|Q⟩OGNITIVE

# AI for Finance:

## Forecasting Equity Returns

# What we are not covering

Applications outside of direct forecasting of daily stock returns

- Reinforcement Learning
  - Interesting papers by e.g. Gordan Ritter (Machine Learning for Trading)

# What we are not covering

Applications outside of direct forecasting of daily stock returns

- LLMs
  - Various obvious applications to text processing for feature generation
  - Time series?  Architectures combining text processing and returns history seem more interesting

# What we are not covering

Applications outside of direct forecasting of daily stock returns

- Intraday forecasting
  - Higher ratio of signal to noise

# Momentum Transformer - a paper to motivate discussion

**Trading with the Momentum Transformer: An Intelligent and Interpretable Architecture**

Kieran Wood*, Sven Giegerich†, Stephen Roberts*, Stefan Zohren*
*Oxford-Man Institute of Quantitative Finance, University of Oxford
†Oxford Internet Institute, University of Oxford

"We introduce the Momentum Transformer, … which outperforms benchmark time-series momentum **and** mean-reversion trading strategies."

# Momentum Transformer – Return test / Target / Loss

"The univariate TSMOM strategies we focus on differ from the cross-sectional approach which studies the comparative performance of assets."

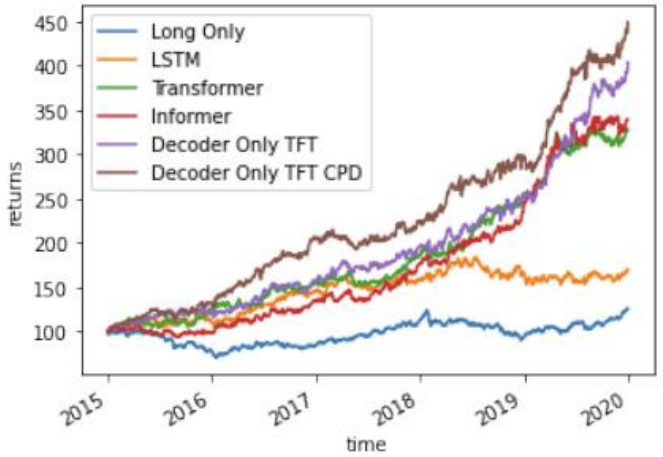
Fine for CTAs, some hedge funds. Target is elsewhere stated as 1 day forward Sharpe ratio, backtest portfolios are rebalanced daily.

"We train … with loss function selected … for maximising Sharpe ratio"

Nice!

# Momentum Transformer – Main Result

| | Returns | Vol. | Sharpe | Down. Dev. | Sortino |
|---|---|---|---|---|---|
| **Average 1995–2020** | | | | | |
| Long-Only | 2.45% | 4.95% | 0.51 | 3.51% | 0.73 |
| TSMOM | 4.43% | 4.47% | 1.03 | 3.11% | 1.51 |
| LSTM | 2.71% | 1.67% | 1.70 | 1.10% | 2.66 |
| Transformer | 3.14% | 2.49% | 1.41 | 1.68% | 2.13 |
| Decoder-Only Trans. | 2.95% | 2.61% | 1.11 | 1.74% | 1.69 |
| Conv. Transformer | 2.94% | 2.75% | 1.07 | 1.87% | 1.60 |
| Informer | 2.39% | 1.38% | 1.72 | 0.89% | 2.67 |
| Decoder-Only TFT | **4.01%** | 1.54% | 2.54 | 0.96% | 4.14 |
| Decoder-Only TFT CPD | 3.70% | **1.37%** | **2.62** | **0.85%** | **4.25** |




What is TSMOM?

# Momentum Transformer – Small Details

What is TSMOM?  Digging through paper, it is not defined.

We do see these interesting, unrelated, notes:

“We winsorise our data by limiting it to be within 5 times its exponentially weighted moving (EWM) standard deviations from its EWM average, using a 252-day half-life.”

“We keep the last 10% of the training data, for each asset, as a validation set. We implement random grid search, as an outer optimisation loop, to select the best hyperparameters, based on the validation set.”

