AI6126: REPORT For Project 1

Names: Joseph UWINEZA
Matric No: G2303477F
Program: MSAI

NTU, March 2024

Abstract

In this report I am going to make best use of my knowledge, to handle this mini challenge of accurately determining the specific attribute labels present in fashion photography. The dataset contains **6000** images, which **5000** are allocated as for training purposes and **1000** for validation. This dataset incorporates **26** distinct attribute labels that are typically used to describe clothing, organized into six main categories. Each image in the dataset is marked with six attributes, representing one from each of these categories. Our task is to successfully identify the attributes associated with each given image, thus framing this challenge as a multi-label classification problem.

1 Introduction

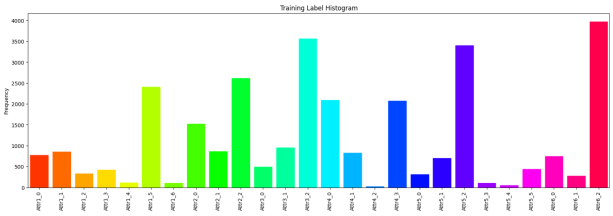
In order to move forward, we will adopt a methodical approach that starts with careful data preprocessing to guarantee ideal model input *(section 2)* and followed with the implementation of a Convolutional Neural Network designed for multi-label classification *(section 3)*. *In section 4*, we will use a Binary Cross-Entropy loss function to measure individual label accuracy. *In section 5*, we will use precise multi-label metrics to assess performance. *In section 6*, we will fine-tune the model through extensive hyperparameter tuning. Finally, *in section 7*, we will provide a nuanced interpretation of the findings to comprehend the model's effectiveness in identifying fashion attributes and the GPU specification I have used.

2 Data Preprocessing

In the data preprocessing phase, we readied the FashionNetDataset, consisting of **6,000** images split into **5,000** for training and **1,000** for validation. Image paths and labels were retrieved from a structured directory, with labels applied only to the training and validation sets. The training images underwent a transformation sequence including cropping, flipping, rotating, and color adjustments for data augmentation, alongside standardization via tensor conversion and normalization. The validation and test sets were resized and cropped for input uniformity but not augmented.

As manifested in figure 1a there is the issue of class imbalance, to address this, we calculated class weights for each attribute based on their frequency in the training set. These weights will later

be instrumental in training the model to ensure equitable learning across all classes. By leveraging DataLoader instances with shuffling for the training set and sequential retrieval for validation and testing, our data preparation phase set a solid foundation for the subsequent model training and evaluation stages.



(a) Shows the histogram of 26 classes just before balance the class.

(b) Shows the histogram of 26 classes just After balance the class.

Figure 1: Graph 1a This image presents a histogram depicting the frequency distribution of labels across **26** classes within a training dataset. Each bar is colored differently, indicating the number of samples for each class attribute in the dataset. The histogram illustrates significant class imbalance, with some classes having many more samples than others, as evident from the varying bar heights)

While graph 1b showing the weighted frequency of **26** classes in a training dataset. Each class is represented by a bar with a distinct color, suggesting the application of weights to balance the class distribution. The bars' varying heights now reflect the adjusted class frequencies after applying the weights, aiming to mitigate class imbalance.

3 Model

I have defined FashionNet, a neural network class that extends PyTorch's nn.Module. It leverages a pre-trained **ResNet-18** model, stripping its final fully connected layer and replacing it with a series of new fully connected layers tailored to the specific number of classes for each attribute in the dataset. Each layer is composed of batch normalization and dropout.

During the forward pass, the model processes input through the ResNet base, and then through each attribute-specific layer, yielding a separate output for each attribute, suitable for multi-label classification tasks. This architecture allows for simultaneous predictions across multiple fashion attributes, utilizing the power of transfer learning for improved performance.

After training the best model gives us **11,195,994** number of parameters

4 Loss Function

I have adopted Cross-Entropy Loss, which can be implemented as nn.CrossEntropyLoss. This function combines nn.LogSoftmax and nn.NLLLoss (negative log-likelihood loss) in one single class.

