SKIN DISEASE CLASSIFICATION MINI PROJECT

Nguyen Hoang Minh Ngoc, Mehta Rishika, Verma Shireen
FDAA Team 6

PRACTICAL MOTIVATION

Early and accurate diagnosis of skin diseases can be challenging due to the wide variety of conditions and their similar visual manifestations.
Using a trained model can enable users to detect these conditions without having to get it checked constantly.

kaggle

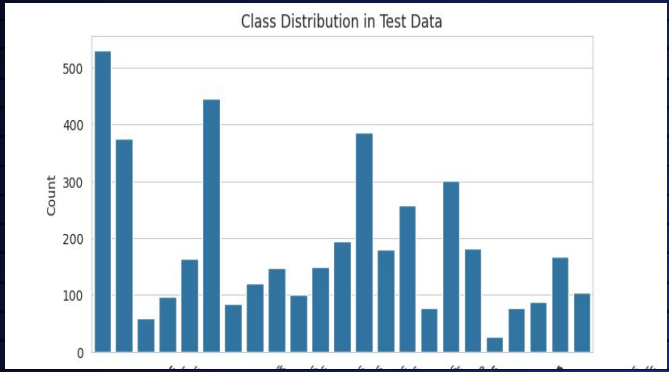
DATASET USED

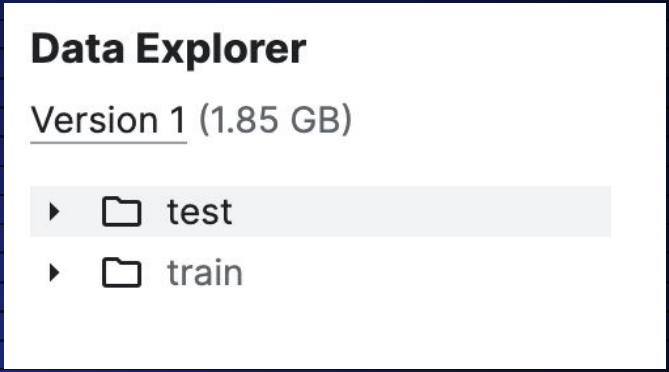
PROBLEM DEFINITION

Skin Disease Detection
Image classification of skin conditions
aiming to assist in automated detection
based on visual symptoms

01

DATA PREPARATION





DATA ANALYSIS

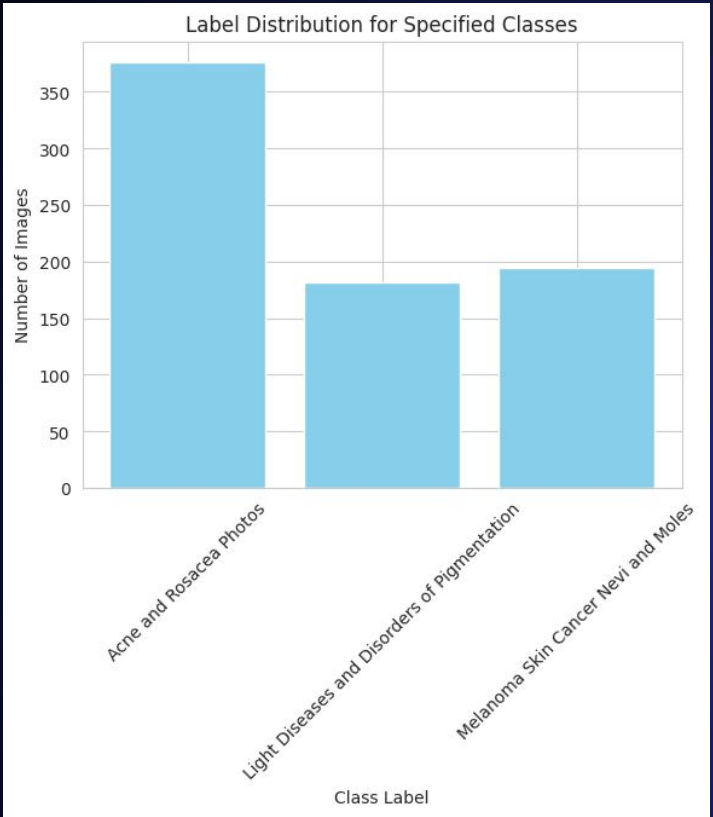
DATASET WITH 23 CLASSES

Taking samples of three classes to make a robust model for accurate detection of the three diseases given their image consistency

```
# Sample Images from specified classes
specified_classes = ['Acne and Rosacea Photos', 'Light Diseases and Disorders of Pigmentation', 'Melanoma Skin Cancer Nevi and Moles']
```

