

Assignment 8 – Neural Beta for Crypto and Cross-Asset Markets

Overview (Weight: 10 %)

The goal of this assignment is to extend the neural-beta methodology you learned in Assignment 7 to settings where the underlying assets are cryptocurrencies. In addition to reinforcing your understanding of multilayer perceptrons (MLPs) and hyper-parameter tuning, this project will allow you to explore how crypto returns co-move with fiat currencies, U.S. equity markets, gold and energy. Recent research suggests that the risk-return trade-off of crypto assets is quite different from those of stocks, currencies and commodities and that returns are largely driven by crypto-specific factors. Market commentary also shows that correlations between bitcoin and traditional hedges such as gold can change abruptly. By completing this assignment you will examine whether neural networks are able to capture these dynamics.

Learning objectives

- Familiarize yourself with the **PyTorch** ecosystems for building neural networks.
- Practice calculating **neural β** using a simple MLP and interpret the results in a multi-asset setting.
- Perform hyper-parameter tuning (hidden units, learning rate, activation functions, look-back window) to improve model performance.
- Analyze the distribution of neural β across cryptocurrencies and across asset pairs (fiat, equity, gold and energy).
- Examine whether adding additional factors (e.g., Fama–French factors) improves model accuracy.

Deliverables

Your submission should include:

1. A **Jupyter notebook** (.ipynb) containing your code, clearly commented and organised.

2. An **HTML** or **PDF** output generated from the notebook showing the results.
3. A **five-page PDF** summarising your analyses and findings.

You **do not** need to submit the raw data files;

Data

We provide **raw daily price series** rather than pre-computed returns so that you can practise cleaning and transforming data yourself. All cryptos and market proxies are delivered as separate CSV files; each file contains a date column and a price column. You will need to **combine these series into a single dataset** so that one neural network can learn from all cryptos simultaneously.

file name	series description and source
CBBTCUSD.csv	Bitcoin price in U.S. dollars from Coinbase (CBBTCUSD, via FRED)
CBETHUSD.csv	Ethereum price in U.S. dollars from Coinbase (CBETHUSD, via FRED)
CBLTCUSD.csv	Litecoin price in U.S. dollars from Coinbase (CBLTCUSD, via FRED)
CBBCHUSD.csv	Bitcoin Cash price in U.S. dollars from Coinbase (CBBCHUSD, via FRED)
DTWEXBGS.csv	Nominal broad U.S. Dollar index (DTWEXBGS) – proxy for fiat
CRSP.csv	Use the value-weighted portfolio of all U.S. securities in the CRSP universe (VWRETD) from CRSP
NASDAQQGLDI.csv	Credit Suisse NASDAQ Gold FLOWS103 Price Index (NASDAQQGLDI) – proxy for gold
VDE.csv	Vanguard Energy ETF (VDE) daily price – proxy for energy

The four crypto price series are sourced from Coinbase via the Federal Reserve Bank of St Louis's FRED database. The fiat index and gold series also come from FRED. The energy series uses daily closing prices for the **Vanguard Energy ETF (VDE)** (ticker **VDE**); this fund tracks a portfolio of U.S. energy companies and serves as a more realistic energy input than crude oil. **These are unprocessed daily observations.** To prepare the data:

1. **Parse and clean** each file: convert the dates to a common datetime format, sort chronologically and handle missing values (e.g., forward fill weekends/holidays or drop missing observations).
2. **Aggregate to monthly frequency** using the last available price in each month. You are welcome to experiment with other frequencies, but monthly is the baseline.
3. **Compute returns** for each series either in log form ($\log P_t - \log P_{t-1}$) or as percentage returns $r_t = P_t/P_{t-1} - 1$, and align the series by date. When constructing features you may use **rolling or sliding windows** of returns rather than just contemporaneous

month-to-month changes; for example, create a feature vector containing the last w months of returns for each asset.

After computing monthly returns and constructing your feature windows, split the sample into training, validation and test sets using the same cut-offs as the original assignment:

- **Training:** 2015-01 through 2019-12
- **Validation:** 2020-01 through 2021-12
- **Test:** 2021-01 through 2023-12

Assignment

This assignment follows the structure of Assignment 7, but you will estimate neural β for cryptocurrencies relative to different asset classes. Throughout the assignment **keep the neural-network architecture and loss function the same**; innovation should come from using different underlying assets. Work through the steps below and clearly document your findings.

