Applied machine learning system ELEC0134 22/23 report
SN: 17062315

1 Abstract

Artificial Intelligence is evolving faster than ever. One domain in particular, computer vision has benefited from a vast amount of data annotated in the past decades and many recent breakthroughs on the software level.

It is now time to develop our own tool built on these discoveries. Its purpose is to tackle challenges in the face recognition area by analysing multiple features of the human face such as its shape, eyes and mouth which are essential to identify individuals and their emotions.

To reach this objective, we rely on annotated data in the form of pictures of celebrities and cartoonish characters.

The information from these datasets are then extracted and cleaned before being trained by various algorithms such as Support Vector Machine, Random Forests, K-nearest neighbour or bagging. The first two represent good alternatives to more complex algorithms such as neural networks and deep learning which imitate the way human learn in order to make the algorithm able to differentiate individuals.

Results are obtained by testing more than 20 variations of models against a subset of the dataset left aside for this specific purpose.

Upon completion of this project, it should be possible to take these proofs of concepts and embed them in more complex systems such as smart cameras and robots which are the most common kind of devices which may benefit from an improved perception.

2 Introduction

Last century, science fiction authors such as Isaac Asimov predicted the rise of intelligent devices [1] [2] and now, the dawn of artificial intelligence is upon us. After, the Tay bot [3], it is now the turn of ChatGPT to earn an immense popularity among tech enthusiasts with its ability to provide detailed answers to any question [4] [5]. And to add to this triumph, Microsoft has shown interest in acquiring OpenAI, the creator of ChatGPT and many other AIs [6].

But while everyone has their eyes on this new chatbot, a more discrete, albeit, just as interesting revolution, has been ongoing for a few decades already in the computer vision domain.

The first visible step toward it was the introduction of captcha [7] in 1997. These challenge-response tests quickly became annotation tools channelling the efforts of humans to train artificial intelligences [8].

Applications are numerous, ranging from biometric tests to disease detection [9] and even robotics with the latest self driving cars [10].

In this report, we shall pay close attention to a specific use case: face recognition. After mentioning the latests relevant breakthroughs, we shall see how we can design softwares able to recognise the gender, emotions, face shape and eye colour of various celebrities and cartoonish characters.

3 Literature survey

Face recognition has been a major topic recently with new algorithms and tools appearing in quick successions as early as last century with, for instance, the creation of Support Vector Machines in 1992, algorithms shining for their adaptability and generalisation ability [11] [12] or OpenCV created in 2000 and which is a library still widely used in many languages such as C++ and Python [13].

Many algorithms were used over the years such as Support Vector Machine and K-nearest neighbour [14]. They all come with their advantages and trade-offs. Convolutional neural networks especially tend to be studied with great attention because of their ability to imitate the human mind which has been AI's goal since its inception [15] [16].

Since various algorithms exist, the main objective nowadays shifted from pure scientific research to engineered solutions to deal with edge cases where computer vision still struggles [17].

Firstly, faces become harder to analyse with low resolution images and and dim lighting which make data extraction less precise [18].

Moreover noise is more important with data obtained from pictures of people belonging to specific demographic subgroups such as old males, people with darker skin types [19] or children which tend to appear as androgynous since gender only becomes more apparent past puberty [20].

These problems can generally be also explained by biased datasets which contain unbalanced representation of said demographic subgroups [21].

It is therefore essential to handle data preprocessing with as much care as the analysis itself as we will see in the next sections.

4 Models Description

CNN are considered as the most efficient model overall out of the one we studied. Which is why we should not rely on them in this report as it is more interesting to not know how a model fares. We will therefore study other popular models including SVMs, Bagging, KNN (K nearest neighbours) and Random Forests and determine which is the best among them for each task.

