The Performance Impact of KNN with Padding Variations

ZEYNEP RANA BASIBUYUK

QUESTION:

Can more efficient results be achieved by combining different padding variations?

Overview

01

Collecting data

Used datasets

- Kaggle :
(product based)
  - small dataset
  - big dataset

02

Cropping Images

Padding percentages used

- 0
- 5
- 10
- 15
- 20
- 25
- 30
- 35
- 40
- 45
- 50

03

Extracting Embeddings from Cropped Data

Extracted embeddings

- {Train + Validation}.npz
- {Test}.npz

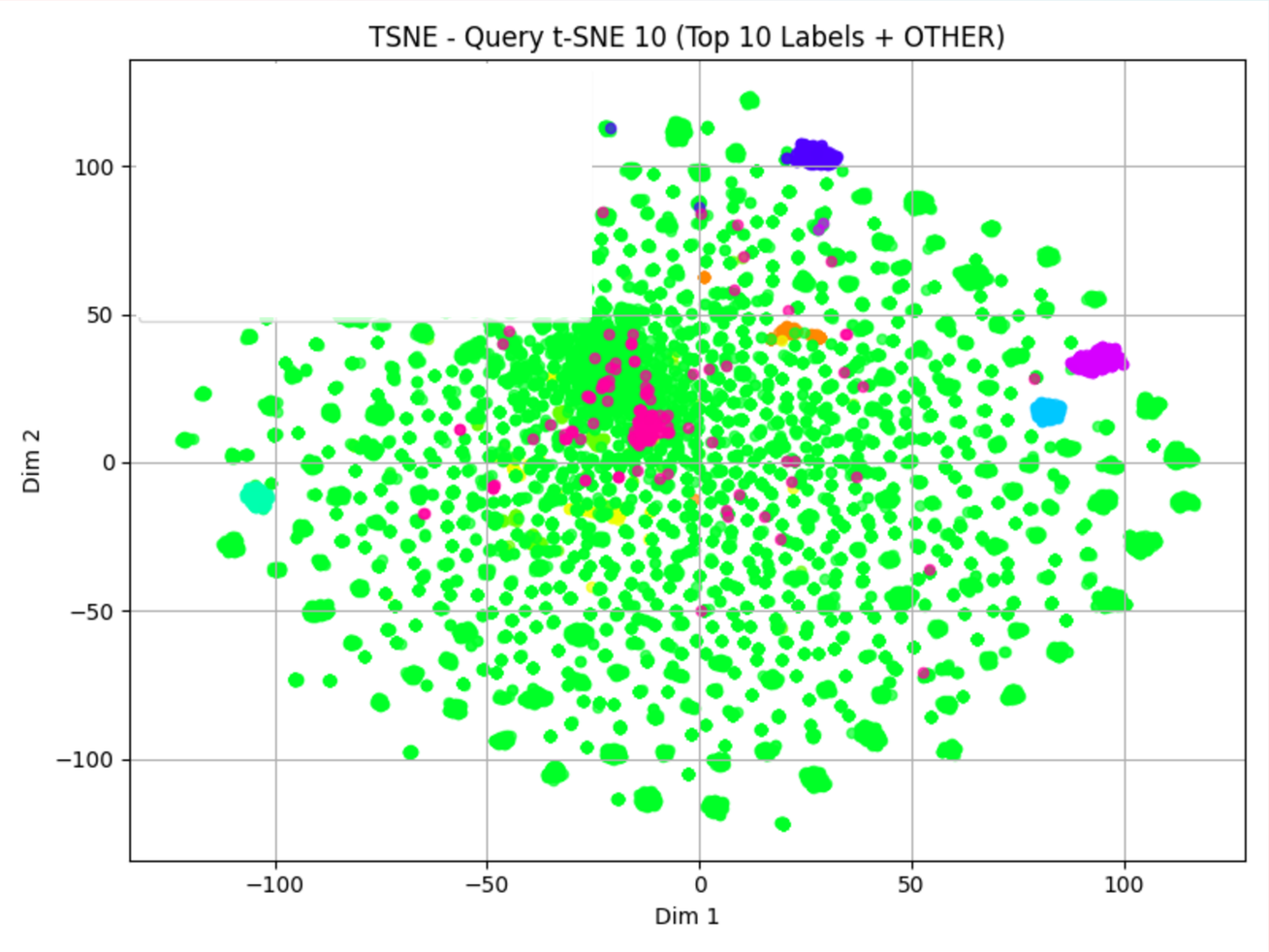
04

Calculating metrics by using FaissKnn

- Accuracy percentage of product-based classification

Small Dataset

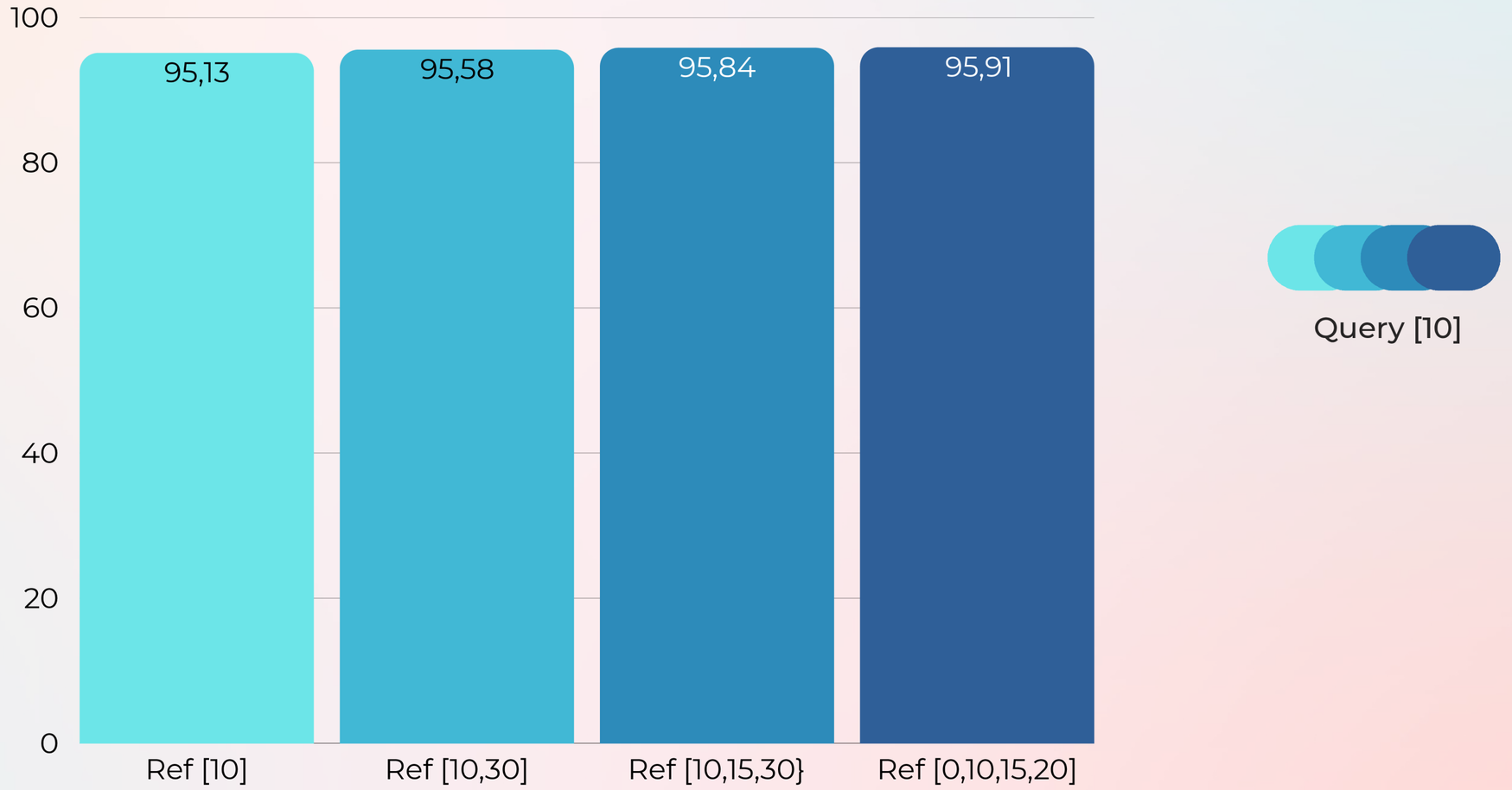
Ref	Query	Acc
[10,0]	[10]	0.9550
[10,0]	[0]	0.9548
[10,0]	[10,0]	0.9549



