Recognizing Roman Emperors By Hair Using Several Machine Learning Methods

Pauline Lettinga
University of Twente
BSc. Electrical Engineering
p.e.lettinga@student.utwente.nl

Abstract—**Paintings and sculptures of the ruler were everywhere in the Roman Empire. Many of these works have survived until today and are being identified manually by experts. Research into facial recognition of these emperors is quite effective but struggles with missing facial parts. As parts of the hair have higher survival rates, this paper looks into the recognition of Roman emperors by hair. Images of emperors are reduced to blocks of hair from which several features are extracted. These features are then used to compare several classification methods. These experiments showed the extracted features contain some information to identify the emperors, but more research should be done before recognition of Roman emperors by hair becomes viable.**

I. Introduction

From the Dutch House of Representatives to City Halls, you can find portraits of the Dutch king in many places. Those portraits and all other media available nowadays make it almost impossible for citizens not to know what the king, or the whole royal family, looks like. In ancient times with less media available, people would only know the appearance of rulers from portraits and sculptures, as empires such as the Roman Empire were too big for an emperor to show himself everywhere. These portraits and sculptures were everywhere, with standardized versions having signature looks per emperor [Ramesh et al., 2022]. Since many of these pieces have survived until now it is interesting to identify the emperors in them. One of the problems with this identification is the state of these works, with many having missing or damaged facial parts. Parts of the hair seem to have a higher survival rate than, for example, a nose. Thus, this paper will look into recognition of ROman emperors using hair and try to answer the research question: "To what extent can statues of Roman emperors be identified by hair using basic machine learning methods?". Before any experiments can be done, the data will go through some preprocessing. To limit the effect of damages to the statues, features related to the texture of the hair will be extracted and compared to see which gives the best results. These featues will then be used to test several classification methods in different experiments.

II. Related Work

A. Facial Recognition on Roman emperors

Research into the recognition of Roman emperors using facial recognition has already been done by van Klink and Ramesh et al. Sculptures might have scratches, broken-off parts or textures might be wildly different causing false identifications [van Klink, 2019]. Thus, the quality of the preserved work greatly impacts the performance of a classifier. With a large enough dataset, the most damaged pieces can be disregarded from the training set for higher performance. These works would still need to be identified manually in a real setting. In his work, van Klink compared the performance of existing facial recognition solutions on Roman emperors and normal modern faces, thus a smaller dataset did not pose much of a problem. While van Klink found that existing solutions could sufficiently recognize Roman emperors, the performance on normal human faces was quite a bit better. Most likely due to the classifiers being trained on normal faces and not sculptures causing it to be more effective on the former.

In the work of Ramesh et al, transfer learning is used to improve performance. With transfer learning, an existing network is retrained using a more relevant dataset. The network should then be better at dealing with the specifics of Roman sculptures like different textures, lighting and poses, damages and restorations, standardized versus non-standardized portraits and the varying skills of the sculptures [Ramesh et al., 2022]. In the research done by Ramesh et al, a network that classified into 1000 classes was retrained to fit the needed 10. A problem that arises with this method is that retraining requires a bigger dataset than the original dataset to be effective, thus the dataset was augmented to essentially make it bigger. By modifying the images e.g., adjusting brightness, flipping or rotating, the dataset was multiplied by six. This method was quite effective at recognizing Roman emperors. The problem of missing facial features did however remain. This makes it very interesting to look into the possibilities of hair recognition.

III. Method

The whole classification process has several steps from images of statues to the results. This pipeline is shown in Figure 1. The pipeline can be divided into three main parts, preprocessing, feature extraction and classification. These parts will be further explained in the following sections.

Before we can dive into the methodology, it is important to know what kind of data we are working with. The original dataset contains 140 pictures of statues from 29 different Roman emperors, with a varying number of pictures per

