Machine learning project report

Image Classification Using

ResNet50

and

Support Vector Machine

Introduction to Deep Learning

Professor: David Ruby

Quoc Dat Cao – 301550055

December 2, 2024

I. DATA PREPARATION

- There are 2562 image in the dataset belong to 31 people represent for 31 classes or labels. The number of images per person are varied. Each image has the dimension 160x160.
- Person with the largest number of images: Brad Pitt (120 images)
- Person with the smallest number of images: Kashyap (30 images)
- All the images were processed from the author, their faces was in the center.
- Faces are varied in angles and lighting.
- For each person's dataset, train-test split

  20% for testing

  80% for training
- Format: images in Dataset are in .jpg format
- Data was normalized using the ImageDataGenerator class with preprocessing_function=preprocess_input to match ResNet50 format
- Unknown or unlabeled classes were handled by introducing a threshold-based mechanism during testing.

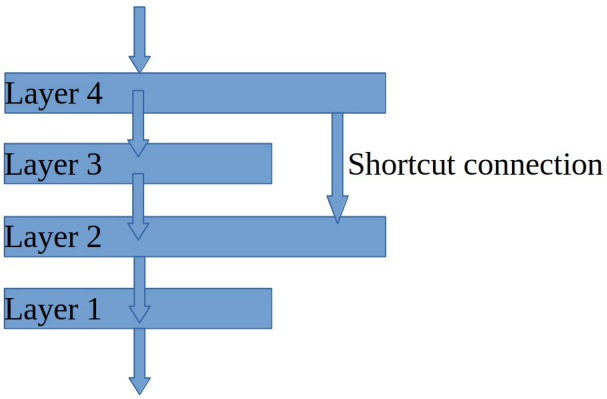
II. MODEL CREATION

1. Feature Extraction

ResNet50 is a deep convolutional neural network with 50 layers designed to handle feature extraction tasks. The pre-trained on ImageNet, is used to extract features from images. By using pre-trained weights, it reduces the training time and improve feature quality. The model does not use the fully connected (dense) layers at the last of the network because this model use Support Vector Machine classifier (svm) for the classification task. Fully connected layers in the default ResNet50 are specific to the ImageNet classification task (1,000 classes). Excluding the top layers leaves only the convolutional base, which outputs feature maps that are general-purpose. This allows me to use ResNet50 for tasks like feature extraction or fine-tuning with my own classifier.

What is Resnet?

Resnet is a CNN that use shortcut connection. The CNN cannot increase its accuracy just by adding more hidden. In Resnet architecture, the signal feeding into a layer is also added to the output of a layer located higher up the stack. We cannot stack too many layers on top each other when using CNN architecture because vanishing gradient occurs. So the idea of using shortcut connection allow the architecture to have many layers without vanishing gradient occur.



After features are extracted, ResNet50 outputs a 4D tensor (feature maps) of shape:(number of samples, height, width, channels). Traditional classifiers like SVM require 2D input, where each sample is represented as a single fixed-length vector. So the 4D tensor from ResNet50 needs to be flattened into a 2D matrix:(number of samples, 2048). By using Global Average Pooling method (GAP). GAP flattens the spatial dimensions (height and width) by computing the average of all pixel values in each channel. Thus, the output has one value per channel, which represents the channel's overall "activation".

$$\text{GAP}(c) = \frac{1}{H \times W} \sum_{h=1}^{H} \sum_{w=1}^{W} X(h, w, c)$$

For example:

$$\text{Feature Map} = \left[\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \right]$$

1. **Channel 1**:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

2. **Channel 2**:

$$\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

1. Compute the average of **Channel 1**:

$$\text{GAP}(1) = \frac{1+2+3+4}{2 \times 2} = \frac{10}{4} = 2.5$$

2. Compute the average of **Channel 2**:

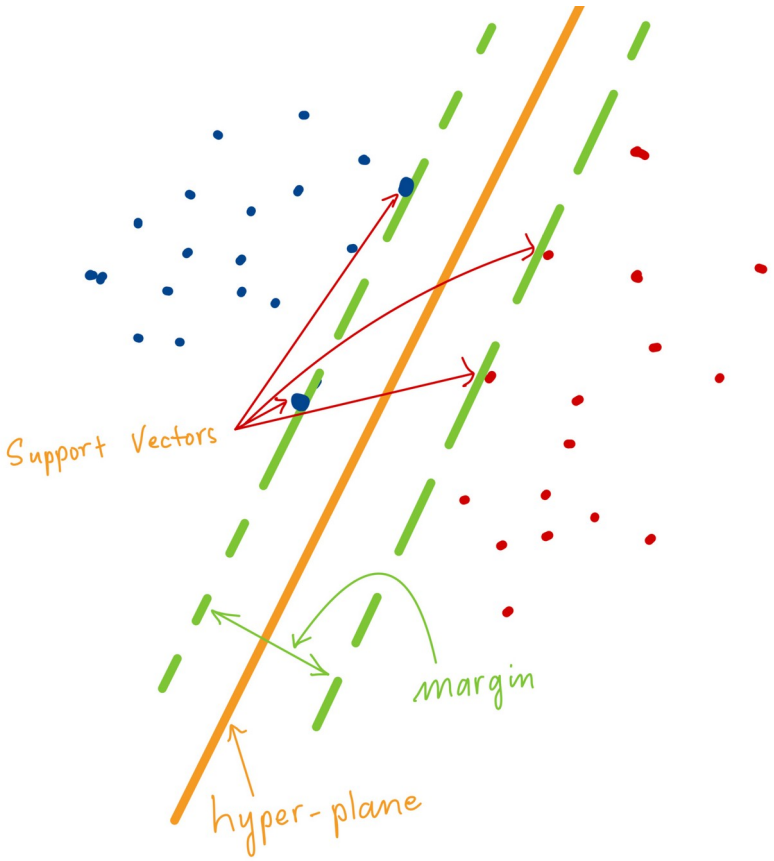
$$\text{GAP}(2) = \frac{5+6+7+8}{2 \times 2} = \frac{26}{4} = 6.5$$