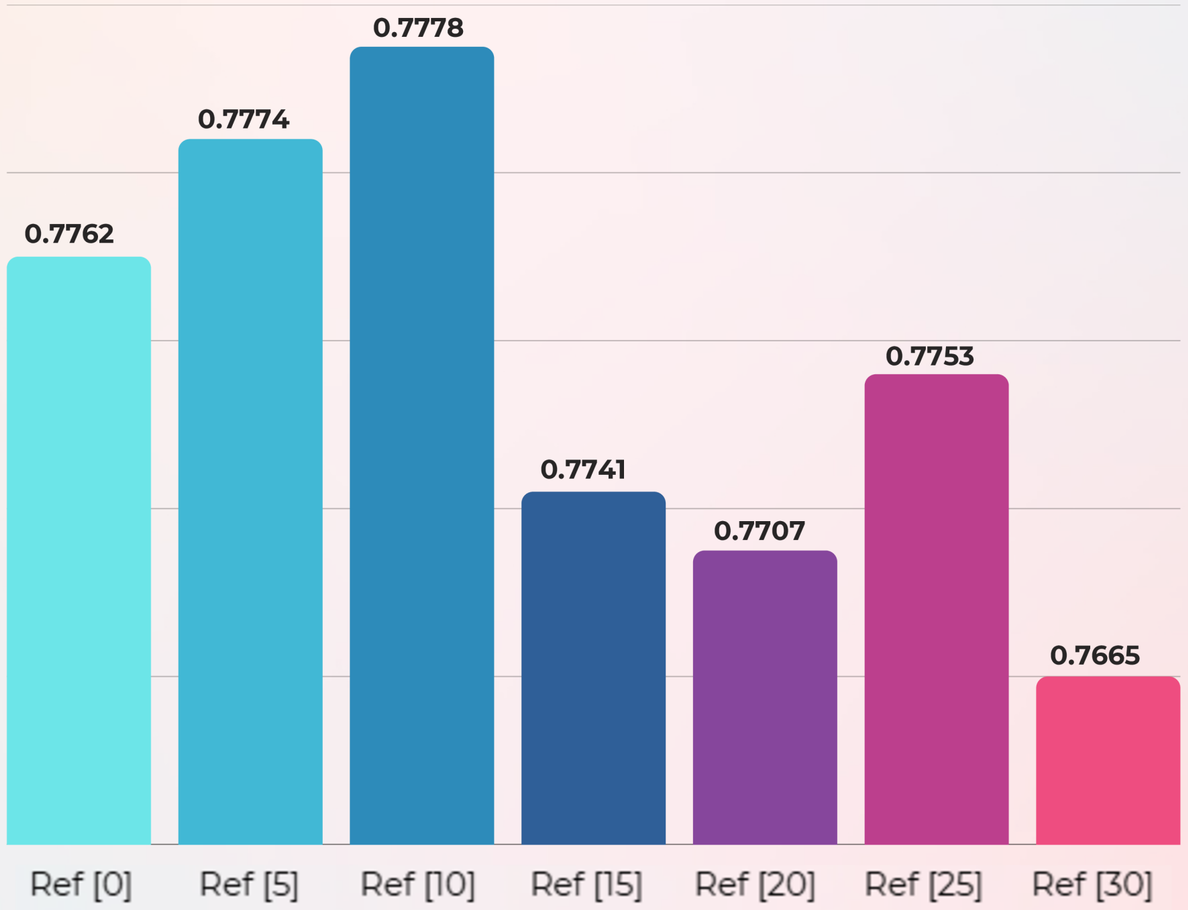
ref[0,10] query[10]

