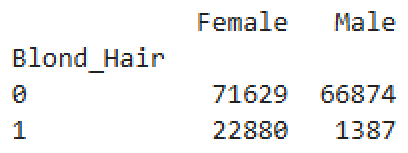
Assignment

General Info

GPU = Kaggle T4 × 2

Batch Size = 64

Full Data Count



```
            Female   Male
Blond_Hair
0            71629  66874
1            22880   1387
```

Final dataset

- required manual intervention

- trained multiple models with different subsets → combine

Training Time

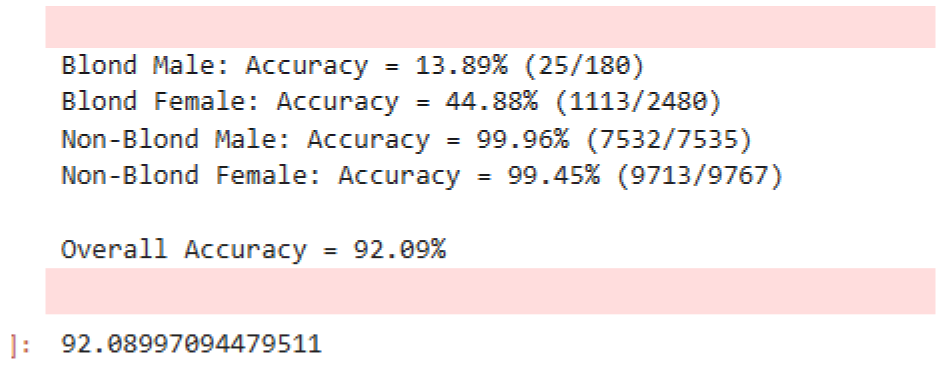
- 25-30 minutes per model

Generation Time

- 7 hours for 5000 images

ResNet Finetuning Time = 5-7 minutes (5 epochs)

Base Model - Actual Data Full



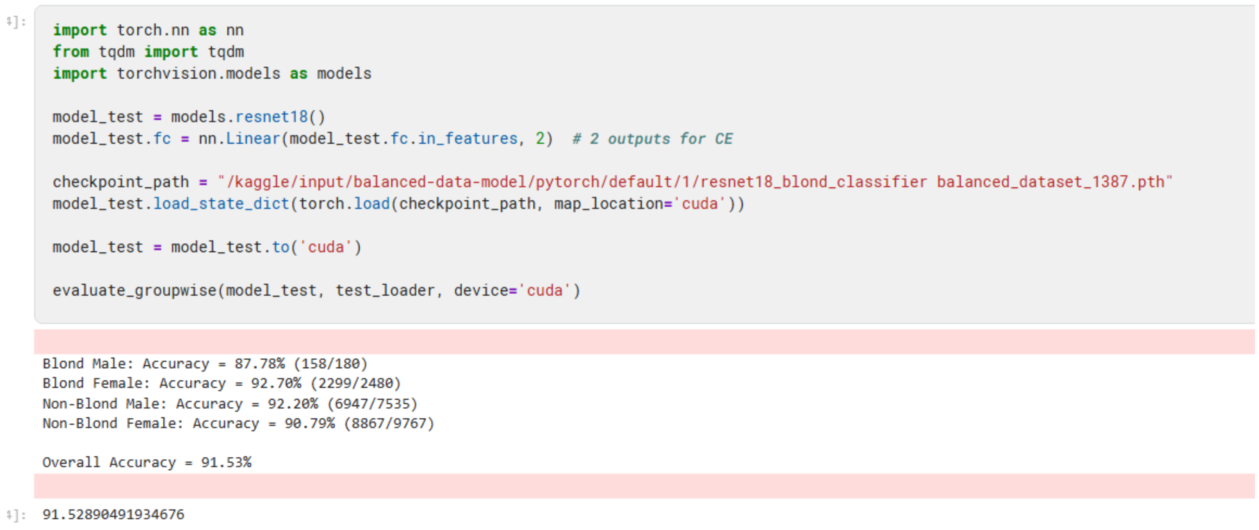
```
Blond Male: Accuracy = 13.89% (25/180)
Blond Female: Accuracy = 44.88% (1113/2480)
Non-Blond Male: Accuracy = 99.96% (7532/7535)
Non-Blond Female: Accuracy = 99.45% (9713/9767)

Overall Accuracy = 92.09%

]:  92.08997094479511
```

- Reasons - highly imbalanced data, very few male blonds, might associate blondness with female, features, so when male blond is shown, it doesnt see female features and so predicts non blond
- model cant generalize
- model **quickly learns to classify majority classes** (like non-blond males/females) correctly. It gets **many correct predictions early**, which reduces the overall loss

Actual Data

> Balanced (1387 per subgroup, other groups undersampled by the upper cap of male blond class)

1st Time

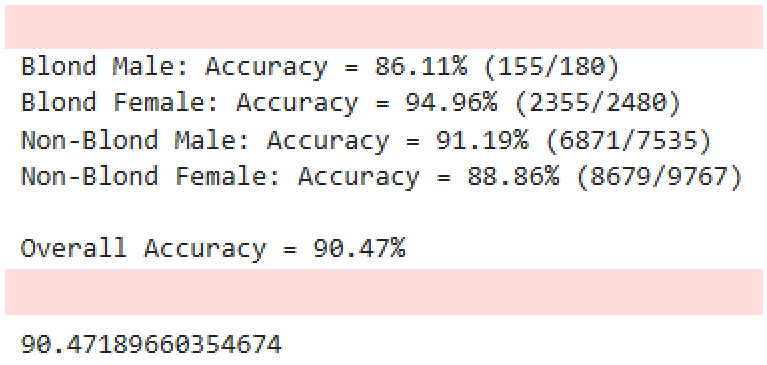


```python
import torch.nn as nn
from tqdm import tqdm
import torchvision.models as models

model_test = models.resnet18()
model_test.fc = nn.Linear(model_test.fc.in_features, 2)  # 2 outputs for CE

checkpoint_path = "/kaggle/input/balanced-data-model/pytorch/default/1/resnet18_blond_classifier balanced_dataset_1387.pth"
model_test.load_state_dict(torch.load(checkpoint_path, map_location='cuda'))

model_test = model_test.to('cuda')

evaluate_groupwise(model_test, test_loader, device='cuda')
```

```
Blond Male: Accuracy = 87.78% (158/180)
Blond Female: Accuracy = 92.70% (2299/2480)
Non-Blond Male: Accuracy = 92.20% (6947/7535)
Non-Blond Female: Accuracy = 90.79% (8867/9767)

Overall Accuracy = 91.53%
```

```
91.52890491934676
```

