

2021 AI First Engineering Proceedings

**Geoffrey C. Fox
Gregor von Laszewski**

Editors

Contact: laszewski@gmail.com

<https://cybertraining-dsc.github.io/pub/docs/ai2021.pdf>

August 16, 2021 - 03:38 PM

2021 AI FIRST ENGINEERING

Geoffrey C. Fox Gregor von Laszewski

(c) Gregor von Laszewski, 2021

2021 AI FIRST ENGINEERING

1 PREFACE

1.1 Disclaimer 

1.1.1 Acknowledgment

1.1.2 Extensions

2 REFERENCES

1 PREFACE

Mon Aug 16 15:38:19 EDT 2021 

1.1 DISCLAIMER

This book has been generated with [Cyberaide Bookmanager](#).

Bookmanager is a tool to create a publication from a number of sources on the internet. It is especially useful to create customized books, lecture notes, or handouts. Content is best integrated in markdown format as it is very fast to produce the output.

Bookmanager has been developed based on our experience over the last 3 years with a more sophisticated approach. Bookmanager takes the lessons from this approach and distributes a tool that can easily be used by others.

The following shields provide some information about it. Feel free to click on them.

1.1.1 ACKNOWLEDGMENT

If you use bookmanager to produce a document you must include the following acknowledgement.

“This document was produced with Cyberaide Bookmanager developed by Gregor von Laszewski available at <https://pypi.python.org/pypi/cyberaide-bookmanager>. It is in the responsibility of the user to make sure an author acknowledgement section is included in your document. Copyright verification of content included in a book is responsibility of the book editor.”

The bibtex entry is

```
@Misc{www-cyberaide-bookmanager,
  author =      {Gregor von Laszewski},
  title =       {{Cyberaide Book Manager}},
  howpublished = {pypi},
  month =        apr,
  year =         2019,
  url={https://pypi.org/project/cyberaide-bookmanager/}
}
```

1.1.2 EXTENSIONS

We are happy to discuss with you bugs, issues and ideas for enhancements.
Please use the convenient github issues at

- <https://github.com/cyberaide/bookmanager/issues>

Please do not file with us issues that relate to an editors book. They will provide you with their own mechanism on how to correct their content.

2 REFERENCES



Reports 2021

Collection of reports contributed in Spring 2021

⌚ 1 minute read

This page contains the list of the reports and projects done in Spring 2021.

List for 2021

Reports and Projects

- sp21-599-359: [Project: Deep Learning in Drug Discovery, Anesu Chaora](#)
 - Stars 0 Check Report passing Status passing open issues 0 code size 773 kB
repo size 4.74 MB
- sp21-599-357: [Project: Structural Protein Sequences Classification, Jiayu Li](#)
 - Stars 0 Check Report passing Status passing open issues 0 code size 331 kB
repo size 1.05 MB
- sp21-599-355: [Project: Chat Bots in Customer Service, Anna Everett](#)
 - Stars 0 Check Report passing Status passing open issues 0 code size 91.7 kB
repo size 1.31 MB
- sp21-599-354: [Project: Identifying Agricultural Weeds with CNN, Paula Madetzke](#)
 - Stars 0 Check Report passing Status passing open issues 0 code size 189 kB
repo size 11.1 MB
- sp21-599-358: [Project: Autonomous Vehicle Simulations Using the CARLA Simulator, Jesus Badillo](#)
 - Stars 0 Check Report passing Status passing open issues 0 code size 52.5 kB
repo size 14.1 MB
- sp21-599-356: [Project: Forecasting Natural Gas Demand/Supply, Baekeun Park](#)
 - Stars 0 Check Report passing Status passing open issues 0 code size 1.2 MB
repo size 6.86 MB
- sp21-599-353: [Project: Stock Level Prediction, Rishabh Agrawal](#)
 - Stars 0 Check Report passing Status passing open issues 0 code size 60.8 kB
repo size 986 kB

Last modified August 13, 2021 : [Update index.md \(ada0403d\)](#)

Project: Stock level prediction

This project includes a deep learning model for stock prediction. It uses LSTM, RNN which is the standart for time series prediction. It seems to be the right approach. The author really loved this project since he loves stocks. He invests often, and is also in love with tech, so he finds ways to combine both of them. Most existing models for stock prediction do not include the volume, and Rishabh intendede to use that as an input, but it did not go exactly as planned.

Tags: [project](#) [ai](#) [finance](#)

⌚ 6 minute read

 Check Report passing  Status passing Status: final, Type: Project

Rishabh Agrawal, [sp21-599-353](#), [Edit](#)

- [Stock Prediction.pynb](#)
- [Stock Prediction.pdf](#)

Abstract

This project includes a deep learning model for stock prediction. It uses LSTM, RNN which is the standart for time series prediction. It seems to be the right approach. The author really loved this project since he loves stocks. He invests often, and is also in love with tech, so he finds ways to combine both of them. Most existing models for stock prediction dont include the volume, and Rishabh intendede to use that as an input, but it didn't go exactly as planned.

Contents

- [1. Introduction](#)
- [2. Existing LSTM Models](#)
- [2.1. What is LSTM?](#)
- [2.2. Existing LSTM models and why they don't do great](#)
- [3. Datasets](#)
- [4. Results](#)
- [6. Benchmark](#)
- [7. Conclusion](#)
- [9. References](#)

Keywords: tensorflow, LSTM, Time Series prediction, transformers.

1. Introduction

Using deep learning for stock level prediction is not a new concept, but this project is trying to address a different issue her. Volume. Almost no model actually uses volume, daily volume or weekly volume. People that are experts in this field and do a technical analysis(TA) of stocks use volume for their prediction extensively, so why not use it in the model? It could be ground breaking.

LSTM is the obvious match for this kind of problem, but this project will also try to incorporate transformers in the model. The data set used will be from Yahoo Finance¹.

2. Existing LSTM Models

There are quite a few models out there for stock prediction. But what exactly is LSTM? How does it work, and how is it the obvious option here? There are other models too. Why the current LSTM models aren't that great? How good/bad do they perform?

2.1. What is LSTM?

LSTM is short for Long Short Term Memory networks. It comes under the branch of Recurrent Neural Networks. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn! All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer². LSTM is also used for other time series forecasting such as weather and climate. It is an area of deep learning that works on considering the last few instances of the time series instead of the entire time series as a whole. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor³.

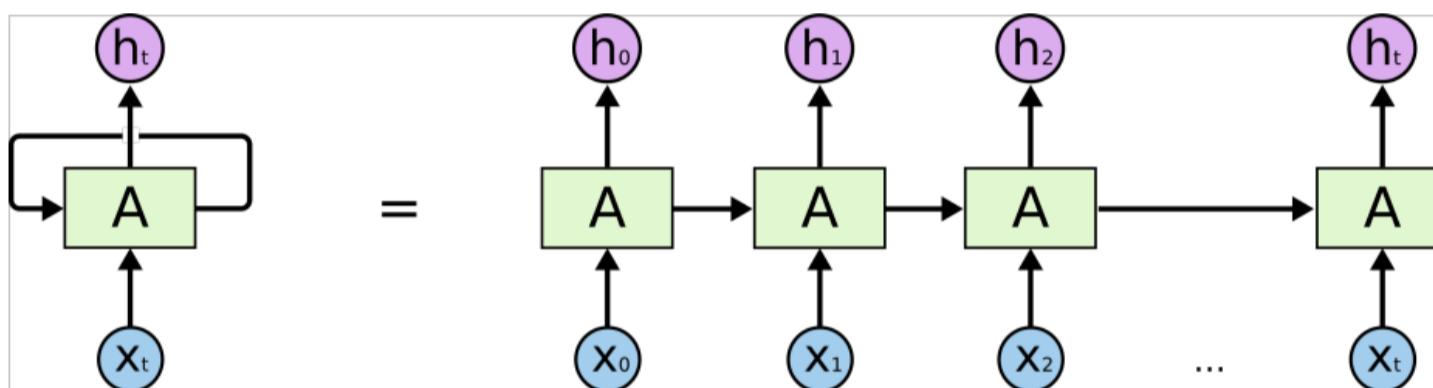


Fig 1 LSTM

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn.

2.2. Existing LSTM models and why they don't do great

This data corroborates what we can see from Figure 4. The low values in RMSE and decent values in R2 show that the LSTM may be good at predicting the next values for the time series in consideration.

Figure 5 shows a sample of 100 actual prices compared to predicted ones, from August 13, 2018 to January 4, 2019.



Figure 5 LSTM predicted vs S&P 500 price for 100 days

Fig 5 LSTM Prediction 1 [Image Source](#)

This figure makes us draw a different conclusion. While in aggregate it seemed that the LSTM is effective at predicting the next day values, in reality the prediction made for the next day is very close to the actual value of the previous day. This can be further seen by Figure 6, which shows the actual prices lagged by 1 day compared to the predicted price⁴.



Figure 6 LSTM predicted vs 1-day lagged S&P 500 price for 100 days

Fig 6 LSTM Prediction 2 [Image Source](#)

In one other model on Google Colab we see an output like this



Fig 7 LSTM Prediction 3 [Image Source](#)

Here we can see that clearly the model did not do well. In this model, the LSTM didn't even get the trends correctly. There is definitely need for some changes in the layers, and dropout coefficient. My guess is that this model was really overfitted since the first half of the prediction did relatively well⁵.

3. Datasets

As mentioned above, the Yahoo finance Data set will be used. It is really easy to get. Data can be download from any time stamp to as new as today. There is a download csv button to do so.

Here is an example of [AAPL stock](#) on Yahoo Finance¹.

For this project the Amazon stock was chosen. The all time max historic data was downloaded from Yahoo Finance in csv format. Here is the [link](#)

4. Results

Fig 8 shows the result of my model. The results weren't as expected. This project was meant to be for adding volume to the input layer. We tried to do that in the model, but it failed. It gave a straight line for the prediction. This project did find a unique way to use the LSTM. It played around with the layers, number of layers, the dropout coefficient to find the most accurate balance. The output shows a more reliable and believable output, and that is some progress. There might be ways to incorporate volume. Later, maybe it needs better scaling. But the prediction did get the trends pretty accurately, even though it might not have gotten the exact price correctly. Amazon stock also soared unbelievably this year due to COVID-19 and many other external factors that were not incorporated in the model at all, so it is really common to see an undervalued prediction.



Fig 8

6. Benchmark

Cloudmesh was used for the benchmark for this project. According to the documentation⁶, I used the Stopwatch.start() and Stopwatch.stop() functions. Fig 9 shows the output. Loading the dataset or prediction of the data doesn't take long at all. The majority of the time is taken for the training which is expected. Google Colab was the tool used and utilized the GPU which is why each epoch just took about 2 seconds. When a personal CPU or the default CPU provided by Google Colab was used, it took about 30-40 seconds for each epoch. You can see the system configuration in Fig 9 too.

Attribute	Value
BUG_REPORT_URL	https://bugs.launchpad.net/ubuntu/
DISTRIB_CODENAME	bionic
DISTRIB_DESCRIPTION	"Ubuntu 18.04.5 LTS"
DISTRIB_ID	Ubuntu
DISTRIB_RELEASE	18.04
HOME_URL	https://www.ubuntu.com/
ID	ubuntu
ID_LIKE	debian
NAME	"Ubuntu"
Pretty Name	"Ubuntu 18.04.5 LTS"
PRIVACY_POLICY_URL	https://www.ubuntu.com/legal/terms-and-policies/privacy-policy
SUPPORT_URL	https://help.ubuntu.com/
UBUNTU_CODENAME	bionic
VERSION	"18.04.5 LTS (Bionic Beaver)"
VERSION_CODENAME	bionic
VERSION_ID	"18.04"
cpu_count	2
mem_active	2.4 GiB
mem_available	10.3 GiB
mem_free	4.6 GiB
mem_inactive	5.2 GiB
mem_percent	19.3 %
mem_total	12.7 GiB
mem_used	3.2 GiB
platform.version	#1 SMP Thu Jul 23 08:00:38 PDT 2020
python	3.7.10 (default, Feb 20 2021, 21:17:23) [GCC 7.5.0]
python.pip	19.3.1
python.version	3.7.10
sys.platform	linux
uname.machine	x86_64
uname.node	933b2ca73d28
uname.processor	x86_64
uname.release	4.19.112+
uname.system	Linux
uname.version	#1 SMP Thu Jul 23 08:00:38 PDT 2020
user	collab

Name	Status	Time	Sum	Start	tag	Node	User	OS	Version
Loading Data	ok	0.224	0.224	2021-05-07 16:51:29		933b2ca73d28	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020
training	ok	259.306	259.306	2021-05-07 16:51:30		933b2ca73d28	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020
Prediction	ok	1.361	1.361	2021-05-07 16:55:56		933b2ca73d28	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020

Fig 9 Cloudmesh Benchmark Results

7. Conclusion

Even though the model did not do as expected, we were not able to add volume as an input, we were still able to find some success with changing the number layers and the coefficient values. We can see that the model can successfully predict the trend of the stock prices, even with the external factors affecting the prices a little. The next step for this project would be to try to scale the volume and add it as an input to the model. One other, but rather difficult add-on could be to try to add some external factors as inputs. For an aggregate input of external factors, we could use sentiment analysis through Twitter tweets as an input to the model too.

9. References

Your report must include at least 6 references. Please use customary academic citation and not just URLs. As we will at one point automatically change the references from superscript to square brackets it is best to introduce a space before the first square bracket.

1. Using dataset for stocks, [Online resource] <https://finance.yahoo.com/> ↗
2. Understanding LSTM Networks, [Online Resource] <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
3. AI stock market forecast, [Online resource] <http://ai-marketers.com/ai-stock-market-forecast/>
4. Why You Should Not Use LSTM's to Predict the Stock Market, [Online resource] <https://www.blueskycapitalmanagement.com/machine-learning-in-finance-why-you-should-not-use-lstms-to-predict-the-stock-market/>
5. Google Stock Price Prediction RNN, [Google Colab] https://colab.research.google.com/github/Mishaall/Geodemographic-Segmentation-ANN/blob/master/Google_Stock_Price_Prediction_RNN.ipynb#scrollTo=skhdvmCywHrr ↗

6. Cloudmesh, [Documentation] <https://cloudmesh.github.io/cloudmesh-manual/autoapi/cloudmeshcommon/cloudmesh/common/StopWatch/index.html#module-cloudmesh-common.cloudmesh.common.StopWatch> 

Project: Identifying Agricultural Weeds with CNN

Weed identification is an important component of agriculture, and can affect the way farmers utilize herbicide. When unable to locate weeds in a large field, farmers are forced to blanket utilize herbicide for weed control. However, this method is bad for the environment, as the herbicide can leech into the water, and bad for the farmer, because they then must pay for far more fertilizer than they really need to control weeds. This project utilizes images from the Aarhus University [^1] dataset to train a CNN to identify images of 12 species of plants. To better simulate actual rows of crops, a subset of the images for testing will be arranged in a list representing a crop row, with weeds being distributed in known locations. Then, the AI is tested on the row, and should be able to determine where in the row the weeds are located.

Tags: [project](#) [ai](#)
🕒 9 minute read

 Check Report passing  Status passing Status: final, Type: Project

Paula Madetzke, [sp21-599-354](#), [Edit](#)

Code:

- [data_prep.ipynb](#)
- [CNN_Code.ipynb](#)

Abstract

Weed identification is an important component of agriculture, and can affect the way farmers utilize herbicide. When unable to locate weeds in a large field, farmers are forced to blanket utilize herbicide for weed control. However, this method is bad for the environment, as the herbicide can leech into the water, and bad for the farmer, because they then must pay for far more fertilizer than they really need to control weeds. This project utilizes images from the Aarhus University [^1] dataset to train a CNN to identify images of 12 species of plants. To better simulate actual rows of crops, a subset of the images for testing will be arranged in a list representing a crop row, with weeds being distributed in known locations. Then, the AI is tested on the row, and should be able to determine where in the row the weeds are located.

Contents

- [1. Introduction](#)
- [2. Pre-Processing The Data](#)
- [3. Running the CNN](#)
- [4. Benchmarking](#)
- [5. Possible Extension](#)
- [6. Conclusion](#)
- [7. Acknowledgments](#)
- [8. References](#)

Keywords: Agriculture, CNN.

1. Introduction

With a growing global population, and a changing climate that can make farm work hostile, it is increasingly important for farms to be efficient food producers. Fertilizers, pesticides, and herbicides have allowed modern farms to produce far higher yields than they would otherwise be able to. However, the environmental impact from the runoff of these chemicals when they leave the farm can be incredibly detrimental to both human and natural wellbeing. This is why agriculture is a field that is ripe for improvement from AI. There are opportunities for AI to be used in the harvesting of crops, predictive analytics, and field monitoring. In this project, AI is used in the field monitoring application. This is helpful for farmers because spot detection of weeds will help them avoid using as much herbicide. This project will explore how pictures of plants can be used to detect weeds in crop fields. The AI has been trained with a CNN on pictures of 12 seedling species including 960 individual plants representing both weeds desired crops [1](#). Although 12 species are available, only 3 will be studied in this project: black grass, shepherd's purse, and sugar beet. These plants were chosen because their seedlings look different enough to give the AI a better chance at successfully detecting differences. Then, the AI is tested to determine the accuracy of the model. First this is accomplished by reserving a subset of the training data to be tested by the AI in order to get a base-line test of accuracy. Next, the test data will be arranged in a list to mimic a row of crops with a desired crop, and several weeds. Then the AI would go through the images and a program would display where a farmer would need to apply the herbicide, according to the AI. The true test would be to see if the program can produce a helpful map to narrow down where herbicide should be applied.

Previous similar work has been done with this dataset [2](#) in Keras with CNN image recognition, while this project is implemented in pytorch. Similar agricultural image recognition with plant disease [3](#) is also available to be studied.

2. Pre-Processing The Data

The plant dataset [1](#) has three sets of image options to choose from. The first has large images of trays with multiple plants of the same species, the second has cropped images of individual seedlings from the larger picture (non-segmented), while the third is both cropped, and has the background replaced with black pixels such that only the leaves remain in the picture (segmented). To train the data, the segmented data is used. This is because it is important for the AI to train such that it detects patterns only for the most important features of the image. If the AI were to train on the images in the background, it might learn features of the sand or border instead of the leaves.



