Food classification on iFood 2019 Challenge

Supervised Learning 2023-2024

Presentation

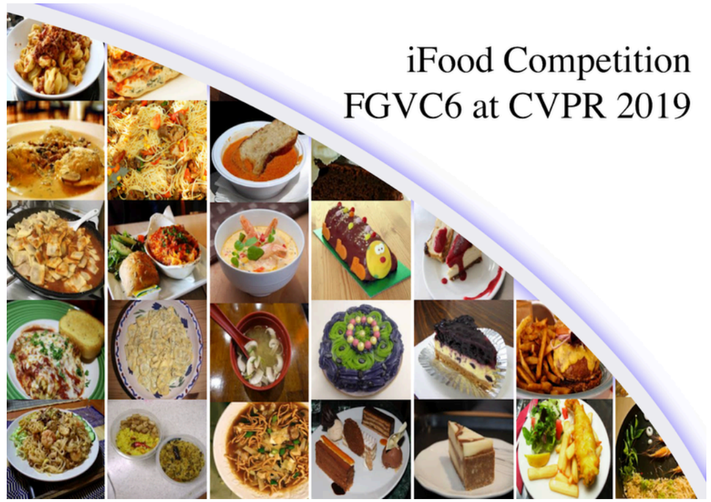
MSc Artificial Intelligence for Science and Technology

Supervised learning

Submitted by **Daniele Cecca Mat.914358**

UNIVERSITÀ DEGLI STUDI DI MILANO BICOCCA

Introduction



This project focuses on the development and imple-mentation of a classification system for food identification.

Automatic food identification can assist with food intake monitoring to maintain a healthy diet.

Food classification is a challenging problem due to the large number of food categories, high visual similarity between different food categories, as well as the lack of datasets that are large enough for training deep models.

PROJECT STRUCTURE

2 BOW with SVM

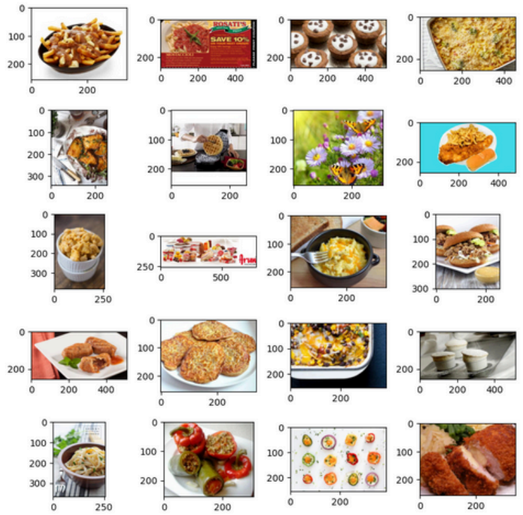
1. Extract **keypoint** locations and descriptors using **SIFT**.
2. Apply dimensionality reduction using **PCA**.
3. Cluster the **descriptors** to form the **vocabulary** through quantization, where each cluster represents a word.
4. Compute the **BoW** histogram for each image by counting the occurrences of each word.
5. Classify the images using an **SVM** model.
6. **Evaluate** the model.

3 Custom CNN

1. Create the **class dataset**.
2. Design the **architecture of the network.**
3. Train the network using different **hyperparameters**.
4. Select the **best-performing network** and train it for additional epochs.
5. **Evaluate** the network's performance.

In both cases, I apply some preprocessing to the data.

Dataset Exploration



The dataset is compose of

- **118,475 training images**
- **11,994 test images.**

- **251 classes**

There are **outliers** and **wrongly labeled image** because the training images are collected from internet.

