FIMC
46 50 04-07-01
LaTeX

Project Report

2024

L. M. B. P. - GitHub: Luismbpr

1 Introduction

Image classification application for predicting different types of food labels.

The project consists on the development and deployment of a web application that can perform inference on a given input (Image) and return an output (Prediction Probability).

2 Scope

This project will cover the following:

- Development and deployment of a Computer Vision model that can perform inference for an input given as an image.
- Key constraints.
  - Decision of the model architecture.
  - Decision if a base model will be used.
    - (If a base model is used).
      Decision on which base model to use.
  - Dataset decision.
    - Dataset amount used for training.
    - Dataset amount used for testing.
  - Model creation.
    - Model training.
    - Model testing.
    - Testing model by performing inference.
    - General Metric Performance Analysis.
      - Model performance.
      - Metric - Accuracy.
      - Metric - Loss.
  - Model selection.
  - Web application.
    - Framework and Tool Selection.
    - General design.
    - Development.

        - Deployment.

- *Model Improvement.
    - Model training.
    - Model testing.
    - Testing model by performing inference.
    - General Metric Performance Analysis.
    - Model selection.
    - *Web Application improvement.
        - Development.
        - Modification and Improvements.
        - Deployment.

This Project Will Not Cover

The project will be limited to the elements defines in the "What This Project Covers" section. Any other aspects not included in this section will be considered outside the project's scope and will not be addressed or considered as requirements.
Items with '' may not be covered depending on various external and internal factors beyond the scope of the project. A key factor that may if key constraints permit it.*

3 Key Constraints

Several constraints had to be taken into consideration prior and during the project development-deployment. These constraints mainly defined the scope limits of the project.

- Time (in general), including:
  - Project duration time.
  - New dataset creation.
  - Model Creation (Training, Testing, metric analysis, etc).
  - App inference time.
- Space.
  - Cloud space.
  - Model (size).
  - Example images (size).
  - Web application space (size).

Project Estimate Duration

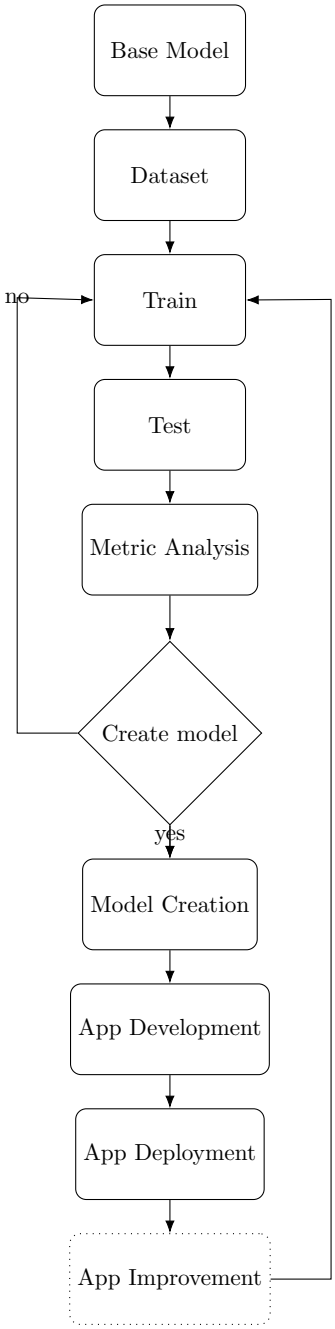
There was a clear idea that this app would be a web application and not a device application. The application needed to perform fast but had to also be light in size. For this reason the app overall, including the model, required to be light.

The project duration was extended longer than what it was intended to be. Many improvements that would lead to better model results and a better web application overall were made. For thar reason and time constraint, the project was sepparated into different phases. Other constraints that involved time and that had to be taken into account during the project were dataset percentage, label selection as well as training for the machine learning model.

4 General Information

Development of an web application that can perform image classification by using a machine learning model for computer vision that outputs the prediction of a food name.

The decision to use base model and train it with specific data was an option. A pre-trained model gives many advantages. The main advantage would be that the model *probably* would learn faster and be more accurate than creating a model architecture and train it from zero.


