A yellow bird with a crown on a blue background

Description automatically generated

Anglia Ruskin University

MACHINE LEARNING FOR IMAGING

Module Code: MOD 009702 TRI1 F01CAM

Academic Year: 2024/2025

Mohammad Ali Jaber 2365662

Contents

[Dataset Curation and Presentation 3](#_Toc185029361)

[Techniques 4](#_Toc185029362)

[1. PCA and KNN 4](#_Toc185029363)

[2. Decision Tree and Random Forest 5](#_Toc185029364)

[3. Stochastic Gradient Descent 6](#_Toc185029365)

[4. MLP 7](#_Toc185029366)

[5. CNN 8](#_Toc185029367)

[6. Pre-Trained Network 9](#_Toc185029368)

[SVM (ORIGINAL IDEA) 11](#_Toc185029369)

[Methods Comparison and Discussion 12](#_Toc185029370)

[Conclusion 13](#_Toc185029371)

[References 14](#_Toc185029372)

# Dataset Curation and Presentation

The dataset used for this report was generated using OpenImages v7 and includes images of two selected classes: Fish and Tiger. The dataset consists of 4900 images: 4445 in the fish class and 455 in the tiger class. However, most of the fish images were too small (between 10-500 KB in size). These images were deleted as they negatively affected the training and acted as noise data. As a result, the Fish dataset was reduced to 738 images, leaving 1193 total images. This reduction improved the balance between the 2 classes, as presented in the pie chart in Figure 1.

Images were pre-processed by resizing them to 256 × 256 pixels and converting them to grayscale to reduce computational complexity using OpenCV (cv2). After preprocessing, the images were flattened into 1D array, and labels were encoded using LabelEncoder from sklearn, where ‘Tiger’ was encoded as ‘1’ and ‘Fish’ as ’0’. Then the dataset was split into training and testing subsets with an 80/20 ratio, Figure 2 below shows the shapes of the data after splitting.

|  |  |
| --- | --- |
| A blue and orange pie chart  Description automatically generated  Figure 1-Class Distribution Pie Chart | A screenshot of a computer code  Description automatically generated  Figure 2-Data Shape After Train-Test Split |

# Techniques

## PCA and KNN

Principal component analysis (PCA) is a dimensionality reduction technique used in this model to reduce the dimensionality of pixels in images while preserving the most critical visual information. After flattening the images, there were 65536 features, and PCA reduced this to 400 components. Notice Figure 3 below that shows how much variance is explained by each principal component.

|  |
| --- |
| Figure 3-Explained variance ratio |

|  |  |
| --- | --- |
| Figure 4-a | Figure 4-b |

Figure 4-Effect of PCA on an image

Figure 4 above shows the effect of PCA on the image data. The first image (Figure 4-a) is a grey-scaled image from the Tiger class and the second image (Figure 4-b) is the same tiger image after applying PCA, as shown in Figure 4-b the shape of a tiger is still noticeable somehow.

These reduced features were used to train a KNN classifier with a number of neighbours equals 7.

The resulting classification accuracy is 61.92%.

## Decision Tree and Random Forest

A Decision Tree classifier was initially implemented, this classifier achieved 68.2% accuracy. However, the train score was 100% which indicates overfitting. To tackle this issue, bagging method was employed, so Random Forest classifier was implemented with a tree depth of 9. This Random Forest model achieved 72.38% accuracy, and no overfitting occurred during the training of this model.

|  |  |
| --- | --- |
| Figure 5-a-Decision Tree confusion matrix | Figure 5-b-Random Forest confusion matrix |

Figures 5-a and 5-b above represents the confusion matrices of the two classifiers: Decision Tree in Figure 5-a and Random Forest in Figure 5-b. As seen in these matrices, the Random Forest had fewer errors in the false positive and negative sections and performed better in the true positive and negative sections compared to Decision Tree model.

## Stochastic Gradient Descent

SGD classifier was implemented with a hinge loss function, a ‘l2’ regularization parameter to mitigate overfitting, and 450 max iterations. This classifier achieved 66.53% accuracy.

The table below represents the classification report of this classifier.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| 0 | 0.68 | 0.87 | 0.76 | 148 |
| 1 | 0.61 | 0.33 | 0.43 | 91 |
| Accuracy | - | - | 0.67 | 239 |
| Macro avg | 0.65 | 0.6 | 0.6 | 239 |
| Weighted avg | 0.65 | 0.67 | 0.64 | 239 |

## MLP

An MLP model was implemented using Keras, starting with an input layer of 512 units and ReLU activation, with the input shape defined by the train data from PCA. It is followed by two hidden layers with 256 and 128 units, separated by two dropout layers (50% each) to reduce overfitting. The output layer which has a single unit and a sigmoid activation function, making it suitable for binary classification tasks.

Figure 6 below shows the summary of the MLP model

|  |
| --- |
| Figure 6-MLP summary |

The graph below represented in Figure 7 plots the training and validation mean absolute errors over the epochs, showing performance trends during training.

A graph showing the growth of the stock market

Description automatically generated with medium confidence

Figure 7-MAE graph for MLP model

Additionally, EarlyStopping was added to the training of this model to prevent overfitting by stopping the training when the validation performance stops improving. And then saving the model’s best performing weights using ModelCheckpoint.