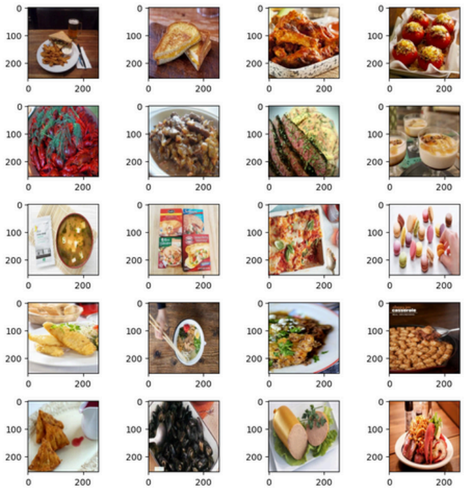
Submitted by Daniele Cecca

Data pre-processing

Because some of the algorithms I will use to build the vocabulary of features, are computationally expensive, I decided to use only a small portion of the data. I took **50 samples** for each class for **training** and **10 samples for testing.**

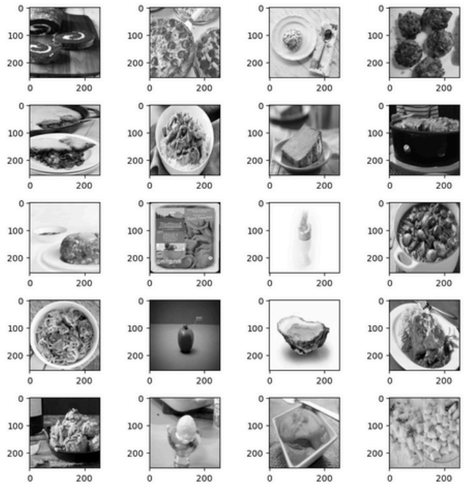
01

Resize Data



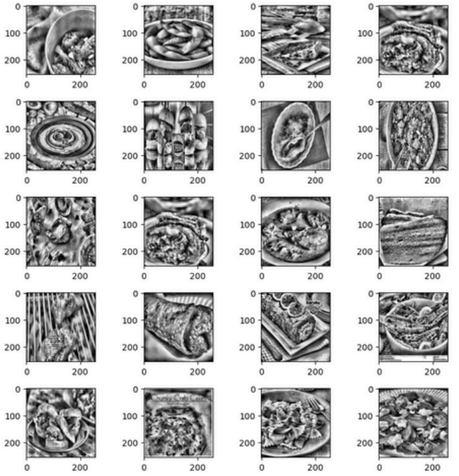
02

Convert data to grayscale



03

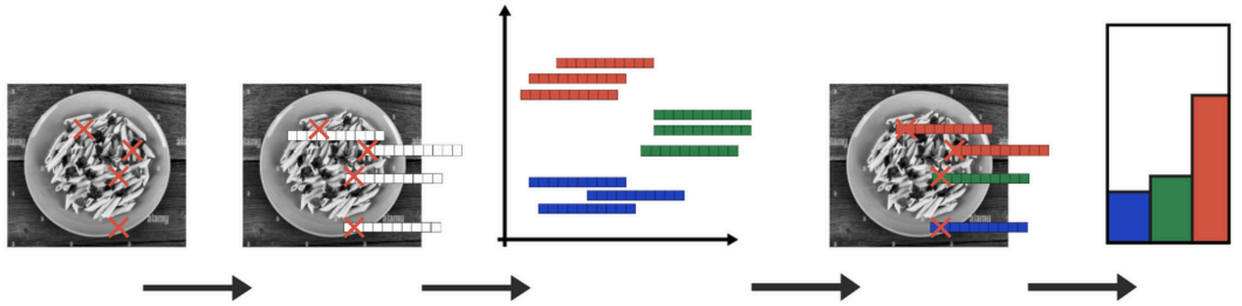
Normalizing pixel values



Bag of features

The process involves three steps:

1. Extract **raw features** (keypoints and descriptors) from all training images using **SIFT**.
2. **Cluster the descriptors** into k clusters using **K-means**, forming the visual vocabulary.
3. For each image, compute the **histogram of visual words** by assigning each feature to a cluster and counting occurrences.