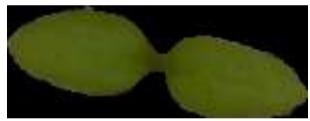


Figure 1 Dataset Image options [1](#): Large image with multiple plants (top), non-segmented (middle), segmented (bottom)

The next component to consider is the image size to be used. The CNN requires all images to be a uniform size to run correctly, while the original segmented data contains images of different sizes, ranging from a few 10s of pixels wide to over 4000. In order to get a sense a balance between having a broad enough dataset to have a large enough representation of the plant pictures to draw meaningful conclusions and the need to have reasonable file sizes, the image's width or height, whichever is larger, was recorded and loaded into a histogram for each plant species. From a visual examination, 300x300 pixels was selected. An additional 100 pixels of padding were added to the final image size to prevent the images from being cut off when rotated. For each image in the segmented dataset, it is expanded to 400x400 pixels, and added to the straight image folder. Then, each image is rotated and versions are saved to a rotation folder. An ideal rotation would allow a fine grain of rotation. However, some resource limitations were met with the GPU used for the neural net. At first, the images were rotated by 10 degrees each before being saved, then 30, then 60, and finally 90. Finally, a csv file with ids and labels is needed to identify the images, and separate the training and the test data. A preprocessing python script can be used to achieve all of these goals.

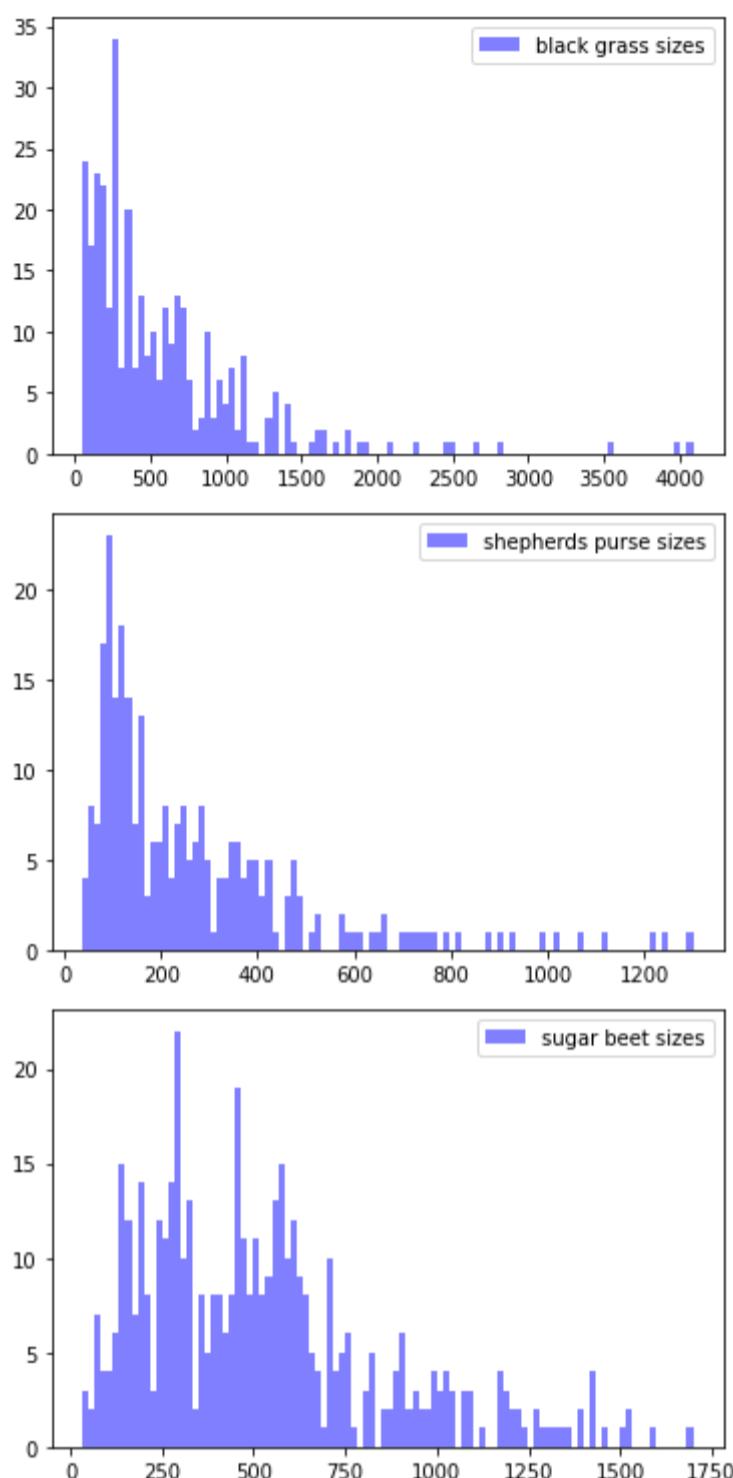


Figure 2 Histograms of the max(width, height) of each of the plant species in the full dataset

3. Running the CNN

The implementation of the CNN is heavily based upon the tutorial on MINST fashion identification [4](#) where the CNN identifies 28x28 pixel images of clothing. It is a 2 layer CNN implemented in pytorch. In the original neural net, there are 70000 of these small images and 9 clothing designations. In this dataset, there are only 3 types of labels, but much larger images. The tutorial had a train image set, a validation set, and a test set. After preprocessing, there were 420 suitable 400x400 unrotated images. Without any changes besides adjusting the file paths and image size parameters, the CNN could inconsistently get approximately 50 percent accuracy with the test data, with the highest accuracy at 15 epochs. Because this implementation has relatively few images compared to the MINST clothing dataset, the next test was to remove the validation set of images, in order to use more of the prepared images for training. The result was a maximum of 79 percent accuracy at 25 epochs.

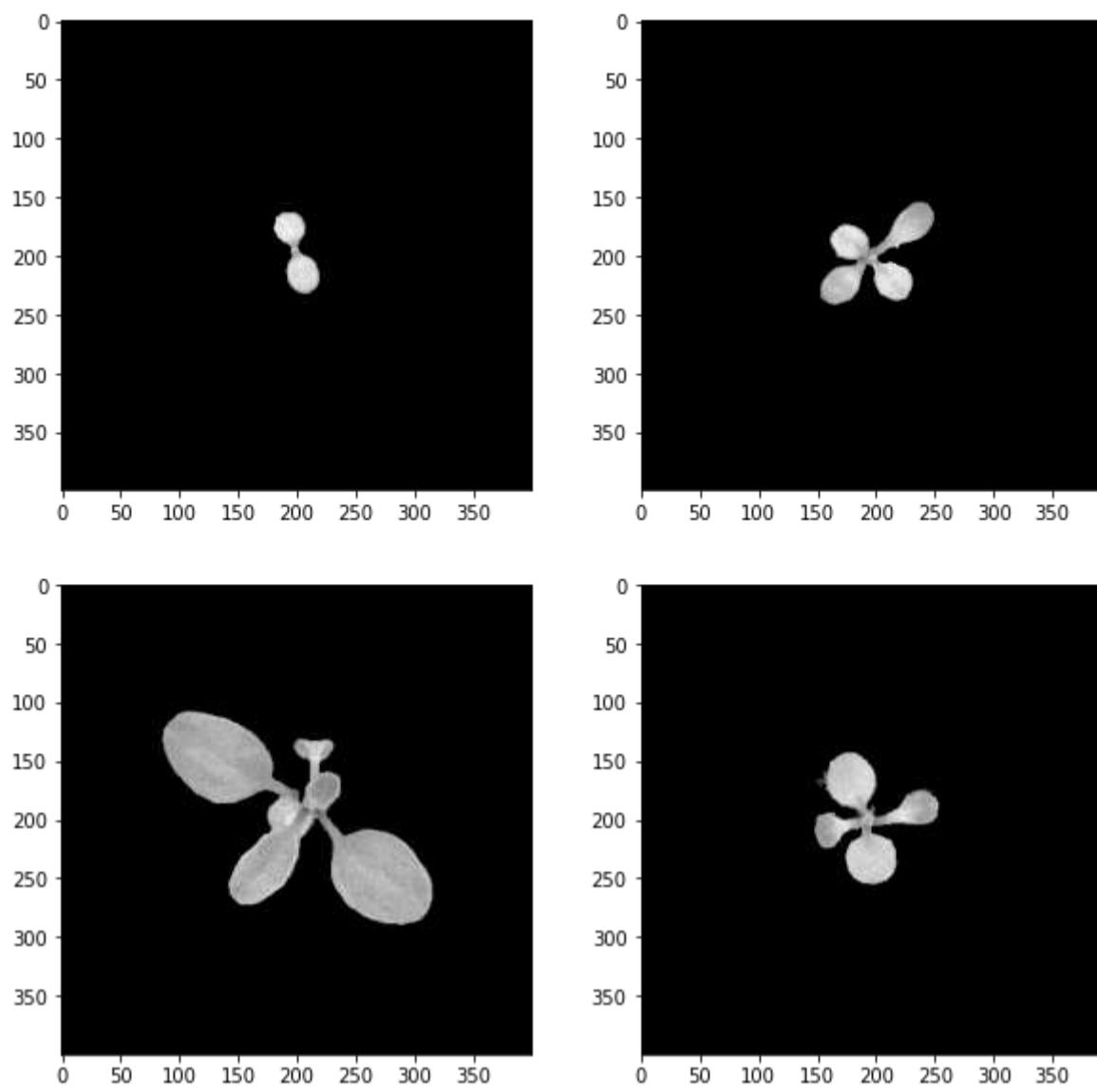


Figure 3 Examples of prepared training images

The testing accuracy of the neural net on the test data tracked closely with the prediction accuracy when train accuracy was around 30 to 50 percent, lagging by only a few percent. However, when the training dataset reached the 70 percent accuracy range, the test set accuracy remained at only around 50 percent. Part of this could be due to the small size of the test set, or the fact each plant species had a slightly different number of suitable test images.

An interesting observation is that although the model was able to consistently reach 60 percent accuracy, it would sometimes fall into a pattern where it would categorize all or almost all of the test set as just one of the plants. There was no consistency in which plant it defaulted to, but in between runs where the model reached decent accuracy, it would repeat this behavior. Because the test set was evenly distributed between the three plants, this would cause the accuracy to go to around 30 percent.

In order to see if a visualization could help a human easily see where the hypothetical herbicide would need to be placed, a chart was created with tiles. The first row is the true layout of the three types of plants in the dataset, where each plant is assigned a color. The second row is the AI prediction of which type of plant the test set is. Even with the 79 percent test accuracy rate, it was not as clear as it could be from the image how accurate the model was. One way of making the visualization both easier to visually determine accuracy and more realistic is to have one type of plant be the dominant crop, and have patches of weeds throughout the row. The major obstacle to this was the fact that there were not enough suitable images to have a dominant plant in the test group. Too many images used in testing would have a detrimental impact on training the model.

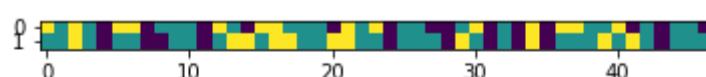


Figure 4 The top row 0 shows the actual species distribution of the plants, while the bottom row 1 shows the AI prediction

4. Benchmarking

The longest operation by far was the first time reading in the images. This had a timed tqdm built in, so the stopwatch function was not used here. Subsequent running of the CNN training program must cache some of the results, so the image loading takes much less time. An initial

loading of the 420 images took approximately 10 minutes. With the GPU accelerator enabled, the training of 25 epochs got to 6.809s while the test only took .015s.

5. Possible Extension

The rotated images of the plants were ultimately not explored. This is due to the fact that the implementation that was followed required GPU resources on Google Colab. With only 420 images to feed into the model, the GPU was not strained. However, even with 90 degree rotations, it appeared that the GPU memory limit for running the entire notebook was reached before even the first epoch. If this project were to be attempted again, the images would either need to be smaller, or fewer straight images should be loaded so that their rotations would not exhaust GPU allocation. One way to go about this would be to have a rotation set with fewer images of individual plants, but to rotate them such that the rotated dataset is still around 420 images. Then, a comparison could be made between the training accuracy of 420 separate examples of each species vs fewer individual plants at a wider range of angles.

6. Conclusion

With 420 non-rotated plant images, a maximum accuracy of 79 percent was able to be reached in part by using the entire training dataset for training rather than validation, as the project this was based on did. With a significantly smaller number of images for the neural net to train on, even a 10 percent change in the available training images mattered in its implementation. Another main conclusion is the importance of resource use and choice in running CNNs. Although in theory there could only be benefits to adding more images to train, with large enough datasets, practical computing limitations become increasingly apparent. Future work would involve choosing a model that either did not require GPU allocation, or modifying the chosen images to take fewer computing resources.

7. Acknowledgments

Dr. Geoffrey Fox

Dr. Greggor von Laszewski

8. References

1. Aarhus University <https://vision.eng.au.dk/plant-seedlings-dataset/> 
 2. Plant Seedling Classification <https://becominghuman.ai/plant-seedlings-classification-using-cnns-ea7474416e65/>
 3. Oluwafemi Tairu <https://towardsdatascience.com/plant-ai-plant-disease-detection-using-convolutional-neural-network-9b58a96f2289>
 4. Pulkit Sharma <https://www.analyticsvidhya.com/blog/2019/10/building-image-classification-models-cnn-pytorch/> 
-

Project: Chat Bots in Customer Service

Automated customer service is a rising phenomon for buisnesses with an online presence. As customer service bots advance in complication of problems they can handle one concern about the altered customer experiece is how the information is conveyed. Using customer support data tweets on twitter this project runs sentiment analysis on it customer tweets and then train a convolutional neural network to examine if conversation tone can be detected early in the conversation.

Tags: [project](#) [ai](#) [nlp](#)

⌚ 7 minute read

 Check Report passing  Status passing Status: final , Type: Project

Anna Everett, [sp21-599-355](#), [Edit](#)

- [twitter_support_analysis.py](#)
- [twitter_support_analysis.pdf](#)

Abstract

Automated customer service is a rising phenomon for buisnesses with an online presence. As customer service bots advance in complication of problems they can handle one concern about the altered customer experiece is how the information is conveyed. Using customer support data tweets on twitter this project runs sentiment analysis on it customer tweets and then train a convolutional neural network to examine if conversation tone can be detected early in the conversation.

Contents

- [1. Introduction](#)
- [2. Procedure](#)
 - [2.1 The Dataset](#)
 - [2.2 Simplifying the Dataset](#)
- [3. The Algorithm](#)
 - [3.1 Sentiment Analysis Overview and Implementation](#)
 - [3.2 Convolutional Neural Networks \(CNN\)](#)
 - [3.3 Splitting Up the Data](#)
- [4. Training the Model](#)
- [5. Conclusion](#)
- [6. References](#)

1. Introduction

2. Procedure

- The Dataset
- Simplifying the dataset

3. The Algorithm

- Sentiment Analysis Overview and Implementation
- Convolutional Neural Networks (CNN)
- Splitting Up the Data

4. Training the Model

5. Conclusion

6. References

Keywords: AI, chat bots, tone, nlp, twitter, customer service.

1. Introduction

Please note that an up to date version of these instructions is available at

- <https://github.com/cybertraining-dsc/hid-example/blob/main/project/index.md>

Some studies report that customers prefer live chat for their customer service interactions¹.

Currently, most issues are simple enough that they can be resolved by a bot; and over 70% of companies are already using or have plans to use some form of software for automation.

Existing AI's in use are limited to simple and common enough questions and/or problems that allow for generalizable communication. As AI technology develops and allows it to handle more complicated problems the communication methods will also have to evolve.

LIVE CHAT IS THE LEADING DIGITAL CONTENT METHOD

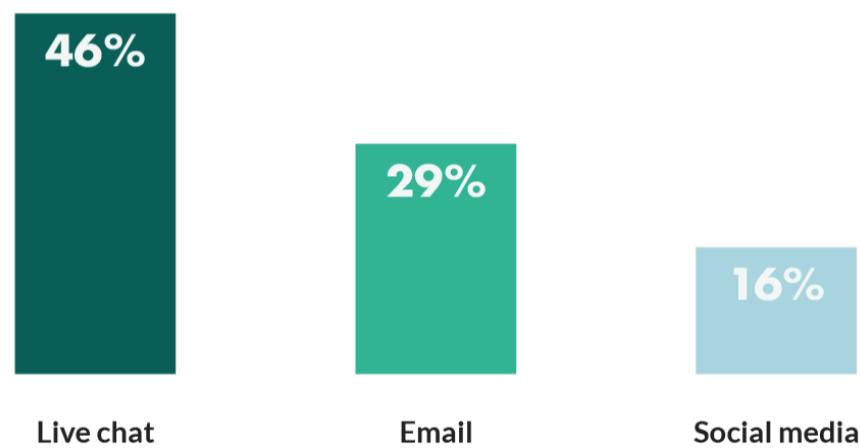


Fig 1: Customer Support Preferences ¹

An article by Forbes "AI Stats News: 86% Of Consumers Prefer Humans To Chatbots", states that only 30% of consumers believe that an AI would be better at solving their problem than a human agent². Human agents are usually preferred because humans are able to personalize conversations to the individual customer. On the other hand, automated software allows for 24/7 assistance if needed, the scale of how many customers a bot would be able to handle is considered larger and more efficient in comparison to what humans can handle, and a significant amount of questions are simple enough to be handled by a bot³.

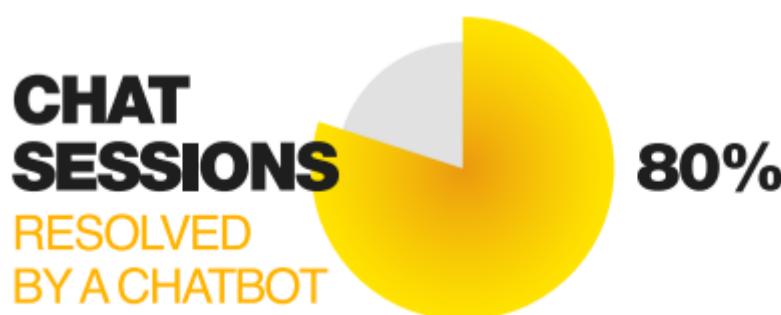


Fig 2: Support Satisfaction [Image](#)

[source](#)

To get the best out of both versions of service, this project uses natural language processing to analyze social media customer service conversations. This is then run through a convolutional neural network to predict if tone can be determined early in the conversation.

2. Procedure

2.1 The Dataset

The dataset comes from the public dataset compilation website, kaggle, and can be found at [Kaggle Dataset](#). This dataset was chosen due to twitter support's informal nature that is expected to come with quicker and more automated customer interactions.

The content of the data consists of over 2.5 million tweets from both customer and various companies that have twitter account representation. Each row of data consists of: the unique tweet id, an anonymized id of the author, if the tweet was sent directly to a company, date and time sent, the content of the tweet, the ids of any tweets that responded to this tweet if any, and the id of the tweet that this was sent in response to if any.