2. Classifier: Support Vector Machine (SVM)

"A Support Vector Machine (SVM) is a very powerful and versatile Machine Learning model, capable of performing linear or nonlinear classification, regression, and even outlier detection. It is one of the most popular models in Machine Learning, and anyone interested in Machine Learning should have it in their toolbox. SVMs are particularly well suited for classification of complex but small- or medium-sized datasets."(Geron 145). The reason I use SVM for classification because it is powerful for a small dataset (~2500 image). It provides robust decision boundaries for high-dimensional features.

What is SVM?

SVM stand for Support Vector Machine. The goal of this classifier is that it tries to find the optimal hyperplane that maximizes the margin (the distance between the hyperplane and the nearest data points of any class) between different classes in a dataset. These points are called support vectors.



In my project, StandardScaler was used to normalize features before feeding them into the SVM. SVM with a linear kernel was trained to classify the extracted features. The classifier also used a threshold to detect unknown or unlabeled classes during testing, in this project, threshold = 0.5.

III. HYPER-PARAMETER EXPLORATION

My first experiment is to change the image size.

svm_classifier = make_pipeline(StandardScaler(), SVC(kernel='linear', C=1, degree=1, probability=True))

Image size	Accuracy
160x160	77.14%
224x224	81.90%
298x298	81.90%

The accuracy stops increasing when the image size increases. I see that 224x224 > 160x160, I think that upscaling typically involves interpolation, which fills in missing pixel values to enlarge

the image. While this process doesn’t add new information, it preserves existing spatial relationships, making the model more effective.

My second experiment is to keep image size 224. SVM performs as poly classifiers

svm_classifier = make_pipeline(StandardScaler(), SVC(kernel='poly', C=1, degree=x, probability=True))

x	Accuracy
2	55.43%
3	23.05%
4	11.05%

Let adjust the C value to in crease the accuracy

svm_classifier = make_pipeline(StandardScaler(), SVC(kernel='poly', C=1, degree=x, probability=True))

x	C	Accuracy
2	0.1	8.76%
2	0.3	23.24%
2	1.5	57.90%
2	2	62.67%
2	10	72.57%
2	20	67.24%

