Master's degree in Computer Engineering for Robotics and Smart Industry

Machine Learning & Artificial Intelligence

Comparison and analysis between different techniques for classification

Enrico Bonoldi Luca Ponti

July 2024

Contents

1 Introduction

This report explores various feature extraction techniques and classification methods applied to a dataset of food images available on Kaggle[1].

Image classification is exploited in many fields such as medical, car industry and security and many more. Nowadays deep convolutional neural networks play a fundamental role providing a more robust method to classify images.

We will compare different techniques such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) for feature extraction, followed by classification using K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Stochastic Gradient Descent (SGD) and K-means. Then we will run the same tests on features extracted via the VGG16 convolutional neural network.

We expect that using the CNN will lead us to better performances increasing accuracy and precision without compromising simplicity(since the neural network comes pre-trained).

1.1 Methodology

All the work was done using python and Jupyter notebooks and it is available at `https://github.com/bonoldie/ML_exam.git`. Machine learning algorithms were provided by the Scikit-learn[2] library, the VGG16 CNN was provided by PyTorch[1].

2 Dataset selection

The choice of dataset is fundamental to the success of any project. A well-selected dataset that is high-quality, representative, balanced, and appropriately processed ensures that the model can learn effectively and generalize well to new data.
Our dataset contains 10 classes of food with a training set of 1500 elements (150 images per class) and a validation set of 500 elements (50 images per class). The dimension of each image varies and must be taken into account in the feature extraction phase.
The dataset is available on Kaggle at `https://www.kaggle.com/datasets/msarmi9/food101tiny`

[1]`https://www.kaggle.com`

Figure 1: Ice-Cream

Figure 2: Bibimbap

Figure 3: Tiramisù

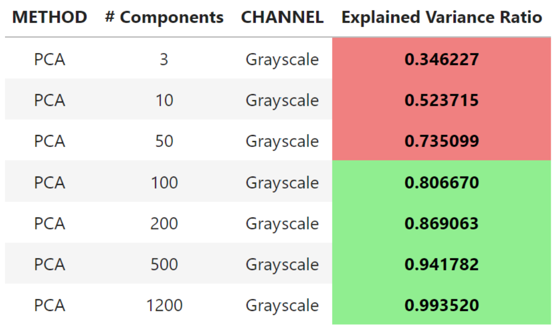
3 Dimensionality Reduction

Feature extraction is a crucial process in machine learning that involves transforming raw data into a set of meaningful features. These features serve as the input to machine learning algorithms, allowing them to learn patterns and make predictions.

3.1 Techniques for Feature Reduction

- Statistical Methods
  - Mean and Standard Deviation: Calculation of basic statistics like mean, variance, and standard deviation for different segments of data.
  - Histogram Features: Analysis of the distribution of data using histograms.
- Dimensionality Reduction Techniques
  - **Principal Component Analysis (PCA)**: Reduce the dimensionality of data by projecting it onto the principal components that capture the most variance.
  - **Linear Discriminant Analysis (LDA)**: Similar to PCA, but also considers class labels to maximize class separability.
  - **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: A technique for visualizing high-dimensional data by reducing it to two or three dimensions.
- Automatic Feature Reduction
  - **Neural Networks**: Uses neural networks to automatically extract features from raw data. For example, convolutional layers in CNNs extract hierarchical features from images. In this report, we will also use this technique to improve classification performance.

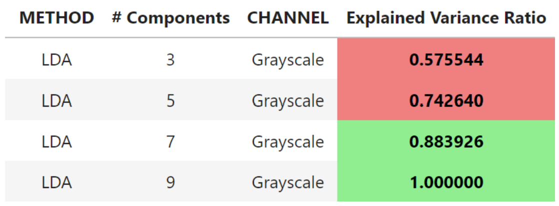
Following this initial overview of key machine learning techniques, we now delve deeper into the core of our project. As previously mentioned, we employed principal component analysis (PCA) as our first method for feature extraction. Initially, we utilized gray scale images for simplicity. We experimented with varying the number of principal components, ranging from as few as 3, which proved insufficient for practical use, to as many as 1200, which captured all the information in the images. To understand the coverage of the features, we analyzed the Explained Variance Ratio, which measures the proportion of the total variance in the original dataset explained by each principal component.
As shown in Figure below, a minimum of 100 components is required to reach 80% coverage.
Additionally, we found that analyzing one of the three color channels or the gray scale image yields a similar explained variance ratio.
Ultimately, using 1200 components, we achieved 99% coverage, indicating that all the features of the image are described by the principal components.



METHOD	# Components	CHANNEL	Explained Variance Ratio
PCA	3	Grayscale	**0.346227**
PCA	10	Grayscale	**0.523715**
PCA	50	Grayscale	**0.735099**
PCA	100	Grayscale	**0.806670**
PCA	200	Grayscale	**0.869063**
PCA	500	Grayscale	**0.941782**
PCA	1200	Grayscale	**0.993520**

Figure 4: Explained Variance Ratio of PCA

In parallel, we applied linear discriminant analysis (LDA) for feature extraction varying the number of features from 3 up to 9. It's important to note that LDA is a supervised method that leverages class labels to maximize the separation between different classes while minimizing the variation within each class.



