m+1,j}, \quad x^{(2)}_{\ell,d} = \mathrm{ReLU}\big(y^{(1)}_{\ell,d}\big).$$

3. **CNN output**
   - Denote $\tilde{x} = x^{(2)} \in \mathbb{R}^{L \times D}$.
   - Column d of $\tilde{x} \approx$ **activation of local pattern** d at each time step.

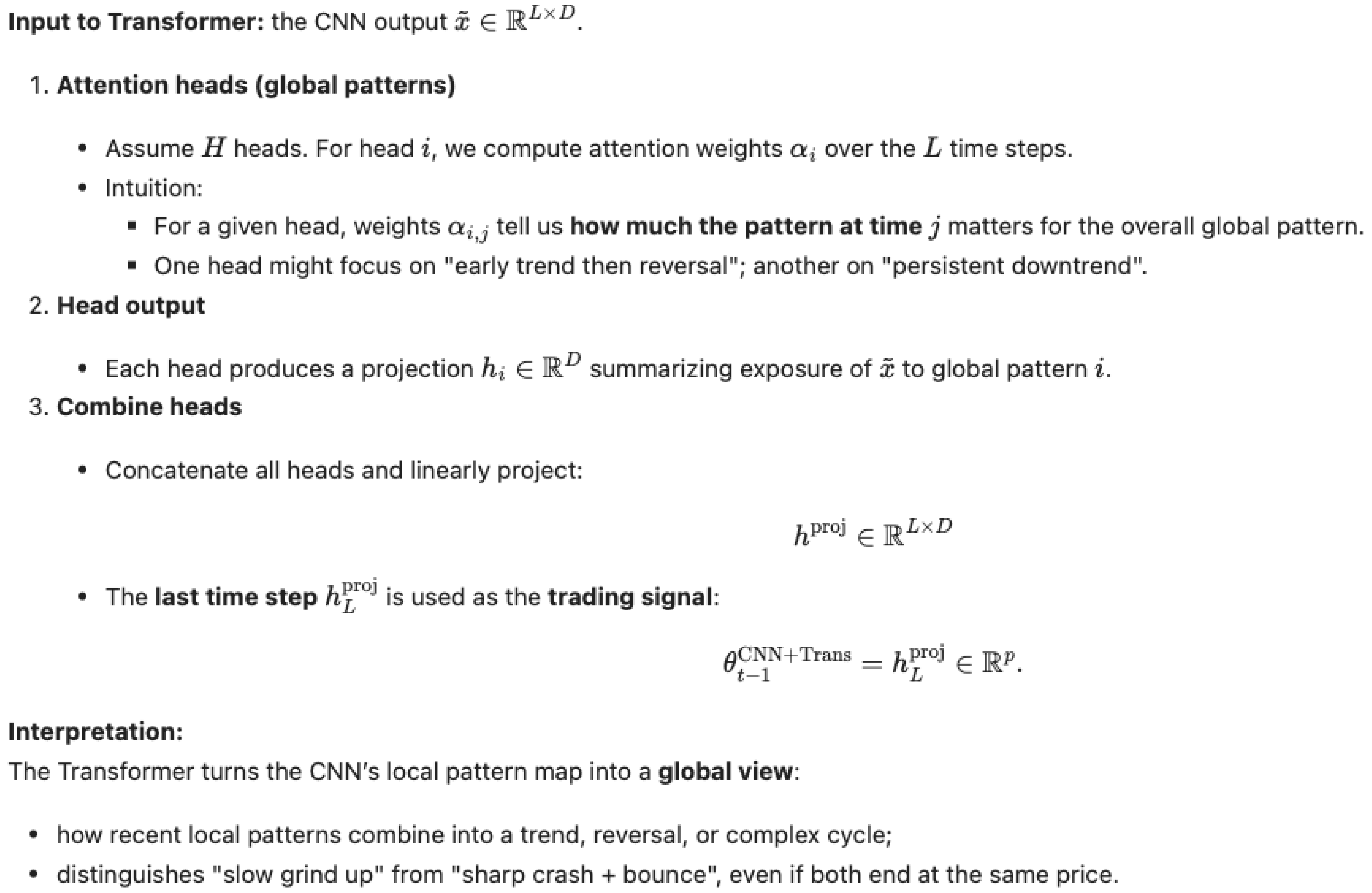
Interpretation:
The CNN converts the raw price-like residual path into a **"local pattern map"** (trend / U-shape / wiggle patterns), which is a much more informative representation for trading than raw returns.

Figure 3: Convolutional Network Architecture


1
Convolution 1
D
Convolution 2
D
Window of L daily cumulative returns
L
D_{size}
$\mathrm{x}^{(0)}$
$\mathrm{x}^{(1)}$
$\mathrm{x}^{(2)}$
$\tilde{\mathrm{x}}$
D convolutional features for each day depending on last $2D_{size} - 1$ days
$\oplus$
Componentwise residual addition

This figure shows the structure of our convolutional network. The network takes as input a window of L consecutive daily cumulative returns a residual, and outputs D features for each block of D_{size} days. Each of the features is a nonlinear function of the observations in the block, and captures a common pattern.

Transformer: Global Dependencies Between Local Patterns



Input to Transformer: the CNN output $\tilde{x} \in \mathbb{R}^{L \times D}$.

