Deep Learning

Developing a convolutional neural network (CNN) for image classification

By Huzaifa Patel

Abstract

In this paper I conducted a comparative study into the analysis of deep learning models developed to classify images into 3 types of clothing: Casual, Formal and Smart Casual. This study details the model architecture, training processes, evaluation and results.

Baseline convolutional neural network (CNN)

Introduction

The goal of this project is to develop a Conventional Neural Network (CNN) capable of classifying thousands of images of a variety of clothing items into 3 categories: Casual, Formal and Smart Casual.

Methodology and Development

This section outlines the steps I have taken to plan, implement and evaluate the CNN. The key stages of the methodology for this model include dataset preparation, data preprocessing, model logic methodology, and training methodology.

Given the impressively large database of clothing items, I started by preparing the dataset. This consisted of a csv file (styles.csv) containing data for over 40,000 clothing items as well as the images directory which contains an image of every item. These files contained in the fashion_data directory are the key resources for the model to label the images and identify the specific category of clothing.

The data preprocessing steps included ensuring the images were correctly labelled based on the information in the csv file.

I split the dataset into 3 sets: training, validation and testing to best assess the model's capability of image classification. The training set is used to build the model with multiple model parameter settings and then the trained model is challenged with the validation set (Xu Y, Goodacre R, 2018). The test set is an independent dataset used to evaluate the model's performance after training. It is important to have an additional blind test set which is not used during the model selection and validation process to have a better estimation of the generalization performance of the model (Xu Y, Goodacre R, 2018).

The training set provided by the client consists of a dataset of thousands of clothing items and includes various information including gender, master category, subcategory, colour, season, year and name. Also included is the 'usage' and the 'id' which are critical information for the model. The usage identified the item's category of clothing including Casual, Ethnic, Formal, Sports and Smart Casual, with some irregularities that would need to be addressed. The id is the identifier for the item in the directory 'images' which contained images of over 40,000 clothing items.

This directory was divided into 3 categories Training, Validation and Testing using a 70%, 15%, 15% split. I sorted the data by usage which gives the following:
Casual: rows 2-34415, Formal: rows 37624-39982, Smart-Casual: rows 40329-40395
The row counts and ratios for each category are as follows:
Casual: 34,414 rows (93.4%).
Formal: 2,359 rows (6.4%).
Smart Casual: 67 rows (0.2%).
These datasets are imbalanced with the Casual images overwhelmingly dominating the underrepresented Formal and Smart Casual images and this greatly affected the model's ability to effectively identify the usage of clothing images. Imbalanced datasets are an important issue for CNNs as they consider using large datasets and therefore accuracy and performance are of big concern (Iren Valova, Et al., 2020). This issue is further explored in the analysis of the more accurate CNN model with the implementation of data augmentation.

The initial step in image preprocessing is to resize all the images used by the model to a uniform size in order to achieve standardized inputs for the neural network, however this was not necessary as in this case the images provided by the client

are all 60px x 80px. This normalization of input data allows the model to train more efficiently and allows more computational power to process the data.

One-Hot Encoding

The information contained in the 'usage' row for all the clothing items included categories of clothing such as Casual and Sports. This CNN model is solving a multi-class classification problem, solving the issue of categorizing clothing into 3 types. In order to allow for the model to solve this, I applied one-hot encoding to the label of each relevant category (Casual, Formal and Smart Casual), converting Casual to [1,0,0], Formal [0,1,0] and Smart Casual [0,0,1]. One-hot encoding transformed the labels into a form that is suitable for the neural network to calculate the loss and accuracy of the model.

Model Architecture

The architecture of the model begins with a Conv2D layer with 32 filters. Convolutional layers are important layers which extract the important features from images. This layer is followed by a MaxPooling2D layer with a 2x2 pool size, reducing the dimensions of the image while maintaining the features essential for identification. Another Conv2D and MaxPooling2D layer follows to gradually reduce the image size and cause the model to focus on the important features.
The flattened output from these layers is passed through the Dense layer which has 60 neurons and uses the ReLU activation to allow the model to learn complex patterns from the inputs.
The output layer has 3 neurons with one neuron for each of the 'usage' categories of clothing and uses the softmax activation. This layer is where the neural network makes the final decision in classification, choosing the highest probability class based on the information provided by the previous layers.
The model was trained for 10 epochs and has a smaller batch size of 5. The validation set, which consists of 15% of the overall dataset (6667 images), was used during training to monitor the model's performance on unseen data.