METHOD	# Components	CHANNEL	Explained Variance Ratio
LDA	3	Grayscale	**0.575544**
LDA	5	Grayscale	**0.742640**
LDA	7	Grayscale	**0.883926**
LDA	9	Grayscale	**1.000000**

Figure 5: Explained Variance Ratio of LDA

The final method we employed is t-Distributed Stochastic Neighbor Embedding (t-SNE). That is not strictly a feature extraction technique, but is an

unsupervised method that reduces the dimensionality of the dataset to 2 or 3 dimensions.
This algorithm models points such that neighboring objects in the original high-dimensional space remain close in the reduced-dimensional space, and distant objects stay far apart, preserving the local structure.

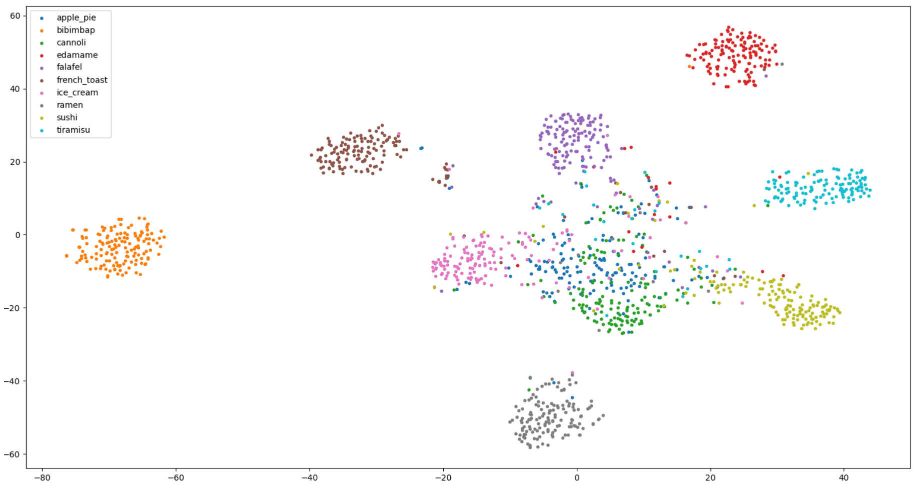


Figure 6: t-SNE 2D

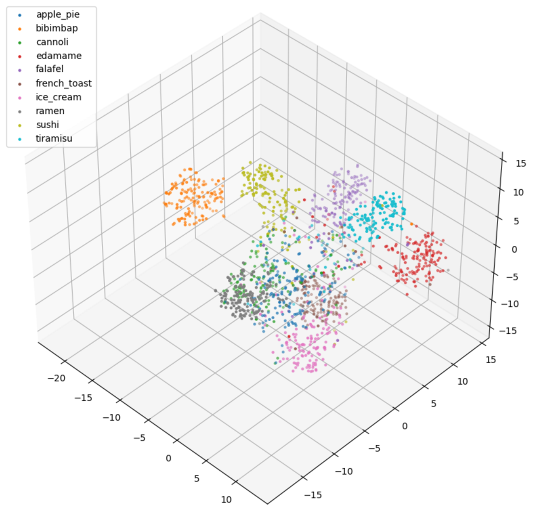


Figure 7: t-SNE 3D

Our results with t-SNE varied based on the initial feature reduction method.

When using PCA-reduced features, the t-SNE results were sub optimal due to the high number of initial features, resulting in less significant dimensionality reduction. Conversely, when using LDA-reduced features, the t-SNE visualization improved. Starting with 9 classes and reducing them to 3 dimensions provided a better separation, as the difference in dimensionality was less drastic and the features were more representative.

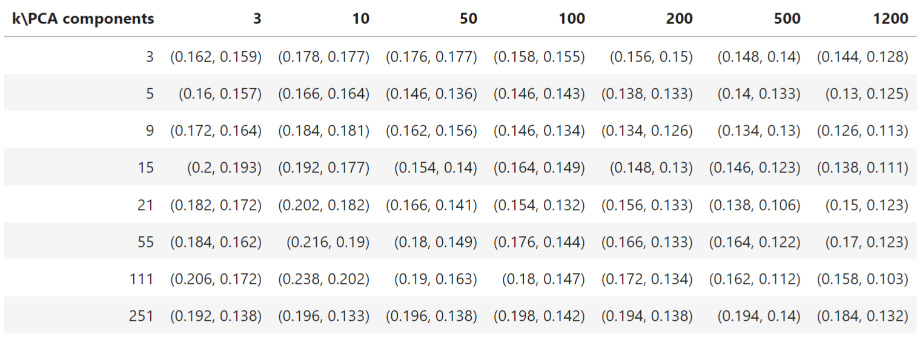
4 Model selection

There are several types of classification methods in machine learning, each with its own strengths and weaknesses.

Following the nature of the algorithm, we choose different metrics to evaluate our results. For the supervised algorithms we use accuracy and f_1 score instead for the unsupervised algorithms we use adjusted random score[2].

Here the classification methods that we used in our project:

- **K-nearest neighbors**: Despite applying PCA for feature extraction, we observed significantly low accuracy and precision.
  Varying the value of K, which is user-defined, did not give to us satisfactory results; the highest f_1 score was only 20%, this because PCA, being an unsupervised learning technique, prioritizes maximizing data variance rather than optimizing class separability. This approach can lead to a loss of classification-relevant information, thereby impacting the effectiveness of KNN in achieving higher precision.