2nd Time



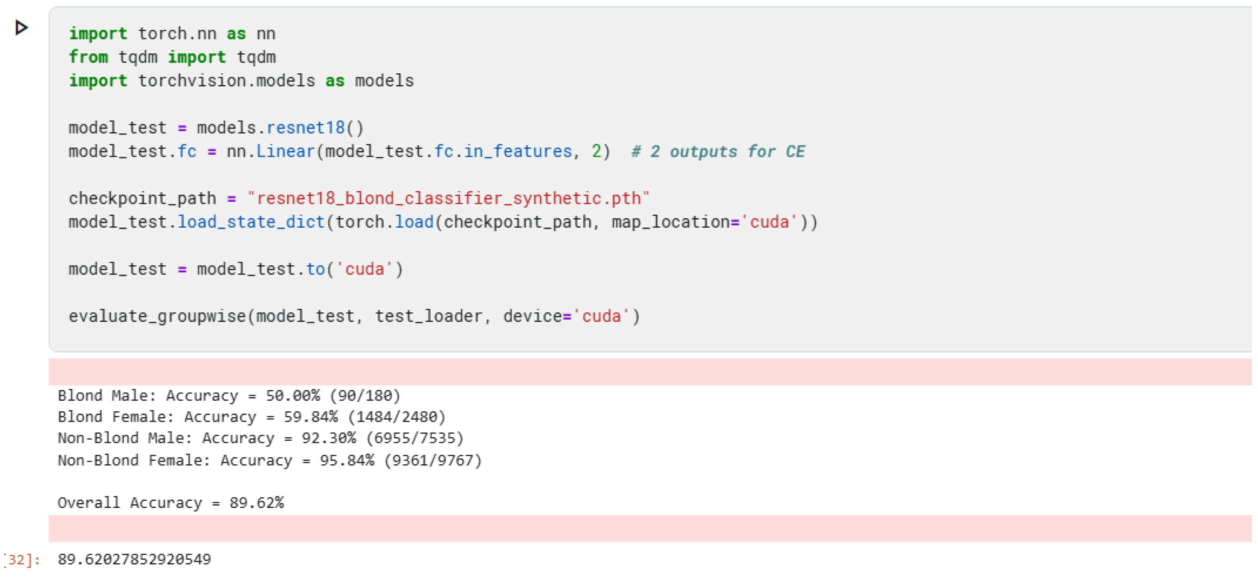
```
Blond Male: Accuracy = 86.11% (155/180)
Blond Female: Accuracy = 94.96% (2355/2480)
Non-Blond Male: Accuracy = 91.19% (6871/7535)
Non-Blond Female: Accuracy = 88.86% (8679/9767)

Overall Accuracy = 90.47%
```

```
90.47189660354674
```

- balancing data improves performance, probably removes spurious correlation between blonds and females, model is forced to learn hair features since it sees varied faces in equal counts now.
- However if balanced sample subset is not varied enough, or very different from test set, model metrics will drop low. Also depends on overlap between classes.

    - Lets say most blond males dont have beard, and non blond males have beard (example taken because there is some tendency in the dataset that many non blonds have beards and blonds have less beards) model might associate hair labels with no beard on face and blond males might end up being misclassified as non blonds since all non blond females have no beard

Synthetic Data - 2782 images per subgroup



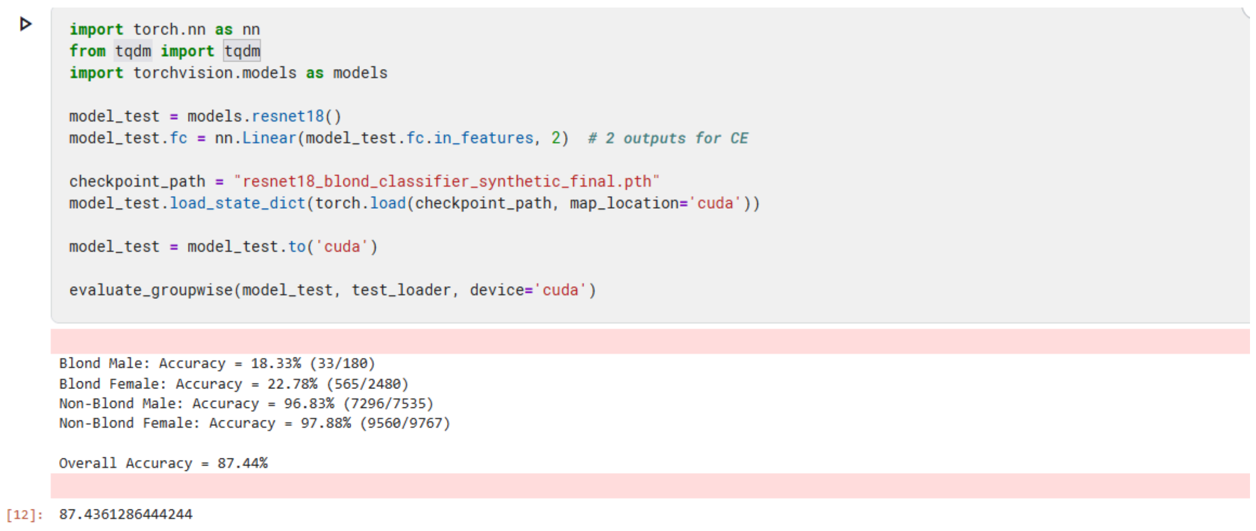
```
import torch.nn as nn
from tqdm import tqdm
import torchvision.models as models

model_test = models.resnet18()
model_test.fc = nn.Linear(model_test.fc.in_features, 2)  # 2 outputs for CE

checkpoint_path = "resnet18_blond_classifier_synthetic.pth"
model_test.load_state_dict(torch.load(checkpoint_path, map_location='cuda'))

model_test = model_test.to('cuda')

evaluate_groupwise(model_test, test_loader, device='cuda')
```

```
Blond Male: Accuracy = 50.00% (90/180)
Blond Female: Accuracy = 59.84% (1484/2480)
Non-Blond Male: Accuracy = 92.30% (6955/7535)
Non-Blond Female: Accuracy = 95.84% (9361/9767)

Overall Accuracy = 89.62%
```

```
[32]: 89.62027852920549
```