The optimizer used is 'adam'. Adam builds upon the idea of norms and momentum and has two decaying sums consisting of the first and second momentums and also incorporates a bias correction to improve some of the instability that can occur when bias terms are initialized to zero. (Iren Valova, Et al., 2020)
The systematic methodology and development of the CNN model was created by firstly identifying all the potential problems, then systematically targeting each problem during development. This model was developed to serve as a baseline for the more comprehensive model that is featured later in this paper.

Evaluation of results

The model's performance was assessed through the following metrics: Test accuracy, Test loss, Validation accuracy, Validation loss.

```
Epoch 1/10
28/28 [==============================] - 3s 45ms/step - loss: 0.7194 - accuracy: 0.8500 - val_loss: 0.6897 - val_accuracy: 0.8333
Epoch 2/10
28/28 [==============================] - 1s 19ms/step - loss: 0.4960 - accuracy: 0.8786 - val_loss: 0.5895 - val_accuracy: 0.8333
Epoch 3/10
28/28 [==============================] - 0s 18ms/step - loss: 0.4647 - accuracy: 0.8786 - val_loss: 0.4695 - val_accuracy: 0.8333
Epoch 4/10
28/28 [==============================] - 0s 18ms/step - loss: 0.3623 - accuracy: 0.8786 - val_loss: 0.4822 - val_accuracy: 0.8333
Epoch 5/10
28/28 [==============================] - 0s 18ms/step - loss: 0.3283 - accuracy: 0.8786 - val_loss: 0.4650 - val_accuracy: 0.8333
Epoch 6/10
28/28 [==============================] - 1s 18ms/step - loss: 0.2920 - accuracy: 0.8857 - val_loss: 0.4390 - val_accuracy: 0.8333
Epoch 7/10
28/28 [==============================] - 1s 18ms/step - loss: 0.2210 - accuracy: 0.9071 - val_loss: 0.3656 - val_accuracy: 0.8333
Epoch 8/10
28/28 [==============================] - 0s 18ms/step - loss: 0.1867 - accuracy: 0.9143 - val_loss: 0.4013 - val_accuracy: 0.8333
Epoch 9/10
28/28 [==============================] - 1s 18ms/step - loss: 0.1989 - accuracy: 0.9286 - val_loss: 0.2877 - val_accuracy: 0.8667
Epoch 10/10
28/28 [==============================] - 0s 17ms/step - loss: 0.1838 - accuracy: 0.9214 - val_loss: 0.3972 - val_accuracy: 0.8333
```

Figure 1: Training and Validation accuracy and loss over 10 epochs – CNN1

The training accuracy gradually increased from 85% to 92.14%. Meanwhile, the validation accuracy started at 83.33% at the first epoch and stayed the same until it increased to 86% on epoch 9. The higher accuracy achieved on the training set compared to the validation set suggests overfitting. This is further evident when looking at the loss values, which initially decreased for validation but then increased over each epoch while the training loss continued to decrease.

Both Formal and Smart-casual clothing items are underrepresented in the dataset which explains the trend in the validation score between the 3 categories. The percentage representation of the 3 classes was Casual-93.4%, Formal-6.4% and

Smart Casual: 0.2% with only 67 clothing items listed in the data. As such, the model was ineffective at identifying Smart-Casual clothes.