# Momentum Transformer – TSMOM Benchmark

What is TSMOM?  Digging through online code we find:

```
returns_data["position"] = intermediate_momentum_position(0, returns_data)
…
def intermediate_momentum_position(w: float, returns_data: pd.DataFrame):
return w * np.sign(returns_data["norm_monthly_return"]) + (1 - w) * np.sign(
        returns_data["norm_annual_return"]
    )
```

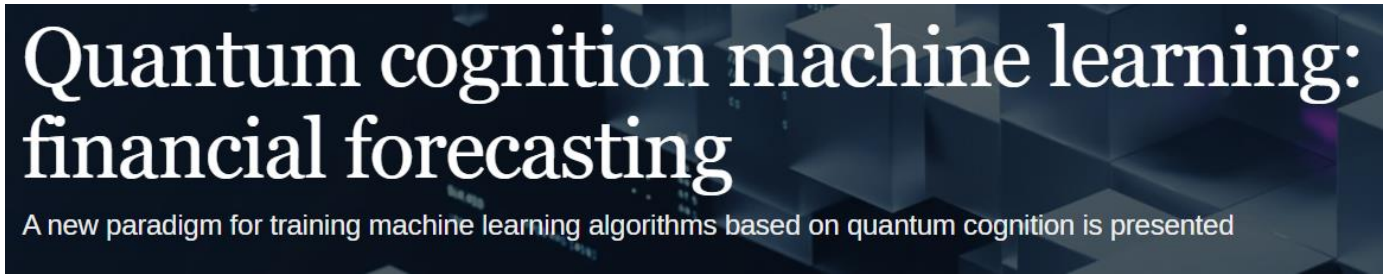
# Momentum Transformer – Reconsidering Main Result

"Our input features … include returns at different timescales … corresponding to daily, monthly, quarterly, biannual and annual returns, which are normalised using ex-ante volatility…

We also use MACD indicators which are a volatility normalised moving average convergence divergence indicator"

| | Returns | Vol. | Sharpe | Down. Dev. | Sortino |
|---|---|---|---|---|---|
| **Average 1995–2020** | | | | | |
| Long-Only | 2.45% | 4.95% | 0.51 | 3.51% | 0.73 |
| TSMOM | 4.43% | 4.47% | 1.03 | 3.11% | 1.51 |
| LSTM | 2.71% | 1.67% | 1.70 | 1.10% | 2.66 |
| Transformer | 3.14% | 2.49% | 1.41 | 1.68% | 2.13 |
| Decoder-Only Trans. | 2.95% | 2.61% | 1.11 | 1.74% | 1.69 |
| Conv. Transformer | 2.94% | 2.75% | 1.07 | 1.87% | 1.60 |
| Informer | 2.39% | 1.38% | 1.72 | 0.89% | 2.67 |
| Decoder-Only TFT | **4.01%** | 1.54% | 2.54 | 0.96% | 4.14 |
| Decoder-Only TFT CPD | 3.70% | **1.37%** | **2.62** | **0.85%** | **4.25** |

# QCML & Neural Net – A reference for design and testing




"Here, Ryan Samson, Jeffrey Berger, Luca Candelori, Vahagn Kirakosyan, Kharen Musaelian and Dario Villani introduce a novel machine learning approach based on the ideas of quantum cognition, which they call quantum cognition machine learning (QCML)."

# QCML & Neural Net – Return test

To test performance, we use covariance estimated daily from daily returns[6] to produce Markowitz optimal investment portfolios, where portfolio weight $w = V^{-1}f$ given covariance $V$ and forecast $f$[7]. Given that the forecasts have been projected away from input and control features, these investment portfolios will also have no exposure to input and control features.[8]

Evaluating cross-sectional returns, after removing the linear effect of input features, as well as additional control features

# QCML & Neural Net - Projection

[5]If you wish to solve for portfolio weights which maximize expected returns, with a penalty for expected portfolio variance, while maintaining zero exposure to a set of controls, then for weights $w$, forecast $f$, asset variance $V$, risk aversion $\mu$, and controls $M$, you need to solve for $w$ which minimizes $-w^\intercal f + 0.5\mu w^\intercal V w$ such that $w^\intercal M = 0$. The solution is $w = V^{-1}Rf$ where $R = I - M(M^\intercal V^{-1} M)^{-1} M^\intercal V^{-1}$. Thus $R$ is a projection operator that projects away from $M$, consistent with our desired investment process. $Rf$ is also the residual from an inverse variance weighted regression of $f$ on $M$