1. **Attention heads (global patterns)**
   - Assume H heads. For head i, we compute attention weights α_i over the L time steps.
   - Intuition:
     - For a given head, weights $\alpha_{i,j}$ tell us **how much the pattern at time j** matters for the overall global pattern.
     - One head might focus on "early trend then reversal"; another on "persistent downtrend".
2. **Head output**
   - Each head produces a projection $h_i \in \mathbb{R}^D$ summarizing exposure of $\tilde{x}$ to global pattern i.
3. **Combine heads**
   - Concatenate all heads and linearly project:

$$h^{\text{proj}} \in \mathbb{R}^{L \times D}$$

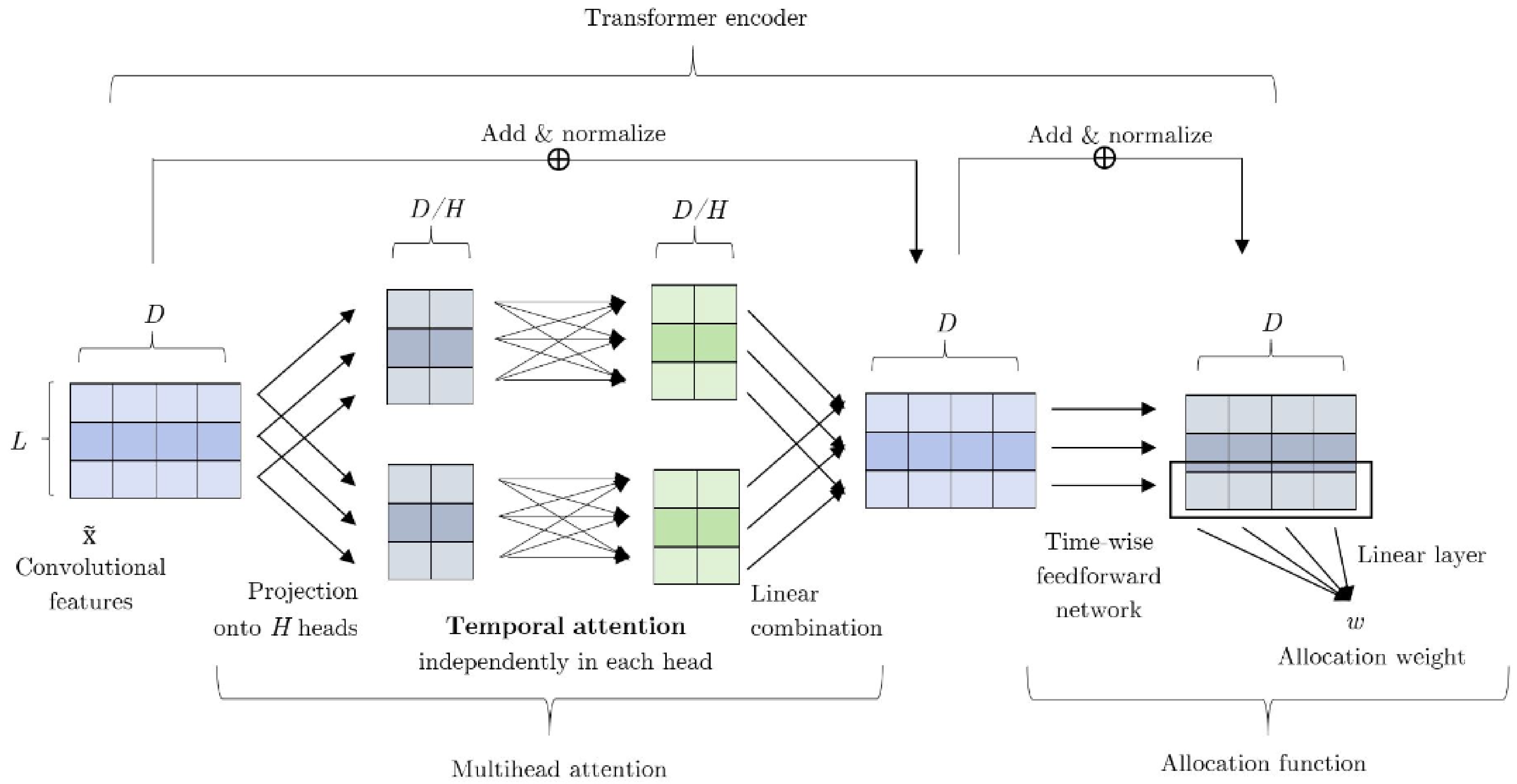
   - The **last time step** h_L^{proj} is used as the **trading signal**:

$$\theta_{t-1}^{\text{CNN+Trans}} = h_L^{\text{proj}} \in \mathbb{R}^p.$$

Interpretation:
The Transformer turns the CNN's local pattern map into a **global view**:

- how recent local patterns combine into a trend, reversal, or complex cycle;
- distinguishes "slow grind up" from "sharp crash + bounce", even if both end at the same price.

Figure 4: Transformer Network Architecture



This figure shows the structure of our transformer network. The model takes as input the matrix $\tilde{x} \in \mathbb{R}^{L\times D}$ that we obtain as output of the convolutional network depicted in Figure 3, which contains D features for each of the L blocks of the original time series. These features are projected onto H attention heads, which capture the global temporal dependency patterns. The projections on these attention heads represent our arbitrage signals, which are the input to the feedforward network that estimates the allocation weight for the residual on the next day.

From Signal to Portfolio: Allocation & Training Objective

1. **Allocation in arbitrage space**

$$w_t^{\varepsilon} = g^{FFN}\big(\theta_{t-1}^{\text{CNN+Trans}}\big).$$

2. **Map to stock space via Φ**

$$\tilde{w}_t^R = \Phi_t^{\top} w_t^{\varepsilon}, \qquad w_t^R = \frac{\tilde{w}_t^R}{\|\tilde{w}_t^R\|_1}.$$

   - Ensures the final portfolio is **factor-neutral** and has fixed L1 leverage.

