https://github.com/Hungry-munk/CRNN-for-OCR-

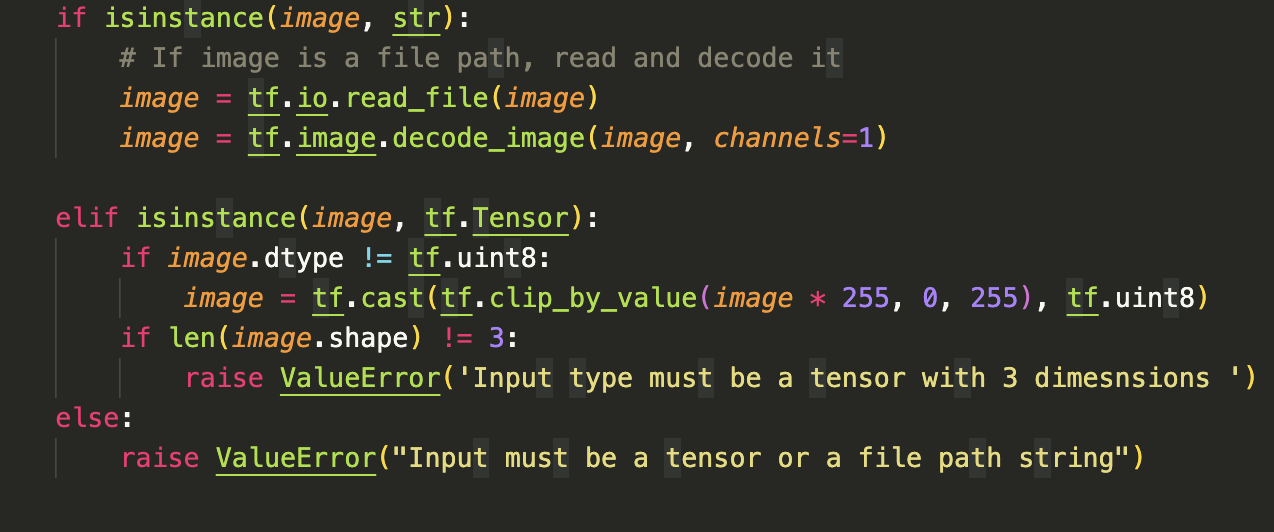
**C06 – Programming skills**

**Validation techniques:**

Various validation techniques were used throughout the code base to ensure correct data types and structures are being used passed into functions and systems.

An example is:

The code below was used in the **random\_pad** function. The validation ensures that the image tensor that was passed in is of the correct data type (uint8) and if it isn’t it will convert it. It will also ensure that the tensor is 3 dimensional (for height, width and channels) and if not throw and error. Finally, if the image is if the data type doesn’t match, as it needs to be either a string with the data path to the image or image tensor.



**Critical and creative thinking (design decisions):**

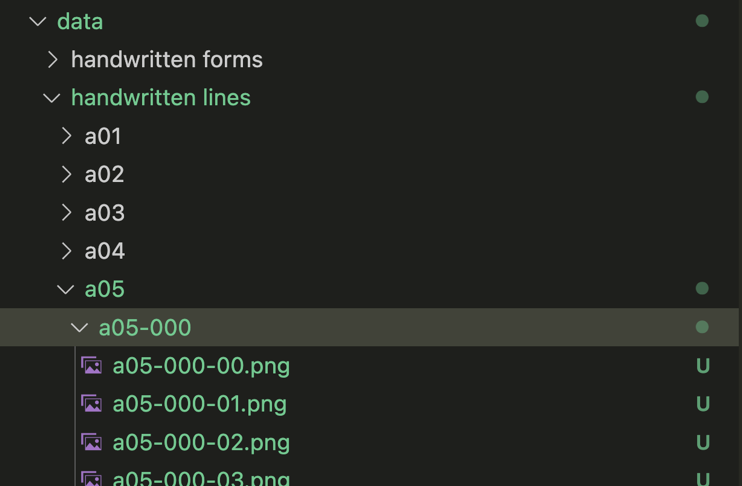
Throughout the entire coding process numerous coding decisions and architectural decisions had to be made after realising that certain things either weren’t possible or didn’t make sense to do.

Some of these solutions to these problems are:

1. **The File iteration problem**

My data set for training the OCR AI engine was the opensource IAM handwriting database. In order to manipulate the image files and get them ready by making TensorFlow tensors.

The challenge was assembling the right string to decode the image file into a tensor, but the names of the files were not consistent alongside the fact that their numerous subfolders that needed accessing also not named consistently.



So, I couldn’t dynamically and programmatically create strings to get the image files paths. The way I resolved this problem was learning that using certain libraries I could iterate through retrieve a string representing for the subfolder in the main data folder. The library I chose to do this was pathlib (there were other alternatives like glob and os but pathlib had the syntax I understood the easiest)

The way I resolved the problem was to iterate through the XML file folder that contained the text that was in its corresponding photo but also the file name.  
using the file names I was able to create a string to the exact location of all the images needed for training.

