HOUSE NUMBER IMAGE INPAINTING

APS360 FINAL REPORT

Samuel Zheng
University of Toronto

Yixin Tian
University of Toronto

Ron Hu
University of Toronto

Jonathan Yan
University of Toronto

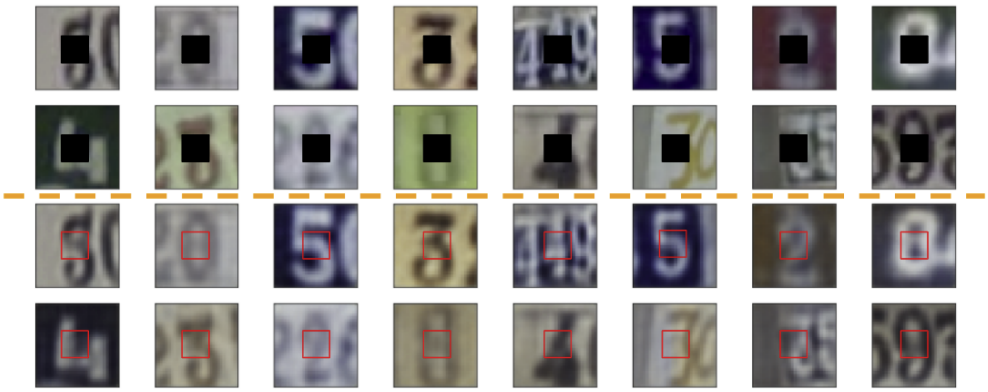

Figure 1: Example inpainting results of our method on images of house number images

1 Introduction

House Number Image Inpainting is a project that seeks to bridge the gap between substandard images and optical character recognition, or OCR. Currently, OCR works well when the characters in the images are explicit, but its performance diminishes rapidly in situations where the character may be blurred, obscured, or ruined in some manner. Our project seeks to use machine learning to repair the images through a technique called image inpainting. Image inpainting works by filling in the damaged sections of images using information from the rest of the image. An autoencoder is well suited for this application because it can learn the most important features of the images by compressing an image into an embedding, reconstructing the image, and comparing it to its original form. As seen in Figure 2, we will train the autoencoder by intentionally masking out sections of house number images and updating the autoencoder based on the difference between the reconstructed image and the ground truth. To test how well our model performs, we will pass the reconstructed images to the OCR to see if it can recognize the correct digits. Image inpainting can be extended to situations such as when a camera takes photos of speeding cars' license plates to use in a license plate recognition software, where there may be blurs or glares.

2 Illustration / Figure

See Figure 2 for the overall training pipeline.

3 Background & Related Work

One popular OCR tool that is well-trained and sophisticated is the Tesseract 4 OCR engine, which is developed and made open sourced by Google since 2006 [1]. The Tesseract operates an LSTM (long short term memory) based neural network that can recognize characters in over 100 languages [2]. However, we will only be selecting numerical outputs from the engine considering our input consists only of house number images.

