Exploring the Impact of Key Factors on the Accuracy of a Keras Machine Learning Model for Text Classification

Xiwen Jiang[a]
Suzhou Foreign Language School, Suzhou, China

Keywords: Text Classification, Machine Learning, Keras.

Abstract: Text classification and emotion classification are crucial components of modern machine learning applications, particularly on social media platforms. These classifications enable efficient information organization, sentiment analysis, and user targeting, essential for advertisements and content creation. Leveraging Tensorflow Keras models, this research explores how varying key parameters—epochs and batch size—affect model accuracy when applied to a large dataset sourced from Reddit, comprising over 3.8 million posts. Utilizing TextBlob, labels were generated for the dataset, transforming an unsupervised problem into a supervised one. The model architecture consists of four layers: embedding, GlobalAveragePooling1D, dense with ReLU activation, and dense with sigmoid activation, targeting binary text classification. The study investigates three different epoch values (10, 20, and 30) and batch sizes (32, 64, and 128), running each experiment multiple times with different random seeds to ensure robustness. Findings indicate that increasing the number of epochs generally improves accuracy, although diminishing returns and potential overfitting occur with excessive epochs. Meanwhile, batch size plays a more complex role, as larger batches can hinder the model's ability to capture detailed patterns. The results highlight the trade-offs between computational efficiency and prediction performance, providing practical insights for optimizing machine learning models in text classification tasks.

1 INTRODUCTION

Both text classification and emotional classification are important branches of machine learning (Kowsari, 2019; Mirończuk, 2018; Gasparetto, 2022; Minaee, 2021). Both of them are very useful in modern technology, especially for those social media platforms. While text classification can help organize the information the platform had gathered and make the maintenance and search of the data much easier, emotion classification can help the platform decide what the current trend and have insights of public events. Also, both of them can help the platform to target the users accurately by getting to know their thoughts and opinions. This can be used for advertisement recommendations, content creation, etc.

To conduct these two jobs, the companies usually build a Tensorflow keras model to do the classification job and make the labels with natural language processing models so that this process can be faster and automated. Apart from these two branches, from texts to images, from voice recordings to real-world objects recognition, machine learning has truly helped out in so many industries. Companies use it to improve their sales, governments use it to make better decisions, and its potential is still massive. A large number of algorithms that have been used widely, such as Decision Tree (Song, 2015; Suthaharan, 2016), Bayesian Network (Chen, 2012; Stephenson, 2000), Markov Chain (Norris, 1998; Chung, 1967), have been applied in these fields. These algorithms play pivotal roles in predictive analytics, helping organizations optimize their operations by making informed predictions based on data trends. As a result, the potential of machine learning extends far beyond current applications; the landscape is ever-expanding, with new methods and techniques being developed constantly, and these three are just a few examples.

This article delves into a specific dataset drawn from Reddit, which contains a staggering 3,848,330

[a] https://orcid.org/0009-0002-0582-8002

Jiang, X.
Exploring the Impact of Key Factors on the Accuracy of a Keras Machine Learning Model for Text Classification.
DOI: 10.5220/0013331400004558

In *Proceedings of the 1st International Conference on Modern Logistics and Supply Chain Management (MLSCM 2024)*, pages 368-371
ISBN: 978-989-758-738-2

posts, averaging around 270 words each for their content and 28 words for summaries. The features of this dataset include various strings such as author, body, normalizedBody, content, summary, subreddit, and subreddit_id. The content is processed as the main text document, while the summary serves its summarization purpose. However, a notable challenge with this dataset is its lack of predefined labels, which are crucial for supervised learning tasks.

To address this limitation, this study turned to TextBlob, a powerful library in Python that simplifies the process of natural language processing. TextBlob enables us to automatically generate labels for the immense dataset, making it feasible to work with a scale that would be impractical through manual labeling. In fact, it seems almost impossible to make the labels manually as 3,848,330 labels are needed. Once the labels are produced, they are consolidated into a .csv file. After that, they can be easily read with Pandas.

The primary focus of this article is to explore how variations in different settings regarding epochs and batch sizes influence the overall accuracy of machine learning models. By conducting experiments with these parameters, this article tries to provide insight into the critical factors that affect model accuracy in predicting outcomes based on the labeled Reddit dataset.

2 METHOD

This project wants to focus on the influence of the number of epochs as well as the batch size. By setting these parameters at different numbers, the effects of these two variables can be analyzed.