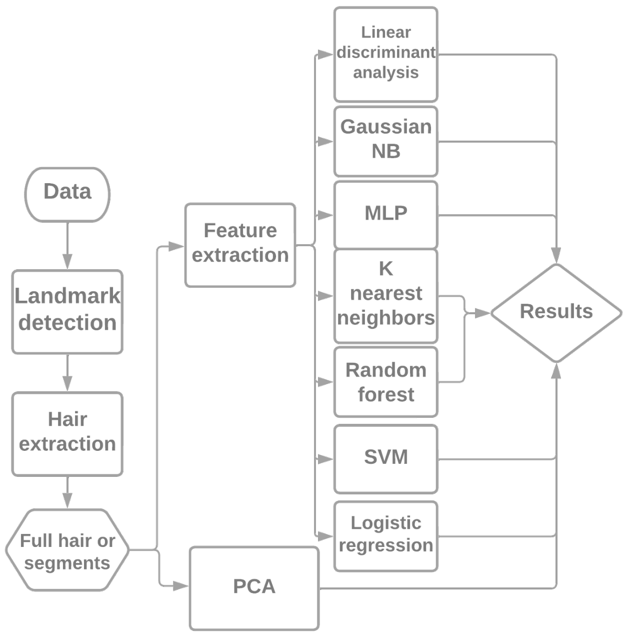


Fig. 1: Pipeline of Classification Process

emperor. All pictures are taken from a frontal perspective, making sure the faces are fully visible.

A. Preprocessing

1) Landmark Detection: While this paper is about classification using hair types, knowing where the hair is in the image is done using facial landmark detection. Specifically, using the face detector from DLIB [King, 2009]. This detector uses a Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid and a sliding window detection scheme to create a facial pose of 68 landmarks [King, 2009]. The faces are then resized and aligned, making sure roughly the same happens for the hair.

2) Hair Extraction and Segmentation: A standard region above the face which contains hair in almost, if not all images, is then selected for the rest of the research. It is important to check the images after this step, to make sure all images contain a block of hair of the same size and thus information. Since the dataset is not very large, the chosen blocks can be split into several smaller blocks, as long as the blocks remain large enough to contain the shapes of the texture in the hair.

B. Feature extraction

Information about the images is stored in a feature vector. At this point, the images are grayscale blocks. Thus, the interesting information to look at are the textures. Both Local Binary Pattern (LBP) and Histogram of Oriented Gradients (HOG) will create a feature vector based on the textures.

1) Local Binary Pattern (LBP): LBP detects texture by comparing a pixel with its surroundings. These surrounding pixels then get a binary label. When all these labels are the same, the area is flat. Different labels next to each other means the pattern is non-uniform showing there is texture. Several LBP's are measured over the whole image. The distribution creates the feature vector for texture [scikit image, 2022b].

2) Histogram of Oriented Gradients (HOG): The HOG descriptor is used a lot for object detection, but the different gradients also show texture. The HOG is made in several stages making it robust and invariant to illumination, shadowing and edge contrast. This will be very helpful, especially with a dataset that has great differences in these parts like the pictures of Roman emperors, this will be very helpful. Like LBP, HOG divides the image into smaller parts to compute local histograms. The local histograms are again normalised and put together in the feature vector [scikit image, 2022a].

3) Gabor filter: Another commonly used method in texture analysis is the use of a Gabor filter. These filters are band-pass filters that can have a certain direction [Shah, 2018]. The frequencies and angles can be changed in several filter banks to create different responses for different textures. The power of these responses can be combined into a feature vector.

4) Principal Component Analysis (PCA): Instead of a feature vector, PCA takes the images of a train set and reduces it to the most crucial information. With Deep Learning a model can then be made that captures a specified variance. PCA essentially lowers the dimensionality of the data. This speeds up the Deep Learning process and achieves a higher accuracy than using original training data [Sharma, 2020].

C. Classification methods

Using the feature vector of either LBP or HOG a classification method can be fitted and used to classify the emperors. The following classifiers were used.

1) Logistic Regression (LRlib): Given the data points, this algorithm makes a regression line. The values are clipped at zero and one and follow a sigmoid curve. New values are then compared to the curve to form the classification. While it works best when the data is linearly separable, a big advantage of logistic regression is that it does not require many computational resources [Banoula, 2022].

2) Linear Discriminant Analysis (LDA): LDA is most often used for feature extraction in pattern classification problems. The method performs best for binary classification, but still handles multiple classification problems quite well. Especially with flexible boundaries. LDA projects data points on a line to maximize the scatter between classes and minimize them within a class [Dash, 2021].

3) K Nearest Neighbour (KNN): Instead of setting boundaries, K nearest neighbour looks at the surrounding data points to classify. Most often, K is set to three or five, meaning it will look at the classification of that number of neighbours to decide the new classification [Simplilearn, 2022].

