Predicting Micro Business Density: A Review of Different Machine Learning Approaches*

1st Asheer Ali
MSc. Data Science
University of Erlangen-Nuremberg
Erlangen, Germany
asheer.ali@fau.de

2nd Melih Hasbi Ekinci
MSc. Artificial Intelligence
University of Erlangen-Nuremberg
Erlangen, Germany
melih.ekinci@fau.de

3nd Johannes Enk
BSc. Medical Engineering
University of Erlangen-Nuremberg
Erlangen, Germany
johannes.enk@fau.de

4nd Julia Wallnig
MSc. Computational Engineering
University of Erlangen-Nuremberg
Erlangen, Germany
julia.wallnig@fau.de

***Abstract*—The growth of microbusinesses in the United States poses a challenge for policymakers who lack access to sufficient economic data to inform their decision-making. By conducting annual surveys and gathering information on more than 20 million microbusinesses nationwide, GoDaddy's Venture Forward team has attempted to close this informational gap. Although current models based on internal and census data have the ability to guide policy decisions, they might be enhanced by more cutting-edge techniques and new information. By comparing several algorithms, this paper attempts to enhance forecasts and better educate microbusiness decision-making in order to build more equitable and resilient economies. Communities and the economy as a whole may be significantly impacted by the results of this investigation.**

I. Introduction

Although time series prediction has been studied in a variety of fields, it is still difficult to predict future series using historical observations and exogenous data from the past. Existing approaches either miss the nuanced interactions between the various exogenous variable components that may have an impact on prediction accuracy or miss the relationships between exogenous and target data. In addition, target series prediction benefits from taking into consideration the intrinsic temporal dynamics of exogenous data. So, solving these problems is essential for creating time series prediction models that are more successful. To address these issues, we propose various end-to-end deep learning models, which incorporates spatio-temporal feature extraction of exogenous variables and temporal dynamics modeling of target variables into a single framework. We carry out comprehensive empirical evaluations with various methods over a microbusiness density dataset, and compared these methods. A git repository is also made to view the results [6].

II. About the Dataset

The provided dataset includes historic economic data for US counties. The dataset includes information about the county, such as its name and state, and data on the microbusiness density, such as the raw count of microbusinesses and the density per 100 people over the age of 18. Additionally, external data sources, such as the US Census Bureau's American Community Survey, can be used to enhance the performance of our models [5].

A. What are Microbusinesses

Micro-businesses are tiny firms that offer specialized goods or services in local or niche markets. Companies provide important contributions to the economy by providing jobs and stimulating innovation, particularly in nations where they account for the majority of all firms. These companies confront unique problems, but they also have flexibility and autonomy. Understanding the density and distribution of microbusinesses is critical for fostering local economic development and supporting the growth of small businesses.

B. Why it's important to predict the density of microbusinesses

The density of micro-businesses is an important indication of a region's economic health, giving information on the number of small enterprises operating in a certain area. These companies serve as the backbone of many local economies, creating jobs and propelling economic progress. Knowing the prevalence of these enterprises in an area can help policymakers make better decisions while also attracting new businesses and investments. Furthermore, the density of micro-businesses may be used to predict major trends such as population growth, urbanization, and migration patterns. Forecasting the density of micro-businesses may also give valuable information to real estate developers, merchants, and other enterprises, allowing them to make educated decisions about where to place their operations and assisting them in growing and succeeding.

III. Method 1

The Hierarchical attention-based Recurrent Highway Network (HRHN), a complete neural network architecture, is presented in this method for forecasting future time series using

just exogenous data and past observations. The contributions of this work are three-fold:

1) We use a convolutional neural network (ConvNet) to learn the spatial interactions among different components of exogenous data, and employ a recurrent highway network (RHN) to summarize the exogenous data into different semantics at different levels in order to fully exploit and model their temporal dynamics.
2) We propose a hierarchical attention mechanism, performing on the discovered semantics at different levels, for selecting relevant information in the prediction.
3) Extensive experimental results on three different datasets demonstrate that the proposed HRHN model can not only yield high accuracy for time series prediction, but also capture sudden changes and oscillations of time series.

A full diagram of the suggested system can be found in Figure 1. Both a ConvNet, which learns spatial correlations between various exogenous data components, and a RHN, which models the temporal connections among the convolved input features at various semantic levels, are included in the encoder module.

In order to choose the pertinent multi-level semantics from the encoded data, a novel hierarchical attention method is developed in the decoder module. The final forecast is then made by utilizing the interactions between historical observations and exogenous data, another RHN is employed to capture long-term temporal relationships among historical observations [1].