- Why this performed better than synthetic full?
    - Better realism and not over dominated by particular traits, also it limited the number of redundant blond males the model saw, so it avoided overfitting on the wrong description of a blond male, so it performed better.

Synthetic Data Full



```python
import torch.nn as nn
from tqdm import tqdm
import torchvision.models as models

model_test = models.resnet18()
model_test.fc = nn.Linear(model_test.fc.in_features, 2)  # 2 outputs for CE

checkpoint_path = "resnet18_blond_classifier_synthetic_final.pth"
model_test.load_state_dict(torch.load(checkpoint_path, map_location='cuda'))

model_test = model_test.to('cuda')

evaluate_groupwise(model_test, test_loader, device='cuda')
```

```
Blond Male: Accuracy = 18.33% (33/180)
Blond Female: Accuracy = 22.78% (565/2480)
Non-Blond Male: Accuracy = 96.83% (7296/7535)
Non-Blond Female: Accuracy = 97.88% (9560/9767)

Overall Accuracy = 87.44%
```

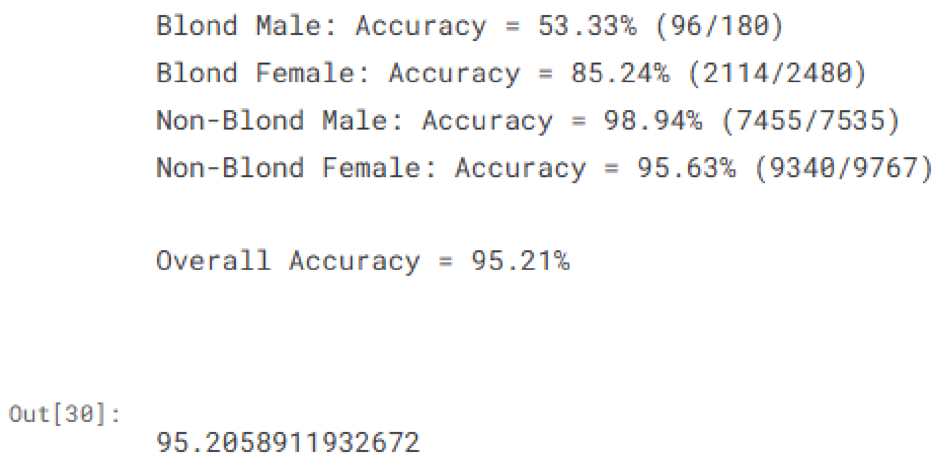
```
[12]: 87.4361286444244
```

- Why I think it didnt perform better than 2782 per subgroup?
  - I had created multiple sub models, for example initially non blonds folder when dealing with 2782 images and brown, brunette, or dark yellow ochre kind hair which may or may not be classified as blond, which is why I thought the accuracy of female blonds was low in the previous experiment since the model must not have been able to clearly distinguish between female blonds and non blonds becuase of dataset overlap
  - So then I trained 2 more sub models, on a different prior and instance set of images, which focused on black, dark brown hair. The first model gave decent images, which were realistic and had good variation in facial features, hairstyles and lighting, but the second model seemed to overfit on one particular face and didnt have much variation. So the non blond group ended up being ~2700 images of varied non blond hair colors, while 2300 had dark black and brown hair. This maybe different from test set. Also the images could be told they were generated
  - Blond Males - trained 3 different models, yet all of them seem to overfit on one particular face. They all had obvious yellow hair. Real images blonds were quite different from synthetic blonds. Lighting was different.

Focal Loss

$$\text{Focal Loss}(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t)$$

- modify $\text{Cross Entropy}(p_t) = -\log(p_t)$
- so if model has higher confidence for a particular label then p_t is close to 1, then weighting term becomes 0, so loss added by that example is 0 so basically easy example ie whose confidence is high will contribute less to loss, eventually contributing less to gradient change, so this way model learns from harder examples
- but if model has learnt spurious correlation that males are usually non blond



```
Blond Male: Accuracy = 53.33% (96/180)
Blond Female: Accuracy = 85.24% (2114/2480)
Non-Blond Male: Accuracy = 98.94% (7455/7535)
Non-Blond Female: Accuracy = 95.63% (9340/9767)

Overall Accuracy = 95.21%


Out[30]:
         95.2058911932672
```

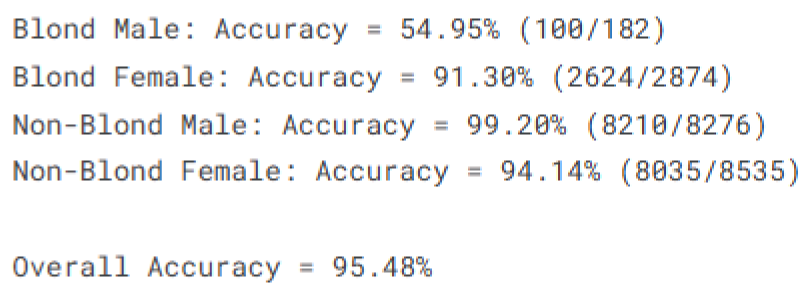
- helped to **focus more on rare, misclassified samples**, such as blond males, model had **more incentive to learn features** that correctly classify minority groups.
- reduced the the spurious "blond = female" or "male" = "non blond".

