Comparative Analysis of House Price Prediction Models Using Machine Learning Techniques

Anonymous CVPR submission

Paper ID *****

Abstract

The study aims to compare the performance of different machine learning models in ***predicting Shenzhen housing prices****. We utilized* ***Convolutional Neural Networks (CNN)****,* ***Recurrent Neural Networks (RNN)****and* ***Fully Connected Neural Networks (FCNN)****,* ***Transformers*** *to analyze a housing price dataset containing multiple features. Cross-validation was employed to evaluate the generalization ability of each model, and a fixed-size test set was used for final assessment. The experimental results demonstrate that different models perform variably in handling nonlinear and complex data relationships. Numerical results and visual charts are presented to show the prediction accuracy, training errors, and test errors of each model. The study concludes with an analysis of the best-performing model for housing price prediction and discusses its strengths and weaknesses. This research provides empirical evidence for selecting the most suitable model for housing price prediction.*

In our group course project, we only used the Transformer model for prediction, whereas my paper utilized various models for prediction and conducted inter-model evaluations.

1. introduction

1.1. Background

House price prediction [1] is a critical issue within the real estate market, holding significant importance for government entities, real estate developers, investors, and the general public [2].

Accurately forecasting house price trends not only guides decision-makers in making informed policy and investment decisions but also assists individuals in effectively planning their housing purchases and investment strategies.

With the increasing complexity and volatility of the real estate market, traditional empirical and intuition-based methods no longer suffice to meet the demand for precise predictions.

Therefore, leveraging advanced machine learning models for house price prediction has become a current research hotspot and necessity.

1.2. Research Status

House price prediction models have evolved from traditional economic models such as ARIMA and linear regression to modern machine learning models like **Convolutional Neural Networks (CNN)**, **Recurrent Neural Networks (RNN)**, and **Transformer models**. Traditional models forecast based on trends in historical data, while modern models leverage complex nonlinear relationships and time-series data, significantly improving prediction accuracy. **Ensemble learning methods** combine the results of multiple models to enhance prediction stability and accuracy. Future research directions include model fusion, optimization of feature engineering, and the application of cross-domain data.

1.3. Research Significance

The significance of this research lies in its potential to enhance the accuracy and reliability of house price predictions [9]. By systematically comparing advanced machine learning models, this study aims to identify the most effective approaches for different market conditions and data characteristics. The findings can inform stakeholders about the most suitable models for their specific needs, contributing to more informed decision-making in the real estate market.

Furthermore, this research can guide future developments in model fusion and feature engineering, paving the way for more robust and adaptive prediction systems.

1.4. Objectives of the Study

The objectives of this study are to compare and analyze the performance of different machine learning models in the task of house price prediction, exploring their differences in prediction accuracy, stability, and interpretability. Specific objectives include:

1. Evaluate the predictive performance of various models on different datasets.

2. Analyze the responsiveness of different models to fluctuations in house prices.

3. Investigate the strengths and limitations of each model under specific conditions.

2. Dataset

2.1. Data Source

The dataset for this study is sourced from Google, providing comprehensive and up-to-date information on house prices.
https://docs.google.com/spreadsheets/d/1Oi7h3tbzSP5moH0b_yTzK2YNoo83andv/edit?usp=drive_link&ouid=100546262440119773541&rtpof=true&sd=true.

2.2. Data Features

Table 1. Statistics of Each File

File Name	Amount	Average per_price
szfj_baoan.xls	1251	6.64w
szfj_dapengxinqu.xls	736	3.26w
szfj_futian.xls	1264	9.16w
szfj_guangming.xls	2073	4.63w
szfj_longgang.xls	1472	4.28w
szfj_longhua.xls	1950	6.17w
szfj_luohu.xls	3299	6.29w
szfj_nanshan.xls	2403	10.66w
szfj_pingshan.xls	2574	3.57w
szfj_yantian.xls	1492	5.04w

2.3. Preprocessing

- **Data Integration:** Combine housing price information from various districts into a single xlsx file.
- **Data Cleaning:** Replace missing values with the average and address any input errors. Remove duplicate values.
- **Encoding Categorical Variables:** Utilize one-hot encoding to convert categorical features into numerical representations.

