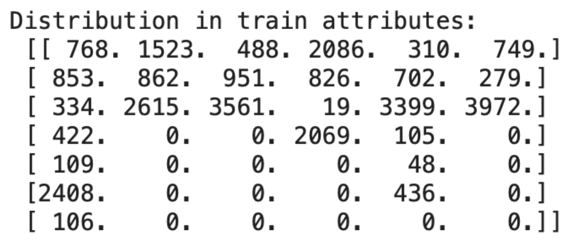
Name: Sean Goh Ann Ray
Matric No.: G2202190G

Preprocessing of data:

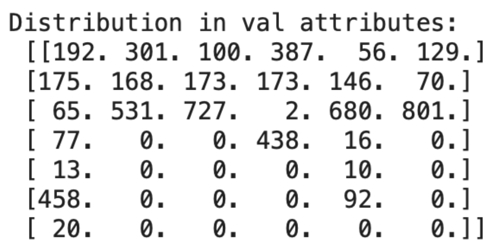
The pre-training of the model was done with images resized to 320x320 pixels using bicubic interpolation, center crop of 300x300 pixels, then normalized using channel means and standard deviations of [0.485, 0.456, 0.406] and [0.229, 0.224, 0.225] respectively. The mean and standard deviation values are that of the ImageNet-1k dataset. As such, the same process of image preprocessing was performed on the FashionDataset provided. Additionally, random horizontal flipping with p=0.5 was performed on the training images.

The train/val/test images were split based on the “train.txt”, “val.txt”, and “test.txt” provided, which included the filenames of the images. The train and val images were then paired with their respective attributes according to the “train_attr.txt” and “val_attr.txt” files. With the image – attributes pairs, the dataset thus becomes a multiclass multilabel dataset, with the 6 different labels having different number of classes, being [7, 3, 3, 4, 6, 3] respectively. Figure 1 and 2 shows the distribution of each class in each of the 6 label categories for the 5000 train data pairs and 1000 val data pairs respectively.



```
Distribution in train attributes:
 [[ 768. 1523.  488. 2086.  310.  749.]
 [ 853.  862.  951.  826.  702.  279.]
 [ 334. 2615. 3561.   19. 3399. 3972.]
 [ 422.    0.    0. 2069.  105.    0.]
 [ 109.    0.    0.    0.   48.    0.]
 [2408.    0.    0.    0.  436.    0.]
 [ 106.    0.    0.    0.    0.    0.]]
```

Figure 1. Distribution of Classes in Train Data



```
Distribution in val attributes:
 [[192. 301. 100. 387.  56. 129.]
 [175. 168. 173. 173. 146.  70.]
 [ 65. 531. 727.   2. 680. 801.]
 [ 77.   0.   0. 438.  16.   0.]
 [ 13.   0.   0.   0.  10.   0.]
 [458.   0.   0.   0.  92.   0.]
 [ 20.   0.   0.   0.   0.   0.]]
```

Figure 2. Distribution of Classes in Val Data

PyTorch DataLoader was used as the train/val/test dataset iterator, with the shuffle setting to True for the train data, and False for the val and test data. Batch sizes of 16, 32, and 64 were explored and a batch size of 32 produced the best results. The num_workers parameter was set to 4 to utilize the GPU’s parallel computing capability for faster computations.

GPU:

Google Colab was used as the platform for the algorithm testing, with the Google Colab Tesla T4 GPU being utilized.

Model:

Several experiments were conducted with different model, but only at their basic settings, and they are ResNet50, EfficientNet_B3, EfficientNet_V2_s. EfficientNet_B3 was selected as it provided a good balance of validation accuracy and training time.

The model used was the EfficientNet_B3 model from PyTorch, as well as the use of pretrained weights that was obtained from training the model on the ImageNet-1k dataset. Modifications were made (following Droste, 2022) to the final layers of the model such that there were 6 fully connected layers, one for each of the label, followed by a softmax layer. Layers with less than the highest number of classes (7) had their outputs concatenated with zeros, such that with an input of [batch size, C, H, W] to the model, all layers will produce a matrix of shape [batch size, 7]. The output from each layer were then concatenated together to produce a matrix of shape [batch size, 7, 6]. Figure 3 shows the architecture of the original EfficientNet_B3 used for the ImageNet-1k dataset on the left and shows the architecture of the modified model used for this FashionDataset on the right.