Class-Balanced Loss

$\text{Effective number: } E_n = \frac{1-\beta}{1-\beta^n}$

- n is the number of samples for a class
- $\beta \in$ [0,1] is a hyperparameter
- Then weight for class c is: $\mathrm{w}_c = \frac{1}{E_n}$

- $\text{CB Loss}(p_t) = -\mathrm{w}_c \log(p_t)$



```
Blond Male: Accuracy = 54.95% (100/182)
Blond Female: Accuracy = 91.30% (2624/2874)
Non-Blond Male: Accuracy = 99.20% (8210/8276)
Non-Blond Female: Accuracy = 94.14% (8035/8535)

Overall Accuracy = 95.48%
```

- **blond males** (with very few samples) get **higher loss weights**, this increases contribution to loss, so gradient change is affected more by minority samples than majority sample, when taking number into account, it balances out the contribution to loss by both majority and minority class
- Prevents **majority classes** from dominating the loss.
- Why it didnt improve drastically → underlying data, model learns better from the male blonds in training data but still cant generalize enough

CB + Focal

- $\text{Focal Loss}(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t)$
- $\text{CB Loss}(p_t) = -\mathrm{w}_c \log(p_t)$
    - $\text{Effective number: } E_n = \frac{1-\beta}{1-\beta^n}$
    - n is the number of samples for a class,
    - β ∈[0,1] is a hyperparameter
    - Then weight for class c is: $\mathrm{w}_c = \frac{1}{E_n}$
- $\text{CBFocal Loss}(p_t) = -\mathrm{w}_c(1-p_t)^\gamma \log(p_t)$
- male blond drop to 13%, probably because model overfit on training samples since we reweighted them (CB loss) and assuming these were the hard

examples (then focal loss also increased their importance), so overall model overfit on training data for male blonds and final output was bad

- If:
    - p_t→0: loss explodes.
    - w_c(CB weights) is large for rare class.

Stage Wise (3 + 2 epochs)

```
Blond Male: Accuracy = 54.44% (98/180)
Blond Female: Accuracy = 89.80% (2227/2480)
Non-Blond Male: Accuracy = 99.02% (7461/7535)
Non-Blond Female: Accuracy = 95.41% (9319/9767)

Overall Accuracy = 95.71%
```

```
Out[36]:
         95.70684300170323
```

Frozen Layers (Only final layer)

```
Blond Male: Accuracy = 44.44% (80/180)
Blond Female: Accuracy = 85.44% (2119/2480)
Non-Blond Male: Accuracy = 99.27% (7480/7535)
Non-Blond Female: Accuracy = 96.41% (9416/9767)

Overall Accuracy = 95.66%
```

```
Out[40]:
         95.65674782085964
```

Dreambooth Issues

- number of steps (800 → 1000)
- learning rate (5e-6 → 2e-6)
- overfitting very quickly on priors
    - initial logic → 100 blonds as instance and 270 non blonds + 30 blonds for prior (output is mostly blond)
    - but when done same for non blonds → in reverse ratio → output is more blond than non blond
        - probable reason being more intraclass variation in non blonds
        - solution → keep priors also as non blonds
        - tried different prompts but that didnt work
        - special token blondmankk/ blondwomankk → a blond man
    - subsets creation method
        - random sampling
        - binary signature unique
        - cherry picking

