Assignment 3, Deep Learning Fundamentals, 2022
RNN for stock price prediction

Duc Nhan Do
The University of Adelaide
Adelaide, SA, 5000
ducnhan.do@student.adelaide.edu.au

Abstract

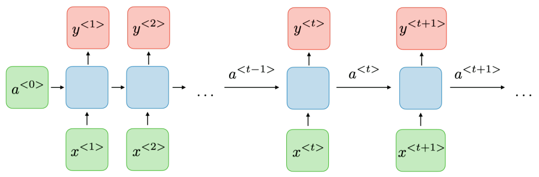
Stock market prediction is a common task that every stock expert wants to be good at so that the predicted price is as close to the actual price as possible. The successful prediction of a stock's future price could yield significant profit. The most commonly used network for stock price problems is the Recurrent Neural Network. This network can effectively extract patterns in the sequence data and take information from previous inputs to influence the current input and output, which makes it a powerful tool for time series data and stock price prediction. In this experiment, as the goal is to predict stock price and gain deeper insights into the Recurrent Neural Network, this network architecture will be self-implemented using Pytorch library and trained on the stock price data of Google. Furthermore, this network is compared to Long Short-Term Memory and Gated Recurrent Unit networks to investigate the performance differences. The evaluation is performed on three models with the same architecture complexity (except its core unit) and training configuration, to determine the best model based on the lowest loss on the validation set

1. Introduction

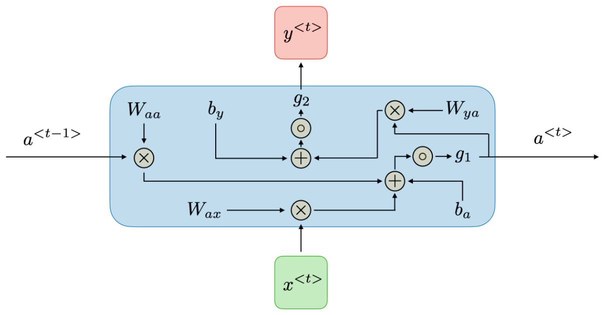
Deep learning (DL) can be considered a subset of *machine learning*. While machine learning uses straightforward concepts, deep learning works with a broader algorithm called artificial neural networks. This algorithm is inspired and designed by the structure and function of the human brain to imitate how humans think and learn. The learning process of DL can be *supervised, semi-supervised* or *unsupervised*. DL has various architectures, such as *Deep Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks* and *Transformers*. These architectures have been applied to computer vision, speech recognition, natural language processing, medical image analysis and produced stunning results which can surpass human expert performance in some cases [1].

A *stock market* is a public market that can be bought and sold shares for publicly listed companies. The stocks, also known as equities, represent ownership in the company. The stock exchange is the mediator that allows the buying and selling of shares [5]. Recently, stock price prediction has been a critical area of research and is one of the top applications of machine learning. Stock market prediction is the process of trying to determine the future value of company stock or other financial instruments traded on an exchange. The successful prediction of a stock's future price could yield significant profit. Many factors might be responsible for determining a particular stock's price, such as the market trend, supply and demand ratio, global economy, public sentiments, sensitive financial information, earning declaration, historical price and many more [2]. These factors explain the challenge of accurate prediction. However, with the advantage of modern technologies like data mining and deep learning, which can help analyze big data and develop an accurate prediction model that avoids some human errors and stock market prediction has since moved into the technological realm. Since the stock price is the *time series data* - a sequence of data points in chronological order used by businesses to analyze past data and make future predictions, the most prominent technique for this task involves the use of *Recurrent Neural Networks* since this network has the capacity to remember previous input through time.

For the above reasons and to investigate the fundamental of Recurrent Neural Network, in this experiment, a simple baseline is implemented using *Pytorch* library to predict the stock price of Google in the dataset collected from the *Alpha Vantage API*. Furthermore, the Recurrent Neural Network is compared with *Long Short-Term Memory* and *Gated Recurrent Unit* networks to find the architecture that leads to the best performance. The implementation code is available at: `https://github.com/DoDucNhan/DL-assignment.git`



(a) RNN acrchitecture.



(b) RNN block architecture.

Figure 1. The architecture of traditional RNN.

