COMP 6721 Phase 2 Report

Group - ResNet

Vibhor Gulati
40238072

Dhyey Nilesh Doshi
40244534

Namrata Pankajkumar Brahmbhatt
40233323

I. Abstract

In the project we aim at solving the problem of Image Classification using different methods of training the model. Same data has been trained using 3 different methods, namely supervised learning, semi-supervised learning and Convolutional Neural Network (CNN). The report entails the methodologies and the details of how each methodology aims at reaching the same goal of Image Classification in their own way. A collection of 3200 images of five different classes was collected from different sources and a major portion was used for training purposes and the rest for test purposes. The results for each of the methods were evaluated using similar metrics, mainly accuracy, precision, recall and F1-measure. The analysis of the results shows that CNN proved to be the best solution among the three followed by supervised learning. Semi-Supervised learning performed the worst and Supervised learning provided a middle ground beside having the simplest approach.

II. Introduction

With data being generated in a huge amount there needs to be a process which helps us figure out the relevance and more use cases of it. One way of figuring out the relevance of the data in different circumstances is by labelling it appropriately so we know which data can be used in which context and make decisions backed by statistics of the operations performed on the data.

Unfortunately, the amount of data being generated is not small and hence labeling is not an easy task and there are often privacy restrictions as well which prevent data from being labeled at the source where it is generated. This leads to an abundance of unlabeled data which is of little significance till we label it correctly. This is a major challenge with respect to Image Classification.

A further challenge in image classification is making sure that models perform well when applied to fresh data. When it comes to photos, overfitting is evident since models are more likely to memorize training data than to discover patterns that may be applied generally.

Managing computational resources and model complexity presents CNNs with additional significant challenges. CNNs are incredibly helpful for classifying images, but their training and inference processes demand a large amount of computing resources.

There have been some solutions proposed for the challenges addressed above which have improved the performance of the model trained. Techniques like data augmentation involve creating additional training examples by applying transformations to existing labeled images, such as rotations, flipping, and color adjustments which help in increasing the diversity of the training set, reducing overfitting, and improving the model's generalization capabilities [1]. Though this is simple to implement and improves the generalization of the model, it is still not enough to address the data scarcity.

Transfer learning is another solution for limited labeled data as it involves pre-training a model on a large dataset and then fine-tuning it on the target dataset with limited labeled data. This approach allows models to use previously learned features from the source dataset, reducing the required amount of labeled data for the target task. The limitation of transfer learning is that the performance is based on the similarity between the source and target datasets. If the datasets are quite different then the performance of the model would not have significant improvement since it is majorly trained on a different type of images.

In this project we have aimed to solve the problem of Image Classification using three different solutions which vary widely in their approach towards the problem. We also aim to improve prediction by tuning different hyperparameters.

Supervised Learning: This is the most basic solution where we train the model on the available labeled data and then predict the classes of the unlabeled data using the previous learnings. In the project, we train the model on about 80% of the images and then predict the rest of the images. The major benefit of this solution is that it is very easy to implement and doesn't contain many complexities and hence doesn't require a lot of computational resources. On the flip side, a large amount of labeled data is required for this which is a major challenge. It also may not generalize well with limited data.

Semi-Supervised Learning is another solution to the problem of Image Classification. It is mostly used in cases where the unlabeled data is of a significant amount. The basic idea behind this is that the model is trained on the available labelled data and then the model pseudo-labels the rest of the unlabeled data. A sub-set of the pseudo labels is selected based on confidence and similar factors and merged with the labelled data on which the model is trained again until we label the entire dataset. Though this seems to be a good approach and is widely used with huge datasets, it is often possible that the pseudo labels generated are incorrect and when merged with the labelled data, the labelled data will consist of wrong labels which significantly hinders the future performance of the model since our model is sensitive to the pseudo labels. In practice, semi-supervised learning is used in combination with other solutions since majority of the data available is unlabeled and the basic principle of semi-supervised learning helps resolving that issue correctly if the model has been trained on a good number of labelled images. In the project, we took 20% of total data as our initial labelled data and pseudo

labelled the rest of it and merged it with the previous one using k-best criterion till we labelled the entirety of the dataset.

The third and the final solution for solving the problem of Image Classification is **using CNNs** which have revolutionized image classification by automatically learning hierarchical feature representations. CNNs have high accuracy and are scalable to large datasets. To provide a higher level of accuracy it is computationally expensive and requires large, labeled datasets and at the same time is also prone to overfitting. In the project we tried various CNN architectures, experimenting with different number of layers and hyper-parameter tuning to optimize the performance.

