Project - TER

An Empirical Study on Hyperparameters for Twitter Sentiment Analysis Using Machine Learning and Deep Learning Techniques

Manideep Reddy KANCHERLA
Mamadou Yaya Cherif DIALLO

M1 Computer Vision and Intelligent Machines
Univeristé Paris Cité
Paris, France
May 2022

Contents

List of Figures

Abstract

Social networking services became an integral part of our society's existence. The amount of data that's being generated everyday contains every bit of our lives. In this context, sentiment analysis plays a key role in various different industries in order to grasp people's opinions and emotions on a person or an entity through text or through visual data. This report conducted an empiric study on hyperparameters for twitter sentiment analysis using machine learning algorithms and deep learning. We developed a detailed methodology in order to train models or predict sentiments. Then, we did a hyperparameter tuning on these algorithms in order to improve the accuracy followed by a discussion of these results. On one hand, we tested various algorithms for machine learning but decided on logistic regression, random forest and support vector machines (SVM) using hyperparameters C for the first two and n_estimators for the last to optimize the algorithms. On the other hand, we used Bidirectional long-short term memory (BiLSTM) neural network and hyperparameter T (Temperature) in the Softmax layer to optimize our model. This report discussed in detail all the above mentioned topics and provided a methodology for twitter sentiment analysis along with hyperparameter tuning.

Machine Learning, Sentiment Analysis, Deep Learning, Twitter, Logistic Regression, Random Forest, SVM, US Airlines, BiLSTM, Hyperparameters, C, n_estimators, Temperature, Softmax

Introduction

Social networking services such as Twitter, Facebook and Whats-app have seen tremendous growth during the last decade and have become an integral part of today's social, political and economical landscape. Most importantly, Twitter has become a hub in today's world to discuss these issues. Therefore, understanding people's behaviour, emotions and opinions on these social networking services has become a key industry.

Sentiment analysis is the analysis of people's thoughts, feelings and opinions towards a certain individual, a group or an entity. It is a computational study which aims to determine whether these opinions are positive, neutral or negative. For example, Uniphore, one of the fastest growing automation companies on the planet, is developing an image-based system called Q for Sales which can deduce a potential customer's sentiment towards products mentioned by a salesperson during virtual meetings using advanced computer vision, speech recognition, natural-language processing and emotion AI to pick up on behavioral cues of the client [**1**]. While this raises a lot of potential privacy issues, this has the potential to create an industry with a market size of US$4.3 billion dollars in 2027 from US$1.6 billion dollars in 2020 [**2**]. Therefore, finding ways to optimize the accuracy of detecting sentiments will be a crucial part this future multi-billion dollar industry. In our project, we found ways to optimize the accuracy by using hyperparameters for twitter sentiment analysis using machine learning and neural networks on a US airlines dataset.

We worked with three different machine learning algorithms: Logistic Regression, Support Vector Machines (SVM) and Random Forest Classifier. For SVM and Logistic Regression, we experimented with hyperparameter C, which is the cost parameter used for regularisation. For the Random Forest Classifier, we used the hyperparameter n_estimators, which defines the number of trees before taking the maximum voting or averages of prediction.

Artificial neural networks have become an integral part of machine learning today, mainly in regards to efficiency of doing classification, recommendation tasks. During this project, we will study the search and application of hyperparameters in order to improve multi-class classification tasks. We will be studying the hyperparameter, called temperature, which can be used to effect the output (probability distribution) of softmax activation function where the output is passed through the cost function.

Our core objective is to study the effect of hyperparameters on the accuracies of machine learning algorithms and the neural network. Then, we did a comparative study of the results. At last, we tried to extract meaningful conclusions on the results we obtained.

In Section 1, we started with a comprehensive literature review to understand the subject better. Next, in Section 2, we defined in detail the problems that we aim to solve. In Section 3, we described the different methodologies applied during the study. Lastly, in Section 4, we presented our experimental results along with a detailed discussion of these results.

1 Literature Review

During the course of our project we read various scientific articles to better understand three core ideas: sentiment analysis, hyperparameter optimization for machine learning and artificial neural networks. We found a lot of informative topics which were the key to attaining out objectives.

1.1 Sentiment Analysis

One of the first studies on feelings/opinions appeared 20 years back [**3**]. In their article, B.Pang et al.[**3**], concluded that sentiment analysis based on machine learning techniques was more accurate than human-generated baseline. However, their dataset was very small and the techniques used in their paper were limited to three standard machine learning algorithms. In 2019, Li et al.[**4**] developed an advanced system called TweetSenti to analyze the sentiments of entities in a sentence. They also developed a web application to allow the users to use TweetSenti in real time to analyze the various entities. This comparison shows the evolution of various approaches we can use today compared to 20 years ago for sentiment analysis.