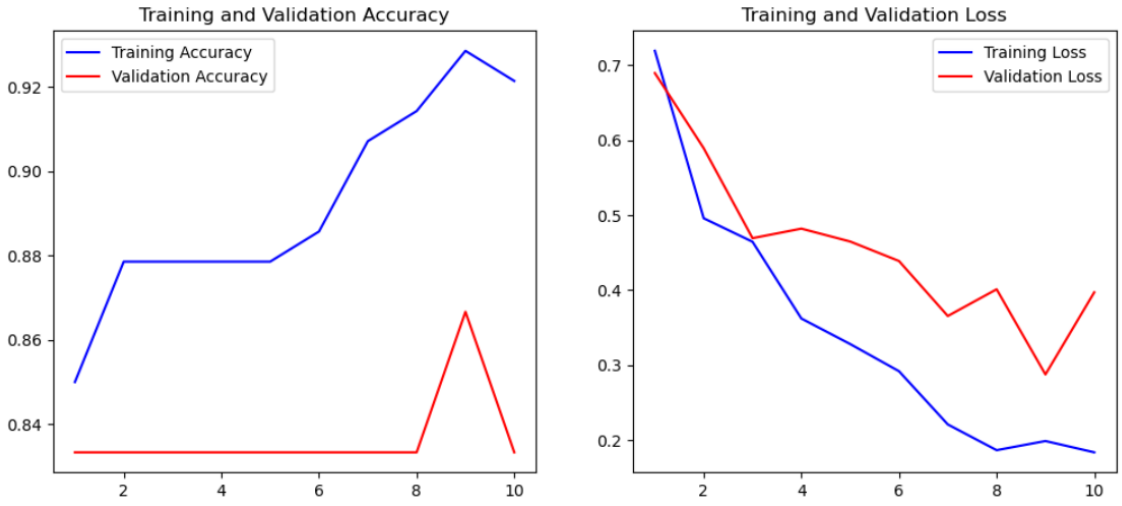


Figure 2: Training and validation accuracy & Training and validation loss – CNN1

The graph generated using matplotlib shows the training accuracy generally increasing until reaching near accurate results by the 10th epoch, displaying how the model is learning from the training data and improving in accuracy. The validation accuracy is mostly consistent and does not show a trend as the model is training. This suggests that there is overfitting as the results should be steadily changing as they have with the training accuracy.

The training loss shows a very steep decline in epochs 1-8 but increases in epoch 9, which is supported by the decrease in training accuracy in epoch 10 shown in the left graph, suggesting signs of underfitting. After epoch 9 we see the training loss continue to decline until it is close to 0. The validation loss generally shows a decline across the epochs however the progress is not continuous as shown by the increase in epoch 9.

These results show the model improves in training accuracy, reaching near perfect accuracy by epoch 10, however the validation accuracy does not show the same progress across the training. This suggests that the model is very much prone to overfitting at this stage. This will be addressed in the development of the complete model using data augmentation to improve the number of Formal and Smart-casual images available to the model.

Refining the model and Experimentation

Data Augmentation

The dataset for this project was heavily unbalanced and in order to solve this problem I decided to implement data augmentation, a technique that applies transformations to images in order to artificially increase the size of the data available to the model. There is a significant gap between the performance estimated from the validation set and the one from the test set for all the data splitting methods employed on small datasets. Such disparity decreased when more samples were available for training/validation. (Xu Y, Goodacre R, 2018) Displaying the class distribution of the dataset gave the following output.

```
Training dataset class distribution:
Casual: 123 images
Formal: 8 images
Smart Casual: 9 images

Validation dataset class distribution:
Casual: 25 images
Formal: 3 images
Smart Casual: 2 images

Test dataset class distribution:
Casual: 25 images
Formal: 3 images
Smart Casual: 2 images
```

Figure 3: Dataset class distribution

This shows the class distribution is highly imbalanced with Formal and Smart Casual categories containing very few images compared to Casual.

The techniques used in data augmentation included rotating the image, shearing to distort the image, flipping the image horizontally and filling the generated pixels during transformation. I then adjusted the number of newly generated images to closely match the number of causal images in the training, validation and test datasets.

```
Training dataset class distribution:
Casual: 845 images
Formal: 812 images
Smart Casual: 850 images

Validation dataset class distribution:
Casual: 200 images
Formal: 220 images
Smart Casual: 250 images

Test dataset class distribution:
Casual: 278 images
Formal: 208 images
Smart Casual: 250 images
```