Small Dataset

Combinations with the best accuracy level

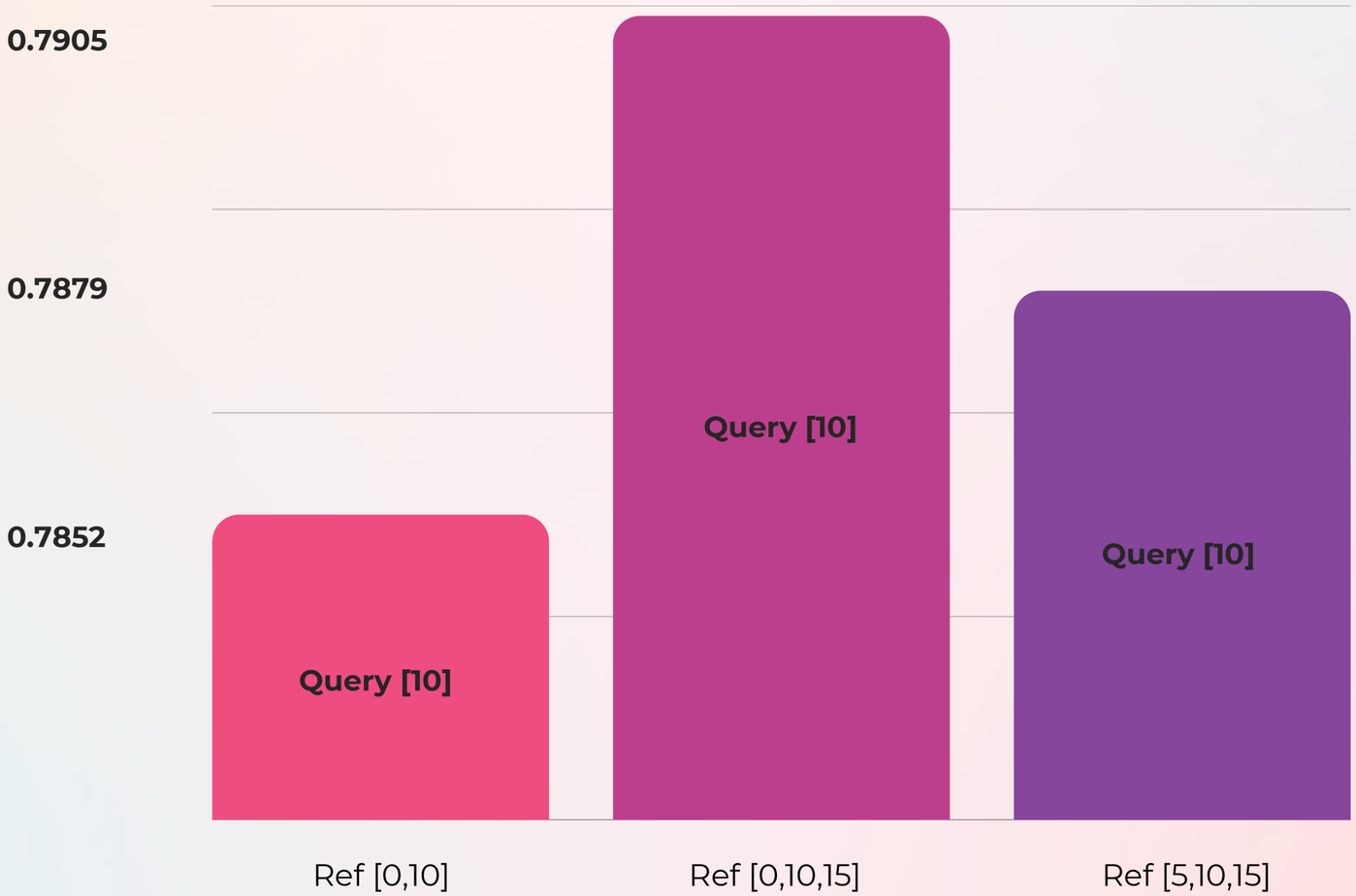


Big Dataset


0.7762
0.7774
0.7778
0.7741
0.7707
0.7753
0.7665
Ref [0]
Ref [5]
Ref [10]
Ref [15]
Ref [20]
Ref [25]
Ref [30]