In 2021, Villavicencio et al.[**5**] worked on COVID-19 dataset acquired through twitter in Philippines to understand the Filipino people's sentiment towards their government's handling of COVID-19. They assigned a sentiment to the tweets using RapidMiner data science software. In order to do the classification, they used the standard Naive Bayes algorithm on dataset of 993 tweets post data preparation and preprocessing. They reported that the accuracy they managed to obtain is higher than other sentiment analysis articles for COVID-19 in the same period. Naive Bayes is the standard algorithm used for sentiment analysis using machine learning. However, the authors don't introduce a novel approach to sentiment analysis but rely on a standard Naive Bayes algorithm. Moreover, their dataset was relatively small in size. In 2017, Song et al.[**6**] developed a novel classification approach based on Naive Bayes algorithm that doesn't use the same number of attributes to estimate the weight of each class and excludes uncountable and meaningless attributes. They compared their novel approach to Multinomial Naive Bayes and Multivariate Naive Bayes algorithms, and demonstrated that their approach significantly increases the accuracy. Their novel approach puts emphasis on two methods to enhance the performance of Naive Bayes, feature selection and attribute weighting. They evaluated their classification model on a dataset of 1.6 million tweets (equal amounts of positive and negative tweets) and used further subsets of training data: 10000, 20000, 30000, 40000, 50000 and 70000. They were able to demonstrate their approach's significant advantage over Multinomial Naive Bayes and Multivariate Naive Bayes but also a bit of improvement compared to feature selection and attribute weighting. However, their comparison with results obtained in other scientific articles isn't completely valid because the size and the context of the training and test sets vary a lot.

Villavicencio et al.[**5**] used a dataset with a mix of English and Filipino tweets to classify them into positive, negative and neutral tweets. In 2022, Al-Hashedi et al.[**7**] worked on people's sentiments towards COVID-19 with the dataset containing all the tweets in Arabic. They used machine learning model to analyze the tweets in Arabic. In this model, they used Word2Vec for word embedding, two pretrained continuous bag-of-words(CBOW) and Naive Bayes as a baseline classifier. Once the baseline was established they used other machine learning algorithms to find the optimal accuracy. Even though all the tweets in our dataset were in English, these articles were an important reference in terms of prerprocessing and machine learning techniques. Moreover, we learnt a lot in terms of word embedding technique in these articles.

In the same vein, our project demonstrated the use of machine learning and deep learning algorithms for sentiment analysis along with the optimisation of these algorithms using hyperparameters. We tested our approach on algorithms with distinct structure and core methodology.

1.2 Neural Networks - Hyperparameter Temperature

Our project centered around understanding the hyperparameter temperature used in the softmax activation function in order to optimize the output of a neural network. In Annexe 1, you can find the demonstration of the derivative of the loss function with respect to the softmax input logits zi. This is adopted from G.Hinton et al. [**8**], where the idea is to train a smaller network that, given an input, will produce the same output that the larger network would produce given the same input. In order to transfer the knowledge of a larger network into a small network, we can use Soft Targets, which are the probabilistic distributions of different categories instead of Hard Targets, which is a category vector with one entry being 1 and the rest 0. So, in order to get the soft targets pi, we divide the averaged logits from the larger network by temperature to get a soft distribution. So, by generating the soft target from a larger network, we are trying to transfer knowledge from a larger model into a newer model. This was an ingenious way of reducing the network size by using softmax by manipulating temperature.

X.Zhang et al. [**9**] proposed a "heating-up" strategy to train a classifier with increasing temperatures, leading the corresponding embeddings to state-of-the-art performance on a variety of metric learning benchmarks. This paper presents an interesting idea to improve the softmax embedding performance with heated-up strategy. Several experiments on metric learning datasets demonstrate the effectiveness of the proposed method with a relatively simple implementation. The motivation was to find a balance between the compactness and "spread-out" embedding. The major weakness was the intermediate temperature selection.

A.Agarwal et al. [**10**] worked on the impact of temperature on learning dynamics or generalization performance. The authors developed a theory of early learning for models trained with softmax + cross entropy loss. They showed theoretically that the learning dynamics depend mainly on the inverse of the temperature as well as the magnitude of the logits at initialization. They conducted a large scale empirical study of a variety of model architectures trained on Cifar10, ImageNet, and IMDB sentiment analysis. The authors were able to find that generalization performance depends strongly on the temperature (as a tunable hyperparameter), but only weakly on the initial logit performance.

Along with these interesting articles, we studied a lot more articles in order to better understand the softmax function along with it's lone hyperparameter temperature. This allowed us to construct a BiLSTM deep learning model for text classification and sentiment analysis optimized by the hyperparameter temperature in the softmax layer. The next step was to better understand the machine learning hyperparameters that served as a comparison.

1.3 Machine Learning Algorithms - Hyperparameters

During our project, we used grid search method to find the hyperparameters for the machine learning algorithms. On one hand, B.H.Shekar et al. [**11**] used grid search based hyperparameter tuning for classification of Microarray Cancer Data. There were many challenges that exist in analyzing microarray cancer data such as the curse of dimensionality, small sample size, noisy data, and imbalance class problem. The authors used grid search-based hyperparameter tuning (GSHPT) for random forest classifier. In order to find the optimal accuracy, they used 10- fold cross validation. This helped them provide the best parameters for the random forest classifier. The empirical study showed a clear improvement over the other state of the art methods and was evaluated on metrics such as classification accuracy, precision, recall, f1-score, confusion matrix and classification. On the other hand, J.Wu et al. [**14**] used Gaussain processes to build the relationship between performance of the machine learning models and their hyperparameters. The authors used Bayesian Optimisation which was based on the Bayesian Theorem. They tested the hyperparameters on random forest algorithm and neural networks, and demonstrated the advantage of hyperparameters. These articles worked with a single hyperparameter tuning technique. Next, we read more articles working with multiple hyperparameter optimization techniques.

