FIMC

10 35 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

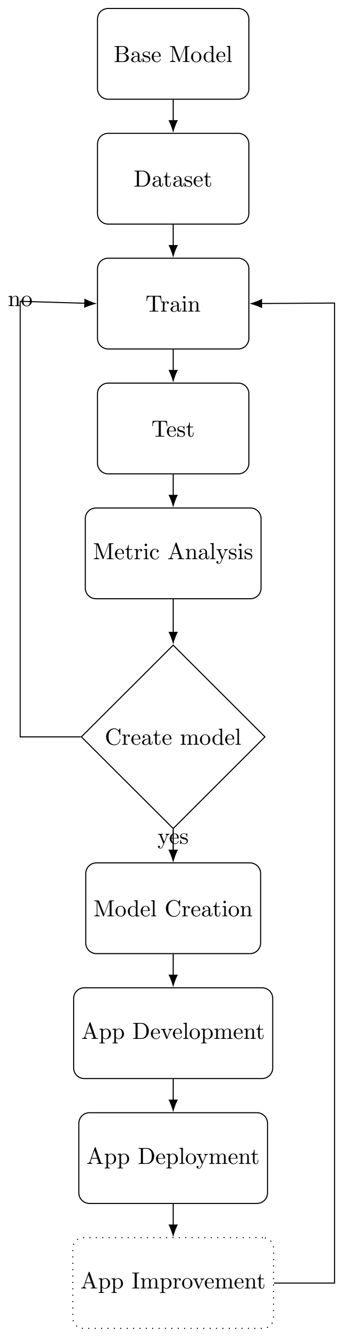
There was a clear idea that this app would be a web application and not a device application. The application needed to perform fast but had to also be light in size. For this reason the app overall, including the model, required to be light.

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.

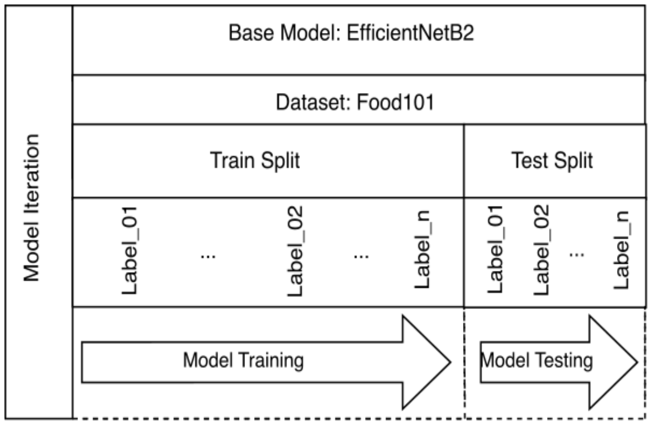


Figure 1: General Base Model and Dataset

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.

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

Table 2: Model Creation Information

7 Project Analysis

Last Iteration

Model information for the last iteration.

10 labels with 35% of the whole dataset.

Trainining Set: 75%

Testing Set: 25%

Percentage of the whole dataset: 2.60% Train - 0.87% Test

The model can predict the following food labels.

1. french fries
2. hamburger
3. hot dog
4. lasagna
5. nachos
6. pizza
7. samosa
8. steak
9. sushi
10. tacos

Labels	Whole-dataset-percentage	Train-split	Test-split	Percentage-Train-Whole-Datatset	Percentage-Test-Whole-Datatset
10	35%	75%	25%	2.60%	0.87%

Table 3: Latest Model Dataset Percentage

The model was trained on different number of epochs.