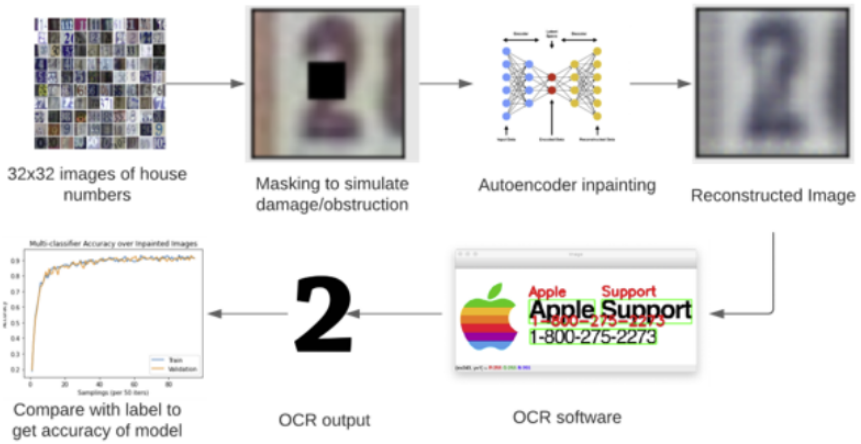

Figure 2: The general pipeline for training the image inpainting model

Image inpainting can be done using a variety of methods, but two popular ones are the Fast Marching and the Navier-Stokes method. The fast marching method is relatively straight forward, and it starts from the boundary of the missing region and works inward, gradually filling each pixel with a weighted sum of its neighbors. On the other hand, the Navier-stokes method is more complex, and it borrows knowledge from fluid dynamics to create edges by matching gradients (brightness), and then finds the color that minimizes variance in the area. [3].

4 Data Processing

Our main source of data used for network's training and evaluation is the Street View House Numbers (SVHN) dataset provided by Stanford [4]. SVHN includes over 630,000 individual digits from street house numbers, pre-cropped into 32x32 pixel size images (Fig. 3a).

(a) Sample of SVHN dataset of 32x32 house number digit images

(b) Fixed 8x8 masking (mid) vs. random masking (bottom)

Figure 3: Data samples and masking type

In order to simulate image obstruction, black squares are drawn onto each image. A sufficient portion of the digit must be covered up such that the OCR will have trouble confidently identifying the value. Our model is trained mainly on 2 separate variations of masking: an 8x8 black square applied in the centre of each image; a black square of random size ranging from 8x8 to 12x12 applied randomly within the image (Fig. 3b). Additionally, data augmentation was

employed to increase the amount of training data. Augmentation techniques include image rotation, colour jitter, random application of black and white filters, and perspective shifting (Fig. 4)

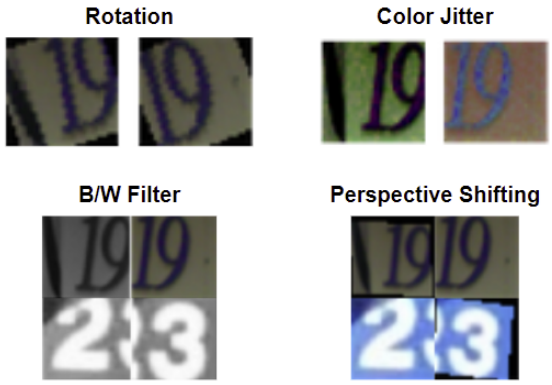

Figure 4: Data augmentation samples

5 Architecture

Our final model is a 3-layer CNN Autoencoder. The encoder consists of the data passing through 3 Conv2D layers, with LeakyReLU applied between intermediate layers as the activation function. Specific layer parameters and LeakyReLU values can be found in Figure 5.

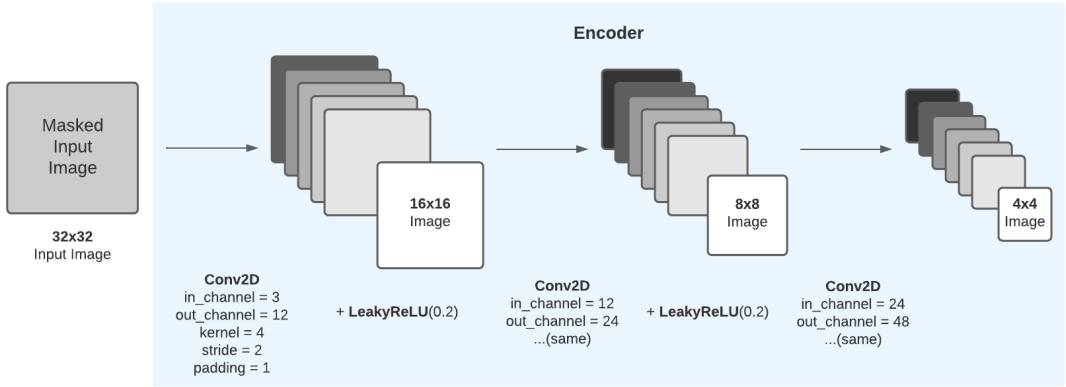

Figure 5: Encoder-side architecture of CNN autoencoder

The decoder is symmetric in design with minor changes: ConvTranpose2D is applied instead of Conv2D and regular ReLU is used instead of LeakyReLU. Layer parameters are mirrored to the encoder and can be found in Figure 6.

