**Google Summer of Code:**

1. **Document Type Classification**

Trained a Convolutional Neural Network based on the InceptionV2 Architecture. The dataset that I used is RVL-CDIP. <https://www.cs.cmu.edu/~aharley/rvl-cdip/>

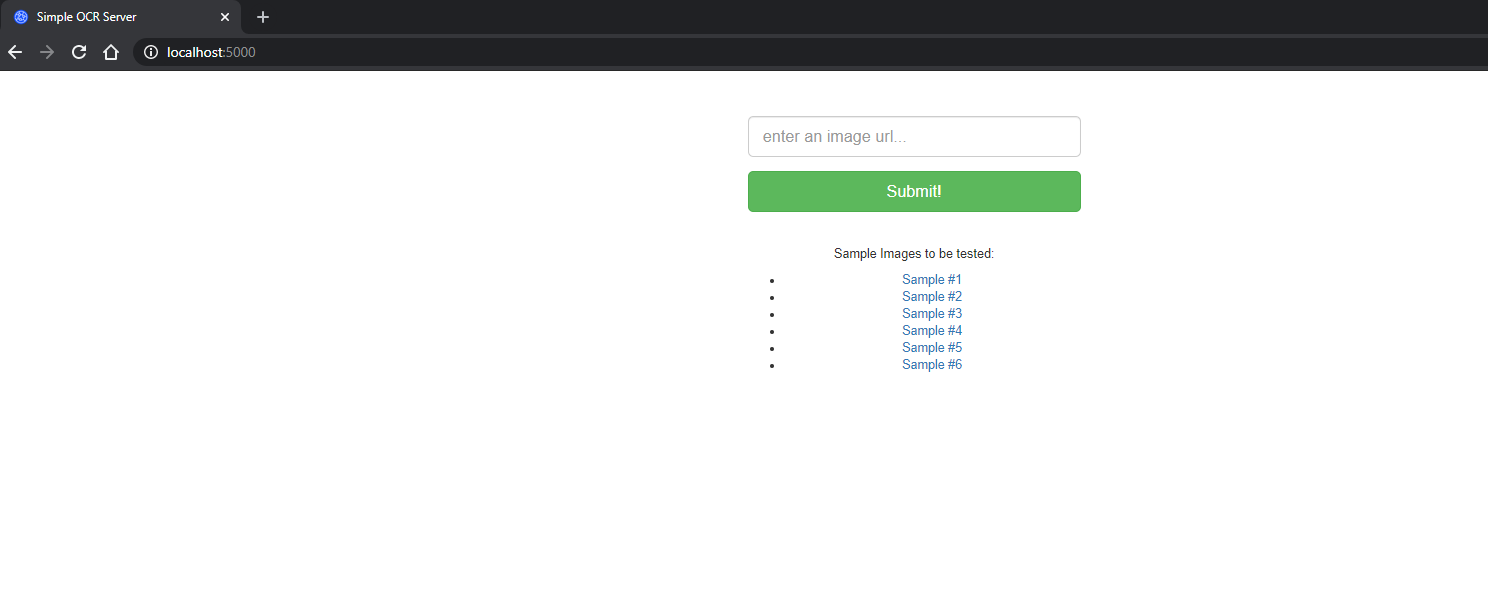
The dataset is of size 37 GB. For a small and quick proof of concept, I have used about only 5GB of the data (2.5k images per class instead of the original 25k images per class) with 70/10/20 training, validation and testing split, trained for 100 epochs and achieved 65% accuracy.

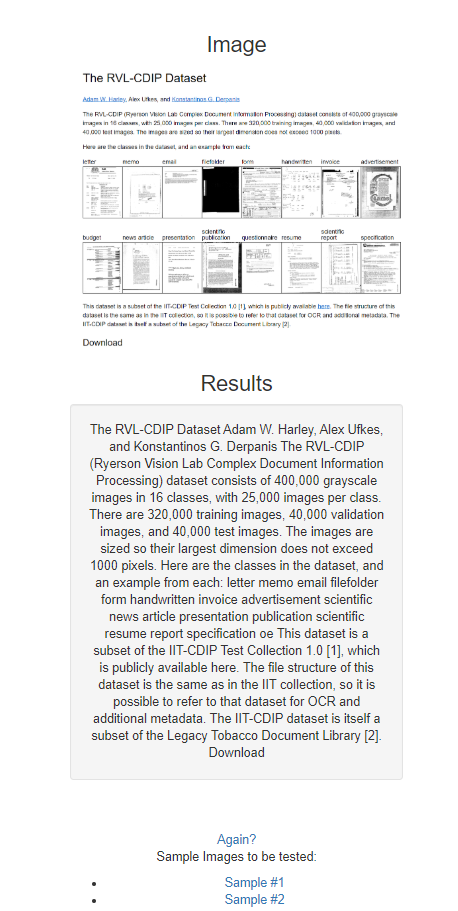
loss: 1.1834 - acc: 0.6500 - categorical\_crossentropy: 1.1834 - val\_loss: 1.2591 - val\_acc: 0.6075 - val\_categorical\_crossentropy: 1.2591