Latest Model

dataset-percentage	train-split	test-split	labels	deployed	epochs-5	epochs-10	epochs-15	epochs-20
0.35	0.75	0.25	10	1	1	1	1	1

Table 4: Latest Model Information

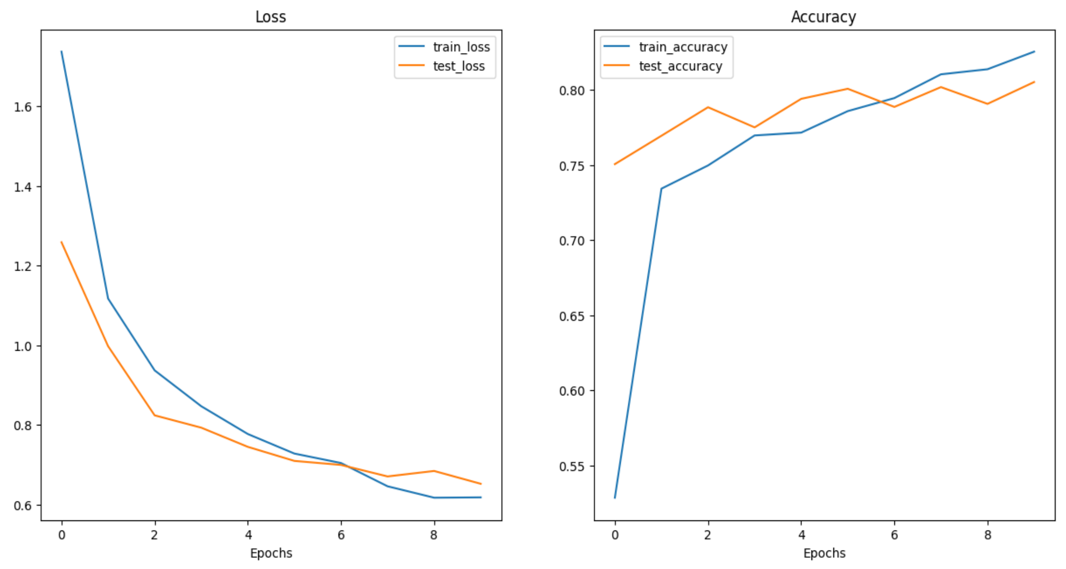


Figure 2: Effnetb2 10 labels 35 percent 10 epochs

The model was first trained on 10 epochs. There was a decision to increase the quantity of epochs. This was possible due to the small amount of labels and small data quantity in comparison with the whole dataset percentage and amount of labels.

The model performed well on the 10 epoch count. The trend on both Loss and Accuracy metrics, for both training, and testing performance showed that there might be a performance improvement if epoch quantity was increased.

The quantity was increased to 15 epochs. There was a slight increase on the test loss but the trend seemed that it could continued to go downwards for both train and test. The same happened to the accuracy metric. The test accuracy decreased and the test seemed that it could go downward but while the train trend seemed to go upward.

More information was required so another training of the model was done with an increase to 20 epochs.

Latest Model Results

The dataset to training and testing the model consisted of 35% of the whole dataset, which was roughtly 350 images per food label. The following were the metrics that resulted of training the latest model for 20 epochs.

