APPLIED MACHINE LEARNING SYSTEM ELEC0134 22/23 REPORT

SN: 19112997

ABSTRACT

This report explores different machine learning models for image classification problems, both binary and multi-class. Models such as SVM and CNNs were tested to see their performance on the datasets. The results show that SVM is a very powerful model for classification tasks and that it is able to perform as well as the CNN and in one case even outperformed it. The github repository for this project is found here: github.com/moshesimon/AMLS_assignment22_23

1. INTRODUCTION

This report aims to explore and implement various machine-learning models for the purpose of image classification. The methodologies adopt-ed in this report include using Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs) as the primary models for the binary and multiclass tasks respectively. The report is structured in the following way: Firstly, the literature survey provides an overview of the existing methods and the current state-of-the-art in the field of image classification. The description of the models' section provides an in-depth examination of the models used in this study and the reasoning behind their selection. The implementation section details the process of implementing the models, including the preprocessing of the data and the tuning of hyperparameters. Finally, the results and analysis section presents the performance of the models on the datasets.

2. LITERATURE SURVEY

The objective of this literature survey is to provide an overview of the various ML methods used in classification tasks, both binary and multi-class. The survey will compare and contrast classical ML algorithms with state-of-the-art approaches, and evaluate their strengths and weaknesses in solving classification problems.

2.1. Classical Algorithms

In this section I will outline some of the most popular classical algorithms used in binary and multi-class classification.

2.1.1. Logistic Regression

2.1.1.1. Overview

Logistic regression is a supervised machine learning algorithm that is used for binary classification problems. It is a statistical method that is used to model the relationship between a dependent variable and one or more independent variables by fitting a logistic curve to the data. The logistic curve is a Sigmoid function, 1, that maps the predicted probability of the dependent variable being a certain class to the real-valued outputThe algorithm works by finding the best coefficients for the independent variables that minimize the error between the predicted probabilities and the true binary outcomes. This is done using an optimization algorithm, such as gradient descent, which iteratively updates the coefficients until the error is minimized. The final coefficients can then be used to make predictions on new data.

$$\mathrm{sigm}(\eta) = \frac{1}{1 + \exp(-\eta)} \tag{1}$$

2.1.1.2. Advantages and Disadvantages

The advantages of logistic regression include its simplicity, interpretability, and ability to handle a large number of predictor variables. It is also computationally efficient and does not require extensive tuning, unlike other machine learning algorithms. Logistic regression is also less prone to overfitting, as it does not allow for complex non-linear relationships between the independent and dependent variables.

On the downside, logistic regression assumes a linear relationship between the independent variables and the log odds of the dependent variable. This can result in poor performance if the relationship is actually non-linear. Logistic regression also assumes that the independent variables are independent of each other, which may not be the case in real-world scenarios. In such cases, the results of logistic regression can be biased and unreliable.

2.1.1.3. Applications in Classification Tasks

The study [1], investigated the application of logistic regression in online marketing. The focus was on demographic targeting, where the aim was to effectively target relevant internet users and predict their likelihood of purchasing promoted products or services. The authors used binomial and multinomial logistic regression models to predict the gender and age category of internet users, respectively. The results

showed that logistic regression models can be used to estimate the probabilities that an internet user is from a target group.

2.1.2. SVM

2.1.2.1. Overview

Support Vector Machines (SVMs) are a popular and powerful class of supervised learning algorithms for both classification and regression tasks. They are based on the idea of finding a hyperplane that best separates a dataset into different classes. SVMs create hyperplanes that separate the classes and that maximise the distance between them. These are called support vectors, see Fig. 1.

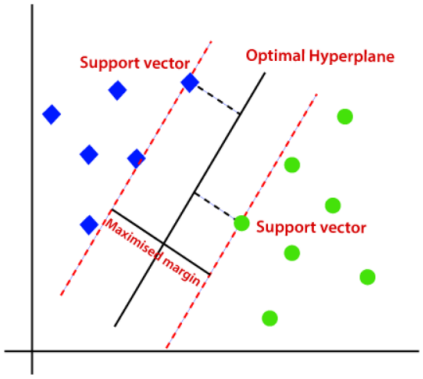

Fig. 1. Support Vectors

SVMs were first introduced in the 1960s by Vapnik and Chervonenkis as a method for pattern recognition. However, it wasn't until the 1990s that the algorithm gained widespread popularity, thanks to the work of Vapnik and his colleagues, who developed the theoretical foundations of SVMs and made them more accessible to a wider audience.

