Midterm Project Report

Lingxiao Zhao

February 22, 2025

1 Problem Definition

In today's highly competitive luxury fashion retail and e-commerce environment, how to quickly and accurately identify the brand of a fashion item through product description is a practical problem. Generally speaking, brand identification is not a big problem because it is obviously part of the product description. However, in many scenarios (especially multi-platform distribution, second-hand transactions, or user self-posting information), products only provide limited text descriptions - there may be no clear brand name, or even an ambiguous nickname. This situation may be even worse in offline communication.

This project aims to build a text classification model that can automatically identify the brand of a fashion item based on its text description (such as style, material, design style, origin, etc.). Specifically:

Input: A text description of a fashion item (including size, color, material, etc., but not explicitly listing the specific brand name).

Output: The brand label of the fashion item

	brand	description	price_usd	type
0	NIKE	Zoom Vomero 5 Rubber-Trimmed Mesh and Leather ...	160	shoes
1	NIKE	Dunk Low Retro PRM NBHD Leather-Trimmed Canvas...	120	shoes
2	SAINT LAURENT	Teddy Polished-Leather Monk-Strap Boots	1190	shoes
3	BALENCIAGA	3XL Distressed Mesh and Rubber Sneakers	1150	shoes
4	DUNHILL	Audley Leather Penny Loafers	875	shoes

Figure 1: Test data sample

2 Dataset curation

The original data set of this project is selected from mr-porter.csv in Net-a-Porter/Mr Porter Fashion Dataset on the Kaggle website. Mr Porter is an authoritative fashion item purchasing website, which contains sufficient fashion item data.

I selected 18 specific brands from this large dataset. The original large-scale data set contains more than 100 brands, some of which appear very rarely in the data, which will result in a large number of niche brands with only a small number of samples, making it difficult for the training model to learn robust features; too many rare brands will cause the model to favor brands with higher frequencies, which will distort the training results.

By filtering out major brands (e.g., keeping only those brands that appear more frequently and are the most representative/mainstream in the fashion industry in the filtered file), we can reduce the sparsity and noise of the data and improve the recognition accuracy of the model for mainstream brands. And

in actual application, unlike other categories of everyday clothes, the most popular brands in fashion and luxury clothes are often the mainstream or popular brands in the market. And during the screening process, the core fields of the data are standardized to remove interference items such as spaces.

For specific data management, X_desc, X_price, y_brand represent the model's input text (description), numerical features (price), and target label (brand) respectively. Use MinMaxScaler() to scale the price to the [0,1] interval to scale the price variable difference to avoid instability in subsequent optimization due to price extreme value differences. Use BertTokenizer to encode the text description and set max_length = 50 to ensure that abnormally long text does not affect data consistency.

3 Word Embeddings and Algorithm Training

For Word Embeddings, as mentioned in the previous paragraph, Bert is used to convert the original text into a series of integer IDs. Each ID corresponds to a token in the BERT vocabulary. Padding+Truncation is used to ensure uniform input size. Descriptions with less than 50 words will be padded, and texts with more than 50 words will be truncated. The model uses a 768-dimensional embedding vector randomly initialized by Keras to represent it.

At this stage, the project used two models, CNN and RNN, for training. The CNN version of the model is defined in the function create_model, using multiple layers of convolution + GlobalMaxPooling. In the function, the three layers of Conv1D use 128, 64, and 32 convolution kernels respectively, the convolution kernel size is set to 5, and the activation function is ReLU. GlobalMaxPooling is used to perform global maximum pooling on the output of the last layer of CNN to retain the most obvious sentence signal. Set batch_size = 32, epochs = 150 to observe the network convergence trend.