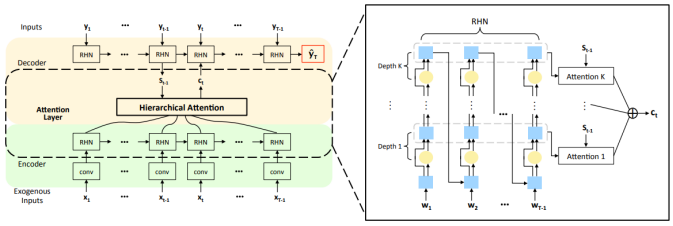

Fig. 1. A graphical illustration of HRHN. In the encoder, a ConvNet extracts the "spatial" information of the exogenous inputs $x_1, x_2, \ldots, x_{T-1}$. Then an RHN learns the temporal dynamics from the representation vectors. Using a hierarchical attention mechanism, the decoder selects the most relevant spatio-temporal features of exogenous data and leverages another RHN to capture the long-term dependencies of target series $(y_1, y_2, \ldots, y_{T-1})$ and produce the future prediction $\hat{y}_T$. The bottom box is an illustration of the hierarchical attention approach. The encoder RHN reads the convolved features $(w_1, w_2, \ldots, w_{T-1})$ and models their temporal dependencies at different semantic levels. Then the hierarchical attention mechanism computes the soft alignment of hidden states in each layer. The context vector c_t that feeds into the decoder RHN is obtained by concatenating all attentions.

A. Performance Evaluation

To determine the effectiveness of the algorithm, a machine learning model's performance must be evaluated. In order to provide a more thorough knowledge of the machine learning model's performance, different metrics were used in this study to assess its performance.

First, the average squared difference between the predicted and actual values was measured first by calculating the Mean Squared Error (MSE). In our situation, the calculated MSE of 2.821555 is comparatively low, indicating the model's predictions are quite near to the actual data. A lower value of MSE suggests higher performance.

Next, the average absolute difference between the predicted and actual values was then measured using the Mean Absolute Error (MAE) formula. A lower MAE value also denotes higher performance. Our model's MAE, which is similarly quite low at 0.32049605, shows that the model's predictions are quite close to the actual data.

In order to assess the model's precision, the accuracy of the SMAPE measure, which is frequently used in time series forecasting, was used. The model's predictions were accurate, with an average error of 5% between the predicted and actual values, as represented by the SMAPE value of 5% in table 1.

By charting the true and projected values, a visual examination of the model's performance was carried out to provide more information. The figure 2 displayed no discernible discrepancy between the true and anticipated numbers, indicating that the model is operating effectively.

Overall, the various criteria that were employed to assess the model's performance show that the model did a good job of projecting the target values. The accurate SMAPE value and low MSE and MAE values show that the model's predictions were very close to the measured values. The model's effectiveness was further verified by the visual examination, which showed that the predicted values were quite close to the actual values. As a result, the created machine learning model may be thought of as a practical choice for correctly forecasting the target values.

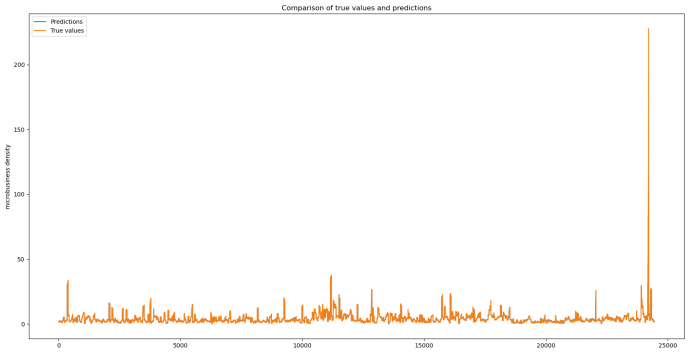

Fig. 2. True values vs Predicted values

IV. Method 2

Transformer based machine learning models can also be used to forecast time series data. Transformers based models utilize self-attention mechanisms to capture both long term and short term dynamics and patterns from time series data. [2]

Self attention mechanisms allows transformer model to process data not in a ordered manner. Instead, it processes

entire sequence and then uncover dependencies by leveraging self-attention. This is one of the advantages of transformers based models compared to sequence models. [2]

The transformer model in our work follows the architecture from (Wu et al., 2020) which also follows the original Transformer Architecture (Vaswani et al., 2017). It consist of encoder and decoder layers.

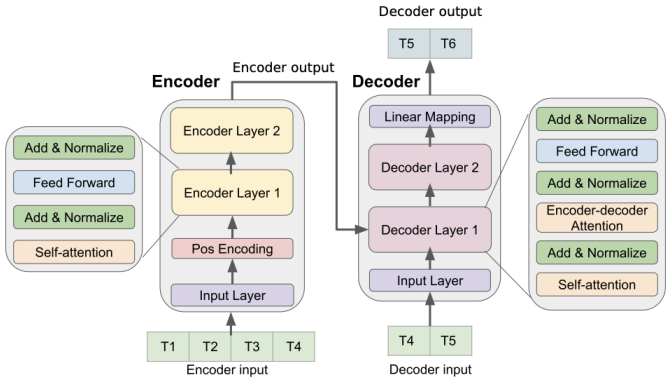

Fig. 3. Transformer-based forecasting model Architecture [2].