- model easily overfits on old females, one particular blond man

Links

DreamBooth fine-tuning with LoRA

DreamBooth

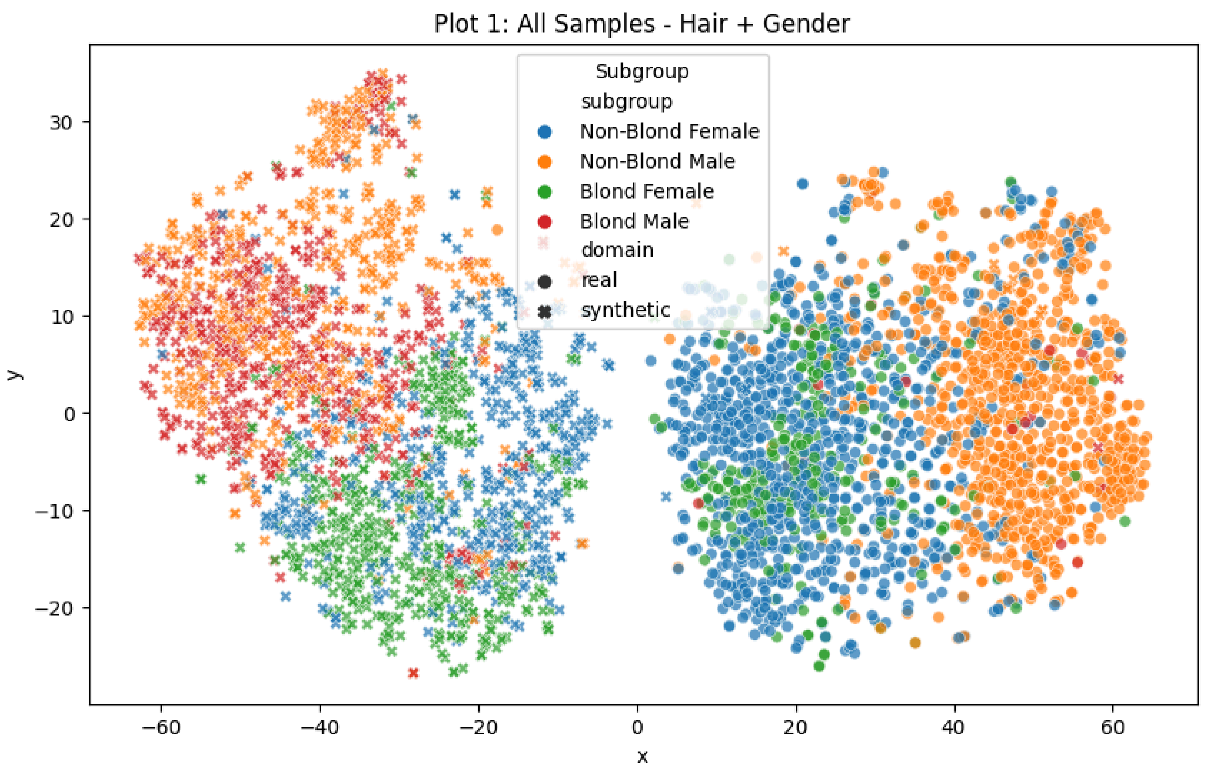
https://huggingface.co/blog/dreambooth

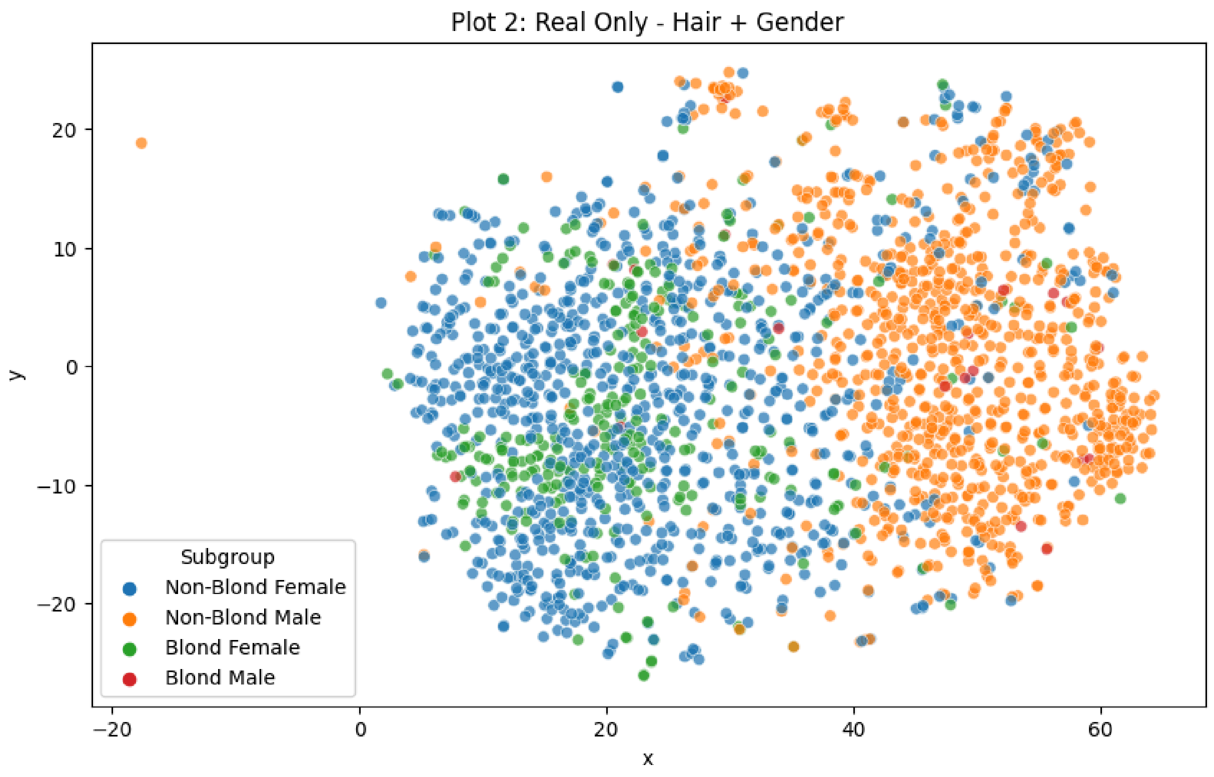
  - this said use ddim when doing inference, it worked ok, not much difference
  - used 30 steps (balanced)
  - did not use lora for training since 30mins training time was acceptable

