Student Number: 181355 & Kaggle Team Name: 181355

1. Approach

1.1. Algorithm Choice

A multi layer perceptron was chosen as an appropriate model to adequately achieve the given task. A multi layer perceptron is neural network formed of layers of interconnected perceptrons - simple logical statements [8]. A supervised learning algorithm from the scikit learn toolkit was chosen [7]. More specifically, the MLPClassifier class from the toolkit was used.

1.2. Algorithm Theory

This works using a neural network that is trained using back propagation. The algorithm takes as input an array of floating point numbers and the classification labels that apply to those numbers [9]. Typically, a stochastic gradient descent algorithm is used in the training of this network. This is where the data is passed into the network one row at a time, with the network processing the data and activating the individual neurons(perceptrons) until an output is produced at the end of the network. The output is then compared to the expected output (the given labels) and an error is calculated - this error is then fed back through the layers and the weights of the individual neurons are updated. This is known as back propagation and the process is repeated for all samples in the data [3].

1.3. Notable Assumptions and Comments on Data

The multi layer perceptron assumes that a relationship exists between features - it uses these relationships to detect patterns and predict labels. Each of the samples was the result of an image being passed through two different image representation techniques with the results concatenated - analysing concatenated results together could lead to difficulty in determining relationships due to differing representation and the different granularity of each.

The original data set supplied consists of 247 samples (209 memorable, 38 non) with 4608 features. This data set is therefore considered low sample high dimensional data - a problem when training a neural network as over-fitting and high variance gradients occur [6]. This problem needed be resolved before a suitable model could be trained.

2. Methodology

2.1. Data Processing, Confidence Based Sampling and Imputation

The data set was split into CNN and GIST representations. This had a two fold benefit - reduced dimensionality and increased likelihood of relationships being detectable. An arbitrary decision was taken to to solely consider the CNN representation moving forward - as the increased granularity of this representation could potentially lead to a better classifying accuracy.

To increase sample size, the supplied incomplete data set was combined with the original data set. The incomplete data set was appended to the end of the original then the entire set was under-sampled using the confidence labels and test proportions. All samples with a non memorable label and only those samples with a memorable label that possessed a confidence in the label of 100% were preserved. This had three benefits - it reduced the number of predicted values in the data; improved the quality of the data such that most samples are certainly either memorable or not; finally, it minimally addressed the domain balance issue between Brighton images and London images by decreasing the imbalance between labels. After under-sampling, the labels and features were split into separate collections.

Next, the missing values in the incomplete data were replaced with predicted values. The scikit toolbox offers multiple options for missing value imputation [11] but considerations had to be made for the available amount of computational power and time. The K Nearest Neighbours algorithm was selected [12] and used with an N of 5.

2.2. Feature Selection and Preprocessing

Feature selection was applied to reduce the dimensionality of the data by removing less relevant features this decreases the chance of overfitting and reduces the training time as well as improving the accuracy of the model [4]. The Extra Trees Classifier from scikit was selected as an appropriate choice as feature selection [1].

Multi layer perceptrons are sensitive to feature scaling [9] so all features were scaled to lay between 0 and 1 [14]. No other preprocessing was performed in order to preserve the relationships between features as the data was originally in image format - so removing correlation through whitening could remove important information.

2.3. Domain Adaptation and Test Label Proportions

The test proportions were examined and compared to the training data proportions. This showed that the training data favoured the memorable label vastly over the non memorable label. This meant that the number of zeros compared to ones in the training data needed to be increased in order to ensure the training domain more closely resembled the testing domain.

Oversampling the minority class using SMOTE as detailed in the article at was used to tackle the domain adaptation (along with previous undersampling) [5].The plan go-

ing forward was to train two models - one trained on augmented data and one trained on non-augmented data due to the risk of overfitting.

2.4. Model Selection, Testing and Final Training

A baseline model was formed through trial and error and throughout the processing mentioned in previous subsections, the model was tested using k-fold cross validation with 5 folds [10].

Model selection was carried out using randomised search with cross validation [13]. The aim was to select hyperparameters for the model that would lead to the best performance as given by accuracy. Randomized search was selected over exhaustive grid search due to lack of computing power.

The model was trained with the entire training corpus and then predictions were created by passing in the testing features (scaled) to the trained model using the predict function. A prediction was generated for each model.

