- To summarize the contents of the document, the performance of models using LSTM and transfer learning (incremental learning) was below expectations. When training an LSTM model with data from numerous features, long-term predictions (over 20 days) showed somewhat acceptable results (up to 70%), but the prediction results for 1 or 2 days ahead were disappointing.
- Therefore, a different approach was needed. Based on the thought that short-term predictions might not require numerous features, I decided to proceed using only index information and its derivative data (5, 20, 60, 120, 240-day lowest, highest, average prices, rate of return), along with trading volume information and its derivative data.
- This new approach was named 'transfer-LSTM-DNN'.
- After training a DNN model with a total of 44 features, including index and volume information plus their derivatives, the 1-day prediction showed a hit rate of up to 67% over a year, and the 20-day prediction showed a rate of up to 80%.
- The training method was also changed. Instead of using all the given train data, only a random portion was used for training, while another portion was used for validation.
- This process was repeated continuously (100-2,500 times) until the validation results were optimal. The model exhibiting the best validation result was then selected for testing.
- Consequently, for practical application in actual trading, I decided to abandon LSTM, which is burdensome in terms of both data and model structure, and further develop the DNN model that uses only price + volume + derivative information.

(Description of Diagram)

- A top box labeled "Train" represents the entire training dataset.
- An arrow points from the "Train" box to a smaller box below it labeled "Real train data". Text next to the arrow reads: "Use only 50% of the Train data, selected randomly."
- An arrow points from the "Real train data" box to an even smaller box at the bottom labeled "validation data". Text next to this arrow reads: "From this, 20% is randomly selected to be used as validation data."

- Here is a detailed explanation of the learning algorithm strategy used.
- First, the entire given dataset is divided as follows.

- After organizing the data this way, the model is trained using the 'real train data', and the trained model's accuracy is then validated using the 'validation data'.
- This process is repeated continuously to select the model with the best validation result.
- What is the reason for doing this?
- Serial data related to predictions contains a lot of noise. Even in identical situations (data), results can differ due to unexpected variables. While this is a natural phenomenon, from a data analysis perspective, it must be treated as noise.
- Data can be formed as if incompatible situations coexist simultaneously, much like quantum superposition. The learning process is a complete function approach. A neural network algorithm probably cannot resolve a phenomenon like quantum superposition.
- Therefore, the best method available to us is to eliminate one of these superimposed data points.
- Even if input data are not perfectly identical, if they share over 90% similarity and have the same target, this can have the same effect as a superposition phenomenon.
- It would be ideal to have a clean algorithm to separate the superimposed data (I have not yet found one), but failing that, the method described above is used as a secondary choice.
- The belief is that if a model trained on randomly selected data shows excellent validation results, it is because the noise has been successfully removed.

- Up to this point, the discussion has been about the prediction system. Now, we must consider what more is needed to use it for actual trading.
- Here is a list of problems that arise in actual trading.
- Problems of manual trading: It is difficult to trade properly when the trading and prediction systems are disconnected. The process involves downloading data, training and predicting, and trading based on prediction results. Disconnections can occur at various stages of this process for several reasons.
- Therefore, a system that can automatically handle this entire process smoothly, like flowing water, is necessary.
- Need for risk management: Even with a system that predicts and trades daily, the nature of futures means one could get a margin call within a single day.
- In other words, you can lose everything. A 100% loss rate is possible. Of course, one can trade with a pre-set stop-loss, but it is difficult to

determine an appropriate stop-loss limit, and frequent stop-losses will act as a negative factor that damages overall returns.