This loss function is used individually for each attribute's prediction by the model[2]. Since the problem is a multi-label classification (with each label considered as a separate classification task),

for each attribute, the loss is computed between the model's output and the true labels. The loss for each attribute is then weighted by class-specific weights to address class imbalance. This means that the classes with fewer samples have a higher weight, thus contributing more to the loss. This helps the model to pay more attention to classes that are underrepresented in the dataset. I have also tried the SeeSaw loss function [1] but it performs poorly.

The total loss for a batch is the sum of the Cross-Entropy Losses for all attributes. During training, this loss is used to perform backpropagation, while during validation, it is used solely to evaluate the model's performance without updating the weights. The running loss is accumulated over all batches, and its average is used to represent the epoch loss for both training and validation phases.

5 Evaluation Metric

Our multi-attribute classification model uses key metrics for performance evaluation: **Average Class Accuracy**, which gauges individual class accuracy per attribute and assigns zero to classes without samples; **Epoch Loss**, the mean loss for an epoch, aggregated from the CrossEntropyLoss across all attributes and batches; and **Epoch Accuracy**, the aggregate accuracy across all attributes, reflecting the model's comprehensive predictive performance.

These metrics offer a comprehensive evaluation by considering both overall and per-class performance, crucial for datasets with multiple attributes and potential class imbalances.

6 Hyperparameter Tuning

The Table 1, shows hyperparameters which play crucial roles in the behavior and performance of the training process for the neural network. They are typically chosen through empirical testing and adjusted for optimal performance on the validation set.

Table 1: Hyperparameters Table.

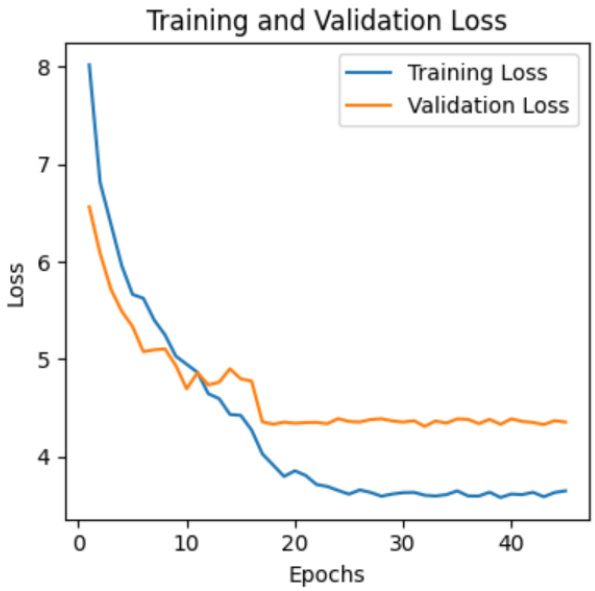
No	Hyperparameter	Values
1	Batch Size	4
2	Weight Decay	0.001
3	Optimizer	Adam
4	Learning rate schedular	0.1
5	Patience	5
6	learning rate	0.0001
7	Droup-out	0.5
8	Number of epochs	45

Table 2: Training and Validation Loss and Accuracy.

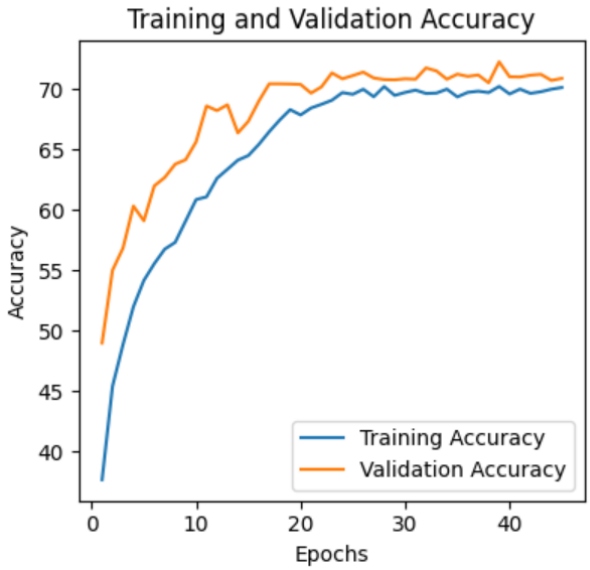
	Accuracy	Loss
Training	70.15%	3.6468
Validation	70.90%	4.3535
Validation Average class	67.96	-
Accuracy on CodaLab	**67.15%**	

7 Results Interpretation

As shown from table 2 the training loss **3.6468** is greater than validation loss **4.3535** which shows that the model learned well and fitt the values.



