MayaNet: MayaScribe Module - Edge Detection Using Convolutional Neural Networks for Epigraphic Studies

Chrsitopher Salguero
Stanford University
450 Serra Mall, Stanford, CA 94305
csalguer@stanford.edu

Abstract

MayaNet aims to utilize a convolutional neural network (CNN) to produce glyph features from images of Mayan codices. These Mayan records are the last of their kind, and contain an ever degrading set of writings inked using the writing system known as Mayan Script. Preservation efforts have leveraged imaging technologies to help maintain a copy of the visual state of these documents. CNN's transform the visual domain, and this project explores how tools like CNN's can extend preservation imaging to facilitate graphic-oriented studies not previously feasible. Novel CNN approaches like holistic nesting of features, coupled with the diverse availability of pre-trained network architectures allow for the use domain adjacent knowledge to inform this visual task, given the scarcity of these glyph images. Processed image data remains highly guarded thanks to the high cost associated with traditional processing methods, such as artistic rendering and computer aided modeling. CNNs can offer lower time and manual costs and even additional data not readily perceived by the human visual system. The glyph features produced by MayaNet could be used for various tasks that before required heavy manual engagement. These tasks include but are not limited to: segmenting and extracting glyph segments, compiling a set of healthy and differentiable references, or for other forms of epigraphic study.

Figure 1: Various Mayan Reference Glyphs

1. Introduction

The study of epigraphy centers around deciphering and classifying inscriptions. The relative visual health of these inscriptions has a direct influence on their differentiability and decipher-ability. Regarding the epigraphic studies done on Mayan Script, one of the major limiting factors in advancements in this field of work has been in the quality and quantity of the available inscriptions. The remaining inscriptions for Mayan Script are few, amounting to four written codices supplemented by various other stone inscriptions on stelae. The inscriptions on stone medium are understandable long-lived; inscriptions found on the tree-bark medium employed by the codices however offers much less protection to the ravages of time. This is perfectly exemplified by the Codex Dresdensis. Despite being considered one of the best preserved specimens, it retains extensive water damage incurred from the bombing of Dresden in World War II. High-resolution imaging has preserved a record of its current state and serves as a warranty against further degradation.

Convolutional Neural networks (CNNs) have changed both the conduct of Machine Learning research and more specifically in the realm image analysis. The image features relevant to the glyph extraction task proposed by this project are edge features and contour features. Edges define boundaries or discontinuities in an image's pixel information. Contours similarly are the curves that these boundaries or discontinuities form across areas of the image. These both are of particular interest, given that written information is contained within areas of defined edges and sweeping contours. These serve to distinguish the written encoding (e.g. a brush-stroke) from the grain of the medium it sits on. To produce a healthy set of inscriptions, then would amount to producing an image with well defined edges and well-defined contours that best capture the glyphs structure and syntactic value. The key goal of this project is to take images that contain Mayan Script as input in order to produce an edge/contour mapping as output.

The image degradation present in the remaining codices comes in in two flavors: occlusion and discontinuities. Both kinds of image defects introduce noise can impact the health of the edge/contour mapping sought by this project. This noise impacts the legibility of inscriptions. The worst affected glyphs have completely compromised edges or contours and as a result are entirely illegible.

