Face Type Classification Report

Authors: Chengkai Yao, Minghan Wu, Yunjin(Grace) Zhu, Yulin Chen, Angelina Zhang, Yue Yin

Abstract

The global skincare market, projected to reach $189.3 billion by 2025 (Statista, 2022), highlights the importance of accurately classifying skin types for effective skincare product selection. In this study, we use the "Oily, Dry and Normal Skin Types Dataset" to develop a skin type classifier. We implemented Support Vector Machines (SVM) and Logistic Regression as baseline models and explored the efficacy of Convolutional Neural Networks (CNNs) for enhanced classification. Our baseline models achieved accuracies of 28.35% (Logistic Regression) and 42.54% (SVM), while our CNN model, featuring inception modules, achieved a notable accuracy of 68.51%, representing a significant improvement over the baseline models. These findings highlight the potential of CNNs in precise skin type classification, paving the way for more personalized and effective skincare solutions.

1 Introduction

The Skin Type Classification Project works on the most modern classification of skin types, "Dry", "Normal", and "Oily", from facial image data. We chose this initiative for its importance in changing how recommendations for personalized skincare and dermatology are made. We aim to develop an accurate model by making use of advanced machine learning techniques, especially Convolutional Neural Networks(CNNs). The project shows how deep learning can work its way into the spheres of medicine and cosmetics. It has much wider-scope implications: obtaining a reliable predictive model would lead to highly individualized approaches to skincare, optimization in dermatology, proper development and positioning of cosmetic products, and a contribution to medical image analysis.

1.1 Dataset

Our datasets consist of different facial pictures, each of size 650 by 650 by 3 (three channel due to RGB structure), and there are in total around 3000 data samples which we will divide into training and testing sets. Each data image is labeled with either “Dry”,

“Normal”, or “Oily”, the facial type we try to predict. Note these pictures are from advertisements and social media, so they do not have a standardized style.

1.2 Preprocessing

To speed up the optimization process during model training, all pixel values are divided by 255 as a normalization. We split the data into training, testing, and validation sets at a ratio of 8:1:1. Each picture in the training set has a true label for skin type classification. We also encoded each label as 0, 1, and 2.

Our datasets are facial images from advertisements and social media, and they do not have a standardized style. This might not go well with some machine learning algorithms like logistic regression and SVM, which rely on pixel position. In order to fix this, we utilized a pre-trained model to cut out human faces and then padded these images with gray color to maintain their original dimension of 650x650 and position the faces at the center.

In our datasets, due to their size, the original images have very high dimensionality (650x650 pixels with 3 color channels results in over a million dimensions per image). This potentially results in "curse of dimensionality": computational complexity, overfitting, difficulty in visualizing or understanding the data, etc.. We choose to use Principal Component Analysis (PCA) to help reduce the dataset's dimensionality while retaining maximum crucial information. We applied PCA to each channel individually. After applying PCA to reduce the dimensionality of the channel to 30 principal components, we have in total 90 dimension feature space for training, making them more manageable for our baseline models.

2 Method

2.1 Baseline model

We used two models as our baseline model: logistic regression model and SVM model.

2.1.1 Logistic Regression Model

We aim to classify skin type into three classes, while logistic regression is designed for binary classification, so we used softmax to enable multi-class classification. We performed feature scaling using standardization to ensure accuracy and reliability. By creating and training this model with 'newton-cg' solver, we decreased the risk of overfitting.

However, the Logistic Regression model only achieved an accuracy of 28.35%. Even after applying PCA for dimensionality reduction, Logistic Regression still faced challenges, because PCA does not always make the data linearly separable. Also, logistic regression models ignores the spatial correlation of the values of the pixels, which undermines its effectiveness for image data.

2.1.2 SVM Model

Unlike Logistic Regression, SVM does not just try to optimize a cost function but rather focuses on maximizing the margin, thus providing a buffer zone for better handling uncertainty in the test data. By using the RBF kernel and the "one-vs-rest" strategy, SVM maps the input data to a higher-dimensional space where classes can be separated more effectively. This approach helps SVM better manage the non-linear relationships within the data. As a result, SVM performs better than Logistic Regression, achieving a moderate accuracy of 42.54%.

An additional explanation for the better performance of SVM is the model's robustness to dimensionality. This robustness is due to two inherent properties of SVM:

- We only find a hyperplane, or equivalently, the normal vector. So the number of parameters is simply d if the data points lie in R^d.
- One can show that the weight vector w must lie in the span of the support vectors, and hence the classifier operates only on a much reduced subspace compared to R^d. That is, feature selection is inherently done by SVM.

2.2 CNN Models

By stacking multiple convolutional, pooling, and fully connected layers, CNNs can learn hierarchical representations of the input data. This hierarchical representation allows the model to capture both low-level features (e.g., edges, corners) and high-level features (e.g., objects, patterns) in the data. We tested 4 CNN models to determine the best structure for the given task. Due to limited computation resources, we trained each model for 20 epochs.

