TASTEME
RESEARCH PROJECT SUBMISSION

Marco Mancini
Department of Computer Science
University of Trento
Trento, TN 38122, Italy
marco.mancini@studenti.unitn.it

1 INTRODUCTION

In the following report we're going to see all the work I've performed on the Research Project for the Machine Learning course of the University of Trento.

The entire project, called Tasteme, is born with the objective of suggesting new food products to the end-user, based on its preferences. This project passed through several hands and different techniques have been applied on the same dataset, without reaching excellent results, but producing interesting ones which have been exploited in this project.

I will start reviewing the old code I had for this project, then I'll introduce and explain the new model selected for this project, called ICNN, also discussing about some result I had in order to test them.

Then we will pass to the clue of this project, starting from an additional data analysis I've performed, passing through the baseline comparison and arriving to a new formulation of the model tuned for this specific task. Finally I will show some qualitative and quantitative result performed over the Tasteme dataset [1].

2 CODE REVIEW

In the following we will resume what happened in the previous Tasteme work. We will also speak about the main model adopted for this project, the Input Convex Neural Networks, also showing some tests I've performed over this model in order to assess it.

2.1 PREVIOUS WORK

As very first step I've been asked to review the existing code, in order to understand what was the path followed by the previous student.

From the scripts analyzed I've understood that the dataset was downloaded as XML files, each one describing one product composed by a set of ingredients and a set of nutrients. The fields were mainly hand-filled by the owners of the products, so there were a plenty of typos and errors. A lot of effort has been put to properly clean and prepare the input to be passed to some Machine Learning model, in fact all those XMLs have been parsed, cleaned and stored into a database.

Then the dataset was passed as input to a Multi-Layer Perceptron, with the objective of predicting nutrients values from ingredients, like in a multi-regression task. Unfortunately the results were very bad, in fact the model seemed to learn nothing and to overfit the data. After several trials, tuning of parameters and changes over the dataset (feature reduction, cut-off strategies, focus on best-distributed nutrients,..), since the performances were again bad, a classification problem was tried. The new problem was to predict if the nutrients were above or under their mean value, using the same MLP tested before. This time the NN was able to reach more than 95% accuracy in very few epochs, so having strange good results.

[1] https://ndb.nal.usda.gov/ndb/

As last reasoning, a data analysis over the nutrient frequencies was run in order to understand the very different behaviour of the same model over those two problems. From the results of this analysis it was possible to see how ~30 nutrients over a total of 38 had meaningless and easy statistical distribution (always the same value for the nutrient's value), letting the classification task be easy to accomplish for the most of the nutrients, so letting the total accuracy increase a lot.

These were the results from which I started working on this project, from now on we will see a new model called ICNN and how I've applied it over Tasteme.

2.2 ICNN

In 2016 a new deep-network model has been studied and proposed by three PhDs and professors at Carnegie Mellon University, called Input Convex Neural Networks Brandon Amos (2017). These potentially deep neural networks are energy-based models, with the peculiarity to have a convex Energy function over a part of the inputs, letting the inference be an easy task.

The structure of the network can be seen in Figure 1, this is the so called Partial Input Convex Neural Network (PICNN), which convex inference is applicable only to the part "Y" of the input. It is a general structure since another one, called Full Input Convex Neural Network (FICNN), exists and it is convex in all the inputs. This structure, together with some constraint over the weights [2] and some mathematical property [3], allows the final Energy function to be convex in Y. In the original paper it is possible to find all the proofs and other properties this model has.

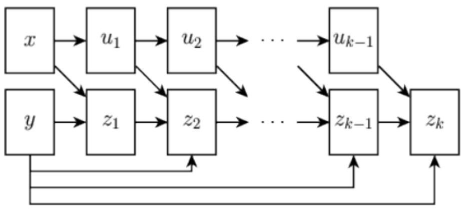


Figure 1: A Partial Input Convex Neural Network

This model has been tested over different domains, such as structured output prediction, continuous action reinforcement learning, image completion, single-classification and multi-label classification tasks, giving good results [4]. So the initial idea was to try it over the Tasteme dataset, but first I had to take the code, understand it and run some experiments in order to assess it.