Base Model
Dataset
Train
no
Test
Metric Analysis
Create model
yes
Model Creation
App Development
App Deployment
App Improvement

5 Project Development

Model Creation

Will train a base model with specific data. The pre-trained model had some attached weights that will later change as the model is being trained with new information. This will help on reducing the time on the creation of the model in comparison to constructing a model starting from the initial stages.

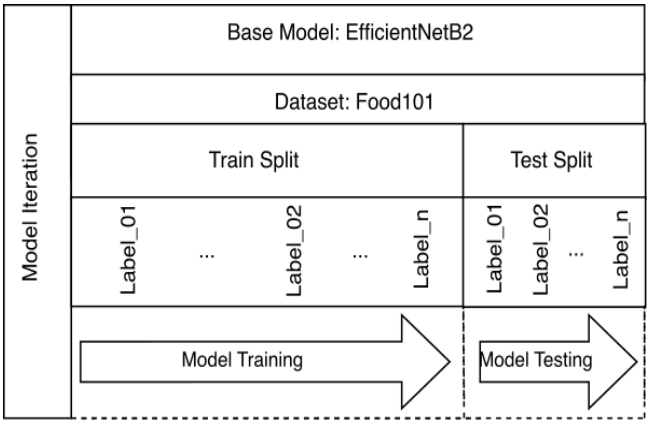


Figure 1: General Base Model and Dataset

There are several pre-trained models to choose from and many model architectures. The decision will be to use 'Efficient Net' as a base model.

Efficient Net As A Base Model

As described on PyTorch's website, "EfficientNet is an image classification model family." [4] This architecture was released on 2019 on a paper named: "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" [11]. The paper describes the improvement of this model's architecture compared to other Convolutional Neural Networks.

Other people had used this architecture and it has performed well. Daniel Bourke created an application where he used this pre-trained model architecture with its pre-trained weights to predict 3 food labels [2]. He also showed some good information about how to work with Neural Networks and PyTorch on YouTube [3] and on his GitHub repository [1].

After choosing the base model it was time to select the model version that would best suit the app. This phase consisted on inspecting Efficient Net's model differences. [7]

Base-Model-Name	Acc-at-1 on ImageNet-1K	Acc-at-5 on ImageNet-1K	num-params	GFLOPS	File-Size-MB
EfficientNet-B0	77.692	93.532	5,288,548	0.39	20.5
EfficientNet-B1	78.642	94.186	7,794,184	0.69	30.1
EfficientNet-B2	80.608	95.31	9,109,994	1.09	35.2
EfficientNet-B3	82.008	96.054	12,233,232	1.83	47.2
EfficientNet-B4	83.384	96.594	19,341,616	4.39	74.5
EfficientNet-B5	83.444	96.628	30,389,784	10.27	116.9
EfficientNet-B6	84.008	96.916	43,040,704	19.07	165.4
EfficientNet-B7	84.122	96.908	66,347,960	37.75	254.7

Table 1: Efficient Net Architecture Version Information

There were some important aspects that needed to be taken into consideration to select the version that would best suit the application. Model size needed to be as low as possible, but still need to have good metric performance. There seems to be a correlation along the number of parameters a model architecture has and the amount of time it takes to train the model.

Base Model Version Comparison

There was a decision to not consider models that had a size of more than 50 MB due to the space size constraint in the cloud. For that reason EfficientNet Model version from B4 to B7 were ruled out. The considered versions were B1, B2 and B3. The accuracy was slightly different on all 3. The size was a little bit high on B3 and the number of parameters was still high compared to B2, meaning that it might take longer to train and might not be a considerable difference in metric performance. B3 was then ruled out.

Base Model Version Selection

EfficientNet B2 version was chosen as the base model version. Along with the pre-trained weights. The ratio between the model size and model parameters was good.

Dataset Decision Selection

The dataset selection was Food101 [6]. A dataset that originally consisted of 101 food classes. Each food image label contains around 1000 images. PyTorch Contains this dataset on [5]

Note: *There are many other model versions and model architectures. This was the chosen model architecture and model version. Experimenting with them is encouraged and might even result in better model metric performance.*

6 Project Development - Model Creation

Many iterations of the model were created using the same base model. This was trained and tested with different percentages of the same dataset.