A.Arafa et al. [**12**] evaluated the performance of grid search, random search, Bayesian Tree Parzen Estimator (TPE) and Simulated Annealing (SA) optimization to determine the best hyperparameters for a logistic regression model in cancer classification. The Bayesian optimization outperformed others but grid search didn't lose out by much. However, the authors succeeded in demonstrating the positive effect of hyperparameters on classification tasks. E.Elgeldawi et al. [**13**] explored even more novel methods of hyperparameter tuning. The authors used grid search, random search, Bayesian optimization, Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) for hyperparameter tuning on various different machine learning algorithms: Logistic Regression, Ridge Classifier, Support Vector Machine Classifier, Decision Tree, Random Forest, and Naive Bayes Classifier to test the performance of each hyperparameter tuning technique. The problem was complex due to the language used being Arabic tested for sentiment classification. They demonstrated the clear advantage of machine learning algorithms after hyperparameter optimization by comparing before and after the tuning.

The above articles allowed us to cover enough ground to understand the advantages of hyperparameter tuning with multiple machine learning algorithms. This along with sentiment analysis and deep learning helped us enormously in developing our methodology to tackle and realise the objectives we set forth.

2 Methodology

The methodology was the main component in our article to better compare the different algorithms along with proving the advantages of hyperparameters on the accuracies. We developed a methodology for machine learning algorithms and neural networks along with text preprocessing and data cleaning. This helped us get interesting results to have a constructive discussion about these results.

Here's a list of the main materials used during our project:

- **Laptop**: AMD Ryzen 7 4800H with Radeon Integrated Graphics / 16.0 GB RAM with a 64-bit operating system, x64-based processor.
- **Google Colab**: Colab is an online Jupyter Notebooks environment form Google. It is in the cloud so no software installation is required. It is a very convenient and performance enhancing tool for machine learning and deep learning techniques.
- **Dataset**: Our dataset recorded the tweets discussing people's opinions of US airline companies and contained three kinds of target labels: 1 for positive, 0 for neutrality, and -1 for negative. The dataset contained around 14700 tweets.

Our methodology consisted mainly of detailed text preprocessing and data cleaning using natural language tool kit. We employed two different preprocessing and cleaning approaches for each of machine learning and deep learning techniques.

2.1 Machine Learning Approach

2.1.1 Installing and Importing Dependencies

Here were some of the important dependencies we used during our project:

- **pandas**: Pandas is an open-source library that is made for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series.
- **matplotlib**: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is the most famous library for data visualisation. Many other visualisation libraries are built on top it like seaborn.
- **nltk (Natural Language Toolkit)**: Natural Language Tool Kit is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.
- **sqlite3)**: SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of SQL query language. The sqlite3 module helps access this database.
- **sklearn)**: Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

2.1.2 Data Acquisition and Data Storage

The dataset was stored in a database using sqlite. Once the database was uploaded on drive, we mounted the drive to google colab and imported the dataset directly from the sqlite database containing the datasets.

2.1.3 Text Preprocessing and Cleaning

This step was one of the most important steps since it had a significant impact on the results. All the functions used for tweet text preprocessing belonged to nltk.

Here's a brief introduction to some of them:

- **Stopwords**: These are words that can be safely ignored as they don't change or add meaning to a sentence. Ex: "in", "a", "the".
- **Cleaning Punctuations**: This function was used to clean all the punctuation marks. Ex: "!", "?".
- **Cleaning URLs**: With the help of this function, we were able to remove links starting with "http" or "www".
- **Stemming)**: Stemming is used to reduce a word to it's root form and to remove the suffix form this word.
- **Lemmatizing)**: It is similar to stemming but it adds context to the words. It links words with similar meaning to one word.

The next step was to split the data into X_train and X_test for machine learning.

Then, we transformed the collection of raw tweet text into vectors on the basis of the frequency of each word that exists in the collection of tweet text using CountVectorizer. This is to help the machine learning algorithms understand the text as a feature.

2.1.4 Text Analysis - Machine Learning Approach

This was the most crucial part of the methodology where we used machine learning algorithms to classify the tweets containing customer sentiments regarding the US airlines into positive, negative and neutral sentiments.

Machine Learning helps us predict the outcome without being programmed to do so by using historical data as input. We used a total of 3 different algorithms.

Here are the 3 Machine Learning algorithms:

- **Support Vector Machines (SVM)**: Support Vector Machine (SVM) is a supervised Machine Learning model used for multiclass classification or regression tasks. Using a data-set, the model learns the parameters needed to generate a hyperplane, w.x - b = 0, that separates the space into different parts in accordance to the number of classes. As you can see in Fig.1, the hyperplane separated the space into two classes.

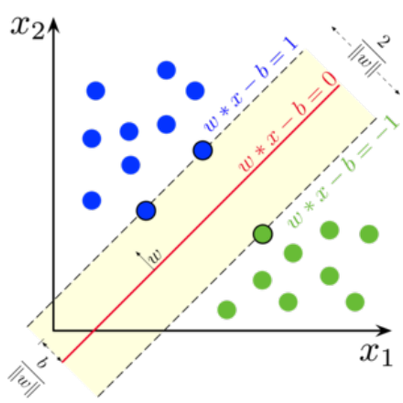

Figure 1: Hyperplane separation using SVM.