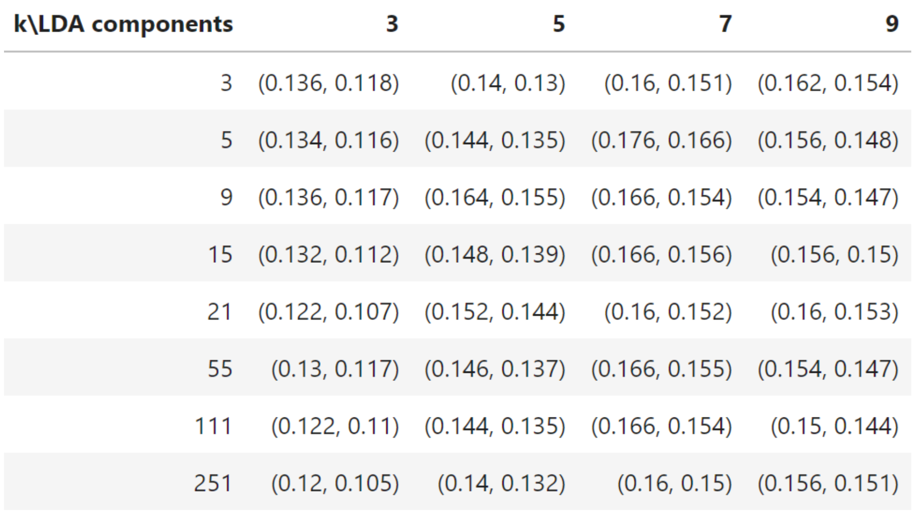


k\PCA components	3	10	50	100	200	500	1200
3	(0.162, 0.159)	(0.178, 0.177)	(0.176, 0.177)	(0.158, 0.155)	(0.156, 0.15)	(0.148, 0.14)	(0.144, 0.128)
5	(0.16, 0.157)	(0.166, 0.164)	(0.146, 0.136)	(0.146, 0.143)	(0.138, 0.133)	(0.14, 0.133)	(0.13, 0.125)
9	(0.172, 0.164)	(0.184, 0.181)	(0.162, 0.156)	(0.146, 0.134)	(0.134, 0.126)	(0.134, 0.13)	(0.126, 0.113)
15	(0.2, 0.193)	(0.192, 0.177)	(0.154, 0.14)	(0.164, 0.149)	(0.148, 0.13)	(0.146, 0.123)	(0.138, 0.111)
21	(0.182, 0.172)	(0.202, 0.182)	(0.166, 0.141)	(0.154, 0.132)	(0.156, 0.133)	(0.138, 0.106)	(0.15, 0.123)
55	(0.184, 0.162)	(0.216, 0.19)	(0.18, 0.149)	(0.176, 0.144)	(0.166, 0.133)	(0.164, 0.122)	(0.17, 0.123)
111	(0.206, 0.172)	(0.238, 0.202)	(0.19, 0.163)	(0.18, 0.147)	(0.172, 0.134)	(0.162, 0.112)	(0.158, 0.103)
251	(0.192, 0.138)	(0.196, 0.133)	(0.196, 0.138)	(0.198, 0.142)	(0.194, 0.138)	(0.194, 0.14)	(0.184, 0.132)

Figure 8: KNN with PCA

As expected, also with LDA the results are very poor, where we reach a f_1 score of only 16% with 7 components:

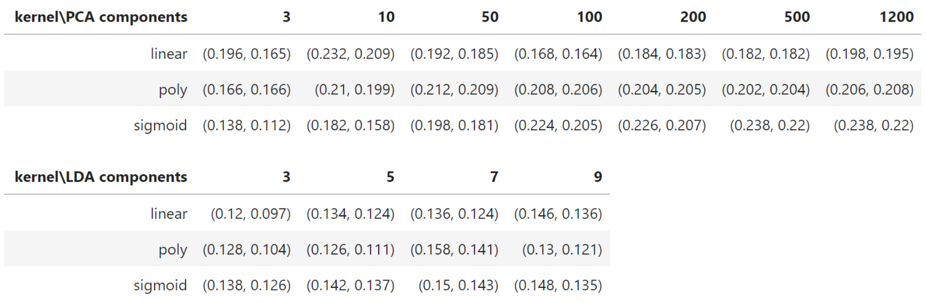
[2]from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html: The Rand Index computes a similarity measure between two clustering by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clustering.



k\LDA components	3	5	7	9
3	(0.136, 0.118)	(0.14, 0.13)	(0.16, 0.151)	(0.162, 0.154)
5	(0.134, 0.116)	(0.144, 0.135)	(0.176, 0.166)	(0.156, 0.148)
9	(0.136, 0.117)	(0.164, 0.155)	(0.166, 0.154)	(0.154, 0.147)
15	(0.132, 0.112)	(0.148, 0.139)	(0.166, 0.156)	(0.156, 0.15)
21	(0.122, 0.107)	(0.152, 0.144)	(0.16, 0.152)	(0.16, 0.153)
55	(0.13, 0.117)	(0.146, 0.137)	(0.166, 0.155)	(0.154, 0.147)
111	(0.122, 0.11)	(0.144, 0.135)	(0.166, 0.154)	(0.15, 0.144)
251	(0.12, 0.105)	(0.14, 0.132)	(0.16, 0.15)	(0.156, 0.151)