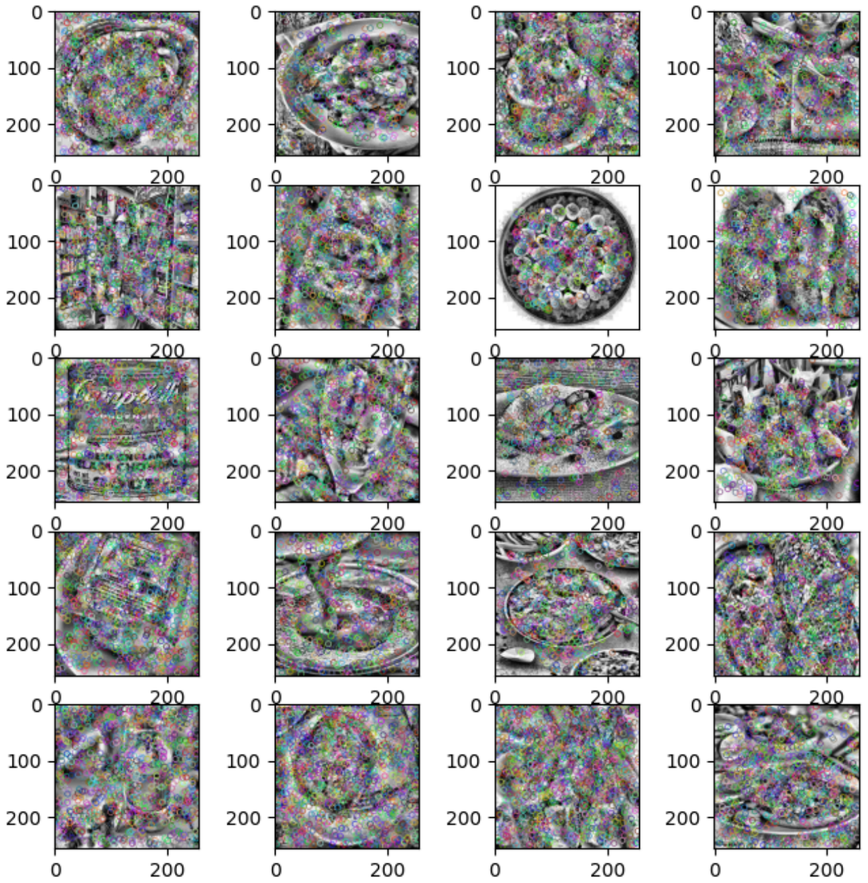
Submitted by Daniele Cecca

Step 1 - SIFT

BAG OF FEATURES

To extract the features, I used the **Scale-Invariant Feature Transform (SIFT) algorithm**.

1. **Scale-space**
2. **DOG(Difference of Gaussian kernel)**
3. **Finding keypoints**
4. **Refinements**
5. **Orientation Assignment**
6. **Keypoint descriptor**



Submitted by Daniele Cecca

Step 2 - K-means

BAG OF FEATURES

I decided to apply **PCA** because the descriptors provide a much sparser set of points to work with.

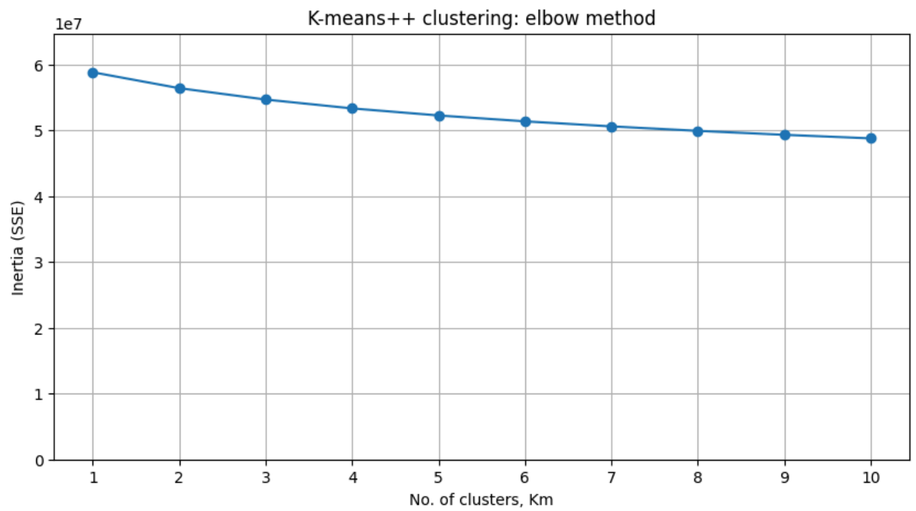
To determine the number of components, I used the **Kaiser rule.**

