# DeePaperScan
A Deep Learning model that detects paper corners in pictures / videos

##Definition
### Project overview

Mobile Scanners often use computer vision(ex. OpenCv) to detect corners of scanned paper images in their smartphone apps. The main reasons are that the solution works well on basic cases and secondly it is easily deployable for real time usage (doesn't require a lot of hardware compute).

My main motivation for this project is to check if training a Deep Neural Network on a labeled dataset of image corners can showcase good results in real time scenario by detecting image corners

###Problem Statement

Image scanner that detects Paper/document corners in Image / video using a Convolution Neural Networks model

There are 4 corners positions to detect of a paper document (Top left, Top right, Bottom left, Bottom right) in different backgrounds.

Some of the most popular similar usecases available in Open Source community are Facial/hand Keypoint Detection, Human pose estimation ...

In Machine Learning terms this is a Regression task that is also known as keypoints / landmark detection. The model takes a resized Gray image with size (197,350,1) as input and produces the expected output that is the x,y coordinates of the 4 corners of a document.

###Metrics

Evaluation metrics that will be used in the training / validation sets :

- loss : mean squared logarithmic loss error

##Analysis
###Data Exploration
The dataset comes from the SMARTDOC 2015 competition (Smartphone Document Capture and OCR Competition)
(link)[https://sites.google.com/site/icdar15smartdoc/challenge-1/challenge1description] [1]

The dataset when downloaded, has the following structure:

```
smartdoc-dataset-set/
 ├── background01
 │   ├── datasheet001.avi
 │   ├── datasheet002.avi
 │   ├── datasheet003.avi
 │   ├── datasheet004.avi
 │   ├── datasheet005.avi
 │   ├── letter001.avi
 │   ├── letter002.avi
 │    ├── letter003.avi
 │    ├── letter004.avi
 │    ├── letter005.avi
 │    ├── magazine001.avi
 │    ├── magazine002.avi
 │    ├── magazine003.avi
 │    ├── magazine004.avi
 │    ├── magazine005.avi
 │    ├── paper001.avi
 │    ├── paper002.avi
 │    ├── paper003.avi
 │    ├── paper004.avi
 │    ├── paper005.avi
 │    ├── patent001.avi
 │    ├── patent002.avi
 │    ├── patent003.avi
 │    ├── patent004.avi
 │    ├── patent005.avi
 │    ├── tax001.avi
 │    ├── tax002.avi
 │    ├── tax003.avi
 │    ├── tax004.avi
 │    └── tax005.avi
 ├── background02
 │    ├── datasheet001.avi
 │   ...
 │    └── tax005.avi
 ├── background03
 │   ...
 ├── background04
 │   ...
 └── background05
    ...
     └── tax005.avi
```

It contains 5 directories(image backgrounds), and 150 video files -30 for each background- (AVI container, XVID codec, no audio), where image frames are taken from different angles.

The videos resolution 1920x1080

The output is given as xml file:

```
smartdoc-dataset-set/
├── background01
│    ├── datasheet001.gt.xml
│    ├── datasheet002.gt.xml
│    ├── datasheet003.gt.xml
│    ├── datasheet004.gt.xml
│    ├── datasheet005.gt.xml
│    ├── letter001.gt.xml
│    ├── letter002.gt.xml
│    ├── letter003.gt.xml
│     ├── letter004.gt.xml
│     ├── letter005.gt.xml
│     ├── magazine001.gt.xml
│     ├── magazine002.gt.xml
│     ├── magazine003.gt.xml
│     ├── magazine004.gt.xml
│     ├── magazine005.gt.xml
│     ├── paper001.gt.xml
│     ├── paper002.gt.xml
│     ├── paper003.gt.xml
│     ├── paper004.gt.xml
│     ├── paper005.gt.xml
│     ├── patent001.gt.xml
│     ├── patent002.gt.xml
│     ├── patent003.gt.xml
│     ├── patent004.gt.xml
│     ├── patent005.gt.xml
│     ├── tax001.gt.xml
│     ├── tax002.gt.xml
│     ├── tax003.gt.xml
│     ├── tax004.gt.xml
│     └── tax005.gt.xml
├── background02
│     ├── datasheet001.gt.xml
│    ...
│     └── tax005.gt.xml
├── background03
│    ...
├── background04
│    ...
└── background05
    ...
     └── tax005.gt.xml
```