The results of the experiments indicated that CNNs outperformed other solutions for the best performance for image classification but also required the greatest number of resources. Semi-supervised learning was the worst solution as expected since it is less effective with smaller datasets. Supervised learning, though simplest of all, provided the middle ground for working on small datasets. The combination of these techniques, tailored to specific datasets and resource constraints, will yield the optimal results for the problem.

Related Work:

a. Abeer Aljuaid & Mohd Anwar [2] published a survey in which they discuss the applications of supervised learning in medical imaging. The survey discusses various techniques including supervised learning using CNNs, transfer learning, data augmentation, etc. They discussed the challenges associated with applying deep learning techniques for medical imaging, with the most significant being the lack of labeled datasets. It has also been discussed that automated medical imaging can reduce the workload from the pathogens and radiologists and still provide precise diagnoses.
b. For Semi-Supervised Learning, Laine et al [3] present a simple and efficient method for training deep neural networks in semi-supervised settings, introducing self-ensembling. The method predicts unknown labels using the network's outputs under different conditions. The method also reduces classification error rates, improves accuracy, and shows good tolerance to incorrect labels. It has been used to set new records for reducing the (non-augmented) classification rate in SVHN with 500 labels and CIFAR-10 with 4000 labels.
c. Baron et al [4] used supervised learning for the detection of late blight in potato crops using CNN enabling the farmers to identify the disease non-invasively by simple inputting an image of the leaf of the crop to the trained model. To offset the limited number of images available, they augmented the data to generate more of them using a script. They also pre-processed the images and transformed the images from RGB to HSV and then performed segmentation on the dataset prepared. After feature extraction and dimensionality reduction using PCA, they classified the images using a linear kernal and CNN. 93% accuracy was achieved using HSV as compared to 88% achieved using the colored models.

III. METHODOLOGIES

i. Dataset and Preprocessing

The dataset comprises images from data sources [5][6][7][8][9] for five different classes: Bar, Casino, Restaurant, Library, and Hospital. The number of images collected for the classes Bar, Casino, Restaurant, Library and Hospital are 602, 651, 640, 600 and 698 respectively in the final dataset. The images collected were in fact far greater than the ones selected but a selection was made among those which were not outliers and were in fact correct images of the class. Restaurants for example had more than 117,000 images in the downloaded dataset but most of them were of some food dish and not an image showing a restaurant. Images for Bar were collected from source [6] and those for Library were collected from source [7]. Images for Casino were collected from sources [6] and [9] while those for Restaurant were collected from sources [6] and [8]. Images for Hospital were collected from sources [5] and [6].

The images are loaded and preprocessed using three steps.

Firstly, images are loaded from a specified directory. During this process, images not in RGB format are filtered out, ensuring consistency in the input data. Secondly, each image is resized to 256x256 pixels to standardize the input dimensions. Lastly, pixel values are normalized to the range [0, 1] by dividing by 255.0.

The dataset was split into test and training in the ratio 17:83 for supervised and semi-supervised learning using Decision Trees and split into test, validation and training in the ratio 1:1:8 for the CNN model.

The 256x256x3 images are flattened into 1D arrays of size 196,608. According to Culurciello, et al. [10] transforming 3D into a series of 1D arrays applied across the channels resulting in an approximate 2 times increase in evaluation speed compared to the standard model. The flattened model maintains or even surpasses the accuracy of traditional models, while utilizing only one-tenth of the parameters.

Principal Component Analysis (PCA) according to Greenacre et al. [11] is a flexible statistical technique used to refine data into its key elements, known as principal components. This method allows for an approximate reconstruction of the original data table using only these significant components. It even further reduces the number of features, significantly lowering the computational cost and potentially improving model performance.

ii. Decision Tree Model

A decision tree is a versatile supervised machine learning technique which is used for both classification and regression purposes[12]. It works by creating a model that predicts the value of target variable using a sequence of simple decision rules generated from the input features of the data. These rules are represented by a tree-like structure known as the decision tree. This method divides the data into multiple sections resulting into overall piecewise constant approximation. The decision trees are widely used as they are simple and interpretable, providing easy visualization of decision-making process.

A DecisionTreeClassifier can perform multi-class

classification on a dataset. The decision tree model was selected for its straightforward approach to classification, interpretability, and effectiveness in handling both categorical and numeric data.