3. **Training objective (over all days)**
   - Maximize **Sharpe**:

$$\text{Sharpe} = \frac{\mathbb{E}\big[w_t^{R\top} R_{t+1}\big]}{\sqrt{\text{Var}\big(w_t^{R\top} R_{t+1}\big)}}$$

   - Or **mean–variance**:

$$\mathbb{E}\big[w_t^{R\top} R_{t+1}\big] - \gamma\,\text{Var}\big(w_t^{R\top} R_{t+1}\big) - \text{trading costs.}$$

4. **End-to-end learning**
   - Gradients flow from the portfolio objective back through:
     - FFN allocation,
     - Transformer,
     - CNN local filters.
   - No separate prediction model; the network directly learns **"history $\rightarrow$ weights"** that optimize portfolio performance.

> **Bottom line:**
> OU and FFT+FFN are interpretable benchmarks.
> **CNN+Transformer+FFN** is the fully data-driven engine that learns both the time-series representation and the trading rule jointly, targeted at Sharpe/mean–variance.

Empirical Analysis

- Data Setup
- Factor Model Estimation
- Implementation
- Main Results

Data Setup

Universe & Sample

- **Source:** CRSP daily equity returns (US stocks), **Jan 1978 – Dec 2016**.
- **Use of sample:**
  - Pre-1998: estimate factor models (FF, PCA, IPCA).
  - **Jan 1998 – Dec 2016:** main **daily trading period** for arbitrage strategies.
- **Risk-free rate:** 1-month Treasury bill from the Kenneth French Data Library.
- **Universe selection (liquidity filter):**
  - Each month, keep only stocks whose **market cap >** 0.01% of total US market cap.
  - Leaves ~**550 largest stocks** on average (roughly S&P 500–like universe).
  - Panel is **unbalanced**: the exact set of stocks can change month to month.

Firm Characteristics

- **46 stock-specific characteristics** from Chen et al. (2022):
  - Built from CRSP/Compustat accounting data and past returns.
  - Cover value, profitability, investment, intangibles, trading frictions, etc.
- For each stock and month:
  - Characteristics are **cross-sectionally standardized and rank-transformed** (robust to outliers and different measurement scales).
  - These transformed characteristics at month $t-1$ are used to build/estimate the factor models (e.g., IPCA).

Daily Time-Series for Trading

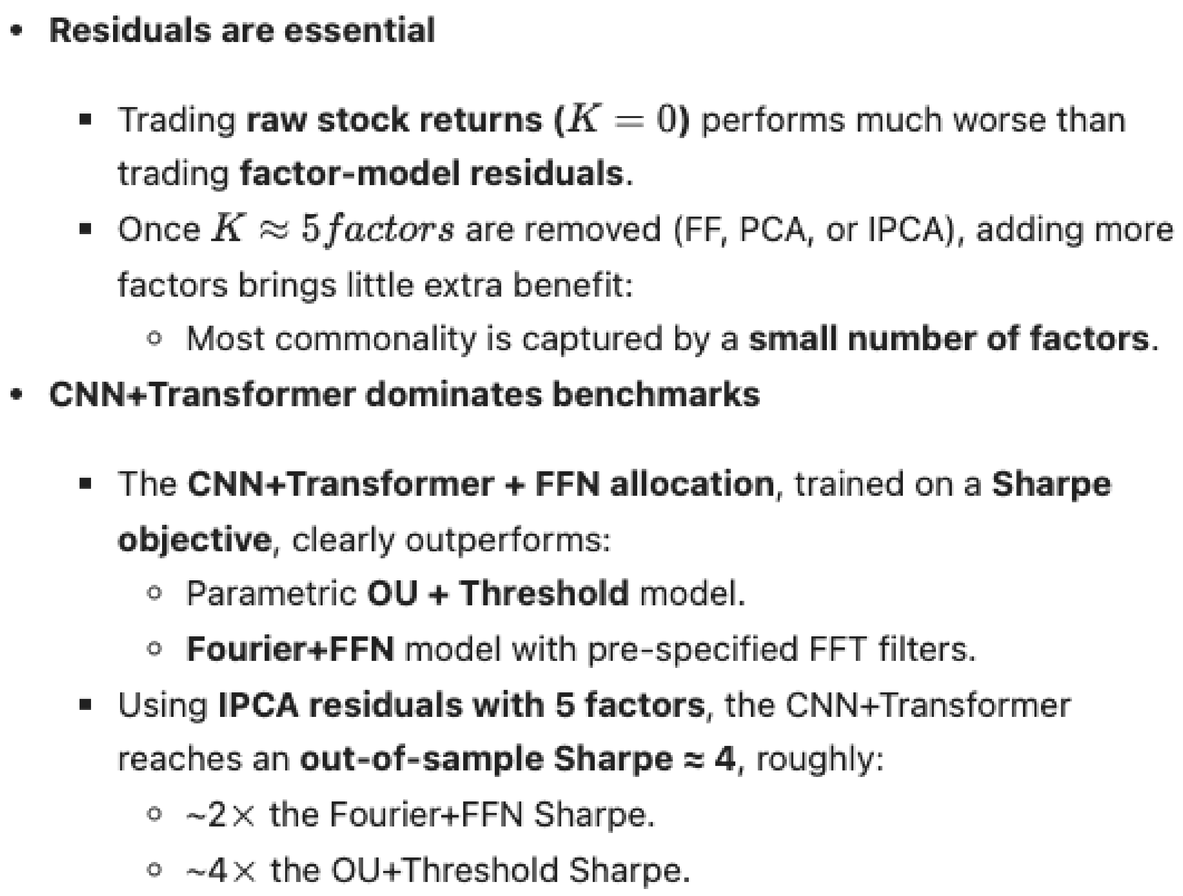
- Daily series for trading start in **1998**, after building a long enough history to estimate factor models and characteristics.
- Use **daily adjusted returns** (dividends and splits) from CRSP.
- Handle missing daily data conservatively:
  - Remove months or stocks with too many missing daily returns.
  - Ensures clean daily residual time series for the arbitrage strategies.