2.1.2.2. Advantages and Disadvantages

SVMs are a powerful algorithm that can map data into a higher-dimensional feature space where it can be linearly separated. This enables SVMs to model complex decision boundaries, making them particularly useful for datasets with a large number of features or when the decision boundary is not linear. Despite its popularity, SVMs do have some limitations. One of the main limitations is that SVMs can be sensitive to the choice of kernel function and the value of the regularization parameter. Additionally, SVMs can be computationally expensive, particularly when dealing with large datasets.

2.1.2.3. Applications in Classification Tasks

VMs have been used for a wide range of applications, including image classification [2], text classification [3] and stock market prediction [4]. In [5], the authors conducted a comparison between various machine learning algorithms including Support Vector Machine (SVM), Decision Tree (C4.5), Naive Bayes (NB) and k Nearest Neighbors (k-NN) on the Wisconsin Breast Cancer dataset. The objective of the study was to evaluate the effectiveness and efficiency of each algorithm in terms of accuracy, precision, sensitivity, and specificity in classifying breast cancer data. The results of the experiment showed that SVM gave the highest accuracy (97.13%) with the lowest error rate, outperforming other algorithms such as Decision Tree, Naive Bayes and k-NN.

In [6], the authors tested a SVM trained on the feature array produced by dlib's 68 face landmarks predictor for the purpose of emotion detection. They found that the combination of the two was a very powerful classification tool that had low computation times and high accuracy.

2.1.3. Random Forest

2.1.3.1. Overview

Random Forest is a popular ensemble learning method for classification and regression tasks. The algorithm creates multiple decision trees and combines their predictions to improve the overall performance of the model. The algorithm starts by selecting a random subset of the training data and building a decision tree from that subset. This process is repeated multiple times, with each tree in the forest making a prediction for the input data. The final output is determined by combining the predictions of all the trees in the forest, typically through a majority vote for classification tasks or averaging for regression tasks.

2.1.3.2. Advantages and Disadvantages

A main advantage of Random Forest models is their ability to handle data that is high-dimensional and large numbers of features. They are also less prone to overfitting compared to individual decision trees, as the ensemble of trees helps to reduce the variance. Additionally, Random Forest models are able to handle missing data and categorical variables, which is an advantage over some other machine learning methods. The drawbacks are that Random Forest models can be computationally expensive and less interpretable compared to other models, and can be prone to overfitting if the number of trees in the forest is too large. They are also not good at handling high-dimensional and sparse data

2.1.3.3. Applications in Classification Tasks

In binary classification, Random Forest models have been used to predict outcomes in fields such as finance, medicine, and ecology. For example, in finance, Random Forest models have been used to predict credit risk and fraudulent transactions. In [7], the authors found that Random Forest outperformed most other machine learning algorithms in terms of

accuracy and ability to help understand the relationships between analyzed attributes. In medicine, they have been used to predict disease outcomes and patient survival. In ecology, they have been used to predict species distribution and conservation management [8].

2.2. State-of-the-art Algorithms

In this section I will outline some of the state-of-the-art algorithms used in binary and multi-class classification.

2.2.1. CNN

2.2.1.1. Overview

Convolutional Neural Networks (CNNs) are a type of deep learning algorithm that are particularly effective for image and audio classification tasks. They are designed to take advantage of the local, spatial relationships in input data, such as an image. CNNs have a unique architecture that includes a series of convolutional, pooling, and activation layers, followed by a fully connected layer. See 2.

In the convolutional layer, the input image is convolved with a set of filters or kernels. These filters are trained to identify different features in the image, such as edges, corners, and textures. The pooling layer then reduces the spatial size of the convolved feature maps, increasing their robustness to small translations and distortions in the input image. The activation layer applies a non-linear activation function, such as the ReLU activation, to the pooled feature maps. The fully connected layer takes the output of the activation layer and computes the final class scores, which determine the final classification of the input image. The weights and biases of the filters, pooling, and activation layers are learned during the training process, allowing the network to learn the representations that are most effective for the given classification task.

