Part 1

What other features could you use to determine the language? Please include additional linguistic features to your machine learning model for this task.

Because of differences in scripts and character sets, we chose characters and words as the linguistic features for this task. Some languages have non-alphabetic characters, such as Japanese, Chinese, and Korean, which we need to process as characters, while in others, we can treat them as words.

Report the hyperparameter combination for your best-performing model on the test set.

For feature extraction model, the word-level TF-IDF vectorizer, the best-performing parameter is:
Max Features: 5000, N-gram Range: (1, 2)
For character-level TF-IDF vectorizer, the best-performing parameter is:
Max Features: 5000, N-gram Range: (1, 2)
For classification model, the logistic regression model, the best-performing parameter is:
Regularization Parameter (C): 10, Multi-Class: Multinomial, Solver: Newton-CG

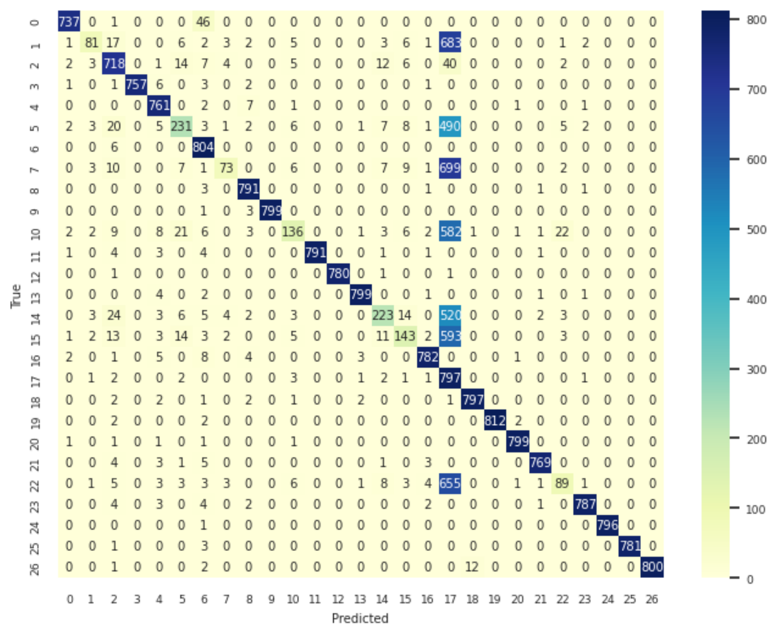
What is the advantage of grid search cross-validation?

During the task, we found the advantages of grid search cross-validation can be summarized as follows:

1) Assistance in parameters selection: automatically assisting to choose the best hyperparameters combinations for enhanced model works;
2) Efficiently explores different combinations of hyperparameters, critical for accommodating language-specific settings.
3) Assesses model performance across multiple data subsets to mitigate the risk of overfitting.
4) Improves transparency and supports reproducibility of research.

Use a confusion matrix to do your error analysis and summarize your answers in your report.

