hello

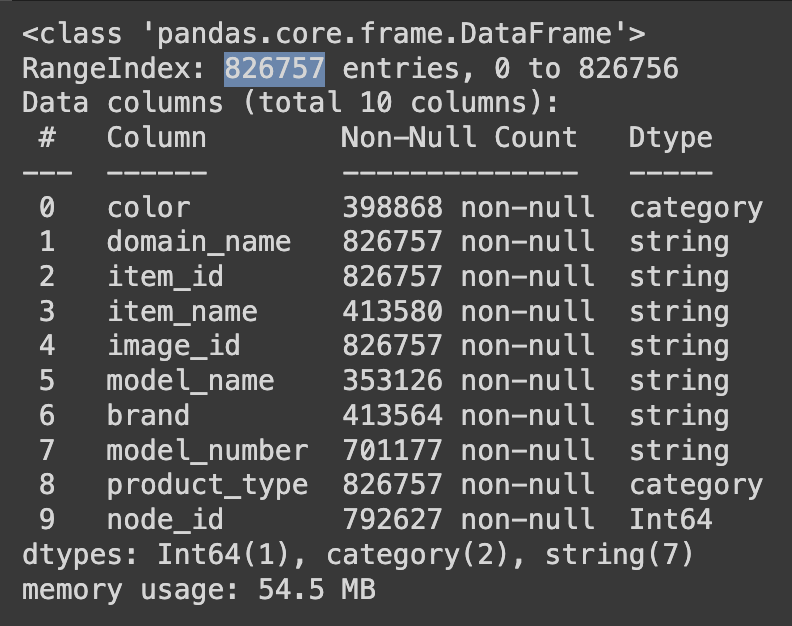
https://drive.google.com/file/d/1EjAnTSAilT7ZBMa9Izb3IaRqN7fAeIJJ/view?usp=sharing

An important point about about this, is that in this case, we just want to predict "product\_type!

-----

Objetivo

The main objective of this project is to develop a neural network that, based on product images, can identify and categorize product characteristics. images of products, can identify and categorize the characteristics of these products accurately. accurately. This includes recognizing brand, color, type and other important specifications. To archive this objective, we must solve a rethink some things of the dataset. This dataset it is about 826.757 entries (fully merged with duplicated values still) and there are a lot of missing values across the possible labels

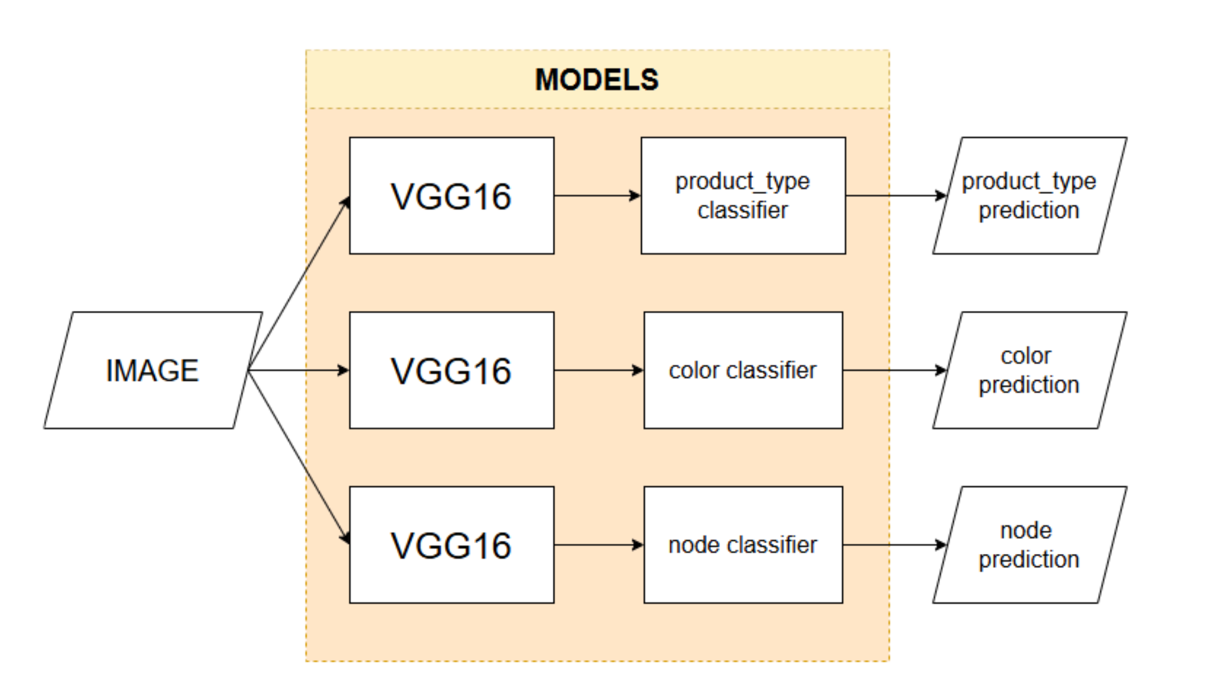


Here we can see that there are some great possible columns with almost no missing values, such as "prodcut\_type" or "node\_id" which has no (or almost) missing values. But others like color or brand will need more prepossessing to make them work in a neural network. Also, we have to keep in mind that the missing values they are not from the same row. For example, to highlight better the problem, suppose that the row number one of the dataset is a chair, number two a table and number three a desk. Peprahs it could happened that in row number one do not have any color assigned (missing value), but in number two and three they have, and row number two do not have brand, but it has all the others. The problem here is that if you want to predict three columns, we have to make three different preprocessing to the data to make it more accurate. Perhaps un one of them is the class is not balanced and in other perhaps it is.