2.2.1 Model 1: Global-average-pooling Model

Model 1 consists of 4 CNN units. Each CNN unit includes a Convolution layer, followed by batched normalization and max-pooling. The 4 Convolution layers each has a

RELU activation function and 3x3 kernels, and they respectively have filters of dimensions 32, 64, 128, and 256. After the convolutional layers follows a GlobalAveragePooling2D layer, two dense layers with 128 and 64 units, and a dropout regularization, in that order. The output layer divides the final categorization into three classes using a softmax activation function.

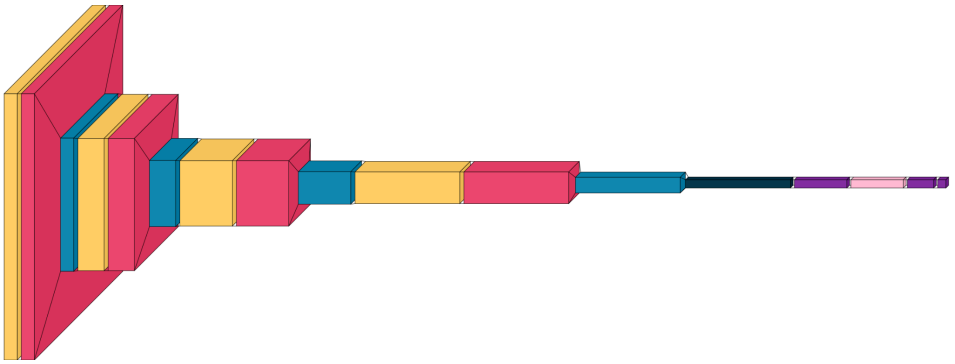

Figure 1. Layer Dimensions of Model 1

2.2.2 Model 2: Flatten Model

This model has everything Model 1 has, but using a flattening layer in place of the GlobalAveragePooling2D layer between the CNN layers and the dense layers.

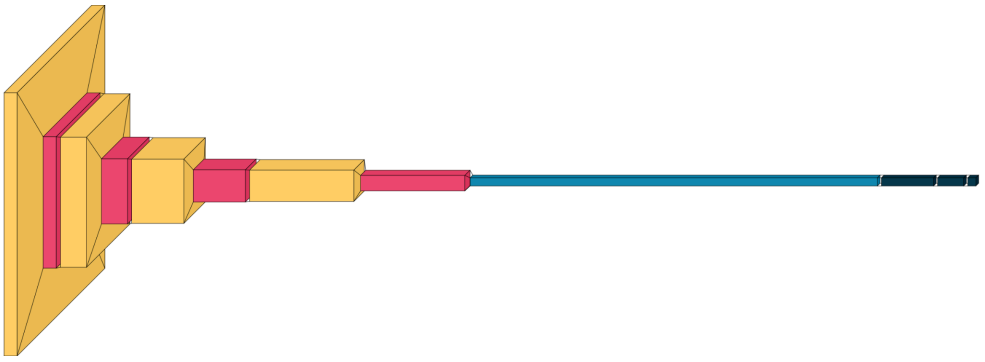

Figure 2. Layer Dimensions of Model 2

2.2.3 Model 3: Resnet Structure Model

Residual blocks with 64, 128 and 256 filters are corporated in this model. It begins with a Conv2D layer with 64 filters and a 7x7 kernel, followed by a max-pooling layer. Two Conv2D layers with ReLU activation and an identity shortcut link make up each residual

block. A flatten layer, two dense layers with 128 and 64 units, and an output layer with softmax activation function completesx the network.

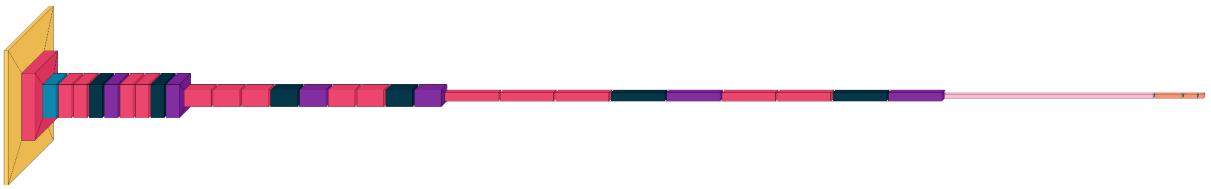

Figure 3. Layer Dimensions of Model 3

2.2.4 Model 4: Inception Structured Model

Inception modules are designed to capture multi-scale features by using parallel convolutional layers with different filter sizes. Each inception module consists of multiple branches, where each branch performs a different type of convolution operation (e.g., 1x1, 3x3, 5x5). These branches are then concatenated to form the output of the inception module. It then flattens all layers and connected to a fully connected neural network.

By combining different filter sizes and allowing the network to learn which features to use, inception modules enable the model to capture both local and global information in the input data. This helps improve the model's ability to recognize complex patterns and objects.