1) Encoder: This layer is composed of input layer, positional encoding layer, and stack of encoder layers. While input layer maps the input time series data to a vector, positional encoding is sued to encode sequential information in the time series data. Resulting vector is used in stack of encoder layers. [2]
2) Decoder: This layer is consist of input layer, stack of decoder layers and output layer. The input layer maps the decoder input to a vector. After getting through stack of decoder layers, there is an output layer which feeds the output of last decoder layer to the target time sequence. [2]

A. Performance Evaluation

We used batch size of 1000 and epoch size of 50 for the transformer model. Optimizer is selected as Adam with learning rate of 1e-3. We applied dropout techniques for regularization, dropout rate of 0.25 has been selected. Number of heads for attention mechanisms is selected as 8 while head size is 512. Number of total transformer blocks is 8. The parameters are slightly different than original paper (Wu et al., 2020) this architecture is based on.

In our study, MSE, MAE and SMAPE has selected as metric for models performance evaluation. MSE, MAE and SMAPE for our transformer model is 9.22, 0.42 and 0.1022 respectively. All of them indicating poor generalization and performance, as lower values of these metrics indicating higher performance. One can find the comparison of models' performances in table 1 at the end.

V. Method 3

[3] N-BEATS or Neural Basis Expansion Anlaysis For Interpretable Time Series Forecasting is a deep learning time-series forecasting model published in 2020. The deep neural architecture is based on very deep stack of fully-connected layers. It was probably the first pure DL work that outperformed statistical approaches on well-established datasets. In addition N-BEATS profides human interpretable outputs, that can be used in a similair way as traditional decomposition techniques.

A. Overall Structure of N-BEATS

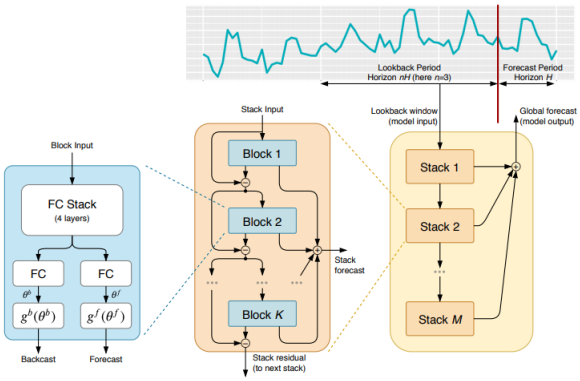

Fig. 4. N-BEATS architecture [6].

In the figure 4 one can see the architecture of the network. On the left there is the block. It is a multi-layer fully connected network. It predicts backcast, which is trying to rebuild the back horizon, and the forecast, which predicts the horizon. In the middle of the figure 4 one can see the firstly described blocks stacked together. The backcast outputs of the blocks are substracted and used in a deeper part of the model.The forecast outputs are added and form the stack forecast. On the right side of the figure 4 one can see the previously described stacked blocks, called stacks, concatinated together. Each stack has again two outputs. The stack residual, which is passed on to the next stack, and the stack forecast. The stack forecasts are summed up and result in the global forecast. The first stack, and therefore the first block, gets as input the model input, that results from a lookback period in the dataset.

B. Performance Evaluation

For this method we select with y the 'cfips'-value, for that the microbusiness density has to be predicted. In the example of figure 5, y has been set to 1007. We chose two blocks per stack and 64 hidden layers. The data set has been splitted into training and test data by a factor of ten. For the training we chose 30 epochs and a batch size of 128. In figure 5 one can see, that the model tends to overfit when reaching the 30 epochs of training. That's why the training has been stopped there. So we end with a loss of 0.0259 and a validation loss of 0.0828.

VI. Method 4

Numerous types of recurrent neural networks (RNNs) are available for working on sequential data. While earlier models

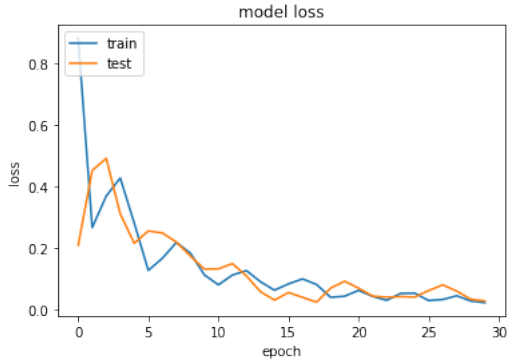

Fig. 5. N-BEATS model loss [6].

had problems with vanishing and exploding gradients, especially for longer input sequences, modern techniques can at least partially resolve those issues. The first upgrade to a basic RNN was the Long Short-Term Memory (LSTM), which introduced three gates to control the information flow. With this technology, networks could work on longer sequences while still achieving good results. The GRU followed after the LSTM to improve the long-term memory further and reduce the computation loss.