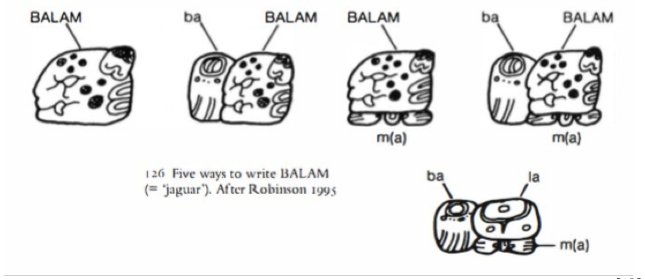
2. Related Work

The features relevant to our task are historical computational problems that have been explored by other works. The edge detection problem in particular has some of the most diverse approaches, many informed by the computational paradigm of their time. They can be grouped into mathematic early works, informed manual approaches , and learning-based approaches. Early works' mathematical approaches are best exemplified by the filter/kernel method of the Sobel operator[4]. The intuition of this method was to define an edge as a discontinuity. Detecting edges then depended on exploring the gradient space for discontinuities. Filters that used central differences could approximate these gradients and saw many iterations like the Prewit, Roberts, and Kayyali that leveraged this first-order derivative[6]. Following the same mathematical intuition, a solution exploring the second order derivative space of images emerged in the Laplacian Operator[5]. These mathematical approaches culminated in a pseudo-mathematical algorithmic approach, the Canny Edge Detector, that remains heavily in use to this day [7]. This method augmented the intuitions prior with the additional step of non-maxima suppression, where only the edges (local maxima) are kept else discarded. These early works arose as the fields of Computer Vision and Image Processing were being formalized.

The subsequent group of works, the informed manual approaches, saw hard mathematics fall in favor of keen hand-designed methods supplemented by the then new area of informatics. Statistical Edges and the Pb/gPb methods leveraged the numerical approaches being developed at the time [8][9]. As these numerical approaches developed, and the field of Learning Theory formalized and grew, the next group of methods, the Learning-Based approaches soon emerged, with Structured Edges serving as exemplar [10]. The newest paradigm shift seen to date is that of the Neural Network, and we see these modern approaches draw from areas of Linear Algebra for admissibility and leverage highly designed features for hierarchical learning. These approaches include the most relevant works to this project's domain: contour aware edge detection CNN's like the recently developed DeepContour project[1], and the Holistically-Nested Edge Detection Schema[2] (HED) which serves as this projects implementation.

DeepContour sought to capitalize on the Deep Learning trend that continues today, whereby hand designed features are instead learned to improve their discriminative power and in turn their generalization. It defined contours as a set of distinct classes or "shapes" to be learned. This set of shape classes leaps from the binary schema of contour and non-contour, to a multivariate one where many types of shapes exist, and combine to form a what we call a contour. HED takes a similar high-leveled approach when it defines a contour. HED is similar to DeepContour in its compositional definition of a contour/edge. Whereas the former defines it by a composition of shapes, the latter defines it as fusion of progressively refined edge mappings. This approach does not subdivide into areas or type, but rather seeks to fully train and predict on images, as seen in its CNN bootstrapping of VGGNet. It also uses a nested set of multi-scale features (the progressively refined edge mappings), a property also leveraged by deep supervised nets, to guide the classification of edges.

While either of these latter two CNN based approaches could serve as the implementation of MayaNet, the HED better lent itself to the the natural intuition of reading – meaning is fully contained in a writing system's basic unit; not much is gleaned from a unit's own parts (or else the subunit would be the basic unit of the writing system). Sub areas then carry little to no semantic meaning and are better explored through a progressive sense of detail to bring an edge into focus. The whole image-to-image nature of the HED extends nicely to our particular task of extracting semantically intact information. Note that while a Mayan glyph may be comprised of sub-components, similar to the radicals found in Chinese Characters, they do not capture as complete a semantic or phonetic picture to warrant learning them. In fact, the degradation and uncertainty it brings on a per radical basis would compound and propagate this uncertainty up our composition of a glyph's edge map. The holistic approach for which HED is named best captures the project's image to image task: to extract as large a semantic area of the glyphs a possible across a highly variable feature space where different granularity of details are required. An application of transfer learning using this model will serve as our approach. The major downside to this approach , however is that it may not have a method of scaling based on local feature relationships. This is of concern due to Mayan Scripts tendency for bi-modal line weighting, where a thicker line-weighting defines the glyph shape at large and and a thinner line-weight is used for detail and substructure. There is also a lack of transformations we can achieve for the purpose of data augmentation while while maintaining glyph integrity.