First Iteration

The first iteration of the model was trained on 3 classes with 15% of the dataset. The model's performance was good but needed improvement. More data was going to be needed to see if the models performance could be improved.

Second Iteration

The second iteration was using the same 3 classes as the previous model but trained with 20% of the dataset. The model had performed well in the training and testing phases. This model was selected for deployment. The application was deleted after the deployment of an improved version. *Note: See 'Project Implementation' for more deployment information for this version of this web application.*

Third Iteration

The amount of labels was increased from 3 to 10 labels. 20% of the whole dataset was used for training and testing. The model's performance results were not as expected. Incrementing the dataset percentage for training might give better results.

Fourth Iteration

This model iteration selected the same 10 labels as the previous iteration. An increment from 20% to 35% of the whole dataset (for training and testing) was used to see if the model could achieve better results than the previous model. This resulted on the best-performing model for the 10 labels selected.

This model was selected for deployment. The application had slight modifications.

Other Iterations

The following are just a few of the most relevant models during those iterations.

Incrementing label amount to the total amount of the dataset and using 35% of the data for training and testing (80-20). This resulted in a lower accuracy (around 64%) which was not bad comparing the amount of labels, but the predictions done on the test set examples for the model deployment indicated that the model's performance was lower than what the accuracy metric displayed. The training time was also higher, so some adjustments were required on the next iteration(s).

Lowering label quantity and/or increasing the dataset percentage was required to have a better performance. Constraints such as amount of time, amount of free training time, amount of space in the cloud were key factors that lead to the decision of the creation of another constraint. Label quantities required to be lower than the total amount of the dataset as well as maintaining a percentage of no more than 60% for the whole dataset creation (for training and testing).

Requirements

- Performace metric accuracy needs to be of at least 65% accuracy when using more than 30 labels.
- When using more than 35% of the dataset the number of epochs needs to be no more than 10 epochs on training.
- The maximum amount of the whole dataset needs to be less than 65% for train and testing.
- Train set percentage should be no more than 95% of the dataset created.
- Test set should be no less than 5% of the dataset created.

Several iterations were created and tested. None of those iterations were selected for deployment. Most of the models performed moderately well in metric results taking into consideration the the small percentage of the dataset selection (35 %) in comparison to the original dataset. All of those models were then tested on a random selection of images selected from the testing dataset. Most of them performed not as expected. *Man in the loop* testing was very important to see that the models did not perform as expected given the metric results. None of these models were deployed due to that reason. Constraints lead to creating a new model. Model improvement was not a possibility for those models given the constraints and what this project scope could reach. More information was required in terms of dataset percentage for training and testing. Training time would be higher since parameter tuning, such as an increase of number of epochs would be required.

Other models

Label quantity reduction

Creating models with a lower label quantity was then required.
A considerable label reduction was done on the last two iterations.

- Reduction from 101 to 86 labels.
- Reduction from 86 to 64 labels.

dataset-percentage	train-split	test-split	labels	deployed	epochs-5	epochs-10	epochs-15	epochs-20
0.15	0.9	0.1	3	0	1	1	0	0
0.2	0.9	0.1	3	1	1	1	0	0
0.2	0.8	0.2	10	0	1	1	1	1
0.35	0.8	0.2	10	1	1	1	1	1
0.35	0.8	0.2	101	0	1	0	0	0
-	-	-	-	-	-	-	-	-
0.65	0.8	0.2	88	0	1	0	0	0
0.65	0.8	0.2	86	0	1	0	0	0
0.45	0.8	0.2	86	0	1	0	0	0
0.45	0.8	0.2	85	0	1	0	0	0
0.45	0.8	0.2	64	0	1	0	0	0
0.5	0.8	0.2	46	1	1	0	0	0

Table 2: Model Creation Information

7 Project Analysis

Last Iteration

Model information for the last iteration.

46 labels with 50% of the whole dataset.

Trainining Set: 80%

Testing Set: 20%

Percentage of the whole dataset: 40% Train - 10% Test

The model can predict the following food labels.