Additional training criterion and hyperparameters are as follows:

- Loss Function: MSELoss
- Optimizer: Adam with weight decay of 10^{-5}
- Learning Rate: 10^{-3}
- Batch Size: 128
- Epoch: 12

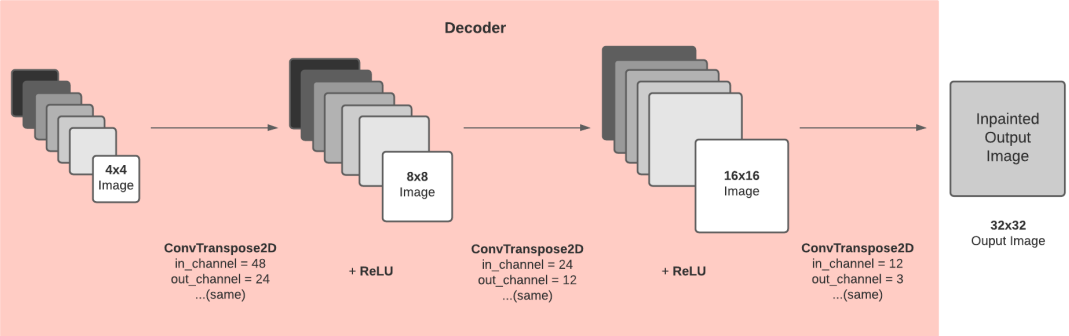

Figure 6: Decoder-side architecture of CNN autoencoder

6 Baseline Model

Our group selected the Telea inpainting algorithm provided by the OpenCV library as our baseline model for its simplicity [5]. It performs well for general purpose inpainting and no extra tuning was required. As such, all one needed was to import the library, provide the image and mask, and call a predefined function to obtain results.

Figure 7: Image inpainting results by OpenCV Telea on masked SVHN dataset

The algorithm decides the colour to inpaint missing pixels by extrapolating surrounding colours. As a result, the algorithm may sometimes extrapolate the wrong colour and inpaint the number with background colours instead, as seen in Figure 7. Therefore, despite the algorithm being able to output decent results, there are aspects where inpainting can still be improved on. Thus, OpenCV's Telea Algorithm serves as a reasonable baseline reference for our autoencoder model.

7 Quantitative Results

Traditional means of measuring a model's performance involve evaluating a model's loss and accuracy. However, this evaluation approach is infeasible for our problem since inpainting is a subjective process. Therefore, we can not evaluate the quality of inpainting solely through measuring the loss and accuracy of our model. To evaluate our model, we feed the output of our model, the inpainted images, into a pre-trained OCR model to obtain an OCR accuracy. The goal is to make the OCR accuracy on the inpainted images as close as possible to the OCR accuracy on the ground-truth images. As the inpainting model improves, the gap between the two OCR accuracy is expected to reduce.

From Table 1, the OCR performs well on ground-truth images, achieving an accuracy of 95.8%. However, the OCR only achieves an accuracy of 13.2% on masked images. This indicates that our masking sufficiently obscures the number such that the OCR cannot properly operate.

Reference Metrics	
OCR accuracy on unmasked ground-truth images	95.8%
OCR accuracy on masked images (8x8 Centre-Masked)	13.2%
OCR accuracy on masked images(Random mask)	10.4%

Table 1: Reference metrics used as absolute standard

From Table 2, the final fine-tuned CNN autoencoder model is able to achieve 95.45% OCR accuracy when inpainting 8x8 centre-masked images, only a 0.35% difference compared to using ground-truth images. The model also performs decently on randomly masked images, achieving 87.14% OCR accuracy. This demonstrates that the inpainted images allow the OCR to operate almost as well as if the images had not been obscured in the first place.