This model achieved an accuracy of 71.13%.

## CNN

The data was split into train and validation datasets using TensorFlow with a 20/80 ratio.

The first layer in the Convolutional Neural Network (CNN) was Rescaling, this layer rescales pixel values to a [0, 1] range by dividing by 256. The next layer is a convolution 2D layer which applies 10 filters of size 7x7 to the data, followed by a Max Pooling layer which reduces the dimensions by a factor of 4. This is repeated with another Conv2D and MaxPooling2D layers to extract deeper features. Then, a Flatten layer is added to convert the 2D features into 1D vector, which is passed to a dense layer with 128 units and ReLU activation. Finally, the output layer with 2 units and sigmoid activation is added to perform binary classification.

Figure 8 below represents the training and validation loss functions graph over epochs. As seen after epoch 10 the validation loss function started increasing, this is when the EarlyStopping method comes to rescue and stops the training and then ModelCheckpoint saves the model at this specific epoch since it carries the best metrics.

A graph with a line and a line

Description automatically generated

Figure 8-loss function graph of CNN

This model achieved 86.13% accuracy.

## Pre-Trained Network

Figure 9 below shows the layers and parameters of the pre trained network.

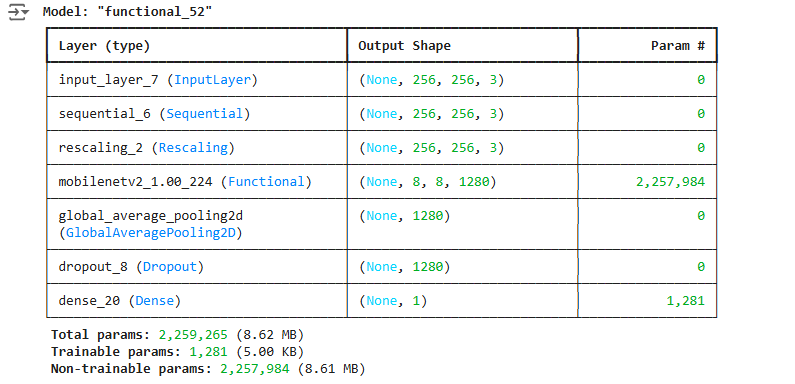


Figure 9-Pre-Trained Network summary

The pipeline of this network starts with data augmentation, including random vertical flips and rotations of -36° and 36° (0.1 is 10% of 360°), followed by a rescaling layer to normalise pixel values to a range of [-1, 1].

Then MobileNetV2 layer representing the base model, pre trained on the ImageNet dataset, is used as a feature extractor with its weights frozen to maintain the learned features. Then the global average layer is added to reduce the dimensions of the extracted features, followed by a dropout layer to prevent overfitting. Finally, the prediction layer which represents the output dense layer.

A graph with a line and a line

Description automatically generated with medium confidence

Figure 10-loss function graph of Pre-Trained Network

The graph in Figure 10 above represents the loss functions of the pre-trained network during training and validation. This graph shows a steady decrease with increasing epochs, reaching low values around 0.01 for the validation loss function, indicating the strong performance of the pre-trained network model.

Similarly, and for the same tuning purposes as part 4 and 5, EarlyStopping and ModelCheckpoint were added to the training part of this model.

This classifier achieved 99.58% accuracy.

# SVM (ORIGINAL IDEA)

In machine learning, support vector machines (SVMs, also support vector networks) are supervised max-margin models with associated learning algorithms that analyze data for classification and regression analysis. Developed at AT&T Bell Laboratories, SVMs are one of the most studied models, being based on statistical learning frameworks of VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974).

In this model, SVM is implemented using a pipeline. First, the data is normalized through the StandardScaler layer, then it is passed to the Support Vector Classifier which is configured with an automatic gamma value to control the kernel coefficient.

This classifier has an accuracy of 75.31%.

The confusion matrix below in Figure 11 illustrates the predictions of this model.

A chart of a blue yellow and purple box

Description automatically generated with medium confidence

Figure 11-SVM confusion matrix

# Methods Comparison and Discussion

The resulting metrics show clear differences in performance between the techniques.

The Decision Tree had a bad performance due to overfitting during training, followed by the KNN classifier with the lowest accuracy of 61.92%. Next was SGD, which achieved a low accuracy of 66.53%. The accuracies then increased, reaching 71.13% with MLP, which is kind of acceptable, while Random Forest performed slightly better at 72.38%. SVM showed a slight improvement, reaching 75.31% accuracy. A significant jump to 86.13% accuracy was achieved by CNN, and finally, the best performance came from the pre-trained network technique, achieving an impressive 99.58% accuracy.

The histogram in Figure 12 below illustrates the difference in the achieved accuracies with relevant to each model.

A graph of different colored rectangular shapes

Description automatically generated

Figure 12-Accuracy comparison

# Conclusion

This study demonstrated the effectiveness of various machine learning and deep learning techniques for image classification. While traditional methods like Random Forest and SGD are reliable, neural networks models, especially pretrained networks, offered superior accuracy. Future work could include exploring other features, such as color features and shape features, to build our models on.

# References

* Wikipedia (2024) *Support vector machine.* Available at: <https://en.wikipedia.org/wiki/Support_vector_machine> (Accessed: 13 December 2024).