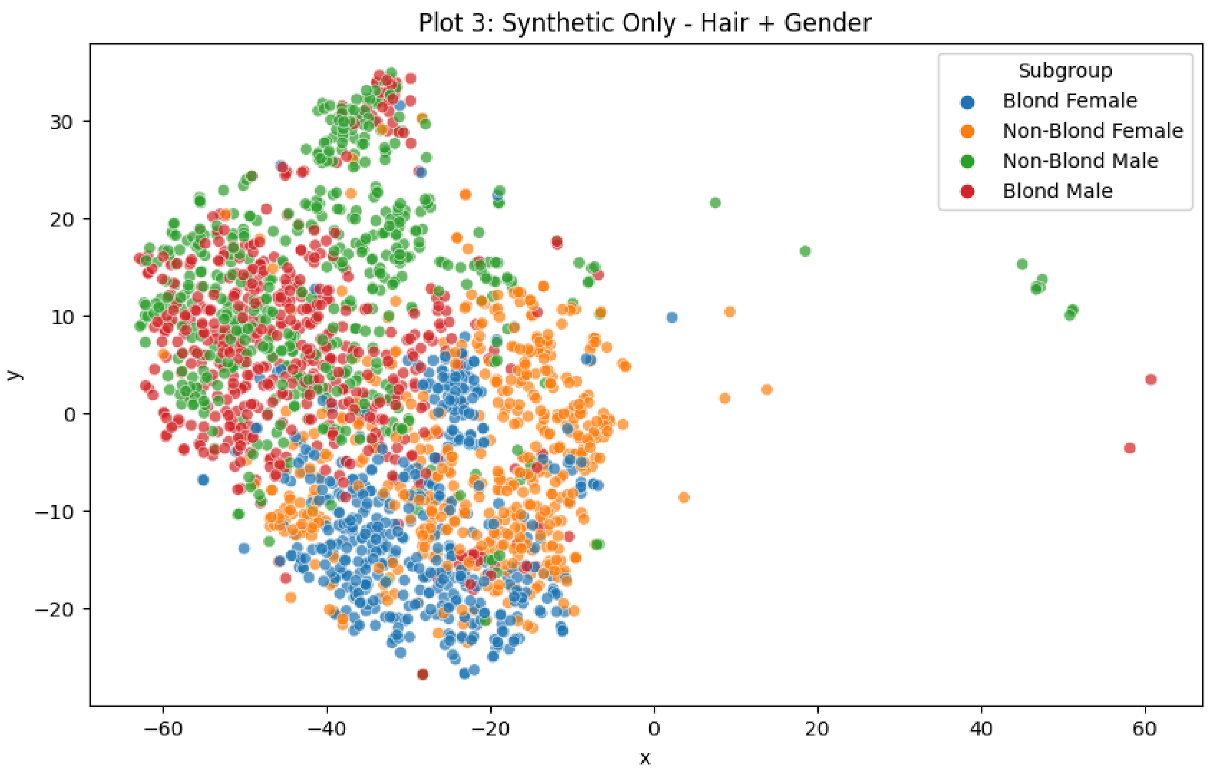
Things I thought of but did not try

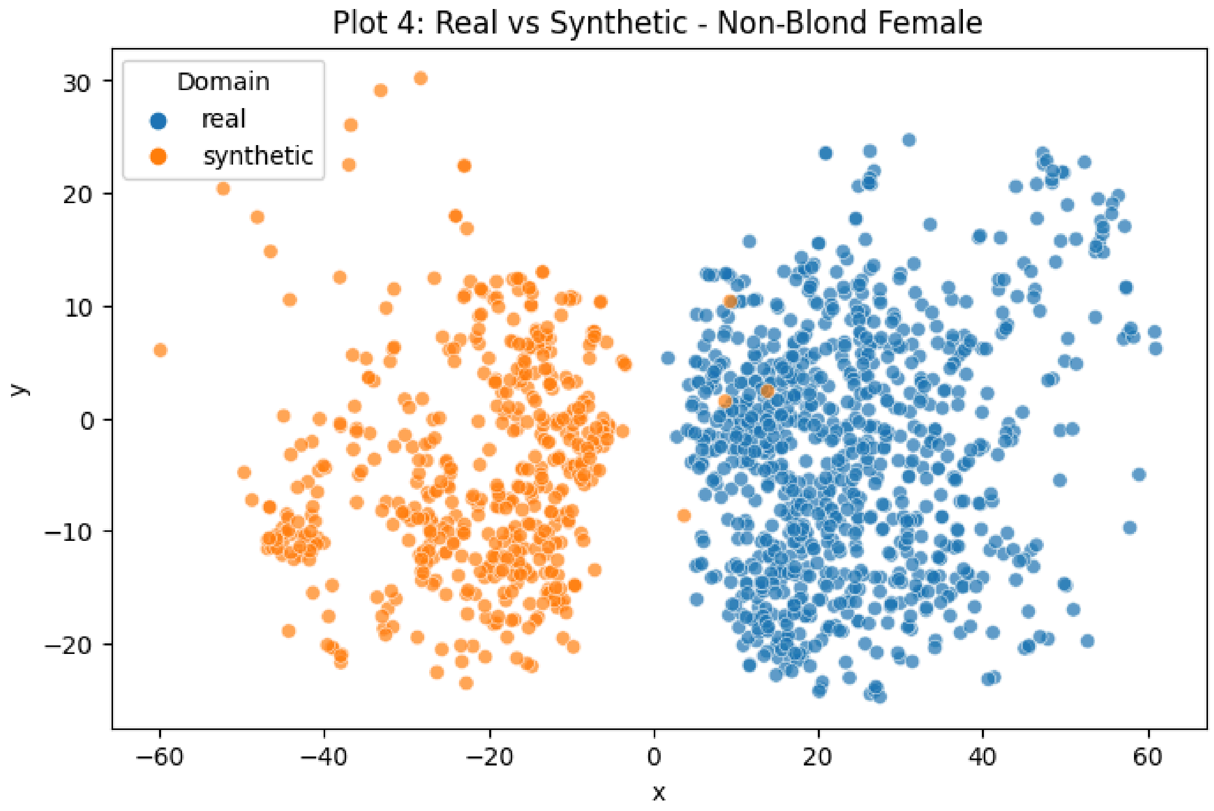
- weighted random sampling
  - Normal sampling ie dataloader with shuffle = true gives batches dominated by the top 2 classes. The rare class (Blond Male) might be seen *only a few times* per epoch.
  - would simply do weighted sampling of blonds with replacement when creating batches, did not do it because I thought model would overfit on training samples
- creating batches in similar distribution to training set
  - was possible if imbalance was not so skewed, did not try because if in a