Figure 9: KNN with LDA

- **Support Vector Machine**: Our exploration into SVM did not produced any better results compared to KNN. Our investigation revealed that the selection of kernel types plays a role in performance. Moving beyond the basic linear kernel, experimenting with more sophisticated kernels such as poly or sigmoid demonstrated varying degrees of effectiveness. In fact, we reach a f_1 score of 22% with 1200 components from PCA



kernel\PCA components	3	10	50	100	200	500	1200
linear	(0.196, 0.165)	(0.232, 0.209)	(0.192, 0.185)	(0.168, 0.164)	(0.184, 0.183)	(0.182, 0.182)	(0.198, 0.195)
poly	(0.166, 0.166)	(0.21, 0.199)	(0.212, 0.209)	(0.208, 0.206)	(0.204, 0.205)	(0.202, 0.204)	(0.206, 0.208)
sigmoid	(0.138, 0.112)	(0.182, 0.158)	(0.198, 0.181)	(0.224, 0.205)	(0.226, 0.207)	(0.238, 0.22)	(0.238, 0.22)

kernel\LDA components	3	5	7	9
linear	(0.12, 0.097)	(0.134, 0.124)	(0.136, 0.124)	(0.146, 0.136)
poly	(0.128, 0.104)	(0.126, 0.111)	(0.158, 0.141)	(0.13, 0.121)
sigmoid	(0.138, 0.126)	(0.142, 0.137)	(0.15, 0.143)	(0.148, 0.135)

Figure 10: SVM with PCA and LDA

- **Stochastic Gradient Descent**: The Stochastic Gradient Descent is a supervised machine learning technique. It's an iterative algorithm that updates the model parameters one data point at a time. With this method we reach a f_1 score of 19% with PCA and only 15% with LDA.

loss\PCA	3	10	50	100	200	500	1200
modified_huber	(0.162, 0.124)	(0.162, 0.138)	(0.168, 0.151)	(0.194, 0.187)	(0.164, 0.155)	(0.182, 0.181)	(0.152, 0.148)
log_loss	(0.148, 0.092)	(0.186, 0.176)	(0.184, 0.174)	(0.156, 0.149)	(0.198, 0.19)	(0.162, 0.159)	(0.16, 0.155)
hinge	(0.144, 0.128)	(0.158, 0.134)	(0.19, 0.173)	(0.16, 0.151)	(0.186, 0.174)	(0.174, 0.172)	(0.162, 0.16)

loss\LDA	3	5	7	9
modified_huber	(0.094, 0.075)	(0.13, 0.113)	(0.134, 0.119)	(0.146, 0.139)
log_loss	(0.134, 0.103)	(0.122, 0.114)	(0.146, 0.131)	(0.158, 0.144)
hinge	(0.132, 0.111)	(0.148, 0.133)	(0.152, 0.13)	(0.134, 0.117)

Figure 11: SGD with PCA and LDA

- **K-means**: K-means is a clustering algorithm. It clusters data by trying to separate samples in k groups of equal variance. With this method we did not even reach 3% of f_1 score, with both feature extraction algorithms.

algorithm\PCA	3	10	50	100	200	500	1200
lloyd	0.016	0.020	0.024	0.028	0.024	0.020	0.033
elkan	0.027	0.022	0.025	0.019	0.023	0.021	0.023

algorithm\LDA	3	5	7	9
lloyd	0.006	0.004	0.013	0.007
elkan	0.006	0.009	0.017	0.002

Figure 12: K-MEANS with PCA and LDA

The classification problem, until now, presents evident challenges, as reflected by the poor results obtained.
Initially, we considered the possibility that the quality of the dataset was to blame, so we changed the dataset several times and also the subjects in the photos, but nothing changed significantly. So, we took the initial dataset of food and thought about the real problem. While it's true that improving the dataset by using higher quality images and reducing noise could help, the core issue lies in the nature of the images themselves. We are working with gray scale or single-channel color images, which significantly limits the available information for each photo.
As illustrated in the image below, elements from different classes can appear very similar in gray scale; for example, shapes like triangles are indistinguishable based solely on their outlines. Without color information, the algorithm struggles to differentiate between these elements, leading to frequent misclassifications and consequently poor accuracy and precision.

Figure 13: apple pie

Figure 14: french toast

4.1 Multiclass classification

Multiclass classification is a type of classification where the goal is to assign instances to one of three or more classes. Two common strategies for handling multiclass classification are ”one-vs-one” and ”one-vs-rest”

- **One-vs-one**: In the one-vs-one approach, a separate binary classifier is trained for every pair of classes. If there are $\mathcal{C}$ classes, this results in $\frac{C(C-1)}{2}$ classifiers where one is trained to distinguish between two specific classes, ignoring all others.

- **One-vs-rest**: Train a binary classifier for each class, distinguishing that class from all other classes combined. This results in $\mathcal{C}$ classifiers where each one is trained to recognize one class against the backdrop of all other classes.

For our classification task, we used the one-vs-one approach with SVM and KNN which resulted in better performance.

4.2 Feature extraction with Neural Net

The solution that we provide for the classification problem is to use a neural net for feature extraction: in particular we use the pre-trained neural net VGG16 composed by 13 convolutional layers and 3 fully connected layers which is used for tasks like classification and object detection. VGG16 is able to classify 1000 images of 1000 different categories with 92.7% accuracy, it is one of the popular convolutional neural networks for image classification and it is shipped with most of the machine learning libraries.
The input image has dimensions 224x224x3, allowing us to process colour images (three channels, RGB). The output is taken from the last 2D max pooling layer which has dimensions 7x7x512.