The git project contained several ICNN codes, one for each domain I've listed above. So I immediately filtered out the Reinforcement Learning and the Image Completion models, since I was focusing more on regression and classification tasks, the ones in which the previous Tasteme project was interested in. Then I took the remaining models, the multi-label one and the synthetic-classification one, with the objective of understanding the code and test them on some synthetic or real dataset in order to see the performances.

Since the model was implemented in two ways, one using the Bundle-entropy method for the inference and the other the standard gradient descent method, I've decided to proceed with the second one, since it was more suitable for the project and also more understandable. Looking at the code and paper it was possible to understand that, for this case, the weights update algorithm was the classical stochastic gradient descent using an Adam optimization, and the inference over the convex Energy function was implemented using again a gradient descent approach, with a momentum term.

Looking at the multi-label code I did realize that a part of the model functions was missing, in fact the code was not supporting PICNN, instead only FICNN were supported. At the end I kept only the synthetic-classification sub-project which was complete but was coded to suit only a single-class classification problem. Then I decided to adjust the code adding the support of multi-label

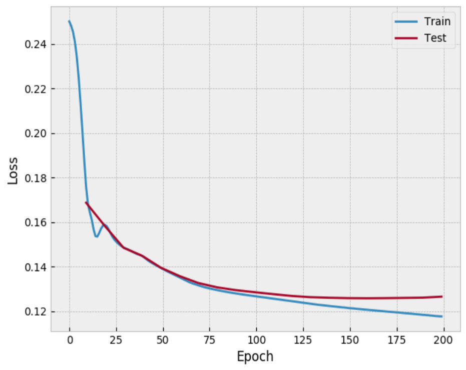
[2]Non-negativity over part of weights

[3]The sum of convex functions is again a convex function

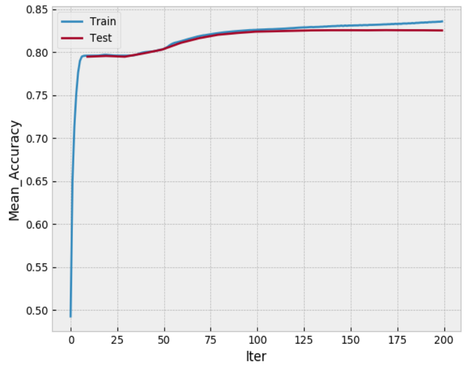
[4]The experiments and the code can be found also on `https://github.com/locuslab/icnn`

classification and multi-regression tasks [5], also adding the possibility to generate synthetic datasets for the above mentioned tasks and including different metrics such as F1, accuracy, precision, recall both for single outputs/labels and also globally calculated.

Some results over multi-label classification tasks can be seen in 2. So it is easy to see how the model seems to perform well on these synthetic datasets. Running other experiments on different synthetic multi-label datasets the performances were more or less always the same, so I moved to test on multi-regression synthetic datasets, which were exactly the same kind of tasks for which Tasteme was born.



(a) Mean-squared-Error



(b) Mean accuracy

Figure 2: Results of PICNN over multi-label classification tasks with a total of 38 classes and 512 floating values inputs

Then I generated multi-regression tasks and used them to fed the network, the loss over both training and test samples can be seen in Figure 3. Also in this case the model learned, but not that much as for the multi-label classification task. In fact the loss is big and seems to start stopping its good behavior when the learning is at 75%. So I tried to pass very different datasets to the model and also to play on the parameters of the model itself [6] but nothing was effective. Finally I also wrote to the PhD student managing the git project, asking him if some multi-regression experiment was run by them but it seemed I was the first to try this task on this model.

