AI6126-ADVANCED COMPUTER VISION
Project 1
Fashion Attributes Classification Challenges

NTAMBARA Etienne
G2304253K
ntam0001@e.ntu.edu.sg
MSAI
Nanyang Technological University (NTU-Singapore)
2023-2024

March 21, 2024

---

1 Project Implementation Report

1.1 Model Architecture or Model Used

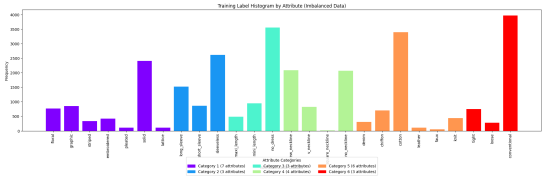
FashionNet, the core of this project, extends the foundational ResNet50 model as a pre-trained model, incorporating significant modifications to tailor it for fashion attribute classification. Key enhancements include the addition of dropout layers to prevent overfitting by randomly omitting subsets of features during training, and batch normalization layers to stabilize and accelerate training by normalizing the input layers. These modifications are instrumental in refining the model's ability to generalize across a wide array of fashion images. more about Resnet50 model [2].

1.2 Loss Functions

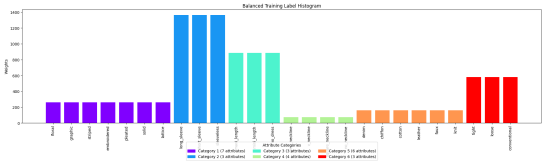
The loss function used for training the model is a sum of **Cross Entropy Losses**, one for each attribute being predicted. The cross-entry loss is computed individually for each classifier's output and then summed to get the total loss. This approach allows the model to simultaneously learn to predict multiple attributes of each fashion item. Additionally, **class weights** are applied to the **Cross-Entropy Loss** to handle class imbalance, ensuring fair treatment of all classes despite differences in their representation in the dataset.

1.3 Class Imbalancing

In addressing class imbalance within our multi-label classification task, the **get_class_weights** method defined the submitted codes it dynamically computes class weights. This process involves collecting labels, determining the frequency of each unique label, and assigning inversely proportional weights to rarer classes. These weights are then used to adjust the loss function during training, ensuring equitable learning across all classes by emphasizing underrepresented labels, thereby improving model performance and fairness [3] [1]. Figure 1 shows both balanced and imbalanced results of our dataset.



(a) Histogram: Before Addressing Class Imbalance



(b) Histogram: After Addressing Class Imbalance

Figure 1: Visualization of Data Distribution for Both Before/After Addressing Class Imbalance

The 1st histogram[1b] presents the original, imbalanced data, with the class frequency depicted as it naturally occurs in the dataset. Some classes are underrepresented (tail classes), and others are overrepresented (head classes), which is a common challenge in machine learning that can bias a model's performance. To handle this is to make a lot of trial and error by applying class weights to the original dataset.

The 2nd histogram [1a] presents the distribution after balancing the classes, where each class within the 6 categories is represented with same weight, regardless of its original frequency. Each category is each category has unique color-coded, showing that classes within the same category have been assigned equal importance. After trial and error I have achieved this balanced results with the help of applying class weights on my orignal dataset.

1.4 Number of Parameters of the Model

number of parameters was printed after training our model then the results of our model are **29826138** parameters.

1.5 GPU USED

The specification of training machine used on Kaggle are those listed:

- Total number of GPUs: 1
- GPU Model: NVIDIA Tesla P100 PCIe with 16 GB memory

Figure 2, shows the specifications of training machine used during the model training.

```
Thu Mar 21 19:16:43 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.129.03             Driver Version: 535.129.03   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla P100-PCIE-16GB           Off | 00000000:00:04.0 Off |                    0 |
| N/A   35C    P0              27W / 250W |      0MiB / 16384MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
```

Figure 2: Specifications of training machine used

1.6 Training Curves And Evaluation Metrics (i.e., Loss, Accuracy for the Validation Set)

- **Loss:** Figure 4 on the left side shows the Training Loss, which decreases as the model learns, indicating optimization of model parameters. Also, Validation Loss decreases, which suggests that the model is generalizing well to new data.
- **Average Loss:** Figure 3 on the left side shows that the Training Loss is a rapid decrease initially and then stabilizes, indicating that the model quickly learns to minimize errors on the training set. while the Validation Loss follows the training loss closely, decreasing significantly and reaching stability, which is a good sign that the model is not overfitting and generalizing well to unseen data.