BALAM
ba
BALAM
BALAM
ba
BALAM
m(a)
m(a)
126 Five ways to write BALAM (= 'jaguar'). After Robinson 1995
ba
la
m(a)
Figure 2: Structural decomposition of a Glyph five ways[13]

3.0 Structural/Schematic Specifications

Previous work done by groups such as the MAAYA project[11], and individual organizations like FAMSI[12] have centered around hand-derived glyph images and manual vectorization of segmented images. This project aims to tackle the same domain of these previous forays by producing pen and paper adjacent output for a glyph (black on white/transparent background). Given the bimodal nature of line weights in Mayan Script, the output edge map should be weighted to allow for enough fine details to be represented so few thin line-weight edges are clipped. Conversely, the larger line weights must also be represented enough to where they contribute to the larger glyph shape as desired but without whitewashing the other layer's contributions within the regions bounded by the edges in this layer. Note that our output corresponds nicely to an ideal yet fuzzy edge mapping. The fuzziness allows for the fine detail to be represented and not thresholded. We make one change to the HED schema for the following reasons --- edges are represented by brightness highs and most everything else maintains a zero or near zero value (corresponding to white edges on a black background). In order to achieve this pen and paper adjacent format from this edge mapping, we must invert it. The details for formalizing the network structuring and methodology is described below.

3.1 Definition & Formalization

Regarding training of our data, we have a dataset *S* represented by *n* tuples of $(X_i, Y_i) i = 1 \ldots n$ where *X* represents the image and *Y* represents the ground truth binary edge map for the image *X*. Conventionally, we also call all standard network layer parameters as **W**. Recall the side output schema mentioned earlier. Each of these side output is associated with a classifier, which also has its own set of parameters. If we want to have *m* different side outputs of progressive detail, then each of these *m* layers requires its own classifier parameters. As objective function,

$$\mathcal{L}_{\text{side}}(\mathbf{W}, \mathbf{w}) = \sum_{m=1}^{M} \alpha_m \ell_{\text{side}}^{(m)}(\mathbf{W}, \mathbf{w}^{(m)}), \qquad (1)$$

expresses the sum of *m* separate image-level losses expressed by the term l_{side} . *Note* though the biased nature of edge mappings on natural images (which HED trains on and which our model will too), where one class, the non-edges, far outnumbers the other, edges, the HED compensates for this by utilizing a class-balanced cross-entropy loss function. We can have an additional balancing weight , β, attune all the contributions of the images spacial dimensions.

$$\ell_{\text{side}}^{(m)}(\mathbf{W}, \mathbf{w}^{(m)}) = -\beta \sum_{j \in Y_+} \log \Pr(y_j = 1 | X; \mathbf{W}, \mathbf{w}^{(m)}) - (1-\beta) \sum_{j \in Y_-} \log \Pr(y_j = 0 | X; \mathbf{W}, \mathbf{w}^{(m)}) \qquad (2)$$

Here, $Y_{-¿}$ $Y_{+¿}$, would be the edge and non edge ground truth label sets respectively. The probaiblity term is computed using the sigmoid function operating on the activation value at pixel *j*. At each side output layer, we obtain the edge map predictions $\hat{}Y(m) side = \sigma(\hat{}A(m) side)$ where $A_{side}^{(m)} \equiv a_j^{(m)}, j = 1, \ldots, Y$ are activations of the side output of layer m. To use these outputs in a constructive manner, we then perform the Holistic part of our Holistically Nested Edge Detector and we have an additional layer that weights these side layer's individual contribution to the fusion. This layer has its own loss function denoted by

$$\mathcal{L}_{\text{fuse}}(\mathbf{W}, \mathbf{w}, \mathbf{h}) = \text{Dist}(Y, \hat{Y}_{\text{fuse}}) \qquad (3)$$