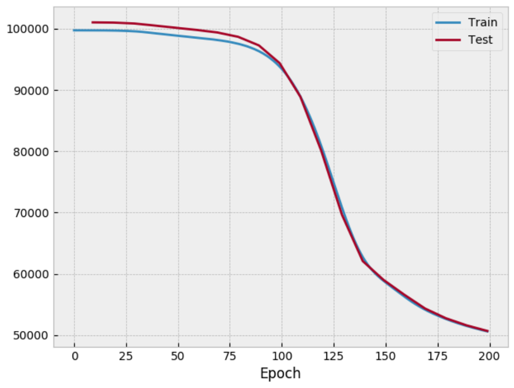


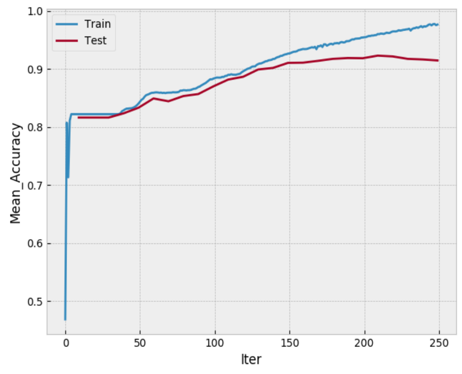
Figure 3: Mean-squared-error calculated over a synthetic multi-regression dataset with 30000 samples, 50 features and 10 outputs.

At this point, since only the multi-label task seemed to give good results, I looked for some real dataset to be used for the testing purpose, and in this Arjun Pakrashi paper I found some interesting

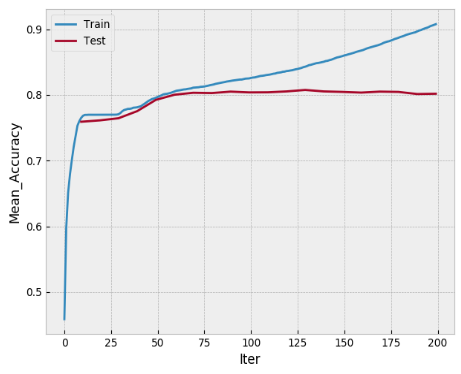
[5]Some adjustments on the structure generation of the model and on the input manipulation.

[6]One important parameter tested was the *noncvx* which allowed the non-convexity of the model, removing the constraints over the weights, and was the most-probable one to give problems for some task.

ones. Then I downloaded [7] them and passed to the network, the results can be seen in 4. The model performed quite well also on these datasets, so this task seemed to be the most suitable for the model in place.



(a) Scene dataset, 2407 instances, 294 features, 6 labels, cardinality 1.074



(b) Yeast dataset, 2417 instances, 103 features, 14 labels, 4.237 cardinality

Figure 4: Results of PICNN over multi-label classification real datasets.

Since the model did perform well over the multi-label task we decided that the best way to apply it to Tasteme was over the reversed dataset, so with nutrients as features and one-hot encoded ingredients as output.

3 Data and Model

In the following we will see how the previously described model has been applied on the reverted Tasteme dataset, starting from the data analysis, passing through the comparison with the baseline and arriving to a new formulation of the inference problem, more addressable for the Tasteme objectives.

3.1 Data analysis

The reversed dataset was composed by more or less 170000 products, the features were the 38 nutrients and the output were the 512 ingredients.

A part of the data analysis was already provided by the previous project, the histograms showing the nutrients distribution and, as already pointed in the previous work, only a small part of these nutrients had a meaningful distribution. In my case the dataset was slightly different, since it was reverted, so I did some deeper analysis on the current input, the set of ingredients per product.

As first I decided to calculate and plot the frequencies of the ingredients. As it is possible to see in Figure 5 there are three dominant ingredients (water, salt and sugar) taking more than 40% of the total amount of ingredients appearances, and this seemed reasonable since their extended usage for cooking purposes.

