INM460 COMPUTER VISION COURSEWORK

Daniel Addai – Marnu - **daniel.addai-marnu@city.ac.uk**

1.0 INTRODUCTION

This paper presents the methods and approaches explored and implemented in developing functions to perform Face Recognition and Object Character Recognition (OCR) per the course problem outline and given image dataset.

The first part of the paper presents the data, the initial processing steps and a brief descriptive statistics of the data. Next is the Face Recognition, which includes a justification and description of the processes implemented, the implementation pipeline and an analysis of the outcome. This is followed by the OCR, which is documented similarly as the Face Recognition. The OCR presentation includes the methods, implementation and the outcome.

The paper ends with a detailed reflection of the work, potential future improvement and a conclusion of a summary of the work.

2.0 DATA

The dataset used in this study is composed of images and videos of students enrolled in the INM460 Computer Vision class. The data consists of two types of captures: individual persons captures, and group captures. The individual persons' captures are made up of portrait images and videos of each individual holding an A4 plain paper (landscape) with an identification number on it. The identification numbers range between 1 and 78, both inclusive. The group captures consist of images and videos of the group taken in class.

A total of 48 individuals participated in the individual captures. In contrast, the group captures consisted of the 48 individuals with other members of the class, implying not all individuals in the group captures were involved in the individual captures. Seven hundred thirty-one (731) files (images and videos) in total were downloaded from Moodle. Table 1 shows a summary of the files downloaded.

Captures	Images	Videos
Individuals	336	333
Group	31	31
Total	367	364

Table 1: Summary of files downloaded.

The videos and photos were captured using the same device (lecturer's phone), and lighting conditions were similar for all captures. The group captures were taken in the lecture theatre whiles the individual captures were in the computer lab. All the images are in jpg format and the videos in mp4 format. The duration of the videos are about 2 seconds each. The videos and photos were taken from different angles to capture enough features as possible.

The task at hand is to perform face recognition, a supervised learning, on the given data. This requires the data to be structured in other to be able to classify them. Thus, the first pre-processing is to structure the data by with their class labels (number allocated to each individual). Each of the files was allocated to a folder with their class label. Forty-eight (48) folders were created for each label, and the files sorted and distributed into them accordingly. This is the initial structuring of the data.

3.0 FACE RECOGNITION

This section details the processes related to the development of the Face Recognition function. The function(script) takes an image as input and returns a matrix: each row represents an individual identified in the image, and the three columns represent the individual's assigned label, the x and y coordinates of the middle point of the detected

face respectively. It also has a creative mode that overlays a face mask on each detected face.

The first part of this section discusses the approaches used in developing the function, followed by the implementation of these techniques and a presentation of the implementation test results.

3.1 METHODS AND TECHNIQUES

The techniques implemented are discussed in this subsection. These are presented in two main categories for clarity: feature extraction techniques and machine learning algorithms.

3.1.1 FEATURE EXTRACTION TECHNIQUES

Feature extraction in computer vision is the process of identifying visual patterns that are used to perform pattern recognition tasks such as concept classification, object detection etc. Extracting features is essential in face recognition, and this section presents the feature detection techniques used to extract features in this development.

3.1.1.1 SCALE-INVARIANT FEATURE TRANSFORM (SIFT)

Lowe[1] in 2004 came up with the Scale Invariant Feature Transform (SIFT) algorithm, a feature extraction technique which extract keypoints from an image and compute its descriptors. An object in a new image is recognised by comparing each feature of the image to the extracted keypoints base on the euclidean distance of their feature vectors. SIFT is suitable for pattern recognition because of its robustness to affine distortions, 3D viewpoint, additive noise and changes in illumination. There are four(4) main steps involved in the SIFT algorithm.

Scale-Space Extrema Detection: It is the first step in the detection of keypoints; identification of candidate points that are invariant to orientation and scale. A Difference of Gaussians (DoG) is used as an approximation to scaled-normalised Laplacian of Gaussian (LoG) to detect stable points in the scale function. The image is searched for local extrema over space and scale after the DoG has been found.

Keypoint Localisation: Once potential keypoints are established, they need to be refined to get more accurate results. Taylor series expansion is used to obtain more precise location of accurate extrema. Points of low intensity (below a threshold) at an extrema (sensitive to noise and along the edges) are rejected, and strong points remain.