The architecture of a decision tree model is described as below:

Root node: The root node depicts the entire dataset, which is divided into subsets of data for further decision-making splits. Hence this node serves as a starting point for the decision-making process.

Internal nodes: These nodes represent features of the dataset. At each internal node, a decision is made based on the value of the feature, resulting in multiple branches. The feature and value chosen for splitting are defined by a criterion that quantifies the split's purity, such as Gini impurity or information gain (entropy).

Splitting criteria: This represents the splitting criterion for splitting the dataset to make a decision. The default is Gini criterion for the Gini impurity to evaluate the quality of the splits.

Branches: Each node splits into 2 or more branches representing the outcomes of the decision at a particular node.

Leaf nodes: These nodes are also known as terminal nodes representing the final decision for the classification. Each leaf node is assigned an appropriate class label which is determined by the majority class of the instances in the particular division of data.

iii. CNN Model

Convolutional Neural Networks have become an anchor in the field of image classification due to their capability to automatically and adaptively learn spatial hierarchies of features. Our model comprises several key layers: convolutional layers, activation functions, pooling layers, fully connected layers, and dropout for regularization. The architecture is initiated with four convolutional layers with increasing numbers of filters (32, 64, 128, 256). These layers extract features from specific areas of the input image by using filters, commonly referred to as kernels. All layers have their kernel sizes set to 3x3. This enables the model to identify minute patterns in the image. To preserve the spatial dimensions during the convolution process, a padding of 1 is also used.

After each convolutional layer is a max pooling layer. Through down sampling, this layer lowers the dimensionality of the feature maps. The feature maps turn into half as broad and half as large, but the most significant activation within the pooling window was retained due to the application of a 2x2 max pooling operation. This procedure lowers computational costs and helps in the control of overfitting.

After every convolutional layer, the model applies "Leaky ReLU" as the activation function. In contrast to the conventional ReLU function, "Leaky ReLU" permits a tiny non-zero gradient for negative inputs, avoiding "dead zones" where gradients are zero and trapping the model. This helped us speed up the convergence and the training process.

The data passes through fully connected layers after the convolutional and pooling phases. By figuring out the connections between the extracted features and the associated class labels, these layers carry out classification. The flattened feature maps are transformed into a vector of 512 units by the first fully connected layer and an output size of 5 by the second layer, which represents the number of classes.

Before the first FC layer, there is a dropout layer. In order to keep the model from overfitting on the training set, dropout randomly removes a percentage of activations during training. This improves the model's ability to generalize to new images.

With its well-structured layers and sophisticated activation and regularization methods, our suggested model offers an efficient and successful method for classifying images. Because of the way its architecture balances performance and complexity, it can be used in a wide range of image classification applications.

iv. Optimization Algorithm

Machine learning models are optimized by finding the optimum set of hyperparameters that result in the best performance on validation data. The model’s hyperparameters try to prevent overfitting and enhance generalization.

Hyperparameter tuning in supervised decision tree model implementation in our work is done using manual Grid Search for parameter optimization. This way of parameter tuning automatically tests each parameter combination hence simplifying the optimization process and saving time and effort. It evaluates the model’s performance throughout the parameter grid, which helps in identifying the optimal combination that improves model’s performance. This in turn improves the tuning process’ efficiency and decreases the possibility of human error.

The parameters that are used in our model’s optimization process are:

max_depth: The max_depth controls the maximum depth of the tree, hence it prevents overfitting by reducing the complexity of the model.

min_samples_split: The min_samples_split defines the minimum number of samples required to split an internal node.

ccp_alpha: Complexity parameter for minimal cost-complexity pruning(CCP). It reduces the size of the decision tree by pruning the branches that have little importance. It evaluates the branches of the trees for pruning based on the cost complexity, which is evaluated based on the number of leaves and the misclassification cost.

The hyperparameter tuning for the semi-supervised model for the decision tree is done in a similar way as the supervised learning approach however, on a very limited amount of labeled data (20%). The parameters max_depth, min_samples_split, and ccp_alpha are used in the Grid Search method for the parameter optimization process. Consecutively, the decision tree model is trained using the best parameters chosen.

Optimizing the performance of Convolutional Neural Networks involves tuning of hyperparameters. For the our

model, we validated and optimized several hyperparameters, including learning rate scheduling, the number and placement of conv and pooling layers, activation functions, and batch sizes.