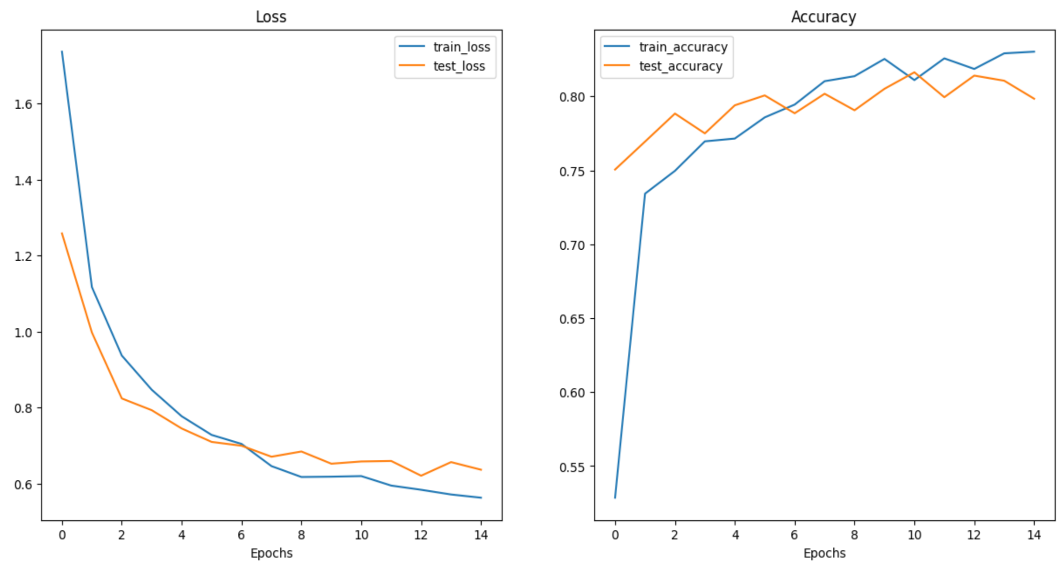


Figure 3: Effnetb2 10 labels 35 percent 15 epochs

The graphs the model *might* start to overfit after the 6th or 7th epoch for Loss and Accuracy metrics. Note: The trends for both metrics are slight and they may or may not be overfitting. Given that the increase was minimum for the loss metric and the increase was also minimum for the accuracy a further analysis was done. Another graph with a zoomed version was required to further analyze this case.

The difference in train loss was around 20 % which is a big difference. The test loss was around 8 %. The difference in train and test accuracy was around 5 %.

The difference in train loss could be a decision factor to discard this model. Even though there was a considerable difference of the train loss of around 20 % there was a decision to keep this model. The test model was not equally as big as on the train loss. The main decision to keep this model was that the accuracy both for train and test was around 80 %. The accuracy was good considering the small quantity of data used to train the model. Note: Given the small percentage of data to test the model the model performance for testing it could be better or worse than the indication on this metric.

The training and test loss were not performing with the desired results, but the model was still decided to be used given into consideration that the accuracy for training and

Epoch	Train-loss	Test-loss	Train-acc	Test-acc
1	0.5501	0.6448	0.8415	0.8018
2	0.5287	0.6715	0.8535	0.7939
3	0.5318	0.6324	0.8494	0.8062
4	0.5263	0.6598	0.8539	0.7939
5	0.5102	0.6354	0.8584	0.8084
6	0.493	0.6344	0.8626	0.8006
7	0.495	0.6497	0.863	0.7919
8	0.4633	0.6265	0.8652	0.7985
9	0.4461	0.6535	0.8727	0.7974
10	0.4621	0.6346	0.869	0.8006
11	0.5244	0.6401	0.8358	0.8074
12	0.508	0.6488	0.8434	0.7928
13	0.5071	0.6123	0.8336	0.824
14	0.4993	0.6494	0.8475	0.8106
15	0.494	0.6309	0.8441	0.7994
16	0.4841	0.6305	0.8419	0.8151
17	0.4474	0.6422	0.8577	0.8119
18	0.4792	0.6183	0.8479	0.8084
19	0.4752	0.6217	0.8468	0.8128
20	0.4354	0.6438	0.8554	0.8128

Table 5: Latest Model Results

testing was good. *Note: There are many ways on which this model could be improved but the key constraint at this phase was time.*

Latest Model Testing Phase

Exploration for the most correct and most incorrect predictions was done.

The model testing phase was performed using the entire test dataset with a batch size of 32. The model prediction results are the following.

The model performed well on almost all of the labels. Sushi, french fries, and steak had the most correct predictions. Nachos was the label where the model had more difficulty predicting. It was almost a 50% chance of getting it right or wrong. Tacos was the second worst prediction.