10,000 Test Images Inpainted By	**OCR Accuracy**
Baseline OpenCV Telea Model (8x8 Centre-Mask)	84.3%
Baseline OpenCV Telea Model (Random Mask)	64.7%
Initial autoencoder (8x8 Centre-Mask)	91.53%
Final fine-tuned CNN autoencoder (8x8 Centre-Masked)	95.45%
Final fine-tuned CNN autoencoder (Random mask)	87.14%

Table 2: Evaluating inpainting performance across different models

8 Qualitative Results

A random selection of inpainting results by our final CNN autoencoder model on both centre-masking and random masking are shown in Figure 8 and 9 below; red squares are selectively added to highlight the inpainted area. We can observe that the model generally performs well and accurately inpaints the missing part of the number. This result is expected, as the OCR is also able to perform well on recognizing the inpainted numbers, achieving accuracies of 87-95% based on the type of masking applied.

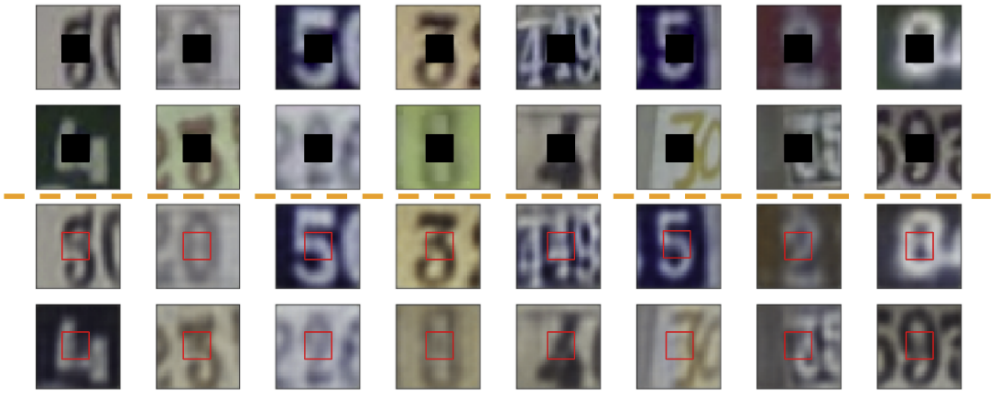

Figure 8: Input fixed-masked images (above) | inpainted images by final model (below)

However, mis-inpainting still exists. Figure 10 shows some examples of mis-inpainted images compared with their ground truths, where ‘4’, ‘9’, ‘0’, are mis-inpainted as ‘1’,’0’, ‘8’ respectively. This mis-inpainting phenomenon may still occur despite high OCR accuracy, thus, precaution should be exercised when using the inpainting model to process obscured image data, as sometimes the inpainted information could be misleading.

Lastly, we found that the model does not perform well if the given inputs are masked with patterns it has not encountered during training. This includes masks of non-square shapes, masks larger than size 12x12 pixels, and masks of other non-black colours. This is because we applied only certain variations of masking along with the limited ability of autoencoders to learn past training data. This will be further elaborated under Discussion.

Figure 9: Inpainted result on randomly masked images. Ground truth (top), inpainted (bottom)

Figure 10: Samples of mis-inpainted images (right) compared with ground truth (left)

9 Evaluate on New Data

To evaluate performance of our model on novel data, we took photos of house numbers that were in our neighborhoods. As seen in Figure 11, we first cropped each digit into size 32x32 images and randomly masked out sections in the same manner. After passing the batch of images into the autoencoder, the OCR was able to recognize the numbers in all 36 images with 95% accuracy. This actually exceeds our model's training accuracy of 87%, which was because the images that we took were of higher quality than some of the dataset images with poor quality, as seen in Figure 12. The high quality images have clearer edges, which helps the autoencoder inpaint the masked section better. As a result, the OCR has a higher chance of recognizing the digit correctly.