4.1 Support Vector Machines

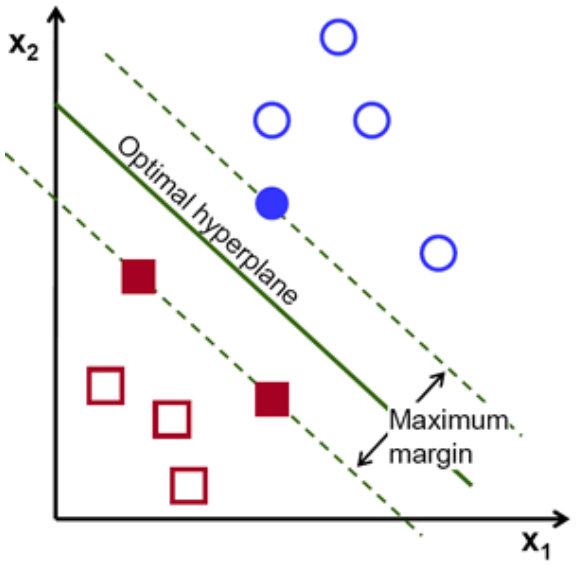

Figure 1: SVM algorithm

The first model, SVM, is very robust for both classification and regression in supervised learning tasks. Their purpose is to find a hyperplane that best divides a dataset into two classes.

For instance, with a linear SVM, we seek w and b such that $\rho = \frac{2}{\|w\|}$ is maximised and for all $(x_i, y_i), i = 1...n : y_i(w^T x_i + b) \geq 1$.

This leads to a quadratic problem where we want $\phi(w) = w^T w$ minimised for the aforementioned values of x and y.

The solution is a dual problem with a Lagrange multiplier α_i summarised as

$f(x) = \sum \alpha_i y_i x_i^T x + b$ with every non-zero a_i indicating the presence of a support vector in the form of x_i.

One of the most important benefits of SVM is their ability work both on linear and multiclass problems with a kernel which can be fine tuned to deal with a specific problem. Most of the time, a standard Polynomial or RBF kernel for multi class problems.

4.2 Random Forest

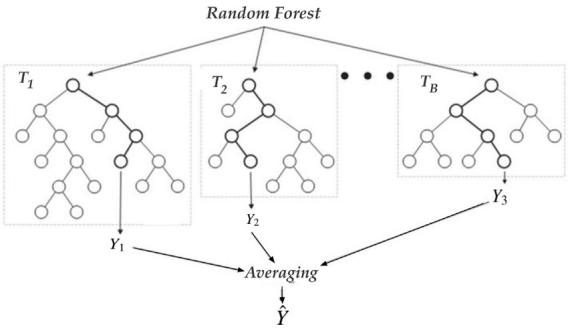

Figure 2: Random forest algorithm

Another model are decision trees which consist in putting if conditions in cascade to find a solution. It relies on the theory of entropy $H(x) = -\sum_{x \in S_x} p_x(x) log p_x(x)$ which is a measure of disorder obtained when many parameters shape a solution.

But random trees tend to converge slowly which is why fancier alternatives such as bagging or random forests are used. In the case of Random Forests, a number of decision trees is created based on bootstrapped training samples.

Each time a split in a tree is considered, a random sample of q attributes is chosen as split candidates from the full set of p attributes as its optimal value is data dependant. The branches are averaged at the end.

4.3 Bagging

When decision trees fail because of overfitting, we can instead rely on ensemble methods such as bagging which can help by decreasing variance of the base model without influencing bias at the cost of a loss of simple structures in comparison to the original model.

In the case of bagging, we pick a base learning model and apply it over many samples. Classification is done by majority vote and also uses a training set consisting of many examples for various models used to get an aggregate overall model.

