David Lattimer

8/13/2021

DSC 680

Food Image Classification: Final Paper

Overview:

Image classification is becoming more and more important in usefulness and how companies/people can use it, so can we create an image classification to identify food into one of eleven possible options? I found a dataset of over 12,000 images of different foods, all split into one of eleven different food categories. From here, can we use machine learning to take these images, learn the characteristics generally involved with a specific food, and then recognize the category new pictures fall into? What does and doesn't our model handle very well? And how can we apply a model that is taking in and identifying trends in specific food into other models in the future?

One of the unique things about image classifications is that you can build a model to look at pictures, not just numbers in a dataframe (even though images ARE a series of numbers to a computer), and then predict other images with what it learns from the previous images. As weird as it sounds, when we see an image, a model sees a bunch of numbers, and if a model can take these numbers, find patterns to associate with one of the eleven categories and then predict which type of food it is by these associations and patterns it notices. Image classification has many use cases, such as the famous MNIST dataset to recognize handwritten numbers, but if we can identify and recognize food, we can expand on this in a myriad of ways.

Defining the Problem:

This is my first delve into image classification and building a model to work with images, and a lot of the ideas I have revolve around adapting these classifications into usable apps and models to help users with problems they may have. In order for me to build an app or website that can help with some of these problems, I need to first get comfortable with image classification and learn about how it works and some of its shortcomings to see if these ideas are even possible.

The idea I have is essentially an app that a user takes a picture of their pantry, refrigerator and freezer (or just all the food and ingredients they have). From here, the app uses image classification to figure out the ingredients the user has (and clear up anything that it may be unsure about), and then uses either an API or scrapes a recipe website and returns all the recipes that a user can create with the ingredients they have on hand. From here, there would need to be a way to sort through recipes, whether it be types of cuisine, cooking times, or filtering out recipes that I expect to see way too often because the ingredients are relatively low. There are lots of cases to look out for and take suggestions on once things get up and running, but it does sound useful and like it could help so many people find recipes, use the food in their house and try new things. Whether it's someone who is tired of throwing out food they bought and didn't end up using, or someone who just wants to cook something new, it could be useful. And with options to sort by time to cook, someone can make something either quickly or open their options up if they have more time.

Now enough of an elevator pitch, there are many stages of this project that would need to be handled and then find a way to combine them all into something that can be used by just about anyone. The first of these is the ability to take a user's photo and identify all the ingredients and items they own, which would likely be done through many pictures of many different products, and then anything we can't identify, we can ask the user to point out.

Although we want the app to be accurate, we have to know it will be less frustrating to use if the user can correct or enter information for the item we can't identify so we can potentially use it in recipes. The best part of this though, is that we now have more images of items and can keep updating our model and build stronger models based on the user submissions that come in. By reinforcing our model with user images, the model should improve and the user experience should keep getting better.

Of course, this project itself is scaled down severely. It mostly won't contain any usefulness for this larger scale project, but is in a familiar realm and will help me become familiar with how a model will work to identify and classify these images of food and identify them in the future. The most complicated part by far is using images of people's pantries to try to identify these items, so if we aren't able to identify and classify just 11 different categories, we will be in trouble when we need to identify far more of them. Once we are able to compile a list of each ingredient a user has, we should be able to scrape and find recipes fairly easily based on the user's request. Maybe we have an option that an ingredient MUST be included, now it filters our search results even more and can get the user what they want to see and have fun cooking.

I know this problem is much different than the one I am proposing, and nearly all the food that we have images of and are identifying and classifying are already cooked dishes and won't be in people's fridge, it still gives me a chance to see how feasible food would be to identify and classify. This is step zero to see if it is even possible and something worth pursuing before trying to tackle the problem and create something that would actually work.

Data Understanding:

The data was taken from the EPFL website and was brought to Kaggle in a more organized and easily usable dataset. The data is over 12,000 images of food in separate folders based on 11 classifications (Bread, Dairy product, Dessert, Egg, Fried food, Meat, Noodles/Pasta, Rice, Seafood, Soup, and Vegetable/Fruit) and split into a training, validation and evaluation grouping. From here, the model feeds the images in and learns with which classification it is under to teach the model the patterns and characteristics of each image. Of course, the model reads through each image pixel by pixel along with the color of each of these pixels and then identifies colors and patterns that are indicative of each of the classifications.