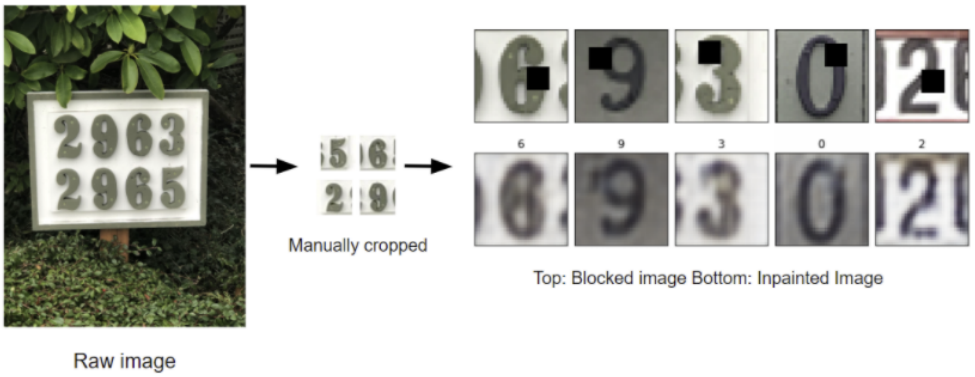

Figure 11: Process of converting novel data to testable data

10 Discussion

Originally, we only developed one model that was trained with 8x8 centre-masked images. Having seen its good performance on images with the same masking, we assumed that the model had fully learned the embeddings and was capable of inpainting other variations of masking. This was unfortunately not the case, as the original model performed terribly when it encountered any mask that was not explicitly an 8x8 square mask in the centre (Fig. 13).

While these results are surprising, they are not fully unexpected. The model behaves just like a standard autoencoder: it learns the image embeddings through training data and reconstructs images based on this embedding when supplied with new input data. As previously discussed under qualitative results, since the original autoencoder never learned the embeddings of any other masks, it would be reasonable to infer that it did not recognize the random masks as "obscurities" and thus, did not inpaint.

Figure 12: Poor quality images

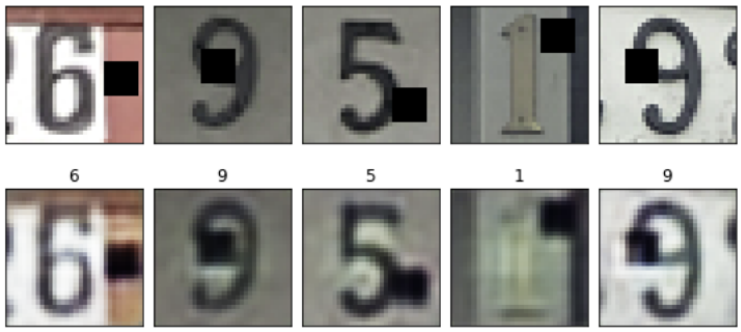

Figure 13: 8x8 centre-mask trained model does not perform well on randomly masked images

This demonstrated to us that autoencoder networks will only learn what it is taught—we can not expect the network to extrapolate its knowledge to solve similar problems. This also became the motivation for us to build the second model trained on randomly masked images. However, it is still important to acknowledge that these models most likely will not work with real-world examples, such as a photo of a license plate obscured by glare from the sun. As such, much more different shapes, colours, and variations of masking must be considered for the network to accurately inpaint. Since there is no feasible way to accommodate every possible "mask," we begin to see the physical limits of autoencoders for inpainting.

Asides from autoencoders, inpainting can also be realized with machine learning through GAN networks. GANs approach the inpainting problem from a different perspective: instead of the network trying to memorise image embeddings for inpainting, it tries to directly generate a realistic inpainting based on past experience and learned patterns in an attempt to fool the discriminator. As GANs learn from patterns and discriminator feedback, there is less need for us to supply the network with every possible "mask", but rather provide a sufficient variety of shapes, colours, and permutations. Furthermore, GANs eliminate the problem of blurry outputs as a result of deeper networks. Autoencoders rely on the depth of the architecture to learn image features for inpainting. As images are processed into a deeper dimensional embedding space, they may be forced to discard important info and thus, result in blurred outputs.

