GokulaKrishnan
Product hacker / Dota enthusiast / International fellow at fast.ai/ oneironaut
Draft

Diving into Deep learning— Part 2— Building fraud detection X-Ray image classifier—First place in leaderboard

Experiments solving tough business problems hosted on Hackerearth by FactorBranded Data Warriors with the [Power of fastai librarv]

Problem statement background:

High-value shipments such as mobile phones, watches, jewellery are usually at risk of sellers shipping a counterfeit product to the customer, or conversely, customers returning a counterfeit product back to the retailer, while keeping the original item for themselves. High-value shipments are hence typically scanned by an x-ray machine during different stages of transit wherein, and an operator manually checks each x-ray image to ensure that the item in the box matches its product description.

Goal:

To automate the solution with computer vision

Magnitude of the dataset given:

100,000 product-names with their descriptions are given (Anything ranging from watches to iPhones to house hold lamps). ~110,000 images are given with their product descriptions and if their x-rays match (**True** -> X-Ray match the product description and **False** -> X-Ray donot match the product description) .

A lot of images and data . lol

Breaking down to small actionable modules to solve this huge problem:

1. Build a **text classifier** to classify product descriptions to their **"Sub-Category" .** Example: Moto-G is a mobile , Saree is women's clothing etc [Note: Tried **Category** based classification , got about 30%-40% accuracy, which was not very optimal]
2. Build an **image classifier** to check if the x-ray match their **Sub-Category**

EDA with Text classifier to predict sub-category(Code):

1. Approximtely 8% of the given train dataset is classified as 'False'
2. Most of the shipments are 'mobiles', around 60,000 mobiles followed by 42,000 clothes (Men and women combined)
3. 12k products are categorised as 'uncategorized'
4. Build an SGD based text classifier to fit the product descriptions with their sub-categories
5. Join product_description.csv (provided as a part of the training dataset) with product description of train data
6. Replace NaN (implying, product description was not in the file given to us) with our Text classifier prediction sub-categories.
7. Remove the negative rows (Target variable = False) and keep it as a cross-validation set
8. Use positive rows to build an image classifier.

Understanding the messiness of the real-life data:

Same products are written in different ways by different sellers and hence building the text classifier is important.

Products and sub-categories are not hand-labelled. I'm guessing a machine learning solution was written to predict category/sub-category based on product-title.

Books are being classified as 'case, cover and screen-guards', lets check the sub-categories of books:

```
for i,row in df.iterrows():
    if 'books' in row['product_description'].lower():
        print(row['sub_category'])

Case, Cover and Screenguards
Books
Case, Cover and Screenguards
Case, Cover and Screenguards
Books
nan
nan
Books
Books
Home Decor and Furnishing
Speakers
Everyday Makeup
Books
Books
Speakers
Men's Clothing
Home Decor and Furnishing
nan
nan
Men's Clothing
```

The underlying machine learning solution is not good enough to accurately predict the sub-categories. We have to consider this to make our model **better**.

Evaluation metric is **F1 score** which is unforgiving, we need to have both good precision and recall to get a good score.

Use our text classifier(which gives 95% accuracy over train dataset) over the test dataset to get "Sub-category" based on the product description.

Note: 95% accuracy is not over a cross-validation set but over the actual training set that was fitted with SGD-based text classifier.

Building an image classifier for Sub-Category (Code)

Used Densenet201 architecture (transfer learning, trained on Imagenet), to get a pretty decent model of about 83% accuracy. Considering, x-ray aren't usually the images classified, this is pretty good.

Had to tune hyperparameters to get the accuracy of about 83% including data augmentation (randomly fliping images), changing dropouts, the size of the image being fed, etc.

Finding fraudulent shipments:

Predict each image on the train dataset to find their sub-category. If the predictions from our text-classifier do not match the predictions from image-classifier, the product maybe fraudulent.