4.4 KNN

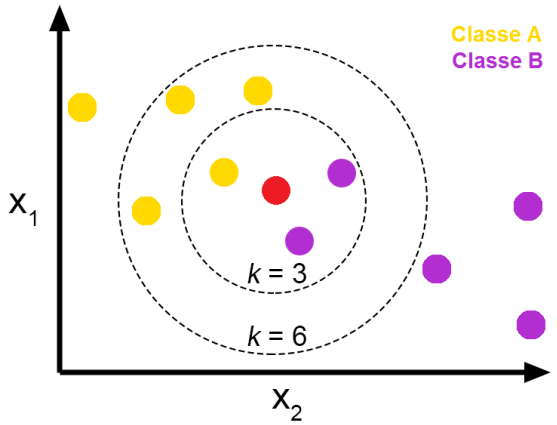

Figure 3: KNN algorithm

Finally, K-nearest neighbour algorithms rely on the entire training set. It relies on a metric ρ that returns a distance between two points such as the Euclidean distance $\sqrt{\sum_{k=1}^{d} (x_k - x'_k)^2}$. It is relatively simple and will always converge at the expense of performance since it needs to solve a very expensive distance problem.

5 Implementation

Because of the decision taken to compare 4 famous models, the code displays a decoupled

nature so that every task can call any model. This leads to a very simple main file similar to the simplified version shown below:

```
task = task_class.task
various_data = task.main_properties

X, y = extract_features_labels(...)
models, results = multi_train_test(...)
plot_data(...)
```

Every task class hold information essential to find the dataset associated to it. With them, it is possible to start extracting features by taking each image from the appropriate dataset directory, detecting its landmarks by greying the image and using 68 data points from a predictor.

Also, if a binary model is used with values equal to -1 or 1, it is essential to normalise them to 0 or 1 in prevision of the training which relies on positive data to avoid bias.

```
def extract_features_labels(...):
    ...
    for imgage_index ...:
    img = image.img_to_array(...)

    features, _ = run_dlib_shape(img)

    if req_label in ["gender", "smile"
    lbl = (np.array(all_labels) + 1)/2
```

The next part of the code is the split_train_test() function which is used to save cpu cycles. Since my laptop is quite slow, testing models with 10000 images in a short amount of time is not reasonable.

Because of that, the training data is cut in two parts for training and testing. This also removes the need to rely on a second dataset for testing. By default, a split of 75% or 80% of training and the rest for testing has yielded good results.

```
def split_train_test(X, y, prop_train):
    ...
    return tr_X, tr_Y, te_X, te_Y ...
```

With the split obtained, training can start. This is the most important part of the code. It is a function which can make predictions with any classifier provided.

```
def general_classifier(img,lb,clf)
    classifier.fit(training data)
    pred = classifier.predict(test)
return accuracy_score(...)
```

It really shines when combined with a function called multi_train_test() which calls a random forest, KNN and bagging 9 times with parameters ranging from 1 to 9 and the 4 most common SVM kernels (linear, polynomial, rbf and sigmoid).

This way we obtain 23 models which are then compared so that only the best of each category remain to be plotted which allows us to determine easily how well they perform for each task.

6 Experimental Results and Analysis

6.1 Task A1

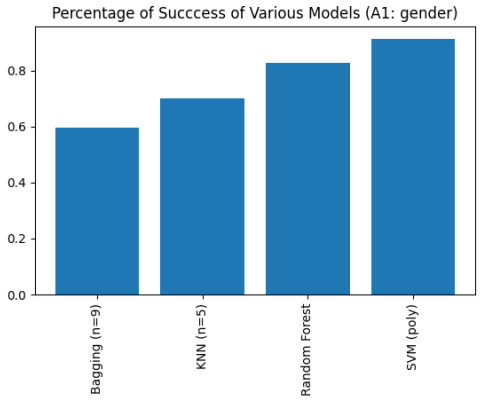

Figure 4: Task A1: gender (5000 images)

Model	Accuracy
Bagging (n=9)	0.5975
KNN (n=5)	0.7
Random Forest	0.8275
SVM (poly)	0.914

For task 1, most models performed reasonably well as the worst one obtained barely less than 60% of accuracy. SVM obtained the best results which is withing expectation as it is the main alternative to CNN in term of efficiency. The only unusual point worth noting is that the poly kernel ended up being the most efficient which is surprising as the linear kernel is tailored for binary classifiers.