1. apple pie
2. breakfast burrito
3. caesar salad
4. carrot cake
5. chicken quesadilla
6. chicken wings
7. chocolate cake
8. churros
9. club sandwich
10. cup cakes
11. deviled eggs
12. donuts
13. eggs benedict
14. filet mignon
15. french fries
16. french toast
17. fried rice
18. garlic bread
19. greek salad
20. grilled salmon
21. guacamole
22. gyoza
23. hamburger
24. hot dog
25. huevos rancheros
26. hummus
27. ice cream
28. lasagna
29. macaroni and cheese

30. nachos
31. omelette
32. onion rings
33. oysters
34. paella
35. pancakes
36. pizza
37. ramen
38. ravioli
39. risotto
40. samosa
41. spaghetti bolognese
42. spaghetti carbonara
43. steak
44. sushi
45. tacos
46. waffles

Labels	Whole-dataset-percentage	Train-split	Test-split	Percentage-Train-Whole-Datatset	Percentage-Test-Whole-Datatset
46	50%	80%	20%	40%	10%

Table 3: Latest Model Dataset Percentage

Latest Model

dataset-percentage	train-split	test-split	labels	deployed	epochs-5	epochs-10	epochs-15	epochs-20
0.5	0.8	0.2	46	1	1	0	0	0

Table 4: Latest Model Information

Latest Model Results

The dataset to training and testing the model consisted of 50% of the whole dataset, which was roughtly 500 images per food label. The following were the metrics that resulted of training the latest model for 10 epochs.

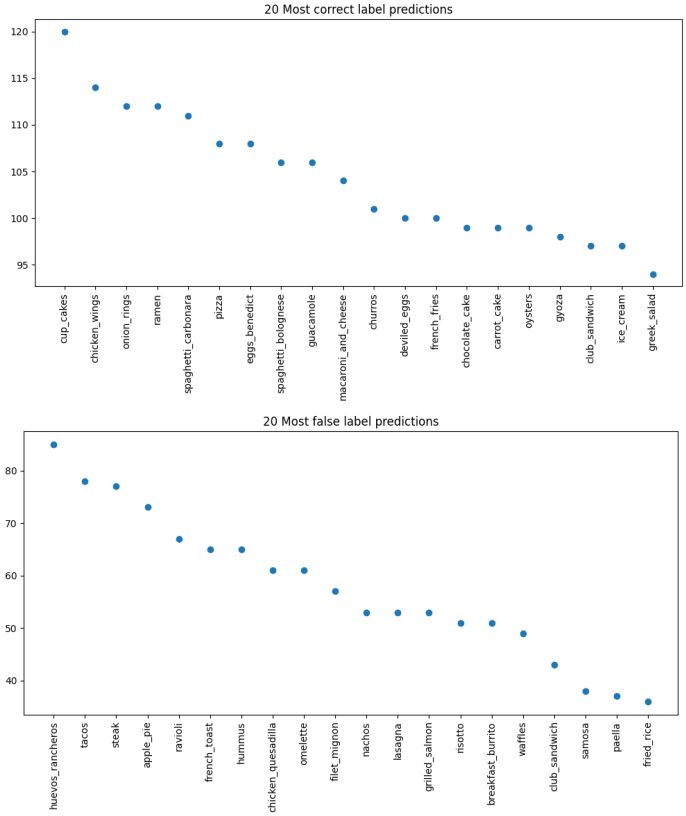
epoch	train-loss	test-loss	train-acc	test-acc
1	2.7337	1.9961	0.4171	0.6358
2	2.1743	1.8329	0.546	0.6627
3	2.0917	1.7729	0.5722	0.6772
4	2.0482	1.7689	0.5761	0.6774
5	2.0257	1.7128	0.5867	0.6876
6	2.0015	1.7233	0.5975	0.6901
7	1.9904	1.7068	0.5982	0.6979
8	1.9828	1.7021	0.5993	0.7003
9	1.9691	1.7067	0.6098	0.6983
10	1.9571	1.7062	0.6095	0.6931

Table 5: Latest Model Results

The overall performance metrics were favorable in comparison to previous models given the amount of labels and the reduction of the dataset percentage selection in comparison to the whole dataset. The training and test loss were not performing as expected, but the training and testing accuracy were around 70 %. Exploration for the most correct and most incorrect predictions was done.