In the end, I used **30 components.**

To determine the number of clusters k, I used the **elbow method**

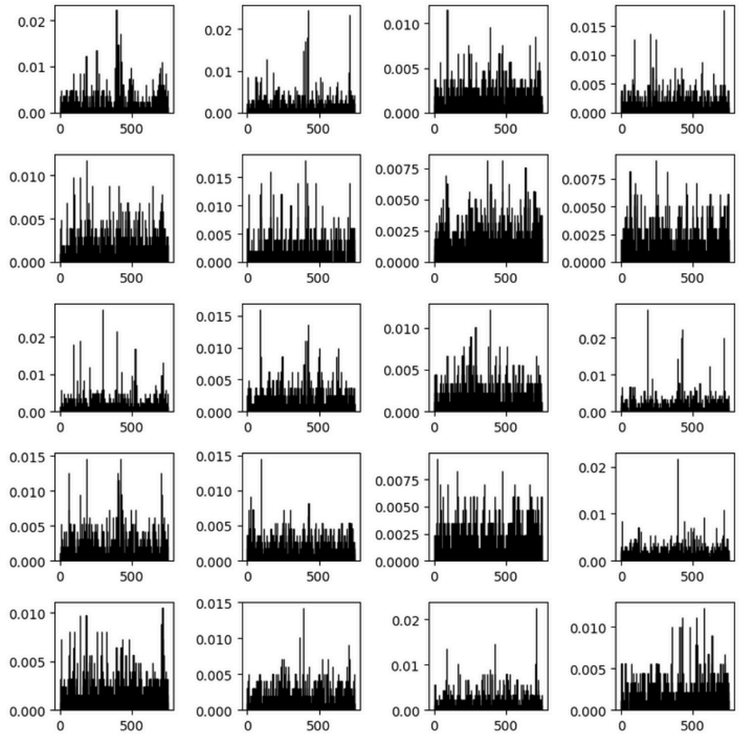
Due to computational limits, I created a smaller dataset with **10** resulting in a total of **1,265,631 features.**

I chose the **6th iteration k=750** because from the graph, the error appears to decrease slowly, making it a good trade-off.



Step 3 - Histograms of features

BAG OF FEATURES



After computing the vocabulary, I generated **histograms of features for each image**, both for the images in the training set and the test set.

SVM

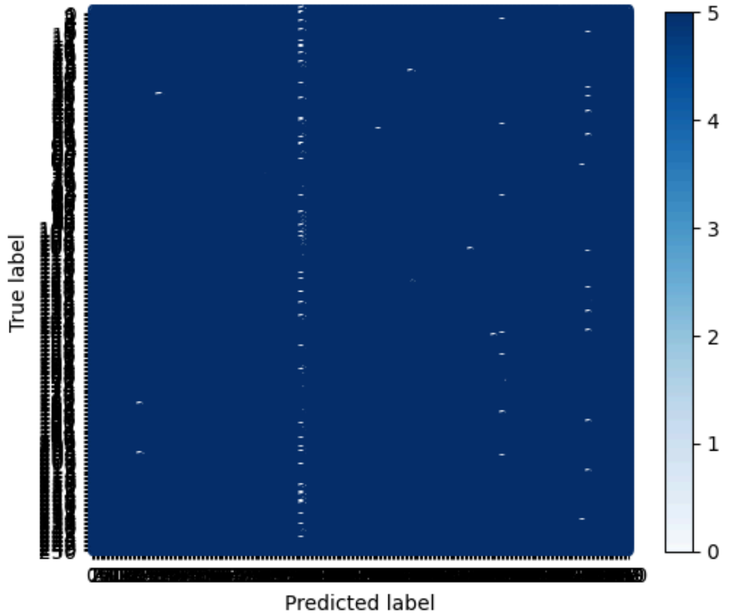
To classify the images, I utilized the **Support Vector Machine (SVM)**.SVM finds an optimal hyperplane to separate data points into different classes by maximizing the margin between classes

I employed Cross-validation. Specifically, I used S**tratified Cross-validation** with **k equal to 3**

To determine the best **hyper-parameters**, I defined a grid of parameters and performed a **grid search**.