4) Random Forest (RFOR): A decision tree makes advantages splits in a dataset to determine a classification. This process is quite simple, yet prone to overfitting and they can be quite unstable. The benefits of a decision tree can be used with fewer disadvantages by combining the outcome of several singular trees, creating a random forest [IBM, 2020].

5) Multi-layer Perceptron (MLP): MLP consists of multiple (hidden) layers of neurons. When features pass through this network, the neurons transform the values with a linear summation and a non-linear activation function. They also add weights to the data. The algorithm uses backpropagation to fit to the training data [scikit learn, 2022b].

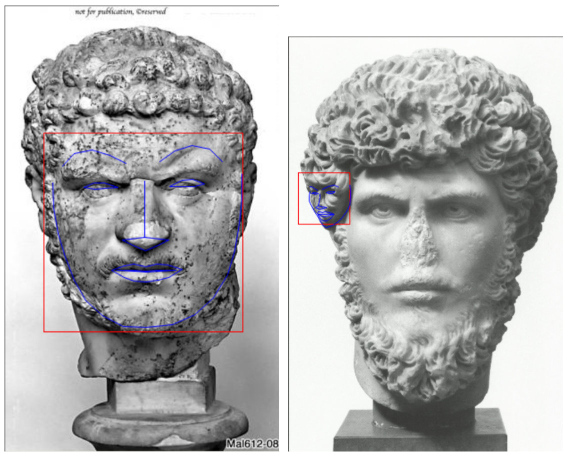
6) Gaussian Naive Bayes (NB): Naive Bayes assumes all features are independent. Furthermore Gaussian NB all classes follow a Gaussian distribution. A big difference with LDA is that the decision boundary is not linear [Hrouda-Rasmussen, 2021].

7) Support Vector Machine (SVM): Last but not least, the SVM algorithm aims to find hyperplanes in a N-dimensional space that classify the data. These planes should have maximum distance between data points of different classes [Gandhi, 2018].

IV. Experiments and Results

A. Preprocessing

Not all original 140 images of 29 emperors were suitable for the preprocessing steps. Figure 2 shows the facial landmarks working correctly for Figure 2a, but identifying a face in the hair of Figure 2b.

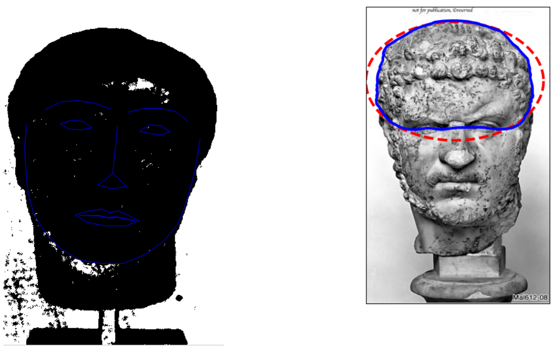


(a) Caracella (b) Lucius Verus

Fig. 2: Facial landmarks on two images

An attempt was made to extract the hair completely automatically using the difference in entropy between the hair and face and by the 'activecontour' function in Python. This showed that the statues have too many imperfections in the facial parts of the emperors that the distinction with the hair was nearly impossible to make. This can be seen in Figure 3b.

After all preprocessing steps, the images are reduced to a 20 by 80 pixels block. Images the steps did work on correctly were removed from the dataset. For some emperors, this meant only one or two pictures remained. To improve the chances of the classification methods, emperors with less than 4 pictures were removed. This meant 105 images of 13 Roman emperors remained for the next parts of the classification process.

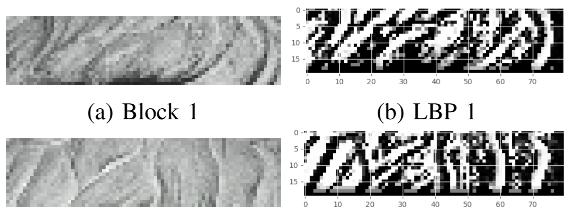


(a) Binary Entropy (b) Active Contour

Fig. 3: Other hair extraction techniques

B. Features

1) LBP: Figure 4 shows two images of different statues of the same emperor Augustus. Next to the hair, the corresponding LBP's are displayed. While the lower block seems to have thicker strands of hair, the LBP shows thinner lines as it has no difference between large or small texture changes. In both cases, the directions of the hair are visible.