02

EXPLORATORY DATA ANALYSIS



Chosen classes:

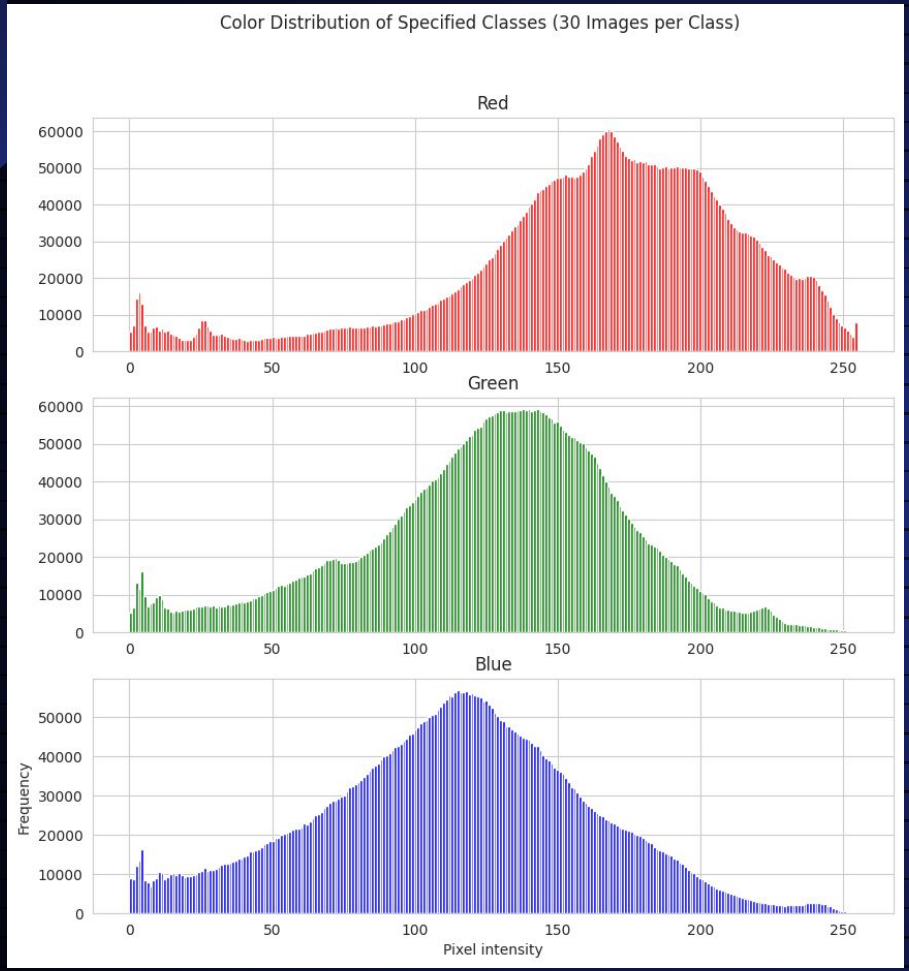
1. Acne and Rosacea
2. Pigmentation disorders
3. Melanoma Skin Cancer Nevi and Moles

Reason:

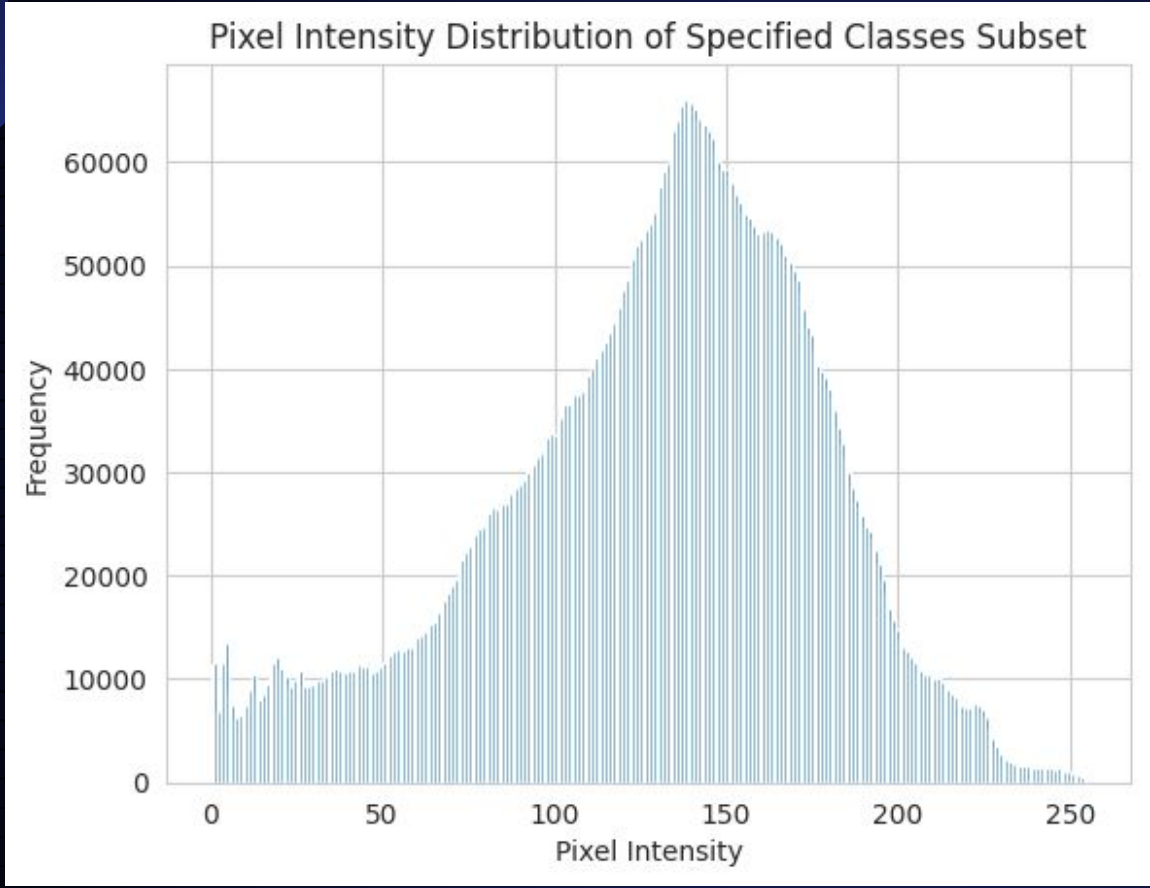
- Best class representation
- Consistent quality of images
- Variance within images (covers similar portions of the body)

```
Overall Mean of Pixel Values: 134.89185287328147
Overall Standard Deviation of Pixel Values: 37.83088909177029
```