- **Logistic Regression**: Logistic Regression is a supervised machine learning algorithm based on probability. It is a predictive analysis algorithm used for classification tasks. In logistic regression, the dependant variable is a binary variable (0 or 1).
- **Random Forest**: A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. In Fig.2, we can observe the structure of a random forest classifier.

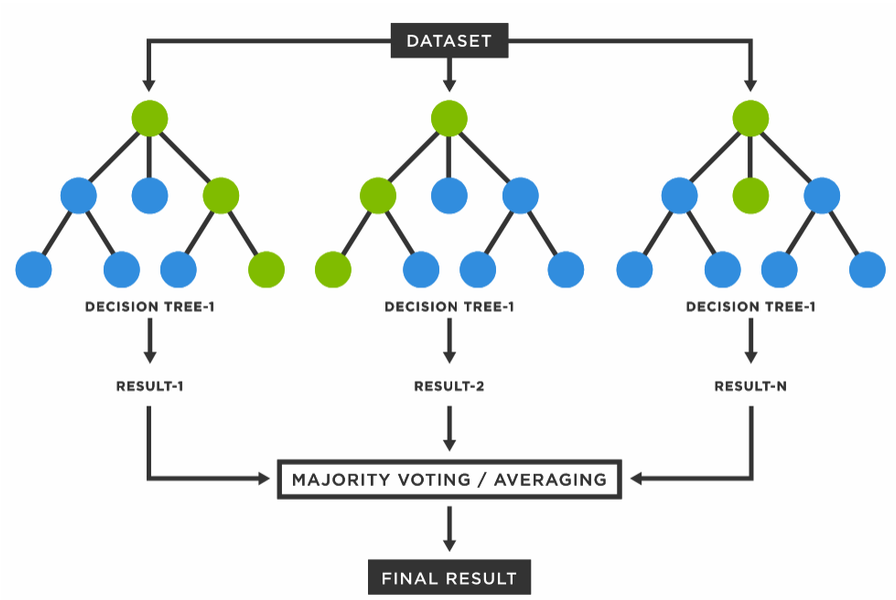

Figure 2: Structure of Random Forest Classifier.

2.1.5 Hyperparameter Tuning

The goal of hyperparameter tuning was to find the most optimal model architecture. We used GridSearch for hyperparameter tuning, is the most basic hyperparameter tuning method. Using grid search, we built a model for each possible combination of all of the hyperparameter values provided, evaluating each model, and selecting the architecture which produces the best results. As a parameter of the classifier during grid search, we employed K-fold cross validation, which is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. During our project, we consistently used a 5-fold cross validation as a parameter in our algorithms.

Here were the parameters used for each of the machine learning algorithms:

- **Hyperparameter C for Logistic Regression**: C is the inverse of regularization strength and it must be a positive float. The smaller values of C specify stronger regularization.
- **Hyperparameter C for Support Vector Machines**: The effect of C parameter is to add a penalty for each misclassified data point. Therefore, if c is large, SVM tries to minimize the number of missclassified points due to the high penalty which results in a decision boundary with a smaller margin. In the same vein, if the c is small, the penalty is low so a decision boundary with a large margin is chosen at the expense of a greater number of missclassifications. Basically, it is used for regularization. As you can see in Fig.3, when C is high it will classify all the data points correctly but it might tend to overfit.
- **Hyperparameter n_estimators for Random Forest Classifier**: This is a very straightforward hyperparameter as it concerns the number of trees used in a random forest classifier.

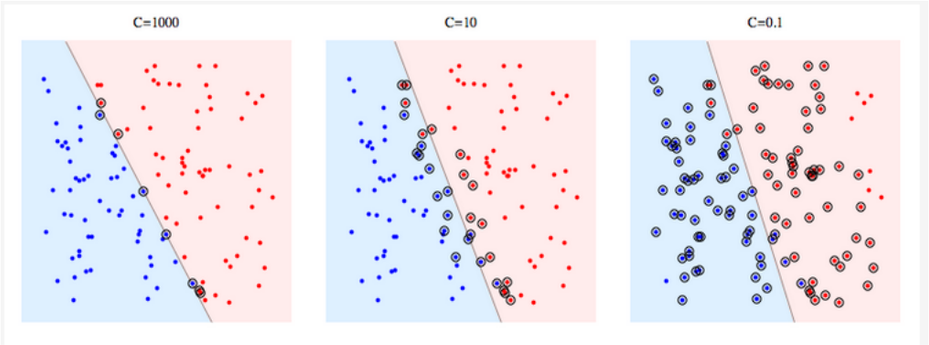

Figure 3: Effect of Hyperparameter C for Support Vector Machines.

2.2 Deep Learning Approach

We used almost the same methodology for the neural network except that the classifiers were different along with changes to preprocessing.

2.2.1 Installing and Importing Dependencies

Along with all the dependencies used for machine learning, here's the key dependency used for deep learning:

- **PyTorch**: Pytorch is a Python package that provides tensor computation with strong GPU acceleration and deep neural networks built on a tape-based autograd system. It is a deep learning research paltform that provides maximum flexibility and speed.