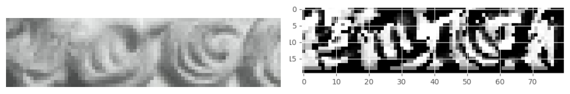


(a) Block 1 (b) LBP 1

(c) Block 2 (d) LBP 2

Fig. 4: Hair of Augustus with LBP

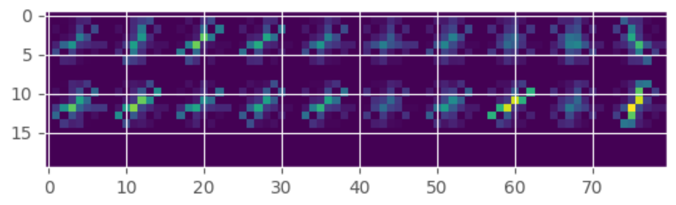
The hair of emperor Hadrian is shown in Figure 5. This hair type has more curls causing the LBP to show different lines. The thickness of the lines is comparable.

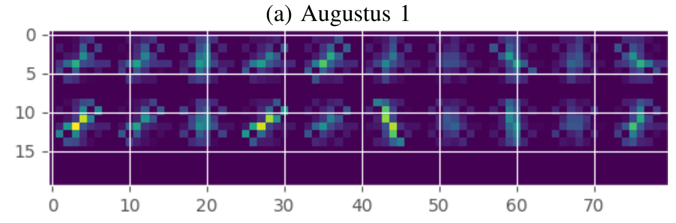


(a) Hair of Hadrian (b) LBP

Fig. 5: Hair of Hadrian with LBP

2) HOG: To get a comparison, the HOG images of the same blocks as in Figure 4 are shown in Figure 6. Both images of Augustus have clear directions with some slight deviations. Contrarily, the image of Hadrian has no clear directions, due to the curls. The intensity of the lines corresponds to the histogram values.


(a) Augustus 1


(b) Augustus 2

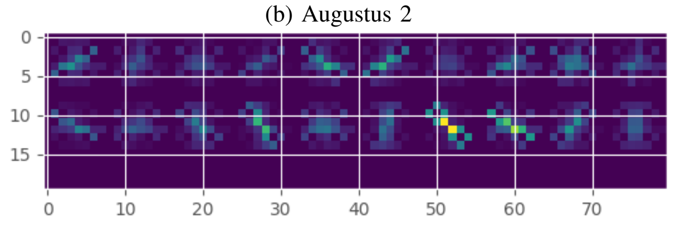

(c) Hadrian

Fig. 6: HOG of the same blocks of hair

3) Gabor filter: The representations of LBP and HOG both show a clear resemblance with the original image. For the response to different Gabor filter kernels, this is not the case. Figure 7 shows the different kernels on the left with the responses of the same first picture of Augustus on the right. The three different directions of the kernels can be seen in the Figure. The four frequencies for each direction are from top to bottom 20, 10, 6,67 and 5 pixels.
As these images are responses to a filter, the lighter parts are the parts that pass through. This shows for this particular image the highest power comes from the biggest filters. As the hair is pretty vertical, the vertical kernel also has the highest power.

4) PCA: As has been mentioned in the method section, PCA gets the components from the images containing the most information. Using the first two of these components, the plot in Figure 8 is made. Ideally, all dots with the same colour and thus label will be grouped. Clear groupings can however not be found.

C. Classification methods

To see how well the classification of Roman Emperors using their hair types works, several experiments are done with the feature vectors. Using Python, all mentioned methods can be tested at once. To optimally use the training data, k-fold cross-validation is used. This eliminates the need for an evaluation set and instead splits the training set into smaller sets. For each fold, another part is used as the validation set [scikit learn, 2022a]. The results are then shown in a boxplot. For all boxplots holds that the box contains a line for the median and extends from the first to the third quartile values