Each Xml file contains labeled data of each frame of the corresponding video:

One frame block contains this form :

```
   <frame index="$frame_index" rejected="false">
     <point name="bl" x="$blx" y="bly"/>
     <point name="tl" x="$tlx" y="tly"/>
     <point name="tr" x="$trx" y="try"/>
     <point name="br" x="$brx" y="bry"/>
   </frame>
```

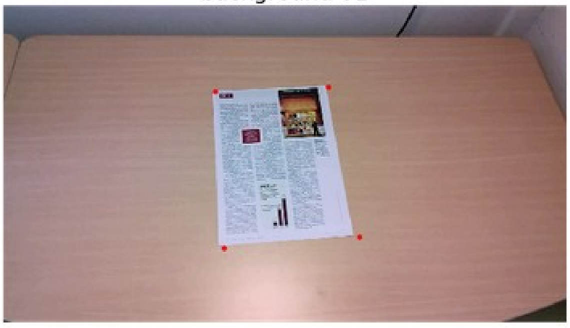
where:
- `$frame_index` is the index of the frame, starting at 1.
- `$blx` and `$bly` are the coordinates of the bottom left point of the object
- `$tlx` and `$tly` are the coordinates of the top   left point of the object
- `$trx` and `$try` are the coordinates of the top   right point of the object

- `$brx` and `$bry` are the coordinates of the bottom right point of the object

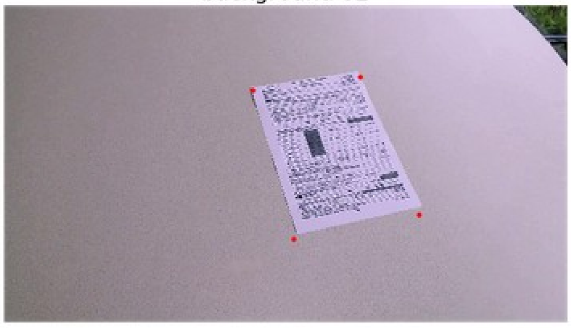
###Exploratory Visualization

Example of frames (taken from videos) different backgrounds and ground truth keypoints (in red)