2.2 Simplifying the Dataset

The raw dataset is large and contains unnecessary information that isn't needed for this purpose. In order to trim the dataset only the first 650 samples are taken.

Next, since the project goal is to predict customer sentiment any tweet and related data sent by a company is removed. Luckily, companies author id's don't get anonymized and therefore we can filter those out by removing any data associated with an author id that contains letters. For speed and simplicity these author id are only checked for vowels.

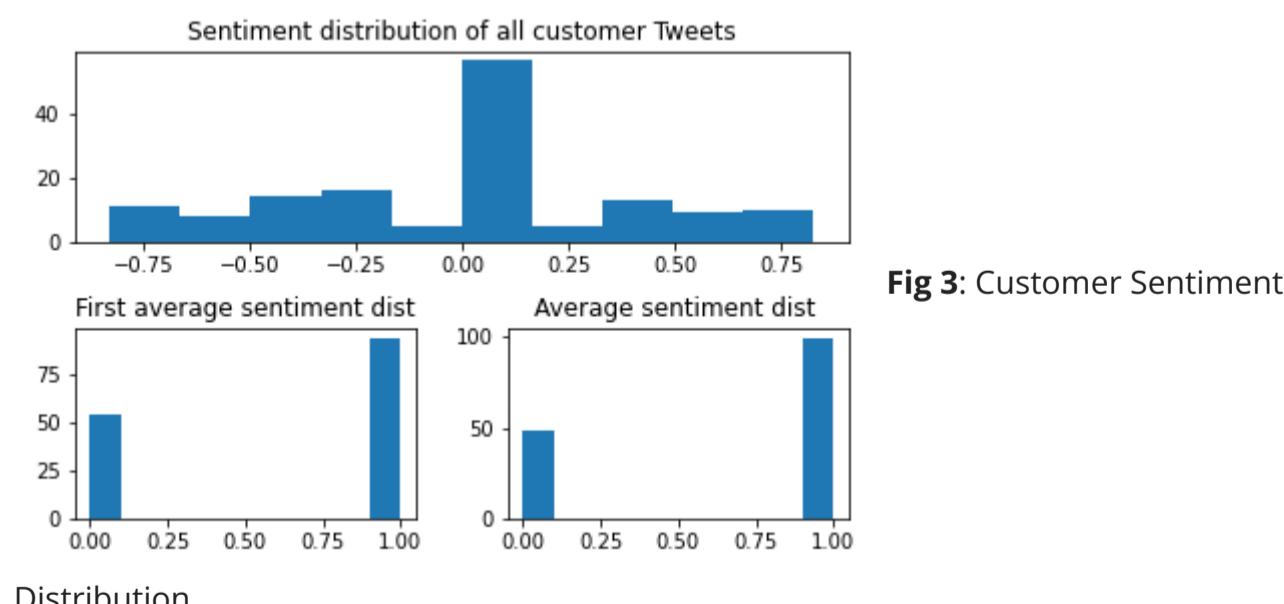
3. The Algorithm

3.1 Sentiment Analysis Overview and Implementation

Sentiment analysis is the process of taking in natural language text and determining if it has a positive or negative sentiment ⁴. This process is useful for when doing market research and tracking attitudes towards a particular company. In a similar fashion this project uses sentiment analysis to determine the text sentiment of the customer initially with their first inquiry and also the sentiment of their side of the conversation as a whole.

The goal of analyzing both the first and general tone of the text is to determine if a general correlation between them can be found.

The main library used for the sentiment analysis of the data was "nltk" and its subpackage "SentimentIntensityAnalyzer"



As can be seen in the Fig 3 the distribution of the sentiment values are generally on a normal distribution. Looking at the binary classifications of both the first and average sentiment distribution it can be seen that while the majority can be classified as positive, 1, there's still a significant amount that are classified as negative, 0.

3.2 Convolutional Neural Networks (CNN)

While convolutional Neural Networks are traditionally used for image processing. [Some articles](#) suggest that CNN's also work well for Natural Language Processing. Traditionally, convolutional neural networks are networks that apply filters over a dataset of pixels and process the pixel as well as those that surrounds it. Typically this is used for images as pictured below in Fig 4, for filtering and edge detecting for identifying objects.

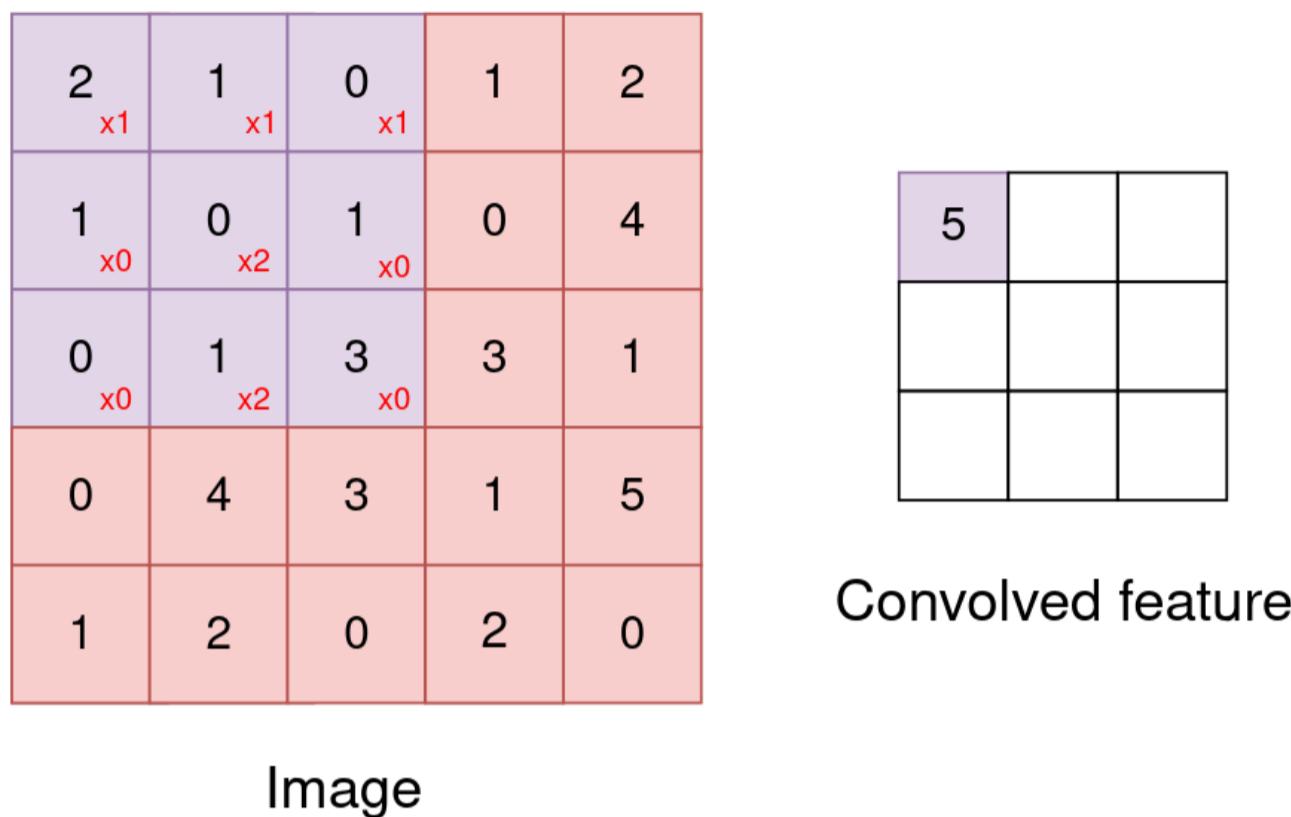


Fig 4: Image convolution [Image source](#)

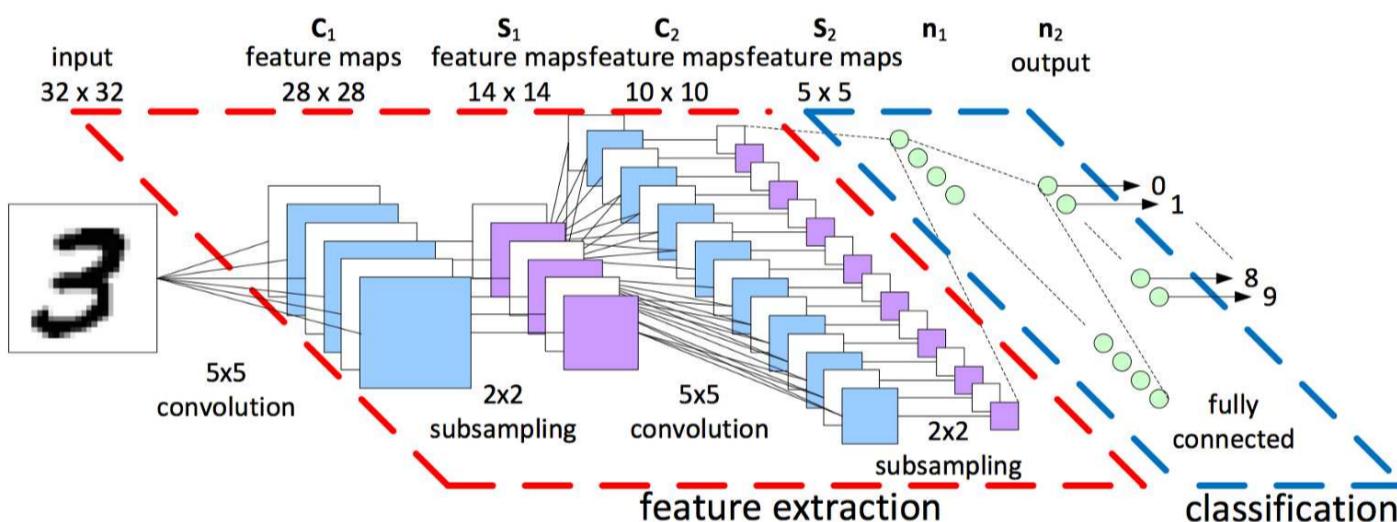
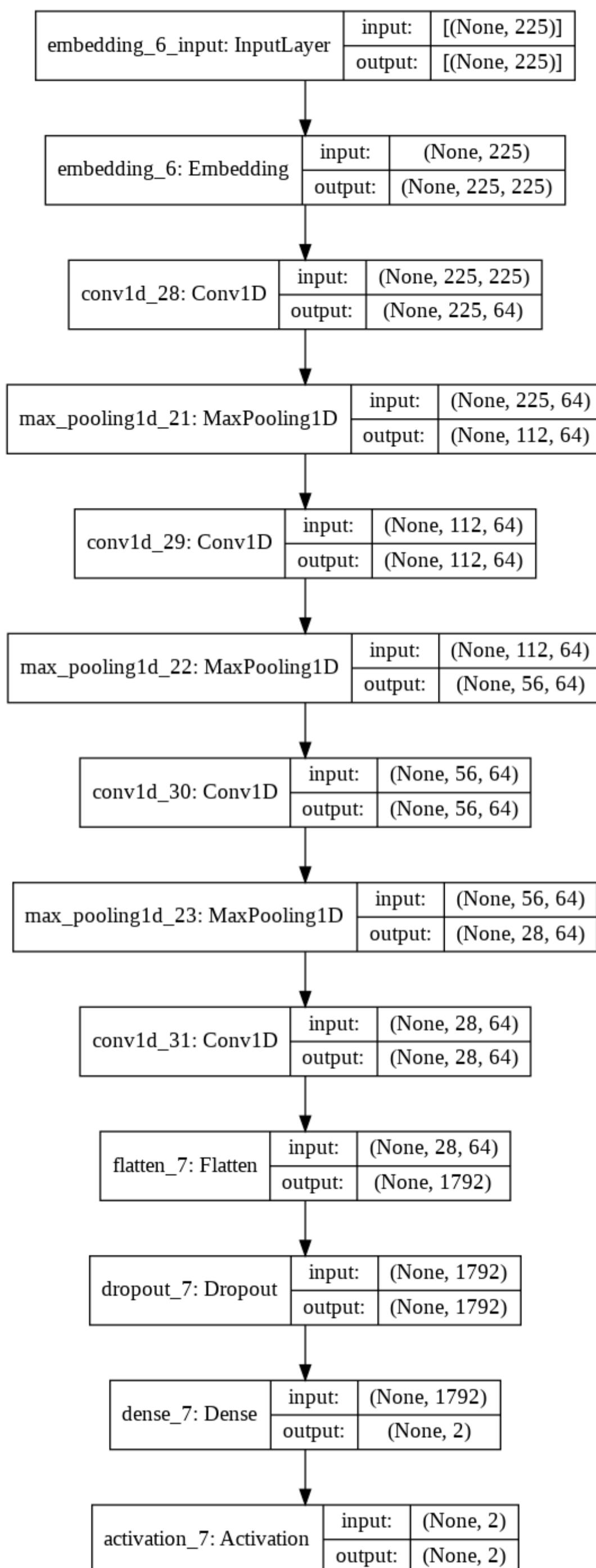


Fig 5: Convolution Visual [Image source](#)

CNN's also work well for natural language processing. Thinking about the English language, meaning and tone of a sentence or text is caused by the relation of words, rather than each word on its own. NLP through CNNs work in a similar fashion to how it processes images but instead of pixels its encoded words that are being convolved.

As can be seen by Fig 6, this project used a multi-layered CNN: beginning with an embedding layer, alternating keras 1 dimension convolution and max pooling layers, a keras dropout layer with a rate of 0.2 to prevent overfitting of the model⁵ and a keras dense layer that implements the activation function into the output⁵.

**Fig 6:** CNN Model

3.3 Splitting Up the Data

In order for the data to be suitable to be run through the CNN the input features must be reshaped into a 2-D array. Afterwards the data was split up using the "sklearn.model_selection" package "train_test_split" with the features being input as the encoded tweets and the labels being input as the general classification sentiment.

4. Training the Model

The model was trained using the training text and training sentiment, because of the small samples used a batch size of 50 was input and run for 10 epochs.

5. Conclusion

In conclusion, while sentiment prediction of customer tweets were not able to be executed there is still useful information found to aid in future attempts. In dealing with the informal raw twitter data ad performing a non-binary sentiment analysis allowed for visualization of the true spread of what can be expected when anticipating dealing with customers, especially in an informal setting such as social media. This project also brought to light the theoretical versatility of convolutional neural networks, though not further examined in this project. Using the model model fit as an indication, there is reasonable evidence that the first inquiry will be enough to predict and take proactive measures in future customer service chat bots. Future execution of this project should include altering the data pre-processing technique and using a larger set of samples from the dataset.

6. References

1. Super Office, [online resource] <https://www.superoffice.com/blog/live-chat-statistics/> 
 2. Forbes, [online resource] <https://www.forbes.com/sites/gilpress/2019/10/02/ai-stats-news-86-of-consumers-prefer-to-interact-with-a-human-agent-rather-than-a-chatbot/?sh=5f5d91422d3b>
 3. Acuire, [online resource] <https://acquire.io/blog/chatbot-vs-live-chat/>
 4. Monkey learn, [online resource] <https://monkeylearn.com/sentiment-analysis/>
 5. Keras, [online documentation] https://keras.io/api/layers/regularization_layers/dropout/ 
-

Project: Forecasting Natural Gas Demand/Supply

Natural Gas(NG) is one of the valuable ones among the other energy resources. It is used as a heating source for homes and businesses through city gas companies and utilized as a raw material for power plants to generate electricity. Through this, it can be seen that various purposes of NG demand arise in the different fields. In addition, it is essential to identify accurate demand for NG as there is growing volatility in energy demand depending on the direction of the government's environmental policy. This project focuses on building the model of forecasting the NG demand and supply amount of South Korea, which relies on imports for much of its energy sources. Datasets for training include various fields such as weather and prices of other energy resources, which are open-source. Also, those are trained by using deep learning methods such as the multi-layer perceptron(MLP) with long short-term memory(LSTM), using Tensorflow. In addition, a combination of the dataset from various factors is created by using pandas for training scenario-wise, and the results are compared by changing the variables and analyzed by different viewpoints.

Tags: [project](#) [ai](#) [energy](#)

⌚ 13 minute read

 Check Report passing  Status passing Status: final, Type: Project

Baekeun Park, [sp21-599-356](#), [Edit](#)

- Code:
 - [Forecasting NG Demand Supply.ipynb](#)

Abstract

Natural Gas(NG) is one of the valuable ones among the other energy resources. It is used as a heating source for homes and businesses through city gas companies and utilized as a raw material for power plants to generate electricity. Through this, it can be seen that various purposes of NG demand arise in the different fields. In addition, it is essential to identify accurate demand for NG as there is growing volatility in energy demand depending on the direction of the government's environmental policy.

This project focuses on building the model of forecasting the NG demand and supply amount of South Korea, which relies on imports for much of its energy sources. Datasets for training include various fields such as weather and prices of other energy resources, which are open-source. Also, those are trained by using deep learning methods such as the multi-layer perceptron(MLP) with long short-term memory(LSTM), using Tensorflow. In addition, a combination of the dataset from various factors is created by using pandas for training scenario-wise, and the results are compared by changing the variables and analyzed by different viewpoints.

Contents

- [1. Introduction](#)
- [2. Related Work](#)
- [3. Datasets](#)

- [4. Methodology](#)
- [4.1. Min-Max scaling](#)
- [4.2. Training](#)
- [4.3. Evaluation](#)
- [4.4. Prediction](#)
- [5. Result](#)
- [5.1 Scenario one\(regional dataset\)](#)
- [5.2 Scenario two\(regional climate dataset\)](#)
- [5.3 Scenario three\(regional temperature dataset\)](#)
- [5.4 Scenario four\(applying timesteps\)](#)
- [5.5 Scenario five\(national dataset\)](#)
- [5.6 Overall results](#)
- [6. Benchmarks](#)
- [7. Conclusion](#)
- [8. Acknowledgments](#)
- [9. Source code](#)
- [10. References](#)

Keywords: Natural Gas, supply, forecasting, South Korea, MLP with LSTM, Tensorflow, various dataset.

1. Introduction

South Korea relies on imports for 92.8 percent of its energy resources as of the first half of 2020 [1](#). Among the energy resources, the Korea Gas Corporation(KOGAS) imports Liquified Natural Gas(LNG) from around the world and supplies it to power generation plants, gas-utility companies, and city gas companies throughout the country [2](#). It produces and supplies NG in order to ensure a stable gas supply for the nation. Moreover, it operates LNG storage tanks at LNG acquisition bases, storing LNG during the season when city gas demand is low and replenish LNG during winter when demand is higher than supply [3](#).

