Duplicate Detection of Product Listing

Kshiteej Manoj Gilda

NIIT University

Problem Statement:

We at some random e commerce company, strive to maintain most organized catalogue of products. One of the key issues we tackle is duplicate detection of product listings. Duplicate listings occur due to couple of reasons, for one, seller could upload the same listings across multiple e-commerce sites and other being the seller uploading the same listings multiple times within a e-commerce site for reasons only known to them. You are supposed to propose a solution to the same.

Proposed Solution:

1. To extract the encodings of the product images using the VGG16 model.
2. Classify the same using a Multi class classification algorithm like K Nearest Neighbours or SVM.
3. The products for which the classification of their images does not match with their class is a duplicate of the product it has been classified into.
4. The features of the images can be combined with the features of the title and description for better classification results.

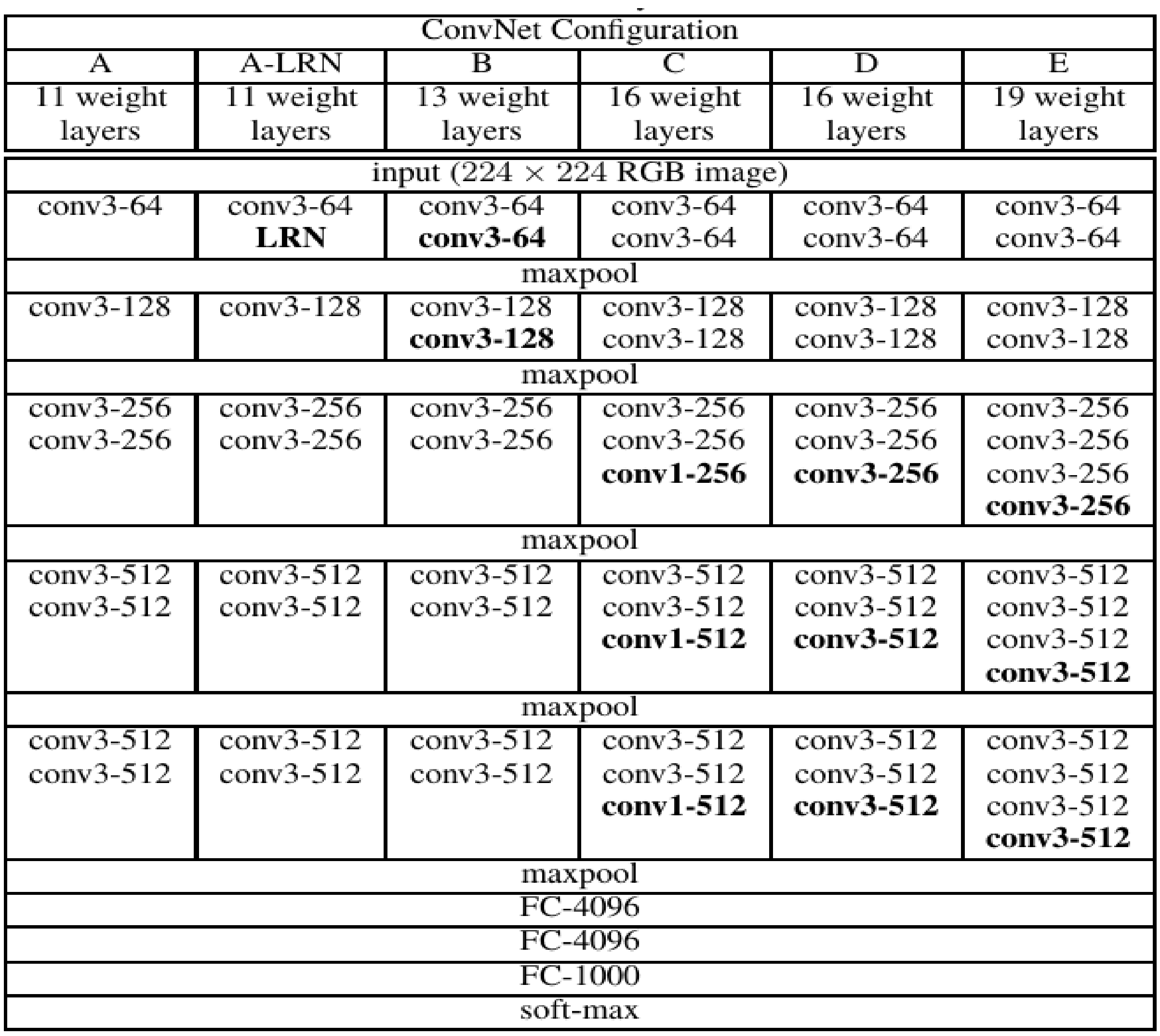
APPROACH

Preparing the Data

As per the given problem, only the data of the subcategory ‘Tunics’ is required. I tried importing the subcategory through Prestashop but the source file was too large for Prestashop. I then looked for the category in Excel and copied 12057 samples to a new csv file. The csv file has 32 columns. The file has a column ‘imageUrlStr’ which has 4 image URL’s per product separated by the delimiter ‘;’. By using the split function, we can create a 2D array of all the URL’s.

Extracting the encodings of the image through the use of VGG – 16 model:

The VGG 16 model is a pretrained 16 layer Convolutional Neural Network model. VGG 16 has been trained on the ImageNet database and promises very high accuracy figures. It is available in the Keras library with pre-trained weights for direct implementation. The architecture of the model is as follow:



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Source : https://arxiv.org/abs/1409.1556

The encodings are basically 1000 floating point numbers per image which can be used for classification. We can calculate the encodings for the images now (I have used 1604 sample product images for the same).

Applying a Multi class classifier on the image:

We can now use the K Nearest Neighbours classifier or SVC to classify the images. I have used the encodings as the input feature and the product indexes are our expected outputs. After training both the classifiers on the 1604 product images, I received the following results:

S. No.	Classifier Used	Accuracy of Predictions
1	Support Vector Classifier	20.82 %
2	K Nearest Neighbours (k = 20, weights = 'distance')	47.88 %

One possible reason, why KNN is performing better than SVC over here could be that the training data is very less in the case above. KNN does not require much data compared to other alternatives to make predictions.

Finding out which products are duplicates:

Finally, now that the model is trained we can now make predictions on the same training data to classify them. The products for which the classification is not matching with the expected classification is a duplicate of the one it has been incorrectly classified as. We can create a dictionary file which has the main product as the key along with the duplicates listed as the array values. The same can now be exported as a JSON file.

Experiments with Text Features:

To improve the accuracy of our predictions we can also use the text values like the title, description and colour of the product. I experimented with a Hashing Vectorizer to generate the features for the same. But the classification results were not that promising. Another promising approach is the usage of Convolutional Neural Networks for text classification. A combination of the same along with the VGG 16 model seems like the perfect fit for our problem.

References :

1. https://www.kaggle.com/keras/vgg16
2. https://github.com/fchollet/deep-learning-models/blob/master/vgg16.py
3. http://cbonnett.github.io/Insight.html
4. https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/
5. http://scikit-learn.org/stable/modules/neighbors.html
6. http://scikit-learn.org/stable/modules/svm.html