2.4. Data Statistics (processed)

- **Total Entries**: The dataset contains 16,450 entries.
- **Missing Values**: Zero.
- **Correlation Analysis** [6]: A matrix, as shown in Figure 1 , displaying the correlation between features and the target variable, indicating which features most predict house prices.

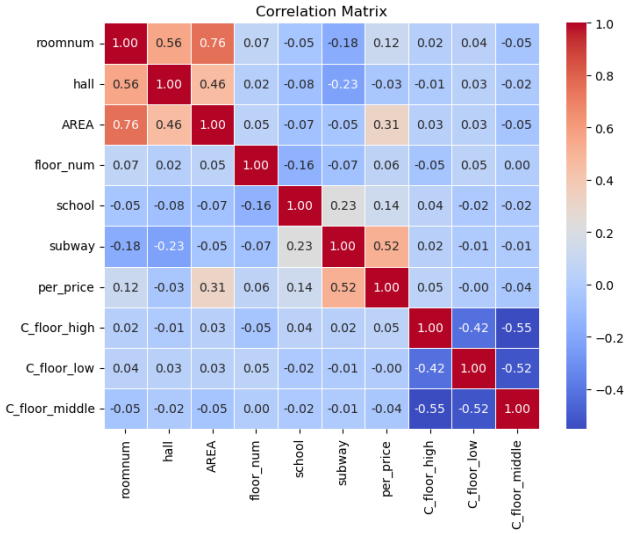

Figure 1. Correlation coefficient matrix

3. Methodology

3.1. Data Splitting

Data is split into training, validation, and test sets.

- **Training set**
  - **Purpose**: Used to train the model. During the training process, the model's parameters are optimized and updated based on the data in the training set, allowing the model to learn and fit patterns from the data.
- **Validation set**
  - **Purpose**: Used to tune hyperparameters and evaluate the performance of different models. During training, the performance on the validation set is used to select models and adjust hyperparameters to improve the model's generalization ability.
- **Test set**
  - **Purpose**: Used to evaluate the performance of the final selected model in a real-world scenario. The test set is the last step in model evaluation, used to estimate the model's predictive ability in future practical applications.

3.2. K-fold Cross-Validation

To ensure robust evaluation, K-fold cross-validation, as shown in Figure 2, is employed. The dataset is randomly partitioned into K subsets (folds), and each model is trained K times, using K-1 folds for training and the remaining fold for validation. This process is repeated K times, with each fold used exactly once as the validation data.

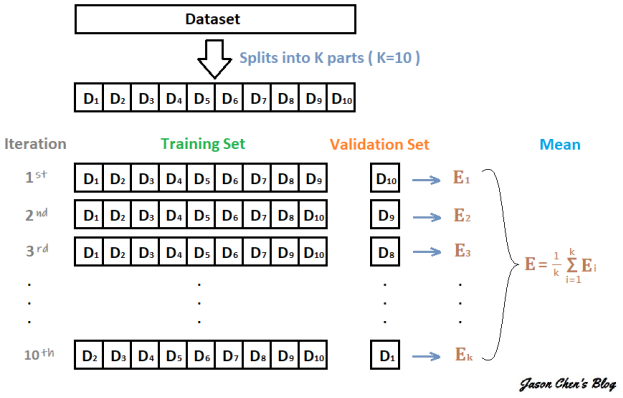

Figure 2. K-fold Cross-Validation diagram

3.3. Model Selection

3.3.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed to handle data with grid-like topology, such as images. CNNs extract local features through convolutional layers and reduce dimensions through pooling layers. A typical CNN structure, as shown in Figure 3, includes an input layer, multiple convolutional layers, pooling layers, and fully connected layers. The convolution operation captures local spatial information from the input data, making CNNs particularly suitable for image classification and recognition tasks. In house price prediction, CNNs can be used to process features with spatial dependencies, such as geographical location-related data.

Basic Principles:

- **Convolutional Layer**: Applies multiple filters (convolutional kernels) to the input data, sliding over it to extract local features through dot products.
- **Activation Function**: Usually employs ReLU (Rectified Linear Unit) to introduce non-linearity.
- **Pooling Layer**: Reduces data dimensions through max pooling or average pooling, retaining the main features.

