Learning Lipstick Colors and Brands from Images

(Computer Vision)

Nina Du

Department of Computer Science

Stanford University

ninacdu@stanford.edu

1. Introduction

“Which lipstick did you use in the photo?” This is one of the most asked questions when someone posts a photo of herself on the social platform, whether the person is a celebrity or your best friend. Many people who are interested in cosmetics have obsessions with lipsticks, and this project can serve as a tool to answer one of their most concerned questions when they see a photo of others with makeup.

We propose a two-network pipeline: an image segmentation model (PyTorch) for identifying lips and then a lipstick classification model using CNN.

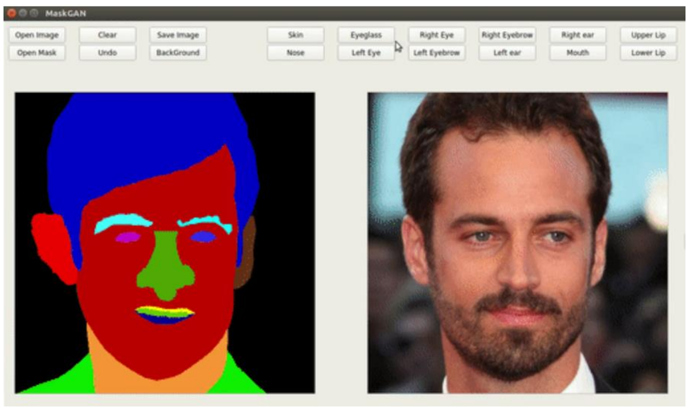
2. Related Work and Literature Review

For this part of the report, we mainly focus on literature review for the first network of the two-network model, the lip segmentation. We found the object instance segmentation method Mask-RCNN[1] and its public code on GitHub [2], the real-time object detection method YOLO [3] and face-parsing.PyTorch model from GitHub most relevant for our project.

We didn’t choose YOLO as our selected methodology, because YOLO outputs a bounding box for the chosen object instead of a mask over the object. If we use it for lip segmentation in our project, there will be too much noise and irrelevant features in the output of lip segmentation, which can greatly increase training errors for the lipstick classification model later. There wasn't a working Mask-RCNN model on face parsing or face segmentation. Mask-RCNN was used more often for large object segmentations rather than facial parts. Face-parsing.PyTorch [4] had good results on segmentation according to the author's face-makeup implementation on GitHub and he also provided a set of tools that we could use. Therefore, after comparing different existing methods for lip segmentation, we decided to use face-parsing.PyTorch as our base model for lip segmentation.

3. Dataset

To train the image segmentation model, we used an open-source CelebAMask-HQ dataset [5]. This large-scale face image dataset has 30,000 high-resolution face images, each image has a segmentation mask of facial attributes such as skin, nose, eyes, eyebrows, ears, lips, etc.



Example of CelebAMask-HQ dataset from Github.com

For the lipstick classification model, since there is no large dataset of lipsticks that fit our needs available on the Internet, we need to manually collect the data. Until now, we have 5 classes of different MAC lipsticks. We downloaded images of people wearing all these 5 different lipsticks from Red(小红书), a Chinese mobile app for product review. For each class, we had around 450 samples, and 2250 samples in total. We use 90% of the dataset as our training examples and the rest 10% as our validation examples. When we load the data, we resize the images to 150x150 in order to standardize the resolution.

The time-consuming nature of this data-collecting process makes it difficult for us to aim for accuracy at that moment.

Examples of raw data

4. Methods

Our overall method is to create a two-network pipeline: a face-parsing model and then a lipstick classification model using CNN.