2.2.2 Data Acquisition and Data Storage

The dataset was stored in a database using sqlite. Once the database was uploaded on drive, we mounted the drive to google colab and imported the dataset directly from the sqlite database containing the dataset.

2.2.3 Text Preprocessing and Data Cleaning

Here were the preprocessing steps used before building the model:

- **Null Value Removal**: We dropped the null values in the dataset.
- **Removal of links and mentions**: We removed all the links and mentions in the tweets that contribute to the algorithm's performance.
- **Tokenization**: We split the text data into tokens to better classify them.
- **Encoding all the tweets**: We used index2word (it contains three tokens: padding (PAD), start of sentence (SOS) and end of sentence (EOS)) and word2index to act as our vocabulary to encode the tweets.

Once all the preprocessing was done, we created the pytorch datasets and data loaders to train our model and predict sentiments.

2.2.4 BiLSTM Model Building

We used a BiLSTM model which was relatively simple to build, where we first built the layers in the initialization, and then created a forward method to compute output tensors from input tensors.

Here were the some of the important things to note:

- **Initialization**: We started with the embedding layer where words were converted into word embeddings, which were vectors that represented each word. Then, the embedding size and the hidden vector size were fed to the LSTM. After the dropout was applied, the fully-connected layer takes in the hidden dimension of the LSTM to output a 3 x 1 vector of the class scores.
- **Forward**: The input of the forward function was transformed to embeddings. Then, it was passed through LSTM followed by the dropout and the fully-connected layer. At last, it passed through the Softmax layer.

We trained our model for 50 epochs before extracting the testing accuracy.

2.2.5 Hyperparameter Tuning - Temperature

Here's an introduction of softmax layer and then temperature along with the demonstration:

Softmax is an output layer that converts the logits, zi, computed for each class into a probability, qi.

Here's the equation of the softmax function:

$$\text{softmax}(x)_i = \frac{e^{\frac{y_i}{T}}}{\sum_j^N e^{\frac{y_j}{T}}}$$

Figure 4: Softmax Function.

Here's how to find the probability qi from the logits zi:

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Figure 5: Softmax Application.

T represents the hyperparameter, which is at the core of our study. The goal is to find the optimal value of temperature which lets us get the best possible accuracy. It's important to note that in our study we will work with alpha, which is the inverse of T, to simplify the notation.

The next step is to define the Cross-Entropy loss function between qi, the probabilities generated from softmax, and pi, the soft-targets generated after softmax, and find the gradient of Cross-Entropy with temperature. In the end, the derivative of the loss function with respect to the softmax input logits zi is (Please refer to Annex 1 for the full demonstration). In our study, we used different values of temperature to better understand the effect of hyperparameters on the accuracy of a model.

3 Results and Discussion

We used different values of the respective hyperparameters for each of the algorithms to make meanignful conclusions.

3.1 Machine Learning Approach

Here were the initial training and test accuracy results before the hyperparameter tuning:

```
Train Accuracy using Logistic Regression:  0.9236680327868853
Test Accuracy using Logistic Regression:  0.8155737704918032

Train Accuracy using SVC:  0.9237534153005464
Accuracy using SVC:  0.7909836065573771

Train Accuracy using RandomForest:  0.9972677595628415
Accuracy using RandomForest:  0.7701502732240437
```

Figure 6: Accuracy values of machine learning algorithms before hyperparameter tuning.

We could observe that the accuracy very encouraging. On one hand, the training accuracies were all above 90%, which proved that our methodology worked very well. The random forest classifier had the highest accuracy along with the highest time to train. On the other hand, the testing accuracies were also very above 75% for all the algorithms with Logistic Regression being the best choice. The results showed that the models worked very well by being able to predict the correct sentiment almost every time.

Here are the accuracy results after hyperparameter tuning:

	param_C	param_kernel	mean_test_score	mean_train_score
0	1	rbf	0.622353	0.622353
1	1	linear	0.761015	0.968579
2	5	rbf	0.622353	0.622353
3	5	linear	0.732497	0.987769
4	10	rbf	0.622609	0.622695
5	10	linear	0.723788	0.991504
6	15	rbf	0.624317	0.624488
7	15	linear	0.716018	0.993041
8	20	rbf	0.631062	0.632642
9	20	linear	0.715506	0.993490
10	25	rbf	0.649334	0.652216
11	25	linear	0.713371	0.994023
12	30	rbf	0.664276	0.669783
13	30	linear	0.714054	0.994386
14	35	rbf	0.678364	0.683444
15	35	linear	0.712774	0.994642
16	40	rbf	0.680241	0.685067
17	40	linear	0.712176	0.994813
18	45	rbf	0.679729	0.683231
19	45	linear	0.711407	0.994941
20	50	rbf	0.680326	0.682847
21	50	linear	0.711066	0.995069
22	75	rbf	0.684766	0.690680
23	75	linear	0.712175	0.995389
24	100	rbf	0.700903	0.713157
25	100	linear	0.710553	0.995539
26	150	rbf	0.728140	0.757023
27	150	linear	0.708334	0.995774
28	200	rbf	0.744449	0.784815
29	200	linear	0.705687	0.995966
30	500	rbf	0.772028	0.853356
31	500	linear	0.702271	0.996265
32	1000	rbf	0.774675	0.893741
33	1000	linear	0.699795	0.996393

Figure 7: Accuracy values of SVM after the hyperparameter tuning.