Step 0 – Sampling

To reduce the computational effort you will not analyse every available cryptocurrency. Instead **focus on four cryptos – Bitcoin (BTC), Ethereum (ETH), Litecoin (LTC) and Bitcoin Cash (BCH)** – and treat each as its own “industry.” Combine these crypto returns into one dataset alongside the four market proxies (fiat, U.S. equities, gold and energy). When constructing your sample, consider the following points:

- Each observation in your modelling dataset corresponds to a particular crypto at a particular time. For each crypto and date, build a **feature window** using the last w months of that crypto’s returns and the last w months of the chosen market return(s). **Use rolling or sliding windows** to construct these inputs; your feature vector should summarise the past w months rather than rely solely on a single month-to-month change.
- You are free to experiment with **different combinations of market inputs**. For example, you might include fiat **and** equity returns together when estimating a crypto-market beta, or focus solely on gold or energy. This flexibility lets you explore how multiple asset classes jointly relate to crypto returns.
- The energy input is now measured by the **Vanguard Energy ETF (VDE)**, which offers a broad representation of the U.S. energy sector. Use this series instead of crude oil prices.

By stacking all cryptos and chosen market returns into a single panel, you can estimate neural betas for each crypto–asset relationship without training a separate model for each pair. This mirrors how Assignment 7 handled many stocks simultaneously. Make sure your features

respect the look-back window (no future information) and utilise rolling or sliding windows to capture past dynamics.

Step 1 – Neural-network architecture

A multilayer perceptron (MLP) is a simple feed-forward network. It consists of an **input layer** of raw predictors, one or more **hidden layers** that interact and non-linearly transform the predictors, and an **output layer** that aggregates the hidden layers into an ultimate outcome. The goal of this step is to construct a neural network using data from time $t - w$ to t that can output an estimate of **neural beta** $\hat{\beta}_{t+1}$ which minimises the error for period $t + 1$:

$$\hat{\beta}_{t+1} = \arg \min_{\beta} L(r_{t+1}, \beta r_{t+1}^{(\text{asset})}),$$

where L is a loss function such as the root-mean-square error (RMSE). Here $\hat{\beta}_{t+1}$ is estimated from the dataset covering the look-back window $t - w$ through t .

Construct the NN model architecture to estimate $\hat{\beta}_{t+1}$ by following these steps:

- **Construct features for the input layer:** using data from $t - w$ to t , include the last w months of the crypto's own returns **and the last w months of your chosen market returns**. These market inputs may consist of one or **multiple asset classes** (fiat, equities, gold, energy or any combination thereof). For example, you might stack both fiat and equity returns alongside crypto returns in the same feature window. Think of other transformations (e.g., moving averages) that might help estimate $\hat{\beta}_{t+1}$.
- **Add a dense hidden layer:** add one or more dense (fully connected) hidden layers that transform the input features using a non-linear activation function.
- **Add an output layer:** include a single-neuron output layer which produces $\hat{\beta}_{t+1}$ with **linear activation**. Reflect on why a linear activation is appropriate for estimating beta.
- **Define a custom loss function:** implement a loss function based on the expression above. You can calculate the RMSE between the realised crypto return r_{t+1} and the product $\hat{\beta}_{t+1} \times r_{t+1}^{(\text{asset})}$, where $\hat{\beta}_{t+1}$ is the output of your network.
- **Decide on additional hyper-parameters:** think about suitable batch size, optimizer (e.g., Adam, SGD), learning rate schedule and momentum terms. These choices will be explored further in Step 2.

Step 2 – Hyper-parameter tuning

Use the validation set to tune the following hyper-parameters:

- **Number of neurons in the hidden layer:** choose three reasonable values (e.g., 4, 8, 16).

- **Learning rate:** choose three values spanning an order of magnitude (e.g., 0.001, 0.01, 0.1).
- **Activation function:** experiment with `linear`, `sigmoid`, `tanh` and `relu` activations.
- **Look-back window w :** try 12, 24 and 36 months.

Report the validation-set root-mean-square error for each combination and identify the best set of hyper-parameters for each **market input choice** (fiat, equity, gold, energy or combinations thereof). You do not need to train a separate network for each crypto; instead evaluate how different hyper-parameters perform when predicting crypto betas across the entire panel of cryptos.