2. Methods

2.1. Recurrent Neural Network

A recurrent neural network (RNN) is a class of artificial neural networks. This architecture is robust because of its node connections, which create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. This feature also allows it to exhibit temporal dynamic behaviour that can work with sequential or time series data. In the training process, RNNs utilize training data to learn, like feedforward and convolutional neural networks (CNNs). However, this network is distinguished by having memory since it takes information from previous inputs to influence the current input and output [7]. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of RNN depends on the prior elements within the sequence. This process makes them applicable to ordinal or temporal tasks such as connected handwriting recognition, speech recognition, language translation, natural language processing (NLP), and image captioning.

The architecture of traditional RNN is described in Fig. 1. For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$
$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 activation functions. The commonly used activation functions in RNN modules are described as follow [3]:

$$\textbf{Sigmoid} : g(z) = \frac{1}{1+e^{-z}}$$
$$\textbf{Tanh} : g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$
$$\textbf{ReLU} : g(z) = \max(0, z)$$

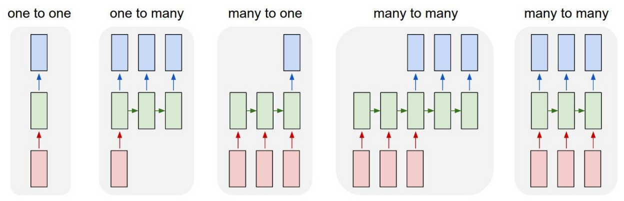


Figure 2. Different types of RNN.

Types of RNN: Normal deep neural networks map one input to one output, but RNN architecture does not have this constraint. Instead, their inputs and outputs can vary in length, and different types of RNNs are based on the number of inputs (T_x) and outputs (T_y). Different types of RNNs are usually expressed using the diagrams in Fig. 2. There are five types of RNN [3]:

1. **One-to-one** ($\mathbf{T_x = T_y = 1}$): The number of inputs and outputs is equal to 1. Traditional neural network is an example of this architecture.

2. **One-to-many** ($\mathbf{T_x = 1, T_y > 1}$): The number of inputs is equal to 1, and the number of outputs is greater than 1. This architecture can be used in *music generation*.

3. **Many-to-one** ($\mathbf{T_x > 1, T_y = 1}$): The number of inputs is greater than 1, and the number of outputs is equal to 1. This architecture can be used in *sentiment classification*.

4. **Many-to-many** ($\mathbf{T_x = T_y}$): The number of inputs is equal outputs. This architecture can be used in *name entity recognition*.

5. **Many-to-many** ($\mathbf{T_x \neq T_y}$): The number of inputs is different from the outputs. This architecture can be used in *machine translation*.

Advantages/Drawbacks:Although RNN offers many benefits in processing sequence data, it also has limitations. The advantages and drawbacks of RNN include the following:

1. **Advantages**:

   - Possibility of processing input of any length
   - Model size not increasing with size of input

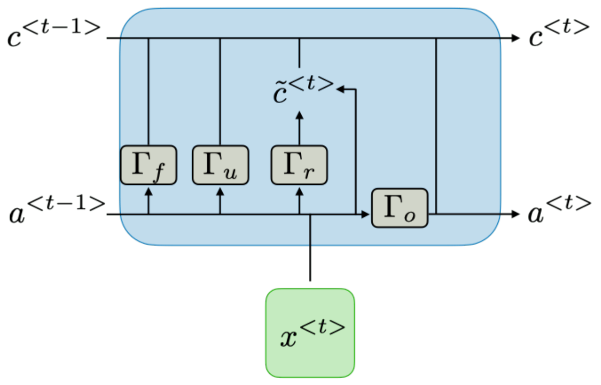


Figure 3. The architecture of LSTM unit.

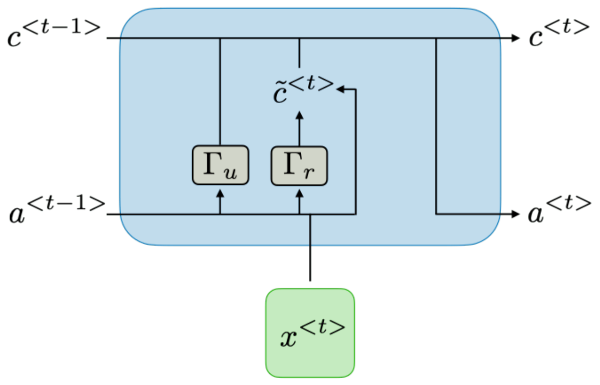


Figure 4. The architecture of GRU unit.