Our projection is:

- Equivalent to residualization with weighted linear regression
- Consistent with portfolio management process
- Focuses results on non-linear component of these forecasts

# QCML & Neural Net – Target

Our target variable for purposes of model training is 15 day forward log returns, projected[5] away from model input features as well as Beta, Size, and GICS Dummies. After projection, target returns are cross-sectionally normalized.

Our target is:

- Multi-day, generally results in better learning
- Projected from inputs / controls – focused on the non-linear component
- Cross-sectionally normalized daily

# QCML & Neural Net – Additional Details

To create more robust forecasts, we partition stocks into randomized groups of approximately 50, training individual NN and QCML models over each subset, and average forecasts for each stock across 100 different such partitions.

# Duality Tests – Importance of multiple seeds

Test:

- 13 Neural Net configurations Duality Group used in production, configurations differ by input feature set
- Models trained on full stock universe
- 16 partitions for each model
  - Each sub-partition has a different random seed derived from main seed
  - Sub-partitions differ by initialization and order training data is applied
  - Final forecast is averaged across 16 seeds
- Run 4 different main seeds, look at forecast correlation across seeds

| Model | Average Forecast Correlation |
|---|---|
| 1 | 0.97 |
| 2 | 0.82 |
| 3 | 0.92 |
| 4 | 0.86 |
| 5 | 0.62 |
| 6 | 0.86 |
| 7 | 0.74 |
| 8 | 0.58 |
| 9 | 0.82 |
| 10 | 0.92 |
| 11 | 0.89 |
| 12 | 0.83 |
| 13 | 0.87 |

# QCML & Neural Net – Additional Details

The NN architecture and training approach we use adheres fairly closely to that recommended in [14], other than our more complex approach to ensembling. Our neural network is implemented in PyTorch and contains 3 hidden layers of 32, 16 and 8 nodes respectively, with batch normalization [16] applied prior to ReLU activation [18, 23]. We use a simple mean squared error loss function, and train using stochastic gradient descent and the Adam optimizer.

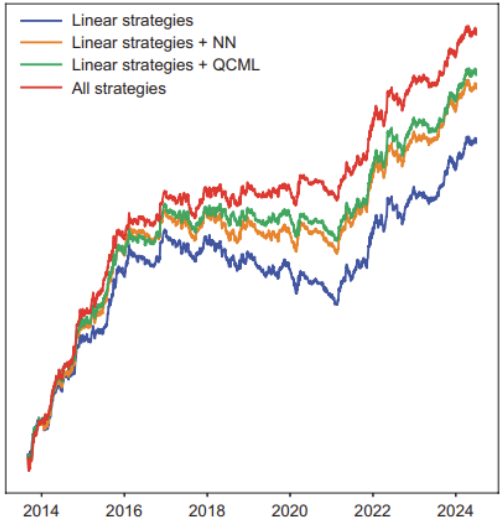
[14] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, 33(5):2223–2273, 02 2020.

# QCML & Neural Net – Main Results

**Table 5:** Sharpe Ratios of returns to equal-risk combined linear forecasts for all features, those linear forecasts with NN Extended Model forecasts added, those linear forecasts with QCML Extended Model forecasts added, and all Extended strategies. The QCML forecast provides better diversification than the NN forecast, but the best Sharpe Ratio is achieved by using both the NN and QCML forecasts.

| Period | Linear Strategies | Linear Strategies + NN | Linear Strategies + QCML | All Strategies |
|---|---|---|---|---|
| Sep 2013 - Jun 2024 | 1.17 | 1.38 | 1.43 | 1.58 |
| Sep 2013 - Apr 2017 | 2.31 | 2.57 | 2.66 | 2.82 |
| May 2017 - Dec 2020 | -0.66 | -0.32 | -0.26 | 0.03 |
| Jan 2021 - Jun 2024 | 1.77 | 1.81 | 1.82 | 1.83 |