The wholesale charges consist of raw material costs (LNG introduction and incidental costs) and gas supply costs [4](#). Therefore, the forecasting NG demand/supply will help establish an optimized mid-to-long-term plan for the introduction of LNG and stable NG supply and economic effects.

The factors which influence NG demand include weather, economic conditions, and petroleum prices. The winter weather strongly influences NG demand, and the hot summer weather can increase electric power demand for NG. In addition, some large-volume fuel consumers such as power plants and iron, steel, and paper mills can switch between NG, coal, and petroleum, depending on the cost of each fuel [5](#).

Therefore, some indicators related to weather, economic conditions, and the price of other energy resources can be used for this project.

2. Related Work

Khotanzad and Elragal (1999) proposed a two-stage system with the first stage containing a combination of artificial neural network(ANN) for prediction of daily NG consumption [6](#), and Khotanzad et al. (2000) combined eight different algorithms to improve the performance of forecasters [7](#). Mustafa Akpinar et al. (2016) used daily NG consumption data to forecast the NG demand by ABC-based ANN [8](#). Also, Athanasios Anagnostis et al. (2019) conducted daily NG demand prediction by a comparative analysis between ANN and LSTM [9](#). Unlike those methods, MLP with LSTM is applied for this project, and external factors affecting NG demand are changed and compared.

3. Datasets

As described, weather datasets like temperature and precipitation, price datasets of other energy resources like crude oil and coal, and economic indicators like exchange rate are used in this project for forecasting NG demand and supply.

There is an NG supply dataset [10](#) from a public data portal in South Korea. It includes four years from 2016 to 2019 of regional monthly NG supply in the nine different cities of South Korea. In addition, climate data such as temperature and precipitation [11](#) for the same period can be obtained from the Korea Meteorological Administration. Similarly, data on the price of four types of crude oil [12](#) and various types of coal price datasets per month [13](#) are also available through corresponding agencies. Finally, the Won-Dollar exchange rate dataset [14](#) with the same period is used.

As mentioned above, each dataset has monthly information and also has average values instead of the NG supply dataset. It is regionally separated or combined according to the test scenario. For example, the NG supply dataset has nine different cities. One column of cities is split from the original dataset and merged with another regional dataset like temperature or precipitation. On the other hand, each regional value's summation is utilized in a scenario where a national dataset is needed.

The dataset is applied differently for each scenario. In scenario one, all datasets such as crude oil price, coal price, exchange rate, and regional temperature and precipitation are merged with regional dataset, especially Seoul. For scenario two, all climate datasets are used with the regional dataset. Only temperature dataset is utilized with regional dataset in scenario three. In addition, in scenario four, all cases are the same as in scenario one, but the timesteps are changed to two months. Finally, the national dataset is used for scenario five.

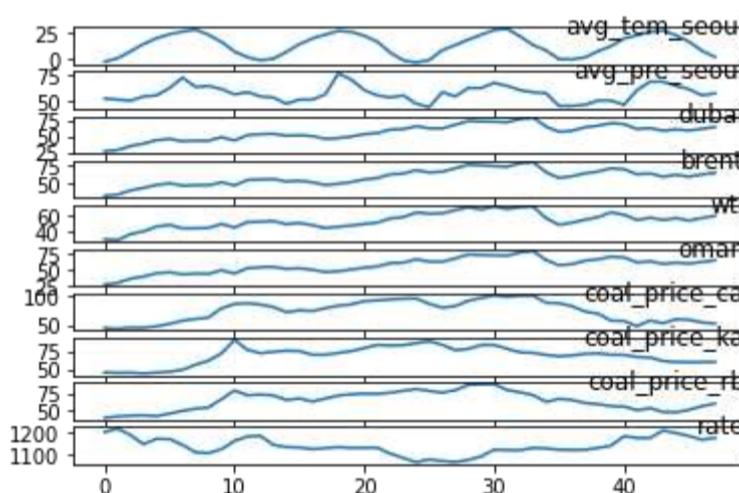


Figure 1: External factors affecting natural gas

4. Methodology

4.1. Min-Max scaling

In this project, all datasets are rescaled between 0 and 1 by Min-Max scaling, one of the most common normalization methods. If there is a feature with anonymous data, The maximum value($\max(x)$) of data is converted to 1, and the minimum value($\min(x)$) of data is converted to 0. The other values between the maximum value and the minimum value get converted to x' , between 0 and 1.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

4.2. Training

For forecasting the NG supply amount from the time series dataset, MLP with LSTM network model is designed by using Tensorflow. The first and second LSTM layers have 100 units, and a total of 3 layers of MLP follow it. Each MLP layer has 100 neurons instead of the final layer, where its neuron is 1. In addition, dropout was designated to prevent overfitting of data, the Adam is used as an optimizer, and the Rectified Linear Unit(ReLU) as an activation function.

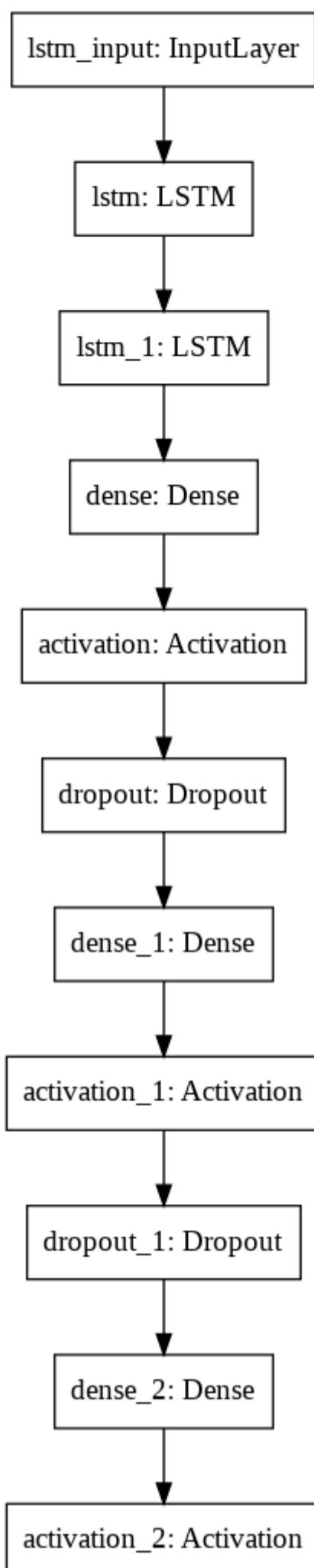


Figure 2: Structure of network model

4.3. Evaluation

Mean Absolute Error(MAE) and Root Mean Squared Error(RMSE) are applied for this time series dataset to evaluate this network model. The MAE measures the average magnitude of the errors and is presented by the formula as following, where n is the number of errors, y_i is the i^{th} true value, and \hat{y}_i is the i^{th} predicted value.

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}$$

Also, The RMSE is used for observing the differences between the actual dataset and prediction values. The following is the formula of RMSE, and each value of this is the same for MAE.

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

4.4. Prediction

Since the datasets used for the training process are normalized between 0 and 1, they get converted to a range of the ground truth values again. From these rescaled datasets, it is possible to obtain the RMSE and compare the differences between the actual value and the predicted value.

5. Result

In all scenarios, main variables such as dropout, learning rate, and epochs are fixed under the same conditions and are 0.1, 0.0005, and 100 in order. In scenarios one, two, three, and five, the training set is applied as twelve months, and in scenario four, next month's prediction comes from the previous two months dataset. For comparative analysis, the results are obtained by changing the size of the training set from twelve months to twenty-four months, and the effect is described. Each scenario shows individual results and is comprehensively compared at the end of this part.

5.1 Scenario one(regional dataset)

The final MAE of the train set is around 0.05, and the one of the test set is around 0.19. Also, the RMSE between actual data and predicted data is around 227018. The predictive graph tends to deviate a lot at the beginning of the part, but it shows a relatively similar shape at the end of the graph.

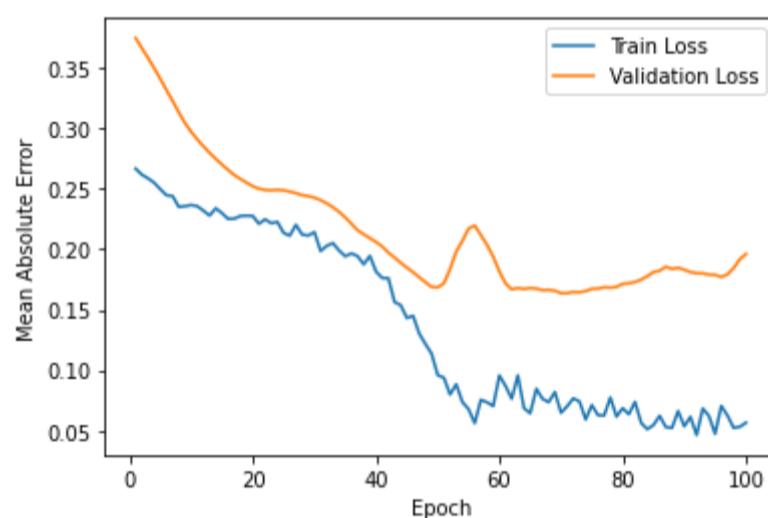


Figure 3: Loss for scenario one

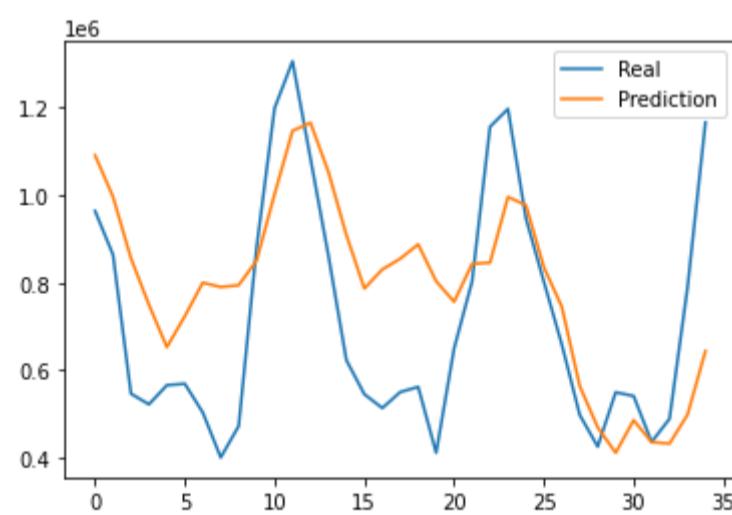


Figure 4: Prediction results for scenario one

5.2 Scenario two(regional climate dataset)

The final MAE of the train set is around 0.10, and the one of the test set is around 0.14. Also, the RMSE is around 185205. Although the predictive graph still differs compared to the actual graph, it shows similar trends in shape.

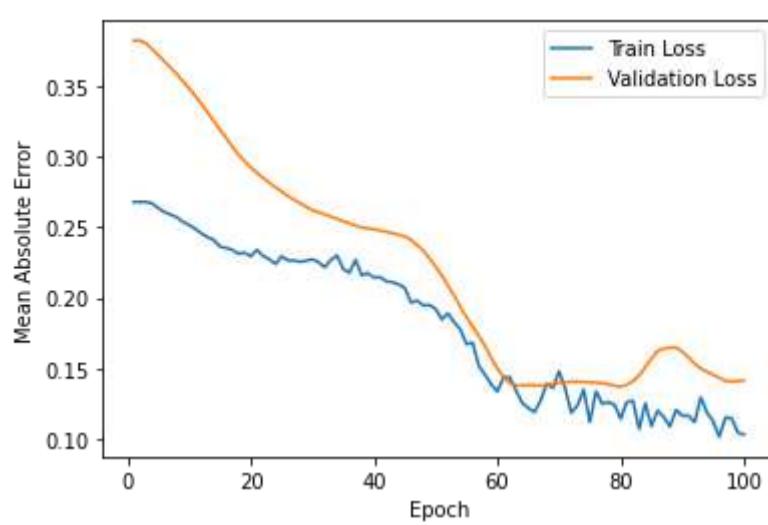


Figure 5: Loss for scenario two

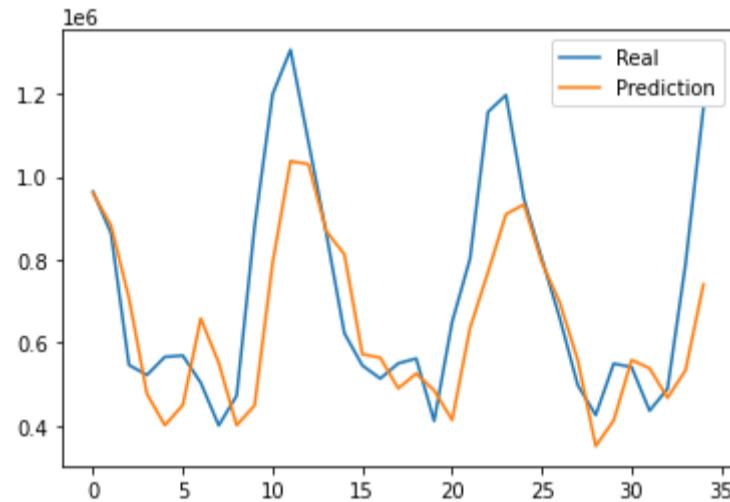


Figure 6: Prediction results for scenario two

5.3 Scenario three (regional temperature dataset)

The final MAE of the train set is around 0.13, and the one of the test set is around 0.14. Also, the RMSE is around 207585. While the tendency to follow high and low seems similar, but changes in the middle seem to be misleading.

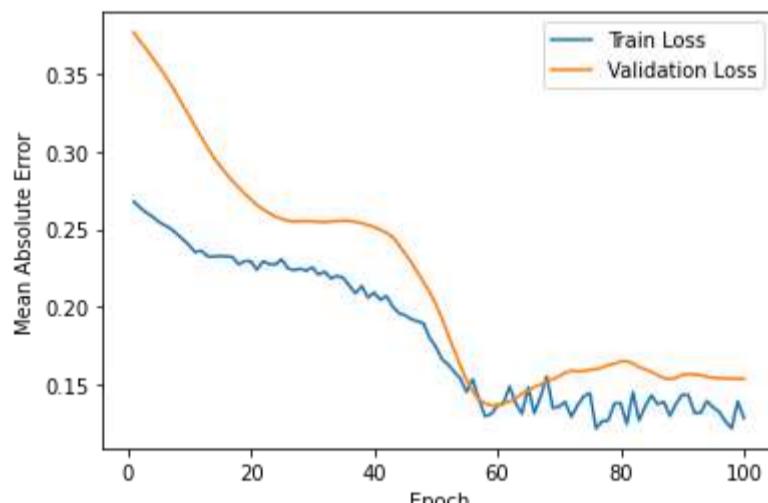


Figure 7: Loss for scenario three

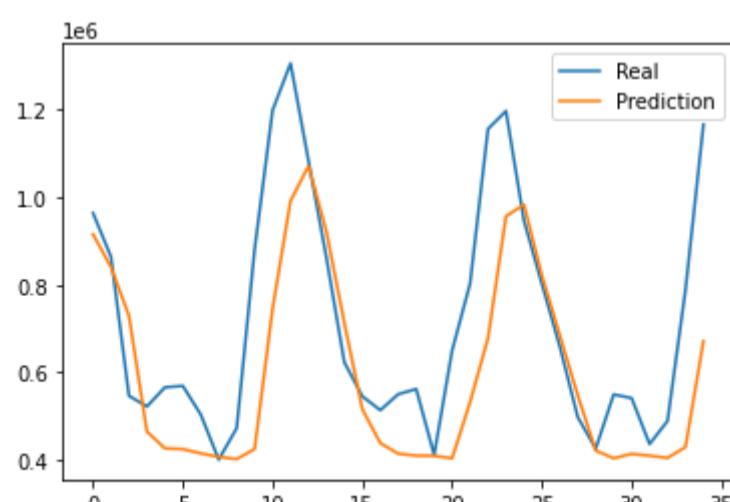


Figure 8: Prediction results for scenario three

5.4 Scenario four(applying timesteps)

The final MAE of the train set is around 0.06, and the one of the test set is around 0.30. Also, the RMSE is around 340843. Out of all scenarios, the predictive graph shows to have the most differences. However, in the last part, there is a somewhat akin tendency.

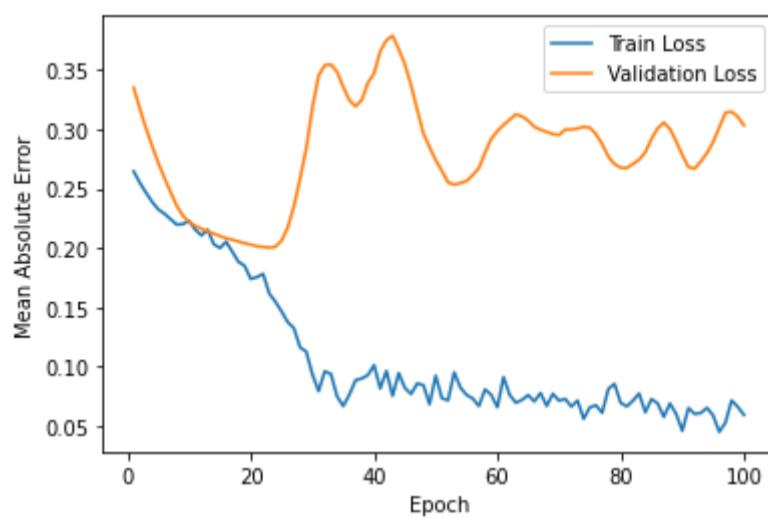


Figure 9: Loss for scenario four

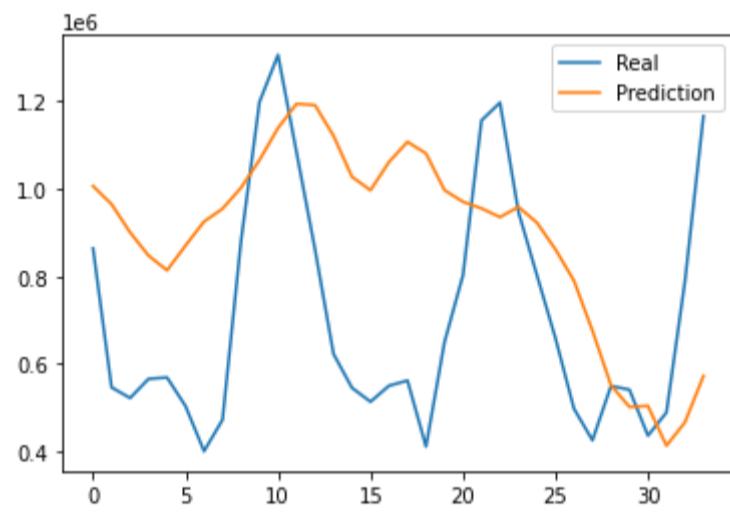


Figure 10: Prediction results for scenario four

5.5 Scenario five(national dataset)

The final MAE of the train set is around 0.03 and the one of test set is around 0.14. Also, the RMSE between real data and predicted data is around 587340. Tremendous RMSE value results, but direct comparisons are not possible because the baseline volume is different from other scenarios. Although the predictive graph shows discrepancy, it tends to be similar to the results in scenario two.