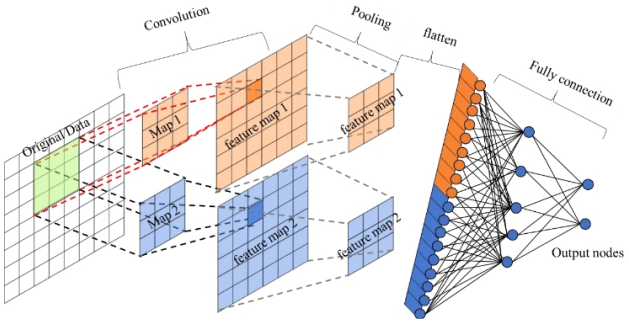

Figure 3. Convolutional Neural Network structure

- **Fully Connected Layer**: Connects all neurons, achieving global feature combination for the final classification or regression task.

3.3.2 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data [5]. RNNs capture temporal dependencies in data by maintaining a hidden state within the network. The key characteristic of RNNs is their recurrent structure, as shown in Figure 4, where the output at each time step depends on the output of the previous time step and the current input. In house price prediction, RNNs can be used to process time-series data, such as historical housing price records.

Basic Principles:

- **Recurrent Units**: Each recurrent unit receives the current input and the hidden state from the previous time step to compute the current hidden state.
- **Hidden State Update**: Updates the hidden state using weight matrices and activation functions like tanh or ReLU.
- **Output Layer**: Typically uses a fully connected layer to map hidden states to the output space.
- **Sequence Processing**: Can handle variable-length input sequences, performing forward propagation and backpropagation through time steps.

3.3.3 Fully Connected Neural Network (FCNN)

Fully Connected Neural Networks (FCNNs) [3] are the most basic neural network structure, as shown in Figure 5, where every neuron in each layer is connected to every neuron in the next layer. FCNNs are usually used to process structured data (tabular data) and are implemented in the form of Multilayer Perceptrons (MLPs). Each layer's neurons perform nonlinear transformations through weighted

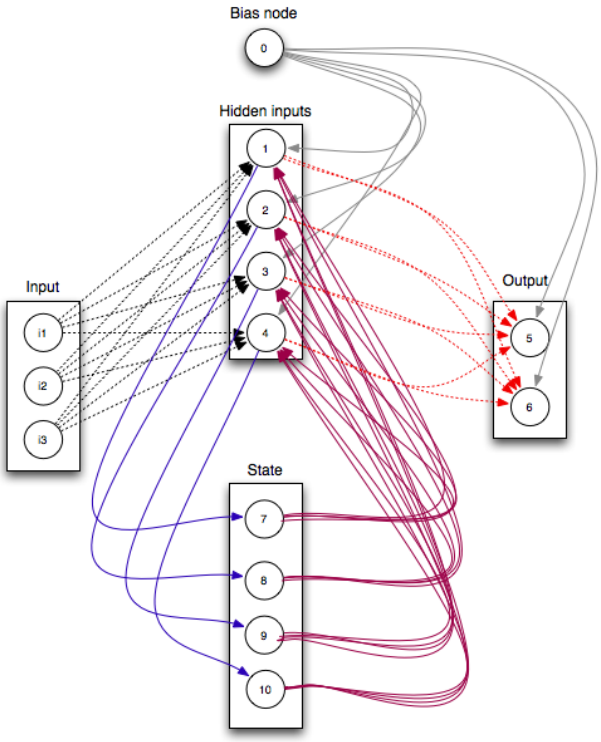

Figure 4. Recurrent Neural Networks structure

sums and activation functions, extracting high-order features from the data layer by layer.