11 Ethical Considerations

There are 3 ethical issues concerning image inpainting: authenticity, reversibility, and documentation. This includes that all modifications do not falsely portray the image, are distinguishable from the original, and are strictly documented [6]. Image inpainting technology may possibly be used for malicious purposes, such as falsifying documents. As such, ethical risks must be acknowledged and the above standards must be upheld to preserve integrity.

Similarly, there are ethical issues regarding data collection. For many public datasets, it is impractical to receive informed consent from primary stakeholders [7]. This includes the SVHN dataset that is used in this project, as its original data is collected from Google Street View [4]. Therefore, it is important to recognize potential damage to public interest, such as the invasion of individual privacy, and evaluate if the benefits of the project justify the risk.

12 Project Difficulty / Quality

The key challenge of image inpainting lies in synthesizing visually realistic pixels for the missing areas of the image. In reality, the missing area could take on different shapes and appear in different spatial locations. Some architectures have been introduced to address this complexity such as GAN with contextual attention but they tend to be both time and resource intensive to train. In addition, finding an effective, quantitative way to measure the inpainting result rather than just using test loss is also one of the major challenges that we face.

12.1 Randomness in Real-world Images

In real-world image data, the missing regions often take on different shapes and spatial locations. We try to simulate this randomness by increasing training data with data augmentation and applying random masking with varying positions and shapes. However, the large amount training data sometimes exceeded Colab RAM limits, thereby constrained our ability to train models.

12.2 Quantitative Evaluation Method

Deciding on an effective way to quantitatively evaluate our inpainting model's performance is also one of the major challenges. We decided to use a pre-trained OCR model on the inpainted images for obtaining accuracy. By comparing the OCR accuracy against ground truth images, we saw a significant increase in accuracy once we fed inpainted images into the OCR instead of directly using masked images.

12.3 Overall Quality

The project result meets the expectation. We trained models that can inpaint images with both fixed maskings and random masking using a CNN autoencoder architecture. The result proves that with sufficient training, the autoencoder can reconstruct images similar to their ground truths, quantified by the OCR accuracy on inpainted images. However, shortcomings still exist. For example, the model does not perform well on inpainted images with any masking variation not provided during training.

13 Link to GitHub

GitHub Repo (click here)

References

[1] "Tesseract OCR: Text Localization and Detection," PyImageSearch, 2020. [Online]. Available: www.pyimagesearch.com/2020/05/25/tesseract-ocr-text-localization-and-detection/. [Accessed Apr.8, 2021].

[2] Tesseract-Ocr, "Tesseract-Ocr/Tesseract," GitHub. [Online]. Available: www.github.com/tesseract-ocr/tesseract. [Accessed Apr.8, 2021].

[3] J. Khan, "Guide to Image Inpainting: Using machine learning to edit and correct defects in photos," Medium.com, 2019. [Online]. Available: https://heartbeat.fritz.ai/guide-to-image-inpainting-using-machine-learning-to-edit-and-correct-defects-in-photos-3c1b0e13bbd0. [Accessed Apr.8, 2021].

[4] Y. Netzer , T. Wang , A. Coates , A. Bissacco , B. Wu , A. Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning ," Google Inc., Stanford University. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf. [Accessed Apr.8, 2021].

[5] OpenCV, "Image Inpainting," OpenCV. [Online].
Available: https://docs.opencv.org/master/df/d3d/tutorial_py_inpainting.html. [Accessed Apr.8, 2021].

[6] American Institute for Conservation of Historic and Artistic Works, "AIC Code of Ethics and Guidelines for Practice," American Institute for Conservation of Historic and Artistic Works, 1994. [Online]. Available: https://www.nps.gov/training/tel/Guides/HPS1022_AIC_Code_of_Ethics.pdf. [Accessed Apr.8, 2021].

[7] D, R. Thomas, S. Pastrana, A. Hutchings, R. Clayton, and A. R. Beresford, "Ethical issues in research using datasets of illicit origin," in Proceedings of IMC '17, 2017. [Online]. Available: https://www.cl.cam.ac.uk/ drt24/papers/2017-ethical-issues.pdf. [Accessed Apr.8, 2021].