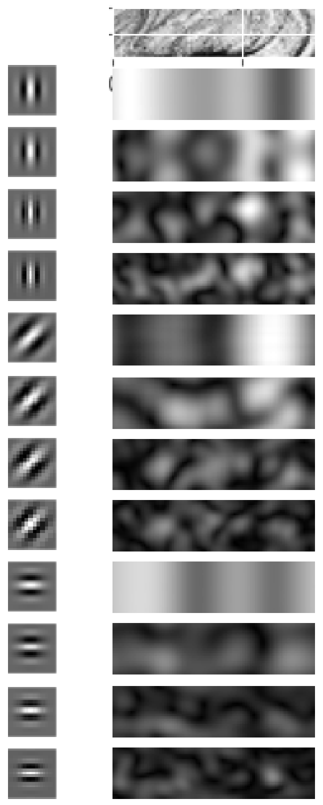


Fig. 7: Gabor filter

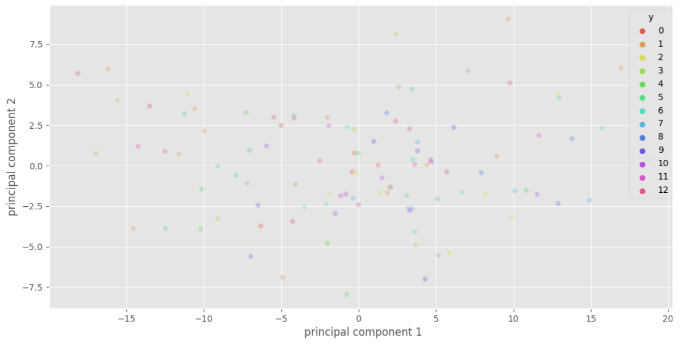
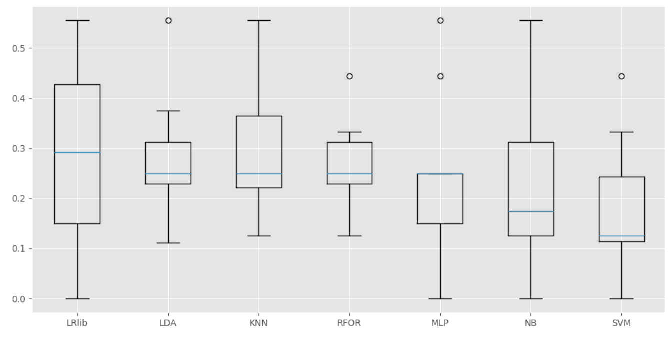


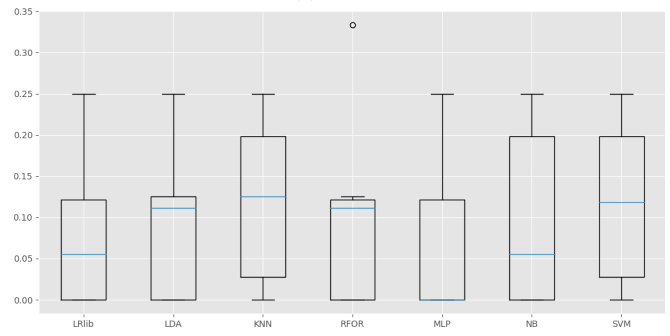
Fig. 8: Visualization of the dataset using PCA

of the cross-validation score. The whiskers contain 1.5 times the inter-quartile range (IQR) with dots representing outliers [Matplotlib, 2022]. Important to note is that the scaling of the boxplots can be different.

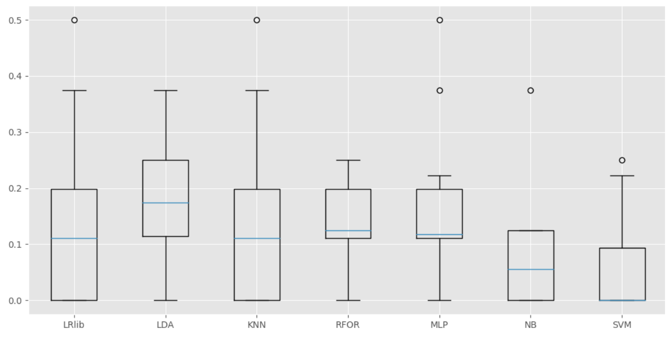
1) Classification with different feature vectors: The previous section has described several ways to extract features from the images. To test what feature works best to classify the Roman emperors they are tested with different classification methods. The boxplots are shown in Figure 9. With 13 different classes, a complete random classification would have an accuracy of about 0.08. Looking at HOG the mean accuracies lie around 0.20 and the highest mean accuracies for LBP are around 0.12. Gabor works well with LDA with a mean accuracy of 0.17, but other methods are closer to 0.10.



(a) HOG



(b) LBP



(c) Gabor filter

Fig. 9: Boxplots using different features

Interesting observations to note are the whiskers of LBP almost all going up to 0.25 and the high outliers of the Gabor filter.