- Computation takes into account historical information
- Weights are shared across time

2. **Drawbacks**:

- Computation being slow
- Difficulty of accessing information from a long time ago
- Cannot consider any future input for the current state

2.2. Long Short-Term Memory

One of the appeals of RNN is the idea that it is able to connect previous information to the present task. However, when the prediction needs prior information from further away from the beginning of the sequence, the gap between the relevant information and the point where it is needed becomes very large. Unfortunately, as that gap grows, RNN becomes unable to learn to connect the information [4]. This problem is also known as the vanishing/exploding gradient. The vanishing and exploding gradient phenomena happen when a multiplicative gradient that can be exponentially decreasing/increasing with respect to the number of layers causes the challenge of capturing long-term dependencies [9, 11]. Long Short-Term Memory (LSTM) [8] is an advanced RNN designed to avoid the long-term dependency problem. The primary point making LSTM differ from RNN is the LSTM unit (Fig. 3). For each timestep t, value of *candidate cell* $\tilde{c}^{<t>}$, *cell state* $c^{<t>}$, and *activation* $a^{<t>}$ are expressed as follow [3]:

$$\begin{aligned}\tilde{c}^{<t>} &= \tanh(W_c[\Gamma_r * a^{<t1>}, x^{<t>}] + b_c)\\ c^{<t>} &= \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t1>}\\ a^{<t>} &= \Gamma_o * c^{<t>}\end{aligned}$$

$\Gamma_u, \Gamma_r, \Gamma_f, \Gamma_o$ in above expression are *update gate, reset gate, forget gate, and output gate* respectively. Update gate indicates *"how much past should matter now?"*, reset gate indicates *"drop previous information?"*, forget gate indicates *"erase a cell or not?"*, and output gate indicates *"how much to reveal of a cell?"*. All gates have the same expression as follow:

$$\Gamma = \sigma(W_x^{<t>} + Ua^{<t-1>} + b)$$

where W, U, b are coefficients specific to the gate and σ is the *sigmoid* function.

2.3. Gated Recurrent Unit

A Gated Recurrent Unit (GRU) [6] is a variant of the RNN architecture and uses gating mechanisms to control and manage the flow of information between cells in the neural network. GRUs were introduced only in 2014 by Cho, et al. and can be considered a relatively new architecture, especially compared to the widely-adopted LSTM. This RNN variant is similar to the LSTMs as it is able to effectively retain long-term dependencies in sequential data and also works to address the short-term memory problem of RNN models [10]. The workflow of the GRU is the same as the RNN and LSTM. However, instead of using a cell state to regulate information, it uses hidden states, and instead of four gates like LSTM, it has two gates, a reset gate and an update gate (Fig. 4). Similar to the gates within LSTMs, the reset and update gates control how much and which information to retain. For each timestep t, value of *candidate cell* $\tilde{c}^{<t>}$, *cell state* $c^{<t>}$, and *activation* $a^{<t>}$ are slightly different from LSTM [3]:

$$\begin{aligned}\tilde{c}^{<t>} &= \tanh(W_c[\Gamma_r * a^{<t1>}, x^{<t>}] + b_c)\\ c^{<t>} &= \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t1>}\\ a^{<t>} &= c^{<t>}\end{aligned}$$

GRU is relatively new, but it is on track to outshine LSTM due to its superior speed while achieving similar accuracy and effectiveness. If the dataset is small, GRU is a better option than LSTM.