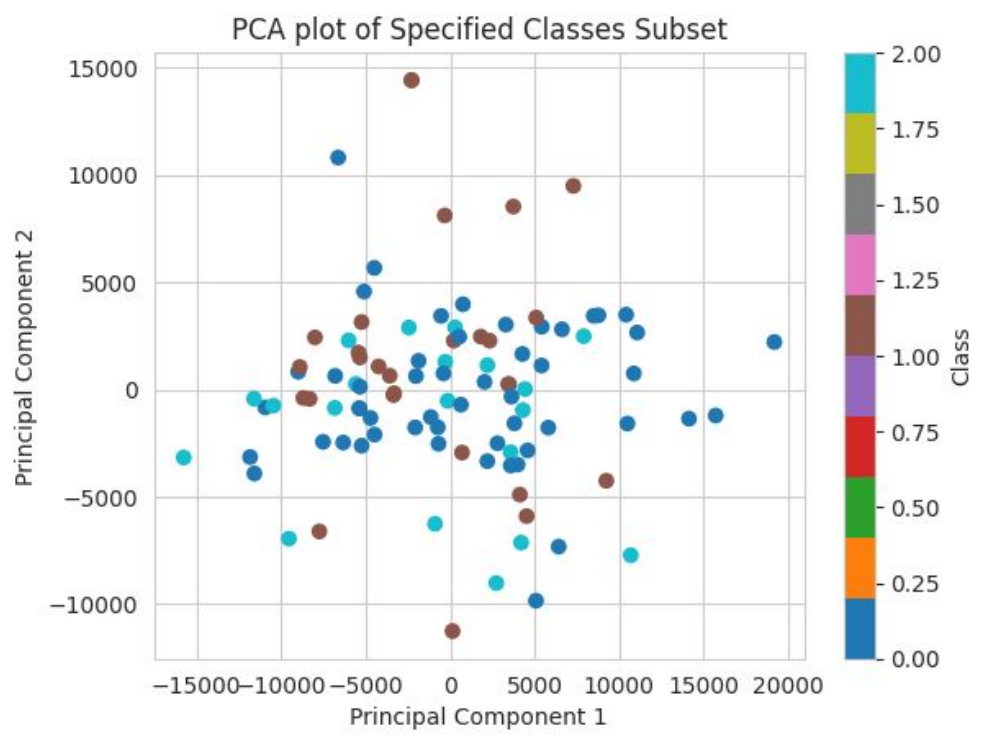
High Standard Deviation due to pixel values being spread out



- 30 images per class
- Conversion to RGB mode, and extraction of RGB pixel values.
- Separated into red, green and blue color channels
- The uniformity between the color distribution indicates common visual characteristics (lighting, camera settings, etc)



- Images from the specified classes and convert each image to grayscale
- Flatten the pixel intensities and append to list
- Plot pixel intensities using Matplotlib



- To reduce the dimensionality of the flattened image data to 2 dimensions
- Numerical labels to class names using LabelEncoder
- Each point an image in the reduced 2D space, colored by class label, providing a visualization of the distribution of images from the specified classes in the reduced feature space

03

MODEL ARCHITECTURE

KNN: K-NEAREST NEIGHBOURS

1.Data Preparation

Images are converted to feature vectors

2.Feature Extraction

Use of multiprocessing; the result is a result of feature vectors representing the images

3.Training Set

300 samples randomly selected, their features are used to train the KNN classifier

4.Testing Set

50 samples randomly selected

5.Model Training

Initialized KNN classifier with ‘k=5’ which considers 5 neighbours during classification.

6.Model Evaluation

Using the test data to make predictions and test accuracy

GAUSSIAN NAIVE BAYES

Algorithm Type

Probabilistic classifier based on Bayes' Theorem

Advantages

- Efficient and Scalable
- Simplified modeling process due to feature independence
- Straightforward interpretation

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

RANDOM FOREST CLASSIFIER

Key Features

An ensemble learning method that builds multiple decision trees independently, and uses extracted features for classification

Advantages

- Helps mitigate overfitting and improves performance
- Robustness to outliers
- Feature importance and accurate image preprocessing

CNN

```
valid_df, test_df = train_test_split(Ts_data, train_size = 0.5, shuffle = True, random_state = 123)
print(valid_df.shape)
print(test_df.shape)
```

```
(471, 2)
(471, 2)
```

```
Train = Tr_G.flow_from_dataframe(Tr_data, x_col = 'Paths', y_col = 'Labels', target_size = img_size, class_mode =
```

```
Found 3842 validated image filenames belonging to 3 classes.
```

```
Valid = Val_G.flow_from_dataframe(valid_df, x_col = 'Paths', y_col = 'Labels', target_size = img_size, class_mode
```

```
Found 471 validated image filenames belonging to 3 classes.
```

```
Test = Test_G.flow_from_dataframe(test_df, x_col = 'Paths', y_col = 'Labels', target_size = img_size, class_mode
```

```
Found 471 validated image filenames belonging to 3 classes.
```

```
L_index = Train.class_indices
L_index
```

```
{'Acne and Rosacea Photos': 0,
 'Light Diseases and Disorders of Pigmentation': 1,
 'Melanoma Skin Cancer Nevi and Moles': 2}
```