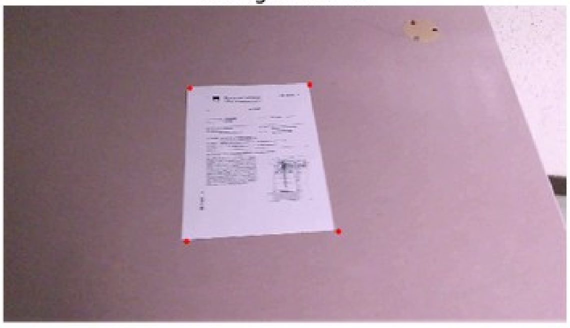
background 01



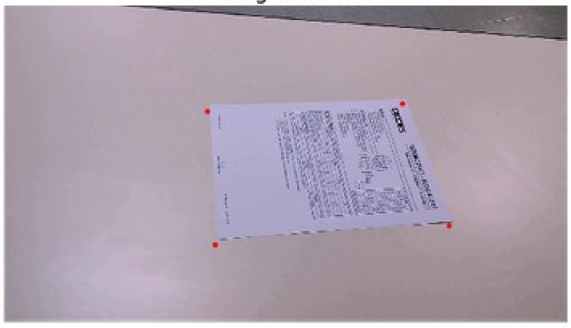
background 02



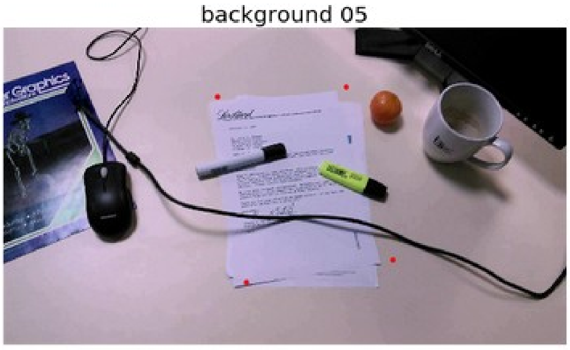
background 03



background 04



background 05



The keypoints as shown in the image above are correctly positioned but not perfectly (ex. Background 5)

The Videos’ frame size : 1920x1080

I discuss later the methodology used to extract videos to images

###Algorithms and Techniques

- libraries used: Keras (with Tensorflow Backend)

The algorithm we will be using is going to extract important features from images, and try to predict image corner coordinates

- The architecture of the CNN model that we’ll be using :

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 195, 348, 32)      320
_________________________________________________________________
activation_5 (Activation)    (None, 195, 348, 32)      0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 97, 174, 32)       0
_________________________________________________________________
dropout_5 (Dropout)          (None, 97, 174, 32)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 96, 173, 64)       8256
_________________________________________________________________
activation_6 (Activation)    (None, 96, 173, 64)       0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 48, 86, 64)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 48, 86, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 47, 85, 128)       32896
_________________________________________________________________
activation_7 (Activation)    (None, 47, 85, 128)       0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 23, 42, 128)       0
_________________________________________________________________
dropout_7 (Dropout)          (None, 23, 42, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 123648)            0
_________________________________________________________________
dense_3 (Dense)              (None, 500)               61824500
_________________________________________________________________
activation_8 (Activation)    (None, 500)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_4 (Dense)              (None, 500)               250500
_________________________________________________________________
activation_9 (Activation)    (None, 500)               0
_________________________________________________________________
dropout_9 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_5 (Dense)              (None, 8)                 4008
=================================================================
Total params: 62,120,480
Trainable params: 62,120,480
Non-trainable params: 0
```

###Benchmark

As The challenge had been canceled, there are no benchmark results for this dataset found on internet.

I will be training the model for 500 epochs to check. If loss is still dropping or not to continue training.

The first step is to test on images that come from the same dataset

The second step is to test on images taken from my smartphone (new image with different background, angle, lighting, resolution ...)

##Methodology
###Data Pre-processing
After downloading the dataset it needs to be pre-processed, here are the main steps to create the train and validation data

1. I need to parse all xml files and store them in One dataframe that contains filenames and corners coordinates. The script:

```
DF_l = []
for background_tup in backgrounds:
    print(background_tup)
    background = background_tup[0]
    bg = background_tup[1]
    
    xmls = [el for el in os.listdir(background) if ".xml" in el]
    for xml in xmls:
        xtree = et.parse(os.path.join(background,xml))
        xroot = xtree.getroot()
        xs, ys, idxs, names = [], [], [], []
        for node in xroot.iter('point'):
            names.append(node.attrib["name"])
            xs.append(node.attrib["x"])
            ys.append(node.attrib["y"])
        for node in xroot.iter('frame'):
            idxs.append(node.attrib["index"])
        # idxs = [x for item in idxs for x in repeat(item, 4)]
        idxs = [bg+"-"+xml.split(".gt")[0]+"-f"+idx+".jpg" for idx in idxs]

        temp_df = pd.DataFrame(list(zip(names,xs,ys)),columns=["names","xs","ys"])
        blx = temp_df[temp_df["names"]=="bl"]["xs"]
        bly = temp_df[temp_df["names"]=="bl"]["ys"]
        tlx = temp_df[temp_df["names"]=="tl"]["xs"]
        tly = temp_df[temp_df["names"]=="tl"]["ys"]
        
        brx = temp_df[temp_df["names"]=="br"]["xs"]
        bry = temp_df[temp_df["names"]=="br"]["ys"]
        trx = temp_df[temp_df["names"]=="tr"]["xs"]
        Try = temp_df[temp_df["names"]=="tr"]["ys"]
        df = pd.DataFrame(list(zip(idxs,blx,bly,tlx,tly,brx,bry,trx,Try)),
                    columns=["filenames","BLx","BLy","TLx","TLy","BRx","BRy","TRx","TRy"])
        DF_l.append(df)
```

The dataframe contains 24889 images (frames)

2. I need to convert videos to frames and save them to images using this script:

```
# save all videos to frames
for background_tup in backgrounds[:]:
    print(background_tup)
    background = background_tup[0]
    bg = background_tup[1]

    videos = [el for el in os.listdir(background) if ".avi" in el]
    for video in videos:
        im_path = os.path.join(background,video)

        vidcap = cv2.VideoCapture(im_path)
        length = int(vidcap.get(cv2.CAP_PROP_FRAME_COUNT))
        print( length )

        success,image = vidcap.read()
        count = 0
        while success:
            output_file = "{}-{}-f{}.jpg".format(bg, video.replace(".avi",""), count)
            cv2.imwrite(os.path.join(output_path,output_file), image)     # save frame as JPEG file
            success,image = vidcap.read()
#           print('Read a new frame: ', success)
            count += 1
```

3. Create a random sample dataframe. As video frames looks each other, Doing a random sampling of sub 10k images seems to be enough for training

```
sample_df = pd.concat(DF_l).sample(7000)
```

4. Split to train/validation

The validation set is 0.01 % of the sample data, I didn't want to shrink the size of the sample used for training

The script splits to train/validation and create corresponding image folders :

```
train, val = train_test_split(clean_df, test_size=0.01)
for sample in [[train,train_path],[val,validation_path]]:
    filenames = sample[0]["filenames"].values.tolist()
    for filename in filenames:
        shutil.copyfile(os.path.join(output_path,filename),os.path.join(sample[1],filename))
```

5. Prepare the input image

Image resizing including landmarks doesn’t come default in Keras. I used a function from this link https://github.com/fizyr/keras-retinanet/blob/8f7e7414db193832572773ea503c26edab405b96/keras_retinanet/utils/image.py that resizes image and returns the scale that we apply to landmarks

The processed images final shape is (197,350,1)

###Implementation

1. Prepare model

The model is wrapped in one function. I had to specify :
- The Height and Width of the image in the first convolution layer that is 197, 350
- The output size in the last dense layer that is 8
- For the model compilation, I used the RMSprop as an optimizer and mean_squared_logarithmic_error as loss metric

2. Training

I had to lower the batch size to 64 as it doesn’t fit the Gpu memory
Training 1000 epochs took around 2.7 hours on GPU

###Refinement

Future works to test/optimize the solution (In order) :

- Data augmentation

Data augmentation seems to be an important step to make a more intelligent / robust model that is going to tackle images orientation (real life senario)

Keras image generation is easy step but only for image classification. For keypoint detection need to prepare a custom function that is going to scale for each generated image keypoints.

Data generation means more images and thus more training time. Worth it if results get better

- Box detection + keypoint detection

All tutorials concerning the facial Keypoint Detection use a kind of object detection to detect the face and crop it, this seems pretty effective to remove noise and focus only on the face. Thus it would minimise considerably the loss when training for keypoint detection.