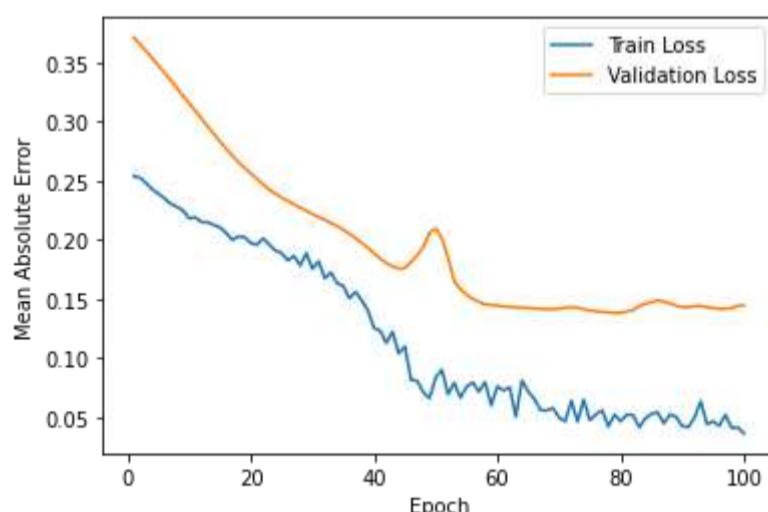


Figure 11: Loss for scenario five

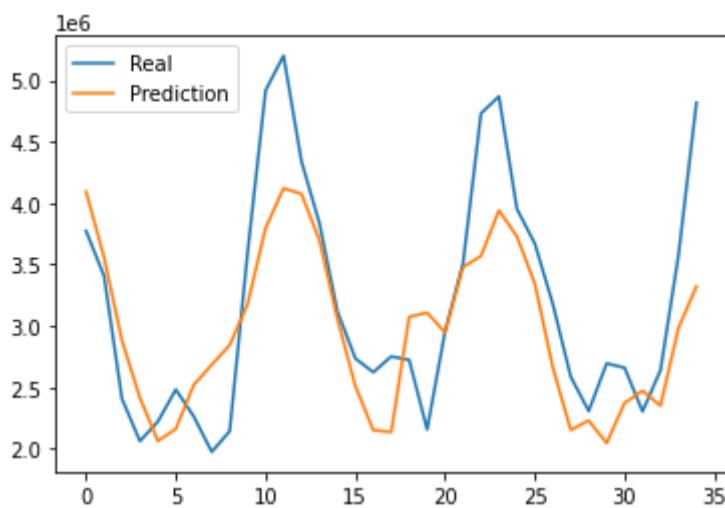


Figure 12: Prediction results for scenario five

5.6 Overall results

Out of the five scenarios in total, the second and third have smaller RMSE than others, and the graphs also show relatively similar results. The first and fourth show differences in the beginning and similar trends in the last part. However, it is noteworthy that the gap at the beginning of them is very large, but it tends to shrink together at the point of decline and stretch together at the point of increase.

In the first and fifth scenarios, all data are identical except that they differ in regional scale in temperature and precipitation. It is also the same that twelve months of data are used as the training set. From the subtle differences in the shape of the resulting graph, it can be seen that the national average data cannot represent the situation in a particular region, and the amount of NG supply differs depending on the circumstances in the region.

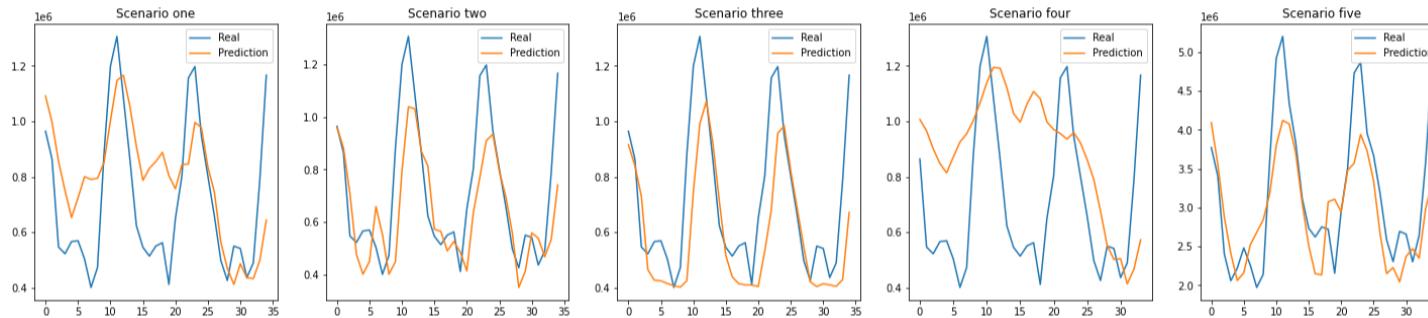


Figure 13: Total prediction results: 12 months training set

After changing the training set from twelve months to twenty-four months, the results are more clearly visible. The second and third prediction graphs have a more similar shape and the RMSE value decreases than the previous setting. The results of other scenarios show that the overall shape has improved; contrarily, the shape of the rapidly changing middle part is better in the previous condition.

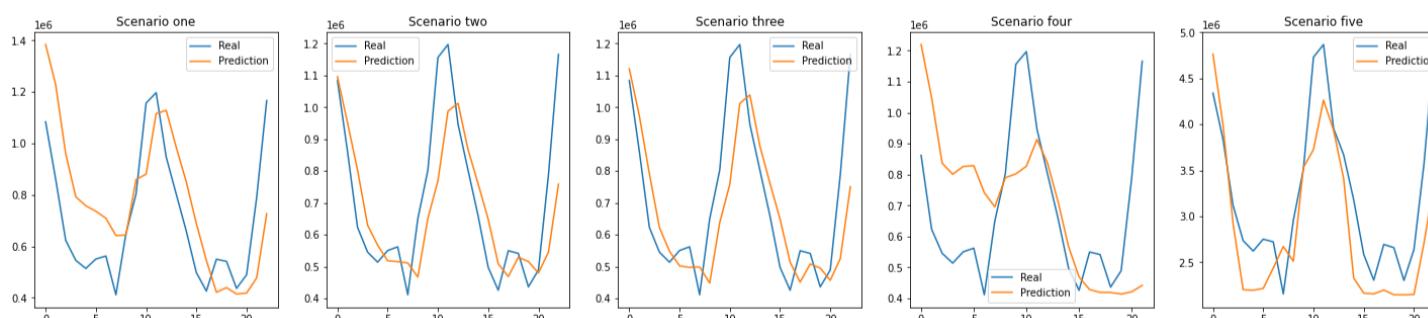


Figure 14: Total prediction results: 24 months training set

6. Benchmarks

For a benchmark, the Cloudmesh StopWatch and Benchmark [15](#) is used to measure the program's performance. The time spent on data load, data preprocessing, network model compile, training, and the prediction was separately measured, and the overall time for execution of all scenarios is around 77 seconds. It can be seen that The training time for the fourth scenario is the longest, and the one for the fifth scenario is the shortest.

Name	status	Time	Sum	Start	tag	Node	User	OS	Version
data-load	ok	0.051	0.051	2021-05-02 08:10:29	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
data-preprocess_1	ok	0.013	0.013	2021-05-02 08:10:31	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
data-preprocess_2	ok	0.009	0.076	2021-05-02 08:25:42	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
compile	ok	0.635	5.718	2021-05-02 08:25:42	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
train	ok	11.979	97.234	2021-05-02 08:25:44	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
predict	ok	0.954	7.557	2021-05-02 08:25:56	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test1-data-preprocess_1	ok	0.003	0.017	2021-05-02 08:26:47	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test1-data-preprocess_2	ok	0.016	0.031	2021-05-02 08:26:47	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test2-compile	ok	0.629	3.637	2021-05-02 08:27:21	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test1-train	ok	11.914	35.913	2021-05-02 08:26:54	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test1-predict	ok	0.945	3.791	2021-05-02 08:27:09	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test2-data-preprocess_1	ok	0.004	0.007	2021-05-02 08:27:13	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test2-data-preprocess_2	ok	0.008	0.014	2021-05-02 08:27:14	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test2-train	ok	12.468	24.362	2021-05-02 08:27:23	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test2-predict	ok	0.975	1.884	2021-05-02 08:27:39	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test3-data-preprocess_1	ok	0.001	0.006	2021-05-02 08:29:07	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test3-data-preprocess_2	ok	0.006	0.025	2021-05-02 08:29:08	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test3-compile	ok	0.602	5.484	2021-05-02 08:31:36	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test3-train	ok	20.282	138.311	2021-05-02 08:31:37	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test3-predict	ok	0.979	7.888	2021-05-02 08:31:57	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test4-data-preprocess_1	ok	0.019	0.032	2021-05-02 08:31:58	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test4-data-preprocess_2	ok	0.009	0.016	2021-05-02 08:31:59	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test4-compile	ok	0.576	2.416	2021-05-02 08:32:52	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test4-train	ok	11.858	47.539	2021-05-02 08:32:53	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	
test4-predict	ok	1.777	5.596	2021-05-02 08:33:05	d91aa3bf059f	collab	Linux	#1 SMP Thu Jul 23 08:00:38 PDT 2020	

Figure 15: Benchmarks

7. Conclusion

From the results of this project, it can be seen that simplifying factors that have a significant impact shows better efficiency than combining various factors. For example, NG consumption tends to increase for heating in cold weather. In addition, there is much precipitation in warm or hot weather; on the contrary, there is relatively little precipitation in the cold weather. It can be seen that these seasonal elements show relatively high consistency for affecting prediction when those are used as training datasets. Also, the predictions are derived more effectively when the seasonal datasets are combined.

However, in training set with a duration of twelve months, the last part of the scenario tends to match the actual data despite using the dataset combined with various factors that appears to be seasonally unrelated. Furthermore, when the training set is doubled on the same dataset, it can be seen that the differences between the actual and prediction graph are decreased than the result of a smaller training set. Based on this, it can be expected that the results could vary if a large amount of dataset with a more extended period is used and the ratio of the training set is appropriately adjusted.

South Korea imports a large amount of its energy resources. Also, the plan for energy demand and supply is being made and operated through nation-led policies. Ironically, the government's plan also shows a sharp change in direction with recent environmental issues, and the volatility of demand in the energy market is increasing than before. Therefore, methodologies for accurate forecasting of energy demand will need to be complemented and developed constantly to prepare for and overcome this variability.

In this project, Forecasting NG demand and supply was carried out using various data factors such as weather and price that is relatively easily obtained than the datasets which are complex economic indicators or classified as confidential. Nevertheless, state-of-the-art deep learning methods show that it has the flexibility and potential to forecast NG demand through the tendency of the results that indicate a relatively consistent with the actual data. From this point of view, it is thought that the research on NG in South Korea should be conducted in an advanced form by utilizing various data and more specialized analysis.

8. Acknowledgments

The author would like to thank Dr. Gregor von Laszewski for his invaluable feedback, continued assistance, and suggestions on this paper, and Dr. Geoffrey Fox for sharing his expertise in Deep Learning and Artificial Intelligence applications throughout this Deep Learning Application: AI-First Engineering course offered in the Spring 2021 semester at Indiana University, Bloomington.

9. Source code

The source code for all experiments and results can be found here as [ipynb link](#) and as [pdf link](#).

10. References

1. 2020 Monthly Energy Statistics, [Online resource]
<http://www.keei.re.kr/keei/download/MES2009.pdf>, Sep. 2020 
 2. KOGAS profile, [Online resource] <https://www.kogas.or.kr:9450/eng/contents.do?key=1498> 
 3. LNG production phase, [Online resource] <https://www.kogas.or.kr:9450/portal/contents.do?key=2014> 
 4. NG wholesale charges, [Online resource] <https://www.kogas.or.kr:9450/portal/contents.do?key=2026> 
 5. Natural gas explained, [Online resource], <https://www.eia.gov/energyexplained/natural-gas/factors-affecting-natural-gas-prices.php>, Aug, 2020 
 6. A. Khotanzad and H. Elragal, "Natural gas load forecasting with combination of adaptive neural networks," IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339), 1999, pp. 4069-4072 vol.6, doi: 10.1109/IJCNN.1999.830812. 
 7. A. Khotanzad, H. Elragal and T. . -L. Lu, "Combination of artificial neural-network forecasters for prediction of natural gas consumption," in IEEE Transactions on Neural Networks, vol. 11, no. 2, pp. 464-473, March 2000, doi: 10.1109/72.839015. 
 8. M. Akpinar, M. F. Adak and N. Yumusak, "Forecasting natural gas consumption with hybrid neural networks — Artificial bee colony," 2016 2nd International Conference on Intelligent Energy and Power Systems (IEPS), 2016, pp. 1-6, doi: 10.1109/IEPS.2016.7521852. 
 9. A. Anagnostis, E. Papageorgiou, V. Dafopoulos and D. Bochtis, "Applying Long Short-Term Memory Networks for natural gas demand prediction," 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), 2019, pp. 1-7, doi: 10.1109/IISA.2019.8900746. 
 10. NG supply dataset, [Online resource], <https://www.data.go.kr/data/15049904/fileData.do>, Apr, 2020 
 11. Regional climate dataset, [Online resource]
<https://data.kma.go.kr/climate/RankState/selectRankStatisticsDivisionList.do?pgmNo=179> 
 12. Crude oil orice dataset, [Online resource] <https://www.pronet.co.kr/main2.jsp> 
 13. Bituminous coal price dataset, [Online resource]
https://www.kores.net/komis/price/mineralprice/ironoreenergy/pricetrend/baseMetals.do?mc_seq=3030003&mnrl_pc_mc_seq=506 
 14. Won-Dollar exchange rate dateset, [Online resource]
<http://ecos.bok.or.kr/flex/EasySearch.jsp?langGubun=K&topCode=022Y013> 
 15. Gregor von Laszewski, Cloudmesh StopWatch and Benchmark from the Cloudmesh Common Library, [GitHub] <https://github.com/cloudmesh/cloudmesh-common> 
-

Project: Structural Protein Sequences Classification

The goal of this project is to predict the family of a protein based on the amino acid sequence of the protein. The structure and function of a protein are determined by the amino acid sequence that composes it. In the protein structure data set, each protein is classified according to its function. Categories include: HYDROLASE, OXYGEN TRANSPORT, VIRUS, SIGNALING PROTEIN, etc. dozens of kinds. In this project, we will use nucleic acid sequences to predict the type of protein. Although there are already protein search engines such as BLAST that can directly query the known protein families. But for unknown proteins, it is still important to use deep learning algorithms to predict their functions. Protein classification is a simpler problem than protein structure prediction. The latter requires the complete spatial structure of the protein, and the required deep learning model is extremely complex.

Tags: [project](#) [ai](#) [biology](#)

⌚ 7 minute read

 Check Report passing  Status passing Status: final, Type: Project

Jiayu Li, [sp21-599-357](#), [Edit](#)

- Code:
 - [benchmark.py](#)
 - [Lstm.ipynb](#)

Abstract

The goal of this project is to predict the family of a protein based on the amino acid sequence of the protein. The structure and function of a protein are determined by the amino acid sequence that composes it. In the protein structure data set, each protein is classified according to its function. Categories include: HYDROLASE, OXYGEN TRANSPORT, VIRUS, SIGNALING PROTEIN, etc. dozens of kinds. In this project, we will use nucleic acid sequences to predict the type of protein.

Although there are already protein search engines such as BLAST[^1] that can directly query the known protein families. But for unknown proteins, it is still important to use deep learning algorithms to predict their functions.

Protein classification is a simpler problem than protein structure prediction[^7]. The latter requires the complete spatial structure of the protein, and the required deep learning model is extremely complex.

Contents

- [1. Introduction](#)
- [2. Dataset](#)
- [3. Deep learning algorithm](#)
 - [3.1 Word Embedding](#)
 - [3.2 LSTM](#)
- [4. Benchmark](#)
 - [4.1 Compare with test benchmark](#)
 - [4.2 The impact of the number of labels on accuracy](#)

- [5. Conclusion](#)
- [6. Acknowledgments](#)
- [7. References](#)

Keywords: Protein Sequences, Deep learning

1. Introduction

The structure and function of a protein are determined by the amino acid sequence that composes it. The amino acid sequence can be regarded as a language composed of 4 different characters. In recent years, due to the development of deep learning, the ability of deep neural networks to process natural language has reached or even surpassed humans in some areas. In this project, we tried to treat the amino acid sequence as a language and use the existing deep learning model to analyze it to achieve the purpose of inferring its function.

The data sets used in the project come from Research Collaboratory for Structural Bioinformatics (RCSB) and Protein Data Bank (PDB)¹.

The data set contains approximately 400,000 amino acid sequences and has been labeled. The label is the family to which the protein belongs. The protein family includes HYDROLASE, HYDROLASE/HYDROLASE INHIBITOR, IMMUNE SYSTEM, LYASE, OXIDOREDUCTASE, etc. Therefore this problem can be regarded as a classification problem. The input of the model is a sequence, the length of the sequence is uncertain, and the output of the model is one of several categories. By comparing DNN, CNN, LSTM and other common models, we have achieved effective prediction of protein energy supply.

2. Dataset

PDB is a data set dedicated to the three-dimensional structure of proteins and nucleic acids. It has a very long history, dating back to 1971. In 2003, PDB developed into an international organization wwPDB. Other members of wwPDB, including PDBe (Europe), RCSB (United States), and PDBj (Japan) also provide PDB with a center for data accumulation, processing and release. Although PDB data is submitted by scientists from all over the world, each piece of data submitted will be reviewed and annotated by wwPDB staff, and whether the data is reasonable or not. The PDB and the software it provides are now free and open to the public. In the past few decades, the number of PDB structures has grown at an exponential rate.

Structural biologists around the world use methods such as X-ray crystallography, NMR spectroscopy, and cryo-electron microscopy to determine the position of each atom relative to each other in the molecule. Then they will submit this structural information, wwPDB will annotate it and publish it to the database publicly.