To enhance our model, we utilized a combination of grid search and random search techniques for hyperparameter tuning. For learning rate, we experimented with SGD and Adam and added an LR scheduler. The number and placement of pooling layers were adjusted using random search to ascertain the most effective configuration. Various activation functions, including ReLU and LeakyReLU, were compared using grid search to evaluate their impact on model performance. Additionally, different batch sizes were tested, with random search determining the batch size that best balanced training time and model accuracy.

To assess the performance of the CNN model and the effectiveness of hyperparameter tuning and optimization algorithms, we considered several evaluation metrics. Accuracy was the primary metric, representing the proportion of correctly classified instances. Cross-entropy loss was monitored to evaluate the model's fit to the training data and to detect overfitting by comparing training and validation loss. Precision, recall, and F1 score were used to provide a detailed view of the model's performance across individual classes. Additionally, learning curves depicting training and validation accuracy and loss over epochs were analyzed to ensure that the model was not overfitting and that it converged appropriately.

IV. RESULTS

i. Experiment Setup

Supervised/Semi supervised:

A grid search approach was employed to identify the optimal hyperparameters for the Decision Tree classifier. The parameters tuned included the maximum depth of the tree (max_depth), the minimum number of samples required to split an internal node (min_samples_split), and the complexity parameter used for Minimal Cost-Complexity Pruning (ccp_alpha). The grid search explored the following ranges:

max_depth: [3, 4, 5, 7, 10, None]

min_samples_split: [2, 3, 5, 10, 20]

ccp_alpha: [0.0, 0.1, 1.0]

CNN:

We explored learning rates within the range of 0.0001 to 0.01, step sizes ranging from 4 to 10 and epochs between 10 and 20. These ranges were chosen to capture a broad spectrum of learning dynamics, ensuring that the model converges efficiently without overshooting or stagnating.

We compared the performance of various activation functions, including ReLU, LeakyReLU where the LeakyReLU activation function with a negative slope of 0.01 was found to be particularly effective in preventing the dying ReLU problem. Manual grid search was employed to systematically evaluate the impact of each activation function on model performance.

Batch sizes were tested in the range of 16 to 128. Smaller batch sizes generally provide more accurate estimates of the gradient, while larger batch sizes benefit from faster convergence. Our best model used 32 as the batch size.

ii. Main Results

We used grid search to find the best parameter value for hyper-parameter tuning.

For supervised learning, the best results were obtained as 0.39 as average precision, 0.38 as average recall and 0.37 as average f1-score. Figure 1 depicts the results of the experiment. Figure 2 shows the confusion matrix for supervised learning.

Accuracy: 0.3848987108655617
Classification Report:

	precision	recall	f1-score	support
Bar	0.52	0.36	0.43	102
Casino	0.35	0.61	0.45	111
Restaurant	0.32	0.34	0.33	109
Library	0.28	0.17	0.21	102
Hospital	0.49	0.42	0.45	119
accuracy			0.38	543
macro avg	0.39	0.38	0.37	543
weighted avg	0.39	0.38	0.38	543

Figure 1

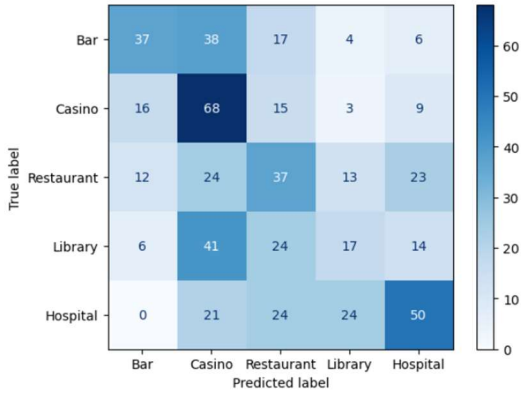

Figure 2

For semi-supervised learning with the same dataset, the best result obtained was 0.31 as average precision, 0.31 as average recall and 0.26 as average f1-score. This is low but it is expected since it is a small dataset.Figure 3 shows the results of the experiment with same dataset as that for supervised learning.