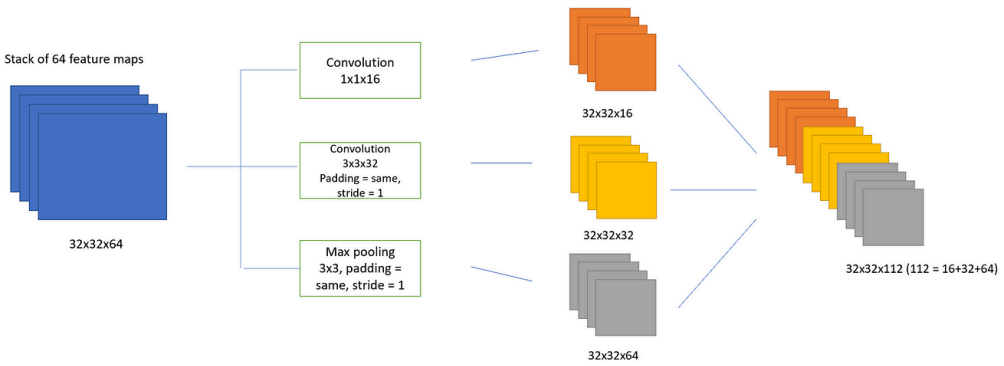

Figure 4. Inception Modules

Our implementation of an inception module consists of 4 branches (1x1, 3x3, 5x5, and maxPooling) which are concatenated in the same manner as in the figure.

The specification of the layers in the model can be seen in this chart:

Filter Dimenstion	Padding	Strides	Output Dimension	Type
7x7	'same'	2	(325, 325, 32)	Conv2D
3x3	/	2	(162, 162, 32)	Maxpooling
/	'same'	/	(162, 162, 128)	Inception Module
/	'same'	/	(162, 162, 256)	Inception Module
3x3	/	2	(80, 80, 256)	Maxpooling
/	'same'	/	(80, 80, 512)	Inception Module
/	'same'	/	(80, 80, 1024)	Inception Module
3x3	/	2	(39, 39, 1024)	Maxpooling
/	/	/	1557504	flatten
/	/	/	128	Dense
/	/	/	64	Dense
/	/	/	3	Dense

Figure 5. Layer Specification for Model 4

Note: All activation functions are “relu”.

We then proceed to train the model using “adam” as optimizer with respect to “categorical_crossentropy”.

Figure 6. Layer Dimensions of Model 4

3 Results

3.1 Training for each Model

Model 1:

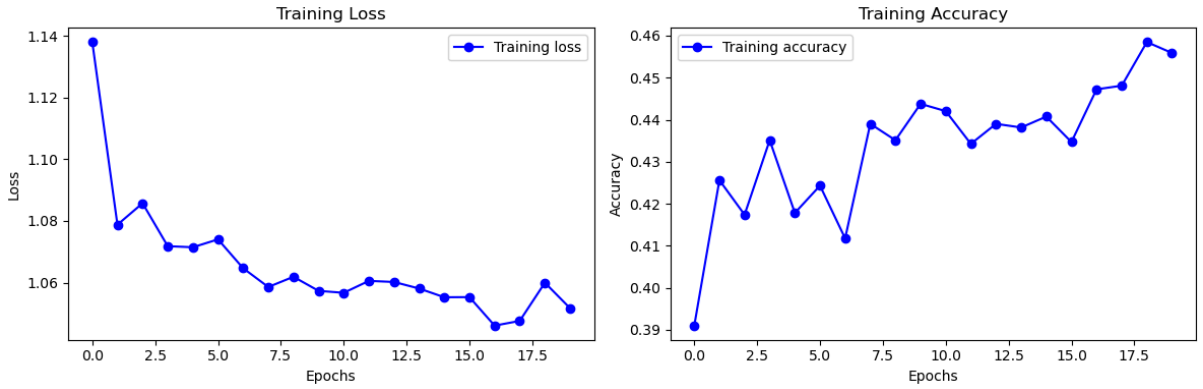

Figure 7. Training Loss and Accuracy for Model 1

Model 2:

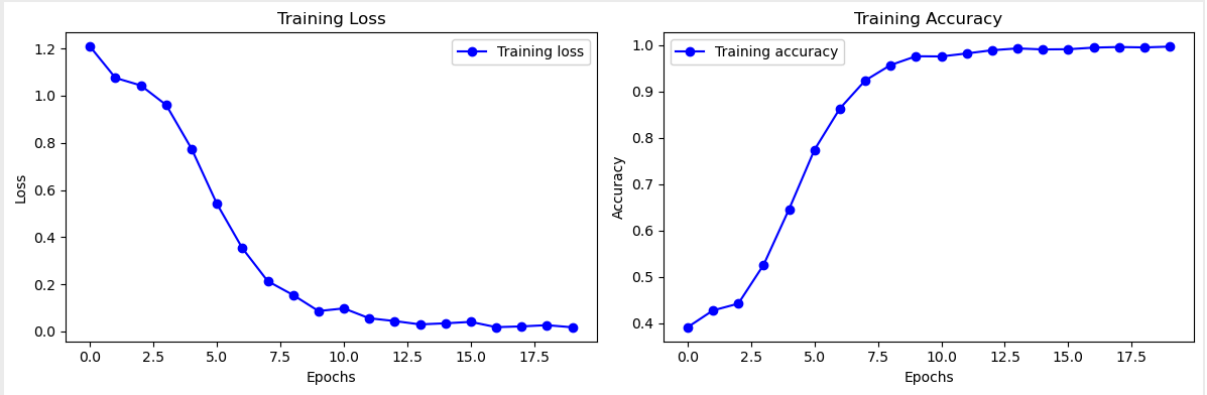

Figure 8. Training Loss and Accuracy for Model 2

Model 3:

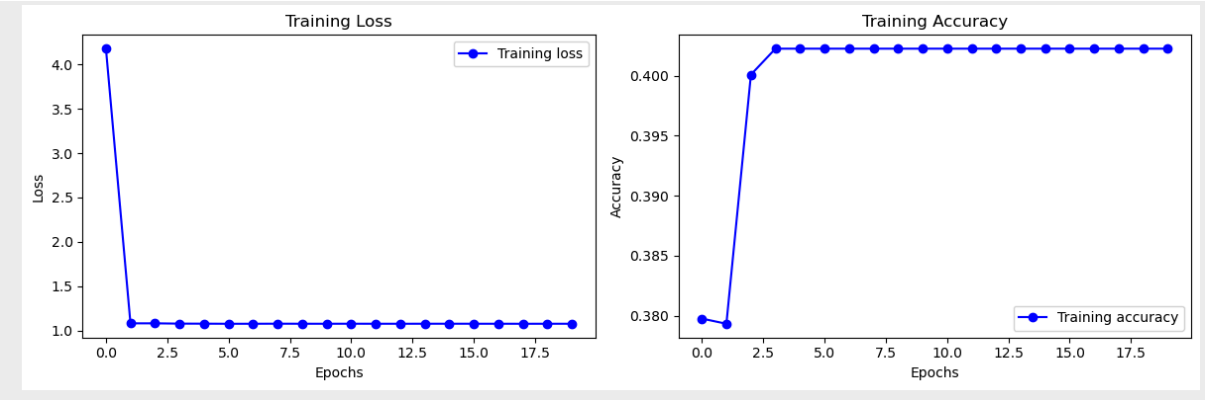

Figure 9. Training Loss and Accuracy for Model 3

Model 4:

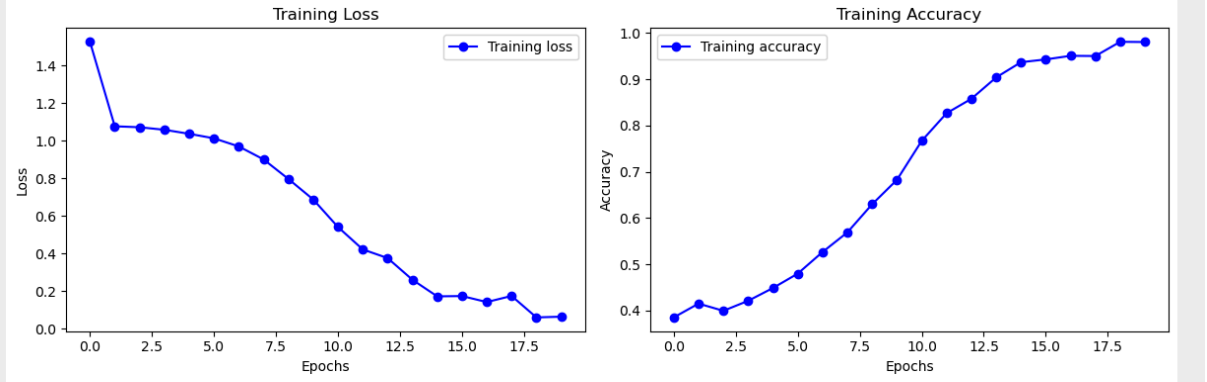

Figure 10. Training Loss and Accuracy for Model 4

3.2 Testing for each model

Each model was trained for 20 epochs and evaluated on the test set. The following results were obtained:

- **Model 1**:
  - Test Loss: 1.3759
  - Test Accuracy: 0.3616
- **Model 2**:
  - Test Loss: 3.2122
  - Test Accuracy: 0.6401
- **Model 3**:
  - Test Loss: 1.0753
  - Test Accuracy: 0.4031
- **Model 4**:

- Test Loss: 3.0333
- Test Accuracy: 0.6851

From the above result, we can see that Model 2 and Model 4 performed the best. Indeed, reaching 60% is way better than the baseline model.

4 Discussion

4.1 Baseline Model Analysis

Dataset exploration initially showed us images that had uniform size, 650x650 pixels, and three color channels, which made it easy to have a standard preprocessing pipeline. The fact that the faces in the images were centered might be a problem for some machine learning algorithms that depend on the position of pixels. We addressed this by running a pre-trained model that cuts out the faces and gray-fill the images while maintaining their dimensions, retaining focus on facial features and normalizing pixel values by dividing by 255. The images were standardized with this approach, but some contextual information from surrounding areas could have been lost. Since the images' dimensionality was very high, we did Principal Component Analysis (PCA) for dimensionality reduction as well; it makes the dataset simpler when working on baseline models such as Logistic Regression and SVM.

4.2 CNN Model Analysis

For the first model, we use GlobalAveragePolling2D. The purpose of putting the GlobalAveragePooling2D layer and Maxpooling while designing would be to decrease the number of parameters, aiming to improve the training speed. Without the Maxpooling, the model can be extremely large; the number of parameters would reach around one billion. However, the model performed very poorly, even lower than our baseline SVM Model. One possible reason we suspect for the low accuracy is that important information is lost during global average pooling, which we removed for our second model.

