Absolutely! Let me break down your **LSTM model architecture** and how it works step-by-step in your code:

**What is LSTM?**

* **LSTM (Long Short-Term Memory)** is a special type of Recurrent Neural Network (RNN) designed to learn from sequential data, like time series.
* It remembers information over long sequences, solving the problem of standard RNNs forgetting early inputs.
* Great for financial data because price/macro indicators have temporal dependencies.

**Your model is a multi-input LSTM with two branches:**

1. **Technical indicators branch** — learns patterns from technical features (price, volume, indicators).
2. **Macroeconomic indicators branch** — learns from macro data (economic reports, surprises, forecasts).

Both branches process sequences over the past time\_steps (e.g., 30 time steps = 30 candles/periods).

**Architecture summary (build\_multi\_input\_5\_layer\_lstm):**

| **Layer type** | **Description** | **Output shape** |
| --- | --- | --- |
| Input (tech) | Input for technical features: (time\_steps, n\_tech) | (None, time\_steps, n\_tech) |
| LSTM 1 (tech) | 128 units, returns sequences | (None, time\_steps, 128) |
| LSTM 2 (tech) | 128 units, no sequences (outputs last step) | (None, 128) |
| Dense (tech) | Fully connected layer with ReLU | (None, 128) |
| BatchNormalization (tech) | Normalize outputs for stable training | (None, 128) |
| Dropout (tech) | Randomly drops 50% neurons (regularization) | (None, 128) |
|  |  |  |
| Input (macro) | Input for macro features: (time\_steps, n\_macro) | (None, time\_steps, n\_macro) |
| LSTM 1 (macro) | 64 units, returns sequences | (None, time\_steps, 64) |
| LSTM 2 (macro) | 64 units, no sequences | (None, 64) |
| Dense (macro) | Fully connected layer with ReLU | (None, 64) |
| BatchNormalization(macro) | Normalize outputs | (None, 64) |
| Dropout (macro) | Dropout 50% neurons | (None, 64) |
|  |  |  |
| Concatenate | Join tech and macro branches | (None, 192) |
| Dense (output) | Final classification with softmax (3 classes) | (None, 3) |

**How data flows:**

1. The **technical input** sequence goes through two stacked LSTM layers (which extract temporal features), then a dense layer to refine those features, batch normalization, and dropout to regularize.
2. The **macro input** sequence goes through a smaller stacked LSTM branch, dense, batch norm, and dropout.
3. Outputs of both branches are concatenated into a single vector representing combined learned patterns.
4. The final dense layer with softmax outputs probabilities for 3 classes: **neutral (0), down/sell (1), up/buy (2)**.

**What happens during training?**

* The model receives many sequences labeled with the future movement (0, 1, or 2).
* It tries to predict the correct class by minimizing the **sparse categorical cross-entropy loss**.
* Optimizer (Adam) updates weights based on prediction error.
* Dropout and batch norm help prevent overfitting and speed up convergence.

**Why two branches?**

* Technical indicators and macro data have different characteristics and time dynamics.
* Separate LSTM branches allow the model to learn specialized representations for each data type before merging.
* This improves overall prediction performance.

**What you provide:**

* X\_tr\_tech\_s: shape (samples, time\_steps, n\_tech) — sequences of scaled technical features.
* X\_tr\_macro\_s: shape (samples, time\_steps, n\_macro) — sequences of scaled macro features.
* y\_train: labels for training sequences.

**What you get:**

* The model outputs probability distribution per sequence across 3 classes.
* You pick the class with highest probability as the predicted label.

**Summary:**

* Your LSTM model is a **two-stream sequential learner** combining technical and macro sequences.
* It learns complex temporal patterns independently in each stream, then combines for classification.
* Regularization layers like dropout and batch norm improve generalization.
* Softmax output gives probabilities for each class to classify price movement.

If you want, I can help you visualize this model architecture or explain any layer or concept deeper!