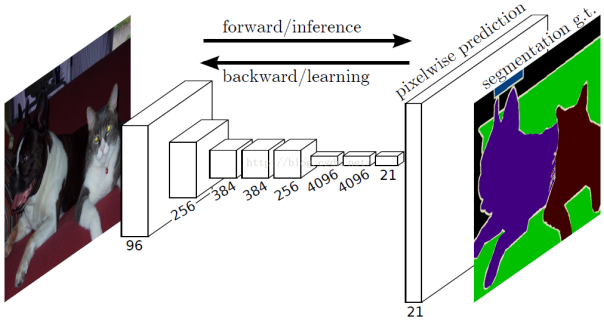

Figure 5. Fully Connected Neural Networks structure

Basic Principles:

- **Input Layer**: Receives raw data inputs.
- **Hidden Layers**: Extract features through weighted sums and activation functions like ReLU or Sigmoid.
- **Output Layer**: Outputs predictions based on the task (classification or regression).
- **Error Backpropagation**: Adjusts network weights by computing the gradient of the loss function, minimizing prediction errors.

3.3.4 Transformer Model

The Transformer model [8] is a deep learning model based on the self-attention mechanism, as shown in Figure 6, initially used for natural language processing (NLP) tasks. Transformers eliminate the recurrent structure of RNNs, significantly improving training efficiency through parallel computation. The Transformer model consists mainly of encoders and decoders, with each encoder and decoder layer containing multiple self-attention heads and feed-forward neural networks.

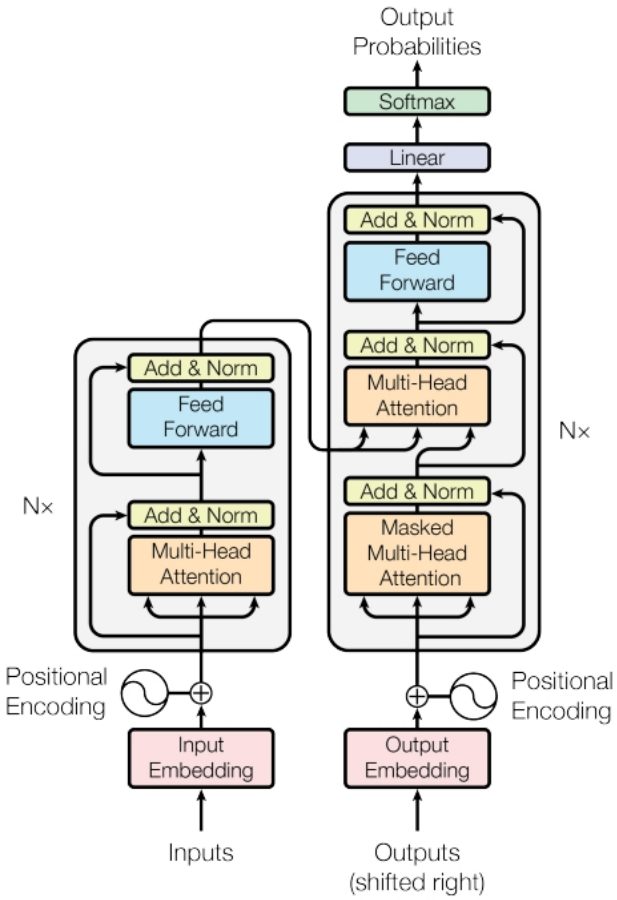

Figure 6. Transformer structure

Basic Principles:

- **Self-Attention Mechanism** [4]: Captures global information by computing correlation weights between each position in the input sequence and every other position.

- **Multi-Head Attention**: Maps input data to different subspaces through multiple linear transformations, performing parallel attention computations, and then concatenating the results.
- **Positional Encoding**: Introduces positional information through positional encoding vectors, as Transformers lack built-in sequence order information.
- **Encoder**: Each encoder layer extracts features from the input through self-attention and feed-forward neural networks.
- **Decoder**: Each decoder layer generates output through self-attention, encoder-decoder attention, and feed-forward neural networks.

3.4. Hyperparameter Adjustment

This experiment uses Grid Search, which systematically searches the specified parameter grid to find the optimal hyperparameter combination. It exhaustively searches all possible parameter combinations, particularly suitable for smaller parameter spaces.

3.5. Evaluation Metrics

To evaluate the performance of the models, we use three different loss functions: Train Loss, Validation Loss, and Test Loss. Each of these loss functions measures how well the model predicts the target values during different phases of training and evaluation.