Here,

$$\hat{Y}_{\text{fuse}} \equiv \sigma\left(\textstyle\sum_{m=1}^{M} h_m \hat{A}_{\text{side}}^{(m)}\right)$$

where *h* is the fusion weight parameter we introduced and given *m* total side outputs, we will have a total of *m* weight scalings. The optimization is achieved by minimizing the combined loss seen from the outputs as well as the loss seen from the final fused layer. Formally, we seek to minimize the set of parameters such that

$$(\mathbf{W}, \mathbf{w}, \mathbf{h})^\star = \text{argmin}(\mathcal{L}_{\text{side}}(\mathbf{W}, \mathbf{w}) + \mathcal{L}_{\text{fuse}}(\mathbf{W}, \mathbf{w}, \mathbf{h})) \qquad (4)$$

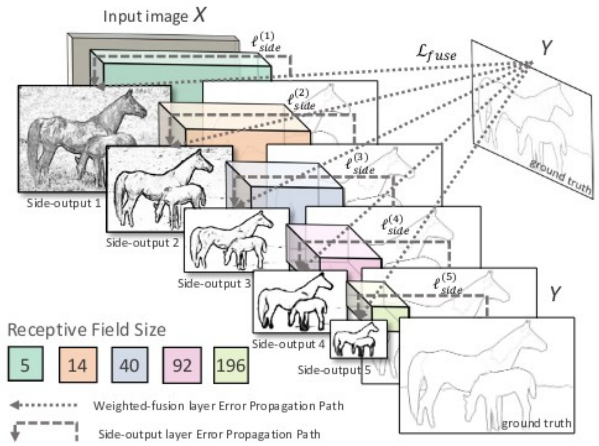


Figure 3: Structure Diagram of the HED Scheme[2]

Note that this projects' implementation of HED scheme also uses the VGGNet architecture for the very same reasons explored by Xie et al.[2] VGGNet pre-trained would supplement any dataset to be fed in for fine-tuned training. Recall that edges are defined to be sharp changes in brightness, or a large gradient change along the brightness axis. This makes edges a low-dimensional feature, and VGGNet has been shown (along with many other pre-trained deep neural networks) to lend well to transfer learning on low dimensional features. Networks always struggle with high-dimensional feature modeling because they require vast amounts of data to be fed in to propagate so deeply inwards to be of effect. The pooling layer after the fifth convolutional block of VGGNet is removed along with all dense layers to "significantly reduce the memory/time cost during both training and testing"[2].

3.2 Methodology & Approach

The implementation of the baseline for this project followed the HED method VGGNET boostrapping explored in the paper by Xie et Tu[2], though the hard requirements for keeping this architecture as opposed to another fully convolutionally "blocked" net like ResNet which also features five convolutional chunks that could be wired in the same format. Given that my dataset's images differ from ImageNet's natural images, fine tuning is a hard requirement for this network to be extensible to the domain of textual images. Training time saw a similar data set used by the creators of HED, which was the Berkeley Segmentation Dataset (BSDS500). Instead of BSDS500's vanilla version, MayaNet utilized the augmented version which has transformations like scaling and rotation to provide more data. While it would be preferred to have the Data Augmentation occur only on the data set provided by this project, the strict rules for glyph transformation along with time constraints , it was not explored but is a hard requirement for any future works. Image transformations normally have an effect on textual data which tends to be very orientation dependent, and the further constraint of Mayan Script as a logo-syllabic script deterred any cursory exploration. Any strong effects produced from training on the augmented set are expected to be watered down by the fine tuning that would occur during the addition training step required for transfer learning.

While the approach outlined in HED allows for the re-correction of bias between the number of non-edges vs. edges, the prevalent fading of the glyphs ink in the images might prove troublesome to this single weighted approach. A fuzzy bias correction with three classes, a strict edge, a non-edge and interstitial/supposed edge might make more sense, but given the already niche nature of the task and degraded visual data, this alteration to HED was saved for future work. The cap at the end of the network is then comprised of an aggregator layer that "fuses" the side inputs into one single image that represents the image's "Holistic Edge Map". Each of these side out-puts is an image in its own right, but contains various degrees of edge strength, with early layers retaining vestige edges of moderately strong features in the base image, and each progressive layer paring down on these vestiges until the final side output highly resembles line art/ground truth. The ground truth labels, were arguably just as hand engineered as the data-set, though not as manually intensive. The ground-truth labeling was set to be a traditional implementation of Canny Edge detection. While this type of edge detection also had vestige edges and resulted in a bubble lettering effect on scripts, it also arguably retained the most granularity in its detail when compared to first order methods like Sobel.