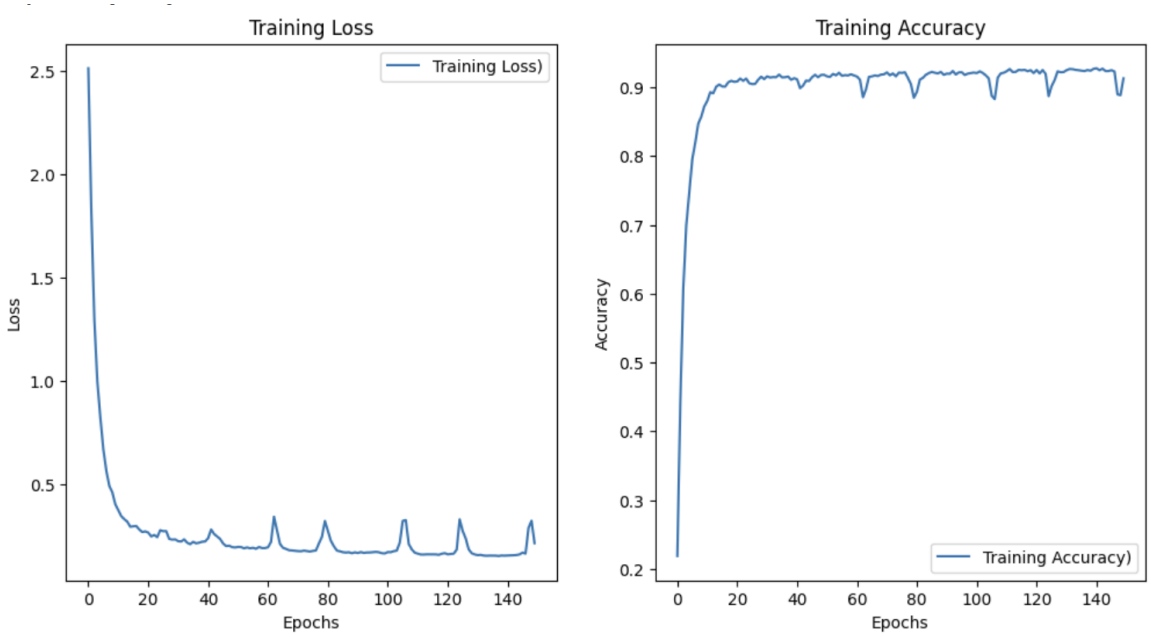

Figure 2: CNN with batch size = 32 and epoch = 150

The RNN algorithm is defined in the create_rnn_model function. The Embedding layer used by the RNN is the same as the CNN model, based on vocab_size = BERT vocabulary size and 768 dimensions. In the RNN model, multi-layer LSTM + GlobalMaxPooling was selected, and the three-layer LSTM contained 128, 64, and 32 hidden units respectively for time series calculation. Similarly, GlobalMaxPooling is used to perform maximum pooling on the output sequence of the last layer of LSTM to

map the sequence into a fixed-size vector. In terms of training, epochs=150, batch_size=32 are also selected, and the Adam optimizer is used for training.

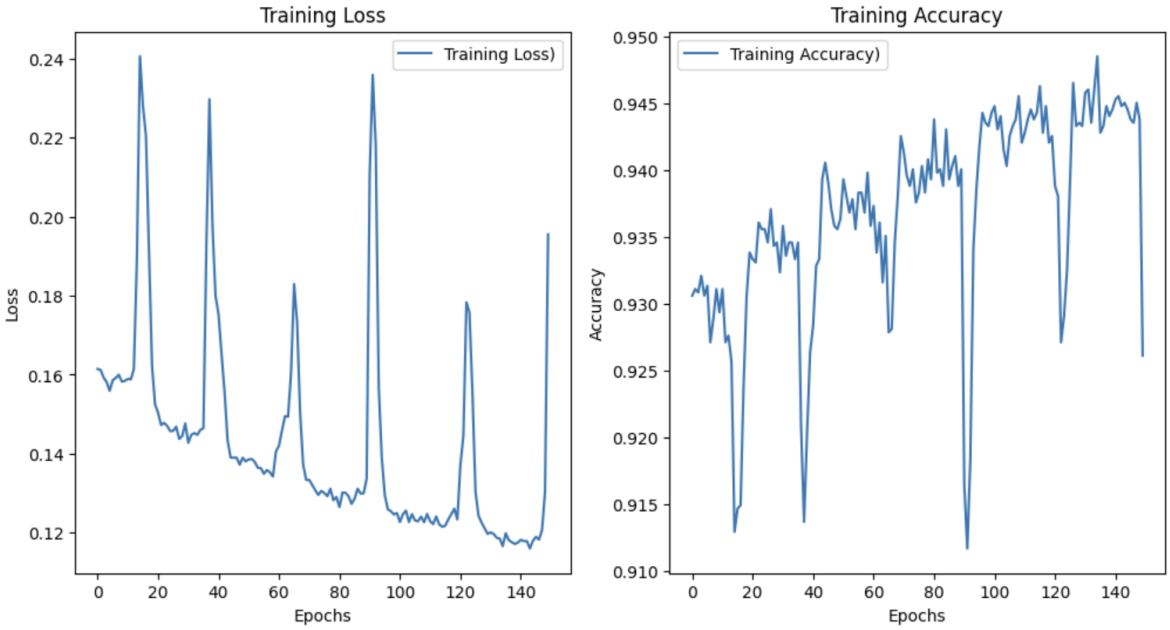

Figure 3: RNN with batch size = 32 and epoch = 150

According to data feedback, the training did not converge and oscillate when the batch_size was adjusted to 32. Therefore, additional training with batch_size of 128 was performed for CNN and RNN.