For the second Model, comparing with model 1, we replaced ‘GlobalAveragePooling2D()’ from model 1 by using a Flatten layer to directly flatten the output of the final convolutional layer before passing it to the dense layers. From training loss and accuracy graphs, the model performs well with no significant sign of overfitting and increases the accuracy. Although it takes longer time to train due to much more parameters,

the increase in accuracy for testing indicates that indeed important information is preserved during feature extraction of the model.

For the third model we implemented a resnet-like structure, since Resnet model has proven to be reliable in many other applications. By theory, the accuracy of the network should be improved by the residual blocks. However, the result is not satisfactory. Upon examining the training loss and accuracy, it appears that the optimizer has become trapped in a local minimum due to vanishing gradient, preventing further improvement in the model's performance. Thus the parameter of the model is not properly trained, hence resulting in poor accuracy. Due to limited time and computation resources, we decided to experiment with other structures.

For our fourth and final model, we implemented an inception-like structure. Inception modules are designed to capture multi-scale features by using parallel convolutional layers with different filter sizes. Each inception module consists of multiple branches, where each branch performs a different type of convolution operation (e.g., 1x1, 3x3, 5x5). These branches are then concatenated to form the output of the inception module. By combining different filter sizes and allowing the network to learn which features to use, inception modules enable the model to capture both local and global information in the input data. The difference in kernel size is what we believe to be particularly helpful for our dataset: as we preprocess our data, we discovered that the facial areas for different images vary in sizes. For smaller faces, a small kernel will satisfy, while for larger faces, a larger kernel size is required for the CNN model to capture all necessary information. The unique design of inception model accommodates this problem effectively, leading to much better performance.

5 Conclusion

Looking back at the Skin Type Classification Project, we can see a lot of knowledge acquired and challenges faced that helped us better understand image classification tasks. The decision we made during our data preprocessing to crop the faces and fill the images with gray while still retaining their dimensions was very important in that it helped the models concentrate on the necessary features, though some information about what was around them contextually might have increased performance. PCA was good for simplifying the dataset, but its limited ability to guarantee linear separability only served to show how much more need there was for advanced dimensionality reduction techniques. Logistic Regression and SVM models provided a baseline for evaluation. The different CNN architectures we implemented clearly showed that the convolution structure is better suited for image

classification tasks. The inception-structure model proved to be the most accurate, demonstrating the importance of multi-scale feature extraction. The most important take-away from this report is to choose the right model for the dataset you have. As seen, this creates a substantial difference.

Future directions would involve much hyperparameter tuning, more advanced regularization techniques, leveraging transfer learning, increasing dataset diversity through data augmentation and exploring some advanced architectures like ResNet or EfficientNet. Lessons have been very useful in showing just how important data preprocessing is, what the limitations of linear models are, and what deep learning techniques can do. Our results are promising; however, continuous improvement and exploration to enhance accuracy and robustness would contribute towards making strides in personalized skincare and dermatological care, but from the perspective of improving overall skin health for a wider audience.

6 Citations

Dissanayake, S. (2024, February 20). Oily, dry and normal skin types dataset. Kaggle. https://www.kaggle.com/datasets/shakyadissanayake/oily-dry-and-normal-skin-types-dataset/data

Fernandes, A. A., Figueiredo Filho, D. B., Rocha, E. C., & Nascimento, W. da. (2020). Read this paper if you want to learn logistic regression. Revista de Sociologia e Política, 28(74). https://doi.org/10.1590/1678-987320287406en

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support Vector Machines. IEEE Intelligent Systems and Their Applications, 13(4), 18–28. https://doi.org/10.1109/5254.708428

Mishra, S., Sarkar, U., Taraphder, S., Datta, S., Swain, D., Saikhom, R., Panda, S., & Laishram, M. (2017). Principal component analysis. International Journal of Livestock Research, 1. https://doi.org/10.5455/ijlr.20170415115235

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. https://doi.org/ https://doi.org/10.48550/arXiv.1511.08458

Saiwaeo, S., Arwatchananukul, S., Mungmai, L., Preedalikit, W., & Aunsri, N. (2023). Human skin type classification using image processing and deep learning approaches. Heliyon, 9(11). https://doi.org/10.1016/j.heliyon.2023.e21176

7 Appendix

See Github for codes. Baseline model are stored in `baseline.ipynb`, and CNN models are stored in `CNN_v1.ipynb`.

8 Statement of Collaboration

Name: Chengkai Yao, Title: Leader, Project Manager, Coder, Writer,

Contribution: Lead and assign members with tasks; design the framework of this project; coded part of preprocessing (Implemented PCA), all of Baseline model and CNN model; contributed on the report's structure, details, and insights(Discussion).

Name: Angelina Zhang

Contributed to preprocessing data, writing report for baseline models and cnn models with Grace. Participated in discussions and managed the Github repository.

Name: Grace(Yunjin) Zhu

Some code in baseline models, writing final report for baseline models and CNN models with Angelina

Name: Yulin Chen

Found the dataset.Some code in baseline models with Grace,writing final report for introduction,discussion and conclusion.