Table A.I: Firm Characteristics by Category

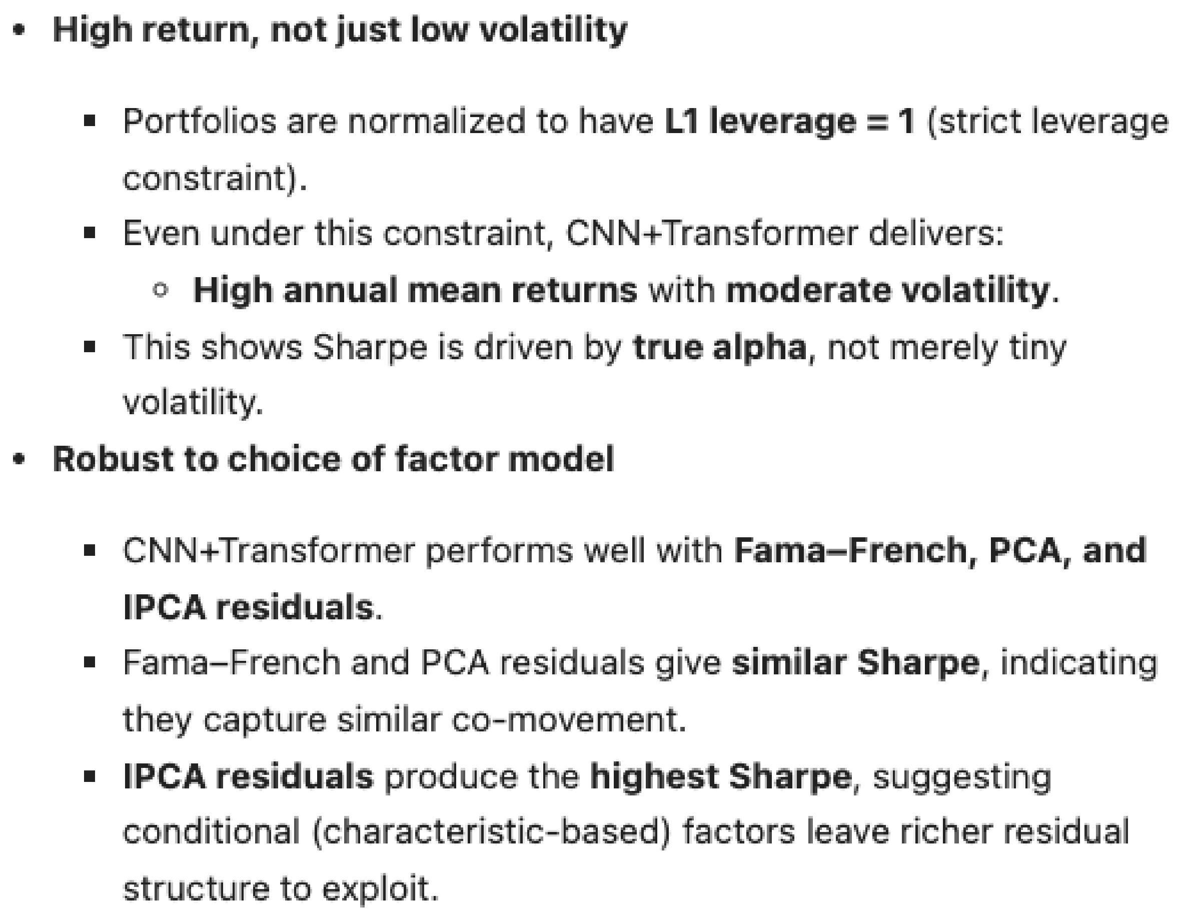
	Past Returns	
(1)	r2_1	Short-term momentum
(2)	r12_2	Momentum
(3)	r12_7	Intermediate momentum
(4)	r36_13	Long-term momentum
(5)	ST_Rev	Short-term reversal
(6)	LT_Rev	Long-term reversal
	Investment	
(7)	Investment	Investment
(8)	NOA	Net operating assets
(9)	DPI2A	Change in property, plants, and equipment
(10)	NI	Net Share Issues
	Profitability	
(11)	PROF	Profitability
(12)	ATO	Net sales over lagged net operating assets
(13)	CTO	Capital turnover
(14)	FC2Y	Fixed costs to sales
(15)	OP	Operating profitability
(16)	PM	Profit margin
(17)	RNA	Return on net operating assets
(18)	ROA	Return on assets
(19)	ROE	Return on equity
(20)	SGA2S	Selling, general and administrative expenses to sales
(21)	D2A	Capital intensity
	Intangibles	
(22)	AC	Accrual
(23)	OA	Operating accruals
(24)	OL	Operating leverage
(25)	PCM	Price to cost margin
	Value	
(26)	A2ME	Assets to market cap
(27)	BEME	Book to Market Ratio
(28)	C	Ratio of cash and short-term investments to total assets
(29)	CF	Free Cash Flow to Book Value
(30)	CF2P	Cashflow to price
(31)	D2P	Dividend Yield
(32)	E2P	Earnings to price
(33)	Q	Tobin's Q
(34)	S2P	Sales to price
(35)	Lev	Leverage
	Trading Frictions	
(36)	AT	Total Assets
(37)	Beta	CAPM Beta
(38)	IdioVol	Idiosyncratic volatility
(39)	LME	Size
(40)	LTurnover	Turnover
(41)	MktBeta	Market Beta
(42)	Rel2High	Closeness to past year high
(43)	Resid_Var	Residual Variance
(44)	Spread	Bid-ask spread
(45)	SUV	Standard unexplained volume
(46)	Variance	Variance

This table shows the 46 firm-specific characteristics sorted into six categories. More details on the construction are in the Internet Appendix of Chen et al. (2022).