	Loss	Loss	Loss	Loss
	Epoch	Train	Epoch	Test
Low	20	0.4354	13	0.6123
High	1	0.5501	2	0.6715
		0.1147		*0.0592*
		20.85 %		*8.82 %*

Table 6: Latest Model Results

	Accuracy	Accuracy	Accuracy	Accuracy
	Epoch	Train	Epoch	Test
Low	13	0.8336	9	0.8727
High	7	0.7919	13	0.824
		0.0417		*0.0487*
		5 %		*5.58 %*

Table 7: Latest Model Results

Model Decision

Although both train and test loss were not performing as expected there was a decision to use this model. The accuracy was decent enough and higher than the required metric value. The model was thought to be overfitting, so a model performance on the whole dataset was required to verify this was not the case. The model performed well on the whole dataset. This was what lead to choosing this model. It had good results on almost all labels. There was a specific label ('Nachos') where the model was not predicting as expected, but the overall performance was good enough since it was tipping towards the correct predictions. The overall performance of the model was relatively good taking into account the small percentage of the dataset used for training and testing it.

Note:

The model had to be created for a second time. The results shown here are the results and performance of this last model. For more information please see 'Project Implementation'.

Class name	Correct false	Correct true	total	False Percent	True Percent
French fries	5	80	85	5.88	94.12
Hamburger	16	61	77	18.82	71.76
Hot dog	15	73	88	17.65	85.88
Lasagna	22	78	100	25.88	91.76
Nachos	40	46	86	47.06	54.12
Pizza	10	79	89	11.76	92.94
Samosa	15	75	90	17.65	88.24
Steak	4	80	84	4.71	94.12
Sushi	9	84	93	10.59	98.82
Tacos	26	57	83	30.59	67.06

Table 8: Latest Model Results

Class name	Correct true	True Percent
Nachos	46	54.12
Tacos	57	67.06
Hamburger	61	71.76
Hot dog	73	85.88
Samosa	75	88.24
Lasagna	78	91.76
Pizza	79	92.94
Steak	80	94.12
French fries	80	94.12
Sushi	84	98.82

Table 9: Latest Model Results

Class name	Correct false	False Percent
Steak	4	4.71
French fries	5	5.88
Sushi	9	10.59
Pizza	10	11.76
Hot dog	15	17.65
Samosa	15	17.65
Hamburger	16	18.82
Lasagna	22	25.88
Tacos	26	30.59
Nachos	40	47.06