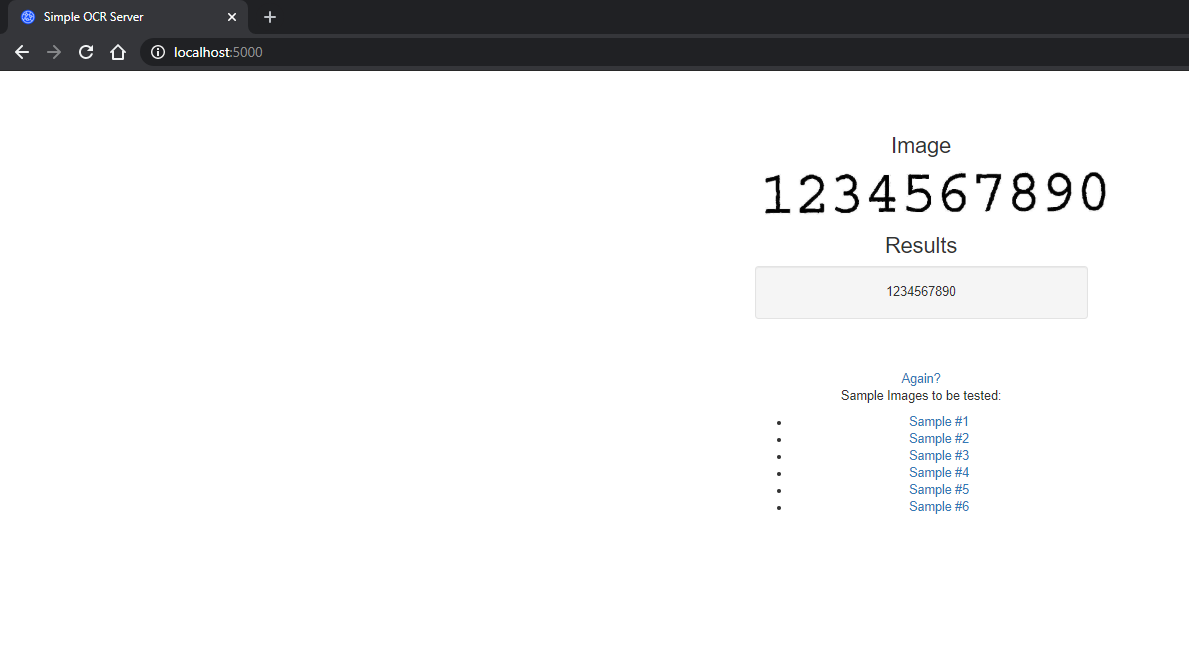
Training took 25 hours on Intel i3 CPU.