Then I calculated the cardinality of the input space, the value obtained was 1.88%, meaning that for a total of 512 possible ingredients in average we have products with 9.6 ingredients, which also seemed a reasonable result. This value also confirms the output sparsity already pointed out from the previous work. In order to better understand the sparsity level I bucketized the ingredients in 512 buckets, each Nth one containing products with up to N ingredients. From Figure 6 it is possible to see how the most of the product have from 11 to 13 ingredients and from the 18th bucket we start to have a big drop of ingredients per product.

As last analysis step I run a Principal Component Analysis over the data input, trying to assess the informativity of my features for a possible feature reduction task. Unfortunately the explained

[7]https://sourceforge.net/projects/meka/files/Datasets/

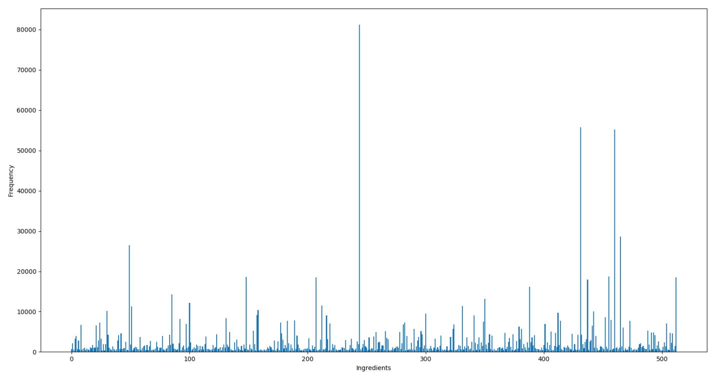


Figure 5: Histogram showing ingredient frequencies. Water, sugar and salt predominate.

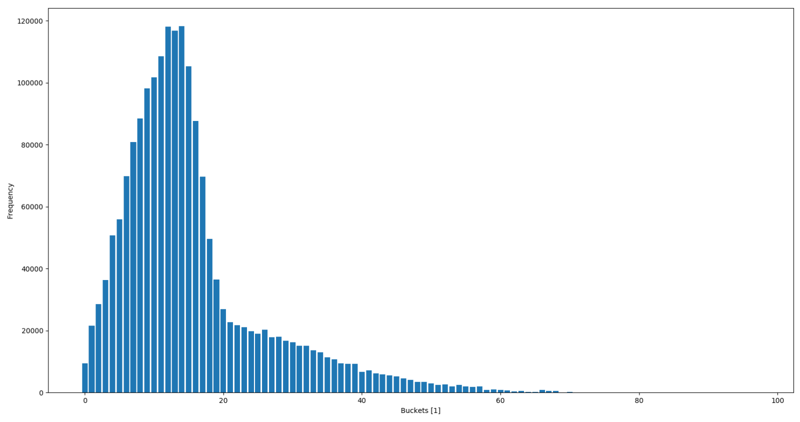


Figure 6: Bucketized ingredient histograms. Bucket I contains product with up to I ingredients.

variance ratio obtained for the most significative feature was 6.1% and the remaining ones were smaller and smaller, meaning that the features were not that much informative and giving me no opportunity to use the principal components as new features.

3.2 ICNN vs Baseline

Then I started modifying the code of the ICNN in order to manage correctly the tasteme dataset.

Even if the multi-regression task seemed not to fit well the ICNN model, I decided to try it over the original non-reverted Tasteme dataset, trying to predict nutrients values from ingredients. As expected the loss was huge, so I decided to binarize the nutrients values as in the previous work, converting the multi-regression problem into a multi-label one. As it is possible to see from the Figure 7 a very few epoch were needed to let the model learn well this task, exactly as the MLP of the previous work over the same problem. This is most-probable due to the fact that the most of the nutrients (+/- 80%) had meaningless distribution and "fixed" values, meaning that their under/over the mean prediction is a very easy task.

So I passed to the multi-label task of predicting ingredients from nutrients values, the results of the model with basic configurations can be seen in Figure 8. It is easy to see how the model was able to reach a mean accuracy of more than 98% over the test-set in very few epochs, letting the task be easily accomplished.