Name: Minghan Wu

Implemented cutting-out-faces and padding for preprocessing; Attempted to fine tune the models on SDSC; Worked on the SVM and Inception parts of this report; Improved the format and language of this report.

Name: Yue Yin

I contributed significantly to our project by leading the data preprocessing, ensuring the dataset was cleaned and prepared for analysis. I wrote the abstract, summarizing our research clearly, and compiled the citation section to credit all sources correctly. Additionally, I refined the final report by proofreading, enhancing clarity, and incorporating feedback

On Readme.md

Face Type Classification Report

Authors: Chengkai Yao, Minghan Wu, Yunjin(Grace) Zhu, Yulin Chen, Angelina Zhang, Yue Yin

Abstract

The global skincare market, projected to reach $189.3 billion by 2025 (Statista, 2022), highlights the importance of accurately classifying skin types for effective skincare product selection. In this study, we use the "Oily, Dry and Normal Skin Types Dataset" to develop a skin type classifier. We implemented Support Vector Machines (SVM) and Logistic Regression as baseline models and explored the efficacy of Convolutional Neural Networks (CNNs) for enhanced classification. Our baseline models achieved accuracies of 28.35% (Logistic Regression) and 42.54% (SVM), while our CNN model, featuring inception modules, achieved a notable accuracy of 68.51%, representing a significant improvement over the baseline models. These findings highlight the potential of CNNs in precise skin type classification, paving the way for more personalized and effective skincare solutions.

1 Introduction

The Skin Type Classification Project works on the most modern classification of skin types, "Dry", "Normal", and "Oily", from facial image data. We chose this initiative for its importance in changing how recommendations for personalized skincare and dermatology are made. We aim to develop an accurate model by making use of advanced machine learning

techniques, especially Convolutional Neural Networks(CNNs). The project shows how deep learning can work its way into the spheres of medicine and cosmetics. It has much wider-scope implications: obtaining a reliable predictive model would lead to highly individualized approaches to skincare, optimization in dermatology, proper development and positioning of cosmetic products, and a contribution to medical image analysis.

1.1 Dataset

Our datasets consist of different facial pictures, each of size 650 by 650 by 3 (three channels due to RGB structure), and there are in total around 3000 data samples which we will divide into training and testing sets. Each data image is labeled with either “Dry”, “Normal”, or “Oily”, the facial type we try to predict. Note these pictures are from advertisements and social media, so they do not have a standardized style.

1.2 Preprocessing

To speed up the optimization process during model training, all pixel values are divided by 255 as a normalization. We split the data into training, testing, and validation sets at a ratio of 8:1:1. Each picture in the training set has a true label for skin type classification. We also encoded each label as 0, 1, and 2.

Our datasets are facial images from advertisements and social media, and they do not have a standardized style. This might not go well with some machine learning algorithms like logistic regression and SVM, which rely on pixel position. In order to fix this, we utilized a pre-trained model to cut out human faces and then padded these images with gray color to maintain their original dimension of 650x650 and position the faces at the center.

In our datasets, due to their size, the original images have very high dimensionality (650x650 pixels with 3 color channels results in over a million dimensions per image). This potentially results in a "curse of dimensionality": computational complexity, overfitting, difficulty in visualizing or understanding the data, etc. We choose to use Principal Component Analysis (PCA) to help reduce the dataset's dimensionality while retaining maximum crucial information. We applied PCA to each channel individually. After applying PCA to reduce the dimensionality of the channel to 30 principal components, we have in total 90 dimension feature space for training, making them more manageable for our baseline models.

2 Method

2.1 Baseline model

We used two models as our baseline model: the logistic regression model and the SVM model.

2.1.1 Logistic Regression Model

We aim to classify skin type into three classes, while logistic regression is designed for binary classification, so we used softmax to enable multi-class classification. We performed feature scaling using standardization to ensure accuracy and reliability. By creating and training this model with a 'newton-cg' solver, we decreased the risk of overfitting.

However, the Logistic Regression model only achieved an accuracy of 28.35%. Even after applying PCA for dimensionality reduction, Logistic Regression still faced challenges, because PCA does not always make the data linearly separable. Also, logistic regression models ignore the spatial correlation of the values of the pixels, which undermines its effectiveness for image data.

2.1.2 SVM Model

Unlike Logistic Regression, SVM does not just try to optimize a cost function but rather focuses on maximizing the margin, thus providing a buffer zone for better handling uncertainty in the test data. By using the RBF kernel and the "one-vs-rest" strategy, SVM maps the input data to a higher-dimensional space where classes can be separated more effectively. This approach helps SVM better manage the non-linear relationships within the data. As a result, SVM performs better than Logistic Regression, achieving a moderate accuracy of 42.54%.

An additional explanation for the better performance of SVM is the model’s robustness to dimensionality. This robustness is due to two inherent properties of SVM:

We only find a hyperplane, or equivalently, the normal vector. So the number of parameters is simply d if the data points lie in Rd.

One can show that the weight vector w must lie in the span of the support vectors, and hence the classifier operates only on a much reduced subspace compared to Rd. That is, feature selection is inherently done by SVM.