Orientation Assignment: Invariance to image rotation is achieved by assigning orientations to keypoints based on local image properties. This is done by computing gradient magnitudes and directions in that region by taking a local neighbourhood around the keypoint location. An orientation histogram is then created: 36 bins covering the 360 degrees of orientation. The highest peak in the histogram and any peak above 80 percent of it are used to compute the orientation. This creates keypoints with the same scale and location but different directions and aids in stability matching.

Keypoint Descriptor: To create the keypoint descriptor, a 16x16 neighbourhood is built around the keypoint. This is further divided into 16 sub-blocks of 4x4 size. An eight(8) bin orientation histogram is created for each sub-block, thus resulting in a total of 128 bin values. These are flattened, concatenated and represented as a vector to generate keypoint descriptor.

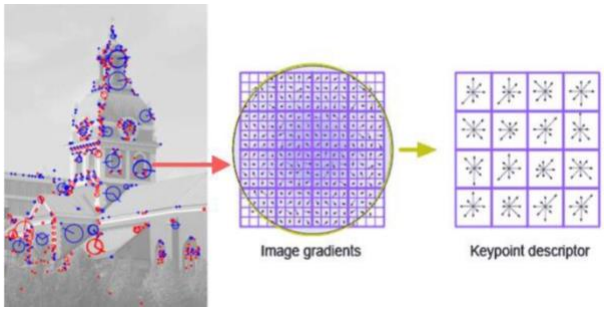

Figure 1: Creating a descriptor from an image keypoint.[1]

3.1.1.2 SPEEP-UP ROBUST FEATURES (SURF)

In 2006, Bay et al.[2] introduced SURF, which is partly inspired by the SIFT descriptor. SIFT, despite its benefits, proved to be comparatively slow and as the name implies, the standard SURF is a speeded-up SIFT. The two descriptors are very similar; the main difference is SIFT approximates Laplacian of Gaussian (LoG) with Difference of

Gaussian (DoG) whiles SURF approximates LoG with box filter in locating scale-space. SIFT advantage over SURF is because of its parallelisation and speed using integral images in computing a convolution with a box filter.

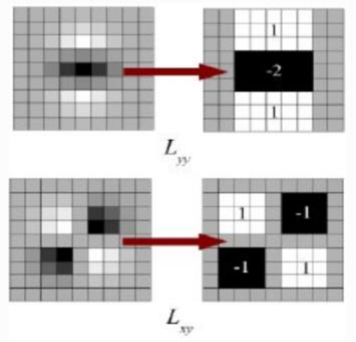

Figure 2: Top represents gaussian second-order partial derivative in the YY direction to its approximation using box filter and down represents a similar approximation in the XY direction.[3]

SURF employs wavelet responses in orientation assignment in vertical and horizontal direction for a neighbourhood of 6s (s=size). This is then plotted like figure 3 in a space. Summing of all the responses within a sliding orientation window of angle 60° (60 degrees) gives an estimate of the dominant orientation. Another essential improvement is of trace of Hessian Matrix (sign of Laplacian) for underlying keypoint.

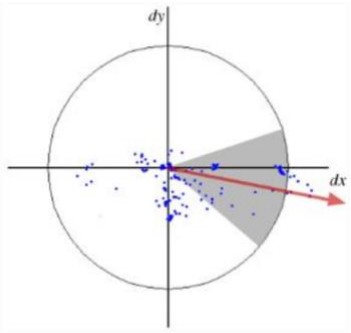

Figure 3: Space plot of wavelet responses in horizontal and vertical direction for a neighbourhood of size 6s.[3]

For feature description, a 20s x20s (s=size) neighbourhood is taken around the keypoint and subsequently subdivided into 4x4 sub-regions. A vertical and horizontal wavelet responses are computed for each sub-region and generate a SURF feature descriptor of 64 dimensions in a vector form as follows:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|).$$

3.1.2 MACHINE LEARNING ALGORITHMS

This section provides a brief description of the various machine learning (including deep learning) algorithms employed in the development of the face recognition function.