1. **The image shape and network architecture problem**

When it came to preprocess images to make them compatible for training with a TensorFlow Keras CRNN (convolutional recurrent neural network) I initially thought the images could keep their current dimensions and be fed into the model directly. Upon further research I realised that with my limited resources (minimal data, understanding of models and computing resources) I couldn’t use images of any size, they either had to be all be the same size or at least all be of the same height, furthermore they all should be grayscale (to save computing resources). After much deliberation I decided to choose the harder and more effective option of having them be the same height but not width and to ensure that the images contained the same correct information I made to resize them such that they had the same aspect ratio as before to not lose spatial information. To make images all grayscale all images were decoded directly into 1 (grayscale) channel instead of the expected 3 (3 channels for R, G, B) when they first being converted into an image tensor from the string file path.

1. **The dataset creation problem MAKE SURE TO WRITE ABOUT**

**Testing techniques:**

The software functions and processes were thoroughly tested some these tests were:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What is being tested | How is it being tested | Expected result | Actual result | Comments or solution |
| Bounding box updater | Running image training data into image cropping function and the images inspected as to whether all the text that should be on the image is on the image | 2 images having been cropped from the original containing required text | 2 cropped images strangely cropped not containing the right text | Initially the functions didn’t output correctly cropped images but after iterative bug fixing the function is now effective |
| Image resizing | Training Images run through the preprocessing function and the shape of the images printed out. | All images are the same height and have the same aspect | All images have the same height and appropriate aspect ratio’s | The function was pretty easy so there were no logic errors in the code but got a few errors from misusing the TensorFlow library. |
| Feature map to sequences | A featured map being fed into the f\_map\_to\_seq function and the shape of the sequence printed | Sequence length that is proportional to the width of the image times the chahnnels of the image (an image that has gone through all the convolution layers) | Fixed sequence length | I had to write code to calculate how the CNN layers convolutions effect the length of the width |
| Split complete data into training, cross validation and testing sets | Arrays are split using array splitting techniques and the shape printed to verify correct splitting | 3 arrays that are such that the training set is 75% of the total data and the other are 12.5% | 3 arrays that are such that the training set is 75% of the total data and the other are 12.5% | Other than some basic syntax errors their were no other problems |
| Batch data shape normalisation | Trying to train engine, if the engine has parameters altered then data batches are valid | For training to occur | Type error | Padding a batch was difficult as it required some more thinking into how normalisation was to be carried out alongside advanced tensor manipulation techniques |
| Valid data batch for training | Try ‘yield’ing data from a function to create custom tf.data.Dataset training data for training by feeding valid arguments and data into data\_generator and lmbda | A functional tf.data.Dataset | Custom error, I was thrown telling to entering valid values for the data set to standard size | Fixed my data\_genrator algorithm such that I was getting correct values to create calid data sets of required tf.data.Dataset data structure. |

**C07 – File Management**

**Data structure Organisation:**

The primary data structures in the code include an SQLite database, used for storing user data, and external image and XML data, which are processed to prepare training data and labels for the OCR AI engine.

**SQLite Database:**

The SQLite database is a core component, manipulated through the Flask Python backend API. Operations include fetching, adding, and deleting data. For instance, when users attempt to log in or access previous chats, the database is queried to retrieve both images and the corresponding OCR-extracted text from those chats. New data is added to the database when users create new chats, upload images, or obtain OCR-generated text. Similarly, data is deleted whenever users remove their chats. This systematic approach ensures that the backend efficiently handles image storage and OCR data retrieval.

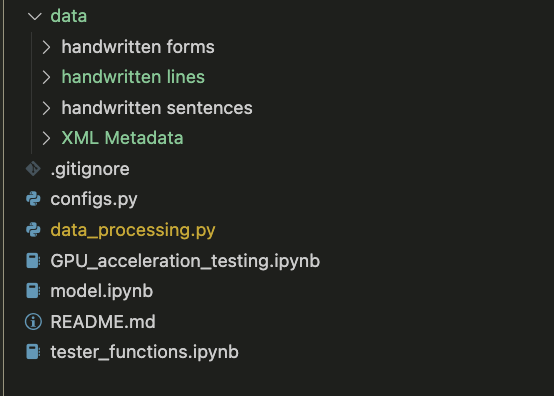
**Image and XML Data for Training:**

The XML and image data are integral to the OCR model's training process. In the data\_preparator function, batches of image data are generated, where the images are converted to floating-point tensors, preparing them for neural network input. Meanwhile, the labels extracted from the XML data are processed into integer sequences. These labels, together with the CTC (Connectionist Temporal Classification) loss function, enable the OCR engine to learn how to recognize characters and words accurately.