- Testing other CNN architectures

- Optimization / GridSearch

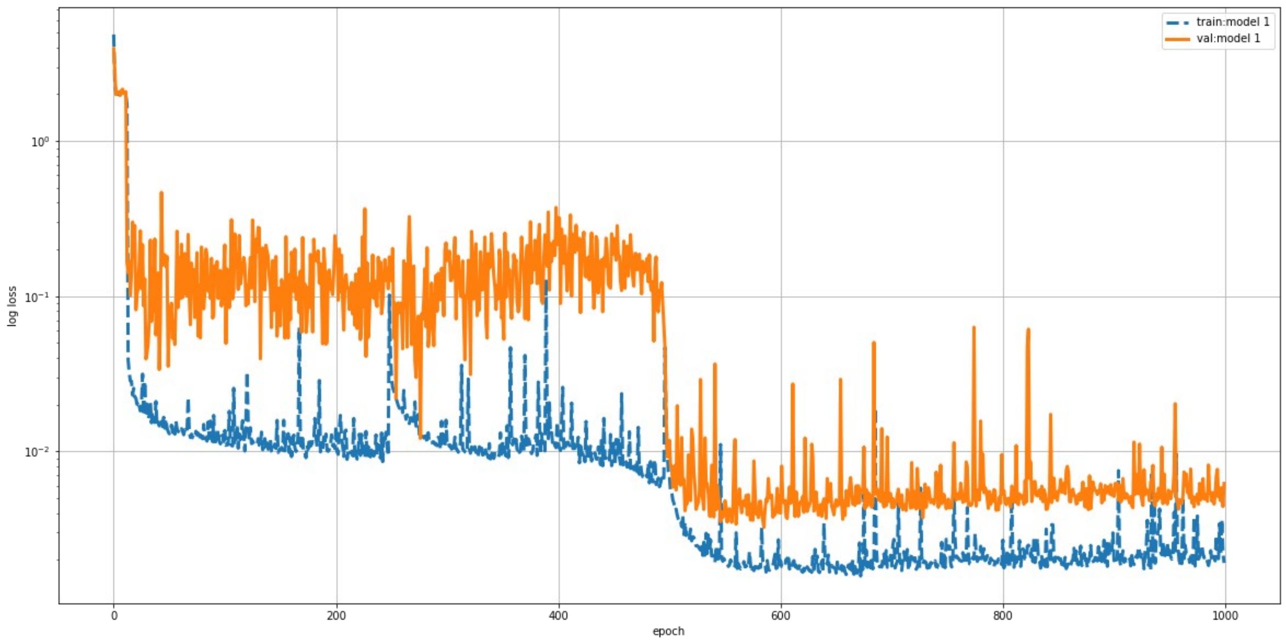
- Specific model

One other method that may work is by creating 1 model for 1 key-point, it means for our use case we need to create 4 different models. However there is a risk of overlapping at the same time, for flipped document, are models trained enough to detect if text if flipped or not (BL instead of TL and vice-versa). This mainly depends on the quality of the dataset.

##Results

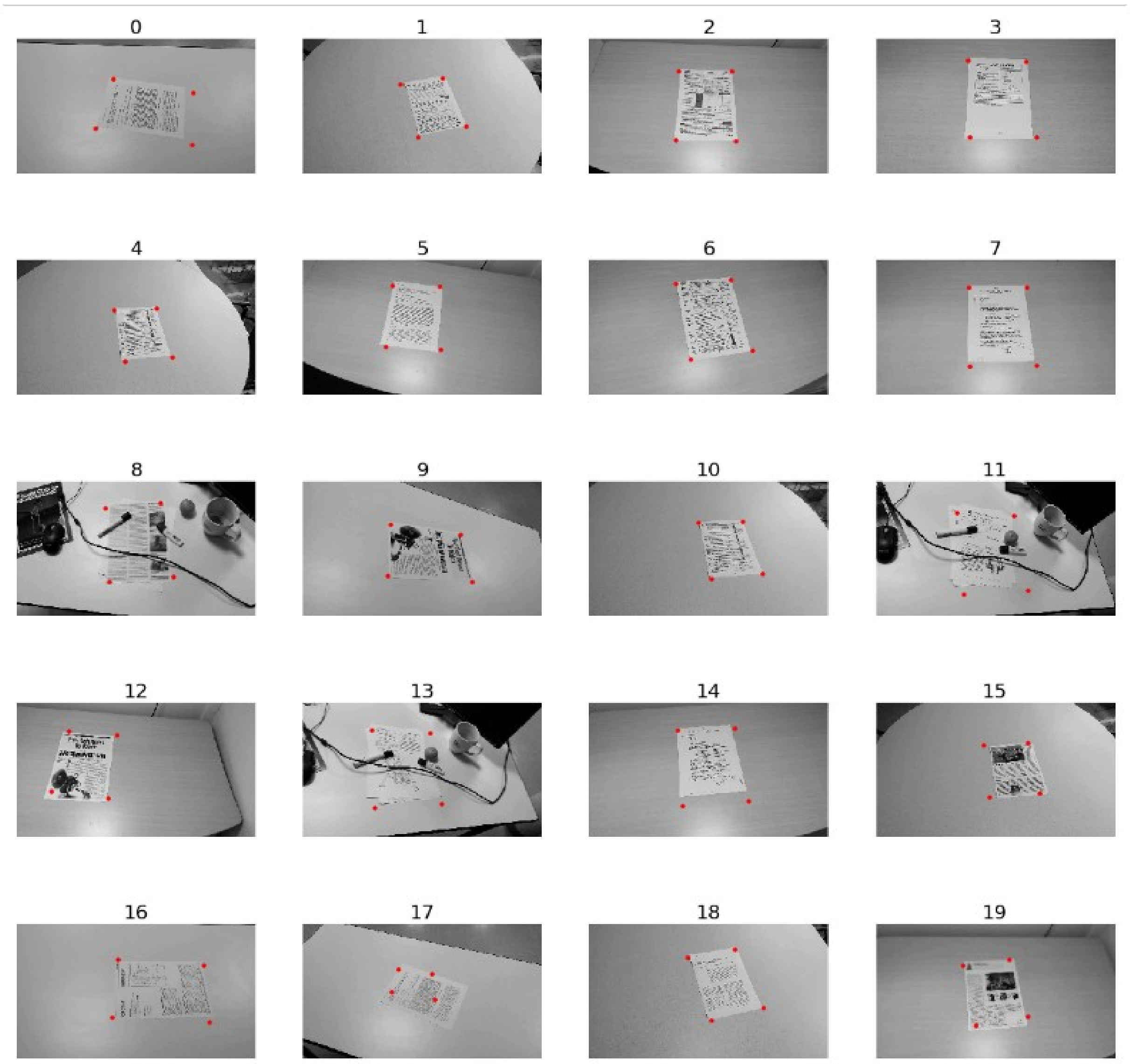
###Model Evaluation and Validation

After 1000 epochs of training, the loss seem to stabilize after 600 epoch. A lot of fluctuation in the validation set, mainly due to the small size of the sample. The train loss usually is smaller that the validation loss, which is coherent.

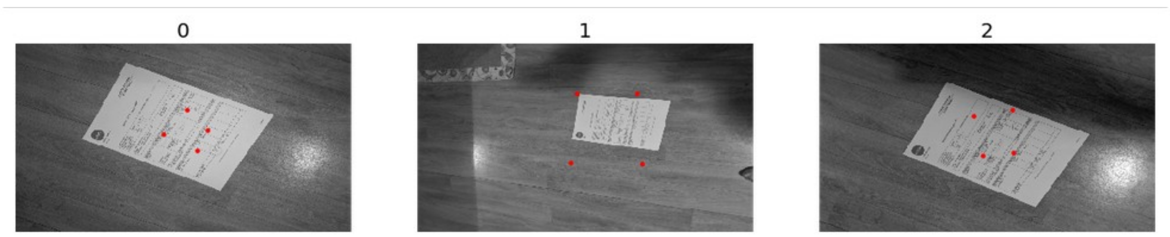


###Justification

Below set of images tested on validation data:



Below resized images taken from smartphone to test the solution:



The final results showcase that the model is specific and not generalized well on non seen images or backgrounds.

This model can work well in similar environment(when optimized) where the pictures have been taken, but would be difficult to apply it to different images.

This model do not solve the recent problem, Further work is needed to optimize the model, and/or to build more generalized dataset (different type of cameras, lighting, backgrounds).