6.2 Task A2

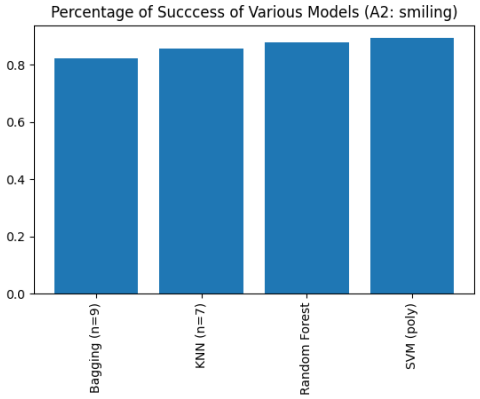

Figure 5: Task A2: emotion (5000 images)

Model	Accuracy
Bagging (n=9)	0.824
KNN (n=7)	0.856
Random Forest	0.878
SVM (poly)	0.894

This task is the one with the best overall results probably related to the fact that smiles tend to be easier to recognise vector wise than gender as they correspond to a curve which is easy to represent mathematically. Because of this, there is not much difference between the predictions from all models which are all in the 80% of success range.

6.3 Task B1

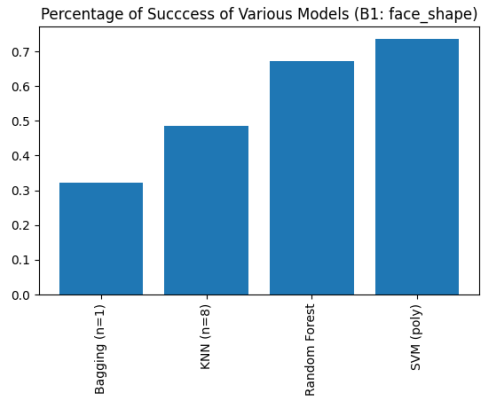

Figure 6: Task B1: Shape (5000 images)

Model	Accuracy
Bagging (n=1)	0.321
KNN (n=8)	0.487
Random Forest	0.672
SVM (poly)	0.736

With the halved training dataset due to my computer's limitations and the addition of more possible values, most models performed noticeably less well. As per usual, the SVM with polynomial kernel got some good result and the random forest came as a close second as it is technically a more refined version of trees / bagging and does not suffer from the KNN limitation which overall needs even more data to compensate the difficulty to obtain distances with the possible values multiplied by a factor 2.5 in comparison to task A.

6.4 Task B2

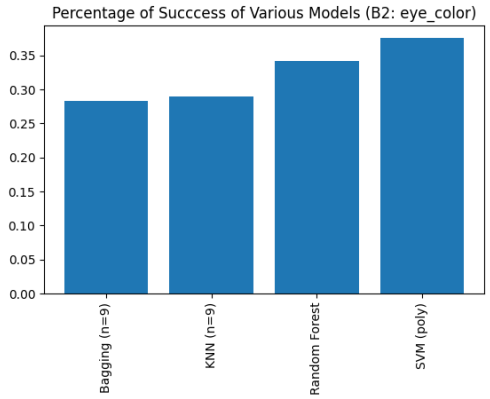

Figure 7: Task B2: Eyes color (5000 images)

Model	Accuracy
Bagging (n=9)	0.284
KNN (n=9)	0.289
Random Forest	0.342
SVM (poly)	0.376

Finally, this task was the most arduous for all models. Identifying eye colour on a grey scaled image is relatively hard even for a human and this is where getting more powerful models such as a CNN really makes a difference. One important point to add is that many characters from the cartoon dataset wear sunglasses which absolutely biases the result by adding unusable data.