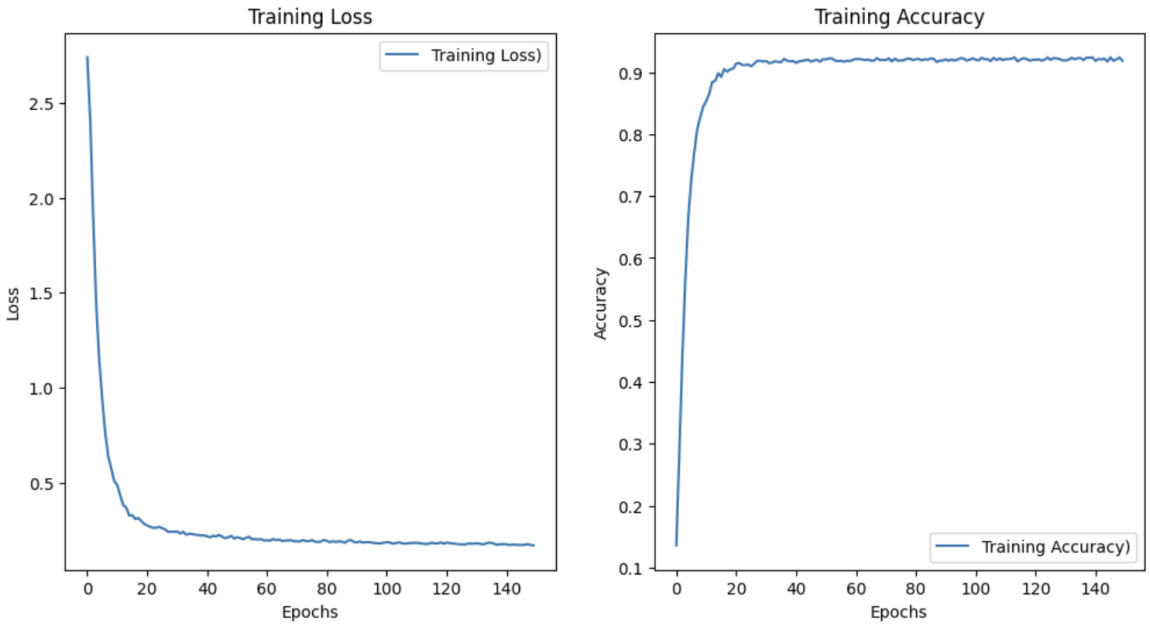

Figure 4: CNN with batch size = 128 and epoch = 150

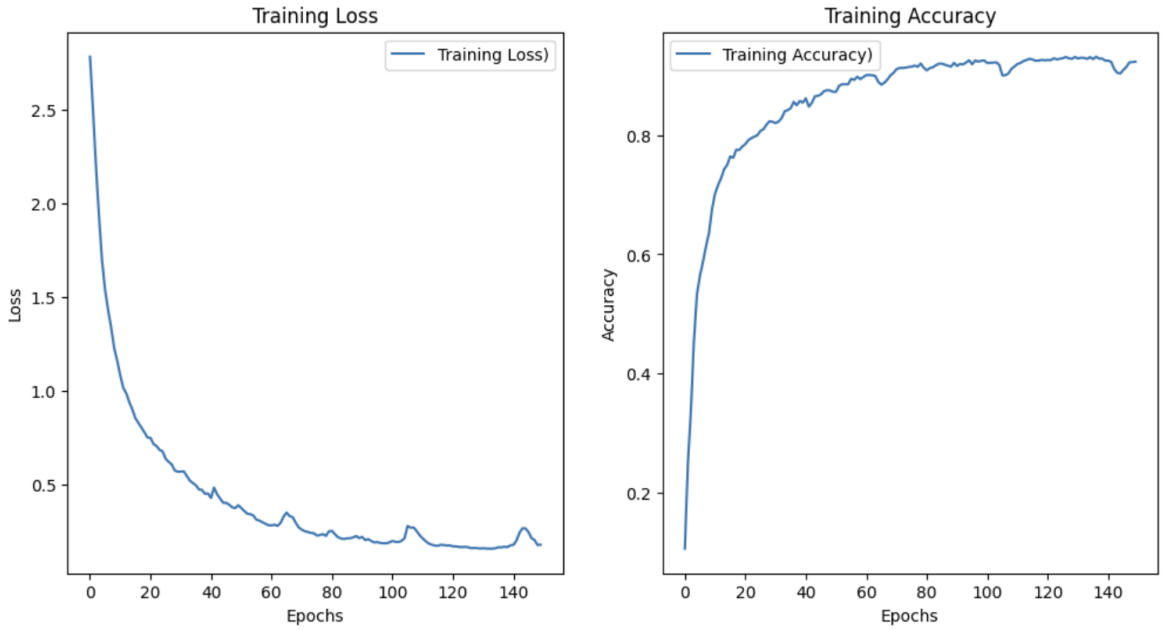

Figure 5: RNN with batch size = 128 and epoch = 150

4 Results and Presented

During deep learning training, the choice of batch_size has a significant impact on the model's training process, convergence speed, and accuracy. According to observations, when batch_size is adjusted to 32, the model does not converge and oscillates, especially for the RNN model. A smaller batch_size means that fewer samples are used for each weight update, so the gradient calculated each time is more susceptible to the influence of a single sample or a small batch of samples. In this way, the gradient will fluctuate more, resulting in oscillations in loss and accuracy. This will eventually lead to large noise and unstable updates, both in CNN and RNN. A smaller batch_size usually results in a larger step size for each update, which makes it easy to cross the local optimal point, resulting in oscillation.