3.1.2.1 SUPPORT VECTOR MACHINES (SVM)

SVM is a supervised machine learning algorithm primarily suited for binary learning but can be extended to perform multi-class learning. SVM operates by finding a hyperplane that separates data points under certain conditions. For separable data, an infinity number of hyperplanes can be found that can separate the data perfectly. The ultimate idea is to locate the best or unique hyperplane, given the concept of large margins. In terms of binary classification, the margin is the distance between the closest data points of the two classes. This concept is illustrated in figure 4.

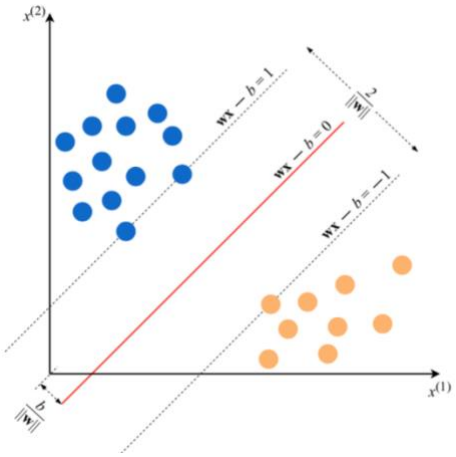

Figure 4: Margins and maximum-margin hyperplane of two classes trained SVM.[4]

SVM can be extended to deal with less perfect cases such as non-linearly separable classes. In such situations, a high loss and cost constraint hyperparameters are introduced. This controls the trade-off of ensuring sure each data point lies in the correct side of the hyperplane and increasing the size of the decision boundary.

Non-linear classification can also be efficiently performed by SVM using the kernel trick, essentially mapping inputs into higher dimensional feature spaces. Polynomial and RBF (gaussian) are the commonly used kernels. The preferred method of optimisation in SVM is quadratic programming.

3.1.2.2 LOGISTIC REGRESSION (LR)

Logistic regression (LR) is a statistical machine learning model which in its basic form, uses a logistic function for binary dependent variable learning. LR belongs to the family of generalised linear models and can be extended to model multi-class dependent variable. LR is a discriminative model as it models the conditional probability of the output given an input. LR models the dependent variable of a given input vector as a sigmoid function evaluated at the value resulting from the linear combination of the inputs. The sigmoid function restricts the output range to [0, 1].

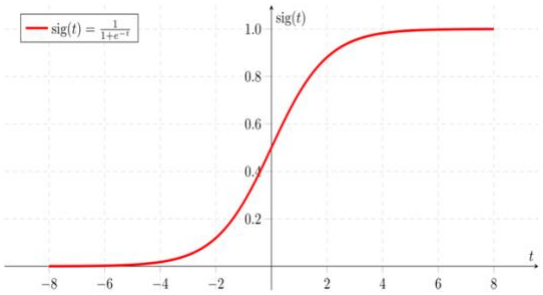

Figure 5: Sigmoid activation function.

Loss arises when maximising the log-likelihood of the data, and the binary cross-entropy loss function is often assumed. When there is no closed-form solution to the optimisation problem, optimisation algorithm such as gradient descent are typically employed.

Overfitting is tackled by employing various regularisation strategies; Lasso (L1) and Ridge (L2) are the typical ones used.

3.1.2.3 RANDOM FOREST (RF)

Random forest (RF) is a supervised ensemble learning method that operates by combining several decision trees (weak learners) and outputs the mean prediction in a regression task or mode of the classes in a classification task.[5] The training algorithm of RF is based on the general technique bagging or bootstrap aggregation.

Training is accomplished by creating a certain number of bootstraps from the given data. A decision tree is trained on each bootstrap using a random subset of independent variables. The final predictions are attained by performing a majority vote over all the learners (decision trees).

3.1.2.4 RESIDUAL NEURAL NETWORK (RESNET)

Residual Neural Network (ResNet) is a deep learning architecture introduced by He et al.[6] that supports residual learning. Deep convolutional networks solve complex tasks (increase pattern recognition accuracy) with more deep number of layers, but the accuracy starts to saturate and degrade as the network goes deeper. Residual training seeks to solve this problem. Residual learning implies subtraction of learned features from the input of that layer. ResNet addresses the problem of degrading accuracy by using shortcut connections.