2.2 CNN Models

By stacking multiple convolutional, pooling, and fully connected layers, CNNs can learn hierarchical representations of the input data. This hierarchical representation allows the

model to capture both low-level features (e.g., edges, corners) and high-level features (e.g., objects, patterns) in the data. We tested 4 CNN models to determine the best structure for the given task. Due to limited computation resources, we trained each model for 20 epochs.

2.2.1 Model 1: Global-average-pooling Model

Model 1 consists of 4 CNN units. Each CNN unit includes a Convolution layer, followed by batched normalization and max-pooling. The 4 Convolution layers each have a RELU activation function and 3x3 kernels, and they respectively have filters of dimensions 32, 64, 128, and 256. After the convolutional layers follow a GlobalAveragePooling2D layer, two dense layers with 128 and 64 units, and a dropout regularization, in that order. The output layer divides the final categorization into three classes using a softmax activation function.

Figure 1. Layer Dimensions of Model 1

2.2.2 Model 2: Flatten Model

This model has everything Model 1 has but uses a flattening layer in place of the GlobalAveragePooling2D layer between the CNN layers and the dense layers.

Figure 2. Layer Dimensions of Model 2

2.2.3 Model 3: Resnet Structure Model

Residual blocks with 64, 128, and 256 filters are corporated in this model. It begins with a Conv2D layer with 64 filters and a 7x7 kernel, followed by a max-pooling layer. Two Conv2D layers with ReLU activation and an identity shortcut link make up each residual block. A flattened layer, two dense layers with 128 and 64 units, and an output layer with a softmax activation function complete the network.

Figure 3. Layer Dimensions of Model 3

2.2.4 Model 4: Inception Structured Model

Inception modules are designed to capture multi-scale features by using parallel convolutional layers with different filter sizes. Each inception module consists of multiple branches, where each branch performs a different type of convolution operation (e.g., 1x1, 3x3, 5x5). These branches are then concatenated to form the output of the inception module. It then flattens all layers and connects to a fully connected neural network.

Combining different filter sizes and allowing the network to learn which features to use inception modules enable the model to capture both local and global information in the input data. This helps improve the model's ability to recognize complex patterns and objects.

Figure 4. Inception Modules

Our implementation of an inception module consists of 4 branches (1x1, 3x3, 5x5, and maxPooling) which are concatenated in the same manner as in the figure.

The specification of the layers in the model can be seen in this chart:

Figure 5. Layer Specification for Model 4

Note: All activation functions are “relu”.

We then proceed to train the model using “adam” as an optimizer with respect to “categorical_crossentropy”.

Figure 6. Layer Dimensions of Model 4

3 Results

3.1 Training for each Model

Model 1:

Figure 7. Training Loss and Accuracy for Model 1

Model 2:

Figure 8. Training Loss and Accuracy for Model 2

Model 3:

Figure 9. Training Loss and Accuracy for Model 3

Model 4:

Figure 10. Training Loss and Accuracy for Model 4

3.2 Testing for each model

Each model was trained for 20 epochs and evaluated on the test set. The following results were obtained:

- **Model 1**:
  - Test Loss: 1.3759
  - Test Accuracy: 0.3616
- **Model 2**:
  - Test Loss: 3.2122
  - Test Accuracy: 0.6401
- **Model 3**:
  - Test Loss: 1.0753
  - Test Accuracy: 0.4031

- **Model 4**:

  - Test Loss: 3.0333

  - Test Accuracy: 0.6851

From the above result, we can see that Model 2 and Model 4 performed the best. Indeed, reaching 60% is way better than the baseline model.

4 Discussion

4.1 Baseline Model Analysis

Dataset exploration initially showed us images that had uniform size, 650x650 pixels, and three color channels, which made it easy to have a standard preprocessing pipeline. The fact that the faces in the images were centered might be a problem for some machine-learning algorithms that depend on the position of pixels. We addressed this by running a pre-trained model that cuts out the faces and gray-fill the images while maintaining their dimensions, retaining focus on facial features, and normalizing pixel values by dividing by 255. The images were standardized with this approach, but some contextual information from surrounding areas could have been lost. Since the images' dimensionality was very high, we did Principal Component Analysis (PCA) for dimensionality reduction as well; it makes the dataset simpler when working on baseline models such as Logistic Regression and SVM.

4.2 CNN Model Analysis

For the first model, we use GlobalAveragePolling2D. The purpose of putting the GlobalAveragePooling2D layer and Maxpooling while designing would be to decrease the number of parameters, aiming to improve the training speed. Without the Maxpooling, the model can be extremely large; the number of parameters would reach around one billion. However, the model performed very poorly, even lower than our baseline SVM Model. One possible reason we suspect for the low accuracy is that important information is lost during global average pooling, which we removed for our second model.\

For the second Model, compared with model 1, we replaced ‘GlobalAveragePooling2D()’ from model 1 by using a flattened layer to directly flatten the output of the final convolutional

layer before passing it to the dense layers. From training loss and accuracy graphs, the model performs well with no significant sign of overfitting and increases the accuracy. Although it takes a longer time to train due to many more parameters, the increase in accuracy for testing indicates that indeed important information is preserved during feature extraction of the model.