6 Cumulative returns from equal-risk combined linear forecasts for all features, from the same linear forecasts but with either the NN or QCML extended-model forecasts added, and from all extended strategies




Returns have been scaled by full-sample realised volatility

# Duality Tests – Simple FCL sizing

Average Sharpe ratio across 10 production configuration Neural Networks

| | Hidden Layer Sizes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Dropout Rate** | **5** | **10** | **20** | **40** | **100** | **5, 5** | **10, 10** | **20, 20** |
| **0** | 0.43 | 0.48 | 0.49 | 0.47 | 0.35 | - | - | - |
| **0.1** | 0.40 | 0.45 | 0.50 | 0.49 | 0.50 | 0.42 | 0.51 | 0.41 |
| **0.25** | 0.50 | 0.50 | 0.50 | 0.46 | 0.46 | 0.18 | 0.32 | 0.41 |
| **0.4** | 0.45 | 0.40 | 0.43 | 0.46 | 0.42 | - | - | - |

# Duality Tests – Simple FCL sizing

Average Sharpe ratio across 78 Neural Networks with random feature pairs

**Average Sharpe**

| | Hidden Layer Sizes | | |
|---|---|---|---|
| **Dropout Rate** | **5** | **10** | **20** |
| **0** | -0.11 | -0.04 | -0.05 |
| **0.1** | -0.17 | -0.17 | -0.16 |
| **0.25** | -0.18 | -0.17 | -0.15 |

# Duality Tests – Prelude to complex architectures

Validation Sharpe Ratios

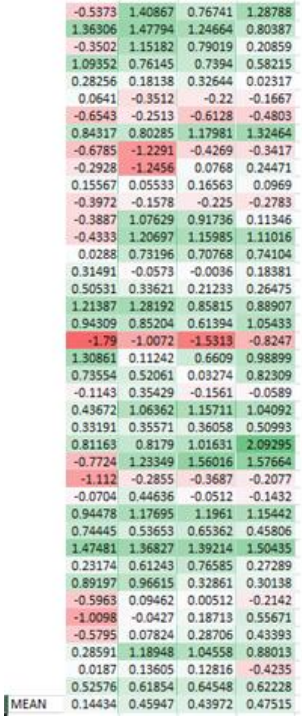
- 1 hidden layer, 10 nodes
- Column 1: 1 lag, 2-4: 8 lags
- Dropout by column 0 / 0 / 0.25 / 0.50

Mean Results:

| Lags | 1 | 8 | 8 | 8 |
|---|---|---|---|---|
| Dropout | 0% | 0% | 25% | 50% |
| Mean Sharpe | 0.14 | 0.46 | 0.44 | 0.48 |

Aggregate Results (8 lags, no dropout):