**File Management:**

All project files are version-controlled using Git, ensuring robust file management practices. Version control has been crucial, allowing the addition, modification, and deletion of files as needed, while also enabling access to past versions when necessary. Frequently, older file versions were retrieved during development for reference or to recover code that had been altered or removed.

**Folder Structure and File Organization:**

To maintain clarity, a logical folder and file structure was implemented. For example, the OCR engine development code is organized into separate files, ensuring that related functionalities are encapsulated, which reduces complexity. Data used in the OCR training process is stored in a dedicated folder, which contains subfolders that organize the data further for easier access and better readability in the code. Subfolder names followed a consistent naming convention, such as [type of data]\_[content], making it clear what kind of data each folder contains—for instance, a folder named handwritten\_sentences would contain images of handwritten sentences.

**Use of Jupyter Notebooks:**

While the majority of the code is written in Python, much of it was developed in Jupyter Notebooks (.ipynb), which offer the flexibility to run code in blocks. This feature was particularly useful during model development, as certain blocks of code, such as data preprocessing or model training, needed to be run multiple times for testing and debugging. This approach made the development process more efficient and allowed for easier experimentation.

**Other File Types:**

Additional file types, such as .md (Markdown) files and .gitignore, were also used effectively. The .md files provided project descriptions in the GitHub repository, improving documentation. The .gitignore file played a crucial role in ensuring that unnecessary files, such as large datasets in the data folder, were not included in version control. This avoided clutter and made file management easier, particularly when dealing with large datasets.

**File Naming Conventions:**

Consistent naming conventions were followed for Python and notebook files. Python files were named using snake\_case, which not only ensured readability but also simplified cross-platform compatibility. For instance, certain operating systems, like macOS and Windows, handle strings differently, and consistent naming conventions help avoid import errors when working across environments. This attention to detail ensured a smoother development experience and minimized potential issues during deployment.

**Data and file security:**

All files and data are securely hosted on GitHub’s servers, which function both as a hosting platform and a backup solution. GitHub ensures security with robust protocols, and private repositories are only accessible to authorized users, enhancing protection. Files and data are safeguarded by my password and multi-factor authentication (MFA), so even if my password is compromised, unauthorized access is prevented without the MFA credentials. Additionally, data encryption prevents man-in-the-middle attacks during file transfers to GitHub’s servers. GitHub's strong security measures make it highly resistant to hacking attempts.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

The files and data are also stored on my local PC and laptop meaning as development occurs on the PC thus GitHub acts as a backup, as when code or data is meaningfully changed that iteration of the software is pushed up for backing up. GitHub backups also allow access to the files from any local machine if needed. This also means that an events-based threat must occur across all devices to lose all code which is incredibly unlikely.

All files on local machines like the PC and mac are also encrypted and password protected on top of access to the devices already being passwords and pin protected.

A screenshot of a computer error

Description automatically generated

**C08 - Usability testing**

**Usability test preparation:**

The core functions of the software solutions are too able to create and account, login and upload and image and manipulate chats.

In order to test for all of the core functionalities effectiveness I will measure the amount of time it takes to be able to complete all of these takes from different states of the software alongside observational notes about user behaviour on the site. Individuals will also be surveyed about the UX and UI experience they had; for instance, the test will include the time it takes to for someone to create an account alongside the ease and accessibility of doing so being queried from them.

**Results documentation:**

User 1:

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Time (rounded) | UI/UX ex | observations |
| Create account | 22 secs | * Like overall design * Didn’t how confirm password box appeared after entering initial password * Felt intuitive | Nothing major |
| Upload | 11 secs | * Liked upload menu design a lot aesthetically * Didn’t like plain white black colour theme * Said it reminded of chat GPT | * Liked upload menu |
| Delete chat | 4 secs | * Was very intuitive | * Went immediately for the right icon |

User2:

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Time (rounded) | UI/UX ex | observations |
| Create account | 22 secs | * Like overall design * Didn’t how confirm password box appeared after entering initial password * Felt intuitive | Nothing major |
| Upload | 11 secs | * Liked upload menu design a lot aesthetically * Didn’t like plain white black colour theme * Said it reminded of chat GPT | * Liked upload menu |
| Delete chat | 4 secs | * Was very intuitive | * Went immediately for the right icon |

User3:

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Time (rounded) | UI/UX ex | observations |
| Create account | 22 secs | * Like overall design * Didn’t how confirm password box appeared after entering initial password * Felt intuitive | Nothing major |
| Upload | 11 secs | * Liked upload menu design a lot aesthetically * Didn’t like plain white black colour theme * Said it reminded of chat GPT | * Liked upload menu |
| Delete chat | 4 secs | * Was very intuitive | * Went immediately for the right icon |

User2:

**C09 – Software Evaluation**

**C10 – Project Plan Assessment**