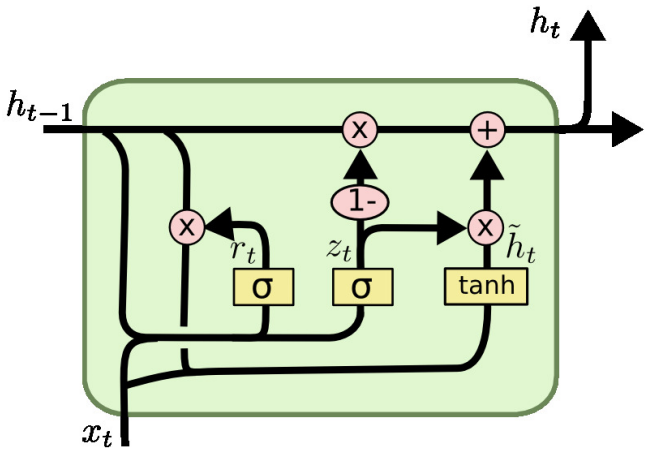

Fig. 6. Structure of a GRU cell

VII. Method 4

A GRU [4] utilizes two different gates, namely an update and a reset gate. The former determines the utilization of previous information, while the latter controls the historical data. Simple GRUs and RNNs, in general, already perform well but come with the drawback of only employing the information from the forward time series. Modern RNN models also started using backward time series data to combat this issue. Such a model makes it possible to capture past and future information.

Li et al. [4] successfully employed a bidirectional Gated recurrent Unit (Bi-GRU) for oilfield production forecasting, influencing the decision-making and investments. The input to the model was a tensor containing the number of samples, timesteps, and relevant features. Based on that input, a Bi-GRU layer extracted the implicit information and passed it on to the linear layers that made the prediction. The network also utilized dropout to reduce the models' complexity and combat overfitting and used the Adam optimizer, as it allows for smooth parameter changes. We mostly followed this description in our approach. We trained for 100 epochs and used a learning rate of 0.001 to get our results.

A. Performance Evaluation

We also relied on MSE, MAE, and SMAPE to evaluate our results for the Bi-GRU. For them, we achieved 3.172, 0.385, and 6.894, respectively. Those numbers indicate that the network can learn the general structure of the underlying data. Still, this also shows that the model might need to generalize better for unseen data. Introducing more data from other areas of the world or better normalization might have helped.

TABLE I
Comparison of Methods

Methods	Error Types		
	MSE	***MAE***	***SMAPE %***
HARHN	2.821555	0.32049605	5.71
Transformer	9.226058	0.42020046	10.22
N-BEATS	0.035692	0.108652	22.58
Bi-GRU	3.172	0.385	6.894

[a]Results achieved with different network architectures

VIII. Conclusion and Outlook

For this work, we participated in a Kaggle challenge by GoDaddy that was concerned with predicting the growth of micro-businesses in the United States to help policymakers to make proper decisions. We relied on different state-of-the-art methods based on artificial neural networks to get good predictions for time-series forecasts. We used popular machine-learning libraries like PyTorch, Keras, and TensorFlow for the implementations. Moreover, we compared the results of our solutions to determine which approach works best. Our observations showed that some models perform better than others for this particular purpose, and this discrepancy could be related to the different implementations and optimization efforts.

On Kaggle, we could observe a magnitude of other practitioners that could surpass our results, indicating that other suitable models exist. Exploring those options might be a task we could explore soon, especially because practitioners continuously develop new and improved architectures. And by combining our efforts, we could give politicians and businesspeople the correct information in the future so that they can make good decisions for us.

References

[1] Tao, Y, et al.: *Hierarchical Attention-Based Recurrent Highway Networks for Time Series Prediction.* arXiv, 201

[2] Wu, Neo et al.: *Deep Transformer Models for Time Series Forecasting: The Influenza Prevelance Case.* arXiv, 2001.08317v1

[3] Oreshkin, B. N. et al.: *N-BEATS: Neural basis expansion analysis for interpretable time series forecasting.* arXiv, 1905.10437

[4] Li, Xuechen et al.: *Time-series production forecasting method based on the integration of Bidirectional Gated Recurrent Unit (Bi-GRU) network and Sparrow Search Algorithm (SSA).* Journal of Petroleum Science and Engineering,
[5] Dataset: godaddy-microbusiness-density-forecasting
[6] asheerali/MLTS_Project