| Train | Validate | Test |
|---|---|---|
| 2.70 | 1.21 | 1.98 |



| | | | | |
|---|---|---|---|---|
| | -0.5373 | 1.40867 | 0.76741 | 1.28788 |
| | 1.36306 | 1.47794 | 1.24664 | 0.80387 |
| | -0.3502 | 1.15182 | 0.79019 | 0.20859 |
| | 1.09352 | 0.76145 | 0.7394 | 0.58215 |
| | 0.28256 | 0.18138 | 0.32644 | 0.02317 |
| | 0.0641 | -0.3512 | -0.22 | -0.1667 |
| | -0.6543 | -0.2513 | -0.6128 | -0.4803 |
| | 0.84317 | 0.80285 | 1.17981 | 1.32464 |
| | -0.6785 | -1.2291 | -0.4269 | -0.3417 |
| | -0.2928 | -1.2456 | 0.0768 | 0.24471 |
| | 0.15567 | 0.05533 | 0.16563 | 0.0969 |
| | -0.3972 | -0.1578 | -0.225 | -0.2783 |
| | -0.3887 | 1.07629 | 0.91736 | 0.11346 |
| | -0.4333 | 1.20697 | 1.15985 | 1.11016 |
| | 0.0288 | 0.73196 | 0.70768 | 0.74104 |
| | 0.31491 | -0.0573 | -0.0036 | 0.18381 |
| | 0.50531 | 0.33621 | 0.21233 | 0.26475 |
| | 1.21387 | 1.28192 | 0.85815 | 0.88907 |
| | 0.94309 | 0.85204 | 0.61394 | 1.05433 |
| | -1.79 | -1.0072 | -1.5313 | -0.8247 |
| | 1.30861 | 0.11242 | 0.6609 | 0.98899 |
| | 0.73554 | 0.52061 | 0.03274 | 0.82309 |
| | -0.1143 | 0.35429 | -0.1561 | -0.0589 |
| | 0.43672 | 1.06362 | 1.15711 | 1.04092 |
| | 0.33191 | 0.35571 | 0.36058 | 0.50993 |
| | 0.81163 | 0.8179 | 1.01631 | 2.09295 |
| | -0.7724 | 1.23349 | 1.56016 | 1.57664 |
| | -1.112 | -0.2855 | -0.3687 | -0.2077 |
| | -0.0704 | 0.44636 | -0.0512 | -0.1432 |
| | 0.94478 | 1.17695 | 1.1961 | 1.15442 |
| | 0.74445 | 0.53653 | 0.65362 | 0.45806 |
| | 1.47481 | 1.36827 | 1.39214 | 1.50435 |
| | 0.23174 | 0.61243 | 0.76585 | 0.27289 |
| | 0.89197 | 0.96615 | 0.32861 | 0.30138 |
| | -0.5963 | 0.09462 | 0.00512 | -0.2142 |
| | -1.0098 | -0.0427 | 0.18713 | 0.55671 |
| | -0.5795 | 0.07824 | 0.28706 | 0.43393 |
| | 0.28591 | 1.18948 | 1.04558 | 0.88013 |
| | 0.0187 | 0.13605 | 0.12816 | -0.4235 |
| | 0.52576 | 0.61854 | 0.64548 | 0.62228 |
| MEAN | 0.14434 | 0.45947 | 0.43972 | 0.47515 |

# Duality Tests – Prelude to complex architectures

Take strongest features from prior test / config, run pairs (validation Sharpe below):

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.12 | 0.28 | 0.18 | 0.20 | 0.31 | 0.42 | 0.82 | 0.68 | 0.66 | 0.51 | 0.24 | 0.88 | 1.13 | 0.63 |
| 0.12 | | 0.67 | 0.56 | 0.53 | 0.54 | 0.64 | 0.65 | 1.15 | 1.07 | 0.90 | 0.54 | 1.07 | 1.33 | 0.91 |
| 0.28 | 0.67 | | 0.97 | 0.77 | 0.78 | 0.75 | 0.97 | 1.72 | 1.31 | 1.49 | 0.77 | 1.27 | 1.54 | 0.97 |
| 0.18 | 0.56 | 0.97 | | 0.71 | 0.63 | 0.70 | 0.73 | 1.41 | 1.12 | 1.14 | 0.72 | 1.17 | 1.40 | 0.92 |
| 0.20 | 0.53 | 0.77 | 0.71 | | 0.53 | 0.58 | 0.62 | 1.17 | 1.08 | 1.03 | 0.62 | 1.28 | 1.41 | 0.87 |
| 0.31 | 0.54 | 0.78 | 0.63 | 0.53 | | 0.57 | 0.60 | 1.12 | 0.94 | 0.94 | 0.65 | 0.97 | 1.23 | 0.76 |
| 0.42 | 0.64 | 0.75 | 0.70 | 0.58 | 0.57 | | 0.45 | 0.96 | 0.86 | 0.85 | 0.67 | 0.99 | 1.16 | 0.81 |
| 0.82 | 0.65 | 0.97 | 0.73 | 0.62 | 0.60 | 0.45 | | 1.01 | 0.76 | 0.97 | 0.55 | 0.69 | 0.92 | 0.48 |
| 0.68 | 1.15 | 1.72 | 1.41 | 1.17 | 1.12 | 0.96 | 1.01 | | 1.20 | 1.85 | 0.99 | 1.34 | 1.56 | 1.22 |
| 0.66 | 1.07 | 1.31 | 1.12 | 1.08 | 0.94 | 0.86 | 0.76 | 1.20 | | 1.45 | 0.98 | 1.23 | 1.43 | 1.05 |
| 0.51 | 0.90 | 1.49 | 1.14 | 1.03 | 0.94 | 0.85 | 0.97 | 1.85 | 1.45 | | 1.09 | 1.40 | 1.62 | 1.17 |
| 0.24 | 0.54 | 0.77 | 0.72 | 0.62 | 0.65 | 0.67 | 0.55 | 0.99 | 0.98 | 1.09 | | 0.99 | 1.23 | 0.89 |
| 0.88 | 1.07 | 1.27 | 1.17 | 1.28 | 0.97 | 0.99 | 0.69 | 1.34 | 1.23 | 1.40 | 0.99 | | 1.35 | 1.04 |
| 1.13 | 1.33 | 1.54 | 1.40 | 1.41 | 1.23 | 1.16 | 0.92 | 1.56 | 1.43 | 1.62 | 1.23 | 1.35 | | 1.16 |
| 0.63 | 0.91 | 0.97 | 0.92 | 0.87 | 0.76 | 0.81 | 0.48 | 1.22 | 1.05 | 1.17 | 0.89 | 1.04 | 1.16 | |