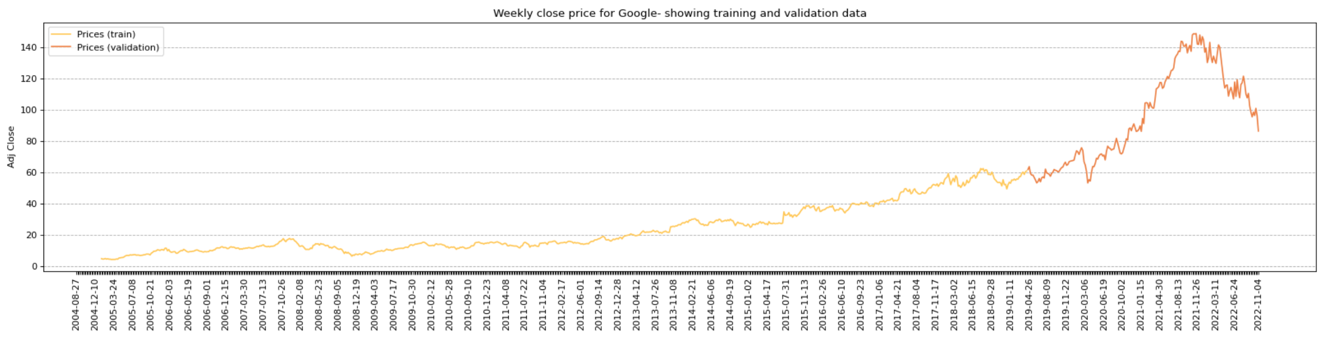


Figure 5. Samples of Google adjusted close price data.

3. Experiment Analysis

In this experiment, *RNN, GRU* and *LSTM* are compared to show the efficiency of variations of RNN in stock price prediction. In addition, three models are shared the same architecture and training configuration to determine how well each variation performs in the stock price dataset from https://www.alphavantage.co. The evaluation metric for model performance is based on ***Mean Squared Error (MSE)***:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i is the actual price and $\hat{y}_i$ is predicted price.

3.1. Dataset

The dataset used in this experiment is the stock price of Google taken from the *Alpha Vantage API*. It is a weekly financial market dataset comprising 951 records from *'2004-08-27'* to *'2022-11-08'* [1] and 7 columns:

- **Open**: opening price
- **High**: maximum price during the day
- **Low**: minimum price during the day
- **Close**: close price adjusted for splits
- **Adj Close**: adjusted close price adjusted for both dividends and splits
- **Volume**: the number of shares that changed hands during a given day
- **Dividend**: payments a company makes to share profits with its stockholders

In this experiment, the adjusted close price is the value predicted by three models, the 80% of the data is used as a train set, and the remaining 20% is a validation set (Fig. 5).

[1]Number of record and date time of stock price can change overtime

Data Preprocessing. The price value can be minimal and tremendous over time, potentially skewing the model performance in unexpected ways. This problem is where data normalization comes in. Normalization can increase the model's accuracy and help the optimization algorithm converge more quickly towards the target minima. By bringing the input data on the same scale and reducing its variance, the model can more efficiently learn from the data and store patterns in the network. Furthermore, RNN and its variations are intrinsically sensitive to the scale of the input data. For all the above reasons, it is crucial to normalize the data using formula as follow:

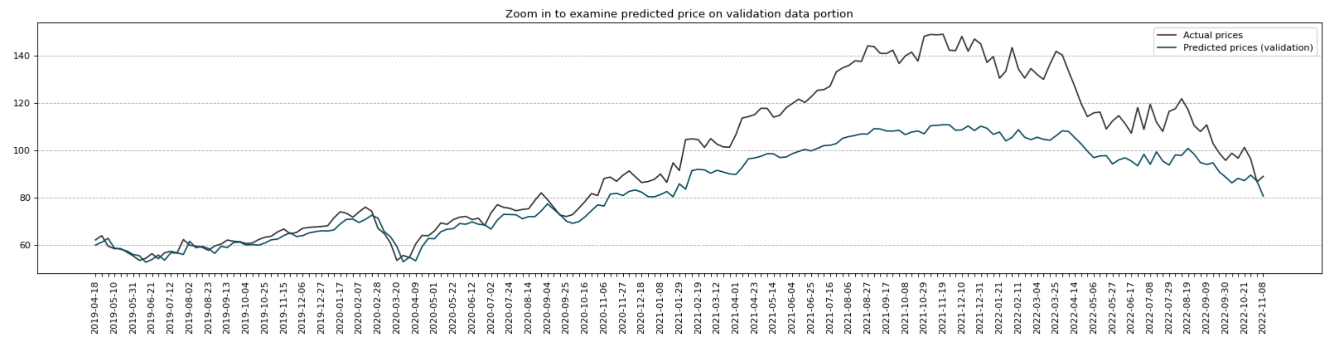
$$x' = \frac{x - \bar{x}}{\sigma}$$

where $\bar{x}$ is the mean of x and σ is its standard deviation.