Overall, there isn't much data associated with the images in this dataset. There are images and tags along with each image that is one of the eleven categories it belongs to. This is split into training and testing data and the image is used to predict the category it is in. No other data exists and the model does all the work. This is the sort of model that clearly works when you run it through and it can predict the classification with high accuracy, but it is hard to know how exactly it works and nearly impossible to know what the model is doing under the hood and look at the sorts of patterns and characteristics the model is finding. Of course, if it works it is useful, but in terms of explaining and breaking down what is happening, it makes it a lot more difficult. Hopefully the results and accuracy is good enough to make up for it.

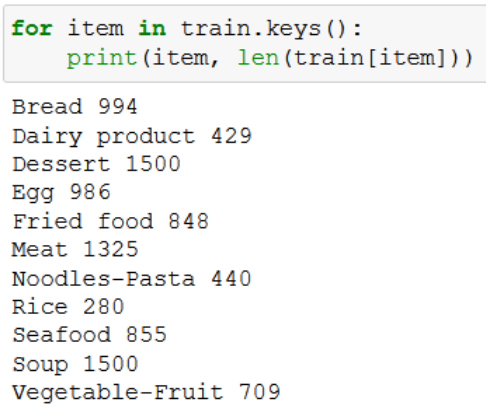
Data Preparation:

There is not a ton of preparation needed to feed the images into the model to be broken down and to find the patterns needed to identify the classification it belongs in. Although the data comes already prepared and ready to be used, there was still a lot needed to handle the data and feed it through the model. Using the os library to navigate from one folder of images to the next and gather both the image itself along with the classification tag along with it was difficult and something new to learn. After learning about that and how to iterate through the folders to jump from one image to the next, there were a couple ways to clean up the images, such as resizing the images to 256x256 so that the model can break it all down pixel by pixel.

Other than these small tweaks and exploring new libraries, this section is pretty minimal because there isn't much that needs to be done. The user who uploaded the images to Kaggle cleaned up and prepared the images into folders so there wasn't much that I needed to do luckily. I imagine image sorting and organizing and moving into usable folders for this project would take a lot of work, especially if we are forced to gather a group of images large enough to run a machine learning model with. Luckily for me, the hard work was all done already and was set up for creating this, so the data preparation was fairly straight forward and minimal.

Modelling:

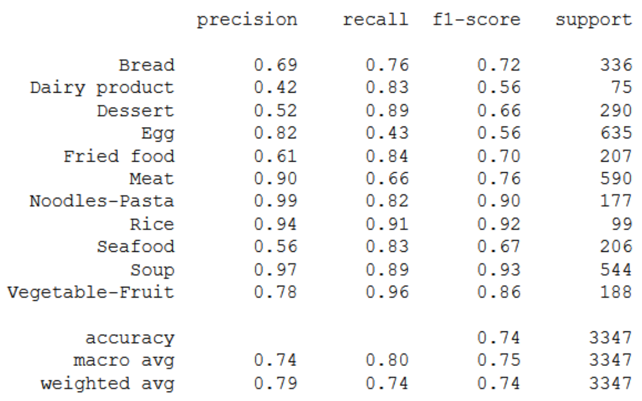
The model that I used to help with image classification is a neural network. The model is called the InceptionResNetV2, which is designed to take in images as an input and can categorize them into one of the eleven classifications we set for it. This takes each image as a 256x256 pixel image (so 65,536 individual pixels per image) and associates patterns and colors typical for each classification. The neural network is then able to be taught with a large group of training images and break down each individual pixel and how the combinations of pixels that make up the image typically fall into a


```
for item in train.keys():
    print(item, len(train[item]))
```
```
Bread 994
Dairy product 429
Dessert 1500
Egg 986
Fried food 848
Meat 1325
Noodles-Pasta 440
Rice 280
Seafood 855
Soup 1500
Vegetable-Fruit 709
```

specific classification. This then gets applied to a set of new images that it doesn't know the category for and can sort them into the eleven categories it is set up with.

The code for this project takes a long time to run, mostly because of the complexity of the images when compared to some other neural network examples. The most common neural network for using image classification is the MNIST dataset, which is a group of 28x28 pixel images that are all in greyscale. When you expand to this dataset where the images are 256x256 and in full color, the neural network has a lot more work to do to find these patterns. It will create a matrix of each color of each pixel in the image and when breaking these down into a series of numbers in matrix form, can associate and recognize characteristics that are in each group of images.