6.5 Improvements

From these results, we can infer that Random Forests and SVM perform better than the two other models which are either less refined in the case of bagging or simply too dependant on wide datasets with few possible values for KNN.

In order to get better results, more preprocessing would have been necessary for the cartoon dataset which ended up being harder to analyse for all models.

Regarding Support Vector Machines, the algorithm performed relatively well for every task as it constantly emerged on top. For further tests, it would be better to keep it and try to conceive a custom kernel for task B2 on which it did not perform well.

Additionally, fine tuning with weight alteration and modification of the cost function always remains an option for SVM as long as it does not lead to overfitting by making the result mostly correlated to a subset of the training data

And if it does not suffice, switching to neural networks or deep learning remains a valid option.

7 Conclusion

Hence, if really paid off to try multiple algorithms as out of all the algorithms tested, Support Vector Machines emerged as the most reliable in term of performance and flexibility with their polynomial or custom kernel.

Random forests come as a close second while the other models do not perform well enough to represent an alternative to a neural network.

We may try next time to mix different strategies such as K-means clustering which is extremely slow and therefore inefficient but which has the advantage of converging with absolute certitude.

Moreover, a more promising alternative would be to try more Deep Learning strategies which imitate the way human learn and work particularly well in tandem with neural networks.

Thankfully, the results gathered from the present research should pave the way to better softwares combining them with other strategies such as reinforcement learning in robots and self driving car which could therefore both recognise humans and learn how to drive around them.

References

[1] Isaac Asimov and Rowena Akinyemi. *I, robot*. University Press, 2000.

[2] Ana Echeberria. "The Impact of AI on Business, Economics and Innovation". In: Jan. 2022, pp. 67–96. ISBN: 978-3-030-88240-2. DOI: 10.1007/978-3-030-88241-9_3.

[3] Gina Neff and Peter Nagy. *Talking to Bots: Symbiotic Agency and the Case of Tay*. Tech. rep. Oct. 2016, pp. 4915–4931.

[4] Rexhep Shijaku and Ercan Canhasi. *ChatGPT Generated Text Detection*. Tech. rep. Jan. 2023. DOI: 10.13140/RG.2.2.21317.52960.

[5] Emmanuel Opara. *Chatgpt for teaching, learning and research: prospects and challenges*. Tech. rep. Jan. 2023. DOI: 10.13140/RG.2.2.21304.42244/1.

[6] Derek Saul. "Microsoft Reportedly Closing In On $10 Billion Investment Into ChatGPT Creator OpenAI". In: *Forbes* (Jan. 2023). URL: https://www.forbes.com/sites/dereksaul/2023/01/10/microsoft-reportedly-closing-in-on-10-billion-investment-into-chatgpt-creator-openai/.

[7] Eran Reshef, Gil Raanan, and Eilon Solan. *Method and system for discriminating a human action from a computerized action*. Tech. rep. May 2005.

[8] Gaurav Goswami et al. "Face recognition CAPTCHA". In: Sept. 2012, pp. 412–417. ISBN: 978-1-4673-1384-1. DOI: 10.1109/BTAS.2012.6374608.

[9] Zuber Khan, Tanay Shubham, and Ravi Kumar Arya. "Skin Cancer Detection Using Computer Vision". In: Apr. 2022, pp. 3–11. DOI: 10.1007/978-981-19-0745-6_1.

[10] Emil Talpes et al. "Compute Solution for Tesla's Full Self Driving Computer". In: *IEEE Micro* PP (Feb. 2020), pp. 1–1. DOI: 10.1109/MM.2020.2975764.

[11] Mustafa Al-Dabagh, Salar Rashid, and Muhammad Ahmad. "Face Recognition System Based on Wavelet Transform, Histograms of Oriented Gradients and Support Vector Machine". In: 10 (July 2020).

[12] Zhao Jun. "The Development and Application of Support Vector Machine". In: *Journal of Physics: Conference Series* 1748 (Jan. 2021), p. 052006. DOI: 10.1088/1742-6596/1748/5/052006.