So, in this article I decide to put focus in the process of just one label, which is "product\_type", but keep in mind that in the final solution we made this process to the other labels. Just not to make it very long

Architecture

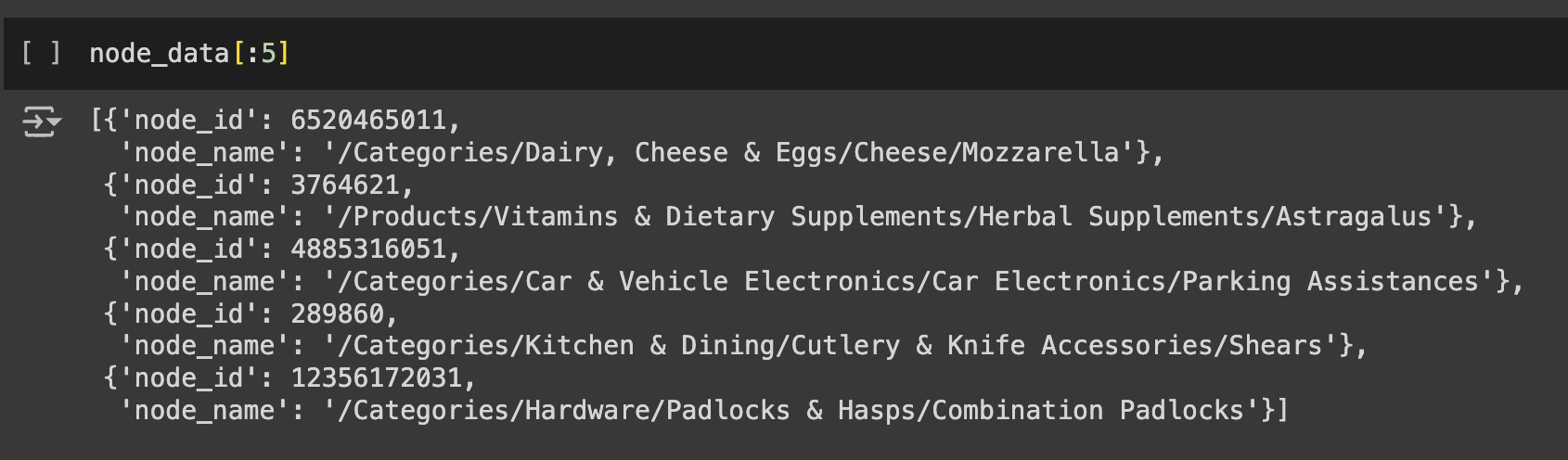
As this is one of my first projects on CNN, I consider a lot of possibilities and decided for one of them, there were a lot of things that make this decision: viability, difficulty and hardware was another one. Keep in mind that I made all this project locally, without any cloud.



This is the final version of the architecture, as I said, we are going to make a different preprocessing for three labels to get three different predictions for each image (it says VGG as an example, we try a lot of models, but this was the idea of the architecture).

Alternative

Other idea that cross my mind was to make a B-CNN: Branch Convolutional Neural Network for Hierarchical Classification, before to briefly explain this solution, I have to explain that it is a solution just for one column, which is "node\_id". As it was seen, almost do not have any missing values, so I tried to use it for a model.



As it is seen, in my case is also divided in subcategories to make it every time more specific, that is the idea of the column, the reality is completely different, because they not have the same root, some of them there are in different languages, and they do not have the same de depth (as it seen). So this column has a lot of potential, but amazon did not make the best job of making this label useful (I explain it quite shortly but I spent hours and even days to make this work). And the solution was pretty original.

The B-CNN is a solution I considered implementing to maintain a hierarchical order between layers and improve the accuracy of predictions. The key idea is that the model would make predictions progressively, from broader categories to more specific ones. This approach can help reduce significant errors. For example, if the model has an accuracy of 70%, I would prefer it to make mistakes on the final, more specific categories, rather than at the broadest level. In other words, it’s better for the model to confuse a bed with a chair than with a mouse—a much larger mistake in terms of classification.

The loss function in the B-CNN is designed to take into account the predictions at each layer, minimizing the chance of a major error by ensuring that predictions become progressively more precise as they pass through each layer.

The idea is way more complex but here is the link where I got the idea:

https://ar5iv.labs.arxiv.org/html/1709.09890

