Introduction:

With the advancements in technology and the fast-paced development in the machine learning/deep learning domains; we have seen a formidable change and increased accuracy in how images are processed, categorized and sorted. Cloud computing and availability of faster computing devices has allowed us to scale and optimize our validation procedures. There has been significant work in the natural language and text domains however, it has been a recent endeavor where machine learning has been applied to images and visuals to identify patterns and harness useful information from "visual" data. Visualization is at the core of human perception and is essential to form contextual information.

From identifying faults in bridges, maintenance schedules for hardware, health and wellness domains, image recognition is bound to increase the utility it provides multiple times in an industry which is already a multibillion-dollar industry.This study aims at using multiple classifiers on greyscale images (28*28) pixels and identifying which classifier gives a more accurate classification with time being an important metric in gauging their performance.

Methods:

Let's explore the data set before looking at the optimal solution for classifying the images into the ten categories identified.
The train data is shaped into 30,000 images of 28*28-pixel dimensions and the test data is shaped into 5,000 images of 28*28 dimensions.
The methodology below has three sections

1- Preprocessing
2- Classification

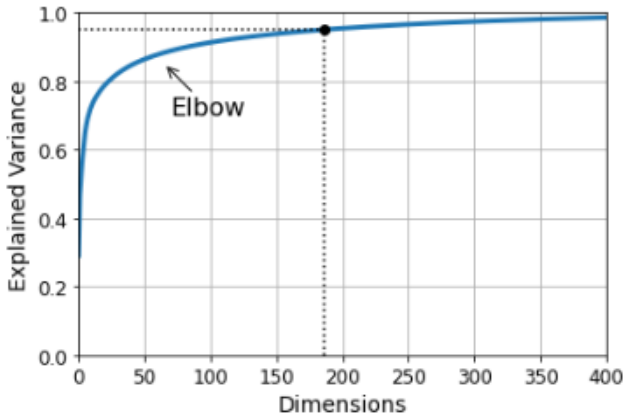
Preprocessing:

Normalization

For preprocessing we have normalized the data using the scikit MinMaxScaler. The reason for choosing the MinMaxScaler is because we have a range of data, and we want to scale it between 1 and 0 so the MinMaxScaler transforms the data to a range of 0 and 1 rather than the Maximum and Minimum of the data range. Normalization allows for better visualization of the data and helps in maintaining data integrity. This allows for elimination of unstructured data and errors in data such as duplicates.

PCA

Principal component analysis is a very useful process where a dataset with a lot of features can be reduced to a smaller dataset with a reduced number of features that can carry the same or slightly less variance data. This is done by reducing the data into its elements which carry the most information called principal components. Principal component analysis can effectively reduce the processing time and further optimize the training dataset for processing, but it is possible that it will have an impact on the accuracy of the data set given the number of features in the images data set is more than 700. We performed a PCA on our dataset and maintained integrity to explain 95% of the variance with reduction in the number of variables to 186 form 700.



Overfitting

Overfitting is a common and statistical problem faced with machine learning algorithms when processing data and classifying. Overfitting generally leads to the model predicting accurately on the given data set but performing with poor accuracy on unseen data which leads to inaccurate classification. The analytics applied to the data were applied with this in mind so that overfitting is avoided.

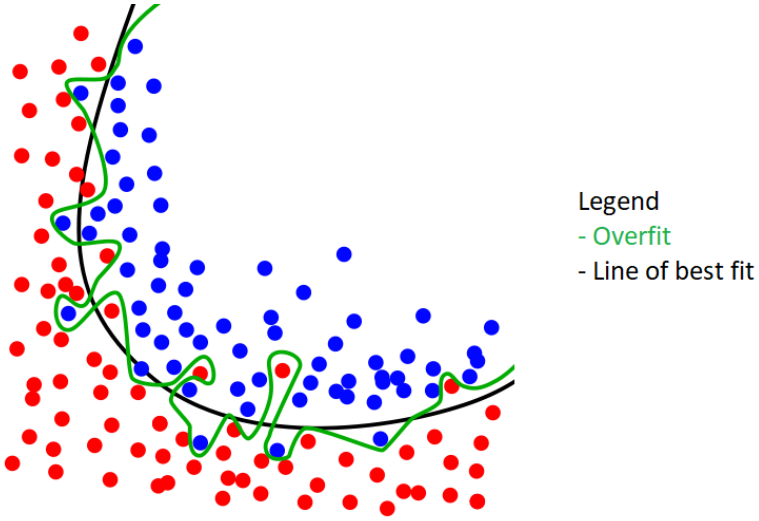


Fig 1 - Overfit vs Line of best fit [1]

Feature Scaling

Machine learning algorithms can be optimized and sped up using feature scaling with tools as the MinMaxScaler. Some algorithms use the K-Nearest Neighbor as their computational factor. Feature scaling can lead to lower processing times with the gradient descent. If the range of the parameters is large it will take a longer time for the algorithm to process all the errors to reach the base of the convex. Therefore a shorter range allows for faster processing.