Main Results (1)

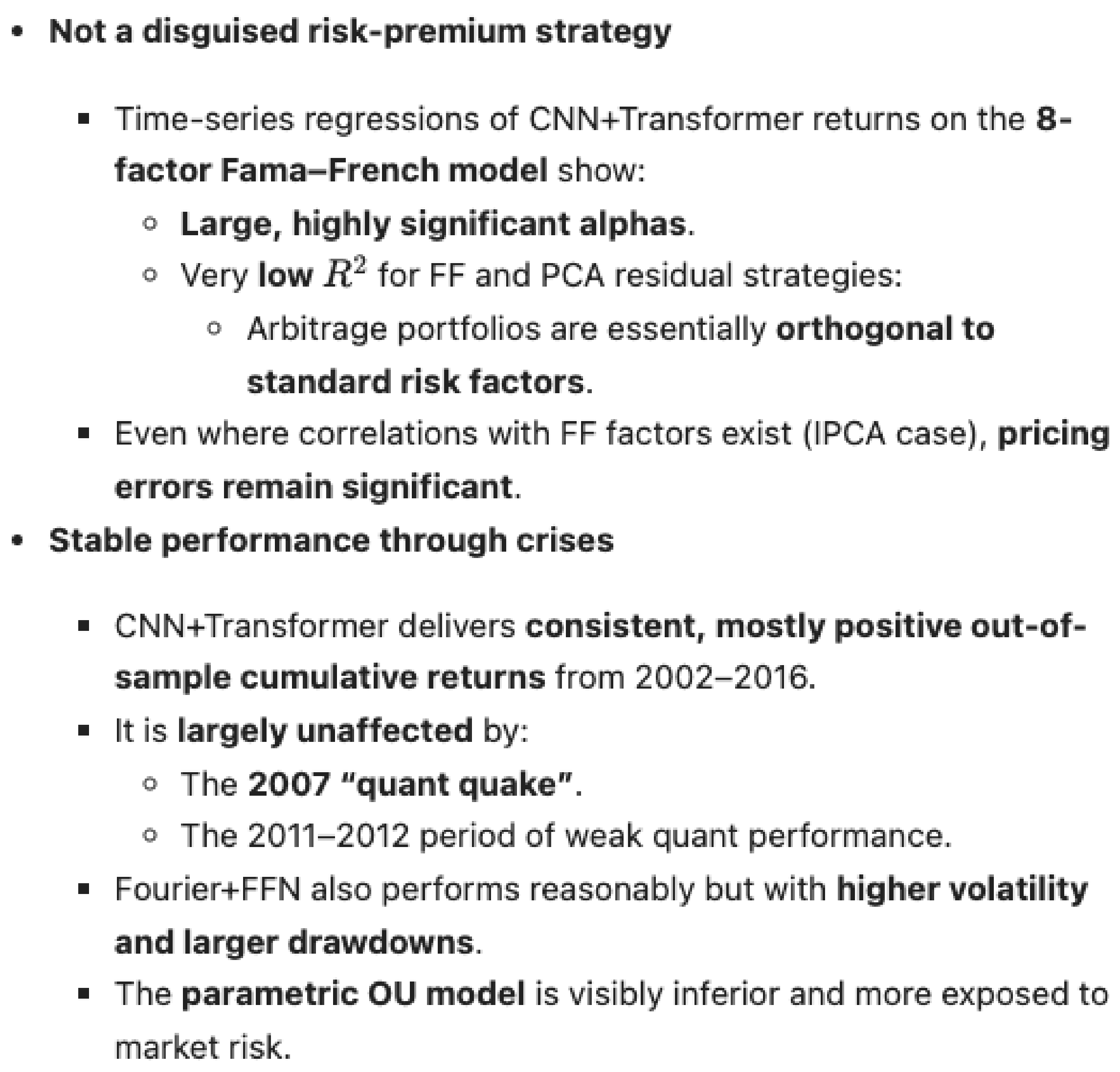


- **Residuals are essential**
  - Trading **raw stock returns ($K = 0$)** performs much worse than trading **factor-model residuals**.
  - Once $K \approx 5 factors$ are removed (FF, PCA, or IPCA), adding more factors brings little extra benefit:
    - Most commonality is captured by a **small number of factors**.
- **CNN+Transformer dominates benchmarks**
  - The **CNN+Transformer + FFN allocation**, trained on a **Sharpe objective**, clearly outperforms:
    - Parametric **OU + Threshold** model.
    - **Fourier+FFN** model with pre-specified FFT filters.
  - Using **IPCA residuals with 5 factors**, the CNN+Transformer reaches an **out-of-sample Sharpe ≈ 4**, roughly:
    - ~2× the Fourier+FFN Sharpe.
    - ~4× the OU+Threshold Sharpe.