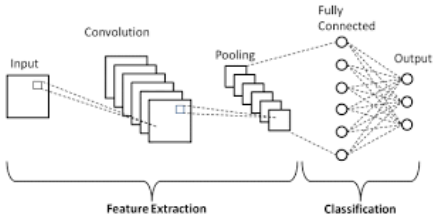

Fig. 2. CNN architecture

2.2.1.2. Advantages and Disadvantages

The advantages of CNNs include their ability to handle large input images, their robustness to translations and distortions, and their ability to learn hierarchical representations of the input data. However, CNNs have proven to not be as effective in smaller datasets. In [9], the authors compare the performance of SVM and CNNs, using both a large sample MNIST dataset and a small sample COREL1000 dataset. The study found that while CNN showed a much higher accuracy on the larger sample MNIST dataset, SVM had a higher accuracy on the small sample dataset, demonstrating the superiority of deep learning frameworks in handling larger sample data sets and its limitations in smaller datasets. Another drawback is that they can be computationally expensive to train, especially for large datasets.

2.2.1.3. Applications in Classification Tasks

CNNs have a wide range of real-word applications but they are mainly used for image classification tasks. In [10] the authors investigated the performance of CNNs in identifying newborn babies. The study uses the IIT (BHU) newborn database and compares the results of the proposed Deep CNN approach with state-of-the-art techniques, including PCA, LDA, ICA, LBP, and SURF. The results show that the proposed Deep CNN approach outperforms these techniques, improving their results by up to 22.09% for neutral-neutral faces. [11]

3. DESCRIPTION OF MODELS

3.1. Task A1: Gender detection

Task A1 is to implement a model that is able to accurately detect the gender of person given a photo of their face. For this task there are a number of different possible approaches. There is the classical supervised learning algorithms like SVMs and Logistic regression, and then there is the more modern algorithms like CNNs. In general, CNNs are more suited for image classification tasks compared to SVMs and logistic regression. This is because CNNs are designed to capture the hierarchical patterns in images, which is a crucial aspect for image classification. CNNs also automatically learn the features from the image data, whereas in SVMs and logistic regression, you need to manually extract the features from the image data. However, since we have a small dataset, either SVM or logistic regression would be the best choice since they tend to generalize well with smaller datasets, whereas CNNs are designed to handle large amounts of data.

Logistic Regression is a simple and straightforward algorithm that models the probability of a binary outcome based on a linear combination of input features. Linear models like logistic regression assume that the relationship between the input features and output class is linear. The main advantage of SVMs is that they are able to handle data that is not linearly separable using a kernel trick. In this male/female image classification problem, the input features and output class are not necessarily linear, so it is possible that a SVM will perform better than logistic regression. I will also test logistic regression for comparison purposes.

Technically it would be possible to input the raw image pixel data into the SVM; however, this would be very computationally expensive. To address this issue, it is common to pre-process the raw pixel data to extract meaningful features that can be used as input data. For this task I have decided to use dlib's 68 face landmarks predictor to detect the face and extract the facial features. This will return an array of the coordinates of 68 facial features ready to be inputted into the SVM for classification.

In summary, for this task I will be using a combination of a pre-trained 68 face landmarks predictor to extract a feature array that will then feed into a SVM model to be trained to perform binary classification.

3.2. Task A2: Smile detection

Task A2 is to train a model to determine whether a person in a given image is smiling or not smiling. This task requires a slightly different approach than Task A1. The reason is because the difference in features between a person who is smiling vs one who is not smiling can sometimes be very subtle. For example a person who is half smiling or smiling from one corner of their mouth. Because this problem is more complex i believe it requires a more advanced model that is able to pick up on these subtleties. I am sceptical that the 68 face landmarks predictor, used in task A1, gives a complete enough picture of the facial features to perform this classification to a high accuracy. Also, SVMs work better when there is a clear separation of classes. This was the case for male/female, but in this instance the line between smiling and not smiling is very thin. For these reasons I will train a CNN on the celeba dataset and allow it to perform the classification itself. I am unsure, however, of the CNNs ability to produce highly accurate results in this case due to the fact that the dataset is so small. Therefore, I will also train a SVM using the 68 face landmarks predictor and compare the results.

3.3. Task B1: Face shape classification

Task B1 is to separate cartoon images into categories of 5 face shapes. The training data contains 10,000 images of cartoon characters with 5 different types of face shapes. The classes in this data are very well defined since there is no variation in pixel position between the face shape features of cartoons with the same face shape. This clear categorisation makes it idea for SVMs to find well defined decision boundary between the data points.