PDB supports searching for ribosomes, oncogenes, drug targets, and even the structure of the entire virus. However, the number of structures archived in the PDB is huge, and finding the information may be a difficult task.

The information in the PDB data set mainly includes: protein/nucleic acid source, protein/nucleic acid molecule composition, atomic coordinates, experimental methods used to determine the structure. Structural Protein Sequences Dataset: <https://www.kaggle.com/shahir/protein-data-set/code>

Protein dataset classification: <https://www.kaggle.com/rafay12/anti-freeze-protein-classification>

RCSB PDB: <https://www.rcsb.org/>

structureId	chainId	sequence	# residueCo...	macromol...
11AS	B	MKTAYIAKQRQISFV KSHFSRQLEERLGLI EVQAPILSRVGDTQ DNLSGAEKAVQVKVK ALPDAQFEVVHSLAK WKRQTLGQHDFSAGE GLYTHMKALR...	660	Protein
11BA	A	KESAAAKFERQHMDS GNSPSSSSNYCNLMM CCRKMTQGKCKPVNT FVHESLADVKAACSQ KKVTCKNGQTNCYQS KSTMRITDCRETGSS KYPNCAYKTT...	248	Protein
11BA	B	KESAAAKFERQHMDS GNSPSSSSNYCNLMM CCRKMTQGKCKPVNT FVHESLADVKAACSQ KKVTCKNGQTNCYQS KSTMRITDCRETGSS KYPNCAYKTT...	248	Protein
11BG	A	KESAAAKFERQHMDS GNSPSSSSNYCNLMM CCRKMTQGKCKPVNT FVHESLADVKAACSQ KKVTCKNGQTNCYQS KSTMRITDCRETGSS KYPNCAYKTT...	248	Protein

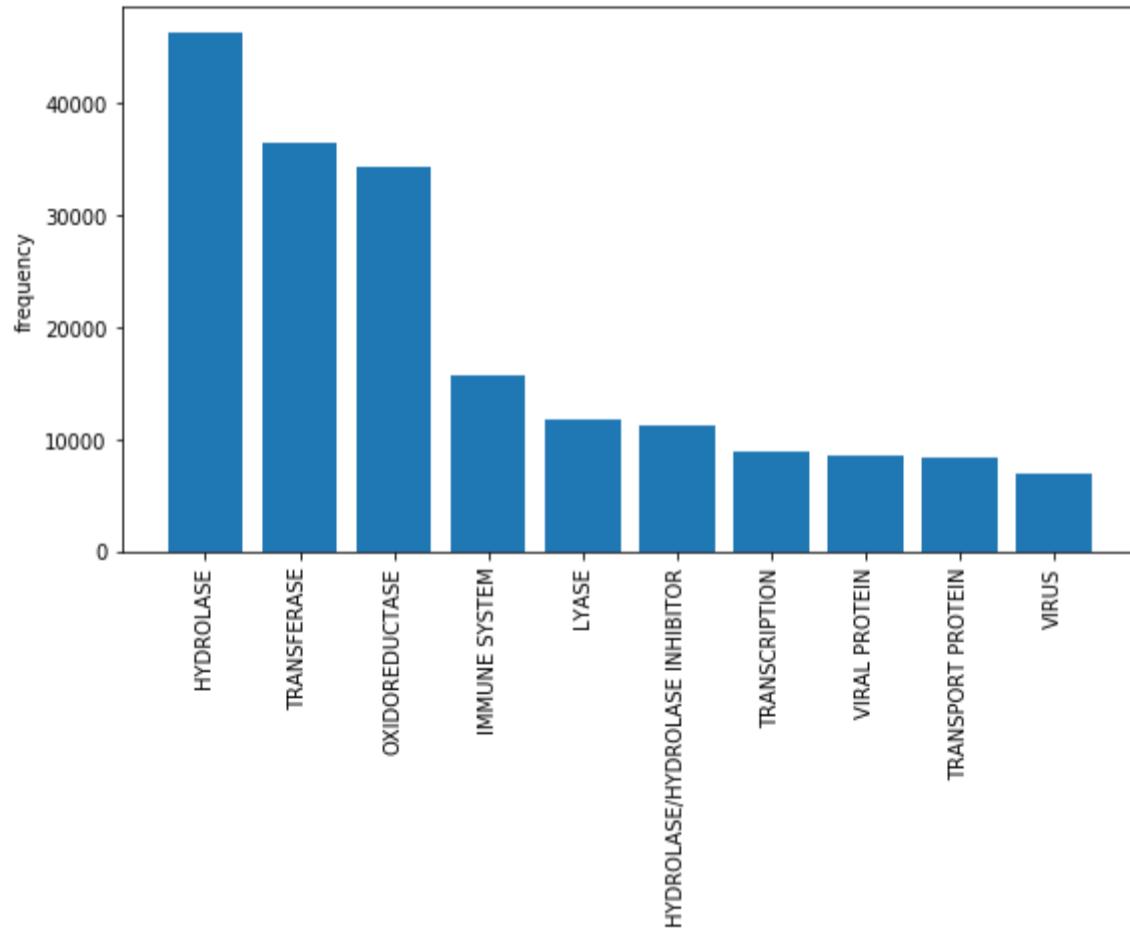
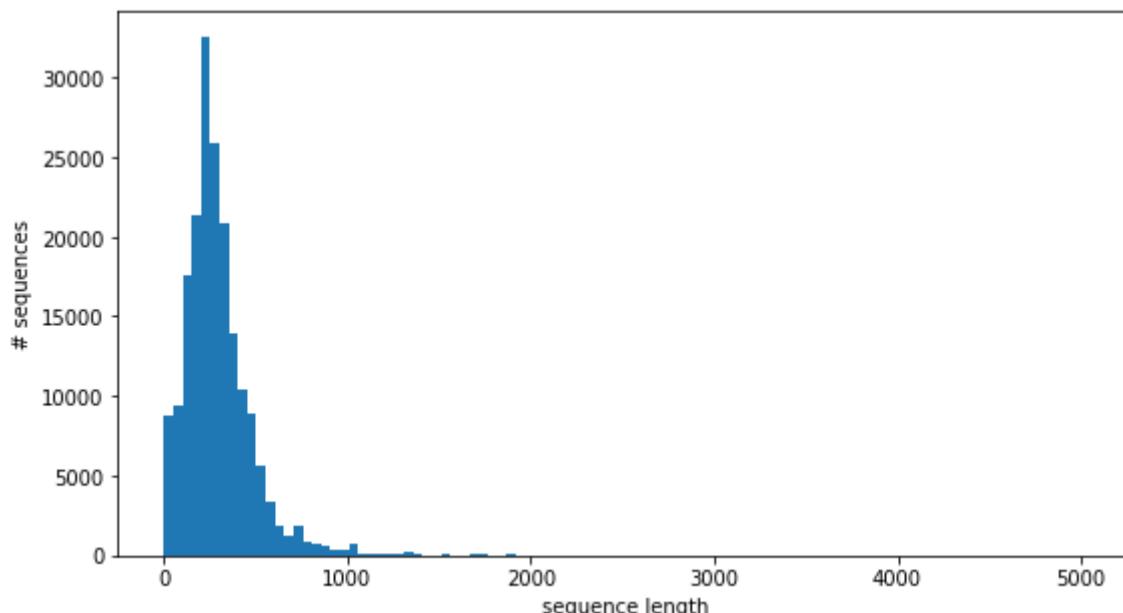
Figure 1: Data set sample.**Figure 2:** The frequency of appearance of different labels.

Figure 3: The length distribution of amino acid sequences.

3. Deep learning algorithm

Two deep learning models, CNN and LSTM², are mainly used in this project. In addition, the Word Embedding algorithm is also used to preprocess the data.

Among these models, the CNN model comes from the Kaggle website³ and will be used as a test benchmark. We will try to build a simpler but more accurate model with an accuracy rate of at least not lower than the test benchmark.

3.1 Word Embedding

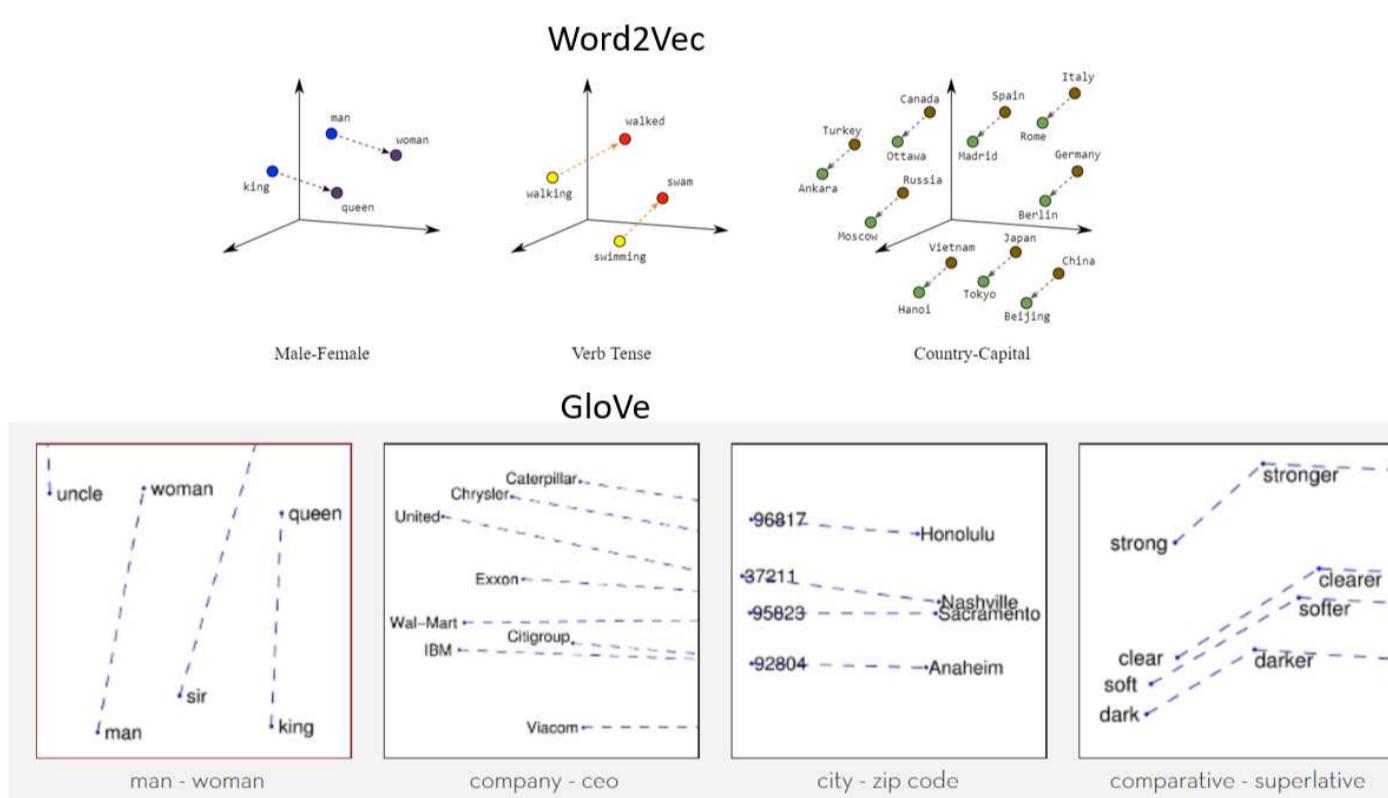
A word embedding is a class of approaches for representing words and documents using a dense vector representation.

In an embedding, words are represented by dense vectors where a vector represents the projection of the word into a continuous vector space.

The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used.

The position of a word in the learned vector space is referred to as its embedding.

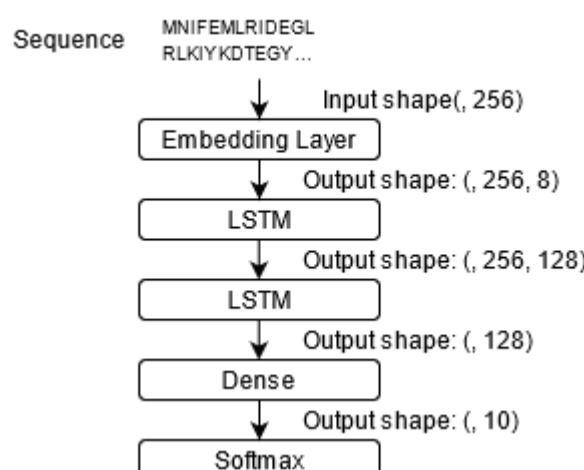
Two popular examples of methods of learning word embeddings from text include: Word2Vec⁴ and GloVe⁵.

**Figure 4:** Word2Vec and GloVe.

3.2 LSTM

Recurrent Neural Network (RNN) is a neural network used to process sequence data. Compared with the general neural network, it can process the data of sequence changes. For example, the meaning of a word will have different meanings because of the different content mentioned above, and RNN can solve this kind of problem well.

Long short-term memory (LSTM) is a special kind of RNN, mainly to solve the problem of gradient disappearance and gradient explosion during long sequence training. Compared to ordinary RNN, LSTM can perform better in longer sequences.

**Figure 5:** LSTM.

4. Benchmark

4.1 Compare with test benchmark

[3](#) is a highly accurate model on the Kaggle website. It is currently one of the models with the highest accuracy on this data set. In this project, CNN model in [3](#) will be used as a test benchmark. We use categorical cross entropy as loss function.

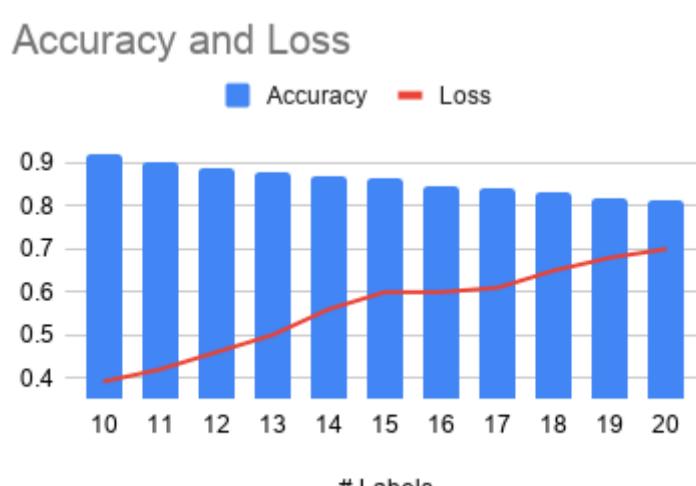
Model	CNN	LSTM
#parameters	273,082	203,226
Accuracy	91.6%	91.9%
Training time	7ms/step	58ms/step
Batch size	128	256
Loss	0.4051	0.3292

Both the CNN model and the LSTM model use the word embedding layer for data dimensionality reduction. After testing, the result is that LSTM uses fewer parameters to achieve the same or even slightly higher accuracy than the benchmark. However, due to the relatively complex structure of LSTM, its training speed and guessing speed are slower.

In order to make a fair comparison with the test benchmark, we only selected the 10 most frequent samples as the data set, which is also the original author's choice. Therefore, the accuracy of more than 90% here is only of relative significance, and does not mean that the same accuracy can be achieved in practical applications (the data set usually has more categories).

4.2 The impact of the number of labels on accuracy

In order to further test the performance of LSTM on different data sets, we further increased the number of labels, gradually increasing from 10 labels to 20 labels. Figure 6 shows the effect of the number of labels on accuracy.

**Figure 6:** The impact of the number of labels on accuracy.

Note that due to the limitation of the data set, the number of samples belonging to different labels is different. If we want to balance different categories, we have to shrink the data set, which will affect the accuracy. This is one of the limitations of this test.

5. Conclusion

The deep learning model gives a prediction accuracy of more than 90% for the 10 most common protein types. If the number of label is increased to 20, the accuracy rate will drop to 80%. The traditional machine learning model is difficult to deal with string data of different lengths.

6. Acknowledgments

The author would like to thank Dr. Gregor von Laszewski for his invaluable feedback on this paper, and Dr. Geoffrey Fox for sharing his expertise in Big Data applications throughout this course.

7. References

1. Sussman, Joel L., et al. "Protein Data Bank (PDB): database of three-dimensional structural information of biological macromolecules." *Acta Crystallographica Section D: Biological Crystallography* 54.6 (1998): 1078-1084. 
 2. Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney. "LSTM neural networks for language modeling." Thirteenth annual conference of the international speech communication association. 2012. [_](#)
 3. Kaggle, Protein Sequence Classification. <https://www.kaggle.com/helmehelmuto/cnn-keras-and-investigate> [_](#)
 4. Church, Kenneth Ward. "Word2Vec." *Natural Language Engineering* 23.1 (2017): 155-162. [_](#)
 5. Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014. 
-

Project: Training A Vehicle Using Camera Feed from Vehicle Simulation

Deep Learning has become the main form of machine learning that has been used to train, test, and gather data for self-driving cars. The CARLA simulator has been developed from the ground up so that researchers who normally do not have the capital to generate their own data for self-driving vehicles can do so to fit their specific model. CARLA provides many tools that can simulate many scenarios that an autonomous vehicle would run into. The benefit of CARLA is that it can simulate scenarios that may be too dangerous for a real vehicle to perform, such as a full self-driving car in a heavily populated area. CARLA has the backing of many companies who lead industry like Toyota who invested \$100,000 dollars in 2018. This project uses the CARLA simulator to visualize how a real camera system based self-driving car sees obstacles and objects.

Tags: [project](#) [ai](#) [transportation](#)

⌚ 9 minute read

 Check Report passing  Status passing Status: final, Type: Project

Jesus Badillo, [sp21-599-358](#), [Edit](#)

- Code:
 - [tutorialEgo.py](#)

Abstract

Deep Learning has become the main form of machine learning that has been used to train, test, and gather data for self-driving cars. The CARLA simulator has been developed from the ground up so that researchers who normally do not have the capital to generate their own data for self-driving vehicles can do so to fit their specific model. CARLA provides many tools that can simulate many scenarios that an autonomous vehicle would run into. The benefit of CARLA is that it can simulate scenarios that may be too dangerous for a real vehicle to perform, such as a full self-driving car in a heavily populated area. CARLA has the backing of many companies who lead industry like Toyota who invested \$100,000 dollars in 2018 [^6]. This project uses the CARLA simulator to visualize how a real camera system based self-driving car sees obstacles and objects.