Training - Testing - Validating

CNN

```
CNN = Sequential([
    Conv2D(filters = 128, kernel_size = (3,3), padding = 'same', activation = 'elu', input_shape = img_shape),
    Conv2D(filters = 128, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    Conv2D(filters = 128, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    MaxPooling2D((2,2)),

    Conv2D(filters = 256, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    Conv2D(filters = 256, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    MaxPooling2D((2,2)),

    Conv2D(filters = 256, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    Conv2D(filters = 256, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    MaxPooling2D((2,2)),

    Conv2D(filters = 128, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    Conv2D(filters = 128, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    MaxPooling2D((2,2)),

    Conv2D(filters = 64, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    Conv2D(filters = 64, kernel_size = (3,3), padding = 'same', activation = 'elu'),
    MaxPooling2D((2,2)),

    Flatten(),

    Dense(256, activation = 'elu'),
    Dense(128, activation = 'elu'),
    Dense(64, activation = 'elu'),
    Dense(32, activation = 'elu'),
    Dense(counter_classes, activation = 'softmax')
])
```

Model architecture

CNN

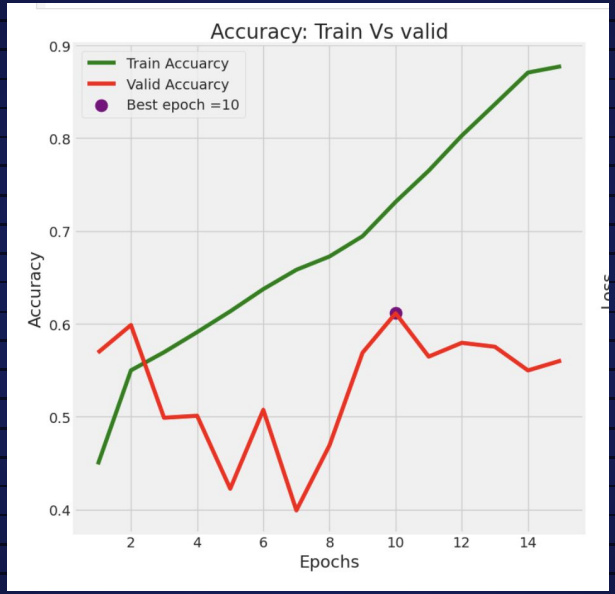
```
epochs = 15

history = CNN.fit(x = Train, epochs = epochs, verbose = 1, validation_data = Valid, shuffle = False)
```

```
Epoch 1/15
193/193 [==============================] - 1057s 5s/step - loss: 1.3046 - accuracy: 0.4487 - val_loss: 0.9633 - va
l_accuracy: 0.5690
Epoch 2/15
193/193 [==============================] - 139s 718ms/step - loss: 0.9545 - accuracy: 0.5500 - val_loss: 0.9500 -
val_accuracy: 0.5987
Epoch 3/15
193/193 [==============================] - 137s 711ms/step - loss: 0.9291 - accuracy: 0.5695 - val_loss: 1.0753 -
val_accuracy: 0.4989
```

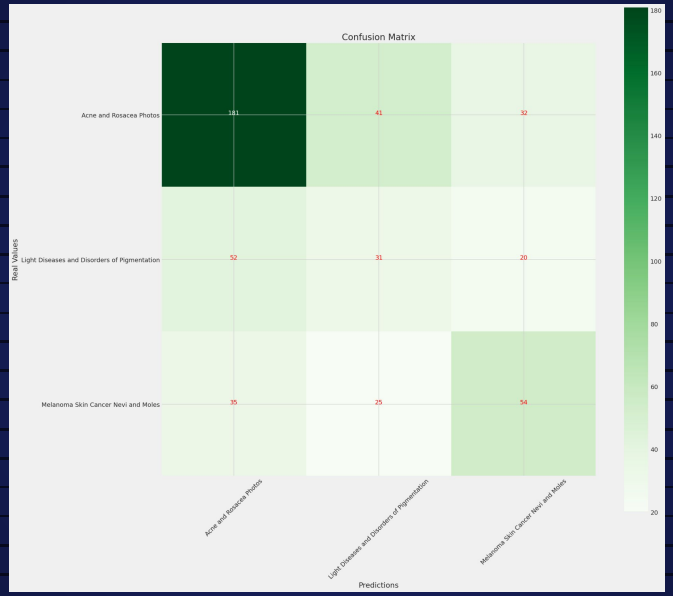
Training the model

CNN


Accuracy: Train Vs valid
Train Accuarcy
Valid Accuarcy
Best epoch =10
Accuracy
Epochs
0.9
0.8
0.7
0.6
0.5
0.4
2
4
6
8
10
12
14