4.0 Dataset & Preparation

Eliminating manual intervention proved difficult in the earliest stages of the data-preparation, so much so that my original explorations into automatic image segmentation ran into both implementation hurdles and accuracy issues. The task of image segmentation might warrant a further exploration, but given the time constraints involved, manual cropping was deemed sufficient. Given that glyphs are much preferred to stay intact and whole (a cleft glyph may not be distinguishable), this project opted instead for a hand-prepared corpus of slight better quality, however, sacrificing in terms of quantity.

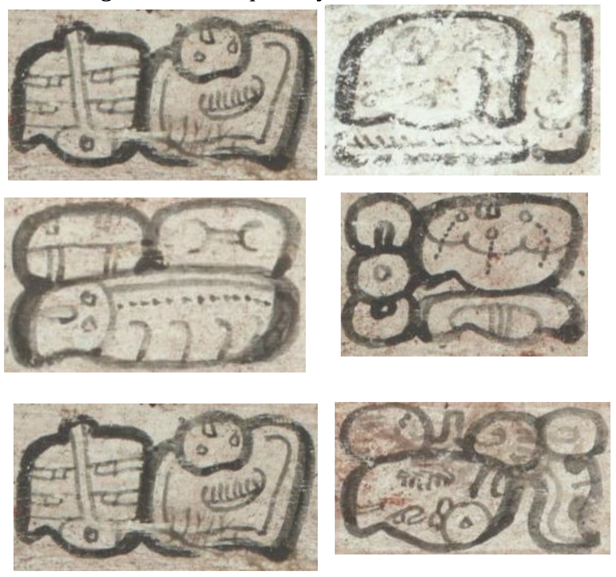


Figure 4: Set of Raw Images

Images were prepared so as to include solely the textual portions from a codex. Mayan Codices are heavily illuminated manuscripts, containing deific pictographs that served as phonetic rebuses and mnemonic supplement to glyph text. While these illuminations technically are abstracted versions of Mayan Script, they are nontheless, *the* scarcest of any of the writing features, due to their high manufacturing cost. Images were originally kept within a scaled dimension of 400 X 200 px blocks of glyphs arranged in a foursquare (2 x 2) pattern. Most glyphs resemble beveled rectangles, and fall into neat grid arrangements. However, the Codex Dresdensis is hand crafted, and thus the glyphs were not perfectly arranged. As a result, some rows tended to drift upwards slightly and some glyphs were squashed to make room for a previous, over-zealously drawn glyph. This made segmentation on the block level very tedious and only slightly affected the glyph level crops. The glyphs, however, did seem to fall within a defined range of aspect ratios 4:(2-2.5] (using inclusive/exclusive range notation), and this was extremely consistent. Given that the model does not contain any fully connected affine layers, the dimensionality requirements for input are rather lax. Images ultimately were allowed to be variable in dimension size using the aspect ratio ranges above to scale any outliers of which I encountered five. These (2 x 2) blocks were further subdivided into individual glyph segments. Blocks became harder and less utile to extract given how much damage was prevalent along the edges of the page medium, likely due to that water damage it had infamously sustained. A total of around 116 text block image partials as well as 312 individual glyph block image partials were cropped to be fed into MayaNet. This figure was originally much closer to double amount seen here . This was due to use of an artistic rendering that had originally been conceived to serve as ground truth, but major inconsistencies in the glyphs forms between the original and reproduction would yield biased results. The 380+ glyph level images were not considered in the current version of MayaNet. There was no elimination of duplicate; such a cross validation would require either scribal knowledge outside of the scope of this project extreme time cost. It is posited that these extraneous duplicates would serve as a passive form of data augmentation, techniques which are a hard requirement for future works.