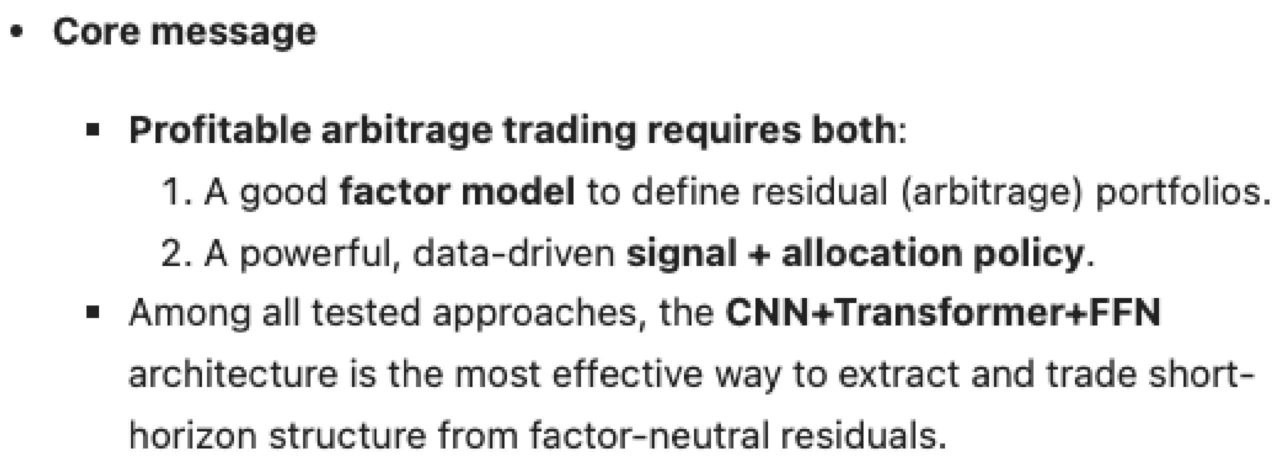


- **High return, not just low volatility**
  - Portfolios are normalized to have **L1 leverage = 1** (strict leverage constraint).
  - Even under this constraint, CNN+Transformer delivers:
    - **High annual mean returns** with **moderate volatility**.
  - This shows Sharpe is driven by **true alpha**, not merely tiny volatility.
- **Robust to choice of factor model**
  - CNN+Transformer performs well with **Fama–French, PCA, and IPCA residuals**.
  - Fama–French and PCA residuals give **similar Sharpe**, indicating they capture similar co-movement.
  - **IPCA residuals** produce the **highest Sharpe**, suggesting conditional (characteristic-based) factors leave richer residual structure to exploit.

Main Results (2)



- **Not a disguised risk-premium strategy**
  - Time-series regressions of CNN+Transformer returns on the **8-factor Fama–French model** show:
    - **Large, highly significant alphas**.
    - Very **low** R^2 for FF and PCA residual strategies:
      - Arbitrage portfolios are essentially **orthogonal to standard risk factors**.
  - Even where correlations with FF factors exist (IPCA case), **pricing errors remain significant**.
- **Stable performance through crises**
  - CNN+Transformer delivers **consistent, mostly positive out-of-sample cumulative returns** from 2002–2016.
  - It is **largely unaffected** by:
    - The **2007 "quant quake"**.
    - The 2011–2012 period of weak quant performance.
  - Fourier+FFN also performs reasonably but with **higher volatility and larger drawdowns**.
  - The **parametric OU model** is visibly inferior and more exposed to market risk.



- **Core message**
  - **Profitable arbitrage trading requires both**:
    1. A good **factor model** to define residual (arbitrage) portfolios.
    2. A powerful, data-driven **signal + allocation policy**.
  - Among all tested approaches, the **CNN+Transformer+FFN** architecture is the most effective way to extract and trade short-horizon structure from factor-neutral residuals.

Table I: OOS Annualized Performance Based on Sharpe Ratio Objective

	Factors	Fama-French			PCA			IPCA		
Model	K	SR	μ	σ	SR	μ	σ	SR	μ	σ
CNN + Trans	0	1.64	13.7%	8.4%	1.64	13.7%	8.4%	1.64	13.7%	8.4%
	1	3.68	7.2%	2.0%	2.74	15.2%	5.5%	3.22	8.7%	2.7%
	3	3.13	5.5%	1.8%	3.56	16.0%	4.5%	3.93	8.6%	2.2%
	5	3.21	4.6%	1.4%	3.36	14.3%	4.2%	4.16	8.7%	2.1%
	8	2.49	3.4%	1.4%	3.02	12.2%	4.0%	3.95	8.2%	2.1%
	10	-	-	-	2.81	10.7%	3.8%	3.97	8.0%	2.0%
	15	-	-	-	2.30	7.6%	3.3%	4.17	8.4%	2.0%
Fourier + FFN	0	0.36	4.9%	13.6%	0.36	4.9%	13.6%	0.36	4.9%	13.6%
	1	0.89	3.2%	3.5%	0.80	8.4%	10.6%	1.24	6.3%	5.0%
	3	1.32	3.5%	2.7%	1.66	11.2%	6.7%	1.77	7.8%	4.4%
	5	1.66	3.1%	1.8%	1.98	12.4%	6.3%	1.90	7.7%	4.1%
	8	1.90	3.1%	1.6%	1.95	10.1%	5.2%	1.94	7.8%	4.0%
	10	-	-	-	1.71	8.2%	4.8%	1.93	7.6%	3.9%
	15	-	-	-	1.14	4.8%	4.2%	2.06	7.9%	3.8%
OU + Thresh	0	-0.18	-2.4%	13.3%	-0.18	-2.4%	13.3%	-0.18	-2.4%	13.3%
	1	0.16	0.6%	3.8%	0.21	2.1%	10.4%	0.60	3.0%	5.1%
	3	0.54	1.6%	3.0%	0.77	5.2%	6.8%	0.88	3.8%	4.3%
	5	0.38	0.9%	2.3%	0.73	4.4%	6.1%	0.97	3.8%	4.0%
	8	1.16	2.8%	2.4%	0.87	4.4%	5.1%	0.91	3.5%	3.8%
	10	-	-	-	0.63	2.9%	4.6%	0.86	3.1%	3.6%
	15	-	-	-	0.62	2.4%	3.8%	0.93	3.2%	3.5%

This table shows the out-of-sample annualized Sharpe ratio (SR), mean return (μ), and volatility (σ) of our three statistical arbitrage models for different numbers of risk factors K, that we use to obtain the residuals. We use the daily out-of-sample residuals from January 1998 to December 2016 and evaluate the out-of-sample arbitrage trading from January 2002 to December 2016. CNN+Trans denotes the convolutional network with transformer model, Fourier+FFN estimates the signal with a FFT and the policy with a feedforward neural network and lastly, OU+Thres is the parametric Ornstein-Uhlenbeck model with thresholding trading policy. The two deep learning models are calibrated on a rolling window of four years and use the Sharpe ratio objective function. The signals are extracted from a rolling window of $L = 30$ days. The $K = 0$ factor model corresponds to directly using stock returns instead of residuals for the signal and trading policy.