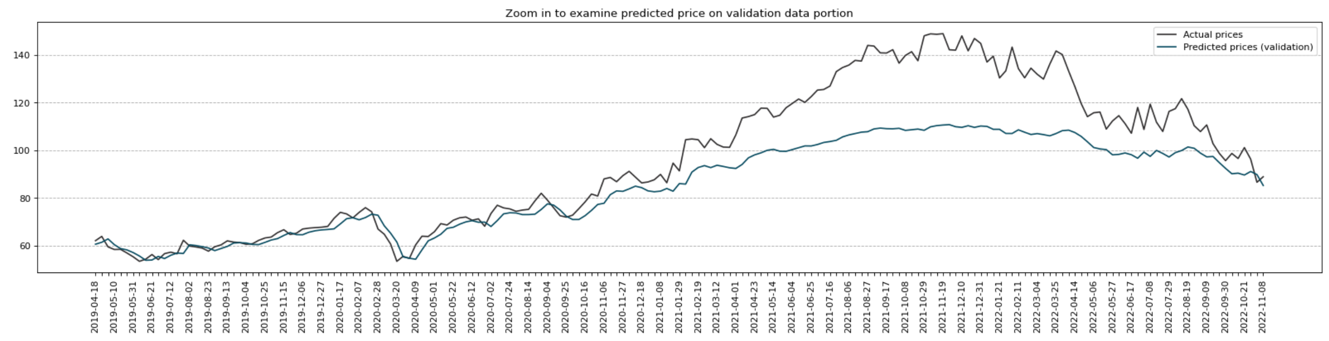
Data Preparation. The model is trained to predict the 21^{st} day price based on the past 20 days' close prices. The number of days, 20, was selected based on a few reasons:

- The length of sequence used in RNN/GRU/LSTM model typically ranges from 15 to 20 words
- Very long input sequences may result in vanishing gradients
- Longer sequences tend to have much longer training times

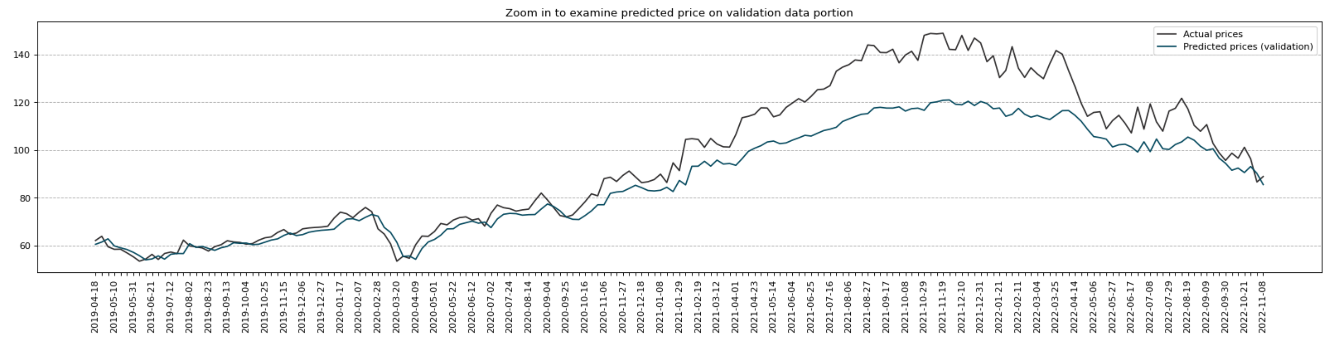
After transforming the dataset into input features and output labels, the shape of feature X is $(951, 20)$, 950 for the number of rows, each row containing a sequence of the past 20 days' prices. The corresponding label Y data shape is $(951,)$, which matches the number of rows in X. These processed data will then be split into train and validation sets.



(a) The prediction of RNN.