For this project, a batch_size of 128 is obviously more suitable. After setting the batch_size to 128, it can be seen that the fitting curve is smoother. As for the setting of the number of rounds, it is more appropriate to set the epoch to around 100, because it can be seen that the results have converged before 100 rounds. For the max_length setting, a token length of 50 is also appropriate, because the description training samples in this example are all within one sentence.

For presentation, this project visualizes the confusion matrix to show the correspondence between the true label (vertical axis) and the predicted label (horizontal axis) of each brand. The larger the diagonal grid value, the more times the brand is correctly identified. We can see that for most brands (such as Balenciaga, Gucci, Celine Homme, etc.) there is a high number of correct predictions on the 128-size CNN diagonal. For RNN, the overall brand performance is worse, but for some brands, the RNN stacking layer can capture more temporal information, such as Saint Laurent.

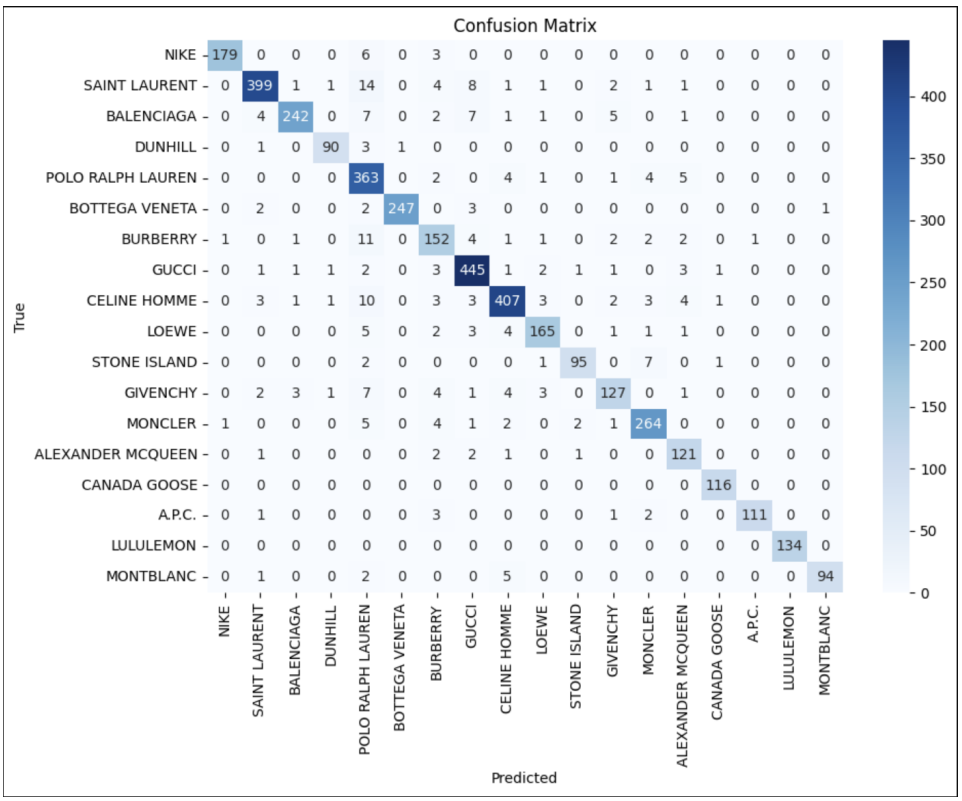

Figure 6: CNN with batch size = 128 and epoch = 150

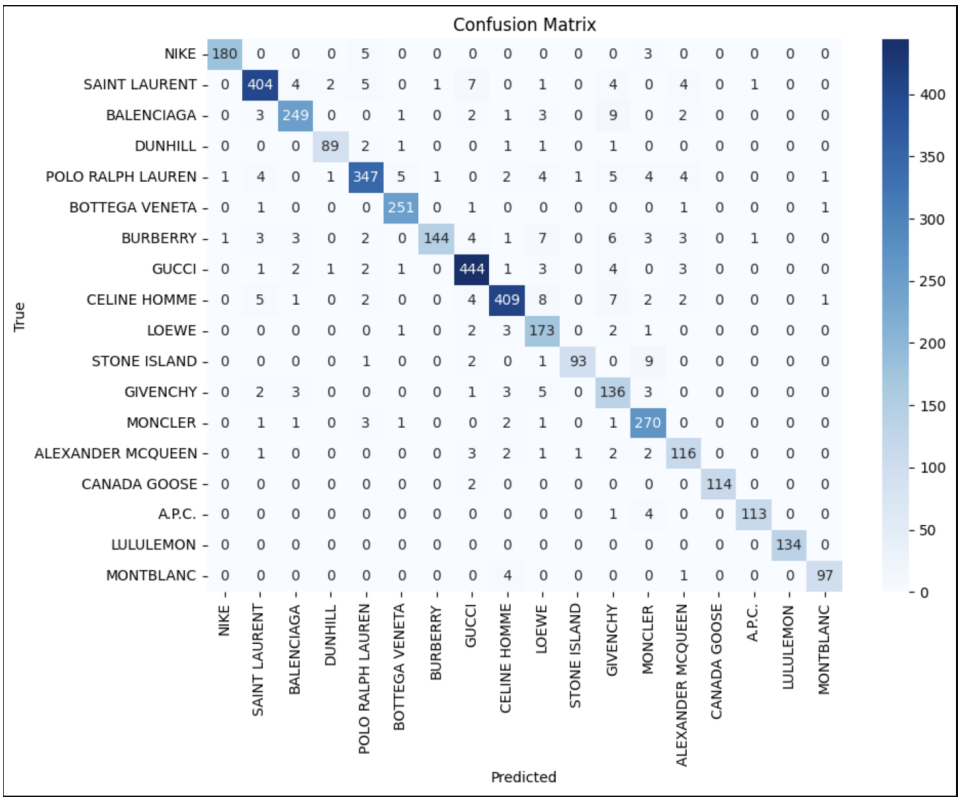

Figure 7: RNN with batch size = 128 and epoch = 150

In the overall comparison between RNN and CNN, RNN shows extremely violent turbulence at 32 size, while their Loss and Accuracy curves converge similarly during the fitting process at 128 size. We can see the gradual convergence and gradual stabilization process as the model is iterated. In general, the larger the batch_size is, the more stable the gradient update is during the fitting process and the better the curve convergence is. However, in the RNN, there are still several small fluctuations at batch_sizes of 70, 110, and 140, indicating that a single batch_size still has some limitations.

5 In-depth analyses/experiments

5.1 RNN segment training

Since RNN shows large fluctuations before and after adjusting the batch_size, more experiments on batch_size should be conducted on RNN. In previous experiments, using a single batch_size=32 caused problems with training curve oscillation and unstable convergence in RNN. To address this problem, the project tried to divide the RNN training into two stages. For the first half of the epoch, batch_size=32 is used to allow the model to perform more frequent parameter updates in smaller batches; batch_size=128 is used in the second half to reduce the jitter caused by gradient updates.