Table II: Significance of Arbitrage Alphas based on Sharpe Ratio Objective

CNN+Trans model

	Fama-French					PCA					IPCA				
K	α	t_α	R^2	μ	t_μ	α	t_α	R^2	μ	t_μ	α	t_α	R^2	μ	t_μ
0	11.6%	6.4***	30.3%	13.7%	6.3***	11.6%	6.4***	30.3%	13.7%	6.3***	11.6%	6.4***	30.3%	13.7%	6.3***
1	7.0%	14***	2.4%	7.2%	14***	14.9%	10***	0.6%	15.2%	11***	8.1%	12***	9.5%	8.7%	12***
3	5.5%	12***	1.2%	5.5%	12***	15.8%	14***	1.7%	16.0%	14***	8.2%	15***	6.0%	8.6%	15***
5	4.5%	12***	2.3%	4.6%	12***	14.1%	13***	1.3%	14.3%	13***	8.3%	16***	3.9%	8.7%	16***
8	3.3%	9.4***	2.1%	3.4%	9.6***	12.0%	12***	0.9%	12.2%	12***	7.8%	15***	5.0%	8.2%	15***
10	-	-	-	-	-	10.5%	11***	0.7%	10.7%	11***	7.7%	15***	4.0%	8.0%	15***
15	-	-	-	-	-	7.5%	8.8***	0.5%	7.6%	8.9***	8.1%	16***	4.2%	8.4%	16***

Fourier+FFN model

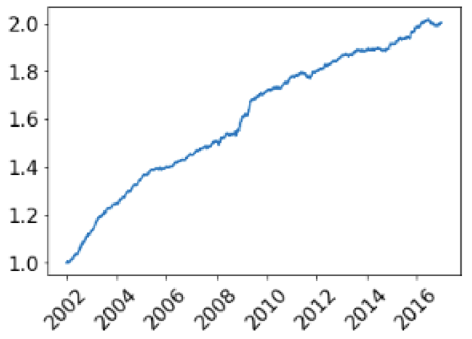
	Fama-French					PCA					IPCA				
K	α	t_α	R^2	μ	t_μ	α	t_α	R^2	μ	t_μ	α	t_α	R^2	μ	t_μ
0	2.7%	0.8	8.6%	4.9%	1.4	2.7%	0.8	8.6%	4.9%	1.4	2.7%	0.8	8.6%	4.9%	1.4
1	3.0%	3.3**	3.3%	3.2%	3.5***	7.4%	2.7**	3.3%	8.4%	3.1**	4.8%	4.0***	16.4%	6.3%	4.8***
3	3.2%	4.7***	4.2%	3.5%	5.1***	10.9%	6.3***	2.2%	11.2%	6.4***	6.8%	6.4***	13.0%	7.8%	6.9***
5	2.9%	6.1***	3.5%	3.1%	6.4***	12.1%	7.5***	1.5%	12.4%	7.6***	6.7%	6.9***	13.3%	7.7%	7.4***
8	3.0%	7.2***	3.2%	3.1%	7.4***	10.0%	7.5***	0.9%	10.1%	7.6***	6.8%	7.0***	13.3%	7.8%	7.5***
10	-	-	-	-	-	8.0%	6.5***	1.0%	8.2%	6.6***	6.8%	7.1***	12.7%	7.6%	7.5***
15	-	-	-	-	-	4.7%	4.3***	0.4%	4.8%	4.4***	7.1%	7.6***	12.2%	7.9%	8.0***

OU+Thresh model

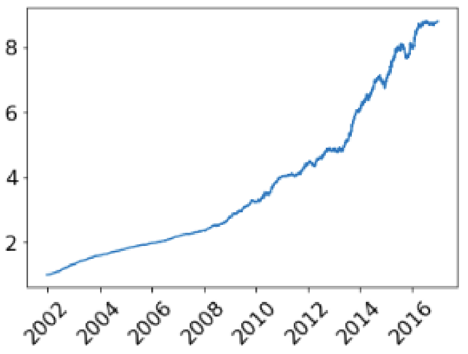
	Fama-French					PCA					IPCA				
K	α	t_α	R^2	μ	t_μ	α	t_α	R^2	μ	t_μ	α	t_α	R^2	μ	t_μ
0	-4.5%	-1.4	13.4%	-2.4%	-0.7	-4.5%	-1.4	13.4%	-2.4%	-0.7	-4.5%	-1.4	13.4%	-2.4%	-0.7
1	-0.2%	-0.2	13.5%	0.6%	0.6	0.7%	0.3	6.3%	2.1%	0.8	1.7%	1.4	18.9%	3.0%	2.3*
3	0.9%	1.2	10.4%	1.6%	2.1*	4.3%	2.5*	4.3%	5.2%	3.0**	2.6%	2.6**	18.8%	3.8%	3.4***
5	0.5%	0.9	6.8%	0.9%	1.5	3.7%	2.4*	3.2%	4.4%	2.8**	2.8%	3.0**	17.7%	3.8%	3.8***
8	0.6%	1.2	5.5%	1.0%	1.9	3.9%	3.0**	1.9%	4.4%	3.4***	2.3%	2.6**	17.6%	3.5%	3.6***
10	-	-	-	-	-	2.6%	2.2*	1.4%	2.9%	2.4*	2.1%	2.5*	17.6%	3.1%	3.3***
15	-	-	-	-	-	2.1%	2.1*	0.7%	2.4%	2.4*	2.3%	2.8**	18.1%	3.2%	3.6***