3.4. Task B2: Eye colour classification

Task B2 is to classify the eye colour of cartoon characters into 5 colours. This data also has very well defined boundaries which also makes it an ideal problem for a SVM. This task won't require the landmark predictor since we are not classifying features but rather the pixel values around the eye. In order to simplify the problem I will crop out the section of the image that is most relevant to be used as training data.

4. IMPLEMENTATION

4.1. Task A1: Gender detection

4.1.1. Data Preparation

The data set we were given as training and testing data for this task was 5000 and 1000 images respectively. The data was stored in a folder called 'celeba/imgs'. The files were named from 0 - 4999 and 0 - 999 and the file type was .jpg. Each image was a photograph of the face of a celebrity. The orientation of the faces varied. In terms of loading the data there were 3 main functions that were used.

- *load_images()* loads all the images in a give directory, using the *load_img()* function from *keras.utils.image_utils* library, converts them to an array of numbers with type uint8 and shape (218,178,3), using the *img_to_array()* function from the above library, and then converts to grayscale, using the *cvt_colour()* function from cv2. The image array is then added into another array. The final array has shape (5000,218,178,1) which is then converted to a numPy array using the array() function and returned. This function can also be used to return the full array of test image arrays with shape (1000,218,178,1).

- *load_labels()* loads the csv file as a data frame from a given directory using pandas *read_csv()* function. The labels are then retrieved from the appropriate column and converted to a numPy array before being returned. The labels represent the gender the person in the image, -1 for a female and 1 for a male.

For this task we will first be extracting the facial features using a pre-trained 68 face landmarks predictor before feeding it to the SVM as training data. There are 2 main functions used to achieve this.

- *extract_features_labels()* receives an array of images along with the labels. It loops through the images and passes the image to the *run_dlib_shape()* function to extract the features. Once returned, the features are added to an array. If no face is detected then *run_dlib_shape()* returns *None*. In this case the label associated with this particular image is deleted from the labels array. In order to maintain the correct indexes in the labels array, each time a label is deleted and arbitrary integer (-5) is added to the front of the array. Once all the images are processed, these added integers are removed and the updated labels array is returned along with an array of all the image features. The shape of the features array is (features.shape[0], 136).

- *run_dlib_shape()* uses a face detector from the dlib library to initially draw a box around any faces it sees in the image. It then loops through each box and makes a prediction of the feature coordinates of that face using dlib's *shape_predictor*. It then keeps the largest face and return its features. The feature array has shape (68,2).

4.1.2. Model training

For this task I have decided to use an SVM for my model. I created the classifier object from the SVC Class from the sklearn library. In order ensure the best performance I tested 3 different values for the C parameter, [0.01, 0.1, 1]. In SVMs, the "C" parameter is a regularization term that controls the trade-off between achieving a low training error and a low testing error. It determines the balance between smooth decision boundary (high bias) and complex decision boundary (high variance). To test these different values I used the *GridSearchCV()* Class from sklearn. This class also performs cross validation on the training dataset. Cross-validation is a technique to evaluate a model's performance by dividing the data into two parts: training and validation. The model is trained on the training data and evaluated on the validation data. The process is repeated k times with each part used once as validation data and the average performance across all iterations is used as the model's performance metric. Cross-validation helps prevent overfitting and provides a robust estimate of performance. In this case I chose k = 5 since a 80%:20% split is popular split for training and validation sets. The result from the grid search was that C=0.01 was brought about the best performance. The kernel I used for the SVM was the linear kernel. The reason is because it is reasonable to assume that the data is linearly separable unless the SVM's performance indicates that the data is not linearly separable.

4.1.3. Model Performance

To evaluate the models performance, the best test is to compare its predictions on the unseen test data with the test labels. This will show how well the the model learnt to generalise to unseen data. I used the *classifier.predict(test_data)* function to get the best models predictions. I then used the *accuracy_score()* function, from the *sklearn.metrics* library, to produce an accuracy score of the prediction. I also used the *confusion_matrix()* function, from the *sklearn.metrics* library, to produce a confusion matrix.