Parameter	Values
C	[0.1, 1, 10, 100]
gamma	[1, 0.1, 0.01, 0.001]
kernel	['linear', 'rbf']

Parameter	Value
classifier__C	100
classifier__gamma	1
classifier__kernel	'rbf'



Metric	Value
Accuracy	0.0044
Precision	0.0054
Recall	0.0044
F1 Score	0.0019

Evaluation

SVM

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

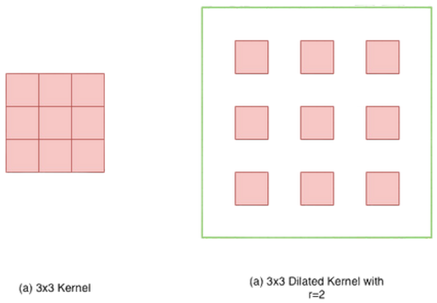
$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

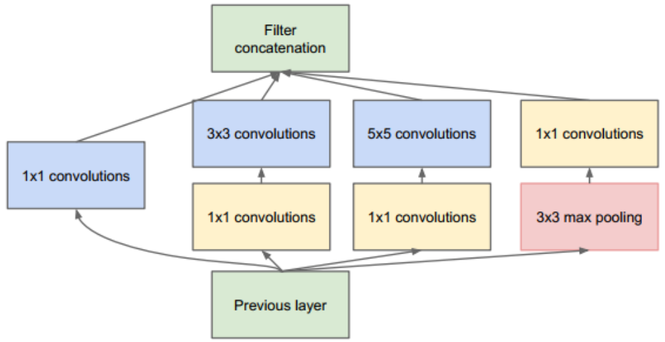
$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

CNN architecture

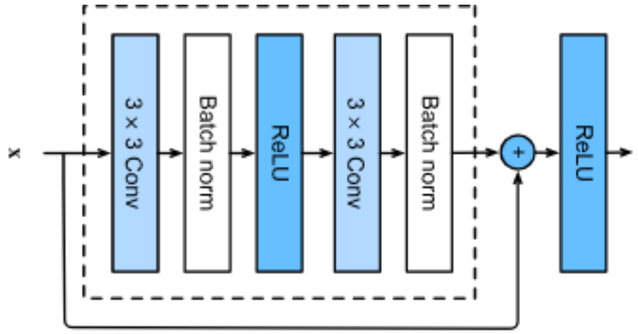
Atrous Convolution



Inception block



Residual block



After each convolutional layer in the residual blocks, **Batch Normalization** is applied, and also **Adaptive Max Pooling**

Initialization

CNN

Initializing the network parameters is critical. To avoid exploding gradient or vanishing gradient, I apply **He initialization**. The biases **β are set to zero**, and the **weights Ω are chosen from a normal distribution** with **mean zero** and **variance 2 Dh** , where Dh is the number of hidden units in the previous layer.

He Normal initialization

$$W \sim N(0, Var(W))$$

$$Var(W) = \sqrt{\frac{2}{n_{in}}}$$

HyperParameters Tuning

CNN

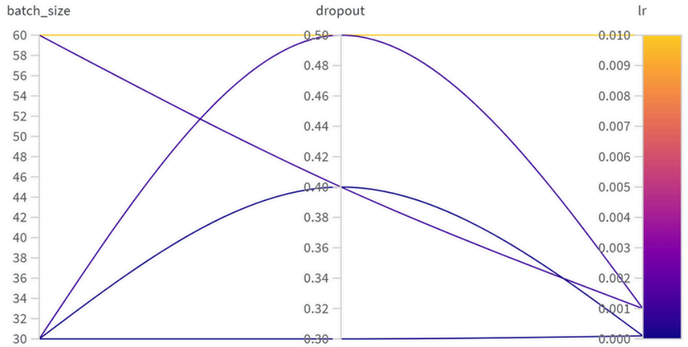
I create a configuration that will be used by the **Weights and Biases agent** to set the different **hyperparameters** during various experiments. I used the simplest agent which chose the combination of the **parameters randomly**.