Figure 6 shows the residual learning building block (shortcut connections). Basically, instead of every few stacked layers fit the underlying function H(x), the stacked layers fit the residual function F (x) = H (x) − x. ResNet exist with different layers, including 18, 34, 50, 101 and 152. It was trained on the ImageNet database designed for use in visual object recognition software research.[7]

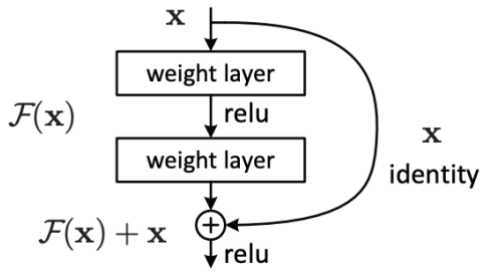

Figure 6: Residual learning building block.[6]

3.1.2.5 K-MEANS

K-means is a method of vector quantisation and an unsupervised learning algorithm. It aims to group data points into clusters in which each data point belongs to the cluster with the nearest mean. Given k clusters, the algorithm works by minimising the distortion (sum-of-squares), typically the euclidean distance.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

where, r_{nk} is a representation of assigning observation x_n to cluster K with centroid μ_k. The algorithm is optimised through an iterative process where the cost function is minimised with respect to r_{nk} with μ_k fixed and vice versa.

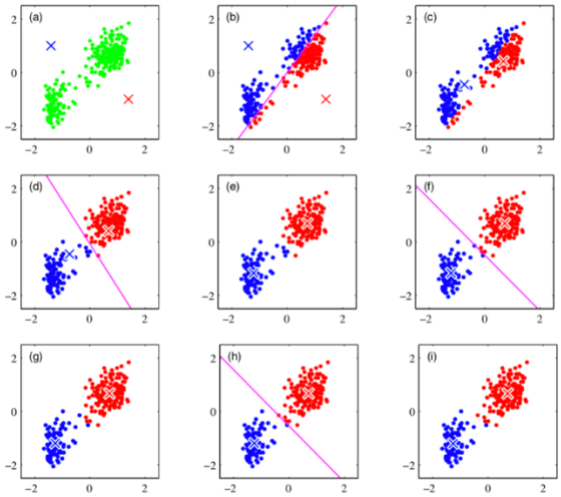

Figure 7: K-mean algorithm applied to toy dataset with two clusters (K=2).[8]

3.1.2.6 YOU ONLY LOOK ONCE (YOLO)

YOLO is a recent deep learning approach for object detection targeted for real-time processing. YOLO frames object detection by spatially separating bounding boxes and related class probabilities as a regression task. This is achieved in one evaluation by a single neural network that predicts bounding boxes from images. YOLO can be optimised on detection performance because the entire detection pipeline consists of a single neural network.

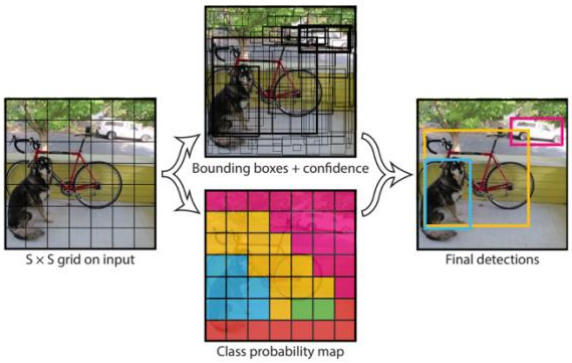

Figure 8: A unified YOLO architecture modelling detection as a regression problem.[9]

3.2 IMPLEMENTATION

This section presents the implementation details of the face recognition function using the various methods and techniques outlined earlier. This has been categorised into two phases; face detection and face classification.

3.2.1 FACE DETECTION

Many face detection libraries, including libraries provided by popular computer vision packages, were considered. They all produced similar results except the deep learning based YOLO v3 algorithm. YOLO v3 has been pre-trained for face detection on the Wider Face dataset.[10] OpenCV was used to load the pre-trained weights of YOLO v3 for face detection or inference. OpenCV has a deep neural network model that supports pre-trained deep neural network models like YOLO. YOLO v3 provided superior accuracy in face detection.