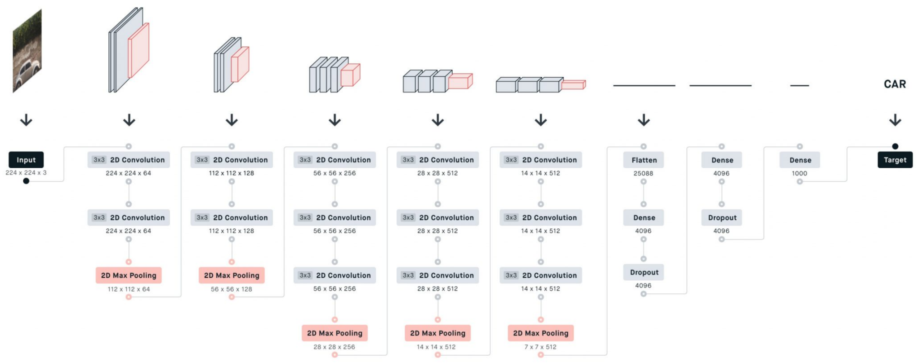


Figure 15: VGG16

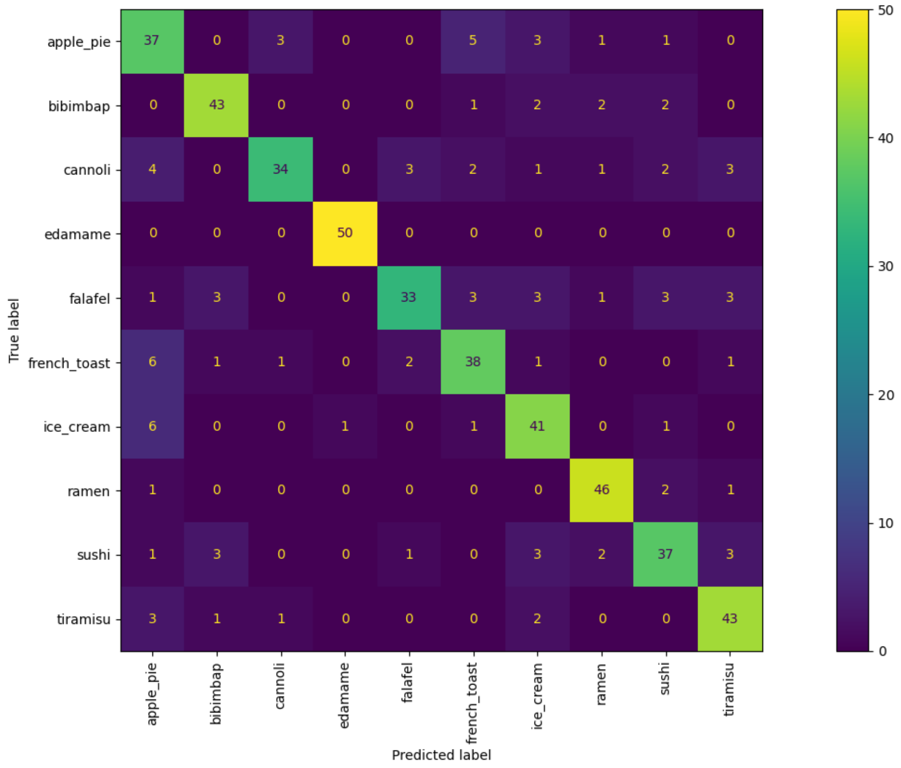
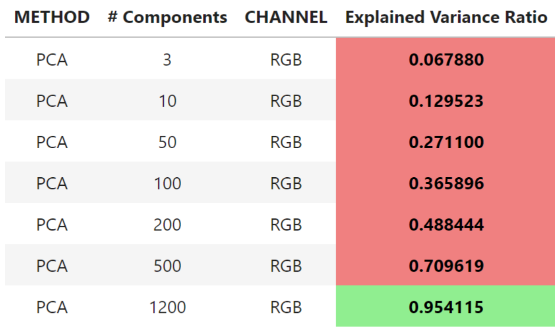


Figure 16: SGD classifier of non-reduced VGG16 feature vectors

4.3 Dimensionality reduction

Now that we have extracted the features using the convolutional neural network, we reapply the previously employed techniques: PCA and LDA for dimensionality reduction.

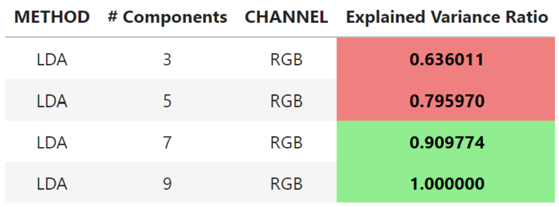
- **Principal Component Analysis**: Unlike before, here the number of features to achieve, at least, 80 per cent coverage is much higher than before, because we are dealing with RGB images and not greyscale images, which therefore have three times as many features.



METHOD	# Components	CHANNEL	Explained Variance Ratio
PCA	3	RGB	**0.067880**
PCA	10	RGB	**0.129523**
PCA	50	RGB	**0.271100**
PCA	100	RGB	**0.365896**
PCA	200	RGB	**0.488444**
PCA	500	RGB	**0.709619**
PCA	1200	RGB	**0.954115**

Figure 17: Explained Variance Ratio PCA with VGG-16

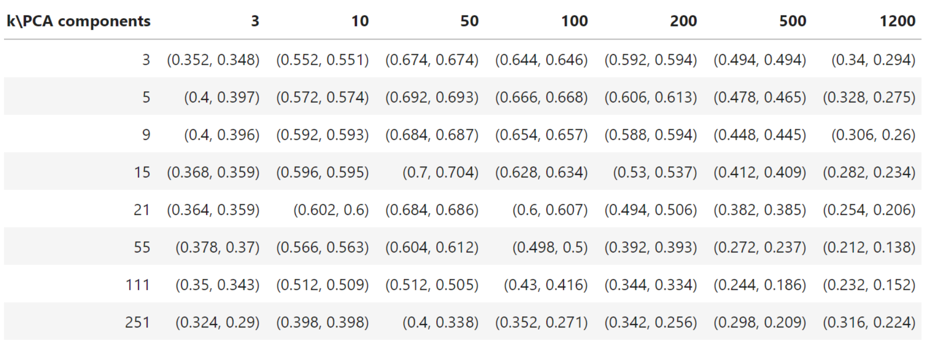
- **Linear Discriminant Analysis**: The results with LDA are similar to the one that we have obtain before:



METHOD	# Components	CHANNEL	Explained Variance Ratio
LDA	3	RGB	**0.636011**
LDA	5	RGB	**0.795970**
LDA	7	RGB	**0.909774**
LDA	9	RGB	**1.000000**

Figure 18: Explained Variance Ratio LDA with VGG-16

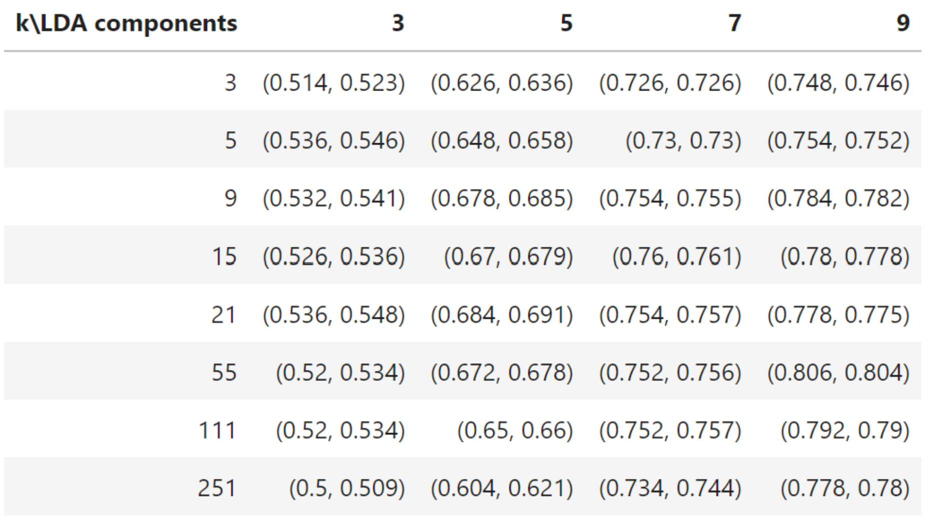
- **K-nearest neighbors**:
  - **KNN with PCA**: This method significantly outperforms the previous test with feature extraction using only PCA. Specifically, we achieve 70% f_1 score with 50 components and the parameter k set to 15. The choice of the right k and the amount of features plays a crucial role to avoid overfitting the data, in which the algorithm learns the training data by heart and does not recognise anything else outside this data.



k\PCA components	3	10	50	100	200	500	1200
3	(0.352, 0.348)	(0.552, 0.551)	(0.674, 0.674)	(0.644, 0.646)	(0.592, 0.594)	(0.494, 0.494)	(0.34, 0.294)
5	(0.4, 0.397)	(0.572, 0.574)	(0.692, 0.693)	(0.666, 0.668)	(0.606, 0.613)	(0.478, 0.465)	(0.328, 0.275)
9	(0.4, 0.396)	(0.592, 0.593)	(0.684, 0.687)	(0.654, 0.657)	(0.588, 0.594)	(0.448, 0.445)	(0.306, 0.26)
15	(0.368, 0.359)	(0.596, 0.595)	(0.7, 0.704)	(0.628, 0.634)	(0.53, 0.537)	(0.412, 0.409)	(0.282, 0.234)
21	(0.364, 0.359)	(0.602, 0.6)	(0.684, 0.686)	(0.6, 0.607)	(0.494, 0.506)	(0.382, 0.385)	(0.254, 0.206)
55	(0.378, 0.37)	(0.566, 0.563)	(0.604, 0.612)	(0.498, 0.5)	(0.392, 0.393)	(0.272, 0.237)	(0.212, 0.138)
111	(0.35, 0.343)	(0.512, 0.509)	(0.512, 0.505)	(0.43, 0.416)	(0.344, 0.334)	(0.244, 0.186)	(0.232, 0.152)
251	(0.324, 0.29)	(0.398, 0.398)	(0.4, 0.338)	(0.352, 0.271)	(0.342, 0.256)	(0.298, 0.209)	(0.316, 0.224)

Figure 19: KNN with PCA after VGG-16

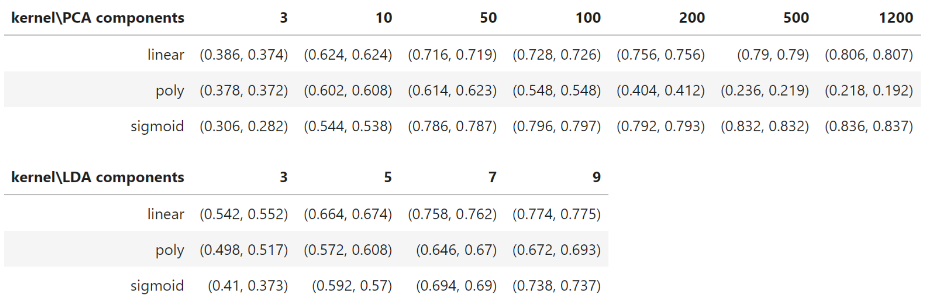
  - **KNN with LDA**: This method yields better results than both the previous test and the tests using PCA. The supervised nature of LDA contributes to this improvement in performances. In this case, we achieve 80% f_1 score with 9 components and k set to 55.