Big Dataset

Combinations with the best accuracy level

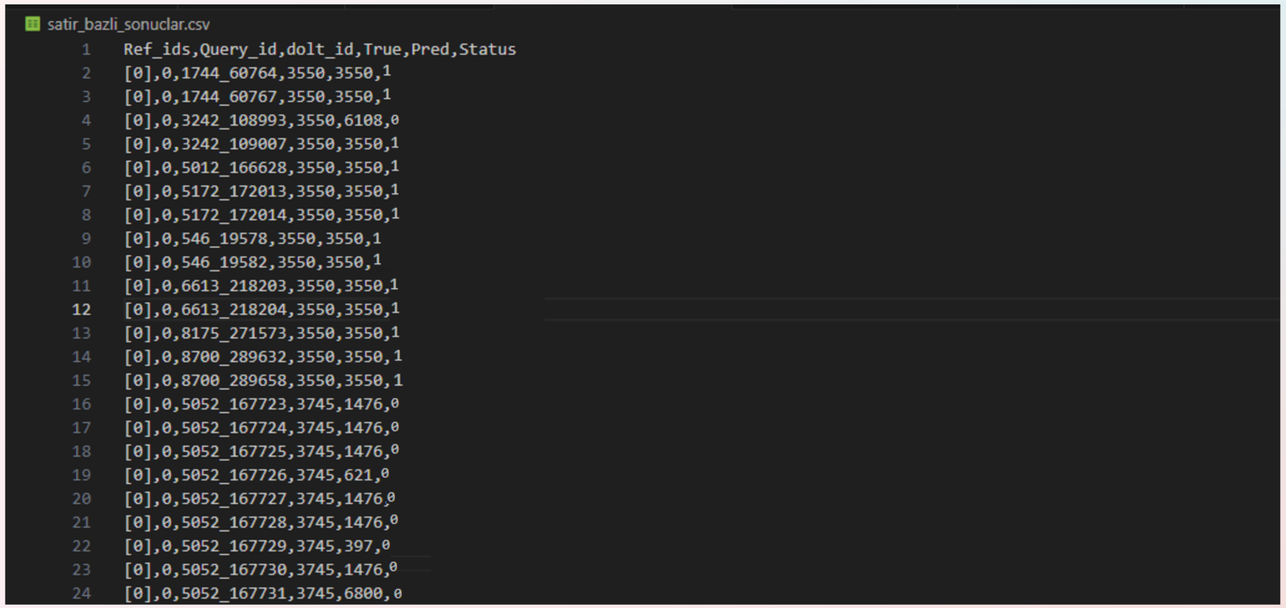


Big Dataset

During metric evaluations, a parquet file was generated for each crop containing the following information:
reference, query, id, class, prediction, and status.

In the parquet file, records with the same ID were checked to see whether they had different statuses in other combination measurements. If such conflicts existed, they were saved into a new parquet file.