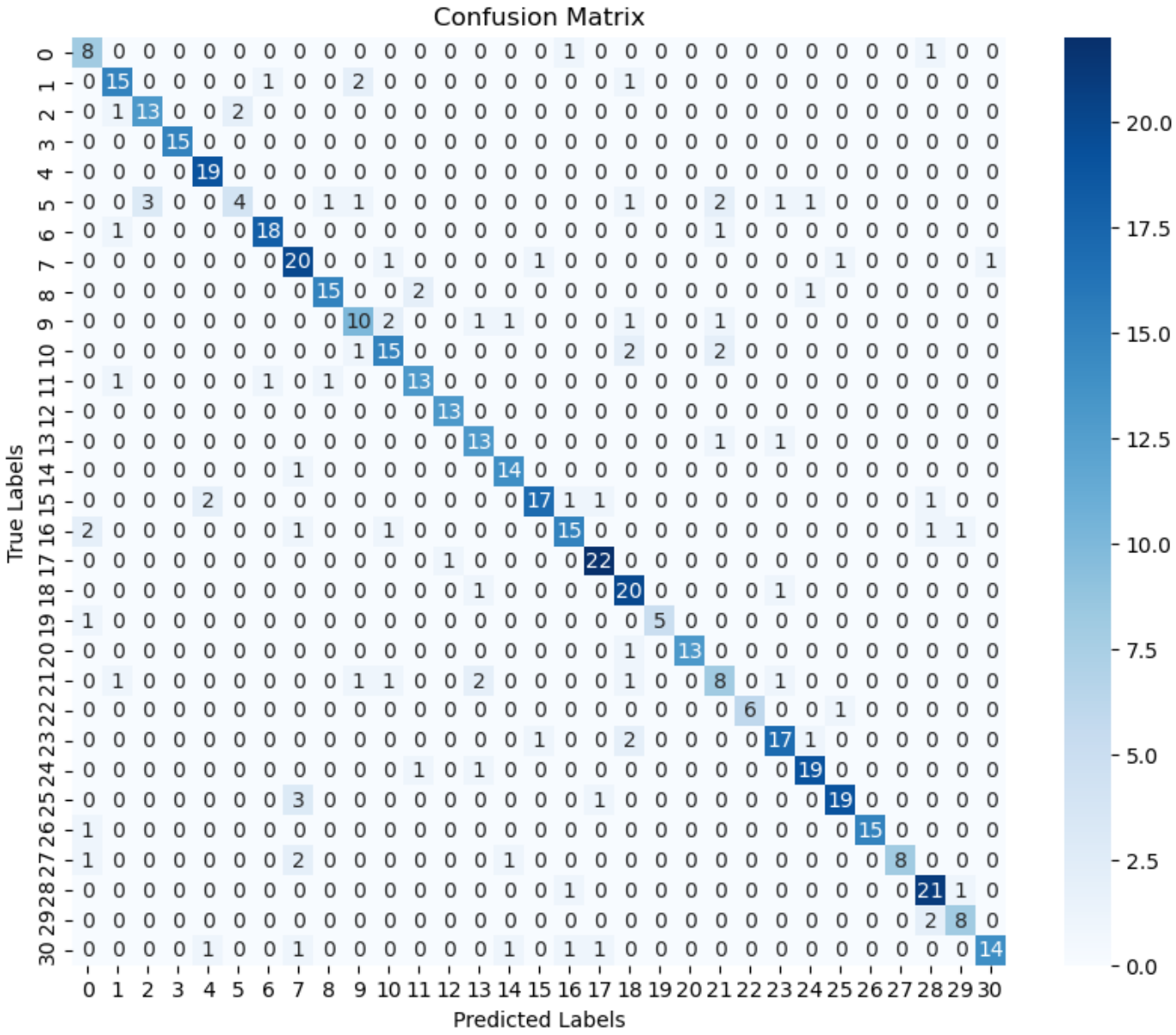
It is interesting that SVM polynomial gives lower accuracy at first. However, after increase C (regularization parameter), the accuracy increases in the case of degree=2. However, the accuracy decrease at C=20. I think that higher-degree polynomial kernels can model very complex decision boundaries, which may fit noise in the training data rather than generalizing to unseen data. This leads to poor performance on the test or validation set.

This is my final setting that get highest accuracy, around 82%.

svm_classifier = make_pipeline(StandardScaler(), SVC(kernel='linear', C=1, degree=1, probability=True))

Some visual results:

Confusion matrix

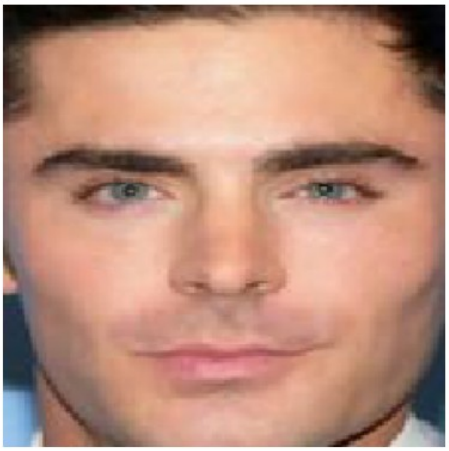


The person that the model classifies exactly:

Predicted: Billie Eilish

Predicted: Tom Cruise

Predicted: Zac Efron



This person does not have any data in training process, the label is unknown because the predict threshold lower than 50%

Conclusion

First of all, this project help me learn more about CNN and its variation like Resnet architecture. Beside, I am able to learn a new classifier SVM which is powerful to handle small database and also learn how to use many build-in tools libraries, especially from TensorFlow.

Beside, the limit of this model is that it is trained just by a small dataset, it is unclear how accuracy it will be and its performance when it processes a large dataset. Moreover, the model is sensitivity to hyper-parameters. SVM performance heavily depends on hyper-parameters like the regularization parameter C and kernel choice, thus, it causes impact when the hyper parameter Poorly chosen hyper-parameters can lead to underfitting or overfitting, and the hyperparameter search process can be computationally expensive.

On the hand, this model can be used for small companies that has small number of employees. It can be used as a system that takes attendance, recognizes strange people in a small group.

One thing that can improve the model is that handling the unbalance dataset. Using techniques like class weighting, oversampling, undersampling etc. to address class imbalance. Thus, Making sure that the model performs well across all classes, not just the dominant ones. And the second one is to experiment with more hyper-parameters. Systematically exploring a broader range of SVM hyper-parameters, such as C, kernel type, etc.

References

Géron, Aurélien. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017.

Prince, Simon J. D. *Understanding Deep Learning.* Draft, October 1, 2024. Licensed to MIT Press, 2023, https://udlbook.com.