The comparison of results with and without normalization (Minmax Scaling) are not present in this study but empirical evidence suggests that classification models are faster with normalized data than data that has not been feature scaled.

Experiments and Results:

Once the preprocessors are in place and the data is optimized; the goal is to classify images from the new data into 10 distinct categories accurately based on the training data.
For this purpose, once the data was preprocessed and ready to be classified, we applied the following classifiers to process the data set:

1- Nearest Neighbor
2- Logistic regression
3- Decision tree
4- Support Vector Machine

And

5- Random Forest for Ensemble

K-Nearest Neighbor:
A classification algorithm by design KNN allows for the segregation and largely accurate classification of data and can be used in conjunction with regression. It takes the straight line distance or the Euclidean distance to classify which data point from the training data set is similar to the test data set and hence classifies them as such. The number of K used to run the KNN analysis was maintained at 5.
Using the KNeighborsClassifier from sklearn.neighbors library we were able to classify 85% of the data accurately.

Logistic regression:
Logistic regression is an easy to access tool and the data can be prepared very much like linear regression in the sense that the assumptions for both the logistic and linear regression are very much the same. Here in our study, we have used logistic regression from the scikit library that has a prebuilt definition for logistic regression. Later in the analysis, hyper parameter tuning is done to assess the ideal number of iterations.
Using logistic regression we were able to classify 84% of the data accurately.

Decision tree:
Decision trees in data science are drawn upside down with the root or the base at the top with the branches growing downwards. Decision trees in machine learning are prone to overfitting the data if they branch too much with multiple successive nodes/edges. For this, the data has to be pruned to a specific "level" to achieve logical information. Decision trees can be used for classification with regression. The data is split in succession continuously according to a specific criteria or parameter.
Using the decision tree algorithm from the scikit-learn library we were able to achieve an accurate classification of 73%

Ensemble; Random Forest:
Another method of ensemble is random forest where instead of having one decision tree we take multiple decision trees and use their aggregate computations for classification. Random forest

takes data in a very similar method to bagging by using subsets of the original dataset called bootstraps. These trees can be pruned to avoid overfitting and combined to give us greatly accurate classifications.
Random Forest can also be considered another form of Bagging where data subsets are used to create predictions/classifications.
Using this method we were able to achieve an accuracy of 76%

Hyper parameter tuning:

Hyper parameter tuning refers to finding the best possible parameters for getting accurate classification. For the purpose of this study we have used grid search cross validation to optimize our classifiers. The classifiers and ensemble method used were tuned for the optimal parameters and their results were gauged.
Through hyper parameter tuning, the best parameters are the number of neighbors equal to 5 and using Manhattan distance as the Minkowski metric as the optimal parameter for KNN classifier.
For logistic regression, lbfgs solver is chosen to handle multiclass problem, in this case, classify the dataset into 10 classes. With a L2 regulation, which is also known as ridge regression, shrink the less contributive coefficient down to zero. Hence, L2 regulation can prevent model overfitting problem. Set Max_iter to 2000 makes the solver converge until 2000 iteration in order to prevent warning.
For decision tree classifier, setting minimum sample split to 30 to prevent overfitting the dataset. Setting the criterion to entropy so that the function to measure the quality of the model is to calculate the information gain.
For Random Forest classifier, set the maximum depth to 70 to prevent model overfitting. Number of trees set to 400 to estimate the prediction.

The table below details the classifiers and their performance:

	Hyperparameter	Accuracy %	Time (seconds)
K-Nearest Neighbour	N_neighbor : 5 , P: 1	85%	105
Logistic regression	Max_iter : 2000	84%	53
Decision Tree	Criterion: Entropy, Splitter = 'Random', Min_samples_split: 30	78%	3
Random Forest	Criterion: Entropy, Min_samples_split: 4, n_estimators :400	88%	7
SVC		88%	128
			296

Accuracy Result on Kaggle

Classifier	Accuracy on Kaggle
K-Nearest Neighbour	77%
Logistic regression	76.3%
Decision Tree	69.8%
Random Forest	57.8%
SVC	9%

Discussion and Conclusion:

The different classifiers were applied in tandem with the preprocessing steps to ensure an optimal mix of efficiency vs performance. A number of experiments and runs were made to see the best performance of the classifiers in terms of accuracy. The accuracy percentage numbers can be further assessed to get a clearer understanding of the predicted data by running a precision, recall and F1 test. These tests can be run on a macro level to understand the overall picture and then at a micro level (hoodies, pullover, shoes etc.) to gauge the performance of the classifiers at a micro level.
This can lead to practical application of the classifiers where they are successful and further testing and research can be done for those micro segments

Performance

The predictions did not vary much in range for the classifiers and the ensemble method used varying from 78% decision tree to 88% using the SVC and Random Forest model. A Naïve Bayes classifier was also run to gauge the data however it did not perform well and hence it was removed from the final discussion.
SVM and Ensemble methods seem to outperform K Nearest Neighbor and Logistic regression. It can be assumed that the dataset due to its large complexity and number of features is not ideal for KNNs lazy algorithm. Logistic regression can possibly have lost its accuracy numbers since the data was normalized and then a PCA analysis was done to reduce the number of features. Thus it can be assumed that logistic regression may have not performed well for these reasons compared to Random Forest or SVM.
The target of this assignment was to process and identify the best classifier using the pre-existing libraries within the python language. These tests were successfully conducted and gauged in comparison to each other by optimization of their key features to obtain accurate classifications for their test data based on the training data of 30,000 grey scale images.

Training time

One of the goals of the assignment was to achieve the classification efficiently within a short duration. Using our initial preprocessing of the data with normalization and PCA, we were able to keep the training time short. However, optimizing the parameters was time intensive.

Critical reflection

The team consisted of two students; Nick and Haseeb. While the team worked with dedication and effort towards achieving the best possible results, it was hampered due to the fact that we connected at a later date after the assignment was published and are both studying remotely (China and Australia respectively).
Nick experienced in coding ran the chunk of the code where Haseeb prioritized on writing the report and compilation.

The analysis of these classifiers has led to a deeper appreciation of the ingenious thought process and effort put in by the developers of libraries such as scikit learn and other classifiers. These tools that are not only immensely helpful to new learners but also help in understanding and performing tasks efficiently which would otherwise be extremely tedious.

The fact that the assignment was challenging in terms of its goals and targets made it tough but also a great learning experience with classification of images giving it a relatable experience and allowing us to connect with real world examples as well.
The concluding remarks must mention that further tests need to be done on perhaps larger data sets to ascertain the performance of the classifiers in comparison to each other. However, with the given data set it is apparent that SVM and Random Forest are optimal performers for classification in a data set this size with multiple features of varying qualities.

Appendix

Instructions on how to run code

1. To read the data from given file, go to section 3.1 to load the dataset for training purposes.
2. Before training the model with our data, pre-process the data first to avoid any bias. Go to section 4.1.1 for data preprocessing and run code block for normalization and PCA.
3. Fit the reduced dimension datasets to normalized dataset.
4. Run all the code block in 4.1.2 section to get all the classifier model accuracy score.
5. Run all the code block in 4.1.3 to tune all the parameters for the classifier.
6. Run all the code block in 4.1.4 to compare the accuracy results for the multiple classifiers with the best parameters in tune.
7. Predict the output of the given test file and run code block 5.

The hardware used to run these computations is as follows:

Processor: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz
RAM: 16 GB – 64 Bit operating system
GPU: NVIDIA GEFORCE 1080TI boosted at 2000 MHz

Citation

En.wikipedia.org. (2019). Overfitting. [online]
Available at: https://en.wikipedia.org/wiki/Overfitting#/media/File:Overfitting.svg [Accessed 25 Apr. 2019].
Asaithambi, S. (2017). Why, how and when to scale your features. [Blog] Grey Atom. Available at:
https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e [Accessed 24 Apr. 2019].
k-nearest neighbors algorithm - Wikipedia. En.wikipedia.org. (2022). Retrieved 7 April 2022, from https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.
K Nearest Neighbor | KNN Algorithm | KNN in Python & R. Analytics Vidhya. (2022). Retrieved 7 April 2022, from https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/.
Molnar, C. (2022). 5.2 Logistic Regression | Interpretable Machine Learning. Christophm.github.io. Retrieved 7 April 2022, from https://christophm.github.io/interpretable-ml-book/logistic.html.
Raschka, S. (2022). How to Select Support Vector Machine Kernels - KDnuggets. KDnuggets. Retrieved 7 April 2022, from https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html.
SVM RBF Kernel Parameters With Code Examples - DZone AI. dzone.com. (2022). Retrieved 7 April 2022, from https://dzone.com/articles/using-jsonb-in-postgresql-how-to-effectively-store-1.