In the case of SVM (Fig. 7), we could already eliminate the Radial basis function (RBF) kernel due to it's low accuracies mainly due to the issues of adaptability. The linear classifier was the best option with the most optimal accuracy for the highest value of C, here for C = 1000 we obtained an accuracy of 0.9964. However, the testing accuracy values weren't better than before the tuning. This meant that our model overfit the data.

In the case of Logistic Regression (Fig.8), the most optimal accuracy was obtained for the highest value of C, here for C = 1000 we obtained an accuracy of 0.9954. However, the testing accuracy values weren't better than before the tuning. This meant that our model overfit the data.

In the case of Random Forest (Fig.9), the most optimal accuracy was obtained for the highest value of n_estimators (which represented the number of trees used for the random

	param_C	mean_test_score	mean_train_score
0	1	0.782787	0.930819
1	5	0.770662	0.974129
2	10	0.764003	0.982091
3	15	0.758026	0.984823
4	20	0.756489	0.986574
5	25	0.755038	0.987641
6	30	0.751537	0.988537
7	35	0.750683	0.989114
8	40	0.749658	0.989626
9	45	0.747695	0.990160
10	50	0.746756	0.990523
11	75	0.742657	0.991483
12	100	0.738985	0.992486
13	150	0.735570	0.993191
14	200	0.732069	0.993724
15	500	0.721397	0.994941
16	1000	0.714395	0.995453

Figure 8: Accuracy values of Logistic Regression after the hyperparameter tuning.

	param_n_estimators	mean_test_score	mean_train_score
0	1	0.621074	0.856429
1	5	0.710723	0.962453
2	10	0.732496	0.982923
3	15	0.737192	0.991590
4	20	0.746328	0.994194
5	25	0.745302	0.995368
6	30	0.747778	0.995987
7	35	0.750171	0.996884
8	40	0.752732	0.997097
9	45	0.749658	0.997182
10	50	0.748803	0.997289
11	75	0.754012	0.997353
12	100	0.754866	0.997439
13	150	0.755805	0.997439
14	200	0.757171	0.997439
15	500	0.756830	0.997439
16	1000	0.755549	0.997439

Figure 9: Accuracy values of the Random Forest Classifier after the hyperparameter tuning.

forest algorithm), here for n_estimators = 1000 we obtained an accuracy of 0.9974. However, the testing accuracy values weren't better than before the tuning. This meant that our model overfit the data.

The training accuracy of every machine learning algorithm increased with the value of hyperparameters. However, the main issue was the overfitting of the data which resulted in lower testing accuracy even though the training accuracies increased.

3.2 Deep Learning Approach

Here were the results of deep learning using temperature hyperparameter:

T (Temperature)	1	5	10	15	20	25	30	35	40	45	50	75	100	150	200	500	1000
Accuracy	0.616	0.63225	0.619	0.587	0.604	0.612	0.601	0.615	0.614	0.6198	0.6123	0.6225	0.6233	0.6239	0.611	0.631	0.6246

Figure 10: Accuracy values of the BiLSTM neural network.

As you can see in Fig.10, the initial value of accuracy with T = 1 was around 62%. This

was less than the machine learning algorithms. However, we implemented a relatively simpler neural network model. This accuracy was optimized using different values of temperature. The maximum accuracy was obtained for a T = 5, which was about 63%.

By comparing the results of machine learning and deep learning after hyperparameter tuning, we could say that the accuracy of detecting a customer's sentiment towards the airline did increase positively. However, there was a risk of overfitting the data and obtaining slightly worse testing accuracies.

Conclusion

Our initial plan was to work on large image datasets but we had problems with the processing of this data and difficulties with the code. However, we were able to answer the question of twitter sentiment analysis along with working multiple different hyperparameters instead of limiting ourselves to temperature.

During this project we were able to do an extensive literature review to familiarize ourselves with the subject of sentiment analysis and hyperparameter tuning of machine learning algorithms and deep learning. Followed by a detailed methodology on how to use two different sentiment analysis approaches to train a model or predict sentiments and enhance these models using hyperparameters. In the end, we were able to obtain interesting results that helped us better understand the challenges and the progress still needed to be done in the field of sentiment analysis. We discovered that the accuracy does increase because of hyperparameter tuning. However, the testing accuracy tends to overfit.

Our initial plan was to use image datasets along with multiple different feature extractors to test the hyperparameter and different corelations in order to find a method to estimate the value of temperature beforehand. In the future, we would like to implement this if the opportunity arises.

References

[1] "Companies are using AI to monitor your mood during sales calls. Zoom might be next." by Kate Kaye, URL: https://www.protocol.com/enterprise/emotion-ai-sales-virtual-zoom.

[2] "Global Sentiment Analysis Software Industry", report by Reportlinker.com

[3] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), pages 79–86. Association for Computational Linguistics.

[4] Quanzhi Li, Qiong Zhang, and Luo Si. 2019. TweetSenti: Target-dependent Tweet Sentiment Analysis. In The World Wide Web Conference (WWW '19). Association for Computing Machinery, New York, NY, USA, 3569–3573. https://doi.org/10.1145/3308558.3314141

[5] Villavicencio, Charlyn, Julio J. Macrohon, X. A. Inbaraj, Jyh-Horng Jeng, and Jer-Guang Hsieh. 2021. "Twitter Sentiment Analysis towards COVID-19 Vaccines in the Philippines Using Naïve Bayes" Information 12, no. 5: 204. https://doi.org/10.3390/info12050204.