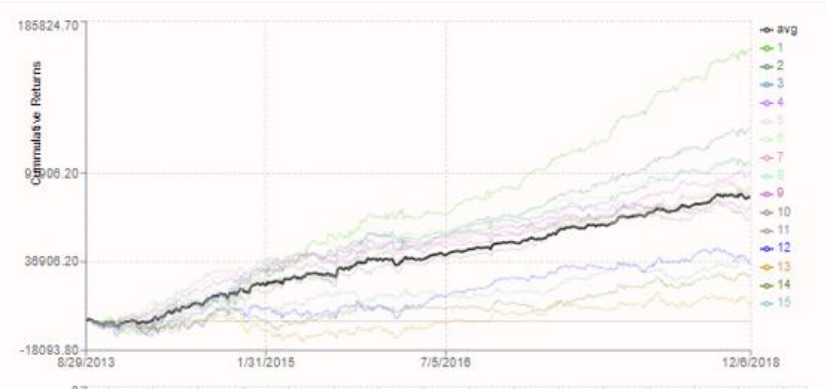
Aggregate results:

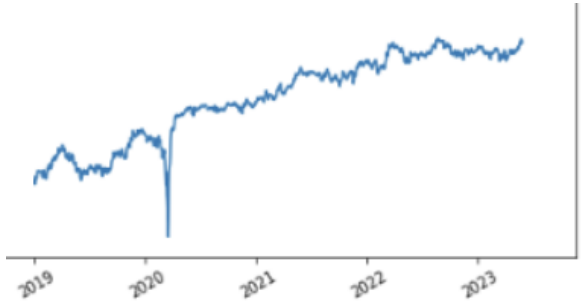
| | Train | Validate | Test |
|---|---|---|---|
| All | 2.47 | 1.24 | 1.81 |
| Strong / Weak | 2.66 | 1.34 | 2.11 |

# Duality Tests – Prelude to complex architectures

Take strongest features from prior test / config, add all weak features:

| | Train | Validate | Test | Embargo |
|---|---|---|---|---|
| Strong / All Weak | 2.82 | 1.72 | 2.13 | 0.71 |

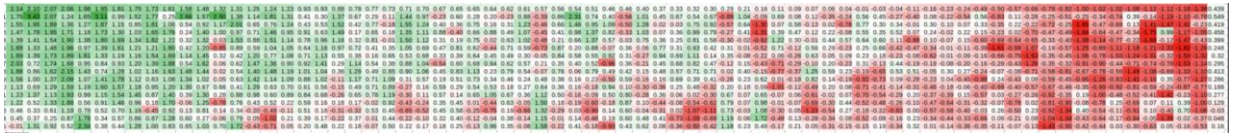


# Duality Tests – RNN

Expanded feature set, 10 lags

Simple FCL: 10 Hidden Nodes. Average Sharpe 0.44



Simple RNN: 5 RNN Nodes, use only final output, aggregate with linear layer. Average Sharpe 0.47