At this point we thought the accuracy was that high since the output level was very very sparse and the most of the ingredients were always off, letting the final problem be easily addressable. So we decided to implement a baseline in order to understand how much the problem was difficult.

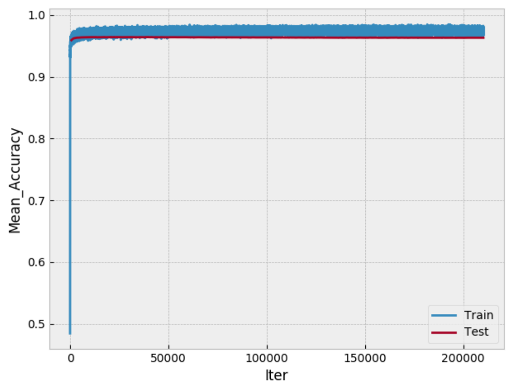
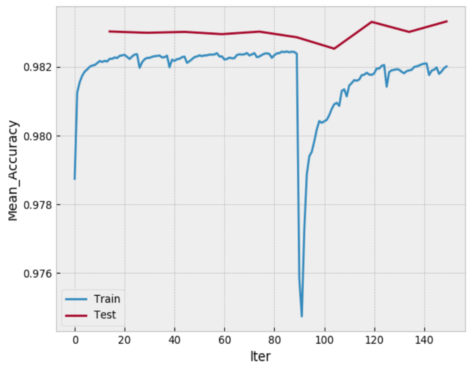
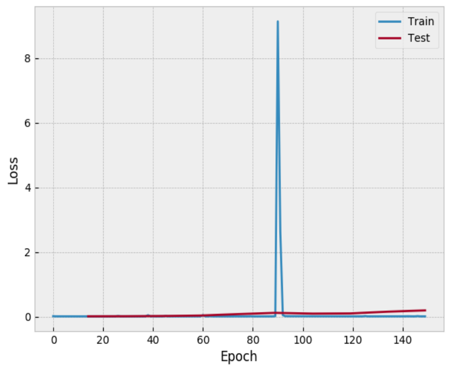


Figure 7: Mean accuracy of the ICNN model over the ingredient to binarized nutrients task



(a) Mean accuracy over 512 ingredients



(b) Mean Squared Error over 512 ingredients

Figure 8: Results of PICNN over multi-label classification Tasteme dataset

The baseline implemented was a simple majority model [8]. Running the baseline over the reversed dataset the mean accuracy was 98.15%, which was very high and close to the ICNN performances, meaning that the problem was probably easy to address.

Then I took the performances over the single ingredients and I compared the performances of the two models over these, trying to understand the reasons of this behavior. As first analysis I compared the difference on the single-accuracies produced by the two models, the main results are showed in Table 1. I decided to report only the three ingredients for which the ICNN had more difference in accuracy, and these are Water, Salt and Sugar, which are also the most-present ingredients in the dataset, as already pointed out in the data analysis section. For the remaining 509 ingredients the ICNN gain was not more than 2% with respect to the baseline, but the accuracies were anyway high for both the model (more than 90% for all the ingredients).

[8] A majority model first count the output frequencies of the training dataset and, given a new input, classify it only based on a threshold over the frequencies previously calculated.

Ingredient	**Baseline accuracy**	**PICNN accuracy**	**Difference**
Salt	57.32%	88.23%	30.91%
Water	60.74%	80.25%	19.50%
Sugar	60.73%	77.94%	17.20%

Table 1: Baseline vs ICNN main accuracies

Ingredient	Baseline F1	PICNN F1	Difference
Water	0%	75.68%	75.68%
Sugar	0%	69.31%	69.31%
Vitamin D3	0%	63.39%	63.39%
Enzyme	0%	47.51%	47.51%

Table 2: Baseline vs ICNN main F1 values

Since the accuracies are not that much informative in a sparse case like this, I performed the same difference analysis over the F1 metrics produced by the two models, the 4 top-most results can be seen in Table 2. In this case the baseline had 0% values for all the ingredients but the salt, instead the PICNN had higher values for more or less 105 ingredients, 1/3 of them higher than 10%. Calculating the difference it was possible to see how , fortunately, the PICNN learned something more with respect to the baseline and the high accuracies of the previous analysis were given by the fact that most of the ingredients were always off.