Contents

- [1. Introduction](#)
- [2. Using the CARLA Simulator](#)
 - [2.1 Existing Work on Carla](#)
- [3. Using the TensorFlow Object Detection API](#)
- [4. Implementation](#)
 - [4.1 System Requirements](#)
 - [4.2 Download and Install CARLA](#)
 - [Download for CARLA version 0.9.9](#)
 - [4.3 Download and Install TensorFlow Object Detection API](#)
 - [From the Downloads folder clone the TensorFlow models git](#)
 - [4.4 Download Protobuf](#)
 - [The link to the ProtoBuf repository download is shown below](#)
 - [Run the pwd command from powershell and get the path from root to Downloads folder](#)
 - [When running the command make sure that you are in ~/Downloads/models-master/research'](#)

- [Make sure that you are in Downloads/models-master/research when running this command](#)
- [Test Installation](#)
- [Test Success](#)
- [4.5 Running Carla With Object Detection](#)
 - [Run CARLA Client](#)
 - [Run Carla Object Detection Program](#)
- [5. Training Model](#)
- [6. Results](#)
- [7. Benchmark](#)
- [8. Conclusion](#)
- [9. Acknowledgments](#)
- [9. References](#)

Keywords: tensorflow, example.

1. Introduction

Making cars self driving has been a problem that many car companies have been trying to tackle in the 21st century. There are many different approaches that have been used which all involve deep learning. The approaches all train data that are gathered from a variety of sensors working together. Lidar and computer vision are the main sensors that are used by commercial companies. Tesla uses video gathered from multiple cameras to train their neural network ¹ which is known as HydraNet. In this project, a simulation of a real driving vehicle with a camera feed will be used to see the objects that a car would need to see to train the vehicle to be self-driving

2. Using the CARLA Simulator

The CARLA simulator which uses the driver inputs and puts into a driving log which contains data of the trajectory and the surroundings of the simulated vehicle. The CARLA simulator uses the steering angle and throttle to act much like the controllable inputs of a real vehicle. CARLA is an open-source and has been developed from the ground up to support development, training, and validation of autonomous driving systems. In addition to open-source code and protocols, CARLA provides open digital urban layouts, buildings, and vehicles that were created for this purpose and can be used freely. The simulation platform supports flexible specification of sensor suites, environmental conditions, full control of all static and dynamic actors, maps generation ². The simulation will be created by driving the vehicle in the simulator and using the camera feed so that the neural network can be trained. Driving in the simulator looks much like Figure 1.



Figure 1 Driving in Carla Simulator ³

2.1 Existing Work on Carla

The tutorials over Carla from the youtuber SentDex provide a good introduction into projects that could use deep learning to train self-driving cars. His tutorials provide a good insight into the Carla Environment so that one could perform their own study ⁴.

3. Using the TensorFlow Object Detection API

The Tensorflow object detection API is used to classify objects with a specific level of confidence. Image recognition is useful for self-driving cars because it can provide known obstacles where the vehicle is prohibited from traveling. The API has been trained on the COCO dataset which is a dataset consisting of about 300,000 of 90 of the most commonly found objects. Google provided this API to improve the state of the Computer vision field. Figure2 shows how the bounding boxes classify images using the object detection API.

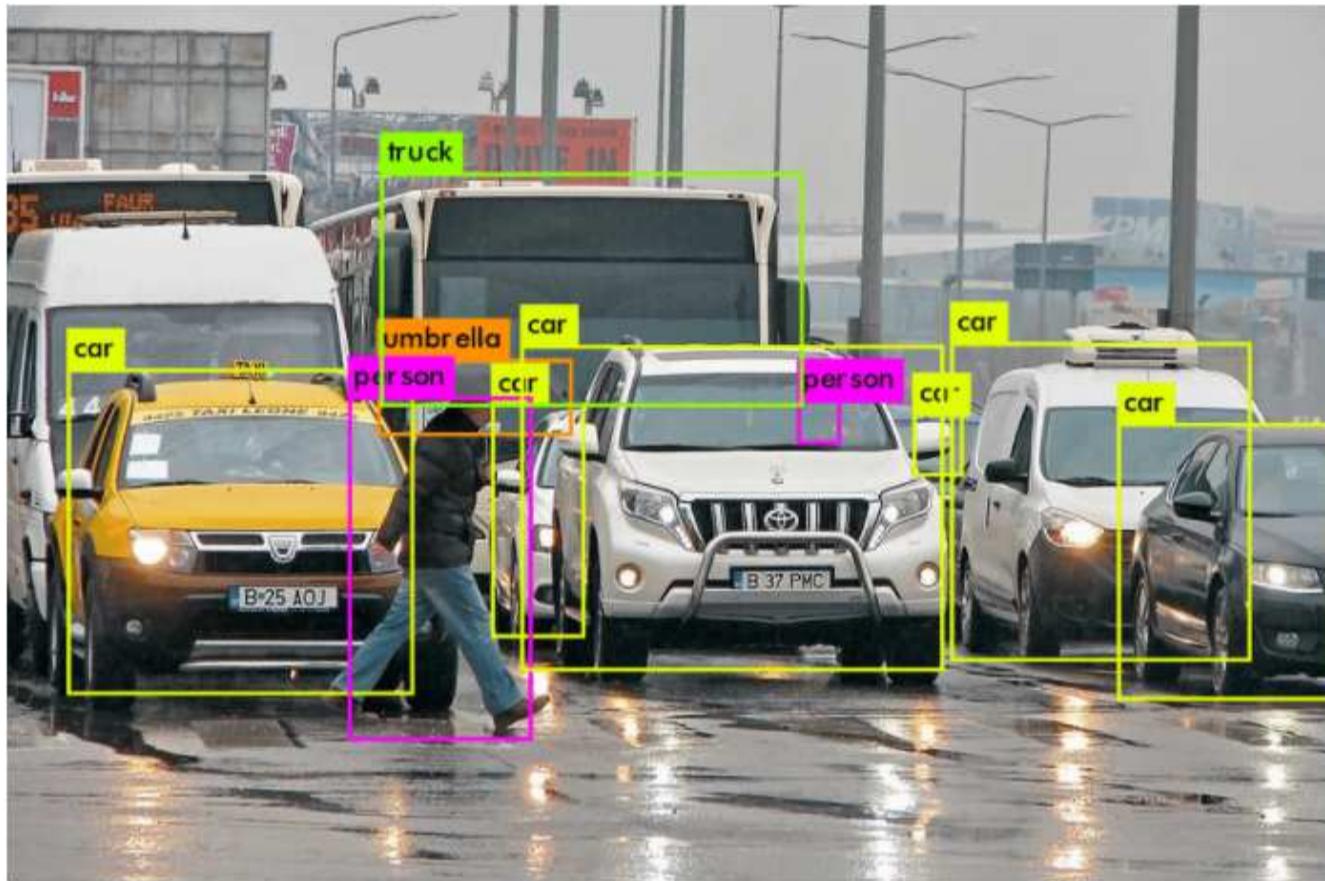


Figure 2 Obect Detection for Cars ⁵

4. Implementation

4.1 System Requirements

This project uses windows 10 along with visual studio code and python 3.7. Note that this project may work with other systems, but CARLA is a resource intensive program.

OS Version	GPU	RAM
Windows 10.0.18363 Build 18363	NVIDIA GTX 1660 Super	32 GB

In this study the CARLA version 0.9.9 is being used along with python 3.7 to control the simulated vehicle in the CARLA simulator.

4.2 Download and Install CARLA

Download for CARLA version 0.9.9

<https://github.com/carla-simulator/carla/releases/tag/0.9.9> ⁶

The file to download is shown below:

CARLA_0.9.9.zip

Make sure to download the compiled version for Windows. The Carla Simulator is around 25GB, so to replicate the study one must have 30-50GB of free disk space. Once the file is finished downloading, extract the content of the CARLA zip file into the Downloads folder.

4.3 Download and Install TensorFlow Object Detection API

From the Downloads folder clone the TensorFlow models git

```
git clone https://github.com/tensorflow/models.git
```

Make sure to clone this git repository into the Downloads folder of your windows machine

4.4 Download Protobuf

The link to the ProtoBuf repository download is shown below

<https://github.com/protocolbuffers/protobuf/releases/download/v3.16.0/protoc-3.16.0-win64.zip> ⁷

The Tensorflow Object Detection API uses Protobufs to configure model and training parameters. Before the framework can be used, the Protobuf libraries must be downloaded and compiled ⁸. Make sure that you extract the file to the Downloads folder. To configure the model within the directory structure run the commands below.

Run the pwd command from powershell and get the path from root to Downloads folder

```
pwd
```

When running the command make sure that you are in
'~/Downloads/models-master/research'

```
'PathFromDownloads/Downloads'/protoc object_detection/protos/*.proto --python_out=.
```

The command shown above configures protobuf so that the object detection API could be used. Make sure you are in the Downloads/models-master/research path. Run the commands below to install all of the necessary packages to run the object detection API.

Make sure that you are in Downloads/models-master/research when running this command

```
cp object_detection/packages/tf2/setup.py .
python -m pip install .
```

After installing the packages test your installation from the Downloads/models-master/research path and run the command below.

Test Installation

```
python object_detection/builders/model_builder_tf2_test.py
```

Test Success

If the test was successful than you will a result similar to the one showed in Figure 3.

```
[ OK ] ModelBuilderTF2Test.test_create_ssd_models_from_config
[ RUN ] ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.0s
I0507 11:17:52.188139 22932 test_util.py:2076] time(_main_.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.0s
[ OK ] ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
[ RUN ] ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
I0507 11:17:52.190131 22932 test_util.py:2076] time(_main_.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
[ OK ] ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
[ RUN ] ModelBuilderTF2Test.test_invalid_model_config_proto
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
I0507 11:17:52.190131 22932 test_util.py:2076] time(_main_.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
[ OK ] ModelBuilderTF2Test.test_invalid_model_config_proto
[ RUN ] ModelBuilderTF2Test.test_invalid_second_stage_batch_size
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
I0507 11:17:52.192126 22932 test_util.py:2076] time(_main_.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
[ OK ] ModelBuilderTF2Test.test_invalid_second_stage_batch_size
[ RUN ] ModelBuilderTF2Test.test_session
[ SKIPPED ] ModelBuilderTF2Test.test_session
[ RUN ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
I0507 11:17:52.194120 22932 test_util.py:2076] time(_main_.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
[ OK ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
[ RUN ] ModelBuilderTF2Test.test_unknown_meta_architecture
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
I0507 11:17:52.195118 22932 test_util.py:2076] time(_main_.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
[ OK ] ModelBuilderTF2Test.test_unknown_meta_architecture
[ RUN ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
INFO:tensorflow:time(_main_.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
I0507 11:17:52.196115 22932 test_util.py:2076] time(_main_.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
[ OK ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
-----
Ran 22 tests in 17.921s
```

Figure 3

4.5 Running Carla With Object Detection

The directory structure for the CARLA for the project shoud have protobuf, the tensorflow models-master directory, and the CARLA_0.9.9 directory all in the Downloads folder. To correctly run this project one would need to open two powershell windows and run the CARLA client and the file which is providid in this git repository called tutorialEgo.py. The two code snippets below show how to both programs

Run CARLA Client

```
"your path"\Downloads\CARLA_0.9.9\WindowsNoEditor> .\CarlaUE4.exe
```

Run Carla Object Detection Program

```
#Make sure to place the tutorialEgo.py in the examples folder from the downloaded carla
"your path"\Downloads\CARLA_0.9.9\WindowsNoEditorPythonAPI\examples> py -3.7 .\tutorialEgo.py
```

5. Training Model

Model Name	Speed	COCO mAP
ssd_mobilenet_v1_coco	fast	21
ssd_inception_v2_coco	fast	24
rfcn_resnet101_coco	medium	30
faster_rcnn_resnet101_coco	medium	32
faster_rcnn_inception_resnet_v2_astrous_coco	slow	37

To perform the object detection in the Cara simulator this project uses the TensorFlow object detection API. The model uses the COCO dataset which contains five different models each with a different mean average precision. The mean average precison, or mAP, is the product of precision and recall on detecting bounding boxes. The higher the mAP score, the more accurate the network is but that slows down the speed of the model [8](#). In this project the ssd_mobilenet_v1_coco model was used because it is the fastest of the 5 models providie for the COCO dataset.

6. Results

The accuracy of the model was not very good at detecting other scenery, but it was able to detect the most important obstacles for self-driving cars such as other vehicles, pedestrians, and traffic signals. The video below shows a simulation in the Carla simulated vehicle with object detection.

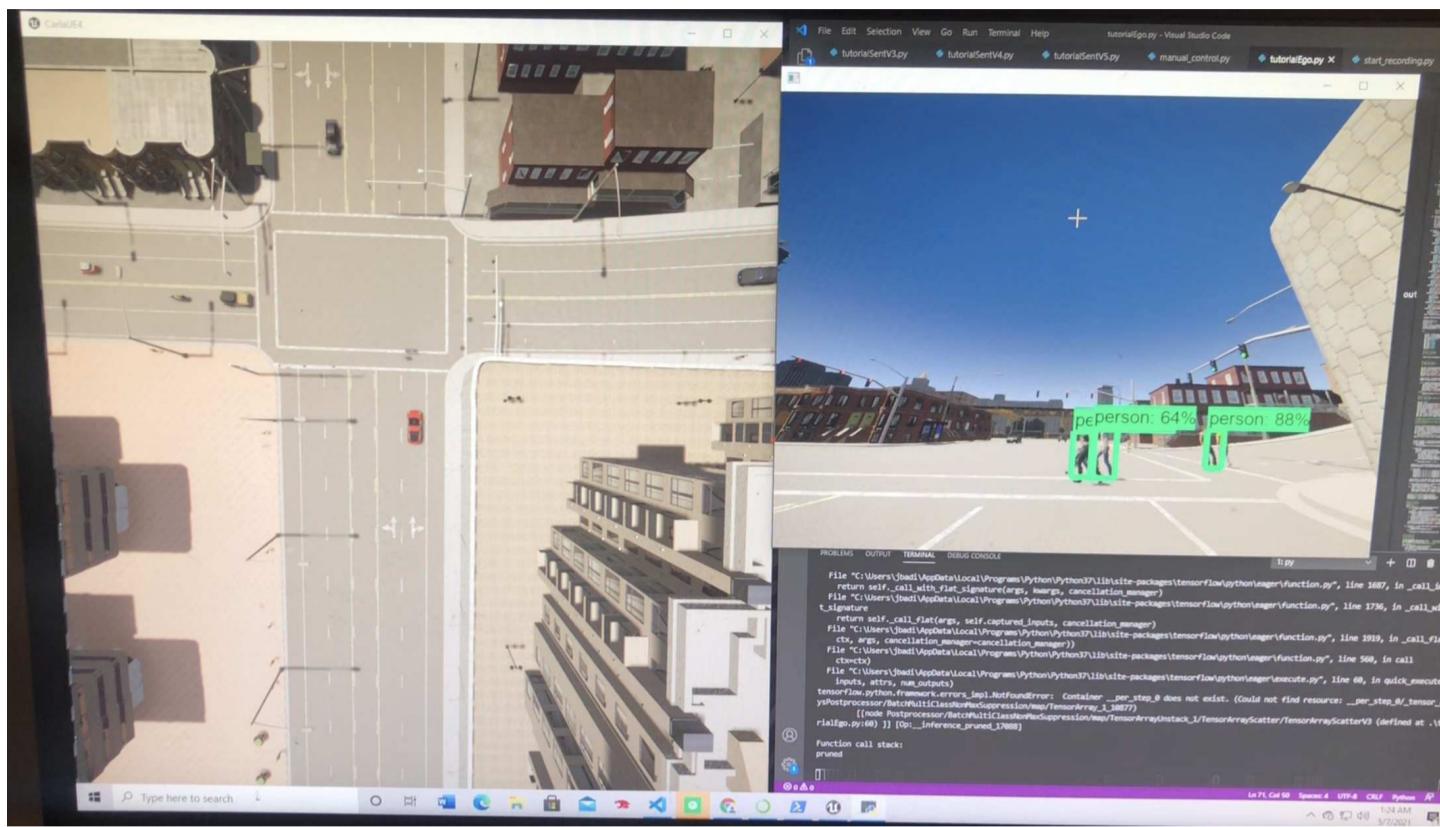


Figure 4 Object Detection in CARLA

https://drive.google.com/file/d/13RXly74APbwSqV_zBs_v59v4ZOnWdtrT/view?usp=sharing

7. Benchmark

The benchmark used for this project was the StopWatch function from the cloudmesh package [9](#). The function can see how long a particular section of code took compared to a different section in the program. In this project the section that took the longest was to setup pedestrian and traffic across the simulated city. This makes sense because there are many vehicles and pedestrians that need to be spawned while also pre computing there trajectories.

Attribute	Value
cpu_count	8
mem.available	5.0 GiB
mem.free	5.0 GiB
mem.percent	68.5 %
mem.total	15.9 GiB
mem.used	10.9 GiB
platform.version	('10', '10.0.18362', 'SP0', '')
python	3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2020, 10:41:24) [MSC v.1900 64 bit (AMD64)]
python.pip	21.1.1
python.version	3.7.7
sys.platform	win32
uname.machine	AMD64
uname.node	DESKTOP-FMK4ED6
uname.processor	Intel64 Family 6 Model 158 Stepping 13, GenuineIntel
uname.release	10
uname.system	Windows
uname.version	10.0.18362
user	jbadi

Name	Status	Time	Sum	Start	tag	Node	User	OS	Version
LoadWorld	ok	0.135	0.135	2021-05-07 20:58:32		DESKTOP-FMK4ED6	jbadi	Windows	('10', '10.0.18362', 'SP0', '')
CameraSetup	ok	0.06	0.06	2021-05-07 20:58:32		DESKTOP-FMK4ED6	jbadi	Windows	('10', '10.0.18362', 'SP0', '')
SetupTraffic	ok	6.681	6.681	2021-05-07 20:58:32		DESKTOP-FMK4ED6	jbadi	Windows	('10', '10.0.18362', 'SP0', '')
SetupSpectator	ok	0.169	0.169	2021-05-07 20:58:39		DESKTOP-FMK4ED6	jbadi	Windows	('10', '10.0.18362', 'SP0', '')

Figure 5

8. Conclusion

The ssd_mobilenet_v1_coco model did not perform as well as it could have because it sometimes classified some objects wrong. For example, some pedestrians walking produced shadows which the object detection models perceived as ski's. The mean average precision of the model was the lowest

of the models trained by the COCO dataset which played a factor in the accuracy of the model. This caused issues in the vehicle's detection of its surroundings. Overall, the model was good at classifying the main objects it needs to know to drive safely such as pedestrians and other vehicles. This project fulfilled its purpose by showcasing that it can use the object detection from the camera feed along with built in collision detector to be able to train a self-driving vehicle in CARLA.