Confusion Matrix before Fine-tune on Training Data

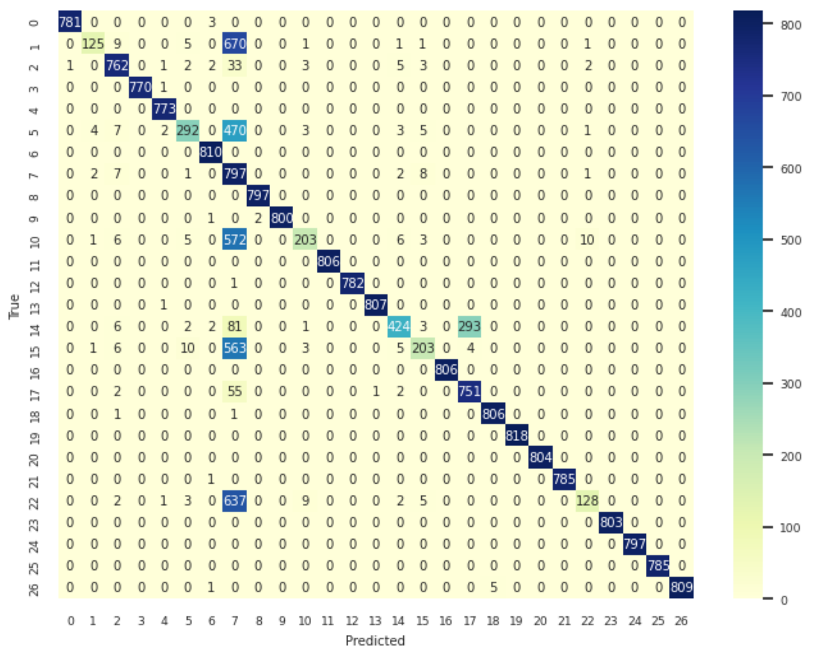


```
        label  label_encoded
128091    tur             25
181983    ita             13
173183    fra              8
60200     deu              4
130655    ell              5
30077     ara              1
97733     nno             19
13631     lzh             17
125180    fas              7
129368    ang              0
27338     hat              9
66737     jpn             14
177997    chr              2
17789     spa             23
115757    dan              3
231195    por             21
67542     ind             11
37014     nld             18
143862    lat             16
69365     hye             10
3192      rus             22
59120     isl             12
89084     kor             15
123481    pol             20
157481    swe             24
153887    eng              6
39989     zea             26
```

The confusion matrix graph before fine-tune indicates that class 19(Norwegian Nynorsk) has the highest number of correct predictions. Classes 6(English), 8(French), 9(Haitian Creole), 11(Indonesian), 12(Icelandic), 13(Italian), 16(Latin), 17(Literary Chinese), 18(Dutch), 20(Polish), 21(Portuguese), 23(spa), 24(Swedish), 25(Turkish), and 26(Zeeuws) have relatively higher numbers.

Conversely, Classes 1(Arabic), 7(Persian), and 22(Russian) exhibit the lowest prediction accuracy. The 18th column represents class 17(Literary Chinese), which was frequently misclassified in instances where other classes, such as class 22(Russian), 7(Persian), and 1(Arabic), were the correct labels.