[6] Junseok Song, Kyung Tae Kim, Byungjun Lee, Sangyoung Kim, and Hee Yong Youn. 2017. A novel classification approach based on Naïve Bayes for Twitter sentiment analysis. KSII Transactions on Internet and Information Systems, 11, 6, (2017), 2996-3011. DOI: 10.3837/tiis.2017.06.011.

[7] Abdullah Al-Hashedi, Belal Al-Fuhaidi, Abdulqader M. Mohsen, Yousef Ali, Hasan Ali Gamal Al-Kaf, Wedad Al-Sorori, Naseebah Maqtary, "Ensemble Classifiers for Arabic Sentiment Analysis of Social Network (Twitter Data) towards COVID-19-Related Conspiracy Theories", Applied Computational Intelligence and Soft Computing, vol. 2022, Article ID 6614730, 10 pages, 2022. https://doi.org/10.1155/2022/6614730

[8] Hinton, Geoffrey and Vinyals, Oriol and Dean, Jeff, Distilling the Knowledge in a Neural Network, arXiv, year = 2015.

[9] Zhang, Xu and Yu, Felix Xinnan and Karaman, Svebor and Zhang, Wei and Chang, Shih-Fu, Heated-Up Softmax Embedding, arXiv, year = 2018.

[10] Agarwala, Atish and Pennington, Jeffrey and Dauphin, Yann and Schoenholz, Sam, Temperature check: theory and practice for training models with softmax-cross-entropy losses, arXiv, year = 2020.

[11] B. H. Shekar and G. Dagnew, "Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data," 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), 2019, pp. 1-8, doi: 10.1109/ICACCP.2019.8882943.

[12] Ahmed Arafa, A., Radad, M., Badawy, M., El-Fishawy, N. (2022). Logistic Regression Hyperparameter Optimization for Cancer Classification. Menoufia Journal of Electronic Engineering Research, 31(1), 1-8. doi: 10.21608/mjeer.2021.70512.1034.

[13] Elgeldawi E, Sayed A, Galal AR, Zaki AM. Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis. Informatics. 2021; 8(4):79. https://doi.org/10.3390/informatics8040079.

[14] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, Si-Hao Deng, Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization, Journal of Electronic Science and Technology, Volume 17, Issue 1, 2019, Pages 26-40, ISSN 1674-862X, https://doi.org/10.11989/JEST.1674-862X.80904120.