Step 3 – Learning curves

For the best-performing hyper-parameters identified above, plot the training and validation loss over epochs. These plots should help you assess whether your network is over- or under-fitting.

Step 4 – Descriptive statistics of neural β

Using the test set (2023-01 through 2025-10), compute the estimated neural β for each crypto–asset pair. For each crypto and asset pair, calculate summary statistics – the number of observations, mean, standard deviation, skewness, kurtosis, minimum, maximum and selected percentiles (1 %, 5 %, 25 %, 50 %, 75 %, 95 %, 99 %). Present these in a clear table. Discuss any notable differences between crypto–fiat, crypto–equity, crypto–gold and crypto–energy betas.

Step 5 – Dynamics of neural β

For each crypto–asset pair, compute the annual mean and standard deviation of neural β over the test period. Plot the annual mean β for each crypto (different lines) to visualise how exposures have evolved over time. Comment on whether certain cryptocurrencies exhibit more stable exposures to particular asset classes.

Step 6 – Findings

Summarise your findings in a few bullet points. Address questions such as: Which asset class has the strongest influence on crypto returns? Do all cryptocurrencies respond similarly to fiat or equity markets? Are there periods when crypto betas are particularly high or low?

Step 7 – Including additional factors

Enhance your models by augmenting the input layer with **Fama–French five-factor data** (SMB, HML, RMW, CMA and the risk-free rate) and any other macro variables you deem relevant (e.g., inflation or interest rate data). Use the same look-back window w as before. Download the factor data from the [Fama–French data library](#) and align it with your monthly sample. Re-estimate neural β for each crypto–asset pair using this expanded feature space. Compare the resulting betas and RMSEs with those obtained in Steps 4–6. Does incorporating additional factors materially improve performance?

Step 8 – Discussion

In a few bullet points, compare the neural β s and prediction errors before and after adding the Fama–French factors. Which factors appear most relevant for cryptocurrencies? Is the marginal improvement worth the added model complexity? Provide intuition for your conclusions.

Step 9 – Sorting on neural β and portfolio returns

As in Assignment 7, we examine the relationship between neural beta and returns by forming portfolios. Because there are only four cryptocurrencies in our sample, a quintile break-down is not possible. Instead:

- **Sort the four cryptocurrencies into quartile portfolios** based on their estimated neural β for each asset class (from lowest β to highest β). When evaluating crypto–equity betas, for example, rank BTC, ETH, LTC and BCH by their β estimates and form four equally sized portfolios (one crypto per portfolio).
- **Form equal-weighted portfolios** for each quartile and compute the average β and the equal-weighted **excess return** (crypto return minus the paired asset return) for each portfolio.
- **Compute the difference** between the highest- β and lowest- β quartiles.
- **Repeat the previous steps for value-weighted portfolios**, where weights are proportional to the absolute mean return of each crypto.

Discuss whether high- β cryptos deliver higher excess returns than low- β cryptos.

Supplementary details

Recommended reading

- **Risks and Returns of Cryptocurrency** by Yukun Liu and Aleh Tsyyvinski: this research paper documents that cryptocurrency returns are largely uncorrelated with stocks, currencies and commodities, but that momentum and investor-attention proxies strongly forecast returns.
- **Gold and Bitcoin Decouple: What's Driving the Divergence?** (CME Group OpenMarkets article): during 2022–2024 bitcoin and gold moved in a relatively tight correlation, but in 2025 the relationship broke down as macroeconomic conditions changes. The article provides context for why crypto–gold betas may vary over time.
- *Cryptoassets: The Guide to Bitcoin, Blockchain, and Cryptocurrency for Investment Professionals* (CFA Institute Research Foundation brief): this report introduces cryptoassets as an investable asset class, explains sources of return and risk, and discusses how crypto fits (or does not fit) into a diversified institutional portfolio. It is useful background for understanding why crypto may not behave like traditional equities, bonds, or commodities.
- *Tokenization of Real and Financial Assets* (CFA Institute Research Foundation report): this report surveys how real-world assets (equity, debt, real estate, commodities, etc.) can be represented and traded on blockchains. It highlights the mechanics, benefits, and regulatory frictions of tokenization, and helps frame how crypto market structure is increasingly linked to traditional finance.