So, at the end of the day, the PICNN won against the baseline and, even if the performances were not that much high (since the F1 analysis) we decided to proceed with the project, this time trying to achieve a recommendation task.

3.3 A NEW MODEL

The standard inference problem an ICNN performs during the prediction phase is

$$\min_{y \epsilon Y} f(x, y, \theta)$$

Where $f(x, y, \theta)$ is the energy function calculated during the training. Since we're aiming at suggesting new products to the end-user, this inference problem could be slightly changed, the new preference algorithm is:

1. The end-user presents a product x',y' which likes. The inputs x' are the set of nutrients values and the outputs y' are the set of ingredients.
2. The end-user modifies the nutrient values x' according to some reasoning (e.g want more energy, more proteins, less fats,...), generating x̂.
3. The pre-trained PICNN, starting from x̂ and y', solve the following new optimization problem.

$$\hat{y} = \min_{y \epsilon Y} f(\hat{x}, y, \theta) + \lambda L(y', y)$$

   Where L indicate a general loss function applied over the new set of ingredients and the original one.

At the end we will have a new product ŷ which should differ from the original y' one but, with a proper lambda value, should respect the tastes of the end-user since the output should not be that far from y', which is supposed to be user-liked.

Looking at the final minimization problem we can say that, if the Loss function used is a convex loss function (e.g MSE), the whole problem is again a convex one. This because we know that $f(x, y, \theta)$ is already convex by the model and the sum of two convex functions is again a convex one, so we do not break any ICNN property.

4 MODEL EVALUATION

As final phase I have evaluated the model just described. Since the output of the final model is strictly user-based and based on its preference, there was no possibility to have a real ground truth

	0	**0.0005**	**0.005**	**0.05**	**0.1**	**0.25**	**0.5**	**0.75**	**1**	**2**
MSE	0.218	0.218387	0.216121	0.204969	0.202	0.1999	0.1934	0.1933	0.1932	0.1933
Accuracy	0.7815	0.7816	0.7839	0.7950	0.7980	0.80	0.8065	0.81	0.8067	0.8066
F1	0.4180	0.4183	0.4259	0.4515	0.4568	0.4591	0.4646	0.4647	0.4648	0.4647
Changes (%)	0.6999	0.6983	0.5559	0.1410	0.0583	0.0319	0.0049	0.0021	0.0040	0.0011
Changes (N)	125832	125545	99934	25357	10475	5729	875	369	714	201

Table 3: Results of the new ICNN inference over a test set of 35044 products (17977572 ingredients). The ratio applied was a 2x over the inputs Energy and Carbohydrate on all the products in the test set.

to be used for a proper quantitative analysis. In the following we will see a quantitative analysis which will assess the ability of the model of generate reasonable outputs and will also explore the best values to be used for the changes to the nutrients. In addition, since the quantitative analysis cannot give a final metric on the performance of the model suggestion, I also tried to perform some more qualitative analysis on a set of products I personally know and taste.

4.1 Quantitative

In order to understand how much the new model generates reasonable outputs I've used the MLP used in the previous work, the one classifying nutrients under or over their mean value, starting from the ingredients and having more than 95% mean accuracy. More in details what happens is the following:

1. Given a set of products X',Y' the user modifies X' according to some idea, generating $\hat{X}$
2. From $\hat{X}$ and Y' the model applies the new inference problem and outputs $\hat{Y}$.
3. Apply the MLP to $\hat{Y}$ and predict a binarized [9] version of the nutrients, called X"
4. Binarize also $\hat{X}$ and compare it with X", saving the performances metrics both globally and on the modified nutrients.
5. Compare also $\hat{Y}$ and Y', in order to understand how much the model changed the product.