3.2.2 FACE CLASSIFICATION

After the faces detection, the next stage is to train the machine learning models to be able to predict the faces (to perform classification). Before modelling, the dataset has to be further processed to increase the quantity and introduce variety to enhance the performance of the models. Three(3) models, SVM, logistic regression and random forest, were trained using a bag-of-visual words approach. The SIFT and SURF feature extractors were employed in the bag-of-visual words. Each machine learning algorithm was trained using the SIFT and SURF feature extractors, thus, resulting in six(6) trained classifiers. Finally, a CNN was trained on the images to arrive at seven(7) classifiers. The following present details of the modelling.

3.2.2.1 DATA PROCESSING

The data is already structured; that is, the observations have been sorted into their labelled folders. The face detector is used to extract face(s) from individual photos and from the video files once every ten(10) frames. The group pictures were also essential in the training since prediction will be performed; thus, extracts from them will be beneficial in the models training. This is because the conditions in the group captures differ slightly from the individual captures since they were taken at a different location. Another reason is the difference in the size of the faces in the individual captures, and the group captures.

Some of the faces detected from the group captures were manually distributed to the appropriate folders. It was time consuming, slow and tedious to manually identify and allocate the faces to their respective labelled folder. It was also

impossible to reach the desired data quantity in this manner. Therefore to generate enough data to meet the desired number, data augmentation was applied to create the remaining data.

Before the data augmentation, all the extracted faces were resized to 80 x80 to bring the size close as it appears in the group photos. The augmentation applied included rotation of 25^{o}, horizontal flip, gaussian blur and brightness and these were applied randomly. The augmented images, though artificial and not perfect, helped to increase the dataset to the desired number. In total, approximately 47000 face images distributed evenly (to prevent imbalance) across 48 classes. Figure 9 shows examples of some augmented faces.

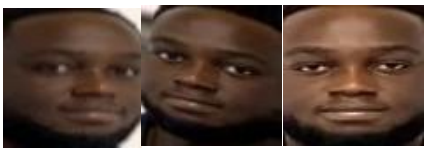

Figure 9:examples of augmented face images

3.2.2.2 MACHINE LEARNING MODELS TRAINING

This section outlines the training of the machine learning models. The models were trained using the Visual-Bag-of-Words (VOB) approach using the various feature descriptors(extractors). The process consists of four main steps:

Feature Extraction: Features extraction from the image using SIFT or SURF.

Visual Vocabulary: Visual vocabulary creation using K-means clustering with K = 100. The result is a vocabulary of 100 visual words represented by cluster centroids.

Vector Quantisation: Application of vector quantisation on every image using the nearest neighbour method to map each feature vector of the image onto the index of its closest centroid. Use it to build a histogram of features by counting the number of times each visual word occurs in the image.

Model Training: Train the classifier using the bag-of-words representation.

The training of the models was relatively long due to the size of the training data; thus, a 5-fold cross-validation grid-search hyper-parameter optimisation was adopted for model selection. Few hyper-parameters were tuned due to the relatively long training time. For SVM, the kernel type (linear and kernel) and the parameter C (constraint/regularisation parameter) were tuned. Whereas for logistic regression, only the regularisation parameter (C) was tuned. Finally, for the random forest, the number of trees, splitting criterion and the maximum number of features were tuned.

3.2.2.3 RESNET50 TRAINING

In the deep learning model selection, two main approaches were considered; building a Convolutional Neural Network (CNN) from scratch or using a pre-trained model. For the pre-trained model, Resnet50, which has 50 layers, was explored. In training the Resnet50, two transfer learning approaches were explored. In the first approach, all the weights were fixed except for the last layer and trained. In the second approach, the whole model was fine-tuned with very low learning rate in order not to change the weights too much. Both the CNN and the ResNet50 had comparatively excellent performance, but the accuracy of the CNN tends to degrade as the network got deeper. Thus, the ResNet50 was chosen for the deep learning training.

The ResNet50 was trained by fine-tuning all its weights gradually as it performed better than training only the top layer. The ResNet was loaded without its last layer and two fully connected layers with size 256 were added with ReLu actions. Dropout of rate 0.5 was introduced, and Adam optimiser with a learning rate of 1e-4 and a decay rate of 1e-6 was the optimiser of choice. Cross-entropy loss function was used, and the output size of the last layer was 48, representing the number of classes. The input size of the model was 224 x 224; thus, the images were resized to the same dimensions for compatibility.