9. Acknowledgments

The author of this project would like to thank Harrison Kinsley from the youtube channel SentDex for providing good resources for how to use deep learning using carla and tensorflow. The author would also like to thank Dr. Gregor von Laszewski for feedback on this report, and Dr. Geoffrey Fox for sharing his knowledge in Deep Learning and Artificial Intelligence throughout this course taught at Indiana University.

9. References

1. Explains architecture of Tesla's Neural Network,[Online Resource]
https://openaccess.thecvf.com/content_cvpr_2018/papers_backup/Mullapudi_HydraNets_Specialized_Dynamic_CVPR_2018_paper.pdf ↗
 2. Link to Carla website, [Online Resource] <http://carla.org/> __
 3. Documentation Explaining Key Features of Carla, [Online Resource]
https://carla.readthedocs.io/en/0.9.9/getting_started/ __
 4. Introduction to Carla with Python, [Online Resource]
<https://pythonprogramming.net/introduction-self-driving-autonomous-cars-carla-python/> __
 5. Object Detection Image, [Online Resource] https://www.researchgate.net/figure/Object-detection-in-a-dense-scene_fig4_329217107 __
 6. Link to the Carla_0.9.9 github, [GitHub] <https://github.com/carla-simulator/carla/releases/tag/0.9.9> __
 7. Protobuf github download, [GitHub]
<https://github.com/protocolbuffers/protobuf/releases/download/v3.16.0/protoc-3.16.0-win64.zip> __
 8. Explains differences between models being used for Object Detection and performance, [Online Resource] <https://towardsdatascience.com/is-google-tensorflow-object-detection-api-the-easiest-way-to-implement-image-recognition-a8bd1f500ea0> __
 9. Gregor von Laszewski, Cloudmesh Stopwatch and Benchmark from the Cloudmesh Common Library, [GitHub] <https://github.com/cloudmesh/cloudmesh-common> ↗
-

Project: Deep Learning in Drug Discovery

Machine learning has been a mainstay in drug discovery for decades. Artificial neural networks have been used in computational approaches to drug discovery since the 1990s. Under traditional approaches, emphasis in drug discovery was placed on understanding chemical molecular fingerprints, in order to predict biological activity. More recently however, deep learning approaches have been adopted instead of computational methods. This paper outlines work conducted in predicting drug molecular activity, using deep learning approaches.

Tags: [project](#) [ai](#) [health](#)

⌚ 12 minute read

 Check Report passing  Status passing Status: final, Type: Project

Anesu Chaora, [sp21-599-359](#), [Edit](#)

- Code: [predicting_molecular_activity.ipynb](#)

Abstract

Machine learning has been a mainstay in drug discovery for decades. Artificial neural networks have been used in computational approaches to drug discovery since the 1990s [^1]. Under traditional approaches, emphasis in drug discovery was placed on understanding chemical molecular fingerprints, in order to predict biological activity. More recently however, deep learning approaches have been adopted instead of computational methods. This paper outlines work conducted in predicting drug molecular activity, using deep learning approaches.

Contents

- [1. Introduction](#)
 - [1.1. De novo molecular design](#)
 - [1.2. Bioactivity prediction](#)
- [2. Related Work](#)
 - [2.1. Merck Molecular Activity Challenge on Kaggle](#)
 - [2.2. The Dataset](#)
 - [2.3. A Deep Learning Algorithm](#)
- [3. Project Implementation](#)
 - [3.1. Tools and Environment](#)
 - [3.2. Implementation Overview](#)
 - [3.3. Benchmarks](#)
 - [3.4. Findings](#)
- [4. Discussion](#)
- [5. Conclusion](#)
- [6. Acknowledgments](#)
- [7. Appendix](#)
- [References](#)

Keywords: Deep Learning, drug discovery.

1. Introduction

1.1. De novo molecular design

Deep learning (DL) is finding uses in developing novel chemical structures. Methods that employ variational autoencoders (VAE) have been used to generate new chemical structures. Approaches have involved encoding input string molecule structures, then reparametrizing the underlying latent variables, before searching for viable solutions in the latent space by using methods such as Bayesian optimizations. The results are then decoded back into simplified molecular-input line-entry system (SMILES) notation, for recovery of molecular descriptors. Variations to this method involve using generative adversarial networks (GAN)s, as subnetworks in the architecture, to generate the new chemical structures [1](#).

Other approaches for developing new chemical structures involve recurrent neural networks (RNN), to generate new valid SMILES strings, after training the RNNs on copious quantities of known SMILES datasets. The RNNs use probability distributions learned from training sets, to generate new strings that correspond to molecular structures [2](#). Variations to this approach incorporate reinforcement learning to reward models for new chemical structures, while punishing them for undesirable results [3](#).

1.2. Bioactivity prediction

Computational methods have been used in drug development for decades [4](#). The emergence of high-throughput screening (HTS), in which automated equipment is used to conduct large assays of scientific experiments on molecular compounds in parallel, has resulted in generation of enormous amounts of data that require processing. Quantitative structure activity relationship (QSAR) models for predicting the biological activity responses to physiochemical properties of predictor chemicals, extensively use machine learning models like support vector machines (SVM) and random decision forests (RF) for processing [1](#) [5](#).

While deep learning (DL) approaches have an advantage over single-layer machine learning methods, when predicting biological activity responses to properties of predictor chemicals, they have only recently been used for this [1](#). The need to interpret how predictions are made through computationally oriented drug discovery, is seen - in part - as a factor to why DL approaches have not been adopted as quickly in this area [6](#). However, because DL models can learn complex non-linear data patterns, using their multiple hidden layers to capture patterns in data, they are better suited for processing complex life sciences data, than other machine learning approaches [6](#).

Their applications have included profiling tumors at molecular level and predicting drug responses, based on pharmacological and biological molecular structures, functions, and dynamics. This is attributed to their ability to handle high dimensionality in data features, making them appealing for use in predicting drug response [5](#).

For example, deep neural networks were used in models that won NIH's Toxi21 Challenge [7](#) on using chemical structure data only to predict compounds of concern to human health [8](#). DL models were also found to perform better than standard RF models [9](#) in predicting the biological activities of molecular compounds in the Merck Molecular Activity Challenge on Kaggle [10](#). Details of the challenge follow.

2. Related Work

2.1. Merck Molecular Activity Challenge on Kaggle

A challenge to identify the best statistical techniques for predicting molecular activity was issued by Merck & Co Pharmaceutical, through Kaggle in October of 2012. The stated goal of the challenge was to 'help develop safe and effective medicines by predicting molecular activity' for effects that were both on and off target [10](#).

2.2. The Dataset

A [dataset](#) was provided for the challenge [10](#). It consisted of 15 molecular activity datasets. Each dataset contained rows corresponding to assays of biological activity for chemical compounds. The datasets were subdivided into training and test set files. The training and test dataset split

was done by dates of testing ¹⁰, with test set dates consisting of assays conducted after the training set assays.

The training set files each had a column with molecular descriptors that were formulated from chemical molecular structures. A second column in the files contained numeric values, corresponding to raw activity measures. These were not normalized, and indicated measures in different units.

The remainder of the columns in each training dataset file indicated disguised substructures of molecules. Values in each row, under the substructure (atom pair and donor-acceptor pair) codes, corresponded to the frequencies at which each of the substructures appeared in each compound. Figure 1 shows part of the head row for one of the training dataset files, and the first records in the file.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	
1	MOLECULE	Act	D_37	D_38	D_39	D_40	D_42	D_43	D_44	D_45	D_46	D_47	D_49	D_51	D_58	D_59	D_60	D_61	D_62	D_63	D_64	D_65
2	ACT7_M_5058	70	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	ACT7_M_6406	74.68	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	ACT7_M_1263	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	ACT7_M_1759	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	ACT7_M_1762	73	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	ACT7_M_1763	79	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	ACT7_M_1766	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	ACT7_M_1766	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	ACT7_M_2086	24.52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	ACT7_M_2198	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	ACT7_M_2216	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	ACT7_M_2409	4.067	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	ACT7_M_2444	0.3049	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	ACT7_M_2695	96.55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	ACT7_M_3095	0.1672	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	ACT7_M_3182	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	ACT7_M_3583	35.78	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
19	ACT7_M_3820	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	ACT7_M_3823	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	ACT7_M_3823	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	ACT7_M_3832	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	ACT7_M_3840	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	ACT7_M_3891	1	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
25	ACT7_M_3904	15	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	6

Figure 1: Head Row of 1 of 15 Training Dataset files

The test dataset files were similar (Figure 2) to the training files, except they did not include the column for activity measures. The challenge presented was to predict the activity measures for the test dataset.

Figure 2: Head Row of 1 of 15 Test Dataset files

2.3. A Deep Learning Algorithm

The entry that won the Merck Molecular Activity Challenge on Kaggle used an ensemble of methods that included a fully connected neural network as the main contributor to the high accuracy in predicting molecular activity ⁹. Evaluations of predictions for molecular activity for the test set assays were then determined using the mean of the correlation coefficient (R₂) of the 15 data sets. Sample code in R was provided for evaluating the correlation coefficient. The code, and formula for R₂ are appended in Appendix 1.

An approach of employing convolutional networks on substructures of molecules, to concentrate learning on localized features, while reducing the number of parameters in the overall network, was also proposed in literature on improving molecular activity predictions. This methodology of identifying molecular substructures as graph convolutions, prior to further processing, was discussed by authors [11](#), [12](#).

In line with the above research, an ensemble of networks for predicting molecular activity was planned for this project, using the Merck dataset, and hyperparameter configurations found optimal by the cited authors. Recognized optimal activation functions, for different neural network types and prediction types [13](#), were also earmarked for use on the project.

3. Project Implementation

Implementation details for the project were as follows:

3.1. Tools and Environment

The Python programming language (version 3.7.10) was used on Google Colab (<https://colab.research.google.com>).

A subscription account to the service was employed, for access to more RAM (High-RAM runtime shape) during development, although the free standard subscription will suffice for the version of code included in this repository.

Google Colab GPU hardware accelerators were used in the runtime configuration.

Prerequisites for the code included packages from [Cloudmesh](#), for benchmarking performance, and from [Kaggle](#), for API access to related data.

Keras libraries were used for implementing the molecular activity prediction model.

3.2. Implementation Overview

This project's implementation of a molecular activity prediction model consisted of a fully connected neural network. The network used the Adam [14](#) optimization algorithm, at a learning rate of 0.001 and beta_1 calibration of 0.5. Mean Squared Error (MSE) was used for the loss function, and R-Squared [15](#) for the metric. Batch sizes were set at 128. These parameter choices were selected by referencing the choices of other prior investigators [16](#).

The network was trained on the 15 datasets separately, by iterating through the storage location containing preprocessed data, and sampling the data into training, evaluation and prediction datasets - before running the training. The evaluation and prediction steps, for each dataset, were also executed during the iteration of each molecular activity dataset. Running the processing in this way was necessitated by the fact that the 15 datasets each had different feature set columns, corresponding to different molecular substructures. As such, they could not be readily processed through a single dataframe.

An additional compounding factor was that the data was missing the molecular activity results (actual readings) associated with the dataset provided for testing. These were not available through Kaggle as the original competition withheld these from contestants, reserving them as a means for evaluating the accuracy of the models submitted. In the absence of this data, for validating the results of this project, the available training data was split into samples that were then used for the exercise. The training of the fully connected network was allocated 80% of the data, while the testing/evaluation of the model was allocated 10% of the data. The remaining data (10%) was used for evaluating predictions.

3.3. Benchmarks

Benchmarks captured during code execution, using cloudmesh-common [7](#), were as follows:

- The data download process from Kaggle, through the Kaggle data API, took 29 seconds.
- Data preprocessing scripts took 8 minutes and 56 seconds to render the data ready for training and evaluation. Preprocessing of data included iterating through the issued datasets separately, since each file contained different combinations of feature columns (molecular substructures).
- The model training, evaluation and prediction step took 7 minutes and 45 seconds.

3.4. Findings

The square of the correlation coefficient (R^2) values obtained (coefficient of determination) [17](#) during training and evaluation were considerably low (< 0.1). A value of one (1) would indicate a goodness of fit for the model that implies that the model is completely on target with predicting accurate outcomes (molecular activity) from the independent variables (substructures/feature sets). Such a model would thus fully account for the predictions, given a set of substructures as inputs. A value of zero (0) would indicate a total lack of correlation between the input feature values and the predicted outputs. As such, it would imply that there is a lot of unexplained

variance in the outputs of the model. The square of the correlation coefficient values obtained for this model (<0.1) therefore imply that it either did not learn enough, or other unexplained (by the model) variance caused unreliable predictions.

4. Discussion

An overwhelming proportion of the data elements provided through the datasets were zeros (0)s, indicating that no frequencies of the molecular substructures/features were present in the molecules represented by particular rows of data elements. This disproportionate representation of absent molecular substructure frequencies, versus the significantly lower instances where there were frequencies appears to have had an effect of dampening the learning of the fully connected neural network.

This supports approaches that advocated for the use of convolutional neural networks [11](#), [12](#) as auxiliary components to help focus learning on pertinent substructures. While the planning phase of this project had incorporated inclusion of such, the investigator ran out of time to implement an ensemble network that would include the suggestions.

Apart from employing convolutions, other preprocessing approaches for rescaling, and normalizing, the data features and activations [16](#) could have helped the learning, and subsequently the predictions made. This reinforces the fact that deep learning models, as is true of other machine learning approaches, rely deeply on the quality and preparation of data fed into them.

5. Conclusion

Deep learning is a very powerful new approach to solving many machine learning problems, including some that have eluded solutions till now. While deep learning models offer robust and sophisticated ways of learning patterns in data, they are still only half the story. The quality and appropriate preparation of the data fed into models is equally important when seeking to have meaningful results.

6. Acknowledgments

Acknowledgements go to Dr. Geoffrey Fox for his excellent guidance on ways to think about deep learning approaches, and for his instructorship of the course 'ENG-E599: AI-First Engineering', for which this project is a deliverable. Acknowledgements also go to Dr. Gregor von Laszewski for his astute tips and recommendations on technical matters, and on coding and documentation etiquette.

7. Appendix

Square of the Correlation Coefficient (R2) Formula:

Predictions for activity will be evaluated using the correlation coefficient $\langle R^2 \rangle$, averaged over the 15 data sets.

$$R^2 = \frac{1}{15} \sum_{s=1}^{15} r_s^2$$

where

[10](#)

$$r_s^2 = \frac{[\sum_{i=1}^{N_s} (x_i - \bar{x})(y_i - \bar{y})]^2}{\sum_{i=1}^{N_s} (x_i - \bar{x})^2 \sum_{i=1}^{N_s} (y_i - \bar{y})^2}$$

where $\langle x \rangle$ is the known activity, $\langle \bar{x} \rangle$ is the mean of the known activity, $\langle y \rangle$ is the predicted activity, $\langle \bar{y} \rangle$ is the mean of the predicted activity, and $\langle N_s \rangle$ is the number of molecules in data set $\langle s \rangle$.

Sample R2 Code in the R Programming Language:

```

Rsquared <- function(x,y) {
  # Returns R-squared.
  #  $R^2 = \frac{[\sum_i(x_i - \bar{x})(y_i - \bar{y})]^2}{\sum_i(x_i - \bar{x})^2 \sum_j(y_j - \bar{y})^2}$ 
  # Arguments: x = solution activities
  #           y = predicted activities
  if ( length(x) != length(y) ) {
    warning("Input vectors must be same length!")
  }
  else {
    avx <- mean(x)
    avy <- mean(y)
    num <- sum( (x-avx)*(y-avy) )
    num <- num*num
    denom <- sum( (x-avx)*(x-avx) ) * sum( (y-avy)*(y-avy) )
    return(num/denom)
  }
}

```

[10](#)

References

1. Hongming Chen, O. E. (2018). The rise of deep learning in drug discovery. Elsevier. [🔗](#)
2. Marwin H. S. Segler, T. K. (2018). Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. America Chemical Society. [🔗](#)
3. N Jaques, S. G. (2017). Sequence Tutor: Conservative Fine-Tuning of Sequence Generation Models with KL-control. Proceedings of the 34th International Conference on Machine Learning, PMLR (pp. 1645-1654). MLResearchPress. [🔗](#)
4. Gregory Sliwoski, S. K. (2014). Computational Methods in Drug Discovery. Pharmacol Rev, 334 - 395. [🔗](#)
5. Delora Baptista, P. G. (2020). Deep learning for drug response prediction in cancer. Briefings in Bioinformatics, 22, 2021, 360–379. [🔗](#)
6. Erik Gawehn, J. A. (2016). Deep Learning in Drug Discovery. Molecular Informatics, 3 - 14. [🔗](#)
7. National Institute of Health. (2014, November 14). Tox21 Data Challenge 2014. Retrieved from [tripod.nih.gov:] <https://tripod.nih.gov/tox21/challenge/> [🔗](#)
8. Andreas Mayr, G. K. (2016). Deeptox: Toxicity Prediction using Deep Learning. Frontiers in Environmental Science. [🔗](#)
9. Junshui Ma, R. P. (2015). Deep Neural Nets as a Method for Quantitative Structure-Activity Relationships. Journal of Chemical Information and Modeling, 263-274. [🔗](#)
10. Kaggle. (n.d.). Merck Molecular Activity Challenge. Retrieved from [Kaggle.com:] <https://www.kaggle.com/c/MerckActivity> [🔗](#)
11. Kearnes, S., McCloskey, K., Berndl, M., Pande, V., & Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. Switzerland: Springer International Publishing. [🔗](#)
12. Mikael Henaff, J. B. (2015). Deep Convolutional Networks on Graph-Structured Data. [🔗](#)
13. Bronlee, J. (2021, January 22). How to Choose an Activation Function for Deep Learning. Retrieved from [https://machinelearningmastery.com:] <https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/> [🔗](#)
14. Keras. (2021). Adam. Retrieved from [https://keras.io:] <https://keras.io/api/optimizers/adam/> [🔗](#)

15. Keras. (2021). Regression Metrics. Retrieved from [https://keras.io:]
https://keras.io/api/metrics/regression_metrics/ ↗
 16. RuwanT (2017, May 16). Merk. Retrieved from [https://github.com:]
<https://github.com/RuwanT/merck/blob/master/README.md> ↗
 17. Wikipedia (2021). Coefficient of Determination. Retrieved from [https://wikipedia.org:]
https://en.wikipedia.org/wiki/Coefficient_of_determination ↗
-