Table 10: Latest Model Results

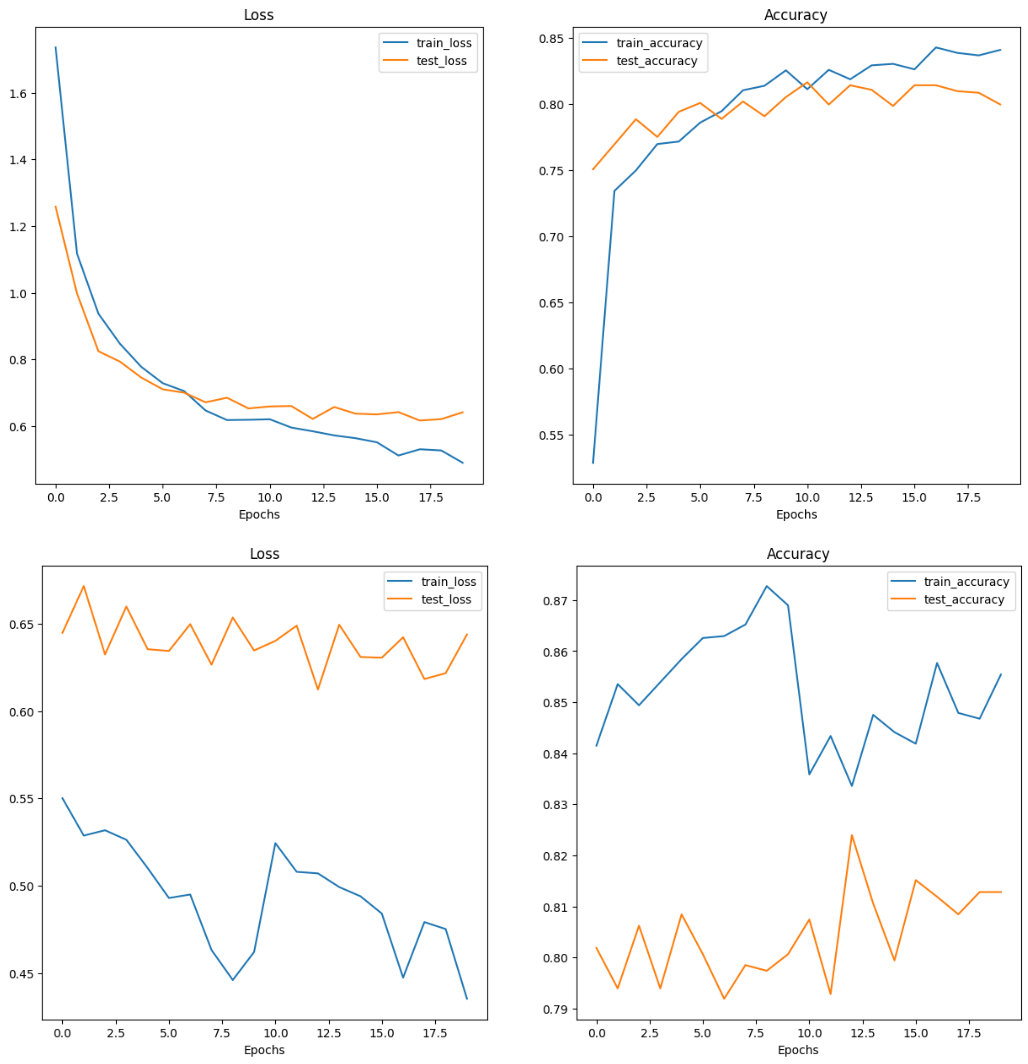


Figure 4: Effnetb2 10 labels 35 percent 20 epochs

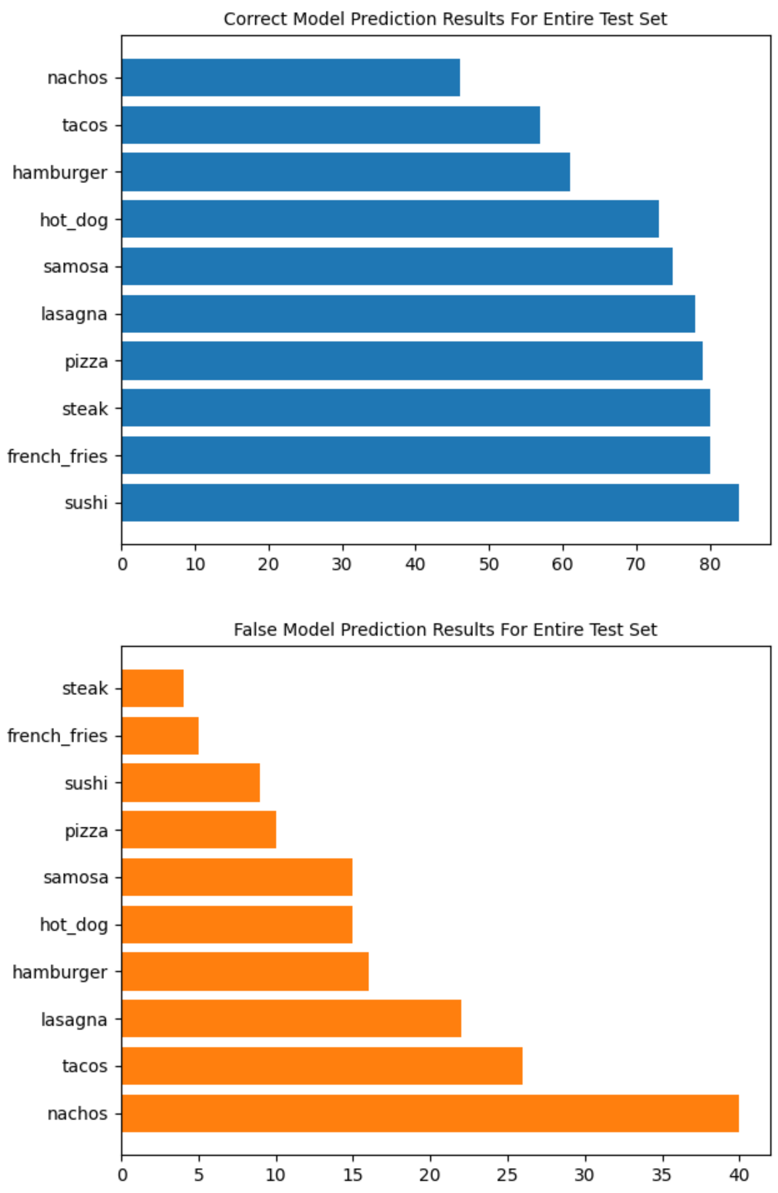