PCA did not generate a boxplot. Instead, it went through 20 epochs of deep learning. It heavily overfitted on the training data. The highest accuracy on the validation set was achieved after 13 epochs. The accuracy was 0.23, with a training accuracy of 0.86. As HOG shows the most promise, only the HOG feature is used for the rest of the experiments.

2) Groups with similar hair type: The dataset used has many emperors, but only a few pictures per emperor. Looking at the hair types they can roughly be divided into three categories: straight, curly and something in between. The boxplot of this experiment can be found in Figure 10. With three labels, the accuracy automatically goes up to 0.33. The mean accuracy with SVM is the highest with slightly over 0.6. The whiskers of LRlib and SVM almost go up to 0.9.

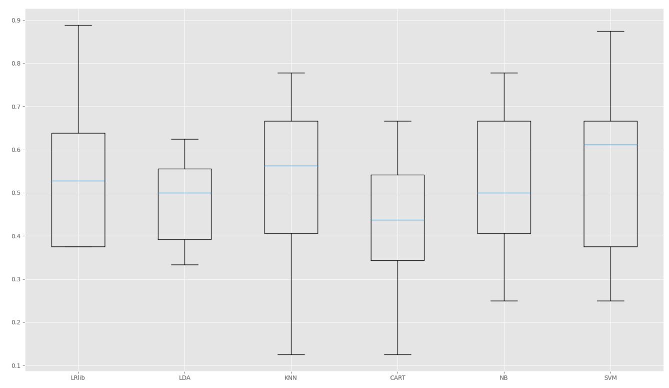


Fig. 10: Boxplot using three classes

3) Smaller blocks: Another way to get more pictures per class is by dividing each block into smaller blocks. To keep large curl patterns intact, the blocks were split into 4 equally sized smaller blocks. Figure 11 shows this gives slightly lower accuracies than the normal blocks. Random Forest and Naive Bayes do score higher.

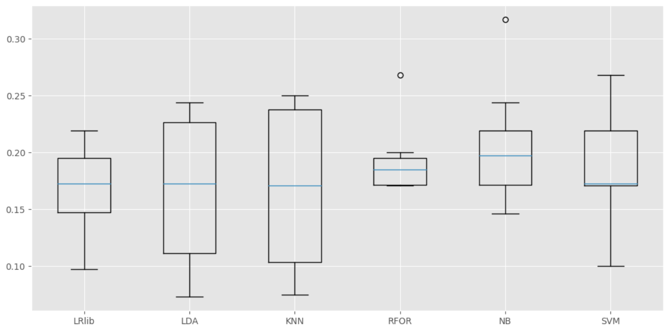


Fig. 11: Boxplot with smaller blocks

V. Discussion

This paper aimed to answer the main question: ”To what extent can statues of Roman emperors be identified by hair using basic machine learning methods?” Looking at the accuracies of the different experiments the outcomes are suboptimal. While experts can identify emperors by eye quite accurately, the different classification methods are only right in a fifth of the cases with 13 emperors, and two out of three times with emperors divided into three classes of hair types. Meanwhile, the research on facial recognition on Roman emperors reached accuracies of 81.1% and 89.2% using 9 emperors and a non-emperor class [Ramesh et al., 2022]. The dataset used in that research was bigger, but very similar.
Looking at the confusion matrices in Figure 12 of KNN and SVM gives some insight into the shortcoming of the classification. The number of pictures per emperor is not uniform in the dataset, the biggest outlier is Augustus (label 1) with 17 pictures. This causes the algorithms to have a bias towards these outliers. For the remaining emperors, this also means many of them only have five or six pictures which is