3.2.2.4 TRAINING RESULTS

The results of the best models are shown in tables 2 and 3. In tuning the regularisation parameter(C) for SVM and LR, the figures tried were 0.01, 0.1, 1.0 and 10. For the Random Forest model, [Gini and Entropy] criterion values were tried, [10, 50, 100, 200] number of trees values were tried and [None, log2 and Auto] maximum number of features were tried. For the RESNET, the values tried for the

learning rate were [1e-1, 1e-2, 1e-3 and 1e-4] and the value tried for the decay rate were [1e-6 and 1e-7].

Classifier	Feature Descriptor	C	Criterion	Number of Trees	Maximum Features	Validation Accuracy
rbf SVM	SIFT	1.0	–	–	–	95.92
	SURF	1.0	–	–	–	96.06
Logistic Regression	SIFT	1.0	–	–	–	88.75
	SURF	1.0	–	–	–	88.90
Random Forest	SIFT	–	Gini	100	Auto	94.25
	SURF	–	Gini	100	Auto	93.60

Table 2: Parameters and results of best trained models

Classifier	Optimiser	Learning Rate	Decay Rate	Validation Accuracy
RESNET	Adam	1E-04	1E-06	99.41

Table 3: Parameters and results of best trained RESNET50

In the model performance, there was no significant difference with respect to the feature extractor/descriptor used though SURF performed slightly better in two models. In visual bag of words, spatial information is lost as it only relies on the occurrence of words to differentiate the various classes. Whiles this is a drawback, it turns out that a concept classification task like this is not influenced by this. What is essential is the presence or absence of keywords that describe the content of the image. SVM performed best because it is able to extend its decision boundary to areas of less data point and less distinction between data points. Random forest is not far off because it leverages the performance of single models to get the best performance. Logistic regression was expected to have comparable performance as the others because the BoW approach sits in a lower dimension of linearly separable manifolds. Its performance could have been improved with additional hyper-parameter tuning, but it would be computationally expensive; thus, we settled with its current performance which is good.

3.2.3 CREATIVE MODE

The creative mode, when called, overlays a face mask on the detected face. It employs the YOLO face detector to detect the face when the function is called. After the face is detected, the region of interest is extracted to overlay the mask in the right position. The pixels associated with the face mask are isolated so that the overlaid mask does not conform to the rectangular shape of the detected face. Therefore, the exact boundaries of the overlaid mask are determined, so it looks natural.

The image of the mask has a white background, and it will look weird if overlaid will look weird. Thus, the exact curvy boundaries of the mask are extracted before it will be overlaid on the detected face. To achieve this, we want only the pixels of the face mask to be visible and make the remaining pixels invisible. A final threshold of 240 (using trial and error) is applied so that all pixels with an intensity value greater than 240 will become 0 and all others become 255. This leaves only the boundaries of the face mask to remain.

Finally, the pixels of the region of interest where the mask will be overlaid has to be blacked out. This is achieved by simply applying the inverse of the mask created. The last step is finally to add the two masked versions to produce the final output image.

3.2.4 PREDICTION PIPELINE

This explains how the RecongniseFace function works when executed. It extracts the face(s) using face detection (YOLO). If a model with a corresponding feature extractor is chosen, then a BoW approach will be used to predict the label(s) of the detected face(s) using the model. For ResNet50 prediction, the feature extractor is set to ‘none’ and the classifier to RESNET. The creative mode should always be set to 0 during prediction. For creative mode set all other arguments to ‘none’ and creativeMode to 1.
The function accepts the following arguments:

- I : image path
- featureType : 'SIFT', 'SURF', 'none'
- classifierType : 'SVM', 'LR', 'RF', 'RESNET', ‘none’
- creativeMode: 0, 1

The face detector (YOLO) seems to be very accurate, although not thoroughly tested. It is able to detect faces that can barely be noticed from some blurry images.

The ResNet50 performed best of all the models as expected. Due to how the function was setup, it classifies unlabelled detected face by assigning a wrong label to it, thus, using a confusion matrix

produced inaccurate performance of the model. Therefore, to test the performance of the model, I took five group pictures and manually identified labelled faces before running the function to classify the faces. After averaging the accuracy of the function on all the five pictures, the best model (RESNET50) had 87.46% accuracy.