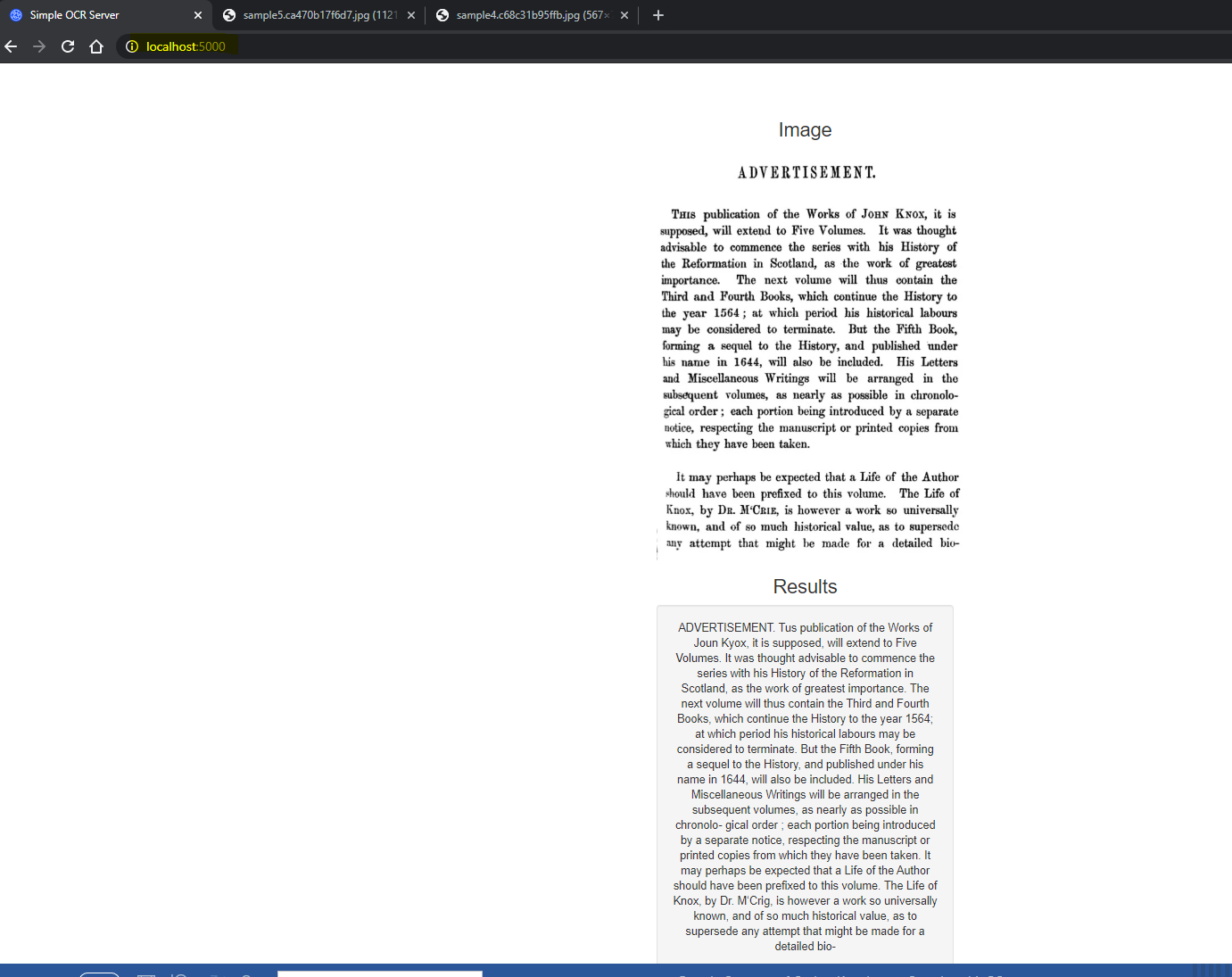
1. **Proof of Concept.**

While training, I wanted to start on the front end, and the flask server. Below is a very very small Proof of Concept that is powered by the Python’s OCR module pytesseract. The goal is to make the PoC modular so that in the future, an alternative machine learning model that performs the same OCR task as the pytesseract can be easily replaced. Below are some results of the working PoC when I upload some random image urls and does OCR on it to extract text. Right now, the app doesn’t know the difference between different document types(memo, email, resume). If I can use my step 1 progress and determine the document type before the OCR step, I can get better results on my OCR by specifically focusing on the keywords that are more likely to exist in that document type. I call this context aware OCR. In the future by doing context aware OCR vs native OCR, I plan to get better results. For now, below are some screenshots of the native OCR engine powered by pytesseract module.









I am using **HTML for front end, Flask for back end and the tesseract module for the OCR**. There is no database, since the server is only accepting an image URL to open and do OCR on. The server is not storing the data. The app is **dockerized** on an ubuntu based image to make the deployment easier for later.

1. **Gluing pieces together**

I have the following document class labels in my rvl-cdip dataset.

1. letter
2. form
3. email
4. handwritten
5. advertisement
6. scientific report
7. scientific publication
8. specification
9. file folder
10. news article
11. budget
12. invoice
13. presentation
14. questionnaire
15. resume
16. memo

The goal is to create a pipeline like the following:

