# DG-MoE Returns Forecasting Model – Review and Recommendations

## Target Variable: 5-Day Returns vs. Price or Drawdown

Using **5-day forward returns** (aggregated 5-day log-return) as the prediction target is a sound choice for this multi-asset model. Financial asset **prices** are non-stationary (they tend to drift upward over time and have no fixed mean), which makes direct price prediction tricky – a model can easily "cheat" by just outputting today's price as tomorrow's forecast and appear accurate without capturing any real insight. In contrast, **returns (price changes)** are much closer to stationary: they oscillate around a relatively constant mean (often near zero) with more stable variance[1][2]. By predicting returns, the model avoids the trivial solution of outputting a slowly-rising price curve and instead must learn genuine patterns in the data. This is why most financial forecasting models work in terms of returns (or log-returns) rather than raw prices.

- **Why 5-day returns?** Aggregating to 5-day forward return (essentially a weekly return) smooths out some daily noise and might capture short-term trends better. It can reduce the random day-to-day fluctuations and emphasize more meaningful movements. Using a 5-day horizon also aligns with a slightly longer decision period (trading on a weekly outlook rather than daily micro-noise). This is reasonable if the goal is to predict a near-term trend. If the model's performance is good with 5-day returns, that's a valid target. You could also experiment with **1-day returns** versus **5-day returns** to see which yields better predictability or more useful signals. Often, a slightly longer horizon (like 5D) can be easier to predict than daily jumps, but too long a horizon dilutes the signal – so 5 days is a fair compromise to start with.

- **Direct price prediction?** Not recommended in this context. If we predicted prices, we would have to deal with each asset's scale (Apple at \$150 vs. S&P500 at 4,500, etc.) and strong autocorrelation. The model would likely learn the non-informative strategy "tomorrow's price ≈ today's price" and achieve low error without truly forecasting changes. In fact, in a random-walk efficient market, the best forecast for tomorrow's price *is* today's price – which is why evaluating on price level can be misleading. By differencing into returns, we remove that dominant trivial signal and force the model to find what makes returns deviate from zero. In short, sticking with returns as the target is the right approach for a learning algorithm[2]. If needed, one can always convert predicted returns back to price forecasts, but training on returns is more robust.

- **What about drawdown as a target?** Predicting **drawdowns** (the peak-to-trough loss over some period) is an unusual choice for a direct model output. Drawdown is a non-linear, path-dependent metric – it's not a single-day value but depends on a sequence of future returns. Forecasting it would be much more complex and is

typically not done in standard models. Instead, to manage drawdown risk, one would forecast returns and perhaps volatility, then derive expected drawdowns or use scenario analysis. If your goal is to control risk or avoid large drawdowns, a better approach is to have the model predict returns (and maybe volatility) and then use those to estimate the probability of a drawdown exceeding X%. In summary, **stick with returns** as the model's prediction target. They are easier for the model to learn, make the time series stationary[1], and any drawdown-related decisions can be made by analyzing the predicted return distribution. Drawdowns could be a **derived metric** you compute from the model's multi-step forecasts, rather than something the model outputs directly.

**Bottom line:** 5-day forward returns as used in the notebook are appropriate. It leverages stationarity and removes trivial upward drift. There's no strong reason to switch to price levels. Only consider alternative targets if you have a specific reason – e.g., if you find the model is having trouble with returns, you could try log-price changes or even classification (up/down) as a target, but typically returns are the standard in financial forecasting. Drawdown prediction should not be a primary target here.

## Impact of Number of Tickers (138 Assets)

Using **138 tickers (assets)** in the model provides a very rich cross-section of the market. There are pros and cons to having so many assets:

- **Information Diversity:** On the upside, more tickers means more information. Each asset is a node in the graph, and including a broad set of assets (indices, ETFs, stocks, commodities, etc.) lets the model capture inter-market relationships. For example, signals from bond markets or gold may help predict stock index moves, and including many stocks can let sectors "talk" to each other. If these 138 tickers cover all major sectors and asset classes, the model essentially sees the whole "weather system" of finance, which fits the idea of a holistic market forecast. Important events (say, a tech crash) will manifest in tech stock nodes and propagate to others via the graph. In this sense, **more tickers can be helpful** – the model isn't missing parts of the network.

- **Noise and Redundancy:** However, adding tickers that are *redundant* or *irrelevant* can introduce noise. If many tickers are highly correlated with each other (for example, multiple S&P500 sector ETFs that all mostly move together, or many individual stocks that just mirror the index), then the model is seeing the same signal multiple times. That isn't harmful per se (it could reinforce the pattern), but it does increase computational load and the risk of overfitting. Similarly, if some of the 138 assets have poor data quality or very idiosyncratic behavior unrelated to the broader market, they might act as noise. The graph message-passing could carry some "random" signals from those into others.

- **Model Complexity:** A graph with 138 nodes is manageable for GNNs (the notebook uses a KNN-based adjacency with about 3 connections per node, so roughly ~138×3 edges). That's not too dense, so computation is fine. Training 138 time series at once is also fine – the dataset had 154k training windows, which the model handled. So from a **capacity standpoint**, 138 is okay. But keep in mind, every additional asset adds parameters in the embedding layers (the model has an asset embedding vector for each asset) and potentially increases the complexity of patterns to learn. If some assets don't add new information, the model might waste some capacity trying to fit their noise.

**Recommendation:** The current number (138) seems to have been chosen carefully to cover a wide range (the printout shows all assets have full date coverage). It's likely a good comprehensive set. I would **not arbitrarily decrease** the number of tickers unless you have evidence that a smaller subset yields similar performance. In fact, if there are important assets missing, one could even consider *increasing* it – but 138 already covers a lot. Quality matters more than quantity: ensure the assets included are all meaningful and have sufficient history. If you suspect redundancy, one approach is to do ablation tests (e.g., remove a cluster of very similar ETFs or stocks and see if performance drops). If it doesn't, that tells you those were not contributing unique info.

That said, having some **duplication of signals** isn't terrible – it can make the model more robust. For example, including both SPY and ^GSPC (S&P 500 ETF and index) is redundant, but if one has missing data on a day, the other might fill in, etc. The code's use of fallback aliases and backups suggests the aim was to have a full set of assets with continuous data. So 138 is fine.

In summary, **more tickers = more coverage**, which is in spirit good for a "global" model. Just be mindful that each ticker should justify its presence. If you wanted to simplify, you could focus on key benchmarks (a few indices, major commodities, major currencies, etc.), but then you'd lose granularity. Given computing power today, 138 series is not too many. Keep it as is, unless you identify specific assets that are hurting performance (e.g., extremely volatile outliers or very low-volume assets that behave erratically). The current set seems well-curated (they even excluded Oil in one option due to perhaps data issues). So, **no strong need to decrease the number of tickers** – the variety is likely helping capture systemic moves.

## Number of Experts in the MoE (Regime Models)

The mixture-of-experts (MoE) architecture is meant to handle **different market regimes**. In theory, each "expert" corresponds to a distinct regime or behavior pattern in the market. The high-level concept described two primary experts: a **"Risk-On" (bull market) expert** and a **"Risk-Off" (bear or crisis) expert**, which aligns with common regime-switching approaches.

In the implementation, it looks like the code allowed for `NUM_EXPERTS = 8` (meaning up to 8 experts), but only explicitly pre-trained Expert 0 on risk-on data and Expert 1 on risk-off