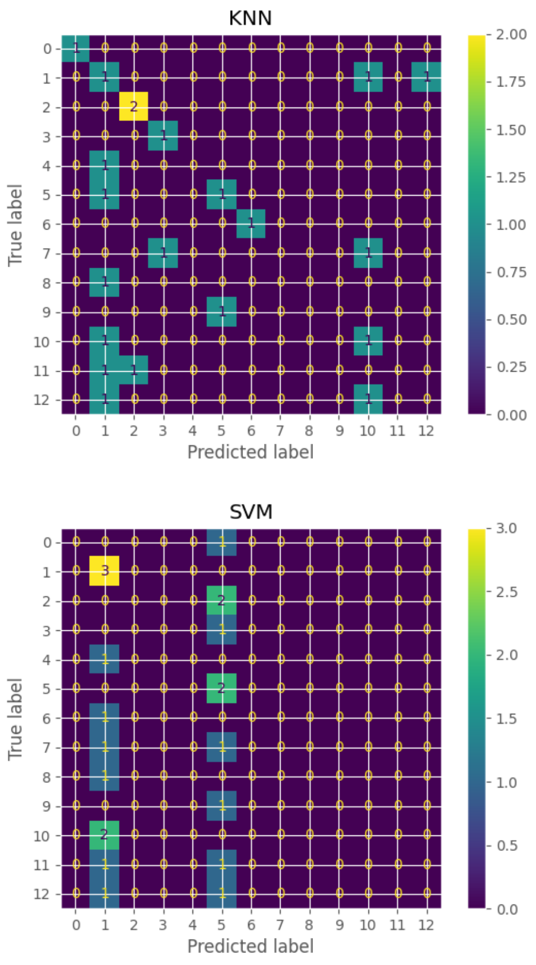


Fig. 12: Confusion matrices for KNN and SVM

very little to fit a model on. That being said, the classification methods did give results indicating there is some recognition going on.

VI. Future Research

To improve these results, a couple of things could still be tried. The different features are all texture based, but all implement it differently. Combining these features before classifying could give the model more data to make distinctions.
Another experiment that could be expanded is augmenting the data to essentially create more. Cutting the data into smaller blocks did slightly lower the results in most cases however, more research could be done on the outcomes of the classifications of these blocks. Perhaps smartly combining these results could improve the overall outcome.
In the preprocessing steps, there might also be some room for improvement. Using landmark detection has caused some otherwise fine pictures to be discarded from the dataset. Either finding a different way to get these blocks or creating them by hand would save some of the images. Especially in cases where facial recognition, which otherwise works better, has it's limitations.

VII. Conclusion

The aim of this paper was to recognize pictures of statues of Roman emperors by their hair. After some preprocessing, blocks of hair were extracted. From these blocks features mostly regarding texture were created using Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP), Gabor filter and Principal Component Analysis (PCA). Except for PCA, the features clearly resembled the original image, indicating some information was gathered. After classification, results were significantly better than random guessing however, paled in comparison to existing facial recognition research. The research done in this paper could still be a nice stepping stone to further research into combining more features and possibly making more use of deep learning methods.

References

[Banoula, 2022] Banoula, M. (2022). An introduction to logistic regression in python.

[Dash, 2021] Dash, S. K. (2021). A brief introduction to linear discriminant analysis. Accessed:14-11-2022.

[Gandhi, 2018] Gandhi, R. (2018). Support vector machine — introduction to machine learning algorithms. Accessed:14-11-2022.

[Hrouda-Rasmussen, 2021] Hrouda-Rasmussen, S. (2021). (gaussian) naive bayes. Accessed:14-11-2022.

[IBM, 2020] IBM (2020). Random forest. Accessed:14-11-2022.

[King, 2009] King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758.

[Matplotlib, 2022] Matplotlib (2022). Matplotlib.pyplot.boxplot - matplotlib 3.6.2 documentation.

[Ramesh et al., 2022] Ramesh, D. S., Heijnen, S., Hekster, O., Spreeuwers, L., and de Wit, F. (2022). Facial recognition as a tool to identify roman emperors: towards a new methodology. *Humanities and Social Sciences Communications 2022 9:1*, 9:1–10.

[scikit image, 2022a] scikit image (2022a). Histogram of oriented gradients - skimage vo.20.0.dev0 docs. Accessed:10-11-2022.

[scikit image, 2022b] scikit image (2022b). Local binary pattern for texture classification - skimage v0.19.2 docs. Accessed:10-11-2022.

[scikit learn, 2022a] scikit learn (2022a). Cross-validation: evaluating estimator performanc. Accessed:10-11-2022.

[scikit learn, 2022b] scikit learn (2022b). Neural network models (supervised). Accessed:14-11-2022.

[Shah, 2018] Shah, A. (2018). Through the eyes of gabor filter. Accessed:14-11-2022.

[Sharma, 2020] Sharma, A. (2020). Principal component analysis (pca) in python tutorial. Accessed:10-11-2022.

[Simplilearn, 2022] Simplilearn (2022). How to leverage knn algorithm in machine learning? Accessed:14-11-2022.

[van Klink, 2019] van Klink, K. (2019). Detection and identification of roman emperors using facial recognition.