In machine learning, models start off with a high bias and low variance and as they become more complex, they tend to have lower bias and higher variance. In general, a model with high bias is more likely to underfit the data and make poor predictions, while a model with high variance is more likely to overfit the data and make good predictions on the training data but poor predictions on new unseen data. The goal is to train the model to the point where there is a balance between bias and variance. Therefore, it is important to stop the training process before the variance increases too much, as this would indicate that the model is overfitting the data and may not generalize well to new unseen data. In this task I decided to allow the model to train on the full set of training data and then look at the learning curve to evaluate whether the model is overfitting the data and would need to be stopped earlier. See Fig. 3 which shows the learning curve for the SVM.

Fig. 3. SVM learning curve

The learning curve graph shows a nice convergence of the training and validation scores with a small generalisation gap. The model looks like it becomes stable after 3500 images which indicates that more training data wouldn't improve performance much. From these observation it does not look like the model has began overfitting the training data. I have therefore decided not to include any stopping criteria in my training. I got the data to plot the learning curve from the *learning_curve()* function from *sklearn*. I then used the *matplotlib* library to plots the graph.

4.2. Task A2: Smile detection

4.2.1. Data Preparation

In this task I will be train a CNN model on the data as well as a SVM + landmark predictor. The data for this task was the celeba dataset, which was also used in Task A1. The data preparation for the SVM is the same as outlined in 4.1.1. The data preparation for the CNN is the same but without the feature extraction step. Instead, the raw pixel data is used as inputs to the CNN. The labels are modified slightly for the CNN by changing each -1 to a 0.

4.2.2. Model training

The SVM + dlib model training for this task is the same as in 4.1.2.

The CNN was built using the *Sequential()* class from the Keras library in the TensorFlow framework. You make a

model object from this class and then you add any layers you like to the model using the *add()* function. I will now outline the CNN architecture and justify any choices I made:

- Rescaling: The first layer, Rescaling, scales the pixel values of the input images from a range of [0, 255] to [0, 1]. This is a standard preprocessing step in image classification tasks that helps the model to converge faster.
- Conv2D Layers: The following three Conv2D layers (with 16, 32, and 64 filters respectively) are designed to learn local features of the input images. The kernel_size of 3x3 and the ReLU activation function allow the model to capture complex patterns in the images while reducing the number of parameters required to be learned.
- MaxPooling2D Layers: The three MaxPooling2D layers are used to reduce the spatial dimensions of the feature maps produced by the Conv2D layers. The pool size of (2, 2) in this CNN architecture is a commonly used choice that helps to reduce the computational requirements of the model while effectively retaining the most important information in the feature maps.
- Flatten: The Flatten layer is used to convert the multi-dimensional feature maps produced by the Conv2D layers into a single vector, which is then fed into the fully connected layers.
- Dense Layers: The two Dense layers (with 128 and 1 neurons respectively) form the final part of the architecture. The first Dense layer uses a ReLU activation function, allowing it to model non-linear relationships between the features learned by the Conv2D layers. The final Dense layer uses a sigmoid activation function, producing the final binary classification probability.

The compile parameters chosen for this binary classification problem were the Adam optimizer with a learning rate of 0.0001 and Binary Crossentropy loss. The Adam optimizer is a popular choice for deep learning tasks as it combines the advantages of both Stochastic Gradient Descent (SGD) and Root Mean Square Propagation (RMSprop) algorithms. It adjusts the learning rate adaptively for each parameter during training, leading to faster convergence and better performance compared to SGD. The learning rate of 0.0001 was chosen as a low value to ensure that the optimizer makes small updates to the weights during each iteration, allowing the model to converge to a good solution.

Binary Crossentropy loss is commonly used for binary classification problems, as it measures the dissimilarity between the true binary label and the predicted probability. The loss function is designed to penalize the model when its predictions are far from the true labels, allowing it to learn to make better predictions over time. The choice of Binary Crossentropy loss and the Adam optimizer, along with the low learning rate, should ensure that the model converges to a good solution and generalizes well to unseen data.

4.2.3. Model Performance

The CNN's performance was evaluated using the *model.evaluate()* function from keras. The *model.fit()* function returns an object that has training and validation accuracy data. I used this data to plot a learning curve graph using matplotlib. The plot is shown in Fig. 4.