Domain Drift Visualization

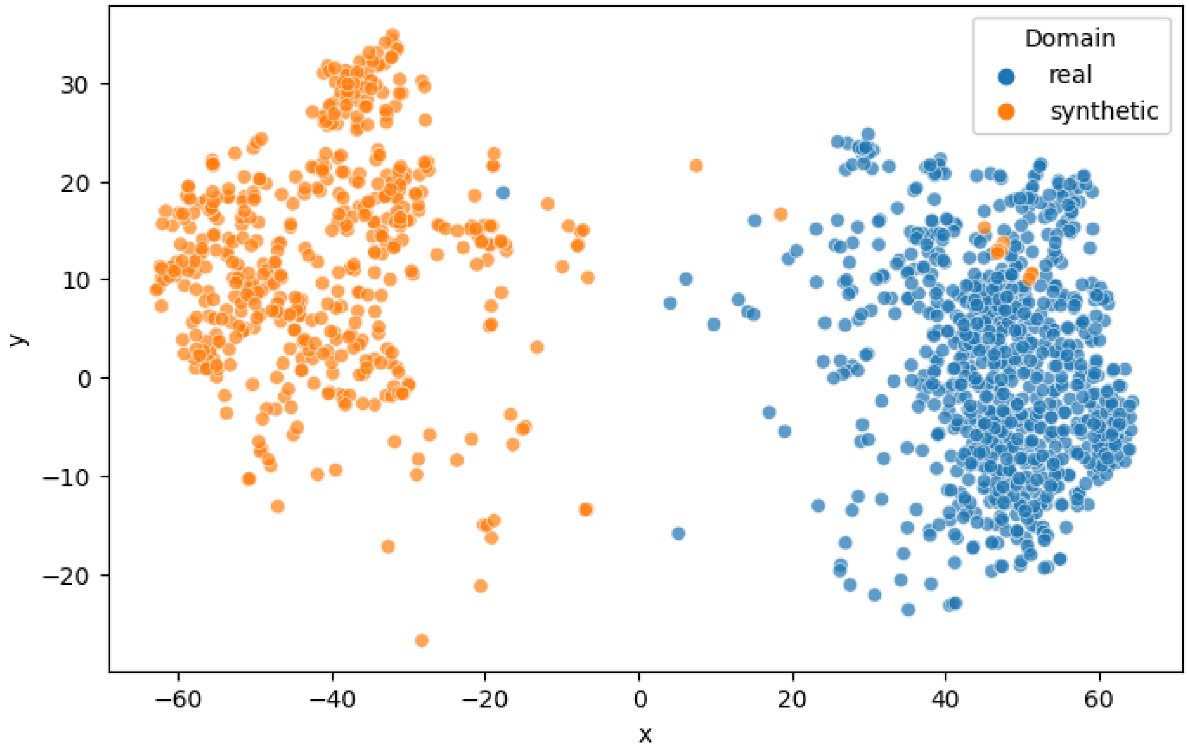

Plot 1: All Samples - Hair + Gender
Subgroup
subgroup
Non-Blond Female
Non-Blond Male
Blond Female
Blond Male
domain
real
synthetic
y
x
30
20
10
0
−10
−20
−60
−40
−20
0
20
40
60


Plot 2: Real Only - Hair + Gender
Subgroup
Non-Blond Female
Non-Blond Male
Blond Female
Blond Male
y
x
20
10
0
−10
−20
−20
0
20
40
60

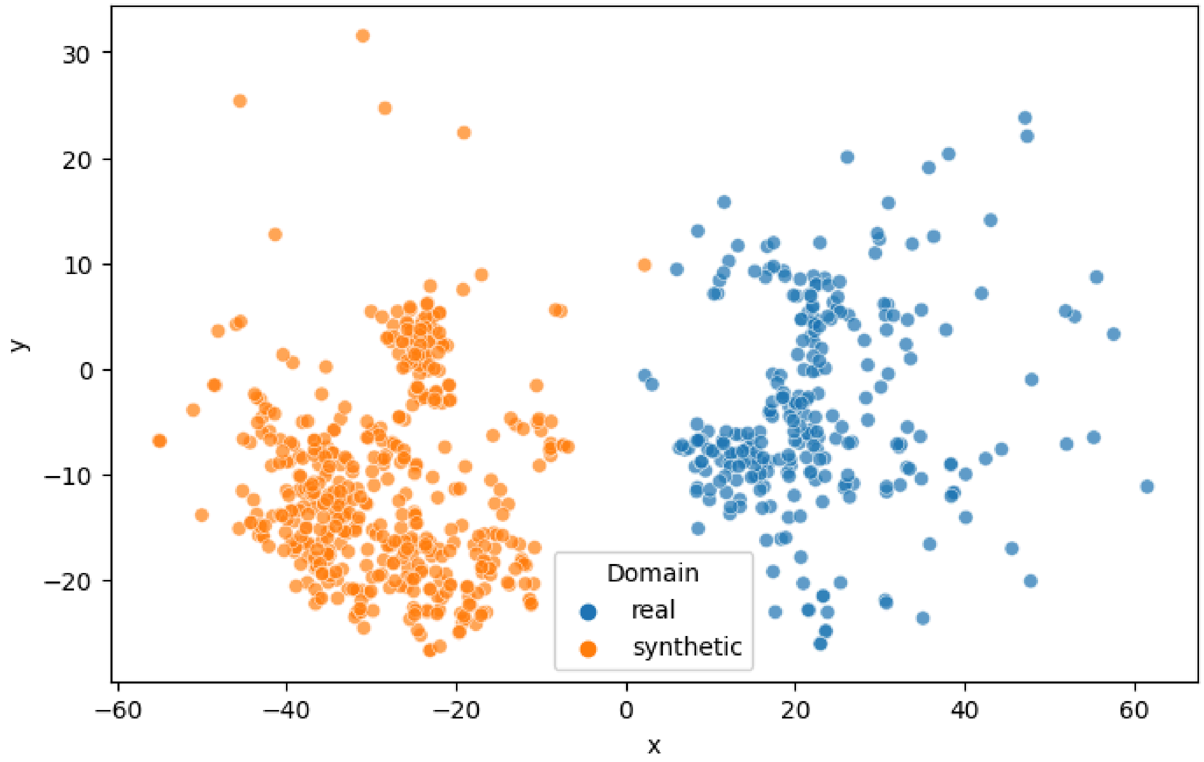

Plot 3: Synthetic Only - Hair + Gender
Subgroup
Blond Female
Non-Blond Female
Non-Blond Male
Blond Male
x
y