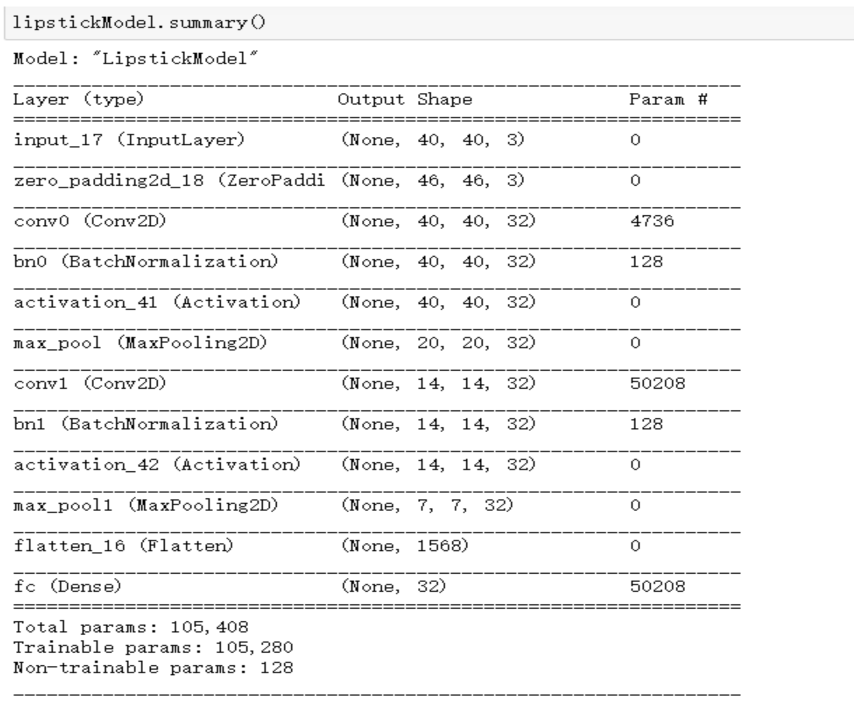
For lips segmentation, we will be using a modified version of the face-parsing.PyTorch model from GitHub to train on the CelebAMask-HQ dataset for lip segmentation. We have successfully generated masks for our own data. However, we are still trying to implement the isolation of lips from the original pictures. We have tried painting other masks white while leaving the lips untouched, but since the original dataset doesn't have a background mask, it only paints the face instead of the entire image. We have emailed the original author of face-parsing.PyTorch for his advice on how to achieve our goal and we hope that will later drastically increase the performance of our lipstickModel.

5. Hyperparameter and Architecture Choices

The hyperparameters that we were concerned with was: the learning-rate, the optimizer choice and the loss function.

We chose the sparse_categorical_crossentropy function due to the nature of our model. Different lipsticks represent sparse categories, so naturally we went with that. We experimented with both the learning rate (0.1, 0.05, 0.01, 0.005, 0.001) and the optimizer(sgd and Adam), the combination of learning rate = 0.01 and Adam optimizer gave us best results (accuracy of around 40%) so that's what we went with.

For the general architecture, we went with the following because going deeper didn't improve our results and going shallower yielded worse ones.



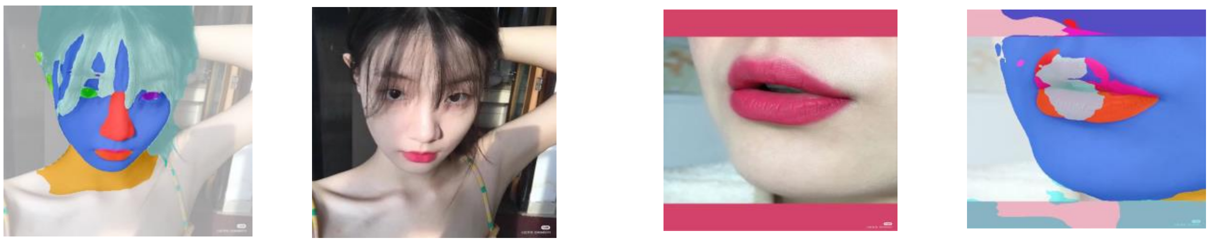
```
lipstickModel.summary()

Model: "LipstickModel"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_17 (InputLayer)        (None, 40, 40, 3)         0
_________________________________________________________________
zero_padding2d_18 (ZeroPaddi (None, 46, 46, 3)         0
_________________________________________________________________
conv0 (Conv2D)               (None, 40, 40, 32)        4736
_________________________________________________________________
bn0 (BatchNormalization)     (None, 40, 40, 32)        128
_________________________________________________________________
activation_41 (Activation)   (None, 40, 40, 32)        0
_________________________________________________________________
max_pool (MaxPooling2D)      (None, 20, 20, 32)        0
_________________________________________________________________
conv1 (Conv2D)               (None, 14, 14, 32)        50208
_________________________________________________________________
bn1 (BatchNormalization)     (None, 14, 14, 32)        128
_________________________________________________________________
activation_42 (Activation)   (None, 14, 14, 32)        0
_________________________________________________________________
max_pool1 (MaxPooling2D)     (None, 7, 7, 32)          0
_________________________________________________________________
flatten_16 (Flatten)         (None, 1568)              0
_________________________________________________________________
fc (Dense)                   (None, 32)                50208
=================================================================
Total params: 105,408
Trainable params: 105,280
Non-trainable params: 128
_________________________________________________________________
```