- **Accuracy:** On the right of Figure 4, the plot shows Training Accuracy, which increases over time, reflecting the model's improving performance on the training set. Validation Accuracy follows a similar upward trend, proving that the model's learning applies to unseen data.
- **Average Accuracy:** Training and Validation Data from Figure 3 The model achieves steady performance on both seen and unseen data, as evidenced by accuracy curves that show a sudden rise and then plateau, demonstrating rapid learning and high generalization.
- **Average Accuracy Per Class:** The curves represent the Training and Validation Average Per-Class Accuracy, respectively, on the right of 3. The fact that both curves rise indicates that the model's prediction performance is constant for several classes, including training and validation sets.
- **Precision, Recall, F1 Score:** From Figure 5 The blue curves in the 1st, 2nd, and 3rd plots represent Training Precision, Recall, and F1 Score, while the red curves represent the respective metrics for the validation set. The steady increase in these curves across epochs shows that the model is becoming more accurate, better at identifying true positives and achieving a balance between precision and recall as proved by the F1 Score.

  Figure 4 and 3 show the training curves for our project.

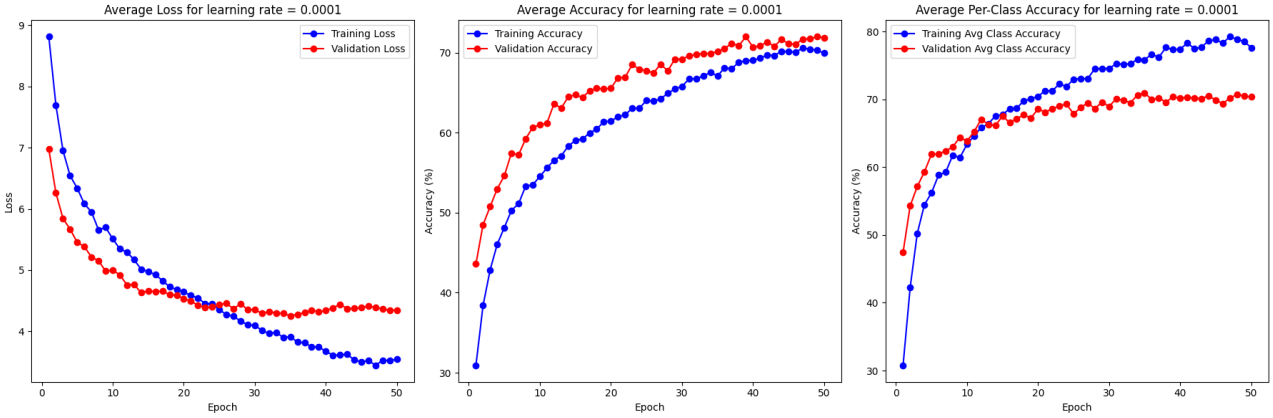


Figure 3: Losses and Accuracies Visualization

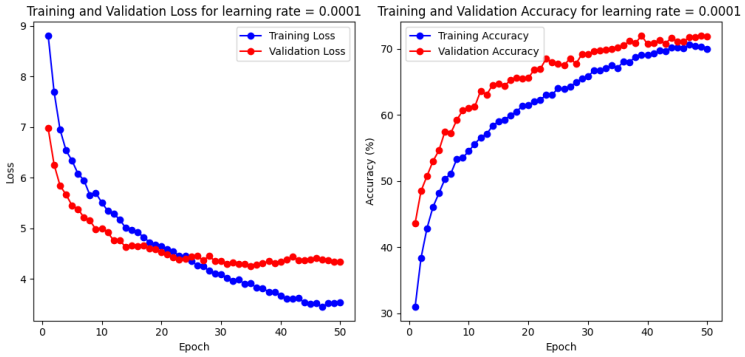


Figure 4: Losses VS Accuracies Visualization

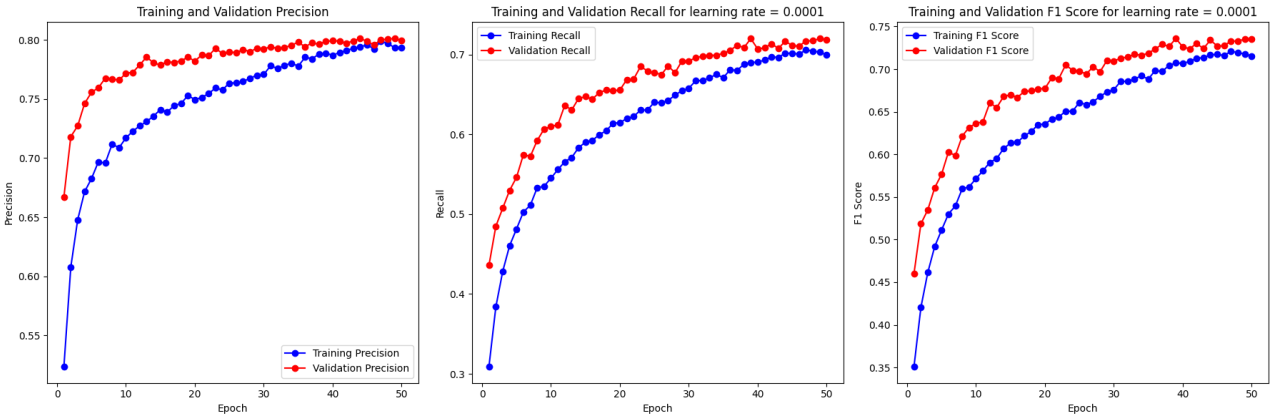


Figure 5: Precision, Recall, and F1-Score Visualization

Overall, these plots' metrics and matching curves offer a multifaceted picture of the model's performance, demonstrating steady progress and strong generalization from the training to the validation set.

1.7 Testing Results

Testing Results are shown below from Codalab after model training and validation. The presented results are attributed to a user identified as ntam0001, who is positioned 2nd in the figure (6).

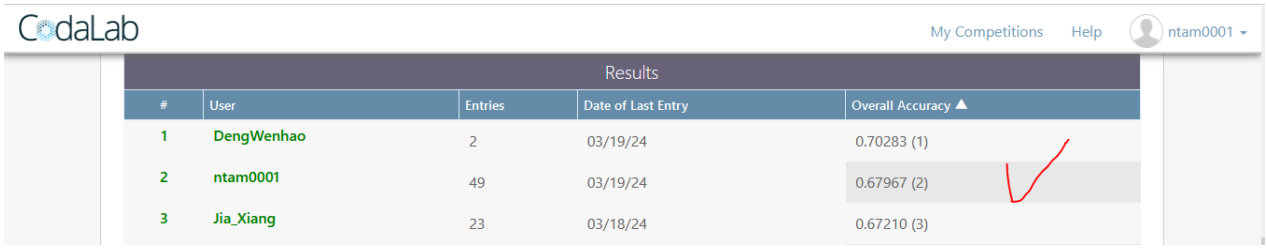


Figure 6: Testing Results over 50 epochs

This pre-trained model was used for an imbalanced dataset called **FashionDataset** with 50 epochs and a learning rate of 0.0001 where I achieved better results as follows in [(table 1)]:

Table 1: Training, Validation, and Testing Results at Epoch 50

Metric	Train	Validation	Testing
Learning Rate	0.0001	-	-