Accuracy: 0.30939226519337015
Classification Report:

	precision	recall	f1-score	support
Bar	0.25	0.26	0.26	102
Casino	0.44	0.40	0.42	111
Restaurant	0.20	0.20	0.20	109
Library	0.26	0.29	0.27	102
Hospital	0.43	0.38	0.40	119
accuracy			0.31	543
macro avg	0.31	0.31	0.31	543
weighted avg	0.32	0.31	0.31	543

Figure 3

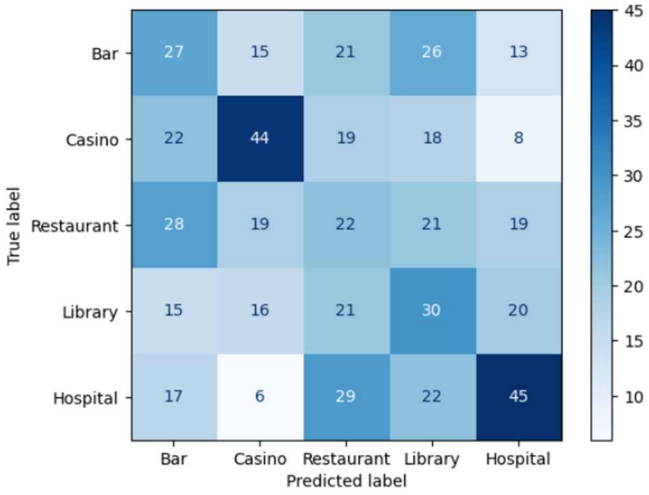

Figure 4

For CNN model, our final model was running 12 epochs and the accuracy for the training data on 12th epoch reached 98.5%. The validation loss was reported as 1.3533 and Accuracy for the validation data reached 67.4%. Figures 5 and 6 show loss and accuracy for the training and validation datasets.

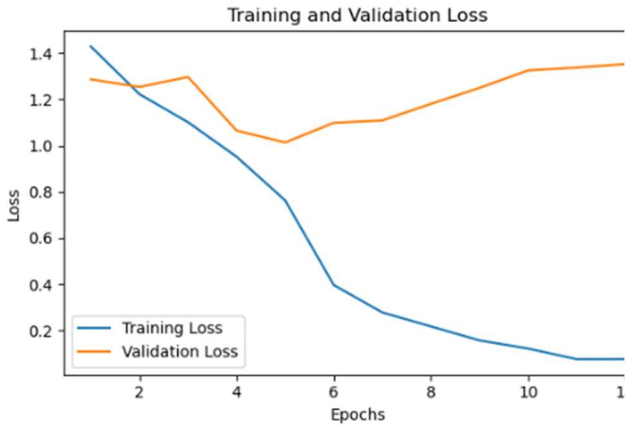

Figure 5

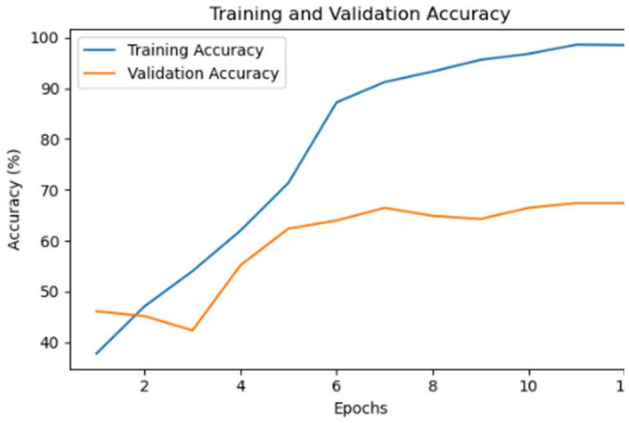

Figure 6

Test loss of 1.1968 was obtained and Test accuracy was reported as 66.88%. Average precision, average recall and average f1-score were all reported as 0.67. Figure 7 shows the Classification report and Figure 8 shows the Confusion matrix.

```
Classification Report:
              precision    recall  f1-score   support

         Bar       0.51      0.43      0.47        63
      Casino       0.74      0.78      0.76        67
    Hospital       0.81      0.80      0.81        70
     Library       0.81      0.74      0.77        58
  Restaurant       0.48      0.58      0.53        62

    accuracy                           0.67       320
   macro avg       0.67      0.67      0.67       320
weighted avg       0.67      0.67      0.67       320
```