For the third model, we implemented a resnet-like structure since the Resnet model has proven to be reliable in many other applications. By theory, the accuracy of the network should be improved by the residual blocks. However, the result is not satisfactory. Upon examining the training loss and accuracy, it appears that the optimizer has become trapped in a local minimum due to a vanishing gradient, preventing further improvement in the model's performance. Thus the parameter of the model is not properly trained, hence resulting in poor accuracy. Due to limited time and computation resources, we decided to experiment with other structures.

For our fourth and final model, we implemented an inception-like structure. Inception modules are designed to capture multi-scale features by using parallel convolutional layers with different filter sizes. Each inception module consists of multiple branches, where each branch performs a different type of convolution operation (e.g., 1x1, 3x3, 5x5). These branches are then concatenated to form the output of the inception module. By combining different filter sizes and allowing the network to learn which features to use in inception modules enable the model to capture both local and global information in the input data. The difference in kernel size is what we believe to be particularly helpful for our dataset: as we preprocessed our data, we discovered that the facial areas for different images vary in size. For smaller faces, a small kernel will satisfy, while for larger faces, a larger kernel size is required for the CNN model to capture all necessary information. The unique design of the inception model accommodates this problem effectively, leading to much better performance.

5 Conclusion

Looking back at the Skin Type Classification Project, we can see a lot of knowledge acquired and challenges faced that helped us better understand image classification tasks. The decision we made during our data preprocessing to crop the faces and fill the images with gray while still retaining their dimensions was very important in that it helped the models concentrate on the necessary features, though some information about what was around them contextually might have increased performance. PCA was good for simplifying the

dataset, but its limited ability to guarantee linear separability only served to show how much more need there was for advanced dimensionality reduction techniques. Logistic Regression and SVM models provided a baseline for evaluation. The different CNN architectures we implemented clearly showed that the convolution structure is better suited for image classification tasks. The inception-structure model proved to be the most accurate, demonstrating the importance of multi-scale feature extraction. The most important takeaway from this report is to choose the right model for the dataset you have. As seen, this creates a substantial difference.

Future directions would involve much hyperparameter tuning, more advanced regularization techniques, leveraging transfer learning, increasing dataset diversity through data augmentation, and exploring some advanced architectures like ResNet or EfficientNet. Lessons have been very useful in showing just how important data preprocessing is, what the limitations of linear models are, and what deep learning techniques can do. Our results are promising; however, continuous improvement and exploration to enhance accuracy and robustness would contribute towards making strides in personalized skincare and dermatological care, but from the perspective of improving overall skin health for a wider audience.

6 Citations

Dissanayake, S. (2024, February 20). Oily, dry and normal skin types dataset. Kaggle. https://www.kaggle.com/datasets/shakyadissanayake/oily-dry-and-normal-skin-types-dataset/data

Fernandes, A. A., Figueiredo Filho, D. B., Rocha, E. C., & Nascimento, W. da. (2020). Read this paper if you want to learn logistic regression. Revista de Sociologia e Política, 28(74). https://doi.org/10.1590/1678-987320287406en

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support Vector Machines. IEEE Intelligent Systems and Their Applications, 13(4), 18–28. https://doi.org/10.1109/5254.708428

Mishra, S., Sarkar, U., Taraphder, S., Datta, S., Swain, D., Saikhom, R., Panda, S., & Laishram, M. (2017). Principal component analysis. International Journal of Livestock Research, 1. https://doi.org/10.5455/ijlr.20170415115235

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. https://doi.org/ https://doi.org/10.48550/arXiv.1511.08458

Saiwaeo, S., Arwatchananukul, S., Mungmai, L., Preedalikit, W., & Aunsri, N. (2023). Human skin type classification using image processing and deep learning approaches. Heliyon, 9(11). https://doi.org/10.1016/j.heliyon.2023.e21176

7 Appendix

See Github for codes. Baseline models are stored in `baseline.ipynb`, and CNN models are stored in `CNN_v1.ipynb`.

8 Statement of Collaboration

Name: Chengkai Yao, Title: Leader, Project Manager, Coder, Writer,

Contribution: Lead and assign members with tasks; design the framework of this project; coded part of preprocessing, all of Baseline model and CNN model; contributed on the report's structure, details, and insights(Discussion).

Name: Angelina Zhang

Contributed to preprocessing data, and writing reports for baseline models and cnn models with Grace. Participated in discussions and managed the GitHub repository.

Name: Grace(Yunjin) Zhu

Some code in baseline models, writing final report for baseline models and CNN models with Angelina

Name: Yulin Chen

Found the dataset.Some code in baseline models with Grace,writing final report for introduction,discussion and conclusion.

Name: Minghan Wu

Implemented cutting-out-faces and padding for preprocessing; Attempted to fine tune the models on SDSC; Worked on the SVM and Inception parts of this report; Improved the format and language of this report.

Name: Yue Yin

I contributed significantly to our project by leading the data preprocessing, ensuring the dataset was cleaned and prepared for analysis. I wrote the abstract, summarizing our research clearly, and compiled the citation section to credit all sources correctly. Additionally, I refined the final report by proofreading, enhancing clarity, and incorporating feedback