Latest Model Testing Phase

The model testing phase was performed using the entire test dataset with a batch size of 32. These are the top 20 best and worst predictions for this model iteration.



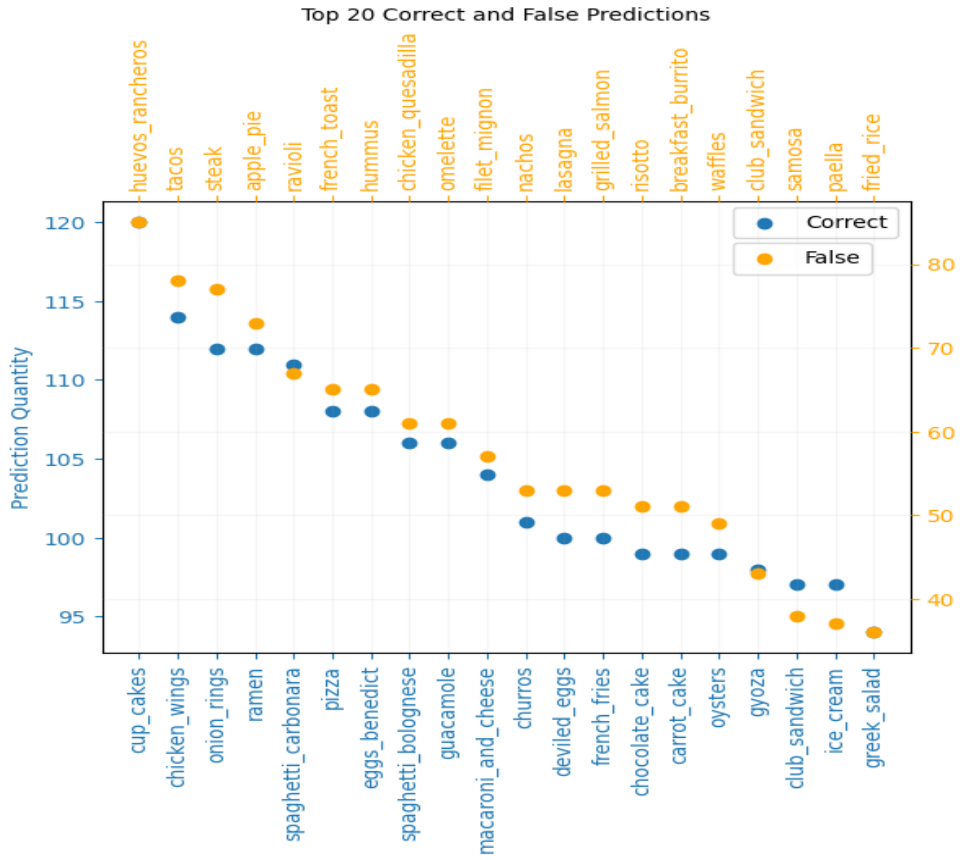
Top label predictions

The 5 most correct label predictions were

- Cupcakes
- Chicken wings
- Onion rings
- Ramen
- Spaghetti carbonara

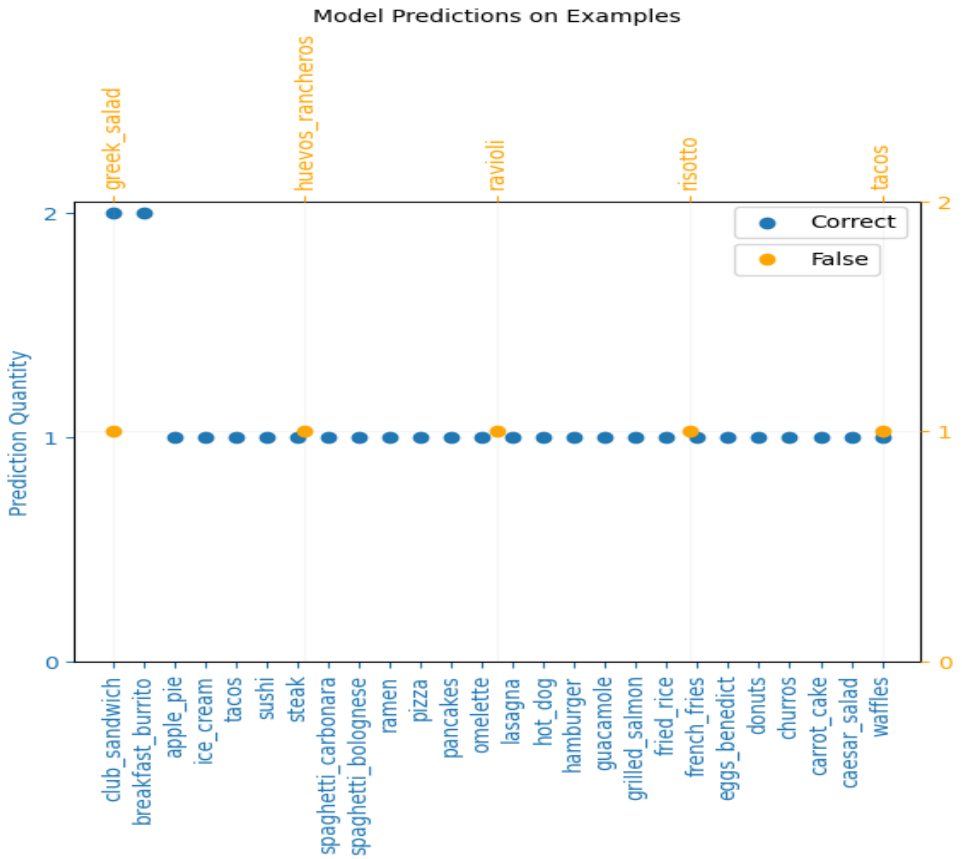
The 5 most incorrect label predictions were

- Huevos rancheros
- Tacos
- Steak
- Apple pie
- Ravioli



Inspection of the food labels where the model was having trouble making correct predictions gave some idea about what could be happening. The model could be having trouble predicting the most incorrect food labels since most of them look similar to other image labels used to train the model. One example is where the labels are almost the same such as filet mignon and steak. Filet mignon could be labelled as steak and vice-versa. Another example are foods that happen to be similar in appearance such as onion rings and donuts. The model might be having trouble in differencing both labels.

Model Example Predictions



Model Decision

Although both train and test loss were not performing as expected accuracy was higher than the requirement. Further inspection was required since the model did not seem to be overfitting nor underfitting.

A *Man in the loop* test was performed. A random and a manual selection of images from the test set were used for testing. The model performed better than what was expected on both, the random and the manuallys selected images. The model deployment was performed.

8 Project Implementation

The user can either input an image or use one of the example images. The model performs inference and outputs its predictions.

Diagram on minimum process requirements for the web application.