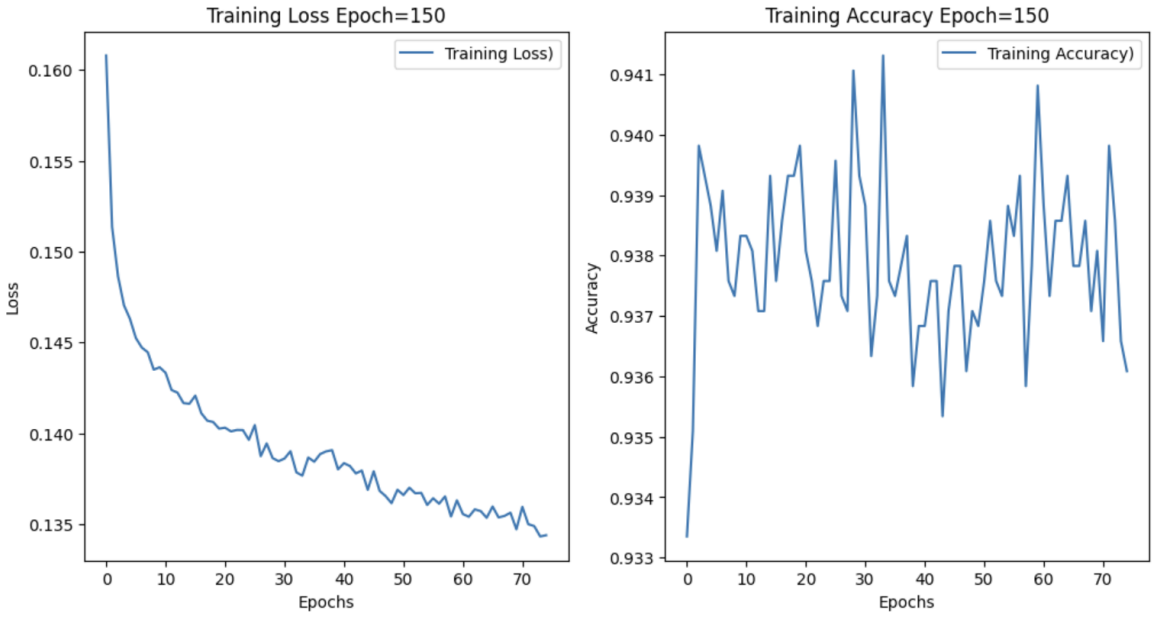

Figure 8: RNN mixed batch size training

For the initial epochs stage, the loss dropped rapidly from around 0.16 to around 0.145, and in the middle and late stages the curve continued to slowly drop to a minimum of around 0.135. The accuracy rate climbed from around 0.933 to a maximum of around 0.94, with some jumps and fluctuations during the period. For the initial epochs stage, the loss dropped rapidly from around 0.16 to around 0.145, and in the middle and late stages the curve continued to slowly drop to a minimum of about 0.135. The accuracy climbed from around 0.933 to a maximum of around 0.94, with some jumps and fluctuations during the period.

In the final convergence process, the mixed size training showed a trend of further optimization, and the performance results also had some ups and downs. Combined with the similar operations of the above CNN, it shows that a large batch_size does help stabilize the gradient in the later stage, thereby impacting higher accuracy and lower loss values.

5.2 RNN and CNN hybrid model training

In previous experiments, pure CNN and pure RNN models were tried respectively, and both have some advantages in capturing text features. Therefore, the project conducted an exploration of "hybrid CNN + RNN", hoping that the combination of CNN's local convolution advantages and RNN's temporal feature extraction capabilities can further improve brand recognition.

As before, the project also tried two batch_size of 32 and 128.