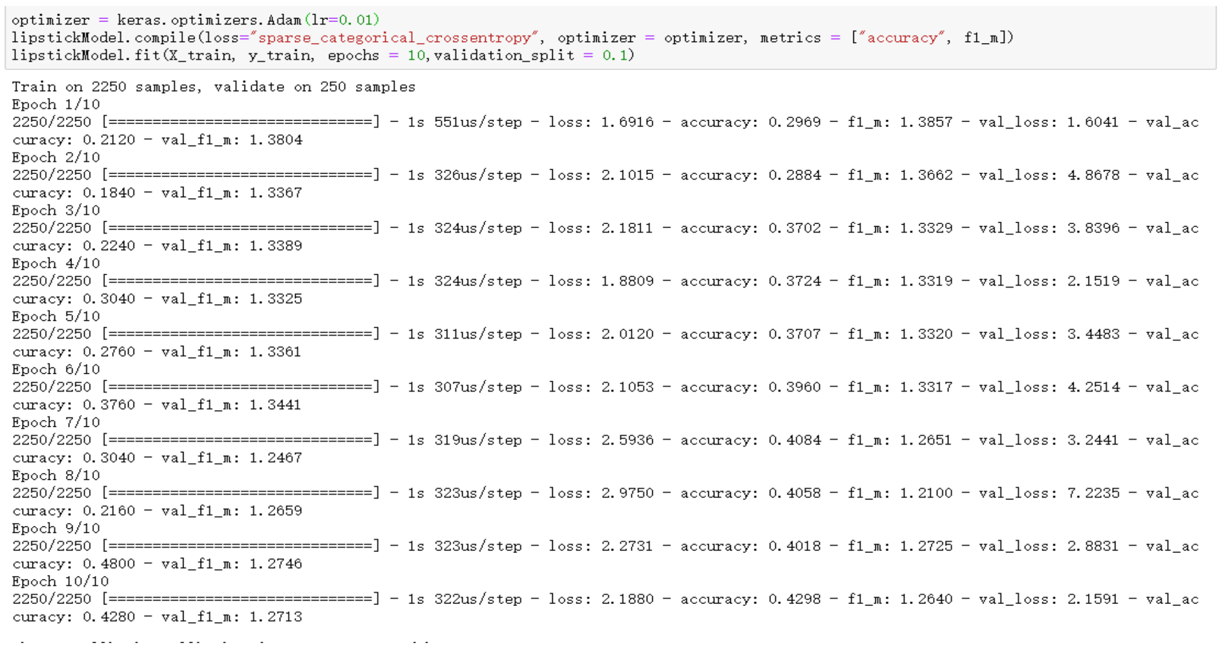
lipstickModel Summary

6. Presentation of Results

After training the edited version of face-parsing.PyTorch on the CelebAMask-HQ dataset, we tried it on our own images. Here are some examples:


Examples of face-parsing model

The second part is to run our lipstickModel on these images. The results with our tweaked hyperparameters are as follows:



```
optimizer = keras.optimizers.Adam(lr=0.01)
lipstickModel.compile(loss="sparse_categorical_crossentropy", optimizer = optimizer, metrics = ["accuracy", f1_m])
lipstickModel.fit(X_train, y_train, epochs = 10,validation_split = 0.1)
```

```
Train on 2250 samples, validate on 250 samples
Epoch 1/10
2250/2250 [==============================] - 1s 551us/step - loss: 1.6916 - accuracy: 0.2969 - f1_m: 1.3857 - val_loss: 1.6041 - val_ac
curacy: 0.2120 - val_f1_m: 1.3804
Epoch 2/10
2250/2250 [==============================] - 1s 326us/step - loss: 2.1015 - accuracy: 0.2884 - f1_m: 1.3662 - val_loss: 4.8678 - val_ac
curacy: 0.1840 - val_f1_m: 1.3367
Epoch 3/10
2250/2250 [==============================] - 1s 324us/step - loss: 2.1811 - accuracy: 0.3702 - f1_m: 1.3329 - val_loss: 3.8396 - val_ac
curacy: 0.2240 - val_f1_m: 1.3389
Epoch 4/10
2250/2250 [==============================] - 1s 324us/step - loss: 1.8809 - accuracy: 0.3724 - f1_m: 1.3319 - val_loss: 2.1519 - val_ac
curacy: 0.3040 - val_f1_m: 1.3325
Epoch 5/10
2250/2250 [==============================] - 1s 311us/step - loss: 2.0120 - accuracy: 0.3707 - f1_m: 1.3320 - val_loss: 3.4483 - val_ac
curacy: 0.2760 - val_f1_m: 1.3361
Epoch 6/10
2250/2250 [==============================] - 1s 307us/step - loss: 2.1053 - accuracy: 0.3960 - f1_m: 1.3317 - val_loss: 4.2514 - val_ac
curacy: 0.3760 - val_f1_m: 1.3441
Epoch 7/10
2250/2250 [==============================] - 1s 319us/step - loss: 2.5936 - accuracy: 0.4084 - f1_m: 1.2651 - val_loss: 3.2441 - val_ac
curacy: 0.3040 - val_f1_m: 1.2467
Epoch 8/10
2250/2250 [==============================] - 1s 323us/step - loss: 2.9750 - accuracy: 0.4058 - f1_m: 1.2100 - val_loss: 7.2235 - val_ac
curacy: 0.2160 - val_f1_m: 1.2659
Epoch 9/10
2250/2250 [==============================] - 1s 323us/step - loss: 2.2731 - accuracy: 0.4018 - f1_m: 1.2725 - val_loss: 2.8831 - val_ac
curacy: 0.4800 - val_f1_m: 1.2746
Epoch 10/10
2250/2250 [==============================] - 1s 322us/step - loss: 2.1880 - accuracy: 0.4298 - f1_m: 1.2640 - val_loss: 2.1591 - val_ac
curacy: 0.4280 - val_f1_m: 1.2713
```

Results of lipstickModel

7. Analysis of Results

In the first segmentation model, most of our images worked very well (as shown in the example on the left in section 6), while some didn't work as expected (example on the right in section 6). We can't point our fingers on what exactly is wrong here. We suspect that the original structure of the image (being a partial face) could be one of the major issues here. We also suspect that changing the color scheme from BGR to others, such as HSV could help the model identify certain areas better.

In our lipstickModel, after tweaking with the last Fully Connected Layer parameters, we finally have a lost function that is not nan anymore. We can see that in our model, loss diminishes, the F1 score gets closer to 1, and accuracy grows from epoch to epoch.

8. Conclusion/Future Work

40% of accuracy is still not ideal for our usage and purposes, and we will be thinking of ways of improving the performance of our model in the future. We still need to keep expanding our dataset. Creating such a dataset by hand is laborious, so we are exploring ways such as web crawling to collect data. For the next step regarding our dataset, we will still stick with the 5 categories we have right now and try to source our data in a more efficient way. We think only when we have better performance in these 5 categories should we expand to other categories.

After we expand to a significant number of categories, we will then consider extreme multilabel classification models for the lipstick classification part of our pipeline.

Poor result of our current model is not caused by the lack of data alone. Our dataset as it is now is quite noisy. We will continue working on the face-parsing part of our pipeline to segment the lips from each image reliably.

After future work, we hope that we can create a tool that provides convenience for people to identify the brands of lipsticks. We will then take this project as our starting point, and explore the classification of brands of other types of makeup such as those of lip gloss, eye palettes and nail polish to build a powerful makeup brands identification and similar products suggestion tool.

References

[1]Bharati, P., & Pramanik, A. (2019). Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey. *Computational Intelligence in Pattern Recognition Advances in Intelligent Systems and Computing*, 657–668. doi: 10.1007/978-981-13-9042-5_56

[2]Abdulla, W. (2017). Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. *GitHub Repository*. Retrieved from https://github.com/matterport/Mask_RCNN

[3] Redmon, J. (n.d.). Retrieved from https://pjreddie.com/darknet/yolo/

[4]Zllrunning. (2019, May 18). zllrunning/face-parsing.PyTorch. Retrieved from https://github.com/zllrunning/face-parsing.PyTorch

[5]Lee, C.-H., Wu, Z., & Luo, P. (2020). MaskGAN: Towards Diverse and Interactive Facial Image Manipulation. Retrieved from https://github.com/switchablenorms/CelebAMask-HQCelebAMask-HQ