Train Loss [7]

The Train Loss is the loss calculated on the training dataset. It is used to update the model's parameters during the training process. The formula for Train Loss is given by:

$$\text{Train Loss} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\hat{y}_i^{\text{train}}, y_i^{\text{train}}) \tag{1}$$

where:

- N is the number of training samples,
- $\hat{y}_i^{\text{train}}$ is the predicted value for the i-th training sample,
- y_i^{train} is the actual value for the i-th training sample,
- $\mathcal{L}$ is the loss function (e.g., Mean Squared Error).

Validation Loss

The Validation Loss is the loss calculated on the validation dataset. It is used to tune the hyperparameters and to evaluate the model during the training process. The formula for Validation Loss is:

$$\text{Validation Loss} = \frac{1}{M} \sum_{j=1}^{M} \mathcal{L}(\hat{y}_j^{\text{val}}, y_j^{\text{val}}) \tag{2}$$

where:

- M is the number of validation samples,
- $\hat{y}_j^{\text{val}}$ is the predicted value for the j-th validation sample,
- y_j^{val} is the actual value for the j-th validation sample.

Test Loss

The Test Loss is the loss calculated on the test dataset. It is used to evaluate the final model performance on unseen data after training is completed. The formula for Test Loss is:

$$\text{Test Loss} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}(\hat{y}_k^{\text{test}}, y_k^{\text{test}}) \tag{3}$$

where:

- K is the number of test samples,
- $\hat{y}_k^{\text{test}}$ is the predicted value for the k-th test sample,
- y_k^{test} is the actual value for the k-th test sample.

These loss functions provide a quantitative measure of the model's performance during different phases of training and evaluation, enabling us to compare the effectiveness of various models and tune them accordingly.

4. Experimental Result

4.1. Model output

Table 2. Model Results

Model	Train Loss	Valid Loss	Test Loss
CNN	1.7938	2.5387	6.5744
RNN	2.4260	2.6402	2.6458
FCNN	2.3876	2.7512	2.7594
Transformer	1.7842	2.5208	6.4260

4.2. Visualization of results

Reference Figure 7 in Page 5

4.3. Model Performance Analysis

CNN : At the end of training, CNN achieves a training loss of 1.7938 and a test loss of 2.5387. However, the final test loss of 6.5744 indicates poor performance on the test data, suggesting potential overfitting issues.

RNN : At the end of training, RNN shows stable training and test losses. The final test loss of 2.6458 demonstrates good generalization ability on the test data, with no significant signs of overfitting or underfitting.

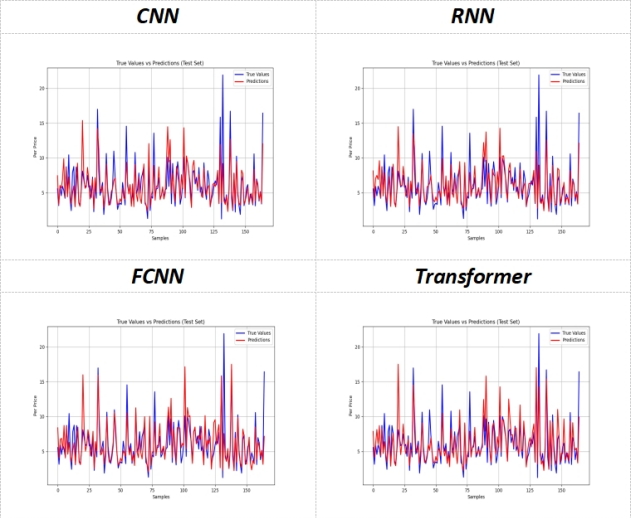

Figure 7. Visualization of results of four models

FCNN : Towards the end of training, FCNN exhibits relatively stable training and test losses. The final test loss of 2.7594, similar to RNN, indicates good generalization ability.

Transformer : Towards the end of training, Transformer maintains low training and test losses. However, the final test loss of 6.4260 suggests similar issues as CNN, potentially indicating overfitting.

4.4. Results Interpretation

Across different neural network models, we observe variations in their performance on the house price prediction task. RNN and FCNN demonstrate good generalization ability, with low and stable test losses, indicating their effectiveness in generalizing to new data post-training. In contrast, CNN and Transformer exhibit higher final test losses, suggesting possible overfitting where the models perform well on training data but poorly on test data. These results underscore the importance of selecting appropriate models and tuning hyperparameters to improve the accuracy and stability of house price prediction.