- I considered trading on an hourly basis (60-minute bars) rather than a daily basis.
- When trading hourly, predicting the next hour at every hour raised doubts about the accuracy and credibility of the predictions.
- Therefore, I created a high/low point model where the target was set to determine if the current time is a high point or a low point within the next 5 to 40 hours. Based on the target type, various models were generated.
- To give a few examples:
- Target Setting Method
    - 1: The current closing price is a high point within 'n' hours compared to the (closing price, high price).
    - 2: The current closing price is a low point within 'n' hours compared to the (closing price, low price).
    - 0: A state that is neither a high point nor a low point.
- Model Type Setting Examples
    - 5C: The target is set by judging if the current hour's closing price is a high or low point based on the closing prices within the next 5 hours.
    - 10HL: The target is set by judging if the current hour's closing price is a high or low point based on the (high price, low price) within the next 10 hours.
    - 40P: The target is set by a simple comparison between the closing price of the current hour and the closing price after 40 hours.

- Using the API provided by Kiwoom, three automated trading systems were developed that receive signals every 60-minute bar. For reference, 'Algorithm Trading with Python,

  http://wikidocs.net/book/110' was consulted.

- Algorithm Trading 2: This is a 60-minute bar automated trading system developed using the high/low point technique described earlier. Every hour, it automatically downloads data (price and volume information) from the Kiwoom API, preprocesses it by adding derivative information such as the highest, lowest, average, rate of return, and rate of change for the past 1-240 bars, and then trades automatically based on the ensemble predictions of pre-trained models.
- The individual programs are listed and explained below.
    - make_model: Trains and tests the models.

    - **ensemble_test:** Tests the ensemble predictions and profits for 'n' models.
    - **ensemble:** The ensemble prediction program used for automated trading.
    - **profit:** Simulates the trading profit based on the predictions of an individual model or an ensemble model.
    - **futureTrader60M_ensemble:** An automated trading program using the Kiwoom API. The UI is implemented using the PyQt5 library. A 32-bit version of Python is required to use the Kiwoom API. Every hour, it automatically runs the ensemble model and trades according to the prediction results, liquidating all positions 5 minutes before the market closes.
    - **futureTrader60M_ensemble2:** Whereas `futureTrader60M_ensemble` relies on only one of the ensemble prediction versions (2, 4, or new), `futureTrader60M_ensemble2` uses the ensemble predictions from all three systems (2, 4, and new) for trading. Therefore, the trading volume for each individual system is allocated as 1/3 of the total tradable amount, with the intention of averaging the trading results of the three systems.

- **Algorithm Trading 4:** The difference from Algorithm Trading 2 lies in the input data composition. ADX and DMI information are added.
- **Algorithm Trading new:** Stochastic Slow information is added to the Algorithm Trading 4 input data. Also, unlike versions 2 and 4, it does not distinguish between model types C, HL, and P. Instead, it simply compares the current closing price with the closing price after 'n' bars (1-10). If it's higher, it's considered a high point; if lower, a low point. It generates a sell signal at a high point and a buy signal at a low point.
- **Automated trading batch files executed from the command line (cmd):**
    - `futureTrader60M_ensemble.bat`:
    - `futureTrader60M_ensemble.bat2`
- **Summary of the Trading Process:**

    1. Before 10:00 AM, execute

       `futureTrader60M_ensemble.bat2`.

    2. Clicking the program execution confirmation will trigger auto-login, the program will run in the command window, and the trading monitor window will appear.
    3. The monitor screen displays the current deposit status, profit/loss status of held positions, etc.

4. The screen also supports manual trading.
5. At 10:00 AM, data is automatically downloaded from the Kiwoom API, the data file is updated, preprocessed, and saved, and the ensemble model makes a prediction.
6. Based on the ensemble prediction, trades such as new orders, additional orders, or liquidating and then placing new orders are processed automatically.

---

7. The process from 1 to 6 occurs once every hour at the 0-minute mark. [cite: 90]
8. Every minute, a stop-loss check is performed. If an index fluctuation of more than 1% from the purchase price occurs, the position is liquidated. [cite: 91]
9. At 3:30 PM, it trades Dollar Futures, Kakao, and Samsung Electronics stocks (on a daily basis). [cite: 92] The data, prediction, and trading are similar to KOSPI futures but are done on a daily basis. [cite: 93] These trades are not executed in reality but are recorded virtually in a trading ledger. [cite: 94]
10. After the market closes, the data for the last hour is downloaded and saved. [cite: 95]

- Each system uses a

  self-reflection technique for ensemble prediction. Self-reflection is a self-correcting adjustment system, meaning it does not fully trust the ensemble results of the trained models.