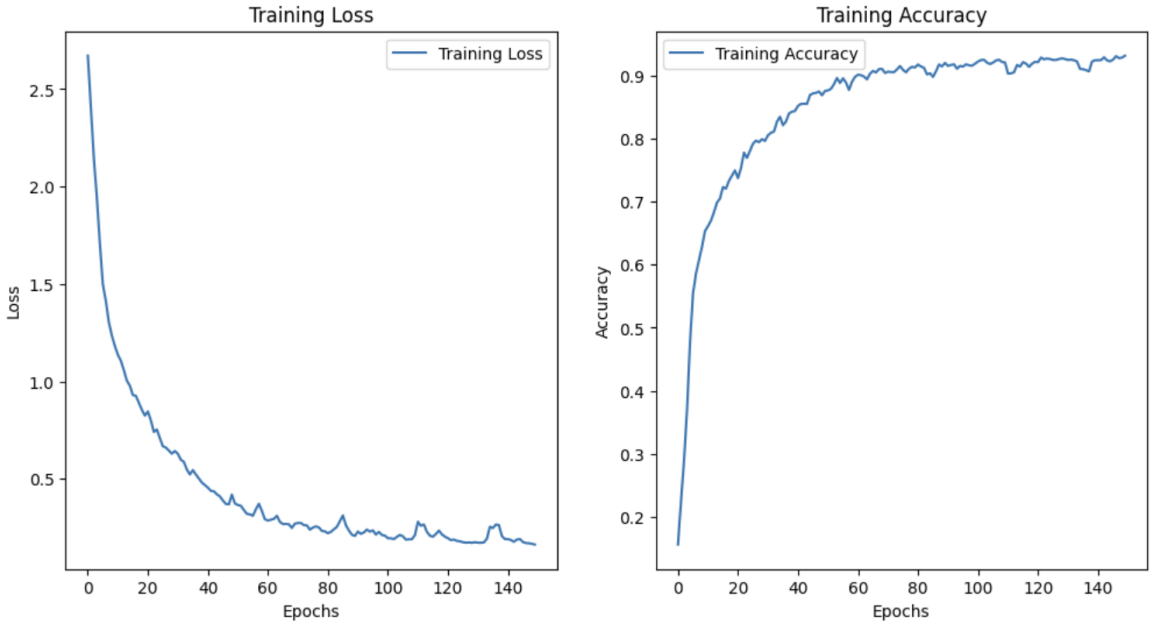

Figure 9: Hybrid Model with batch size = 32 and epoch = 150

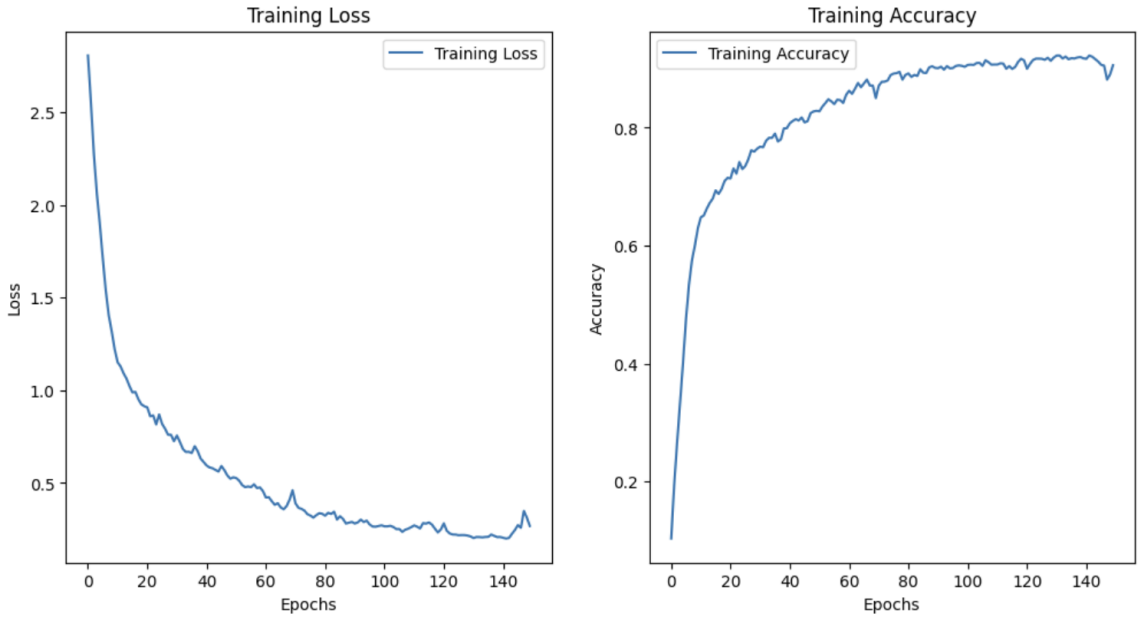

Figure 10: Hybrid Model with batch size = 128 and epoch = 150

However, at both sizes, the performance of the hybrid model is inferior to that of using either CNN or RNN alone. This may be because when the network structure is too complex (CNN + RNN multi-layer stacking), a generalization bottleneck may have appeared at this data scale. In order to conduct further experiments, the project chose to remove the last two convolutional and recurrent layers, reduce the overall network depth, and keep batch_size = 128 for training again.