In detail, the number of epochs is set to 10, 20 and 30 respectively, and the batch size is set to 32, 64 and 128 respectively. All the experiments will be repeated 3 times, using different seeds for the random generator of the train and test dataset.

2.1 Dataset Preparation

This project uses the dataset “Reddit” from tensorflow package. This dataset contains 3,848,330 posts from Reddit, averaging around 270 words each for their content and 28 words for summaries. The features of this dataset include various strings such as author, body, normalizedBody, content, summary, subreddit, and subreddit_id. The content is processed as the main text document, while the summary serves its summarization purpose. Originally, this dataset doesn’t contain labels, so the labels used in this project are generated with TextBlob, which is a Python library for processing textual data, providing a simple API for diving into common natural language processing (NLP) tasks. Random 20% of this dataset is used as the test set, while the remaining 80% of the dataset is used as the train set. By using different seeds for the random generator, it can be ensured that the split is reproducible, and that the generator won’t affect the results of the model. In exact numbers, the seeds that were used by this project are 42,58 and 84 respectively.

2.2 Deep Learning-based Classification

This project uses a 4-layer Tensorflow keras sequential model (Pang, 2020). The 4 layers are the embedding layer, the GlobalAveragePooling1D layer, the dense layer with ReLU activation, and the dense layer with Sigmoid activation.

The embedding layer takes integers as input and maps them to dense vector representations. The parameter “input_dim” is set as 5000, so that the input data will consist of integers between 0 and 4999. The “output_dim” is set at 128, which means each word is represented as a 128-dimensional vector. This layer enables the model to learn a vector with a fixed length from each word, capturing semantic meaning.

The GlobalAveragePooling1D layer reduces the number of dimensions in the output from the embedding layer by averaging all sequences for each feature, resulting in a single vector per input sample, hence reducing the complexity of the model.

The dense layer with ReLU activation is a fully connected layer that takes the output from the pooling layer and applies transformations to learn complex patterns. The number of neurons in this layer is set to 10, and each neuron learns different features of the input data. The Rectified Linear Unit activation function outputs the input directly if it is positive and outputs 0 if it is not. This helps introduce non-linearity and allows the model to learn better. By doing so, this layer empowers the model to be more expressive as the model is able to learn higher-level features after the pooling layer.

The dense layer with Sigmoid activation acts as the final layer, generating the model’s output. In this layer, the number of neurons is set to 1, as it is just for the output. The sigmoid function is used here, mapping the output to a value between 0 and 1, representing the probability of the positive class in the binary text classification. This layer takes the learned features from the previous layer, and outputs the final prediction.

Table 1: The performance of the model based on various hyperparameters-1.

Seed	42	42	42	42	42	42	42	42	42
No. epochs	10	10	10	20	20	20	30	30	30
Batch size	32	64	128	32	64	128	32	64	128
Accuracy	0.8663	0.8666	0.8694	0.8708	0.8712	0.8708	0.8709	0.8701	0.8723
Loss	0.3091	0.3081	0.3026	0.3009	0.2999	0.3002	0.3006	0.3025	0.2985
Val_accuracy	0.8617	0.8621	0.8608	0.8602	0.8615	0.8610	0.8602	0.8612	0.8596
Val_loss	0.3197	0.3165	0.3197	0.3219	0.3203	0.3203	0.3232	0.3192	0.3225
Test accuracy	0.8617	0.8621	0.8608	0.8602	0.8615	0.8610	0.8602	0.8606	0.8596

Table 2: The performance of the model based on various hyperparameters-2.

Seed	58	58	58	58	58	58	58	58	58
No. epochs	10	10	10	20	20	20	30	30	30
Batch size	32	64	128	32	64	128	32	64	128
Accuracy	0.8721	0.8727	0.8735	0.8716	0.8725	0.8735	0.8715	0.8706	0.8704
Loss	0.2990	0.2975	0.2958	0.2999	0.2977	0.2958	0.3003	0.3009	0.3010
Val_accuracy	0.8580	0.8604	0.8594	0.8607	0.8602	0.8599	0.8615	0.8613	0.8604
Val_loss	0.3256	0.3225	0.3235	0.3216	0.3221	0.3225	0.3214	0.3196	0.3215
Test accuracy	0.8604	0.8604	0.8594	0.8607	0.8602	0.8599	0.8605	0.8613	0.8604

Table 3: The performance of the model based on various hyperparameters-3.