Figure 4: Dataset class distribution after data augmentation

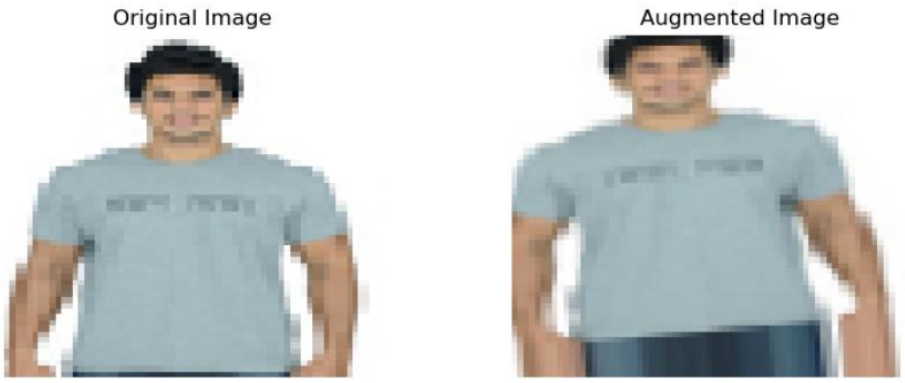
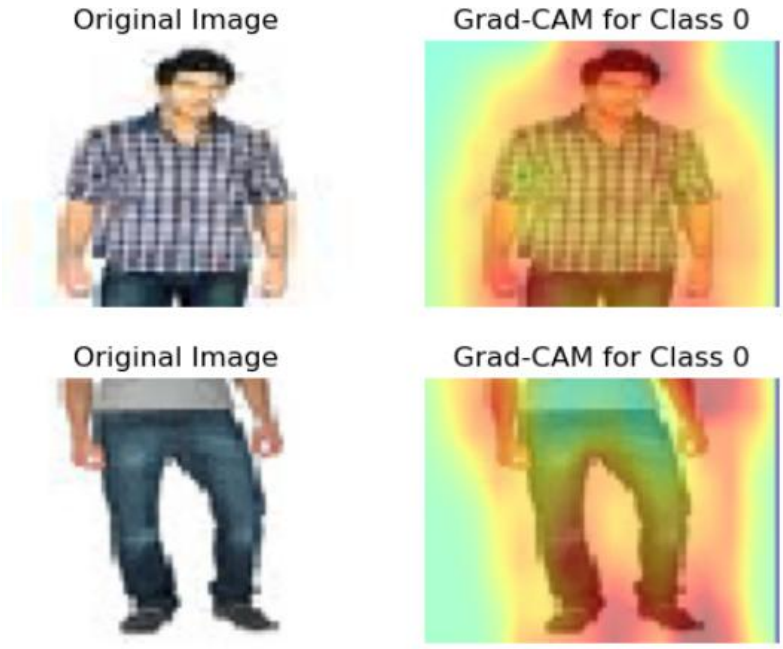


Figure 5: Results of data augmentation on the dataset

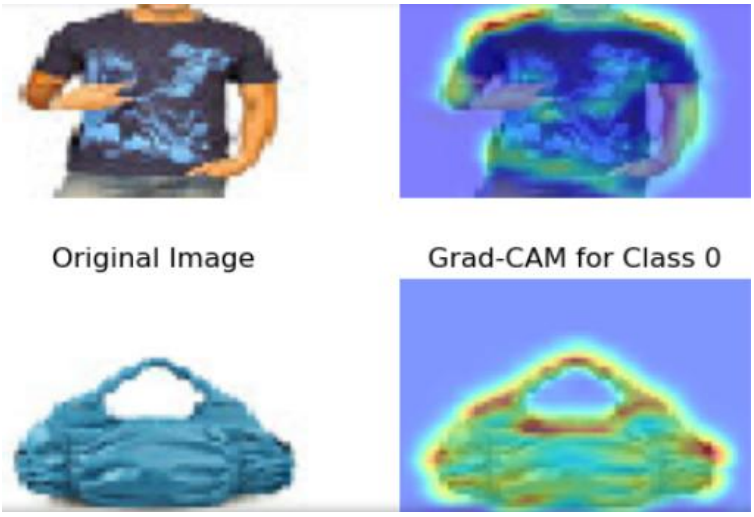
By introducing variation of images, the model will now learn to recognize specific patterns in the images. As well as this, there will now be a diverse training, validation and testing set which will reduce overfitting and prevent the model from specifically training on casual images as they were previously the largest category.

Visualisation

GRAD-CAM produces a heatmap to highlight different regions of the image that the model identifies as important for image classification. These heatmaps were generated for each class during evaluation. After viewing the image, I had a general sense of where the model could be improved.