Confusion Matrix after Fine-tune on Training Data

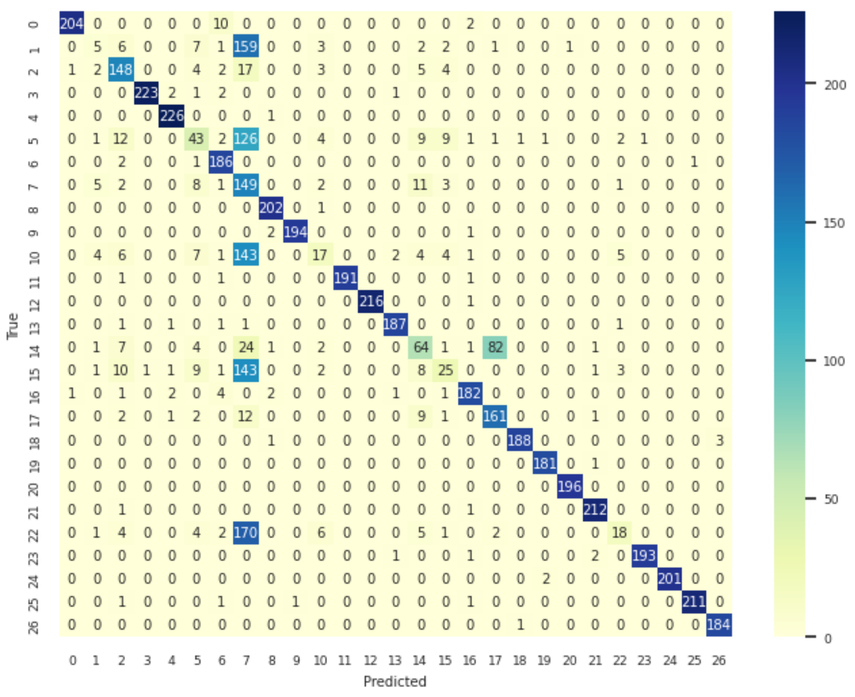


True \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
0	781	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	125	9	0	0	5	0	670	0	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0
2	1	0	762	0	1	2	2	33	0	0	3	0	0	0	5	3	0	0	0	0	0	0	2	0	0	0	0
3	0	0	0	770	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	773	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	4	7	0	2	292	0	470	0	0	3	0	0	0	3	5	0	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	810	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	2	7	0	0	1	0	797	0	0	0	0	0	0	2	8	0	0	0	0	0	0	1	0	0	0	0
8	0	0	0	0	0	0	0	0	797	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1	0	2	800	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	1	6	0	0	5	0	572	0	0	203	0	0	0	6	3	0	0	0	0	0	0	10	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	806	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	1	0	0	0	0	782	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	1	0	0	0	0	0	0	0	0	807	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	6	0	0	2	2	81	0	0	1	0	0	0	424	3	0	293	0	0	0	0	0	0	0	0	0
15	0	1	6	0	0	10	0	563	0	0	3	0	0	0	5	203	0	4	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	806	0	0	0	0	0	0	0	0	0	0
17	0	0	2	0	0	0	0	55	0	0	0	0	0	1	2	0	0	751	0	0	0	0	0	0	0	0	0
18	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	806	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	818	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	804	0	0	0	0	0	0
21	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	785	0	0	0	0	0
22	0	0	2	0	1	3	0	637	0	0	9	0	0	0	2	5	0	0	0	0	0	0	128	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	803	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	797	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	785	0
26	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	809

After fine-tune, the confusion matrix on training data shows that Class 19(Norwegian Nynorsk) maintains the highest number of correct predictions. We can observe that the prediction accuracy has increased for most of the classes, though Class 17(Literary Chinese) has seen a decrease in numbers.

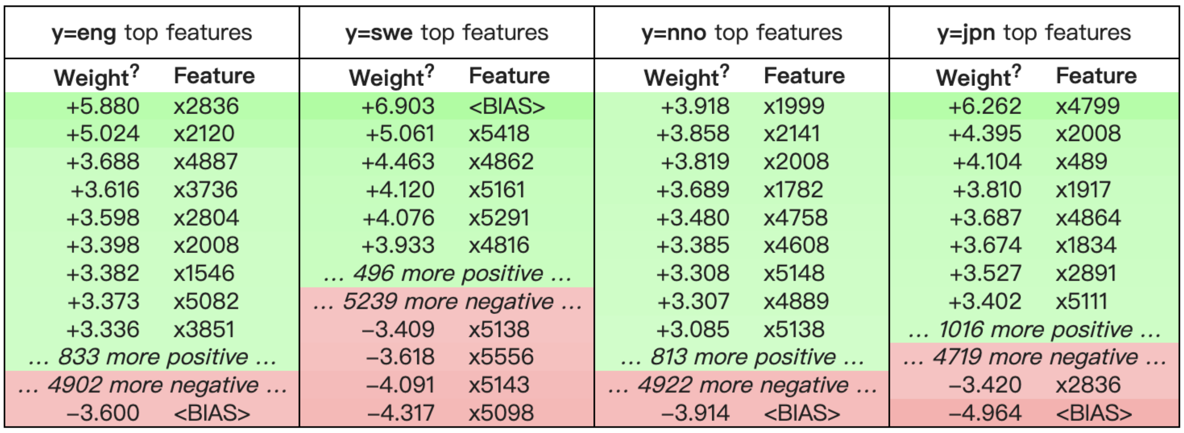
At the same time, elements with high off-diagonal values have reduced in Class 17(Literary Chinese). Class 7(Persian) has witnessed a significant increase in correct predictions, from 73 to 797, but also making it the most frequently mispredicted language, Class 1(Arabic), 5(Modern Greek), 10(Armenian), 15(Korean), 22(Russian) are wrongly predicted as Class 7(Persian) in this case. Even though Class 7(Persian) is no longer the lowest-ranked class, Class 1(Arabic) has now taken that position.