[13] J. Manikandan et al. "Face Detection and Recognition using Open CV Based on Fisher Faces Algorithm". In: *International Journal of Recent Technology and Engineering (IJRTE)* 8 (Jan. 2020), pp. 1204–1208. DOI: 10.35940/ijrte.E5735.018520.

[14] Sanmoy Paul and Sameer Kumar Acharya. "A Comparative Study on Facial Recognition Algorithms". In: *First Pan IIT International Management Conference* (Dec. 2020). URL: http://dx.doi.org/10.2139/ssrn.3753064.

[15] Abhishek Pandey et al. "Face Detection Using Convolutional Neural Network". In: Jan. 2023, pp. 673–679. ISBN: 978-981-19-1905-3. DOI: 10.1007/978-981-19-1906-0_55.

[16] Tijana Vukovic et al. "Thermal Image Degradation Influence on R-CNN Face Detection Performance". In: Nov. 2019, pp. 1–4. DOI: 10.1109/TELFOR48224.2019.8971128.

[17] Pawan Kumar. "Face Reading with AI (Artificial Intelligence)". In: (Dec. 2022).

[18] Mozhdeh Rouhsedaghat et al. "FaceHop: A Light-Weight Low-Resolution Face Gender Classification Method". In: Feb. 2021, pp. 169–183. ISBN: 978-3-030-68792-2. DOI: 10.1007/978-3-030-68793-9_12.

[19] Samuel Dooley et al. *Robustness Disparities in Face Detection*. Tech. rep. Nov. 2022. DOI: 10.48550/arXiv.2211.15937.

[20] Sumithra Ram, Devanur Guru, and Manjunath Aradhya. "Face Image-Based Gender Classification of Children". In: Jan. 2023, pp. 213–228. ISBN: 978-3-031-22404-1. DOI: 10.1007/978-3-031-22405-8_17.

[21] Surbhi Mittal et al. *Are Face Detection Models Biased?* Tech. rep. Nov. 2022. DOI: 10.48550/arXiv.2211.03588.

8 Appendix (github link and code output)

This is the output obtained when running the four tasks with 5000 images.

8.1 A1

```
: Working on task A1 with label gender from dataset celeba
: Proceeding to get 5000 images including 0.75% for training
:
: 100% 5000/5000 [01:57<00:00, 42.63it/s]
: Bagging (n=9) 0.5975
: KNN (n=5) 0.7
: Random Forest 0.8275
: SVM (poly) 0.9141666666666667
```

8.2 A2

```
: Working on task A2 with label smiling from dataset celeba
: Proceeding to get 5000 images including 0.75% for training
:
: 100% 5000/5000 [01:57<00:00, 42.66it/s]
: Bagging (n=9) 0.8241666666666667
: KNN (n=7) 0.8558333333333333
: Random Forest 0.8783333333333333
: SVM (poly) 0.8941666666666667
```

8.3 B1

```
: Working on task B1 with label face_shape from dataset cartoon_set
: Proceeding to get 5000 images including 0.75% for training
:
: 100% 5000/5000 [08:58<00:00,  9.29it/s]
: Bagging (n=1) 0.3209028459273798
: KNN (n=8) 0.4867517173699706
: Random Forest 0.6722276741903828
: SVM (poly) 0.7360157016683022
```

8.4 B2

```
: Working on task B2 with label eye_color from dataset cartoon_set
: Proceeding to get 5000 images including 0.75% for training
:
: 100% 5000/5000 [08:59<00:00,  9.27it/s]
: Bagging (n=9) 0.28361138370951916
: KNN (n=9) 0.28949950932286556
: Random Forest 0.34151128557409227
: SVM (poly) 0.3758586849852797
```

8.5 Github Link

Github: `https://github.com/Ubunteous/applied-ml-final-version`