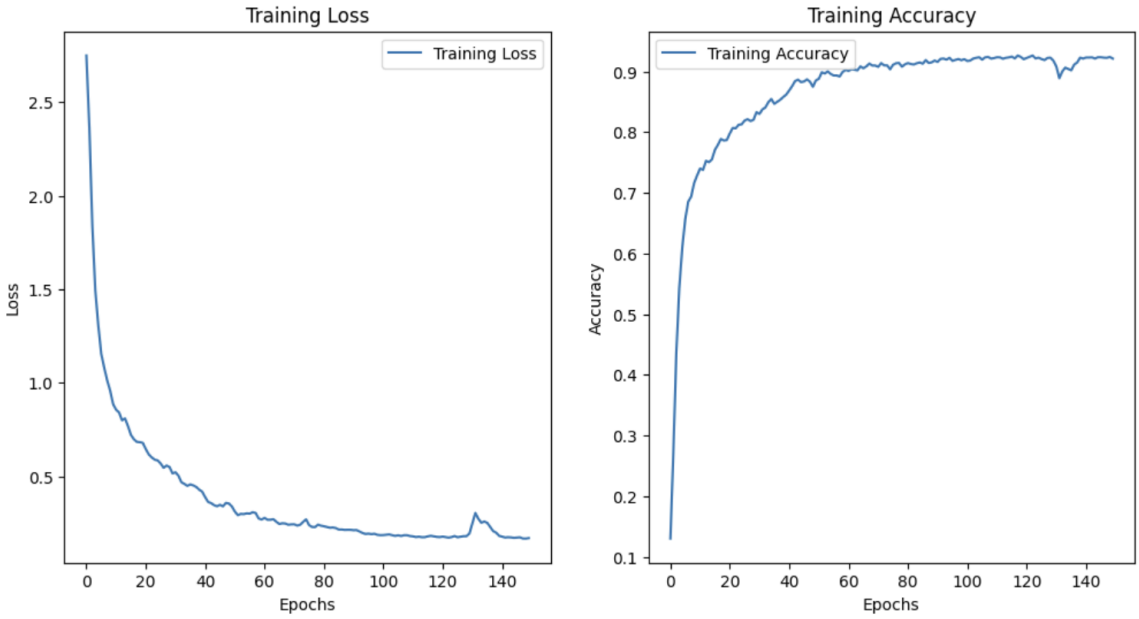

Figure 11: Simplified Hybrid Model with batch size = 128 and epoch = 150

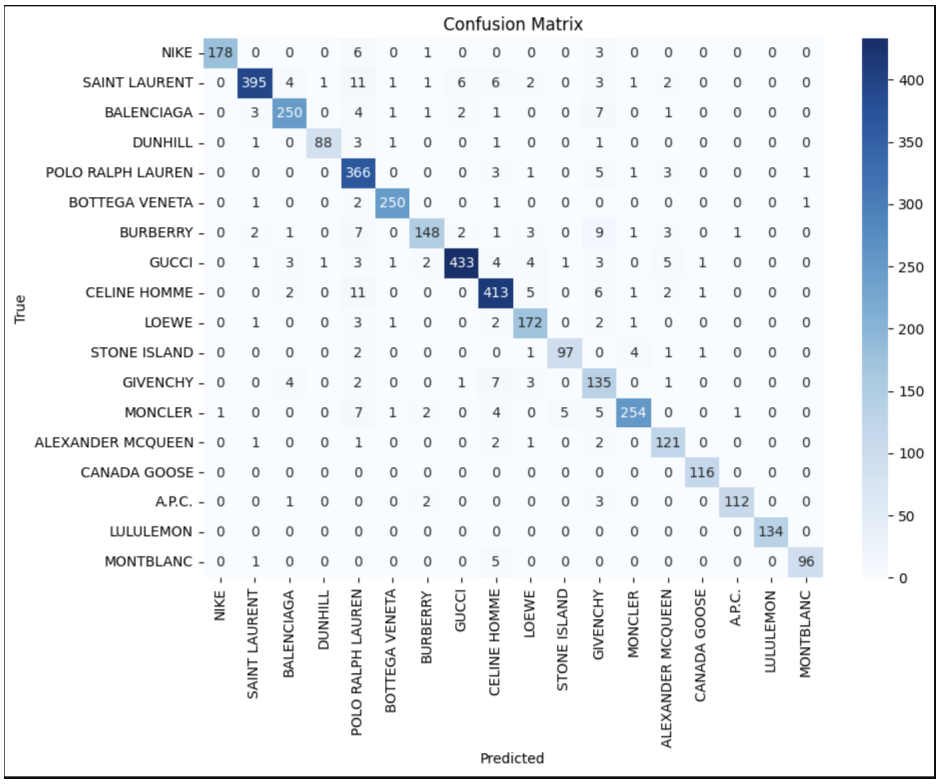

Figure 12: Simplified Hybrid Model with batch size = 128 and epoch = 150

According to the Loss and Accuracy curves and the confusion matrix, most mainstream brands still maintain a high recognition rate on the diagonal. The Accuracy curve is more stable, and the process of continuously climbing from the initial accuracy of 0.1 to 0.9 0.92 is faster than before reducing convolution and loop. In the later stage, it fluctuated slightly in the area around 0.9, and was relatively stable overall. Overall, by reducing the number of layers, the model converges faster and the results are slightly better than the hybrid model with more layers.

In this project, both RNN and CNN can achieve good results, but the effect is slightly worse when using RNN and CNN together, only 0.92. As the number of layers of the hybrid model decreases, the effect improves slightly, which should be due to the fact that the initial hybrid model is too complicated.

6 Lessons and Experience

In the face of the original huge data, filtering out a few relatively mainstream brands with sufficient samples in the filtered file may be a more appropriate approach. From the perspective of model training, this can reduce the noise caused by a large amount of abnormal data, but it may also cause problems in model generalization due to the lack of data outside the filter. However, for the fashion and luxury goods industry, the data focus effect is very large, and other brands except the top brands have almost no room for survival. This is another reason for data filter.

CNN can quickly capture local features, and RNN is good at temporal dependencies, but in this project the difference between the two is almost negligible. The hybrid model of CNN and RNN should theoretically have better performance, but in practice it is not as good as either CNN or RNN. This is because the data in this project is too small to support a deep network, and the brand characteristics are not sufficient. For hybrid models, not only will performance be improved, but it is also possible that $1 + 1 < 2$ will happen like this project.

For the adjustment of batch_size, small batch updates can make the gradient more noisy, which can help the model jump out of the local minimum more quickly and accelerate the learning of core features. However, this will cause subsequent training curve jitter and instability, which is especially evident in RNN. In this project, large batches (batch_size=128) are better than small batches in all cases. Large batches can reduce the variance of the gradient and can significantly make the model converge in a more gentle direction.

As learned in class, visualization of confusion matrices can help analyze specific data. The diagonal elements can help you see which brands have stable classification effects, and can also quickly identify brand combinations that are easily confused. The loss and accuracy curves are also very meaningful. By continuously tracking the trend of the curves, we can see overfitting, convergence bottlenecks or oscillations during the training process.