4.0 Optical Character Recognition

This section presents the development of the Optical Character Recognition (OCR). The goal is to develop a function, detectNum, that takes an image/video as an input and outputs a number on a paper held by the person in the image/video. The process follows a similar layout as that of the face recognition function development; summary of methods and techniques used, then how they are implemented and finally the results of the implementation.

4.1 Methods and Techniques

The following sections will give a summary of the main techniques (algorithms) used in the development. This includes HOG feature extractor, KNN algorithm and RetinaNet.

4.1.1 Histogram of Oriented Gradients (HOG)

The HOG feature descriptor, introduced by Dalal and Triggs[11], is used for the purpose of object detection. The descriptor works by counting the occurrences of gradient orientation in localised portions of an image. The image is divided into cells, and a histogram of gradient orientations is computed for the pixel in each cell. Each cell is discretised into angular bins in accordance with the gradient orientation. Adjacent cells are grouped into blocks which is the basis of grouping and normalisation of histograms. The set of block histograms represents the descriptor.

4.1.2 K-Nearest Neighbors (KNN)

KNN is a non-parametric algorithm used in pattern recognition. There is no training involved. The input is consists of the k closest training examples in the feature space. In classification, the output is a class membership, where an object is assigned to the class most common among its k nearest neighbours. In regression, the output is the average of the value of its k nearest neighbours. Figure 10 shows an example of KNN with different values of K.

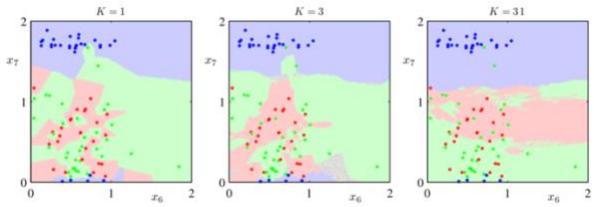

Figure 10: KNN on a bi-dimensional dataset with 3 classes with k = 1, 3 and 31 respectively.[8]

4.1.3 RetinaNet

RetinaNet is one of the best one-stage object detection algorithms proposed by Lin et al. [12] It uses Feature Pyramid Network (FPN) and Focal Loss for training. FPN produces feature maps of different scale on multiple levels in the network. The focal loss address the imbalance in the single-stage object detection where there is a large number of possible background noise and a few foreground classes. It is based on cross-entropy loss, and the loss contribution can be reduced by adjusting the gamma parameter. Figure 11 shows the RetinaNet model architecture.

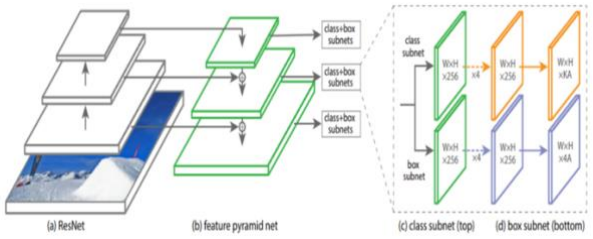

Figure 11: RetinaNet model architecture

4.2 Implementation

This section details the two main stages of the implementation of the detectNum function. The first stage is the detection of the white A4 paper held by the individual and the second stage is the detection of the number on the A4 paper. The paper detection is essential as it defines the area containing the number, thereby eliminating most of the noise that may interfere with the number detection.

4.2.1 PAPER DETECTION

Paper detection is a single-object detection where, in this case, the single object is the A4 paper. Two methods were explored in this circumstance; pre-trained models and transfer-learning. There exist no pre-trained model to detect white A4paper only, so the next obvious step was to try transfer learning, and if that was not successful, then a new model would be built from scratch. The transfer learning approach was very successful using RetinaNet, and the following outlines the data creation and model training:

Data Creation: For the data, 240 (5 for each person) of the original image data provided were picked. Since this task is a single label (object) detection, the images have to be relabelled to be able to perform single object detection. RectLabel, a labelling tool, was used to relabel the images. Labelling here refers to drawing a box around the white A4 paper in each image. RectLabel automatically converts the boxes into XML files of the same name as the input image which is acceptable input for RetinaNet. For the transfer learning data of 240 proved enough and successful.