Annex 1

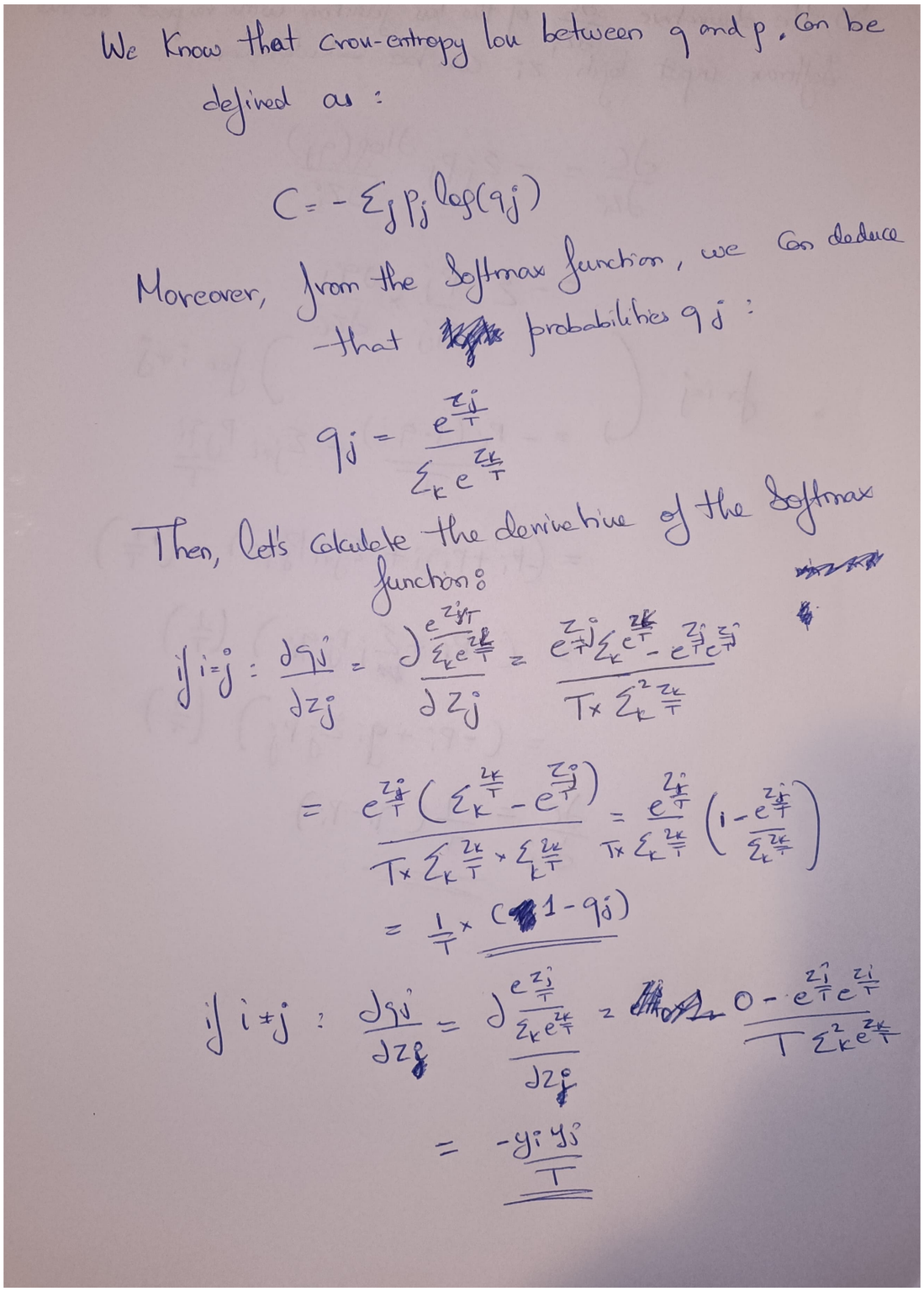

Figure 11: Gradient of Cross-Entropy with Temperature - Part 1.

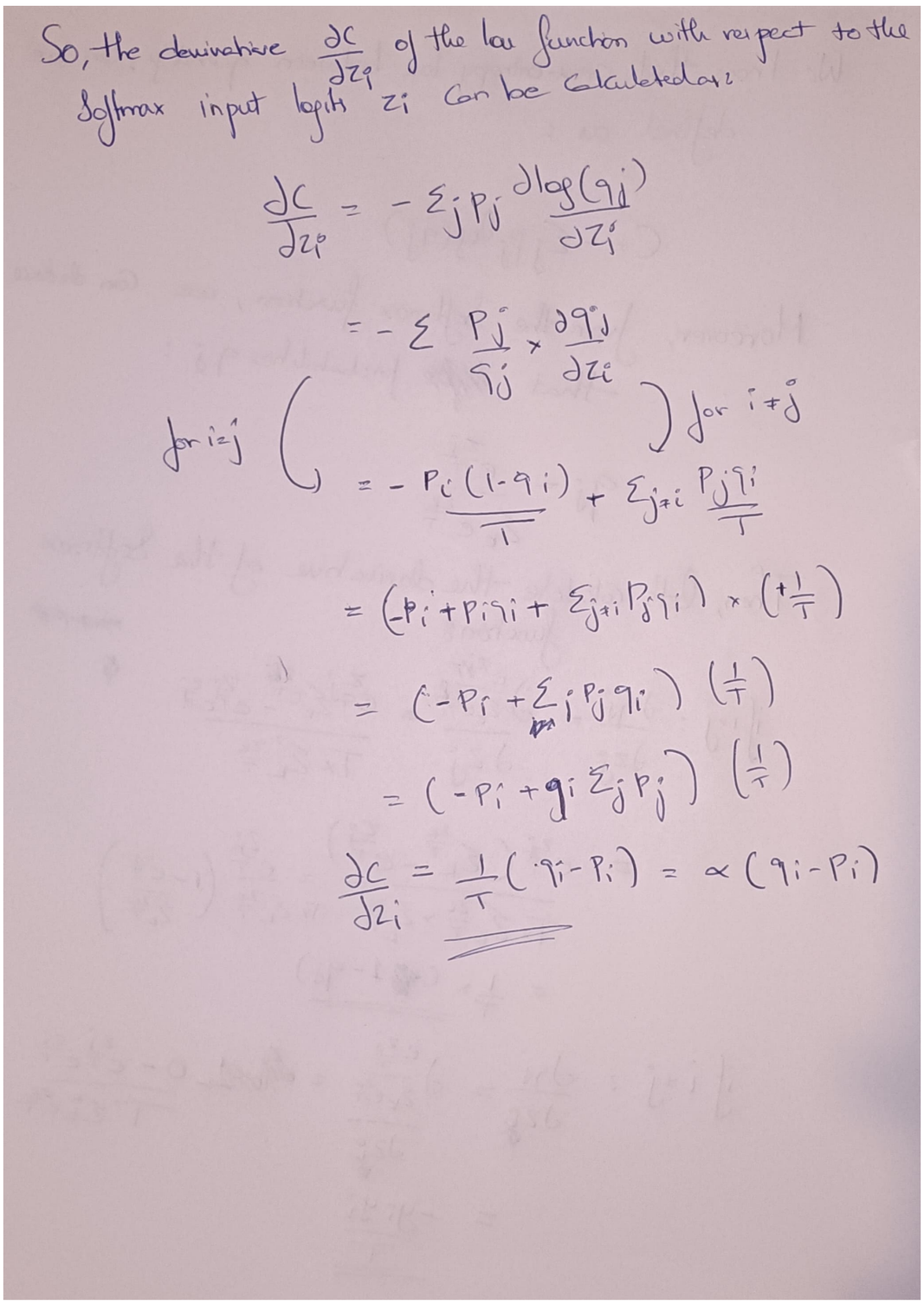

Figure 12: Gradient of Cross-Entropy with Temperature - Part 2.