Additional improvement from residual connection

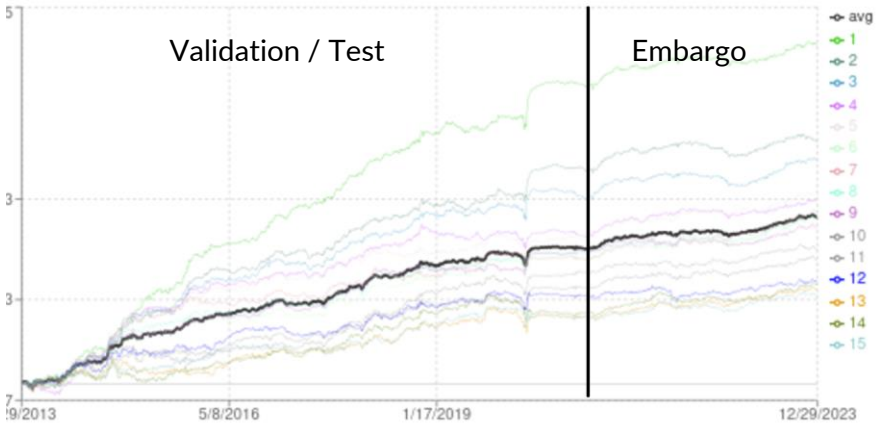
# Duality Tests – RNN

Additional variations (using all hidden states, varying layers, varying how layers stack, biases, residual connections etc.). Aggregate Sharpe ratios:

1 Layer, Final State Only, 3 RNN Nodes, Add Residual Connection →

| modules | A |
|---|---|
| #fcl_(10_5_T) | 1.33 |
| #rnn_L2ph3_(W10_I1_H6_B) | 1.32 |
| #rnn_L2ph3_(W10_I1_H6_B_RN) | 1.55 |
| #rnn_L2_NS_(W10_I1_H3_B) | 1.39 |
| #rnn_L2_NS_(W10_I1_H3_B_RN) | 1.59 |
| #rnn_L2_S_(W10_I1_H3_B) | 1.44 |
| #rnn_L2_S_(W10_I1_H3_B_RN) | 1.61 |
| #rnn_L1_NS_(W10_I1_H3_B) | 1.31 |
| #rnn_L1_NS_(W10_I1_H3_B_RN) | 1.64 |
| #rnn_L1_S_(W10_I1_H3_B) | 1.71 |
| #rnn_L1_S_(W10_I1_H3_B_RN) | 1.53 |

# Duality Tests – RNN, final result




Minor Marginal Impact on Backtest


Later replaced with an even **simpler** architecture

# Duality Tests – Final Comments

Also experimented with CNN, LSTM, Attention, none of which “made the cut”

# Duality Tests – Final Comments

We were a small team – we didn't try everything, or everything perfectly.

For example, in Momentum Transformer paper, huge variation in performance across exact architecture (though unclear if multiple seeds were used to reduce noise)

| | Returns | Vol. | Sharpe | Down. Dev. | Sortino |
|---|---|---|---|---|---|
| **Average 1995–2020** | | | | | |
| Long-Only | 2.45% | 4.95% | 0.51 | 3.51% | 0.73 |
| TSMOM | 4.43% | 4.47% | 1.03 | 3.11% | 1.51 |
| LSTM | 2.71% | 1.67% | 1.70 | 1.10% | 2.66 |
| Transformer | 3.14% | 2.49% | 1.41 | 1.68% | 2.13 |
| Decoder-Only Trans. | 2.95% | 2.61% | 1.11 | 1.74% | 1.69 |
| Conv. Transformer | 2.94% | 2.75% | 1.07 | 1.87% | 1.60 |
| Informer | 2.39% | 1.38% | 1.72 | 0.89% | 2.67 |
| Decoder-Only TFT | **4.01%** | 1.54% | 2.54 | 0.96% | 4.14 |
| Decoder-Only TFT CPD | 3.70% | **1.37%** | **2.62** | **0.85%** | **4.25** |

# Duality Tests – Final Comments

**When evaluating an architecture, accurately define the benchmark**

# Thank you!