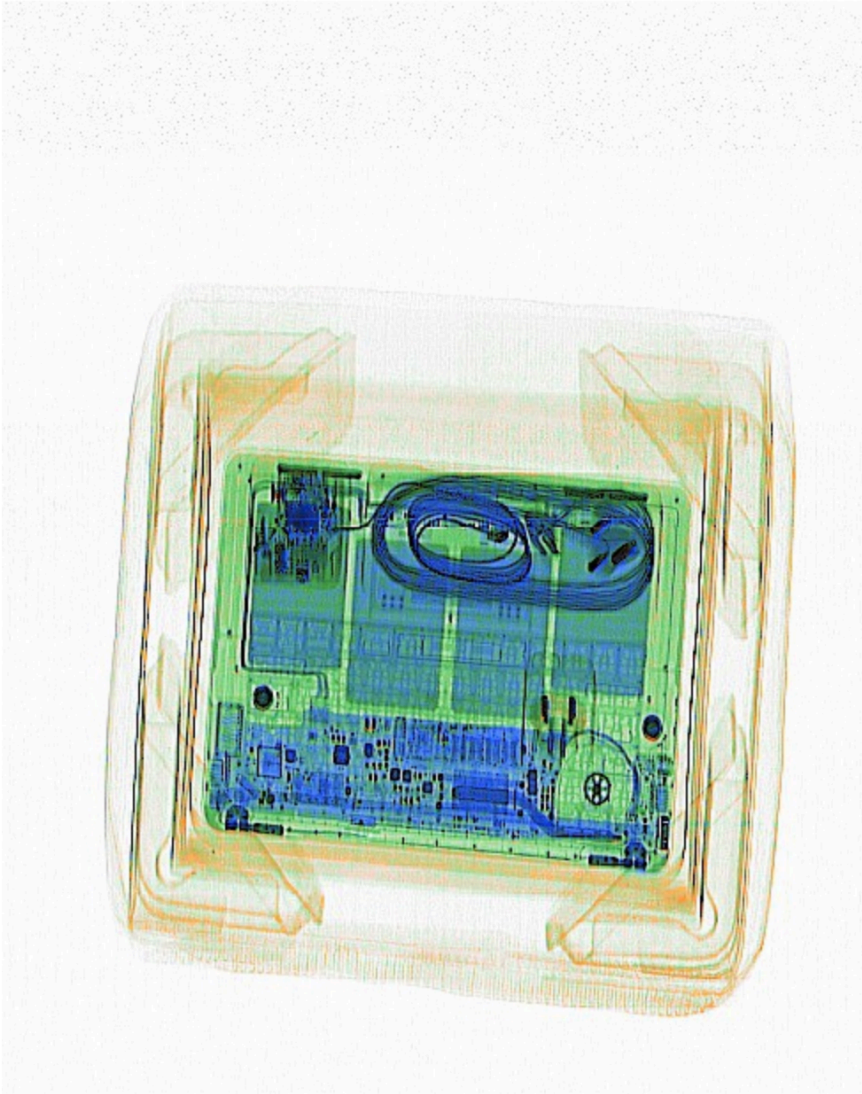
Spending time with the data:

A deeper EDA into the training dataset revealed some interesting insights.

~9000 shipments were false. Some of the product_descriptions classified as false with their numbers,

```
iPhone: 918
redmi:816
galaxy:986
notebook:921
note:868
macbook:88
watch:598
```

Lets take a macbook pro true X-Ray image:



Macbook pro clearly has a different kind of packing in comparison with other laptops, the same with iPhones, galaxy etc

Instead of building classifier over 70 sub-categories, extend the model to classify iPhones,Galaxy,Macbooks etc separately and instead of classifying mobiles all together, classify separately based on the manufacturer.

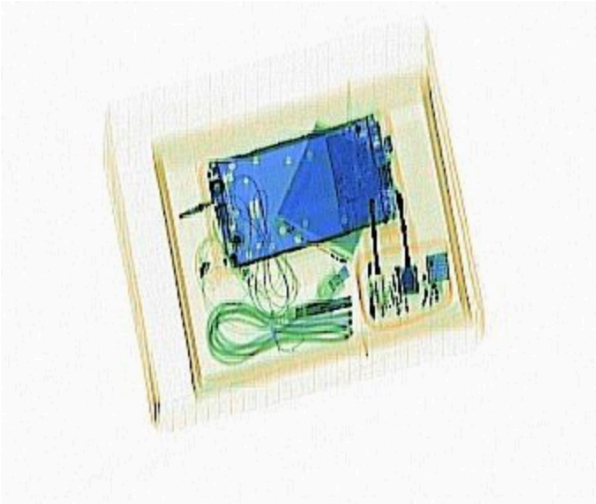
The final best performing model (Since our metric is F1, We only make “false” predictions only when we are absolutely confident that the fraudulent x-ray images are actually counterfeit):

```
pred = learn.predict_array(image[None])
    try:
        our_pred = row['predictions']
        actual_pred = data.classes[np.argmax(pred[0,:])]
        if our_pred != actual_pred:
            if our_pred == 'mobiles' or our_pred == 'laptops' or
our_pred == 'watches' or our_pred == "women's clothing" or our_pred
== 'iphone' or our_pred == 'redmi' or our_pred == 'galaxy' or
our_pred == 'note' or our_pred == 'notebook' or our_pred=='dslr' or
our_pred=='macbook' or our_pred=='saree':
                print(str(i)+" "+our_pred+" "+actual_pred)
                df_test_org.loc[i,'product_description']=False
            else:
                df_test_org.loc[i,'product_description']=True
            if True:
                index_of=data.classes.index(our_pred)
                if np.exp(pred[0,index_of])>0.1:
                    df_test_org.loc[i,'product_description']=True
```

Insights and going production level:

Applying to Train dataset false images: Applying this model over the false data given predicts ~8000 (CSV file of the predictions along with our image classifier prediction) out of ~9000 fraudulent shipments. What about the other ~1000?

One possible explanation is, the products aren't actually counterfeit but the operator had to make sure they weren't. Our model predicts with 97% accuracy that this is an mobile but is actually a fraudulent shipment. Might have to know why such images are marked as "False"



Going production level: Instead of broadly classifying as Mobiles (Since much of the counterfeit occurs in mobiles), build an image classifier based on manufacturer (or cluster them based on their similarity of the product). Eg: Redmi, Xiomi, Galaxy might have similar patterns on a X-ray and we can cluster them together.

We do pretty well with watches/jewellery/clothes etc with the model above and training with much larger dataset should suffice.

Build a better image classifier: Whilst 83% is decent enough to get good enough score, this will not be optimal under production settings and we have to classify better. One thing might be : To weed out wrong images being fed to the classifier.

Would appreciate any thoughts on improving the above automated solution.

If interested, do take a look at Part 1 of the contest (Classifying T-Shirts by myntra)