k\LDA components	3	5	7	9
3	(0.514, 0.523)	(0.626, 0.636)	(0.726, 0.726)	(0.748, 0.746)
5	(0.536, 0.546)	(0.648, 0.658)	(0.73, 0.73)	(0.754, 0.752)
9	(0.532, 0.541)	(0.678, 0.685)	(0.754, 0.755)	(0.784, 0.782)
15	(0.526, 0.536)	(0.67, 0.679)	(0.76, 0.761)	(0.78, 0.778)
21	(0.536, 0.548)	(0.684, 0.691)	(0.754, 0.757)	(0.778, 0.775)
55	(0.52, 0.534)	(0.672, 0.678)	(0.752, 0.756)	(0.806, 0.804)
111	(0.52, 0.534)	(0.65, 0.66)	(0.752, 0.757)	(0.792, 0.79)
251	(0.5, 0.509)	(0.604, 0.621)	(0.734, 0.744)	(0.778, 0.78)

Figure 20: KNN with LDA after VGG-16

- **Support Vector Machine**:
  - **SVM with PCA**: In this case the best result (83%) is reached with 1200 components and the kernel sigmoid.
  - **SVM with LDA**: against all expectations in this case the results are worst then the ones with PCA, in fact we reached 78% of f_1 score with 9 components and again the kernel poly.

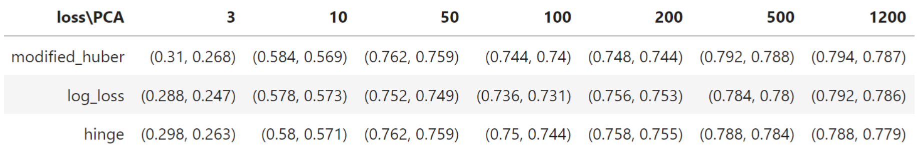


kernel\PCA components	3	10	50	100	200	500	1200
linear	(0.386, 0.374)	(0.624, 0.624)	(0.716, 0.719)	(0.728, 0.726)	(0.756, 0.756)	(0.79, 0.79)	(0.806, 0.807)
poly	(0.378, 0.372)	(0.602, 0.608)	(0.614, 0.623)	(0.548, 0.548)	(0.404, 0.412)	(0.236, 0.219)	(0.218, 0.192)
sigmoid	(0.306, 0.282)	(0.544, 0.538)	(0.786, 0.787)	(0.796, 0.797)	(0.792, 0.793)	(0.832, 0.832)	(0.836, 0.837)

kernel\LDA components	3	5	7	9
linear	(0.542, 0.552)	(0.664, 0.674)	(0.758, 0.762)	(0.774, 0.775)
poly	(0.498, 0.517)	(0.572, 0.608)	(0.646, 0.67)	(0.672, 0.693)
sigmoid	(0.41, 0.373)	(0.592, 0.57)	(0.694, 0.69)	(0.738, 0.737)

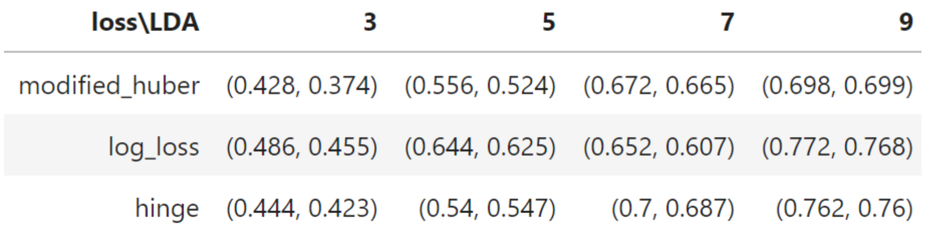
Figure 21: SVM with PCA and LDA after VGG-16

- **Stochastic Gradient Descent**:
  - **SGD with PCA**: Very similar results are obtained with 500 and 1200 components with both scoring a f_1 score of 78 % . Also the choice of the loss function plays a marginal role with the 'modified huber' loss function giving slightly better results.
  - **SGD with LDA**:It scores slightly below the PCA (76%).



loss\PCA	3	10	50	100	200	500	1200
modified_huber	(0.31, 0.268)	(0.584, 0.569)	(0.762, 0.759)	(0.744, 0.74)	(0.748, 0.744)	(0.792, 0.788)	(0.794, 0.787)
log_loss	(0.288, 0.247)	(0.578, 0.573)	(0.752, 0.749)	(0.736, 0.731)	(0.756, 0.753)	(0.784, 0.78)	(0.792, 0.786)
hinge	(0.298, 0.263)	(0.58, 0.571)	(0.762, 0.759)	(0.75, 0.744)	(0.758, 0.755)	(0.788, 0.784)	(0.788, 0.779)

Figure 22: SGD with PCA after VGG-16



loss\LDA	3	5	7	9
modified_huber	(0.428, 0.374)	(0.556, 0.524)	(0.672, 0.665)	(0.698, 0.699)
log_loss	(0.486, 0.455)	(0.644, 0.625)	(0.652, 0.607)	(0.772, 0.768)
hinge	(0.444, 0.423)	(0.54, 0.547)	(0.7, 0.687)	(0.762, 0.76)

Figure 23: SGD with LDA after VGG-16

- **K-means**:
  - **K-Means with PCA**: The clustering algorithm is outperformed by all the methods aforementioned, having an adjusted random score[3]of 18% with 50 components.