3. Results and Discussion

3.1. Data Augmentation

An experiment was carried out varying the sampling strategy for SMOTE and examining the change in accuracy. The extremely high CV accuracy with the auto strategy suggested overfitting - it is not reasonable to achieve a high accuracy such as this with the supplied dataset. As such, 0.44 was selected as the sampling strategy as this represented a compromise between working towards solving the domain strategy whilst reducing overfitting due to oversampling.

Augmentation Strategy A	CV Accuracy
None	70.44%
Auto	85.68%
0.3	69.81%
0.35	72.68%
0.4	73.53%
0.43	75.93%
0.44	77.43%
0.45	75.53%

Table 1: Augmentation Strategy with CV Accuracy

3.2. Model Selection Results

Table 2 shows that the best parameters to choose were batch size 1472, layer sizes (1644,1000), alpha of 0.0001.

Table 3 shows that layer size: (1468,1000,500), batch: 1316, alpha 0.0112 was the best option for the model trained with non-augmented data with 0.717 score.

Layer Size	Batch Size	Alpha	Mean Test Score
1000	1472	0.0223	0.77316146
1644	1472	0.0778	0.76890615
1644	786	0.0556	0.75796204
1644,1000	1472	0.0001	0.77316888
1644	271	0.0667	0.76829824
1644	1644	0.0334	0.76708985
1644	786	0.0667	0.76039551
1644,1644	1300	0.0001	0.76647639
1644	271	0.0778	0.73910594
1644	1472	0.0112	0.76891356

Table 2: Augmented data model selection results

Layer Size	Batch Size	Alpha	Mean Test Score
1468, 1000, 500	1316	0.0112	0.71730863
1468, 1000, 500	404	0.0667	0.68325555
1468	252	0.0001	0.71391423
500	1468	0.0001	0.70778946
1468, 1000, 500	1316	0.1	0.70642195
1468, 1000	252	0.0889	0.6751039
500	252	0.0223	0.70710223
1468,1468	404	0.1	0.66216248
500	1164	0.0445	0.7139212
1468,1000,500	708	0.0001	0.71526317

Table 3: Non-Augmented data model selection results

3.3. Critique of Strategies Chosen and Room For Improvement

Taking into account the incomplete data improved training of the model as neural networks are more suited to a higher sample size [6]. Undersampling using confidence improved the quality of the dataset as this removed any samples where the classification had a level of doubt. Taking into account the domain adaptation problem in addition to the test label proportions had questionable benefit to the end test domain accuracy - this is most likely due to the SMOTE generation.

The main area where there is the possibility of improvement is in model selection. Random Search was used for only a few iterations due to the lack of computational power available. An exhaustive search would have been a better choice in terms of accuracy. Gist Data could have been considered, with the same strategies applied the GIST data would have had a higher sample count than feature count - it would no longer have been high dimensional low sample size data and could potentially have performed better in training. This approach would be improved by creating more samples- this could be achieved using the original images and applying image augmentation techniques to generate more samples [2].

References

[1] AlindGupta. Ml — extra tree classifier for feature selection. Available at https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/. 1

[2] Alpha26. Python — data augmentation. Available at https://www.geeksforgeeks.org/python-data-augmentation/. 2

[3] J. Brownlee. Crash course on multi-layer perceptron neural networks, Aug 2019. Available at https://machinelearningmastery.com/neural-networks-crash-course/. 1

[4] J. Brownlee. Feature selection for machine learning in python, Dec 2019. Available at https://machinelearningmastery.com/feature-selection-machine-learning-python/. 1

[5] J. Brownlee. Smote for imbalanced classification with python, April 2020. Available at https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/. 1

[6] B. Liu, Y. Wei, Y. Zhang, and Q. Yang. Deep neural networks for high dimension, low sample size data. pages 2287–2293, 2017. 1, 2

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 1

[8] G. Saporito. What is a perceptron?, Sep 2019. Available at https://towardsdatascience.com/what-is-a-perceptron-210a50190c3b. 1

[9] scikit learn. 1.17. neural network models (supervised). Available at https://scikit-learn.org/stable/modules/neural_networks_supervised.html. 1

[10] scikit learn. 3.1. cross-validation: evaluating estimator performance. Available at https://scikit-learn.org/stable/modules/cross_validation.html. 2

[11] scikit learn. 6.4. imputation of missing values¶. Available at https://scikit-learn.org/stable/modules/impute.html. 1

[12] scikit learn. sklearn.impute.knnimputer. Available at https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html#sklearn.impute.KNNImputer. 1

[13] scikit learn. sklearn.model_selection.randomizedsearchcv. Available at https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html. 2

[14] scikit learn. sklearn.preprocessing.standardscaler. Available at https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. 1