At this point the lambda value of the new inference problem was the most important hyperparameter to set. In fact this value balances how much the Loss function weights in the final convex problem.

Then I created a set of experiments to find a good value for the lambda, these experiments put their focus on two nutrients (Energy and Carbohydrate) out of 38, selected as the best ones in terms of distribution and follows the above-mentioned algorithm using as ratio for the nutrients a 2x, so doubling their values.

The results can be seen in Table 3. How it is possible to notice, the lower the lambda value, the more the final product differs from the starting one. This is probably due to the fact that the Loss function weights a lot in the final convex problem, letting small lambda values be optimal for the changes to the final product. Looking at the performances we can say that the model seems to generate reasonable products, since the global accuracy over 513 ingredients seems to be quite high (+/- 80%) even if the F1 is not that high.

At this point, since our objective was to suggest new and different products to the end-user, I decided to fix the value for lambda to 0.0005, which seemed one of the most promising one in terms of changes. Then I started to run some experiments in order to find the best nutrients on which to play in order to have changes on the final product, they will be then used in the qualitative analysis.

Even if the space of nutrients was not that high (38 nutrients), I decided to start the tests from the well distributed ones which are Sodium, Sugars, Energy, Carbohydrates and Proteins and the main results [10] can be seen in Table 4. How it is possible to see, the higher the ratio the more the changes we have to the final product. It is not the case if we want to decrease the value of the nutrient, in fact we can see how values of 0.25x and 0.5x cause less then 0.15% of changes, which means less than

[9] Binarized means under (0) or over (1) the mean classification

[10] Also the other 2 non-listed nutrients gave me results pretty similar to the ones showed

Energy	**0.25x**	**0.5x**	**10x**	**100x**
MSE	0.2073	0.2062	0.2552	0.2370
ACC	0.7927	0.7938	0.7448	0.7630
F1	0.4519	0.4528	0.3363	0.2179
ACC (single)	0.8696	0.8669	0.7221	0.2992
F1 (single)	0.8688	0.8657	0.7061	0.1432
Changes(N)	27639	24001	464984	3862392
Changes(%)	0.1537	0.1335	2.5865	21.4845
Carbohydrates				
MSE	0.2089	0.2069	0.2362	0.2367
ACC	0.7911	0.7931	0.7638	0.7633
F1	0.4509	0.4527	0.3519	0.1979
ACC (single)	0.8910	0.8897	0.6932	0.3720
F1 (single)	0.8702	0.8677	0.5994	0.0003
Changes(N)	26247	22761	656125	3974916
Changes(%)	0.1460	0.1266	3.6497	22.1104
Sugars				
MSE	0.2062	0.2069	0.2294	0.2531
ACC	0.7938	0.7931	0.7706	0.7469
F1	0.4552	0.4544	0.3548	0.2392
ACC (single)	0.8989	0.9001	0.6875	0.5021
F1 (single)	0.8448	0.8467	0.4360	0.0000
Changes(N)	20736	19863	352598	2637113
Changes(%)	0.1153	0.1105	1.9613	14.6689

Table 4: Results of the new ICNN inference over a test of 35044 products (17977572 ingredients). The lambda value was set to 0.0005.

1 ingredient per product [11]. On the other hand we have that ratios of 10x cause, in average, more than 2% of changes, meaning that the model add/remove more or less 10 ingredients per product, which seems reasonable. Higher ratio values (100x, 1000x,...) generate very big amount of changes (> 15% changes) which seemed to distort the product. Another interesting fact we can notice is that the performances tend to decrease as the ratio increase. This behavior is probably due to the fact that the metrics are generated on the predictions of the MLP used in the previous project, which has been trained over a dataset with "standard" nutrient values and not with the enhanced[12] ones. So it is pretty impossible for the MLP to predict such such binarized values, and probably for most of the products the MLP predicts a 0 value (under the mean) when, instead, the nutrient was enhanced and over the mean.