(a) Shows Training and Validation **Loss** over 45 epochs.



(b) Shows Training and Validation **Accuracy** over 45 epochs

Figure 2: As shown from graph 2a both losses decrease sharply in the initial epochs, indicating that the model is learning. The training loss continues to decline gradually and flattens out as epochs increase, suggesting that the model fits the training data well. The validation loss decreases alongside the training loss initially but starts to plateau much earlier and remains relatively flat, indicating the model's generalization to unseen data.

Graph 2b shows that the training accuracy increases sharply at first and then more gradually, reaching a plateau. The validation accuracy also increases quickly early on but then has a slower rate of improvement, showing some signs of flattening out but not as distinctly as the training accuracy. Both accuracies appear to converge as the number of epochs increases.

The model's performance improves with training, but there seems to be a point where additional training does not lead to significant gains in validation accuracy or decreases in validation loss, indicating a possible point of diminishing returns. This can be a sign that the model is well-fitted to the current dataset, and additional epochs might not lead to better generalization on validation data. There's no clear sign of overfitting since the validation loss does not increase and the validation accuracy does not decrease, which would typically indicate that the model is beginning to memorize the training data.

Results

#	User	Entries	Date of Last Entry	Overall Accuracy ▲
1	DengWenhao	26	03/23/24	0.71881 (1)
2	overfit	78	03/22/24	0.70989 (2)
3	Zhao_Peizhu	24	03/22/24	0.70201 (3)
4	zang18	56	03/23/24	0.69177 (4)
5	lchan025	75	03/23/24	0.68998 (5)
6	Huang_Yupeng	2	03/21/24	0.68235 (6)
7	hs210a	39	03/23/24	0.67988 (7)
8	ntam0001	51	03/19/24	0.67967 (8)
9	Jia_Xiang	27	03/18/24	0.67210 (9)
10	UZAJoseph	60	03/23/24	0.67155 (10)

Figure 3: Shows the resulted accuracy I have obtained, under the username of **UZAJoseph**

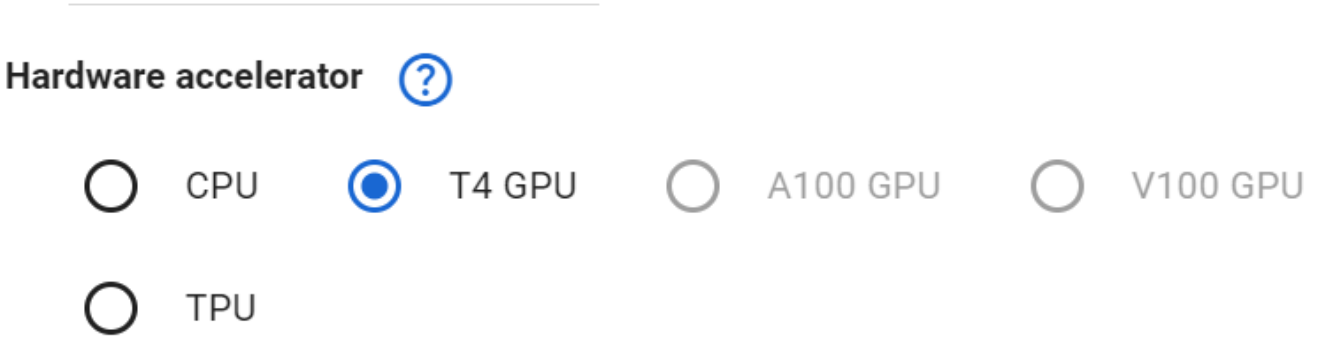


Figure 4: Shows the type of GPU I have used in this challenge

```
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   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 T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   56C    P8              13W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
```

Figure 5: Shows the specifications of GPU

References

[1] Jang et al. Seesaw loss for long-tailed instance segmentation, 2020. Accessed: 2024-03-14.

[2] Takumi Kobayashi. Two-way multi-label loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page [3/10], 2023. Accessed: 2024-03-13.