Confusion Matrix after Fine-tune on Test Data



Generally, the confusion matrix after fine-tune on test data has the similar pattern with training data after fine-tune, the prediction accuracy is highest with Class 3(Danish), 4(German), while the model perform not so well on Class 1(Achinese), 5(Modern Greek), 10(Armenian), 14(Japanese), 15(Korean), 22(Russian), and we can still observe the similar mistake predictions occurring in Class 7(Persian), and Class 17(Literary Chinese).

Generate a feature importance table for the top ten features (please have the features named) for the languages English, Swedish, Norwegian, and Japanese. What is more important, extra features or the outputs of the vectorizer, discuss.



y=eng top features		y=swe top features		y=nno top features		y=jpn top features	
Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature
+5.880	x2836	+6.903	<BIAS>	+3.918	x1999	+6.262	x4799
+5.024	x2120	+5.061	x5418	+3.858	x2141	+4.395	x2008
+3.688	x4887	+4.463	x4862	+3.819	x2008	+4.104	x489
+3.616	x3736	+4.120	x5161	+3.689	x1782	+3.810	x1917
+3.598	x2804	+4.076	x5291	+3.480	x4758	+3.687	x4864
+3.398	x2008	+3.933	x4816	+3.385	x4608	+3.674	x1834
+3.382	x1546	*… 496 more positive …*		+3.308	x5148	+3.527	x2891
+3.373	x5082	*… 5239 more negative …*		+3.307	x4889	+3.402	x5111
+3.336	x3851	−3.409	x5138	+3.085	x5138	*… 1016 more positive …*	
… 833 more positive …		−3.618	x5556	*… 813 more positive …*		*… 4719 more negative …*	
… 4902 more negative …		−4.091	x5143	*… 4922 more negative …*		−3.420	x2836
−3.600	<BIAS>	−4.317	x5098	−3.914	<BIAS>	−4.964	<BIAS>

Text features (in this case TF-IDF weights) typically contribute significantly to classification tasks because they contain information about the text content. Extra features provide additional context and background information to further enhance the model's performance.

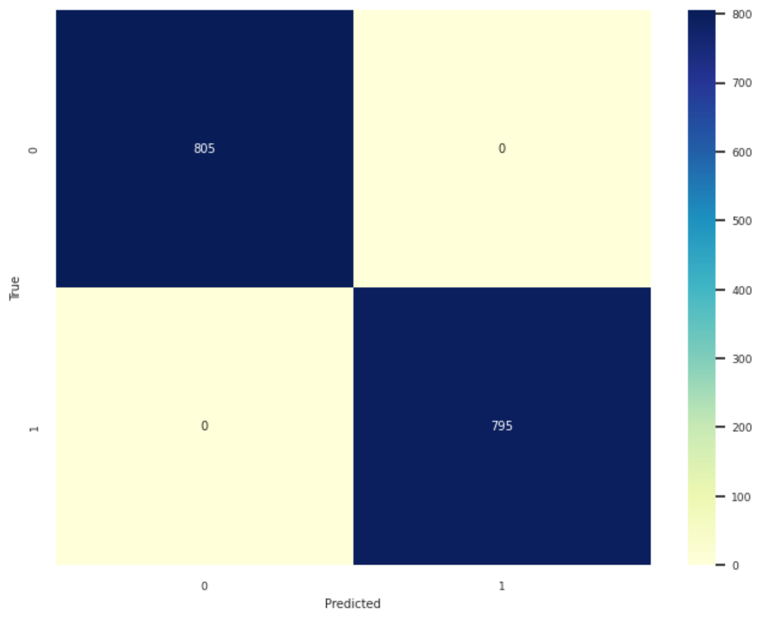
- In the context of English language classification, the features x2836 and x2120 are pivotal, bearing weights of 5.880 and 5.024, respectively. These features play a significant role in distinguishing the English language.
- For Swedish language classification, the bias term in the logistic regression model has a substantial influence on classification outcomes. Additionally, the feature x5418 holds significant importance, contributing notably to the classification results.
- In the case of Norwegian language classification, features x1999, x2141, and x2008 stand out with the highest weights across the feature set. Their considerable weights indicate their primary role in the classification process.
- Japanese language classification is notably impacted by the feature x4788, boasting the highest weight of 6.262. This particular feature plays a decisive role in determining the classification results.