This table shows the out-of-sample pricing errors α of the arbitrage strategies relative of the Fama-French 8 factor model and their mean returns μ for the different arbitrage models and different number of factors K that we use to obtain the residuals. We run a time-series regression of the out-of-sample returns of the arbitrage strategies on the 8-factor model (Fama-French 5 factors + momentum + short-term reversal + long-term reversal) and report the annualized α, accompanying t-statistic value t_α, and the R^2 of the regression. In addition, we report the annualized mean return μ along with its accompanying t-statistic t_μ. The hypothesis test are two-sided and stars indicate p-values of 5% (*), 1% (**), and 0.1% (***). All results use the out-of-sample daily returns from January 2002 to December 2016 and the deep learning models are based on a Sharpe ratio objective.

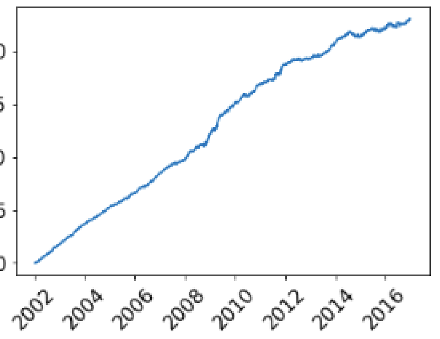
Figure 5. Cumulative OOS Returns of Different Arbitrage Strategies



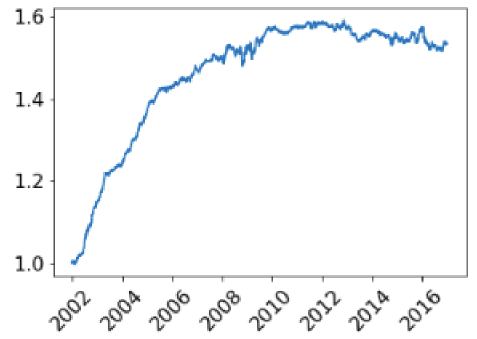
(a) CNN+Trans, Fama-French 5



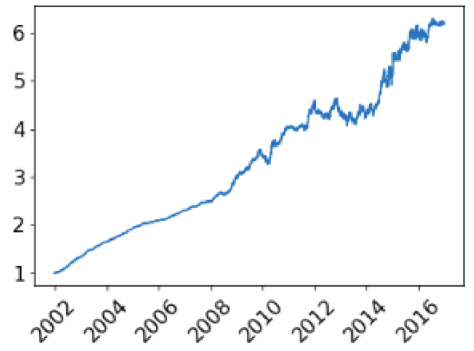
(b) CNN+Trans, PCA 5



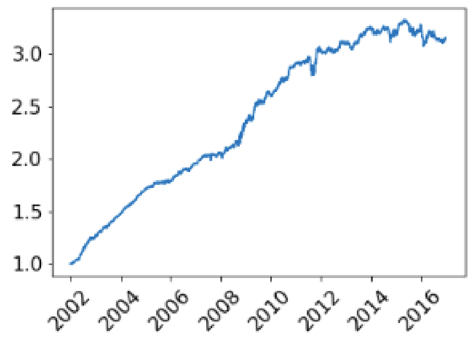
(c) CNN+Trans, IPCA 5



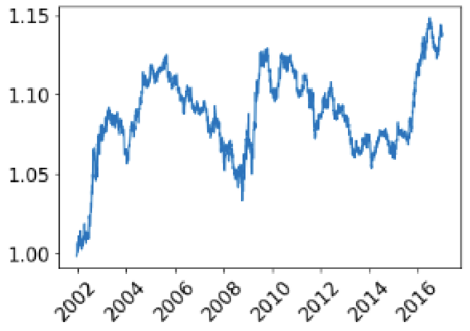
(d) Fourier+FFN, Fama-French 5



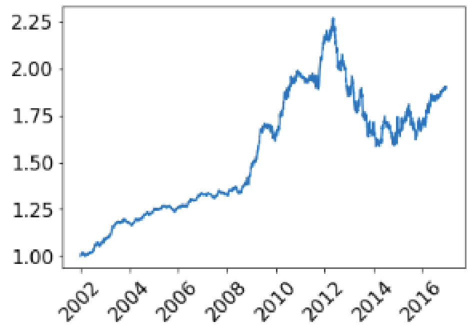
(e) Fourier+FFN, PCA 5



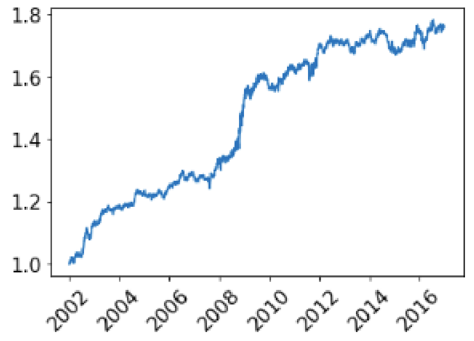
(f) Fourier+FFN, IPCA 5



(g) OU+Thresh Fama-French 5



(h) OU+Thresh PCA 5



(i) OU+Thresh IPCA 5

These figures show the cumulative daily returns of the arbitrage strategies for our representative models on the out-of-sample trading period between January 2002 and December 2016. We estimate the optimal arbitrage trading strategies for our three benchmark models based on the out-of-sample residuals of the Fama-French, PCA and IPCA 5-factor models. The deep learning models use the Sharpe ratio objective.

Replication repo

https://github.com/omroot/DeepLearningStatisticalArbitrage_PaperReplication

References

- Jorge Guijarro-Ordonez, Markus Pelger, Greg Zanotti (2021), Deep Learning Statistical Arbitrage