Seed	84	84	84	84	84	84	84	84	84
No. epochs	10	10	10	20	20	20	30	30	30
Batch size	32	64	128	32	64	128	32	64	128
Accuracy	0.8667	0.8697	0.8715	0.8712	0.8716	0.8724	0.8713	0.8727	0.8733
Loss	0.3079	0.3021	0.2989	0.3002	0.2986	0.2966	0.2995	0.2964	0.2954
Val_accuracy	0.8626	0.8631	0.8625	0.8604	0.8610	0.8620	0.8616	0.8604	0.8607
Val_loss	0.3156	0.3166	0.3171	0.3213	0.3219	0.3196	0.3209	0.3225	0.3222
Test accuracy	0.8626	0.8631	0.8625	0.8604	0.8610	0.8620	0.8616	0.8604	0.8607

2.3 Implementation Details

By default, the generator seed is set at 42, the number of epochs is set to 10, and the batch size is set to 32.

3 RESULTS AND DISCUSSION

By setting different parameters into different values, the following results shown in Table 1, Table 2 and Table 3 can be got.

3.1 The Effects of Number of Epochs

As the number of epochs in a training process increases, a rise in both accuracy and validation accuracy is observed. This trend suggests that with more training iterations, the model can better recognize patterns in the training data, hence achieving better performance. However, a few counterexamples exist in the results as well. This is potentially due to overfitting or other model complexities that can arise with excessive epochs.

In contrast to the accuracy and validation accuracy that are relatively stable improving, the loss and validation loss show fluctuations in the results. This shows that the model is relatively complex, and the fluctuation can indicate that the model is finding local minima and is difficult to converge on a consistent or optimal solution.

As for test accuracy, it also shows a fluctuation as the number of epochs increases. This shows that the model is relatively complicated and to get the best performance, a balance between learning and generalization should be achieved.

3.2 The Effects of Batch Size

The condition becomes way more complicated when it comes to the effects of batch size. Accuracy and validation accuracy, as well as loss and validation loss, and even the test accuracy, all observed fluctuations as batch size increases. While larger batches can accelerate learning, they can also hinder the model's ability to capture the nuanced patterns in the training data. As for test accuracy, while it may not improve linearly with batch size, it can peak at a certain point before declining as the batch size becomes excessively large.

The results show that there is a trade-off between computational efficiency and model performance. This is another balance that should be achieved – between short training time and predictive test accuracy.

4 CONCLUSIONS

This study examined the effects of varying epochs and batch sizes on the accuracy of a Keras-based model for text classification. Through multiple experiments using different configurations, the research found that increasing the number of epochs generally improves both training and validation accuracy, allowing the model to learn more complex patterns. However, excessive epochs can lead to overfitting, where the model performs well on the training data but struggles to generalize to new data. This highlights the importance of finding an optimal number of epochs that balances learning and generalization.

Batch size had a more nuanced impact. While larger batch sizes can speed up training time and make the model more computationally efficient, they may also hinder the model's ability to learn finer patterns in the data. A larger batch size could cause the model to miss subtle but crucial relationships, leading to fluctuations in test accuracy. Therefore, an appropriate balance between batch size and accuracy must be achieved to ensure optimal performance without sacrificing training speed.

Overall, the results emphasize the need for careful tuning of model parameters, such as epochs and batch size, to maximize the efficiency and accuracy of text classification tasks using machine learning models.

REFERENCES

Chen, S. H., & Pollino, C. A. 2012. Good practice in Bayesian network modelling. Environmental Modelling & Software, 37, 134-145.

Chung, K. L. 1967. Markov chains. Springer-Verlag, New York.

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. 2022. A survey on text classification algorithms: From text to predictions. Information, 13(2), 83.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. 2019. Text classification algorithms: A survey. Information, 10(4), 150.

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. 2021. Deep learning-based text classification: A comprehensive review. ACM computing surveys (CSUR), 54(3), 1-40.

Mirończuk, M. M., & Protasiewicz, J. 2018. A recent overview of the state-of-the-art elements of text classification. Expert Systems with Applications, 106, 36-54.

Norris, J. R. 1998. Markov chains (No. 2). Cambridge university press.

Pang, B., Nijkamp, E., & Wu, Y. N. 2020. Deep learning with tensorflow: A review. Journal of Educational and Behavioral Statistics, 45(2), 227-248.

Song, Y. Y., & Ying, L. U. 2015. Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry, 27(2), 130.

Stephenson, T. A. 2000. An introduction to Bayesian network theory and usage.

Suthaharan, S., & Suthaharan, S. 2016. Decision tree learning. Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning, 237-269.