Parameter	Values
epochs	10
learning rate	{0.01, 0.001, 0.0001}
dropout	{0.3, 0.4, 0.5}
batch size	{30, 60, 90}
loss function	CrossEntropyLoss
optimizer	Adam
scheduler	StepLR(optimizer, step_size $= 21, \gamma = 0.2$)

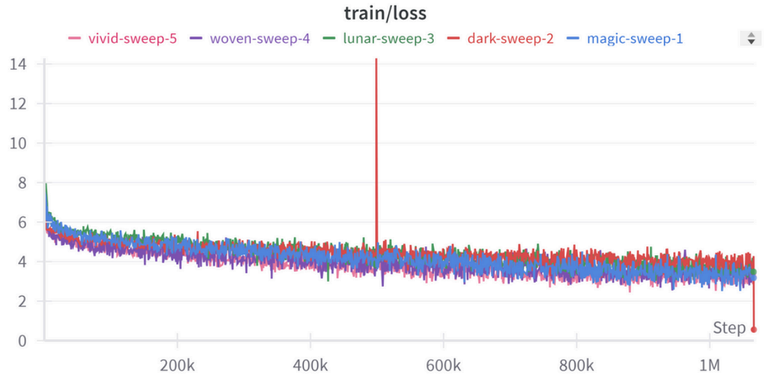
Submitted by Daniele Cecca

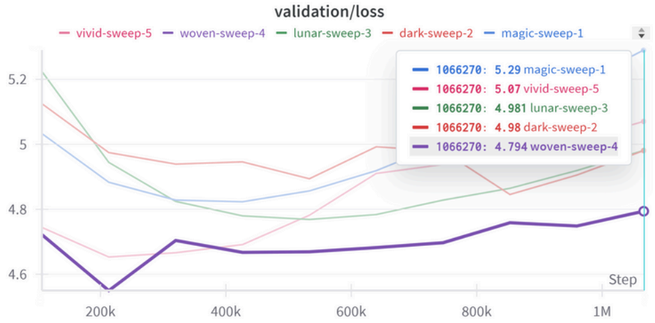
Training

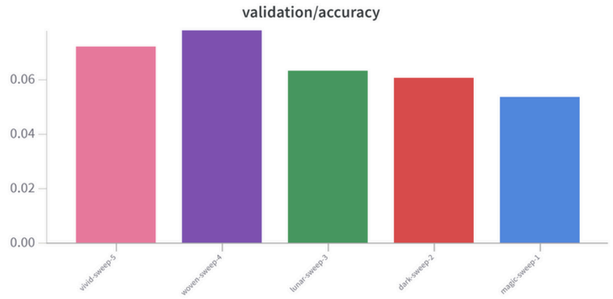
CNN



Parameter	Value
Batch Size	30
Dropout	0.5
Epochs	10
Learning Rate (lr)	0.001





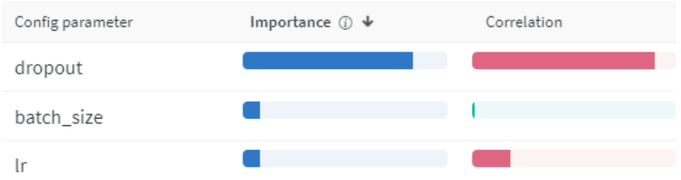


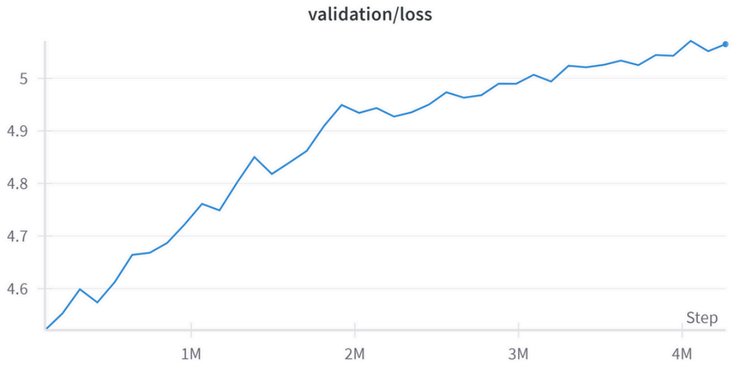
Training the best model

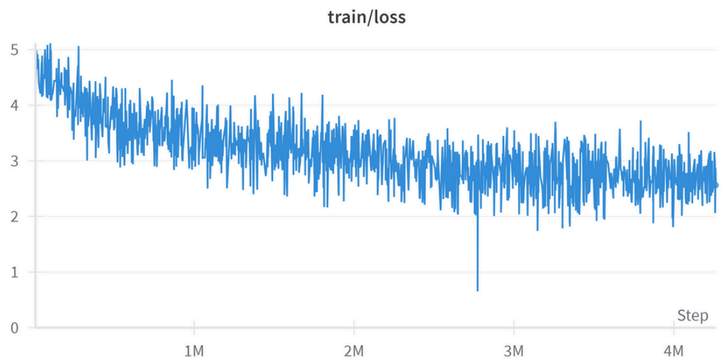
CNN

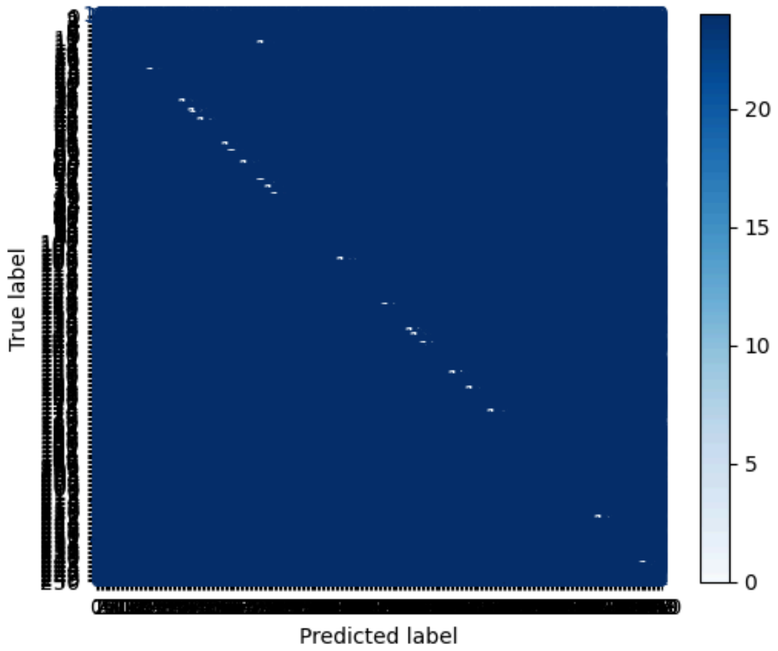
It appears that our **model may be overfitting** .This suspicion aligns with the fact that **dropout, a highly correlated hyperparameter**, tends to improve results as its value increases.

Alternatively, the increasing loss could indicate **underfitting**, becausee loss remains within a small interval









Metric	Value
Accuracy	0.095
Precision	0.086
Recall	0.090
F1 Score	0.084

Evaluation

CNN

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

FUTURE WORKS

BOW with SVM

- Remove outliers.
- Utilizing a larger portion of the original dataset.
- Integrating color descriptors, either through early fusion or late fusion approaches.
- Try other models

Custom CNN

- Remove outliers.
- Exploring advanced deep learning architectures specific to food image recognition.
- Trying other convolutional layers and hyperparameters to enhance model generalization and efficiency.

THANKS FOR ATTENTION

DANIELE CECCA MAT 918358