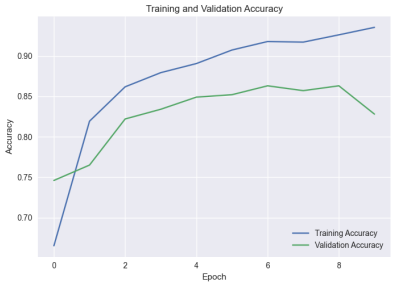

Fig. 4. SVM learning curve

The graph shows that the model starts to overfit the data after the 7th epoch because the validation score starts to drop while the training score continues to rise. I therefore decided to stop the training after 8 epochs. The learning curve with stopping is shown in Fig. 5.

4.3. Task B1: Face shape classification

4.3.1. Data Preparation

For this task we were given the cartoon dataset which contains 10,000 png images of cartoon faces in the training set and 2500 in the test set. To extract the pixel images the I used the *load_images()* function outlined in 4.1.1. In this task I trained a SVM on the raw pixel data as well as on the feature array extracted by the 68 face landmarks predictor using the

Fig. 5. SVM learning curve with stopping

extract_feature_labels() function also outlined in 4.1.1. To input the pixel data into a SVM it needs to be in 1D. So I used the *reshape()* function from numPy to transform the shape of the image array from (10000, 500, 500) to (10000, 250000).

4.3.2. Model training

I used the same functions for training the SVM + dlib as in 4.1.2. I also used the same functions for training SVM alone just with the flattened image pxel array instead. Due to system limitations I had to use the UCL servers to train the SVM for this task. The reason is because each image has 250,000 features and there are 10,000 images in total to process. It would take my laptop many hours to compute.

4.3.3. Model Performance

The SVM model was trained on the entire dataset and achieved a score of 99.96% accuracy on the test set. I didn't run any performance checks like plotting the learning curve because I didn't think need model could possibly perform any better.

4.4. Task B2: Eye colour classification

4.4.1. Data Preparation

This task requires some additional data preparation steps than the previous 3 tasks. This is because we need to classify cartoon images based on eye colour. The first change was I set the grayscale parameter in the *load_images()* function to False. A problem that I encountered with this dataset is that many of the images in the train and test datasets have sunglasses over their eyes. This will confuse the model and so they need to be removed. Another step will be to crop the image to only include one of the eyes. This is the only part of the image that is relevant for the model to perform its classification.

These two adjustments are made simultaneously by calling the *extract_eyes()* function.

- *extract_eyes()* receives an array of images with shape (10000,500,500,3) and an array of labels. It loops through the images and immediately checks if the image has sunglasses by calling the *has_sunglasses()* function. If the image does have sunglasses then its label is removed from the label array and the for loop continues. If the image does not have glasses then the left eye, located at [257:273,198:227], is cropped out and saved to an array of eyes. The array of eyes and eye labels are then returned.

- *has_sunglasses()* uses the whites of the eyes to tell if the cartoon is wearing sunglasses. It receives an image and crops out the place where the white of the eye is supposed to be, [250:273,190:222]. This area is then flattened using the *flatten()* function, and the average is over all the values in the array. The cutoff boundary of 200 is used to determine if sunglasses are being worn. If none are worn the average should be 255. I have set the boundary to 200 because in some cases the cartoon is waring reading glasses which slightly dim the brightness of the white but don't change the colour dramatically enough to warrant removal. The function returns a bool value.

4.4.2. Model training

The SVM was trained on the raw eye image pixels. The process is the same as in 4.3.2 with flattened eye pixel arrays as inputs. The CNN was trained the same way as in 4.2

4.4.3. Model Performance

The SVM model training was incredibly quick even though it was trained on the pixel data. This is because each eye image array is only shape (23,32). Again, the model had a very high accuracy so no stopping criteria was implemented.

5. EXPERIMENTAL RESULTS AND ANALYSIS

This section will discuss the results of the models I tested for the various tasks. The models performance is judged based on its accuracy score on the unseen test data set. Below is a table showing each model tested for each task, along with it's accuracy score. See Fig. 6.

Task	Model	Parameters	Accuracy
A1	Logistic Regression + feature detection	C = 1.0	0.9268
	SVM + feature detection	C = 1.0 Kernel= linear	0.9309
A2	SVM + feature detection	C = 0.01 Kernel= linear	0.9082
	CNN	Learning rate = 0.001	0.8
B1	SVM	Kernel= linear	0.9996
	SVM + feature detection	Kernel= linear	0.7380
	CNN	Learning rate = 0.001	0.9996
B2	SVM	Kernel= linear	0.8304
	SVM	Sunglasses removed	1.0
	CNN	Learning rate = 0.001	1.0

Fig. 6. Table of results

5.1. Task A1

The results for Task A1 show that SVM + dlib feature detector is a powerful combination for image classification tasks. We can also see that Logistic Regression performed well on this task but not as well as the SVM.