GRAD-CAM image of the simple CNN model



GRAD-CAM image of the improved CNN model

From these images we can see the difference between the 2 models. The first model focuses on the shape of the clothing item whereas the improved model only focuses on the shape of the item where it is relevant, in this case it does so with the handbag. The improved model also identifies the model's pose, which it has seen to have a correlation with the category. Models wearing casual clothing change the position of their arms compared to those wearing smart-casual or formal clothing.

Transfer learning (ResNet50)

I utilized the pre-trained model ResNet50 as a feature extractor in the model for the improved CNN. This architecture was formed to defeat quandaries in deep learning training because deep learning training, in general, takes quite a lot of time and is limited to a certain number of layers. (*Dewi Sarwinda*, Et al., 2021) I added custom dense and dropout layers to adapt the model to the specific fashion categories. ResNet brought a high level of accuracy from it's deep architecture, which helped the model to generalize much better.

This is a brief overview of the results

```
Epoch 1/10
280/280 [==============================] - 19s 62ms/step - loss: 0.3293 - accuracy: 0.9200 - val_loss: 0.1445 - val_accuracy: 0.9833
Epoch 2/10
280/280 [==============================] - 17s 62ms/step - loss: 0.2081 - accuracy: 0.9393 - val_loss: 0.0883 - val_accuracy: 0.9900
Epoch 3/10
280/280 [==============================] - 17s 62ms/step - loss: 0.1583 - accuracy: 0.9543 - val_loss: 0.1362 - val_accuracy: 0.9733
Epoch 4/10
280/280 [==============================] - 17s 61ms/step - loss: 0.1121 - accuracy: 0.9600 - val_loss: 0.1453 - val_accuracy: 0.9567
Epoch 5/10
280/280 [==============================] - 17s 61ms/step - loss: 0.0854 - accuracy: 0.9671 - val_loss: 0.1870 - val_accuracy: 0.9367
Epoch 6/10
280/280 [==============================] - 17s 62ms/step - loss: 0.0787 - accuracy: 0.9729 - val_loss: 0.1198 - val_accuracy: 0.9667
Epoch 7/10
280/280 [==============================] - 17s 61ms/step - loss: 0.1472 - accuracy: 0.9664 - val_loss: 0.1399 - val_accuracy: 0.9600
Epoch 8/10
280/280 [==============================] - 17s 62ms/step - loss: 0.0938 - accuracy: 0.9671 - val_loss: 0.1241 - val_accuracy: 0.9700
Epoch 9/10
280/280 [==============================] - 17s 62ms/step - loss: 0.0513 - accuracy: 0.9829 - val_loss: 0.1974 - val_accuracy: 0.9500
Epoch 10/10
280/280 [==============================] - 17s 62ms/step - loss: 0.0269 - accuracy: 0.9936 - val_loss: 0.2710 - val_accuracy: 0.9167
```

Figure 8: Training and Validation accuracy and loss over 10 epochs

The training accuracy gradually increased from 92% to 99.36%. Meanwhile, the validation accuracy started at 98.33% at the first epoch and dropped to 91.67%. The higher accuracy achieved on the training set compared to the validation set suggests overfitting. This is further evident when looking at the loss values, which initially decreased for validation but then increased over each epoch while the training loss continued to decrease.