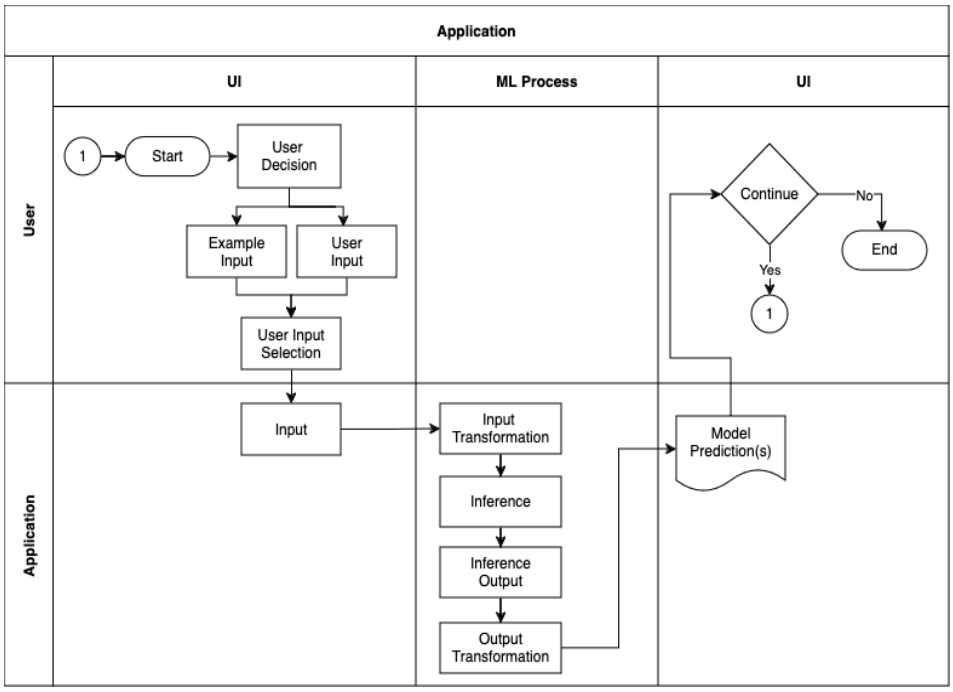


Figure 3: General Web App Process

App Deployment

Part of this deployment consists on uploading all the required files, including the trained model and the interface code into the repository. The repository hosts the required files and they are available via the server they offer.

Hugging Face Spaces offered a simple solution to hosting and deploying the app [9]. It was a good option/solution for hosting and deployment of this app. For the creation of the User Interface Gradio was used [8]. Both, HFS and Gradio had good documentation and helpful examples that made app development clearer.

Note: Several frameworks and services were considered for the deployment. The previous mentioned services were simple and fast to deploy..

Each Hugging Face Spaces environment offers a capacity of 16GB RAM and 8 CPU Cores. Each environment is currently limited to 16GB RAM and 8 CPU cores. [10]

Deployment 001

The first deployment was focused on making sure that the web application worked correctly. Some adjustments had to be made before the deployment. The web app interface was simple, but showed the required outputs correctly.

- Model
  - Whole Dataset percentage: 20%
  - Train Split: 90 %
  - Test Split: 10 %
  - Labels: 3
- Versions
  - Python==3.11.9
  - gradio==4.36.0
  - torch==2.3.1
  - torchvision==0.18.1

Deployment 001 - Project File Structure

- foodimgclass-03-20-01
  - app.py
  - model.py
  - trained-model
  - requirements.txt
  - README.MD
    - examples
      - example-01
      - example-02
      - example-03

Deployment 002

This version was focused on improving the web application by increasing the label size from 3 to 10 food labels.

- Model
  - Whole Dataset percentage: 35%
  - Train Split: 90 %
  - Test Split: 10 %
  - Labels: 10
- Versions
  - Python==3.11.9
  - gradio==4.36.0
  - torch==2.3.1
  - torchvision==0.18.1

Deployment 002 - Project File Structure

- foodimgclass-10-35-01
  - app.py
  - model.py
  - trained-model
  - requirements.txt
  - README.MD
    - examples
      - example-01
      - example-02
      - ...
      - example-10
  - Extras
    - README-02.MD

Deployment 003

Third deployment was focused on increasing the label size from 10 to 46 food labels. Some improvements were made to the interface.

- Model
  - Whole Dataset percentage: 50%
  - Train Split: 80 %
  - Test Split: 20 %
  - Labels: 46 %
- Versions
  - Python==3.10.12
  - gradio==4.36.1
  - torch==2.4.0
  - torchvision==0.19.0

Deployment 003 - Project File Structure

- foodimgclass-46-50-01
  - app.py
  - model.py
  - trained-model
  - requirements.txt
  - README.MD
    - examples
      - example-01
      - example-02
      - ...
      - example-12
  - Extras
    - data
      - class-names.txt
    - python-version.txt
    - README-02.MD

Third deployment

epoch	train-loss	test-loss	train-acc	test-acc
1	2.7337	1.9961	0.4171	0.6358
2	2.1743	1.8329	0.546	0.6627
3	2.0917	1.7729	0.5722	0.6772
4	2.0482	1.7689	0.5761	0.6774
5	2.0257	1.7128	0.5867	0.6876
6	2.0015	1.7233	0.5975	0.6901
7	1.9904	1.7068	0.5982	0.6979
8	1.9828	1.7021	0.5993	0.7003
9	1.9691	1.7067	0.6098	0.6983
10	1.9571	1.7062	0.6095	0.6931

Table 6: Latest Model Results

Deployment Improvement

The first iteration consisted on making sure that the app worked without any issues. The model performed well, but the amount of food labels was small. A label increment was necessary to improve the app. 10 labels is still a small percentage of the whole dataset. using a few percentage of the whole dataset.

