When does pytesseract give errors, is there a pattern?

Testing the tesseract-ocr with frames where text is dispersed or when frames are transitioning between embedded texts, we can see a pattern that indicates that these frames get incorrect predictions most of the time.
The frames are binarized, so I dont think it is due to preprocessing.
Preprocessing plays a huge role in the predictions by the network.

The test program is test.py and the test images can be found in the “images” folder. Also the outputs of these frame tests are in the “outputs” folder.

Can you explain how pytesseract works?

Pytesseract is a python wrapper for the Google tesseract-ocr. OCR implementation by tesseract is as follows:

1) The image is hopefully pre-processed, so we have a binary(white and black), tesseract forms outlining and separates them into “blobs”.

2) Blobs are organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces and fuzzy spaces.

3) tesseract uses an Adaptive classifier(LSTM). Recognition proceeds in a 2 phase manner, In the first phase, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page.
Since the adaptive classifier may have learned something useful too late to make a contribution near the top of the page, a second pass is run over the page, in which words that were not recognized well enough are recognized again.

4) A final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate small-cap text.

More elaboration can be found at:
https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf

What other way can this task be performed using deep neural networks?

We will start by pre-processing of the image to aid our OCR algorithm:

> Binary Image: black and white image

> The size of the image preferably not too big

> Delimit the area of interest

> Highlight the text over the background

> Avoid noise, remove pixels that are not part of the text.

Now that our image is ready, we can use the techniques in the paper stated above.

As tesseract-ocr is open-source we can replicate the pipeline and get the state-of-the-art.