1. Problem statement:
Given an image of a promotional pamphlet, extract information about such items on sale and their selling price.

2. About the data:
One single image of size 3056×1538 containing 10 pamphlets merged together.

3. Primary idea:
Divide and Conquer approach: Divide the main image into multiple sub-images of pamphlets then detect the offer/sale boxes as they are very similar to each other, after that run any pre-trained OCR model to detect the text.

4. Data preprocessing:

a) Dividing into separate sub-images:
As all the pamphlets were adequately aligned, it was very straightforward to just crop the image in a specific manner using OpenCV. Each image was roughly sized around 650×750.

b) Experiment with traditional CV:
The goal was to detect all the offers from the images with bounding boxes but soon noticed that each offer contains a different price and different name also no two offer boxes are shaped similarly to each other. Traditional computer vision approaches failed to incorporate these varieties of data resulting in failure to detect offers from the images.

c) Training YOLOv5 with custom dataset:
This was the last resort as any state-of-the-art deep-learning model needs a significant amount of data to perform the task fairly well and the data in hand was not sufficient enough to train the model.
Creating the dataset: Used image augmentation to increase the data points. Horizontal flip and Gaussian Noise was two of the augmentation technique used on the images. Images were labeled and fed through the YOLOv5 network with pre-trained weights. YOLOv5 is the best model available today to detect objects from images. This performed fairly well with the test data.
The output was in bounding boxes so the offer boxes were cropped from the images with bounding box information and cropped offer images were stored for the next stage.

d) Upscaling the images:
Although the pamphlets were fair quality the cropped offer boxes were very poor in quality so much so that it's even hard for a human to interpret the text information. Used OpenCV interface for implementing Super Resolution (SR) based on a deep learning model EDSR. But this resulted in a smoothed image which is sharpened by a convolution sharpen operation.

5. Experiment with OCR models:

a) Pytesseract:

This LSTM based OCR engine performed most poorly among the other two, which was expected given the low-quality images also this OCR engine performs fairly well with the binarized data as well as less noisy clean data making it one of the fastest among the three.

b) Easy-OCR:

This ResNet + LSTM based OCR model also works very well with clean high-res images but with these images easy-ocr performed similarly to the keras-ocr but slightly less accurate than keras-ocr.

c) Keras-OCR:

This is a slightly polished and packaged version of the Keras CRNN implementation and the published CRAFT text detection model. This worked surprisingly better than the other two models with the given images. But still many small almost no-visible texts were not detected.

d) About the output:

The final output file: output.csv consists of item names that are currently on sale and their prices respectively. Although the price of the items was detected fairly well the name of the items weren't detected accurately due to the fact that the images were very low in quality. The name of the items needed to be processed separately for each item because the names were very different from one another and the structure of the paragraph and line breaks also in different places.

6. Conclusion:

Given the poor image quality and the limited size of the available data, it is evident that the OCR models performed fairly well. But this can easily be improved with quality data as well as a larger quantity of data. Then it is possible to train a custom OCR model based on the requirement that will perform significantly better than these pre-trained models.