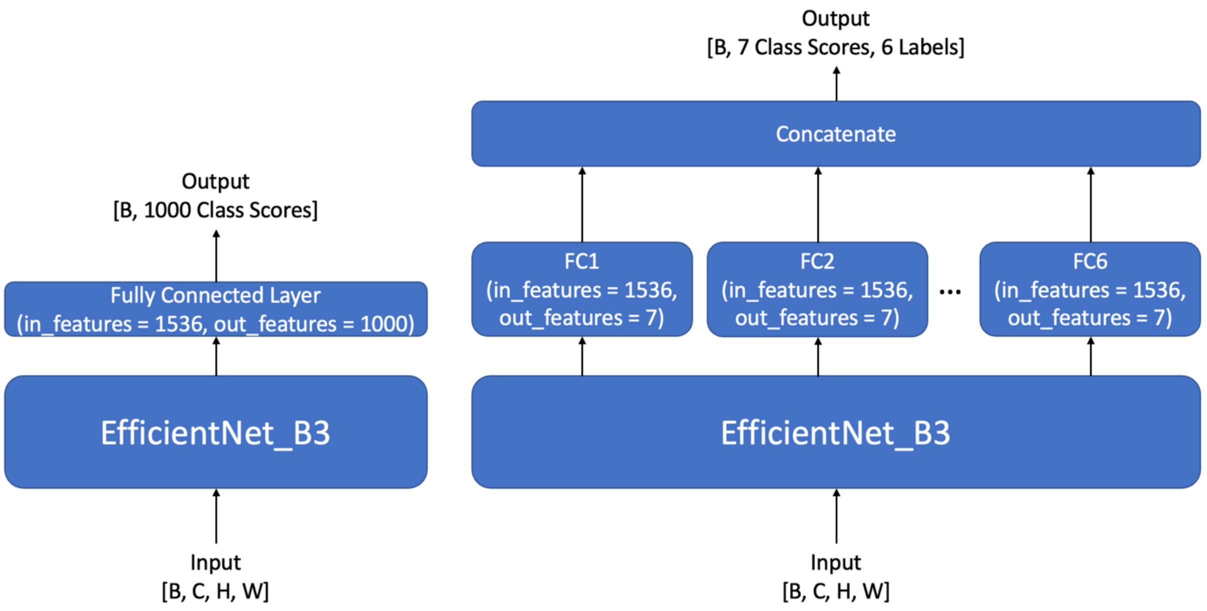

Output
[B, 1000 Class Scores]
Fully Connected Layer
(in_features = 1536, out_features = 1000)
EfficientNet_B3
Input
[B, C, H, W]
Output
[B, 7 Class Scores, 6 Labels]
Concatenate
FC1
(in_features = 1536, out_features = 7)
FC2
(in_features = 1536, out_features = 7)
...
FC6
(in_features = 1536, out_features = 7)
EfficientNet_B3
Input
[B, C, H, W]

Figure 3. Architecture of Models

To improve generalization, a dropout parameter was added to each of the 6 fully connected layers, and after several experiments, a dropout with p=0.2 was found to produce the most optimal val loss.

The model was trained with 10, 20, 30, and 50 epochs, and the best results were obtained with 50 epochs, improving the val accuracy by 1% from that of 30 epochs.

Number of parameters of model:

Using the code provided, the modified model has a total of 12,273,194 parameters.

Loss functions:

Stochastic Gradient Descent (SGD) and the Adam optimizers were explored, but the optimizer which provided the best results after several experiments was the SGD optimizer with the following parameters:

Learning Rate = 0.1
Learning Rate Scheduler = Cosine Annealing
Momentum = 0.9
Weight Decay = 0.00001

The loss function used was the Cross Entropy Loss, and the accuracy was calculated as the number of correctly predicted class labels divided by the total number of label predictions made (5000 x 6 for train data and 1000 x 6 for val data).

Training curves:

The training curves of the model training which produced the best val loss and accuracy are shown in Figures 4 and 5 below.