- Trained models are bound to make predictions based on past data. The stock market, by its nature, frequently undergoes paradigm shifts, so relying on a single model or a single group of models is risky.
- Attempting to reflect these paradigms by frequently retraining models can have the adverse effect of increasing noise.
- A simple solution is to observe the prediction models for a certain period to evaluate their accuracy. If the evaluation result is below expectations, a prediction value opposite to the model's prediction is generated.
- In sports like soccer, one can often see a team trying the opposite strategy if their initial strategy doesn't work well.
- Of course, the model or the ensemble itself can be switched to another one based on periodic evaluation. This would be done manually based on human intuition.

- The self-reflection described previously is embedded in the ensemble prediction process and is handled automatically.
- Therefore, it is a method that can efficiently respond to temporary environmental changes while maintaining the long-term reliability of the model.
- Self-reflection is implemented in a module called

  `make_reinfo`.

- There are modules like `make_reinfo_profit_comp` and `make_reinfo_updown` that reinforce prediction results by observing the trading profit from model predictions. However, since the accuracy of a model's prediction signals does not always align with trading profit, there is a lack of consistency.
- The problem with applying these modules is that the probability of a model selected based on a previous run showing similar results in the future is much lower than that of the original model. Therefore, I decided to abandon the application of these modules.
- The following is a graph of the return rate from Kiwoom mock investment. (From 2023-01-02, it is actual investment).

(Description of Graph)

- Title: 2022/01/20 ~ 2023/01/12/13:03 Return: 229.62%
- Y-axis: Return
- Blue Line: Ensemble strategy of 3 deep learning models
- Orange Line: Futures index fluctuation with lev. 13.3
- The graph shows the blue line (ensemble strategy) achieving a cumulative return of over 3 (or 300%), while the orange line (futures index) fluctuates and ends with a negative cumulative return of around -1.5 (or -150%).

- Base Models for Algorithm Trading
  - Algorithm Trading 2: [5C, 5HL, 5P, 10C, 10HL, 10P, 15C, 15HL, 15P, 20C, 20HL, 20P, 25C, 25HL, 25P, 30C, 30HL, 30P, 40C, 40HL, 40P]

    (Note: 5C is listed twice in the original source)

- Algorithm Trading 4: [5C, 5HL, 5P, 10C, 10HL, 10P, 15C, 15HL, 15P, 20C, 20HL, 20P, 25C, 25HL, 25P, 30C, 30HL, 30P, 40C, 40HL, 40P]

  (Note: 5C is listed twice in the original source)

- Algorithm Trading new: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- Method for Selecting the Model to Apply in Algorithm Trading

  1. The base models are trained every 15 days.
  2. The recent 1-year returns of all possible ensembles (of 3 models) from the trained base models are compared to select the best 10 ensembles. At this time, the configuration variables for the ensembles are set up with the same variables from the previously applied ensemble.

(pred_term, reinfo_th, losscut, profit_cut...) - see create_all_termensemble.py [cite: 130, 131]
3. For the 10 ensembles selected in step 2, using values around the existing configuration variables (pred_term, reinfo_th), the returns and standard deviations for the periods 2019-2021 and 2022-present are extracted. [cite: 132, 133] Only those where the annualized compound return for 2022-present is better than the annualized average for 2019-2021 are chosen, and the two values are averaged. [cite: 134, 135]
4. The best 5 from step 3 are selected to evaluate their recent returns (30-bar, 7-bar). [cite: 136]
5. Among those with good 30-bar returns, the one showing a significant recent inc