Model Training: The model used is retinaNet, and the method is transfer learning, that is, to train the pre-trained retinaNet to leverage its already acquire knowledge to learn to classify the new data. Keras implementation of RetinaNet was used in this training.

4.2.2 NUMBER RECOGNITION

After the paper detector extracts the paper from the image, the next step is the number recognition of the number on the paper. After exploring various options, the conclusion was to extract the number, separate the digits and perform digit recognition. The following describes the pipeline for extracting the number:

- Transform detected paper into grayscale.
- Crop the edges of the paper to zoom in on the number in the middle. A cropped area of 40% was applied.
- Gaussian noise is applied to denoise the image (kernel size of 4x4)
- Inver binarisation is applied with OTSU thresholding.
- Detect edges of digits on the image using canny edge detector.
- Filter out contours that are likely not digits.
- Extract the digit from the binarised image.
- Resize digit images to 30x50.

The extracted digits were used to create a dataset of digits. The data sorted into folders labelled 0-9. Random augmentation was applied to increase the data (to around 100) and also aid the accuracy of the model. The augmentations applied include brightness, blur and rotation. The dataset was trained on a KNN classifier (K = 6). HOG feature descriptor was used for feature extraction of the digit images.

After training the model, the detectNum was constructed using the trained model. The function works by taking an input image, pre-process it and extract the HOG feature vector. To obtain the prediction, the extracted feature vector is fed into the classifier.

4.2.3 OCR RESULTS

The RetinaNet paper detector predicted right any test tried with it; however, there was no clear way to evaluate it properly. The KNN classifier had a validation accuracy of 96.64%. The overall performance of the detectNum function was 95% when tested on 200 images. The only problem with the function is when digital contours are not detected properly, the function fails to predict.

5.0 CONCLUSION

This presentation discussed various computer vision algorithms and techniques to perform face recognition and object character recognition. The face recognition involved face detection and face classification. Sift, and Surf feature extractors were tried with machine learning algorithms (SVM, LR, RF). A deep learning approach was also tried with RESNET50. The face detection algorithm used was YOLO. The OCR implementation was also built from grounds up without making use of any pre-existing methods.

One problem encountered in the face recognition function was classifying detected faces without assigned labels in group photos. The function tends

to force out a prediction for each detected face, thus, misclassifying them. In future, this can be resolved by using one-shot learning approach to recognise the faces where each detected face is run against and compare to a database of labelled individuals for similarities and make predictions with a specified threshold. Another possible future work will be extending the face recognition to real-time scenarios.

REFERENCES

[1] Lowe, D.G. (2004). Distinctive image features from scale- invariant keypoints. *International journal of computer vision*. 60(2), pp.91-110.

[2] Bay, H., Tuytelaars, T. and Van Gool, L. (May, 2006). Surf: Speeded Up Robust Features. *European Conference on Computer Vision*. (pp. 404-417). Springer, Berlin, Heidelberg.

[3] SURF, OpenCV Documentation. https://opencv pythontutroals.readthedocs.io/en/latest/py_tutor ials/py_feature2d/py_surf_intro/py_surf_intro.ht ml

[4] Burkov, A. (2018). The Hundred-page Machine Learning book.

[5] Ho, T. K. (1998). The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 20 (8): 832–844. doi:10.1109/34.709601

[6] He K., Zhang X., Ren S. and Sun J. (2016). Deep Residual Learning For Image Recognition. *The IEEE Conference On Computer Vision And Pattern Recognition.* (pp. 770-778).

[7] IMAGENET. http://www.image-net.org/

[8] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

[9] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *The IEEE Conference on Computer Vision and Pattern Recognition.* (pp. 779-788).

[10] Yang, S., Luo, P., Loy, C.C. and Tang, X. (2016). Wider Face: A Face Detection Benchmark. *The IEEE Conference on Computer Vision and Pattern Recognition.* (pp. 5525-5533).

[11] Dalal, N. and Triggs, B. (June 2005). Histograms of Oriented Gradients For Human Detection. *The International Conference on Computer Vision & Pattern Recognition. (CVPR'05) (Vol. 1, pp. 886-893).*

[12] Lin T.Y., Goyal P., Girshick R., He K. and Dollár P. (2017). Focal Loss For Dense Object Detection. *The IEEE International Conference On Computer Vision*. (pp. 2980-2988).