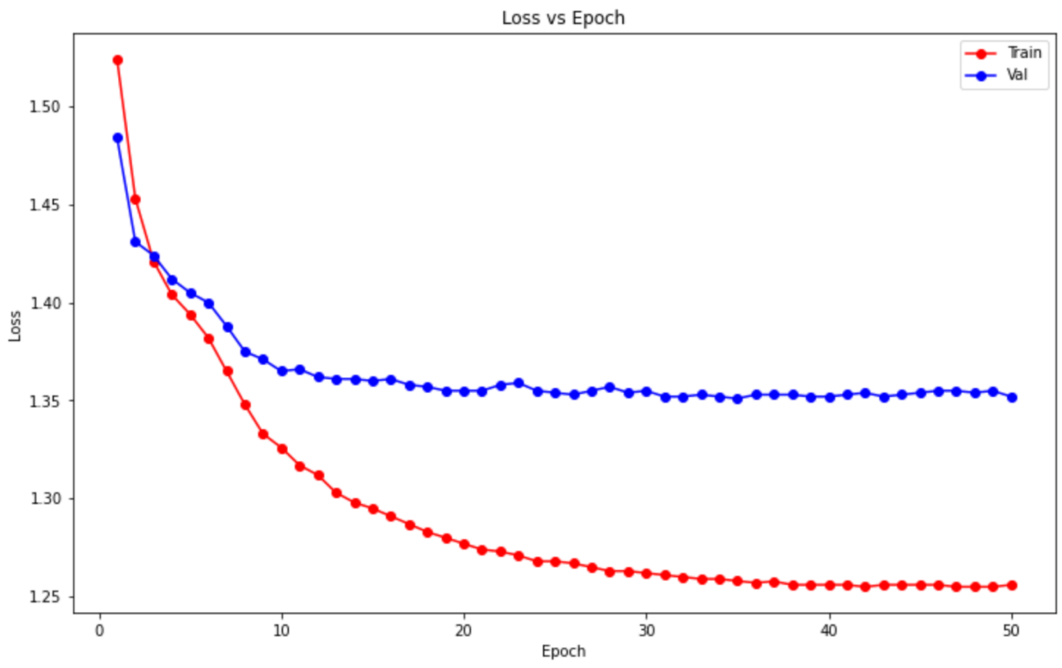


Figure 4. Loss vs Epoch of Model Training

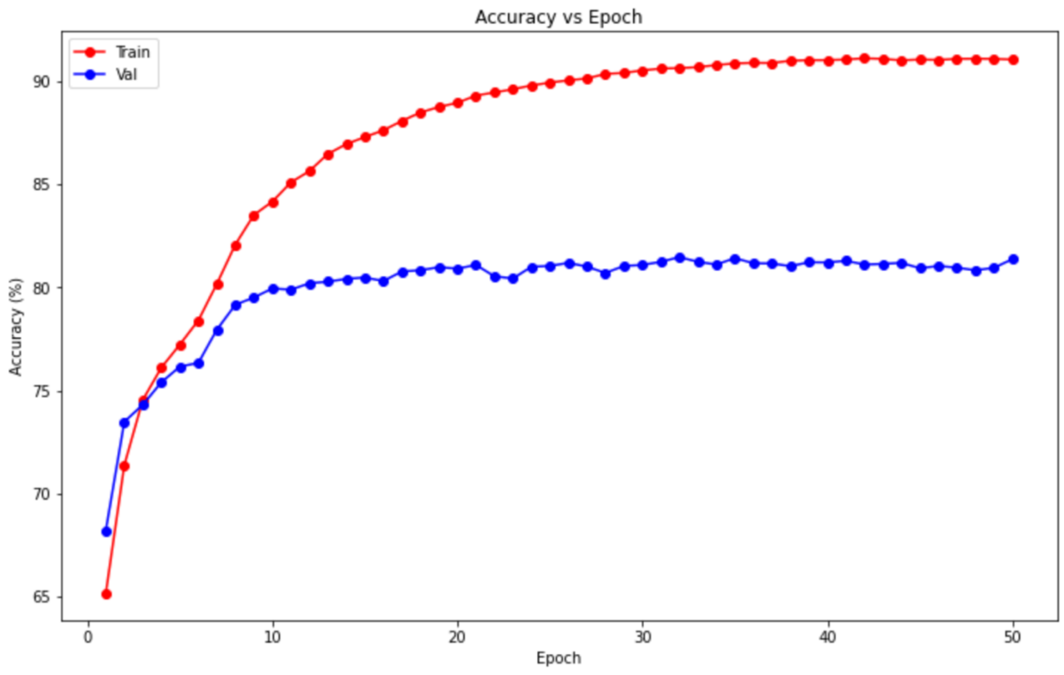
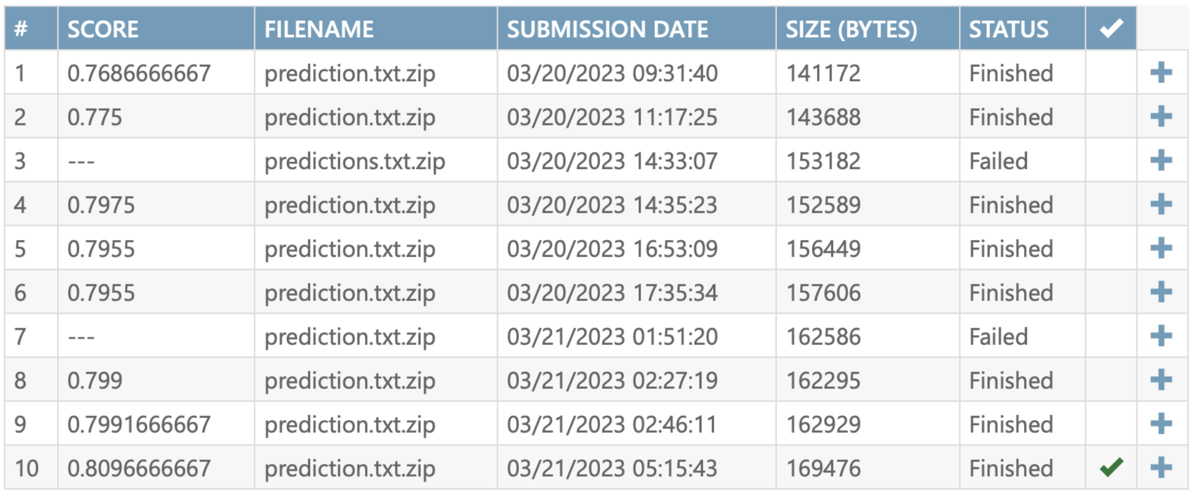


Figure 5. Accuracy vs Epoch of Model Training

The best train and val loss achieved were 1.255 and 1.351 respectively, and the best train and val accuracy achieved were 91.12% and 81.48% respectively. The model checkpoint saved was the one with the best val loss, which was then used to make the prediction on the test data before the submission on CodaLab.

The best model was the 10th submission on CodaLab as shown in Figure 6, which achieves 80.97% on the test dataset.



#	SCORE	FILENAME	SUBMISSION DATE	SIZE (BYTES)	STATUS	✓	
1	0.7686666667	prediction.txt.zip	03/20/2023 09:31:40	141172	Finished		+
2	0.775	prediction.txt.zip	03/20/2023 11:17:25	143688	Finished		+
3	---	predictions.txt.zip	03/20/2023 14:33:07	153182	Failed		+
4	0.7975	prediction.txt.zip	03/20/2023 14:35:23	152589	Finished		+
5	0.7955	prediction.txt.zip	03/20/2023 16:53:09	156449	Finished		+
6	0.7955	prediction.txt.zip	03/20/2023 17:35:34	157606	Finished		+
7	---	prediction.txt.zip	03/21/2023 01:51:20	162586	Failed		+
8	0.799	prediction.txt.zip	03/21/2023 02:27:19	162295	Finished		+
9	0.7991666667	prediction.txt.zip	03/21/2023 02:46:11	162929	Finished		+
10	0.8096666667	prediction.txt.zip	03/21/2023 05:15:43	169476	Finished	✓	+

Figure 6. Screenshot of CodaLab Score

References:

Droste, B. (2022) Multilabel classification with Pytorch in 5 minutes, Medium. Towards Data Science. Available at: https://towardsdatascience.com/multilabel-classification-with-pytorch-in-5-minutes-a4fa8993cbc7 (Accessed: March 16, 2023).