– **K-Means with LDA**: The supervised nature of the LDA plays a crucial role here drastically improving the performances and scoring a 46% of adjusted random score.

algorithm\PCA	3	10	50	100	200	500	1200
lloyd	0.167	0.163	0.179	0.172	0.156	0.154	0.104
elkan	0.138	0.163	0.179	0.190	0.167	0.196	0.113

algorithm\LDA	3	5	7	9
lloyd	0.250	0.374	0.445	0.412
elkan	0.272	0.374	0.467	0.447

Figure 24: K-Means with PCA and LDA after VGG-16

4.4 Data visualization

The visualisation with the t-SNE dimensionality reduction algorithm is very powerful and helped us to understand the separation of classes in both the 2-D and 3-D cases.

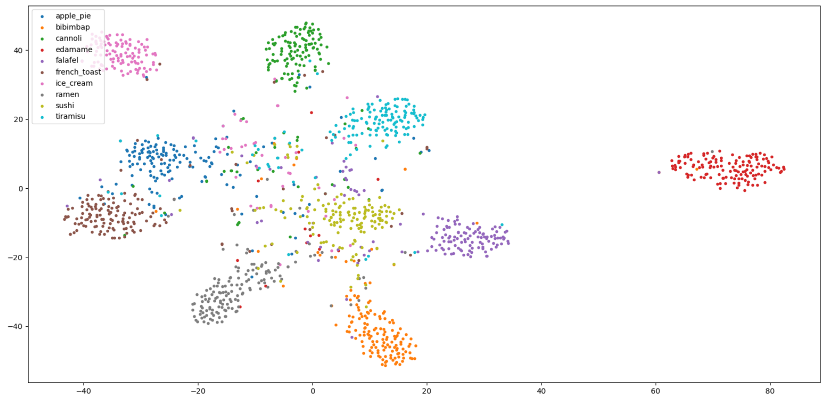


Figure 25: T-SNE 2D LDA reduced feature after VGG-16

[3]from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html: The Rand Index computes a similarity measure between two clustering by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clustering.

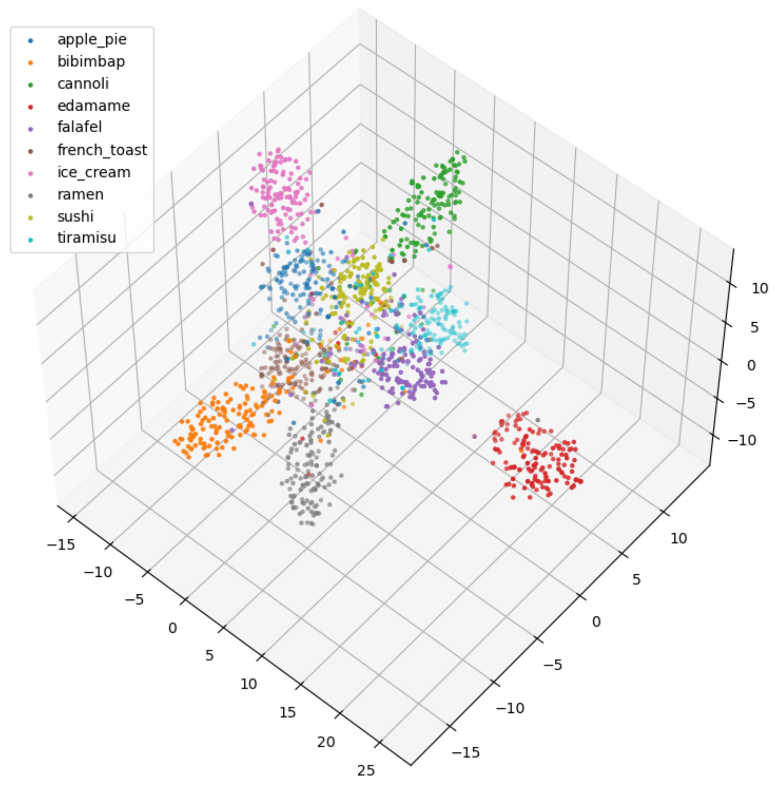


Figure 26: T-SNE 3D LDA reduced feature after VGG-16

5 Conclusion

In this report, we compared and analyzed various feature extraction and classification techniques on a set of food images.
Our initial experiments with PCA and LDA for feature extraction and traditional classifiers such as KNN, SVM, SGD and K-means on greyscale images bringed to us a sub-optimal results, emphasising the limitations of these methods in handling complex image data.

To take an extra step, we used the VGG-16 convolutional neural network for feature extraction. This approach improved classification performance by exploiting the deep learning model's ability to capture features from colour images.
Our results demonstrate that merging advanced feature extraction techniques, such as VGG-16, with traditional classifiers can lead to very important improvements in accuracy and precision.

We also ran the classification algorithms on the 7x7x512 feature vectors extracted with the CNN which leaded us to very similar results to the one obtained

after the dimensionality reduction. This allows us to confirm that dimensionality reduction does not affect the classification performances and decrease greatly the time needed by classification algorithms to complete (for instance the time needed by the classification with SVM decreased from over 10 minutes on raw VGG16 features vectors to about 50 seconds on PCA-reduced feature vectors). The best performances are achieved via SVM on the PCA-reduced features (500 components) extracted with the VGG16 neural network.
Some future investigations may be done in the classification part using more advanced methods such as neural networks.

Overall, our results emphasise the importance of using robust feature extraction methods to improve feature quality and importance in combination with dimensionality reduction algorithms to speed up the classification process.

References

[1] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[2] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.