Badeed Saleh
Zach Santangelo
Professor Feng, Lui
March 2022

# Undecoder

Unblurring text by estimation

## Abstract

Licenses, faces, passwords, and other sensitive information captured on camera or screenshots can be blurred before being published to "protect" individuals, but this obscured information can still fall into the hands of bad actors. This project seeks to quickly identify the top likeliest set of characters provided to it as a string of text by obfuscating then deobfuscating in various text offsets as a brute force tactic that can be surprisingly effective at finding a likely match by minimizing the error function (difference between the image guess and blurred input) and retrieving information from the pixels available. Some depth in the recursion were discovered by the program before moving on to a different offset due to letters further down being cropped with some error..

## Introduction

Undecoder's goal is to compute the lowest error guess of the provided text as quickly as possible using Python's numpy library and openCV by tackling the hurdles discussed in the article by Dan Petro "*Never, Ever, Ever Use Pixelation for Redacting Text*" in python into an extensible and modular set of components. It is vital to become aware that simply blurring information cannot completely secure one's privacy, and that enough information could be intact for a computer to estimate low difference character sequences for blurred images. Approaches published today often include some deep learning aspect, such as the 2018 paper by Fatma et al. implementing a "Super Resolution Convolutional Neural Network" that enhances the quality of a given

image[2]. Improving the resolution and reversing intentional obfuscation of text information has many implications on security of the uninformed who blur text intending to render it devoid of relevant data. Methods to extract information from any source is of interest to many entities for various reasons. The most secure way to remove the information of sensitive text is by completely replacing it with solid color (black or white).

## Method

Initial relevant variables to consider that are held constant are the differences between individual programs rendering the "same" text with the same font and size. This attempt is rendered in Google Chrome with a standardized html layout for the input message. Guesses made by the program were evaluated for their difference from the input image to reduce the error function (difference between the input text being blurred and compared to provided blurred text).

The decode function catches the error of the guess compared to the input text and the offsets associated with the character string guessed and built based on the error array. Below, one can still faintly make out some of the letters with some offsets, indicating there is enough information that some letters may be computationally extracted by comparing a guess image using a similar enough font, renderer, offset, size, pixelation function,

Figure 1. Example blurred text via convolution of
the pixel values and setting them to the
average of the blockSize standard.

Estimating the difference between two images to obtain the error of the guess required rescaling of the image dimensions to fit one another, which also requires adjusting the indicators placed for the proper width of the image of obfuscated text.

Empirically, this took adjusting through some observation of the size of the text locked in for this trial. Estimations of text sampling comparisons requires sampling for generated colored indicators placed at the end of the input text rendering to allow the right most of the image to be aligned and rendered properly, which can also introduce error in the last block and can require correction if there is a remainder of the image width to blockSize.

In further trials, the font, size, and renderer of the text (chrome, firefox, etc), could be searched through with later installations to further automate the process. Additionally, different pixelation functions with hyperparameters could be incrementally compared and hashed into a database for an optimized system designed for cracking various screen text via the error estimation system.

## Experiments

Shuffled strings of the alphabet and space character "abcdefghijklmnopqrstuvwxyz " were chosen to test how the solution of comparing blurred letters column by column handled against the challenges of whitespace, variable width, and varying offsets. This did prove slow, as was abandoned quickly to shorter strings during debugging, such as this blurred sample of the string "zach code", which was temporarily used as a standard for the images loaded in later to be scaled to a fixed blockSize (8 in our trials).

Undecoder found some low error guess on trials and followed into a two character deep guess for the string "badeedq" for which it found a low error for "b" at a given offset and then attempted "ba" before moving on to another offset as it would have if "b" with the guessed offset was a poor match. Challenges were faced primarily at the empirical difficulties of properly cropping the rendered images to provide reliable

comparisons and results after scaling to allow comparison of the guess with the current text string guess being checked each iteration.

A threshold for the whitespace guesses that is more lenient to the effects of bleed over by other characters to count as a "low" error allowed better detection of white spaces in the provided blurred text image to decode. The larger hurdle was obtaining the proper difference threshold functions empirically for our model with the given tools and properly cropping for comparing. The program matched some inputs that appear close to the initial kernel for the recursive guess function, indicating some promise to the approach with further improvements.

## Sources

[1] Dan Petro, *Never, Ever, Ever Use Pixelation for Redacting Text*
https://bishopfox.com/blog/unredacter-tool-never-pixelation

[2] Albluwi, Fatma & Krylov, Vladimir A. & Dahyot, Rozenn. (2018). Image Deblurring and Super-Resolution Using Deep Convolutional Neural Networks. 1-6. 10.1109/MLSP.2018.8516983.
https://www.researchgate.net/publication/328985265_Image_Deblurring_and_Super-Resolution_Using_Deep_Convolutional_Neural_Networks#fullTextFileContent