Deployed-Iteration	Labels-From	Labels-To	Whole-Dataset-Percentage (%)	Label-Increment (%)
Iteration 1	3	3	2.97	0
Iteration 2	3	10	9.90	333
Iteration 3	10	46	45.54	460

Table 7: Model Improvement Over Iterations

Results of Deployed App on Example Images

The app was tested on the example images of the deployed web application. These are the results of the performance of the trained model.

The model could correctly predict all of the example images. The average model inference time for this test was 0.6647 seconds. There were some prediction percentages that were below 50%. The model still predicted the correct label but it was not confident enough

No	real label	Prediction label	Prediction Percentage	Prediction Time (s)	Prediction - True = 1
1	Hot dog	Hot dog	96	0.5413	1
2	Club Sandwich	Club Sandwich	72	0.3969	1
3	Breakfast Burrito	Breakfast Burrito	81	0.5939	1
4	Breakfast Burrito	Breakfast Burrito	96	0.9076	1
5	Apple Pie	Apple Pie	69	0.7614	1
6	Lasagna	Lasagna	19	0.4346	1
7	Caesar Salad	Caesar Salad	50	1.1197	1
8	Donuts	Donuts	48	0.6061	1
9	Pancakes	Pancakes	78	0.8936	1
10	Tacos	Tacos	35	0.7219	1
11	Fried Rice	Fried Rice	50	0.4028	1
12	Grilled Salmon	Grilled Salmon	91	0.5964	1
-	-	-	Mean:	0.6647	-

Table 8: Deployed Example Results

to label them with a greater score. Some of the lowest prediction probability scores were Lasagna (19%), Tacos (35%) and Donuts(48%). Some of the highest prediction percentages (for this example results) were Grilled salmon (91%), Breakfast burrito(96%) and Hot dog (96%).

9 Conclusion

The Web application was completed and performed well. Although the model decreased in accuracy performance in comparison to the first and second iteration, but labels were incremented in a greater amount. The overall application was improved since it performed well on the example images.

Room for Improvement

There is still a lot of room for improvement. Label quantity increment would be one of the main improvements to do. More resources would be required such as amount of data and/or longer training time (epochs increment). Another improvements that could be done were making the web application inference more efficient and reducing the inference rate (ms).

10 Acknowledgement

I would like to express my gratitude. To PyTorch for the amount of information and amount of work they have done. To Gradio for making deployment simple; To the people who created and released the dataset used (Food101).

I would like to express my gratitude to Mr. Daniel Bourke, especially for the resources he has put available.

Thank you to all who make information, resources and knowledge available, especially to everyone who do it without any paywall.

Thank you to everyone who has contributed towards the improvement of this fascinating area of Artificial Intelligence.

References

[1] https://github.com/mrdbourke/pytorch-deep-learning/. 2021.

[2] https://www.youtube.com/watch?v=ueLolShyFqs&list=PL8IpyNZ21vUQw-TYaf2xF6SbUrqRKbGxh&index=6. 2022.

[3] https://www.youtube.com/watch?v=V_xro1bcAuA&t=4695s. 2022.

[4] https://pytorch.org/hub/nvidia_deeplearningexamples_efficientnet/.

[5] https://pytorch.org/vision/main/generated/torchvision.datasets.Food101.html.

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. "Food-101 – Mining Discriminative Components with Random Forests". In: *European Conference on Computer Vision*. 2014.

[7] *EfficienNet*. https://pytorch.org/vision/main/models/efficientnet.html.

[8] *Gradio*. https://www.gradio.app/.

[9] *Hugging Face*. https://huggingface.co/.

[10] *Hugging Face Spaces*. https://huggingface.co/spaces/launch/.

[11] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *ArXiv* abs/1905.11946 (2019). URL: https://api.semanticscholar.org/CorpusID:167217261.