Then I tried to run a 10x ratio also on all the remaining nutrients, just to understand which were the most effective ones. At the end of the experiments I remained with a total of 8 nutrients on which it was possible to play, which were:

1. Energy
2. Carbohydrates
3. Proteins
4. Sugars
5. Sodium
6. Vitamin A
7. Fiber
8. Fat

[11]Other very small values have been tried, without reaching any big amount of changes

[12]With enhanced I mean with very high nutrient values like the 10/100/1000x with respect to the standard ones

Just for completeness I've run also other tests, trying to focus on pairs and other tuples of the above nutrients and all the tests shown me the same insights I've already mentioned above. So I decided to pass to a qualitative analysis, in order to trying to understand how this model works for a recommendation task.

4.2 Qualitative

For a more qualitative analysis I've selected a set of 30 products I know, subdivided in 5 categories, which are:

1. Cereals and Biscuits
2. Pasta
3. Drinks
4. Dinners
5. Desserts

Then I started playing on the 8 nutrients selected from the quantitative analysis, trying different ratio values and seeing which ingredients have been added and removed by the model. Furthermore, since the output ingredients were not that easily interpretable in order to give a new final product, I've decided to also suggest the closest product in the dataset, with respect to the set of ingredients suggested.

I can confirm what already noticed in the previous analysis, so that the low ratios do not vary the final product as much as the higher ones do. Moreover I've noticed that, in practice, also a 10x value does not vary all the 30 selected products but, in average, varies 7/8 products. More in details, Sodium, Fats, Vitamin A and Fiber need at least a ratio of 20x in order to have changes in a good part of the selected test set, in fact the qualitative tests gave me lower changes of the final product if these nutrients were enhanced.

Playing on the nutrients one by one I can say that the model is not that much able of suggesting good new products. For example, increasing the Sugars ratio of 10x, the PICNN over the desserts did remove a lot of sugars (e.g dextrose, glucose syrup, fructose...) and, sometimes, it adds some sugar. Increasing the ratio of the energy I've noticed that the model tend to add, in most of the cases, the water and a lot of sauces, but these are not that much full of calories. On the proteins the 10x ratio was enough to transform the "PASTA & BEANS" into "SODASTREAM", which should not be that much full of protein. The carbohydrates were the only ones to give me something reasonable, suggesting biscuits for biscuits and drinks for drinks but, at this point, I think it was just lucky.

4.3 A last trial

During the quantitative analysis I've noticed how the performances were decreasing once the ratio was increasing, and I discussed the fact that it was probably due by the values which the MLP did never seen, for this reason a new idea was tried.

The objective was to increase the low performances and, since the MLP was performing "bad" with those unseen values, we thought to increase the training dataset of the MLP using also the enhanced pairs $\hat{X}$,$\hat{Y}$.

So I started to generate different datasets simply passing the original dataset to the enhancing function (with different ratios) and then passing the enhanced nutrients to the PICNN, generating $\hat{Y}$. These datasets then have been concatenated one under the other together with the original one, generating a big dataset of more than 15Gb.

I then tried to pass the entire dataset on the server [13], since the training of the MLP was not possible on my machine with all those samples. Unfortunately the dataset was too big to fit in the space I had on the server.

So I started reasoning on this task and I thought that it was quite useless for the final result, since it was only intended for raise up the performances of the MLP and this would not have changed

[13] The server used was the lion0a

the outputs of the PICNN. So I decided to not proceed with some smaller dataset generation also because, in order to have good performances, I think the whole dataset should be used, and not only part of it.

REFERENCES

Brian Mac Namee Arjun Pakrashi, Derek Greene. Benchmarking multi-label classification algorithms.

J. Zico Kolter Brandon Amos, Lei Xu. Input convex neural networks. 2017.