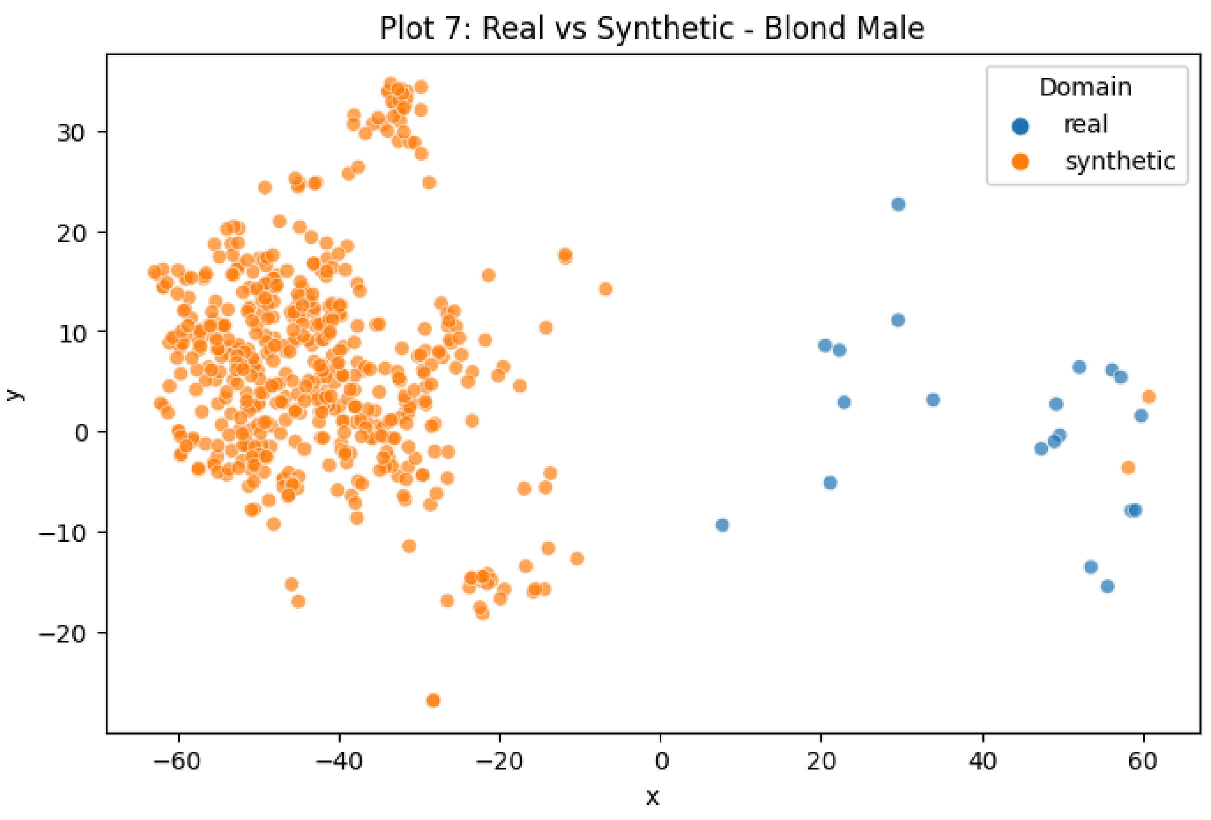

Plot 4: Real vs Synthetic - Non-Blond Female
Domain
real
synthetic
x
y

Plot 5: Real vs Synthetic - Non-Blond Male



Plot 6: Real vs Synthetic - Blond Female





Overall observation

Overall performance drop

Accuracy dropped from **~95% to 89%** after switching from real to synthetic training data.

> Synthetic data distribution is quite different from real. It could also be visibly seen, mainly in male blond images which had very narrow variation

Effect on subgroups

- These subgroups are **minorities in the real test set**:

  - Blond Male: **180 samples**
  - Blond Female: **2,480 samples**

When we created 5k synthetic images from these, we created more from loss, the dreambooth model overfit on the 100 instances we provided and we are now using that distribution as the basis for training, so obviously it does not generalize well (basically the softmax probabilities when predicting male non blonds is high so model defaults to non blond to improve overall metrics)

- Visually non blonds and female blonds were generated better than male blonds

2782 > 5000

- indirectly helped remove bias created when we generated more synthetic from less diverse data
- help prevent overfitting to synthetic training data so generalized better on real data

Domain Adaptation

- Hacked together a basic DANN from https://github.com/tadeephuy/GradientReversal/ https://github.com/fungtion/DANN and GPT

Since reference real data distribution was test split, I evaluated on both validation and test set

Validation Set

```
[14]: evaluate_groupwise(model, val_loader, device='cuda')
```

```
Blond Male: Accuracy = 89.56% (163/182)
Blond Female: Accuracy = 84.31% (2423/2874)
Non-Blond Male: Accuracy = 51.22% (4239/8276)
Non-Blond Female: Accuracy = 82.78% (7065/8535)

Overall Accuracy = 69.91%
```

```
[14]: 69.91493431318267
```

Test Set

```
: evaluate_groupwise(model, test_loader, device='cuda')
```

```
Blond Male: Accuracy = 91.67% (165/180)
Blond Female: Accuracy = 83.27% (2065/2480)
Non-Blond Male: Accuracy = 51.61% (3889/7535)
Non-Blond Female: Accuracy = 85.06% (8308/9767)

Overall Accuracy = 72.27%
```

```
: 72.27231740306583
```

Basically domain adaptation reduced the sub group bias, but overall accuracy dropped. I just did this as an experiment, i havent fully understood DANN and yet to read the paper.