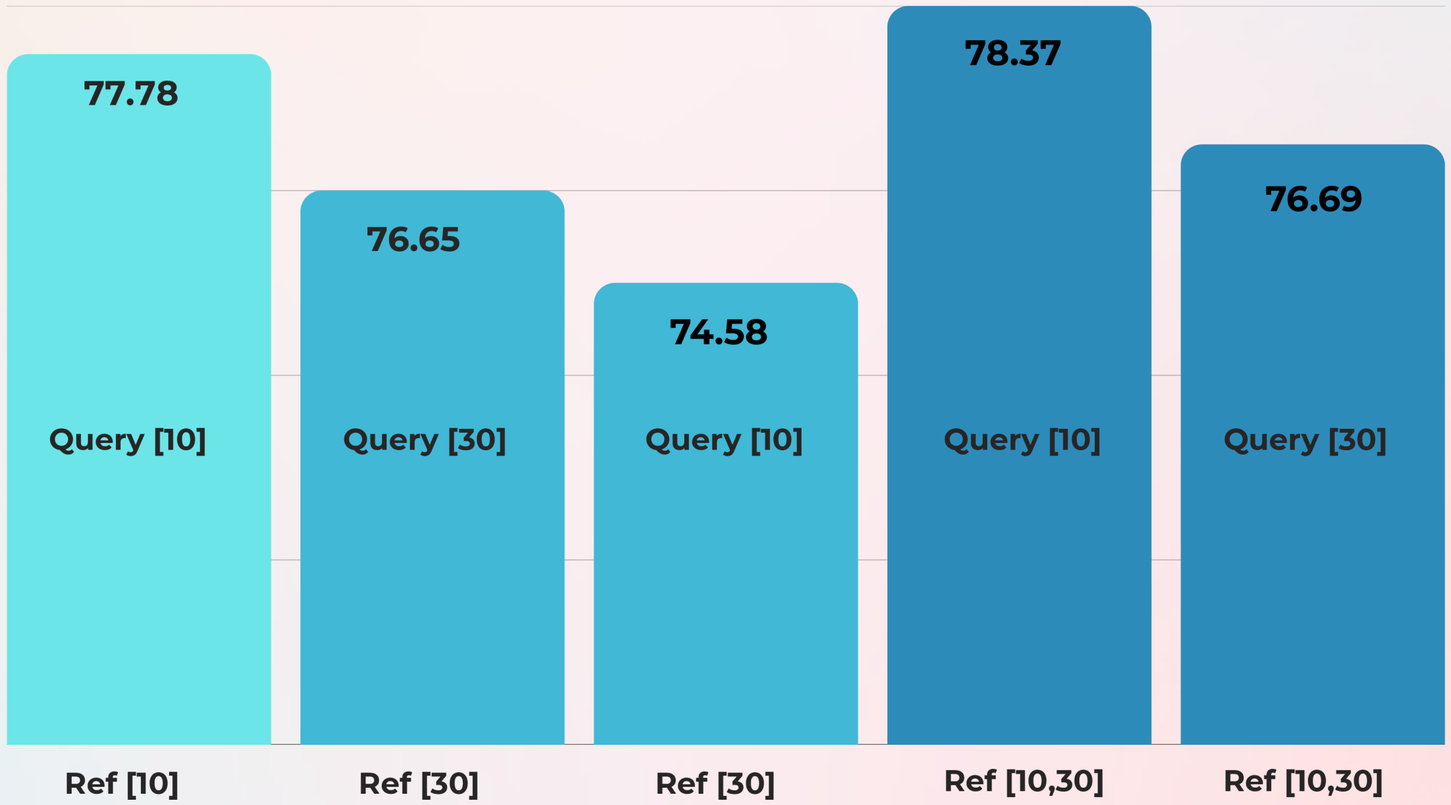
A new JSON file was created using some IDs as examples for cropping.


satir_bazli_sonuclar.csv

```
Ref_ids,Query_id,dolt_id,True,Pred,Status
[0],0,1744_60764,3550,3550,1
[0],0,1744_60767,3550,3550,1
[0],0,3242_108993,3550,6108,0
[0],0,3242_109007,3550,3550,1
[0],0,5012_166628,3550,3550,1
[0],0,5172_172013,3550,3550,1
[0],0,5172_172014,3550,3550,1
[0],0,546_19578,3550,3550,1
[0],0,546_19582,3550,3550,1
[0],0,6613_218203,3550,3550,1
[0],0,6613_218204,3550,3550,1
[0],0,8175_271573,3550,3550,1
[0],0,8700_289632,3550,3550,1
[0],0,8700_289658,3550,3550,1
[0],0,5052_167723,3745,1476,0
[0],0,5052_167724,3745,1476,0
[0],0,5052_167725,3745,1476,0
[0],0,5052_167726,3745,621,0
[0],0,5052_167727,3745,1476,0
[0],0,5052_167728,3745,1476,0
[0],0,5052_167729,3745,397,0
[0],0,5052_167730,3745,1476,0
[0],0,5052_167731,3745,6800,0
```