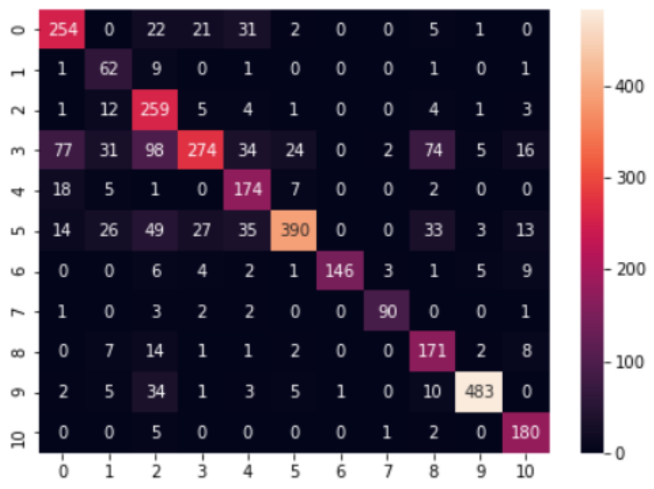
When learning from all the training images and then weighing the inputs based on these images, it can take in the testing images and sort them into one of the eleven categories relatively successfully. Of course, we can look into which of the classifications it handles well and which it struggles with. And we can see where the model might be getting confused as well. As we can see, the model handles soup, rice and noodles/pasta very well, and struggles with dairy products and eggs. Overall, we have an accuracy across all the testing images of 0.74, which is very successful. And now we can take a look at the confusion matrix to see the



```
                 precision    recall  f1-score   support

          Bread       0.69      0.76      0.72       336
  Dairy product       0.42      0.83      0.56        75
        Dessert       0.52      0.89      0.66       290
            Egg       0.82      0.43      0.56       635
     Fried food       0.61      0.84      0.70       207
           Meat       0.90      0.66      0.76       590
  Noodles-Pasta       0.99      0.82      0.90       177
           Rice       0.94      0.91      0.92        99
        Seafood       0.56      0.83      0.67       206
           Soup       0.97      0.89      0.93       544
Vegetable-Fruit       0.78      0.96      0.86       188

       accuracy                           0.74      3347
      macro avg       0.74      0.80      0.75      3347
   weighted avg       0.79      0.74      0.74      3347
```



	0	1	2	3	4	5	6	7	8	9	10
0	254	0	22	21	31	2	0	0	5	1	0
1	1	62	9	0	1	0	0	0	1	0	1
2	1	12	259	5	4	1	0	0	4	1	3
3	77	31	98	274	34	24	0	2	74	5	16
4	18	5	1	0	174	7	0	0	2	0	0
5	14	26	49	27	35	390	0	0	33	3	13
6	0	0	6	4	2	1	146	3	1	5	9
7	1	0	3	2	2	0	0	90	0	0	1
8	0	7	14	1	1	2	0	0	171	2	8
9	2	5	34	1	3	5	1	0	10	483	0
10	0	0	5	0	0	0	0	1	2	0	180

shortcomings of our models. We saw issues with dairy products and eggs from the classification report (1 and 3 in the confusion matrix). Although dairy products don't seem to have performed too badly, we have a very small sample and so it was easily swayed. On the flip side, we have a ton of egg pictures and they are being classified all over the place. Overall, our model performed well and could be tweaked to perform even better. Adding pictures to the training and test data would also always help, but 0.74 total accuracy between eleven different classifications is great.

Conclusion:

Although this project is a different concept than the overall project I want to be working on and trying to develop, it is good to see it perform relatively well. Getting an accuracy of 0.74 using this model is great and can keep getting improved, especially if we start to incorporate user images. Of course, there are many other issues that need to be handled and adapted to make a usable app, such as the ability to recognize many different products in the same picture to match them to a similar classification such as this and working on the speed the model can return results. Luckily speed will be cut down when examining just a single picture, but there is a lot to do on many sides of the potential app.

For this project in particular, the neural network performed well and returned a relatively accurate result, although it handles certain food categories better than others. It is encouraging to see it perform well, especially due to how hard it is to see exactly what is happening under the hood and how hard it is to think about a matrix of each pixel being made for each picture and patterns getting associated with food groups. A neural network to recognize images and their contents would be the way to go if I wanted to pursue making an app, along with all the next steps from that point, but this was a good jumping off point to working on that sort of work.