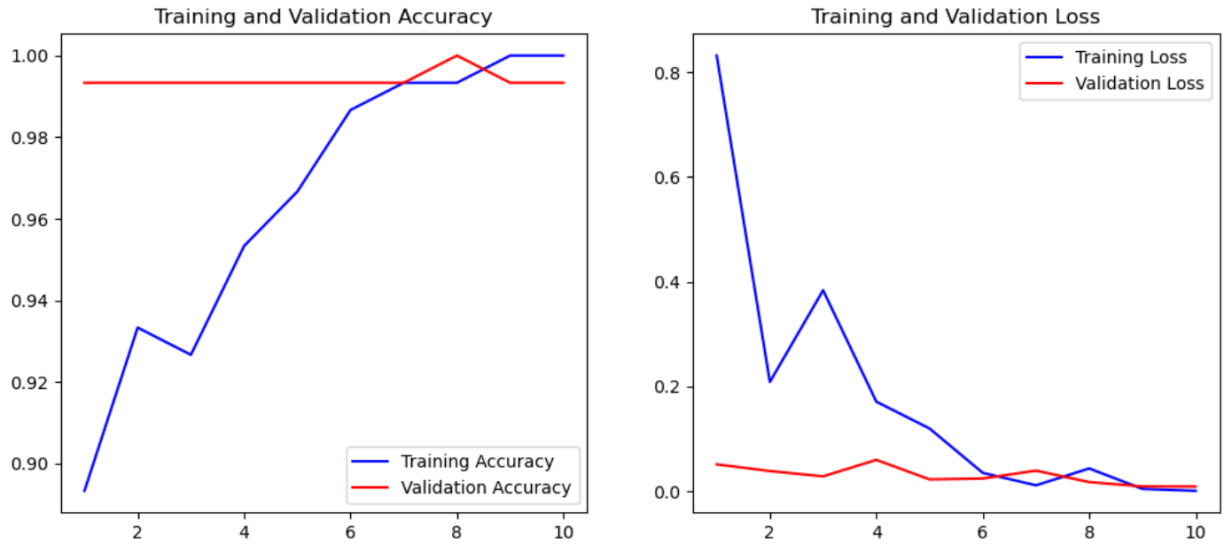


Figure 9: Training and validation accuracy & Training and validation loss

The graph generated using matplotlib shows the training accuracy generally increasing until reaching near accurate results by the 10th epoch, displaying how the model is learning from the training data and improving in accuracy. The validation accuracy is mostly consistent and does not show a trend as the model is training. This suggests that there is overfitting as the results should be steadily changing as they have with the training accuracy.

The training loss shows a very steep decline in epochs 1-2 but increases in epoch 3, which is supported by the decrease in training accuracy in epoch 3 shown in the left graph, suggesting signs of underfitting. After epoch 3 we see the training loss continue to decline until the 10th epoch where it is close to 0. The validation loss generally shows a decline across the epochs however the progress is not continuous as shown by the increase in epochs 4 and 7.

These results show the model improves in training accuracy, reaching near perfect accuracy by epoch 10.

Comparison of models

Architecture

First model:
For this model I built a CNN with 2 Conv2D layers with ReLU activation followed by MaxPooling2d. A Flatten layer followed, feeding into 2 Dense layers and used softmax for the final activation. This model achieved modest accuracy but struggled with balancing the dataset as the Formal and Smart Casual images were heavily underrepresented.

Improved model:
I worked on implementing Transfer Learning into the model and added custom dense layers which greatly helps the model to avoid underfitting. (*Devvi Sarwinda*, Et al., 2021) For this I decided to use ResNet50 as it is highly efficient in image classification.

Data augmentation techniques allowed me to balance the images used in the train, test and validate datasets by only increasing the number of Formal and Smart Casual examples.

In order to allow visualization, I implemented GRAD-CAM which displayed heat maps. These images allowed me to see exactly which features the model was focusing on when deciding to classify an image and examples of these heat maps are evident above.