Results

CNN



Confusion matrix

04

STATISTICAL INFERENCE

Models Accuracy

KNN

```
Accuracy: 0.49666666666666665
```

Random Forest Classifier

```
Accuracy: 0.5033333333333333
```

Naive Bayes

```
Accuracy: 0.44333333333333336
```

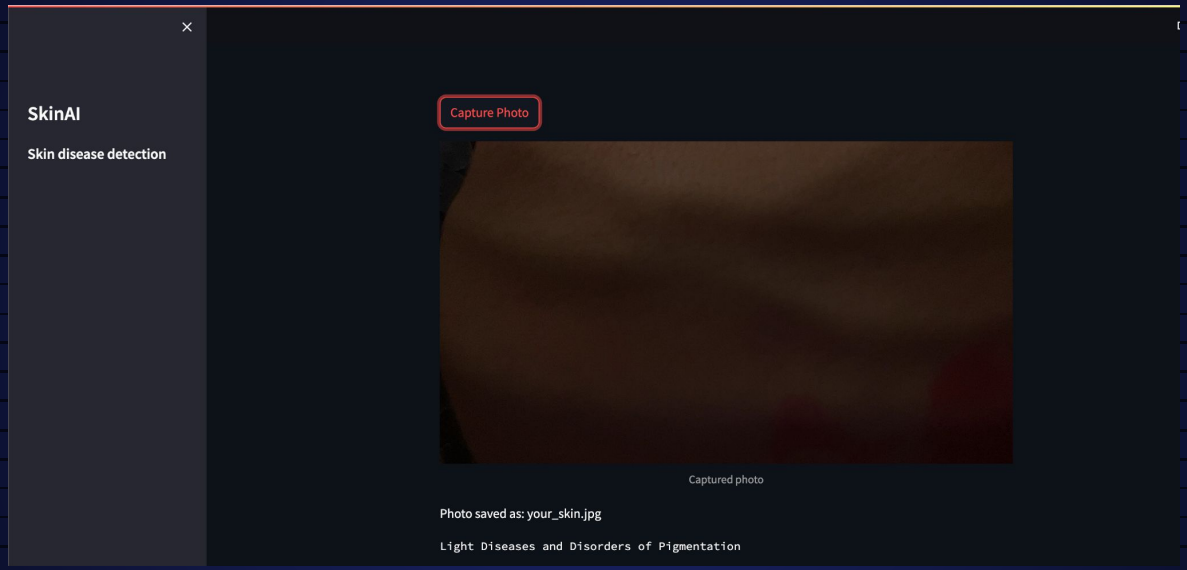
CNN

```
Test Scores :
    accuracy: 0.5647558569908142
      Loss:  1.3970633745193481
```

05

MODEL DEPLOYMENT

Streamlit



User Interface

06

KEY TAKEAWAYS

KEY TAKEAWAYS

OUTCOME OF THE PROJECT

- Skills: Preparing datasets, developing machine learning models and deploying models
- Topics: CNN, KNN, Naive Bayes and Random Forest Classifier

LIMITATIONS

- Lack of GPU to deal with large datasets
- Imbalanced datasets

FUTURE DIRECTIONS

- Applying the pre-trained models to enhance the accuracy
- Compiling more reliable datasets

RESOURCES

- Dataset: https://www.kaggle.com/datasets/umairshahab/dermnet-skin-diesease-images/data
- Deep learning model training: https://www.analyticsvidhya.com/blog/2022/07/building-a-brain-tumor-classifier-using-deep-learning/
- Machine learning models: https://www.analyticsvidhya.com/blog/2022/01/image-classification-using-machine-learning/
- Streamlit model deployment: https://blog.streamlit.io/deep-learning-apps-for-image-processing-made-easy-a-step-by-step-guide/

THANK YOU