First, choose the two languages for which the classifier worked best. Next, re-fit the best working model several times, each time reducing the number of characters per instance in the training set (1. All characters, 2. 500 characters, 3. 100 characters). How does the ablation affect the performance of the classifier?

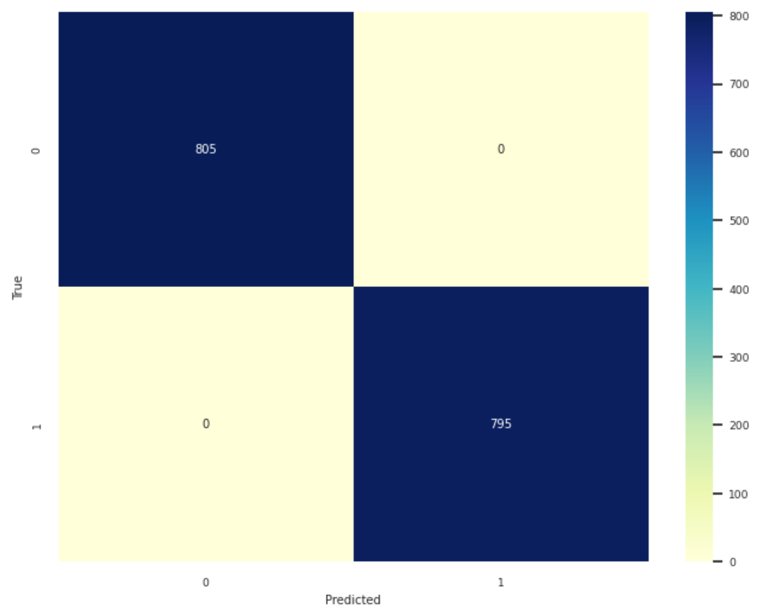
The F1-score of Polish and Swedish is equal to 1, so we choose the two languages from the whole set, and then split them into training dataset and test dataset to re-fit the best working model.

The result turns out that the precision, recall, and f1-score of the three conditions(all characters, 500 characters and 100 characters) are the same, all equal to 1.0, which means that there are no false positives or false negatives, and all predictions are correct.

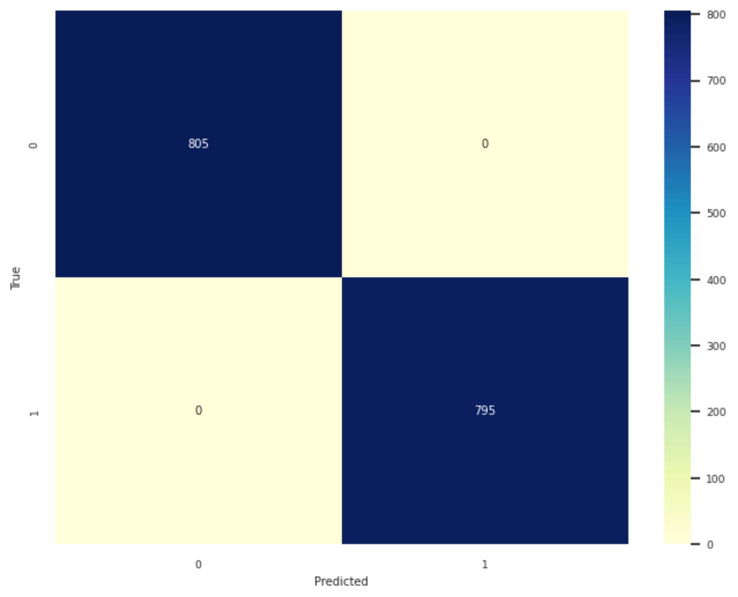
the Confusion Matrix of All Characters



the Confusion Matrix of 500 Characters



the Confusion Matrix of 100 Characters



That is, in such a scenario, it suggests that the classifier is able to make accurate predictions regardless of the character limit in the training data. But this is not a universal result, there are possible reasons that can explain this:

1. Vectorization: It is possibly because the 'max_features' of TF-IDF vectorization in our best model is 5000, which may includes all the unique features in the text data, so regardless of the length of the characters in each sentence, as a result, when the classifier trains on this rich feature set, it may perform exceptionally well, achieving perfect scores.
2. Language Characteristics: The Polish and Swedish for evaluation may have characteristics that make them easily distinguishable even with limited text.
3. Feature Extraction: The features used by the classifier may be highly indicative of language identity, regardless of the amount of text available.

Part 2

Please improve the neural network to at least up to 80% accuracy. You can also use GridSearchCV but be aware that training a neural network takes much more time. Play around with 5 different sets of hyperparameters including layer sizes, activation functions, solvers, early stopping, vectorizer parameters, and report your best hyperparameter combination. Do you achieve higher performance? Why/why not? Importantly, please use the same data splits as for Part 1.

Firstly, we splitted a validation data set form the train set, to prevent overfitting. We then changed num_classes to 27 and used xngram_range=(2,2), max_features=100 (original setting) for the initial model training. At this point we observed a steady increase in validation accuracy. The architecture seems to effectively capture patterns in the data. However, a low validation accuracy (65.37%) suggests that the model needs to be optimized.

We maintained the setting of xngram_range=(2,2) but increased max_features to 1000. After this adjustment, the model showed a significant improvement in performance. By the 20th epoch, it reached a validation accuracy of 96.57% with a validation loss of 0.1409. This adjustment allowed the model to more effectively capture the data patterns, although there was a minor increase in validation loss in the later epochs.

Next, we attempt to add a weights Initialization, to increase the model's accuracy. After the introduction of a weight initialisation , the model showed slight performance improvements. It reached a validation accuracy of 96.83% by the 20th epoch. While the training loss consistently declined, the validation loss experienced a slight increase in the latter epochs same as the above one.

Subsequently, after adding early stopping with a patience of 5, we performed a grid search over five sets of hyperparameters:

- Layer Sizes: ‘module__num_units’: [100, 200, 300] ,
- Activation Functions: ‘module__nonlin’: [F.relu, torch.sigmoid],
- Learning Rate: ‘lr’: [1, 0.1],
- Epochs: 'max_epochs': [10, 20]
- optimizer: [torch.optim.SGD, torch.optim.Adam]

The optimal parameters were identified as:
'lr': 0.1, 'max_epochs': 20, 'module__nonlin': <function relu at 0x7a6a85279b40>,
'module__num_units': 200, 'optimizer': <class 'torch.optim.sgd.SGD'>

Finally, we updated these parameters and re-trained the model. The results show that the training loss of the model effectively decreased from 2.7510 to 0.0440 and the validation accuracy improved from

66.64% to 96.50%. However, there doesn't seem to be any significant improvement when comparing the results before and after changing the hyperparameters.
Both models have a similar final training loss of 0.0440. In addition, the model with the original hyperparameters slightly outperformed the model with the updated hyperparameters (96.83% vs. 96.50%). In short, the updated parameters did not produce a better result than the original ones.

We believe potential reasons for this could be:

- Overfitting: We observed that the training loss continues to decrease while the validation loss starts to increase. We took this as an indication that the model may have over-adapted to the training data. To further improve this model, we might consider dropout, changing the architecture, or other techniques to address this issue.
- Restricted Choice of Hyperparameters: Due to computational constraints, we didn't utilize a vast range of sets for each hyperparameter. Given the limited search space for hyperparameters, there's a possibility that the optimal combination was not explored.

We tested the finalized model and recorded the following outcomes:

Training Accuracy: 98.99%
Validation Accuracy: 96.50%
Test Accuracy: 96.41%