5. Discussion

5.1. Performance Analysis of Each Model

CNN (Convolutional Neural Network):
CNN demonstrates a significant difference between training and test losses, with a final test loss of 6.5744, indicating poor generalization. It suggests that CNN might have overfit the training data due to its ability to capture intricate spatial patterns in images rather than sequential data like house prices.

RNN (Recurrent Neural Network):
RNN exhibits stable training and test losses, with a final test loss of 2.6458. This suggests that RNN effectively captures temporal dependencies in sequential data, making it suitable for time-series prediction tasks like house prices. Its ability to retain memory over time contributes to its good generalization performance.

FCNN (Fully Connected Neural Network):
Similar to RNN, FCNN shows stable training and test losses, concluding with a final test loss of 2.7594. FCNN's architecture allows it to learn complex relationships in the data, making it effective for tasks where features interact nonlinearly, such as predicting house prices.

Transformer (Transformer Model):
Transformer maintains low training and test losses during training epochs but ends with a relatively high final test loss of 6.4260. This could indicate that while Transformers excel in capturing long-range dependencies in sequential data, they might require more sophisticated regularization techniques or hyperparameter tuning to prevent overfitting on smaller datasets.

5.2. Advantages and Disadvantages of Each Model

CNN:

- **Advantages:** Effective for spatial data, can capture hierarchical patterns.
- **Disadvantages:** Prone to overfitting, complex to train on non-image data.

RNN:

- **Advantages:** Good for sequential data, handles temporal dependencies well.
- **Disadvantages:** Vulnerable to vanishing/exploding gradient problems, limited memory retention.

FCNN:

- **Advantages:** Simple architecture, suitable for complex nonlinear relationships.
- **Disadvantages:** May struggle with sequential data, less effective with time-series aspects.

Transformer:

- **Advantages:** Captures long-range dependencies, scalable to large datasets.
- **Disadvantages:** Needs extensive computational resources, prone to overfitting on smaller datasets.

5.3. Factors Affecting Model Performance

Several factors influence the performance of these models:

- **Model Architecture:** Different architectures handle different types of data and relationships.
- **Dataset Size:** Larger datasets generally lead to better generalization.
- **Hyperparameters:** Proper tuning of learning rates, batch sizes, and regularization parameters is crucial.
- **Data Quality:** Clean and relevant data enhances model performance.
- **Training Duration:** Longer training times can lead to better convergence but also increase the risk of overfitting.

5.4. Insights from Visualization

Visualizations such as learning curves, feature importance plots, and error distributions provide insights into:

- **Model Convergence:** How quickly and stably the model converges during training.
- **Feature Relevance:** Which features contribute most to the predictions.
- **Prediction Errors:** Distribution and patterns of errors can highlight areas where models struggle, informing future improvements.

In conclusion, while RNN and FCNN demonstrate robust performance in predicting house prices with good generalization, CNN and Transformer show strengths and weaknesses related to their ability to handle spatial and sequential data complexities. Choosing the right model depends on the specific characteristics of the data and the desired outcome, alongside careful consideration of model architecture and tuning.

References

[1] Guangliang Gao, Zhifeng Bao, Jie Cao, A Kai Qin, and Timos Sellis. Location-centered house price prediction: A multi-task learning approach. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(2):1–25, 2022. 1

[2] Salim Lahmiri, Stelios Bekiros, and Christos Avdoulas. A comparative assessment of machine learning methods for predicting housing prices using bayesian optimization. *Decision Analytics Journal*, 6:100166, 2023. 1

[3] Harry Pratt, Bryan Williams, Frans Coenen, and Yalin Zheng. Fcnn: Fourier convolutional neural networks. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I 17*, pages 786–798. Springer, 2017. 3

[4] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 4

[5] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020. 3

[6] James H Steiger. Tests for comparing elements of a correlation matrix. *Psychological bulletin*, 87(2):245, 1980. 2

[7] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017. 5

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4

[9] Yu Zhou and Hongru Guo. Is shenzhen housing price bubble that high? a perspective of shenzhen hong kong cross-border integration. *International Real Estate Review*, 18(3), 2015. 1