Comparison with Published Results

Gender, Season, Color, Usage Classification

Notebook Input Output Logs Comments (0)

```
2408.3s  234  Epoch [95/100], Step [200/251], Loss: 1.1000
2423.3s  235  Epoch [96/100], Step [100/251], Loss: 1.1797
2432.7s  236  Epoch [96/100], Step [200/251], Loss: 1.3092
2447.7s  237  Epoch [97/100], Step [100/251], Loss: 1.1431
2457.2s  238  Epoch [97/100], Step [200/251], Loss: 1.2263
2472.4s  239  Epoch [98/100], Step [100/251], Loss: 1.2086
2482.5s  240  Epoch [98/100], Step [200/251], Loss: 1.0531
2496.5s  241  Epoch [99/100], Step [100/251], Loss: 1.3761
2506.7s  242  Epoch [99/100], Step [200/251], Loss: 1.1040
2521.7s  243  Epoch [100/100], Step [100/251], Loss: 1.1291
2531.6s  244  Epoch [100/100], Step [200/251], Loss: 1.2012
```

```
Epoch 1/10
280/280 [==============================] - 19s 62ms/step - loss: 0.3293 - accuracy: 0.9200 - val_loss: 0.1445 - val_accuracy: 0.9833
Epoch 2/10
280/280 [==============================] - 17s 62ms/step - loss: 0.2081 - accuracy: 0.9393 - val_loss: 0.0883 - val_accuracy: 0.9900
Epoch 3/10
280/280 [==============================] - 17s 62ms/step - loss: 0.1583 - accuracy: 0.9543 - val_loss: 0.1362 - val_accuracy: 0.9733
Epoch 4/10
280/280 [==============================] - 17s 61ms/step - loss: 0.1121 - accuracy: 0.9600 - val_loss: 0.1453 - val_accuracy: 0.9567
Epoch 5/10
280/280 [==============================] - 17s 61ms/step - loss: 0.0854 - accuracy: 0.9671 - val_loss: 0.1870 - val_accuracy: 0.9367
Epoch 6/10
280/280 [==============================] - 17s 62ms/step - loss: 0.0787 - accuracy: 0.9729 - val_loss: 0.1198 - val_accuracy: 0.9667
Epoch 7/10
280/280 [==============================] - 17s 61ms/step - loss: 0.1472 - accuracy: 0.9664 - val_loss: 0.1399 - val_accuracy: 0.9600
Epoch 8/10
280/280 [==============================] - 17s 62ms/step - loss: 0.0938 - accuracy: 0.9671 - val_loss: 0.1241 - val_accuracy: 0.9700
Epoch 9/10
280/280 [==============================] - 17s 62ms/step - loss: 0.0513 - accuracy: 0.9829 - val_loss: 0.1974 - val_accuracy: 0.9500
Epoch 10/10
280/280 [==============================] - 17s 62ms/step - loss: 0.0269 - accuracy: 0.9936 - val_loss: 0.2710 - val_accuracy: 0.9167
```

Figures 10 & 11: Results published online (kaggle.com) (Görkem KOLA, 2024) and my improved model's results.

This published study achieved a loss of 1.2012 after 100 epochs, whereas my improved model started with a loss of 0.4717 at the first epoch and steadily decreased over epochs. This shows the improvements that can be made by using data augmentation to balance datasets and transfer learning models such as ResNet50. My model was trained for just 10 epochs, compared to Görkem's model which was trained for 100 epochs, and the results show that my model had room for improvement.

Lessons Learned

Using pre-trained models such as ResNet50 including dropout reduces the need to extensively train the model and helps to produce more accuracy in feature extraction, as shown by the heatmaps. These models can have millions of

parameters, so it is important to adjust the weights to ensure the model is identifying the necessary features in the data.

Data augmentation is necessary to balance datasets as it allows the model to equally train each category, leading to higher accuracy. It is important to avoid too much data augmentation as well as setting the incorrect parameters as this could have the opposite effect on model accuracy.

Using visualization tools such as GRAD-CAM is an effective way to see how your model is interpreting the images. This helps with debugging and optimizing the model's parameters to improve accuracy.

Conclusion

The implementation of ResNet50, dropout, data augmentation and visualization have vastly improved the model and vastly increased its accuracy. The data augmentation and transfer learning model were tailored to the dataset and addressed the limitations of the previous CNN model. This model can now classify images of fashion products more reliably and achieves these results while requiring fewer epochs than both the first model and the published results.

References

Xu Y, Goodacre R. On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. 2018 Oct 29. (Xu Y, Goodacre R, 2018)

https://pmc.ncbi.nlm.nih.gov/articles/PMC6373628/

Optimization of Convolutional Neural Networks for Imbalanced Set Classification Iren Valova , Christopher Harris, Tony Mai, Natacha Gueorguieva. 2 October 2020 (Iren Valova, Et al., 2020)

https://doi.org/10.1016/j.procs.2020.09.038

Gender, Season, Color, Usage Classification (Görkem KOLA, 2024)

https://www.kaggle.com/code/grkemkola/gender-season-color-usage-classification/notebook

Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer (january, 2021)

Devvi Sarwinda a, Radifa Hilya Paradisa a, Alhadi Bustamam a, Pinkie Anggia b

https://doi.org/10.1016/j.procs.2021.01.025