(b) The prediction of LSTM.



(c) The prediction of GRU.

Figure 6. The visualisation of prediction on validation set of three models.

3.2. Model Architecture & Training Configuration

In this comparison, three different models are initialized with the same architecture as follows:

- **RNN/GRU/LSTM unit**: input size is 1, hidden dimension is 32, number of layers is 2.
- **Dropout**: dropout with $p = 0.2$ to prevent overfitting.
- **Fully-connected (FC) layer**: one FC layer with input size of number of layers $\times$ hidden dimension(2×32), and output size is 1.

The training process is performed with the same configuration:

- **Batch size**: 64
- **Epochs**: 100
- **Learning rate**: 0.01
- **Optimizer**: Adaptive Moment Estimation (Adam)
- **Loss**: Mean Squared Error (MSE)

The learning rate scheduler is also applied to the training process in order to gain higher performance. This scheduler will decay the learning rate of each parameter group by 0.1 every 40 epochs.

3.3. Training Results

After training with 100 epochs, all three models can generalise data well and notice the price trend over time. Even

Model	Loss	Params
RNN	0.004079	3,297
LSTM	0.003603	12,993
GRU	0.002126	9,761

Table 1. Model's validation loss and number of parameters comparison.

though RNN is a vanilla model, it can still predict the price not too far from the actual price. Nevertheless, the price prediction from LSTM and GRU is more promising and closer to the actual price compared to RNN (Fig. 6). This result is understandable since two later-developed models have been refined to address the vanishing gradient problem through their ability to capture information from the prior input. Surprisingly, the GRU model with fewer parameters, $9,761$ compared to $12,993$ in LSTM, has a higher performance with only 0.002126 loss in the validation set (Table. 1). This result has justified the superior performance of GRU, and this model is a good option if memory resources are limited and fast results are desired. However, when it comes to a larger dataset, it would be better to choose LSTM since the dataset might have a long-term trend, and LSTM can exploit that information better for forecasting.

4. Conclusion

In the data preprocessing for stock price prediction, it is crucial to apply normalisation to avoid skewing the result since RNN, and its variants are sensitive to the scale of input data. Furthermore, the model comparison in section 3.3 shows that the RNN model is less effective at forecasting time series data than GRU and LSTM. In addition, GRU is relatively new, but its performance is on par with LSTM and computationally more efficient. However, there is yet to be a clear answer to which variant performs better, which still depends on the task and dataset. For future direction, because the stock price can be easily affected by its company earnings or another company status, one approach to achieve better results is adding more features for training. Another way is to increase the model complexity by adding the FC layer before GRU/LSTM unit to map input values into a high-dimensional feature space, transforming the features for GRU/LSTM unit.

References

[1] Deep learning - wikipedia. https://en.wikipedia.org/wiki/Deep_learning. (Accessed on 10/10/2022). 1

[2] Stock market prediction - wikipedia. https://en.wikipedia.org/wiki/Stock_market_prediction#Fundamental_analysis. (Accessed on 11/02/2022). 1

[3] Afshine Amidi and Shervine Amidi. Cs 230 - recurrent neural networks cheatsheet. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. (Accessed on 11/02/2022). 2, 3

[4] Avijeet Biswal. Recurrent neural network (rnn) tutorial: Types and examples. https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn, October 2022. (Accessed on 11/02/2022). 3

[5] Avijeet Biswal. Stock price prediction using machine learning: An easy guide. https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning, October 2022. (Accessed on 11/02/2022). 1

[6] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014. 3

[7] IBM Cloud Education. What are recurrent neural networks? — ibm. https://www.ibm.com/cloud/learn/recurrent-neural-networks, September 2020. (Accessed on 11/02/2022). 2

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. 3

[9] Vijaysinh Lendave. Lstm vs gru in recurrent neural network: A comparative study. https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/, August 2021. (Accessed on 11/02/2022). 3

[10] Gabriel Loye. Gated recurrent unit (gru) with pytorch. https://blog.floydhub.com/gru-with-pytorch/, July 2019. (Accessed on 11/02/2022). 3

[11] Hemanth Pedamallu. Rnn vs gru vs lstm. https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573, November 2020. (Accessed on 11/02/2022). 3