Figure 5: Model Performance Results

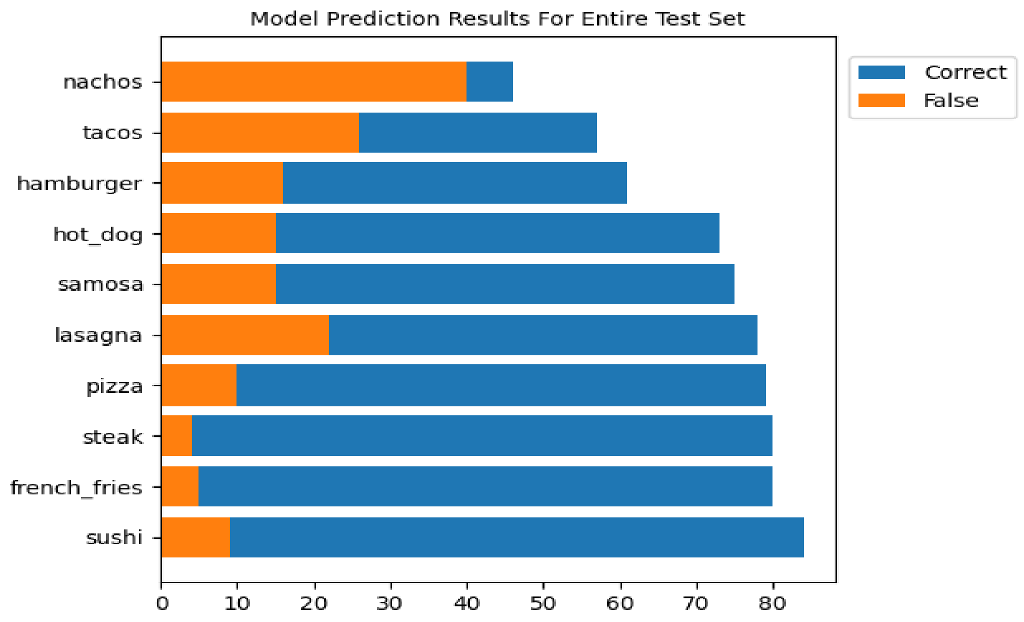


Figure 6: Model Performance Results

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.