Big Dataset

Padding percentages that give the highest and lowest accuracy

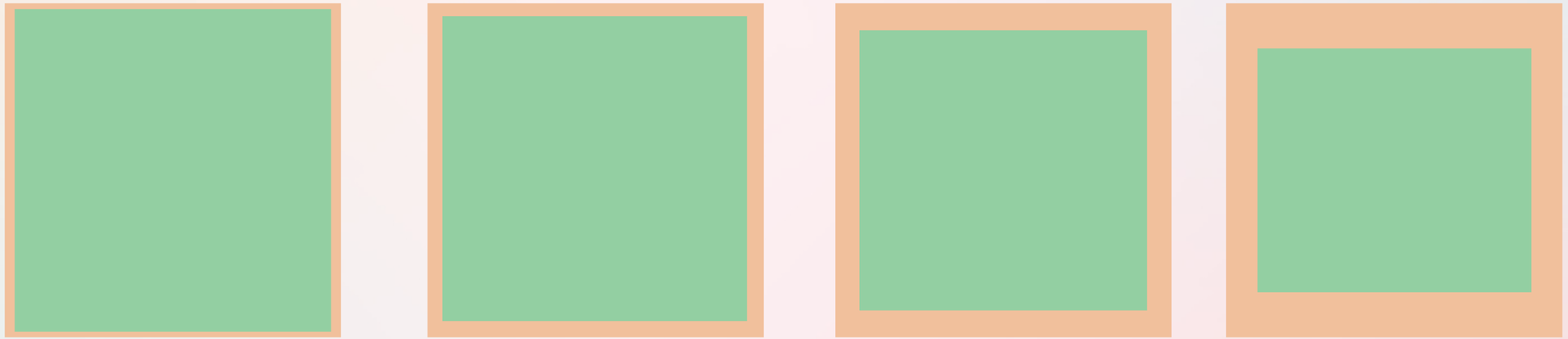


Big Dataset

Some examples of turning false into true with combinations

6532_216105

crop_id



Padding_0 | Padding_5 | Padding_10 | Padding_15

Model's prediction

Big Dataset

Some examples of turning false into true with combinations

Referance	Query	Status
0	**0**	✗
10	**10**	✓
15	**15**	✓
0-10	0	✗
0-10	10	✓
0-10-15	15	✓

6532_216105

Referance	Query	Status
0	**0**	✓
10	**10**	✗
15	**15**	✗
0-10	0	✓
0-10	10	✗
0-10-15	15	✗

228607_3258147

Big Dataset

Some examples of combinations that turn true into false

Referance	Query	Status
0	**0**	✓
10	**10**	✗
15	**15**	✓
0-10	0	✗
0-10	10	✗
0-10-15	15	✗

6642_219073

Referance	Query	Status
0	**0**	✗
10	**10**	✗
15	**15**	✓
0-10	0	✗
0-10	10	✗
0-10-15	15	✗

230147_3307706

Results

- Combining references and queries with different padding percentages showed improved SKU classification accuracy.
- Query [10] consistently gave the highest accuracy.
- Padding >20% generally reduced performance.
- Pairing the best query with the worst reference did not improve accuracy; best results occur when each padding uses its own reference/query.
- Some crops misclassified individually were correctly predicted in combinations, and vice versa. No clear pattern emerged, highlighting the complex behavior of ML models.

Files created throughout the project

- crop.py
- extract_embeddings.py
- evaluate_combo.py
- greedy_combo.py
- npz_to_parquet.py
- row_based_reluts.csv