5.2. Task A2

The results for Task A2 show that the SVM outperformed the CNN. This is probably due to the small dataset size. The results for the SVM were worse than the other tasks. This is maybe due to outliers being present in the dataset.

5.3. Task B1

I first tested the same approach as in Task A1, by using the dlib 68 face landmarks predictor to prepare feature arrays for the SVM to train on. I was surprised to see that the results were worse than I expected with a test accuracy of around 74%. I thought that the SVM is certainly advanced enough to be able to classify such clear categories and able to handle any outliers in the data. So I thought maybe it is the 68 face landmarks predictor that is doing a bad job of capturing the face shapes so I decided to test the SVM without the 68 face landmarks predictor. I trained a SVM on the raw image pixel data and it performed remarkably better with a test accuracy of 99.96%. I also trained a CNN and got a similar result. This shows that the dlib predictor was the reasons for the previous bad performance.

5.4. Task B2

The results for Task B2 show that removing the sunglasses improved hr accuracy result from 83% to 100%. This shows the effect that outlines have on the model's performance.

6. CONCLUSION

This report explored different machine learning classification tasks. There where two binary tasks using a celebrity dataset and another two multi class task uing a cartoon dataset. SVMs were the most used model to solve these tasks, showing impressive performance in its ability to handle different tasks and datasets. The models were implemented using the SKlearn python library.

In order to improve the performance of the models in tasks A1 and A2, additional prepossessing steps should be implemented to remove outliers and focus on relevant features. It would also be interesting to see how more advanced models such as RNNs would perform on these tasks.

7. REFERENCES

[1] Serkan Akinci, Erdener Kaynak, Eda Atilgan, and Safak Aksoy, "Where does the logistic regression analysis stand in marketing literature?: A comparison of the market positioning of prominent marketing journals," *European Journal of Marketing*, vol. 41, pp. 537–567, 06 2007.

[2] Mayank Arya Chandra and S S Bedi, "Survey on SVM and their application in image classification," *International Journal of Information Technology*, vol. 13, no. 5, pp. 1–11, Oct. 2021.

[3] Zi-qiang Wang, Xia Sun, De-xian Zhang, and Xin Li, "An optimal svm-based text classification algorithm," in *2006 International Conference on Machine Learning and Cybernetics*, 2006, pp. 1378–1381.

[4] WEN Fenghua, XIAO Jihong, HE Zhifang, and GONG Xu, "Stock price prediction based on ssa and svm," *Procedia Computer Science*, vol. 31, pp. 625–631, 2014.

[5] Hiba Asri, Hajar Mousannif, Hassan Al Moatassime, and Thomas Noel, "Using machine learning algorithms for breast cancer risk prediction and diagnosis," *Procedia Computer Science*, vol. 83, pp. 1064–1069, 2016.

[6] K. M. Rajesh and M. Naveenkumar, "A robust method for face recognition and face emotion detection system using support vector machines," in *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, 2016, pp. 1–5.

[7] Nazeeh Ghatasheh, "Business analytics using random forest trees for credit risk prediction: A comparison study," *International Journal of Advanced Science and Technology*, vol. 72, pp. 19–30, 11 2014.

[8] Chunrong Mi, Falk Huettmann, Yumin Guo, Xuesong Han, and Lijia Wen, "Why choose random forest to predict rare species distribution with few samples in large undersampled areas? three asian crane species models provide supporting evidence," *PeerJ*, vol. 5, pp. e2849, 2017.

[9] Pin Wang, En Fan, and Peng Wang, "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning," *Pattern Recognition Letters*, vol. 141, pp. 61–67, 2021.

[10] Rishav Singh and Hari Om, "Newborn face recognition using deep convolutional neural network," *Multimedia Tools and Applications*, vol. 76, no. 18, pp. 19005–19015, 2017.

[11] Hang Li, "Deep learning for natural language processing: advantages and challenges," *National Science Review*, vol. 5, no. 1, pp. 24–26, 09 2017.