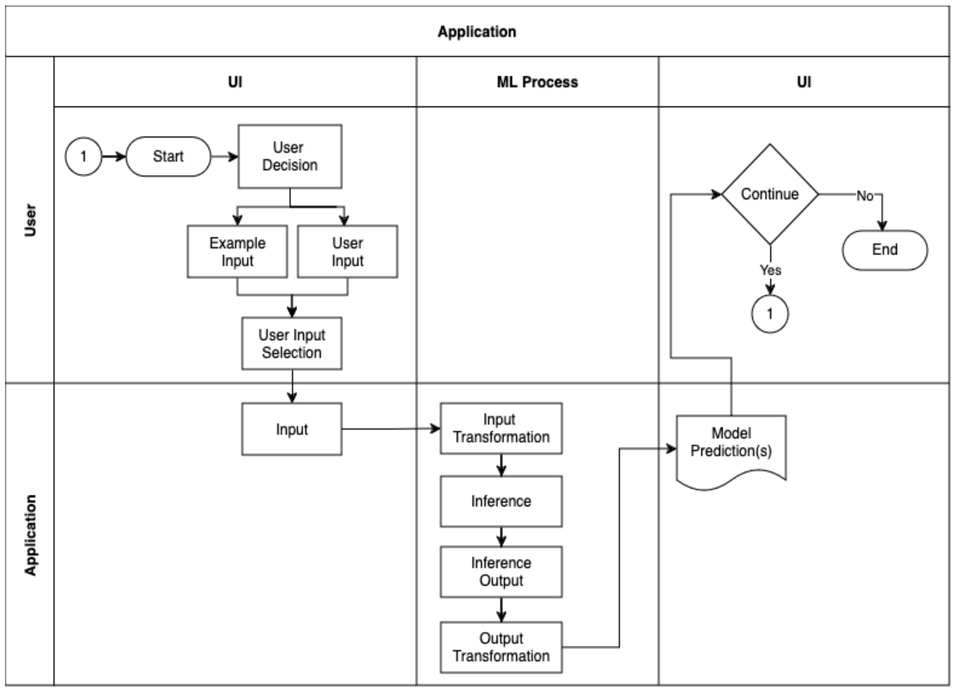


Figure 7: General Web App Process

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]

** At the moment. Please note this might change.*

Deployment 001

This first deployment was focused on making the whole app work correctly. There were several iterations and changes that had to be done in order for this web application to work.

Deployment 001 Debugging Phase The model chose to be deployed was selected and the general process was defined. Two main files were created: app.py, model.py model.py contained the code to create the instance of the model app.py contained all the necessary code to use the model created previously to perform inference.

Issues started to appear at this stage and more time than what was expected had to be required for issues to be resolved. The model was not able to make predictions. After debuging and changing several pieces of the code it was later resolved. The chosen model had to be created for a second time with the same parameters but with different library versions. The results shown on the Project Analysis are for this second version.

After solving the main issues some other modifications to the code were done. The model was deployed and worked correctly. The web app interface was simplified, 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

Results of Deployed App on Example Images

The app was tested on the example images of the deployed web application. Since there were only a few examples the need to use random food images from the internet were used. These are the results of the performance of the trained model used on deployment.

No.	image	Real Label	Prediction label	Pred Percentage	Pred Time (s)	Pred True = 1
1	Example images	hot dog	hot dog	99	1.1536	1
2	Example images	tacos	tacos	95	1.24171	1
3	Example images	french fries	french fries	100	1.00098	1
4	Example images	hamburger	hamburger	100	1.40078	1
5	Image from internet	nachos	nachos	73	0.51269	1
6	Image from internet	sushi	sushi	99	0.20362	1
7	Image from internet	sushi	sushi	99	0.2928	1
8	Image from internet	samosa	samosa	100	1.06309	1
9	Image from internet	pizza	pizza	95	0.83481	1
10	Image from internet	steak	steak	93	1.37752	1
11	Image from internet	steak	steak	88	0.80548	1
12	Image from internet	steak	steak	45	3.90251	1
-	-	-	-	Mean:	1.1491325	-

Table 11: Deployed Example Results

The model could correctly predict all of the example images and the food images from the internet. The average model inference time for this test was 1.1491 seconds. The overall confidence of the model percentages was better than expected. Most of the examples were in the upper 90's. Nachos which was the label the model had most incorrect predictions on the test set had a good prediction with high confidence. There was only a low prediction probability below the 50% mark (steak) and it coincided to be the prediction that took more time (almost 4 seconds).

The model's inference time was higher than anticipated but nevertheless the model performance was better than anticipated*.

**Note: The model's performance on the examples and images taken form the internet might actually be lower. 12 images is still considered a low sample size.*

9 Conclusion

The deployment of the web app was deployed. The overall application was simple but the model performed well, expecially for the amount of information the model was created with.

Room for Improvement

Numerous actions can be implemented to enhance the web app performance. One of the main actions can be model improvement by incrementing label quantity.

PContinuation

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.