Loss	3.5420	4.3427	-
Accuracy (%)	69.98	71.92	67.967 = (0.67967)
Average Class Accuracy (%)	77.65	70.35	-

After training the model I have achieved the best testing results to present as seen in table 1 and figure 6.

1.8 Used Hyper-parameters to achieve above results

Parameter	Value
Number of Attributes (NUM_ATTR)	*6*
Number of Classes (num_classes)	[7, 3, 3, 4, 6, 3]
Learning Rate (lr)	0.0001
Batch Size (batch_size)	32
Weight Decay (weight_decay)	1×10^{-4}
Device	GPU (”cuda”) or CPU (”cpu”)
Epochs (num_epochs)	50
Dropout Rate	0.5
Neurons in Hidden Layer	512
Patience for LR Scheduler	5
Factor for LR Reduction	0.1 (10%)
Noise Proportion	0.1 (10%)
Transformations	Random resizing, cropping, flipping, rotation, color jitter
Normalization	mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
Weight Calculation	Inversely proportional to class frequencies
Simulated Mean Loss Calculation	For clean and noisy samples illustration
Momentum	0.9
Optimizer	SGD

Table 2: Hyper-parameters and settings used for the model training and evaluation.

2 Conclusion

In the competition, I showcased the application of the FashionNet model, a sophisticated adaptation of ResNet50, for multi-label fashion attribute classification, achieving significant accuracy and class accuracy through architectural modifications, data preparation, and class imbalance management. The results underscore the model's wide applicability and the efficacy of chosen methodologies. Yet, the project identifies further enhancement potential by exploring models like ResNet18, VGG16, and EfficientNet, suggesting that these advances could exceed current achievements. This endeavour not only promises to revolutionize automated fashion categorization but also contributes valuable insights to the field of computer vision, paving the way for future research.

Acknowledgment

Credits to ChatGPT, for helping in understanding and implementing this project.

References

[1] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018. arXiv preprint arXiv:1710.05381.

[2] Nitish Kundu. *Exploring ResNet50: An In-Depth Look at the Model Architecture and Code Implementation*. 2023. Accessed: 2024-03-15.

[3] Huafeng Liu, Mengmeng Sheng, Zeren Sun, Yazhou Yao, Xian-Sheng Hua, and Heng-Tao Shen. Learning with imbalanced noisy data by preventing bias in sample selection. *arXiv preprint arXiv:2402.11242*, Feb 2024. Accepted by IEEE Transactions on Multimedia.