Figure 8

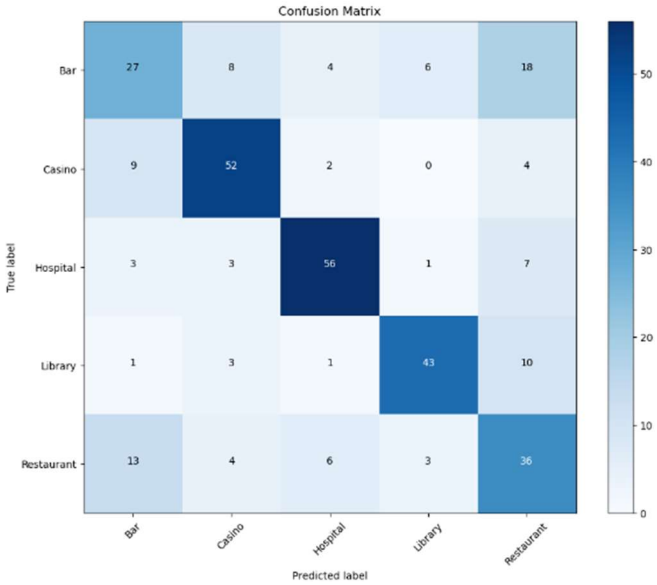

Figure 9

We also tested the model by downloading images from the internet and the pre-trained model was able to predict them with good accuracy.

iii. Ablative Study

The ablative study involves systematically tweaking different hyperparameters to understand their impact on the performance of the models. This section details the ablation results for supervised learning (decision tree), semi-supervised learning, and Convolutional Neural Network (CNN) models.

Supervised Learning:

Tree Depth:

Experiment: Tested tree depths of 3, 4, 5, 7, 10 and unlimited.

Observation: Shallower trees (depth 5) showed a higher bias but were less likely to overfit. Deeper trees (depth 15) incorporated more data subtleties but overfitted on training data, resulting in less generalization. An ideal depth of 10 balances bias and variance.

Semi-Supervised Learning:

Pseudo-Labeling Threshold:

Experiment: Varied the confidence threshold for pseudo-labeling from 0.5 to 0.9.

Observation: A threshold of 0.5 included too many incorrect pseudo-labels, harming performance. Increasing the threshold improved the quality of pseudo-labels but reduced the number of usable unlabeled samples. A threshold of 0.7 provided a good balance.

Convolutional Neural Network:

Number of Epochs:

Experiment: Ran training for 10 to 20 epochs.

Observation: Fewer epochs (10) were insufficient for convergence. Too many epochs (20) led to overfitting. Training for 12 epochs yielded the best performance with early stopping based on validation loss.

V. REFERENCES

[1] Krizhevsky et al. (2012) applied extensive data augmentation to the ImageNet dataset to boost the performance of AlexNet, leading to a significant improvement in image classification accuracy.

[2] Aljuaid, A., Anwar, M. Survey of Supervised Learning for Medical Image Processing. SN COMPUT. SCI. 3, 292 (2022). https://doi.org/10.1007/s42979-022-01166-1

[3] Laine, Samuli, and Timo Aila. "Temporal ensembling for semi-supervised learning." arXiv preprint arXiv:1610.02242 (2016).

[4] Suarez Baron, M.J.; Gomez, A.L.; Diaz, J.E.E. Supervised Learning-Based Image Classification for the Detection of Late Blight in Potato Crops. Appl. Sci. 2022, 12, 9371. https://doi.org/10.3390/app12189371

[5] Mittal, S. Places, Version 10. Retrieved June 1, 2024 from https://www.kaggle.com/datasets/mittalshubham/images256/version/10

[6] Ahmad, M. MIT Indoor Scenes, Retrieved June 1, 2024 from https://www.kaggle.com/datasets/itsahmad/indoor-scenes-cvpr-2019/data

[7] a Joson, N. Places-2_MIT_Dataset, Version 2, Retrieved June 1, 2024 from https://www.kaggle.com/datasets/nickj26/places2-mit-dataset/version/2

[8] Anh, V. Restaurants, Retrieved June 3, 2024 from https://www.kaggle.com/datasets/airbornbird88/restaurants

[9] images.cv | Labeled image datasets, Retrieved June 1, 2024 from https://images.cv

[10] Jin, Jonghoon, Aysegul Dundar, and Eugenio Culurciello. "Flattened convolutional neural networks for feedforward acceleration." arXiv preprint arXiv:1412.5474 (2014)

[11] Greenacre, M., Groenen, P. J., Hastie, T., d'Enza, A. I., Markos, A., & Tuzhilina, E. (2022). Principal component analysis. Nature Reviews Methods Primers, 2(1), 100.

[12] https://scikit-learn.org/stable/modules/tree.html#tree

[13] M. Hammad Hassan, Using Grid Search For Hyper-Parameter Tuning (2023) https://medium.com/@hammad.ai/using-grid-search-for-hyper-parameter-tuning-bad6756324cc (Last accessed June 6, 2024)