4.1 Data Format & Pre/Post Processing

All images are lossless PNGs. Input images are also mean centered and normalized, then thresholded to remove the majority of the background image. The ground truths are generated from this preprocessed image and are Canny Edge Maps using an "auto-thresholding" technique of being within one standard deviation of median value of the image. This standard deviation was not calculated from the image but maintained across the data set much as the mean centering values applied to each of the RGB channels in the image. The output of our system will be a monochrome image, containing only the edge/contour features we seek to extract from the input inscription images. The output of our baseline implementations have their coloring schema inversed as described in §2.0, so an additional step to invert the image is required to properly render the inscription as if on paper (black outline of character with a white background).

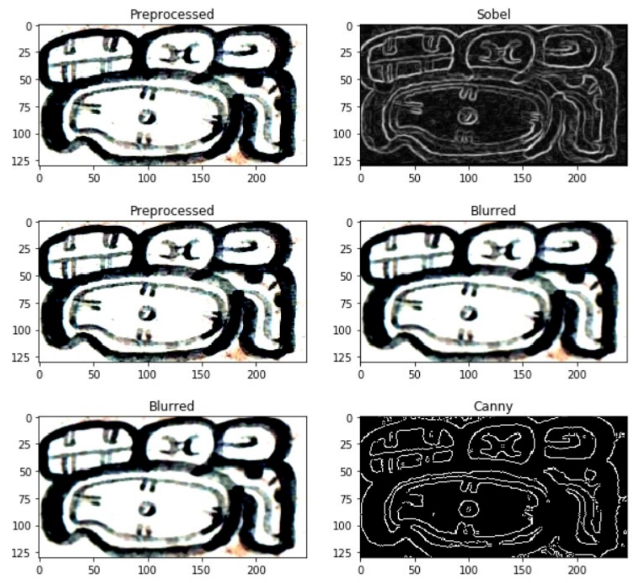


Figure 5: Input vs. Sobel Ground Truth (top two), preprocessed vs. Gaussian Filter Loss (middle two), Blurred vs Canny Edge mapping (bottom two)

5. Results & Analyses

The implementation of HED had varying results. While the images produced, did indeed have strongly defined contour bounds, the line-weighting can be bit strong to the point of excess. Note that this line weight corresponds to the last of our layers, and thus high dimensional. We see major fogging/blurring of internal regions bounded by these strong line-weights as it had been expected if the outer bounding weights were too strongly weighted. It seems that fine-tuning had negligible effect on the attenuating these high dimensional signals. It may be that it is the actual holistic fusing step that completely removes the detail from these glyphs but given the inertia of the deepest convolution block, it is rather unlikely. This is apparent in solitary glyph image outputs as well as whole text-block inputs. This might have to do with the amount of side output layers used. Textual image data may not benefit from convolution chunk depths of more than four. A reduced number of these stacked convolution chunks might be more appropriate given the feature we seek extracted is low dimensional and better captured within these first layers anyways. Given the feature boost sought from pretrained models however, the project is tied to using this fixed beginning 5conv chunk architecture. Changing the number of chunks, experimenting on their arrangement using extraneous chunks, and other variant modifications to the neural network schema is not possible to any sizeable degree. This avenue of model variation was not explored.

Uncanny detection, while horrendously jagged and inconsistent still maintained a remarkable amount of detail when compared side-by-side with the HED output. The blurring had minor to no effect on the glyphs that were heavily geometric in shape (as these tend to be the strongest outlined). As expected, there was loss encountered when considering the more delicate glyphs. The downside of this method however is the Gaussian blur filtering that must happen to reduce noise in Canny Edge Mappings. Thankfully the raw images were of high enough resolution where this blur operation was not terribly destructive . The few examples where this was destructive was in highly faded glyph (pages 1-3 of Codex Dresdensis). These images lost definition in the thin-weighted sections, and the worst results occurr in glyphs with hatchmarks or dots .

Concerning future representation of these results, it might be prudent to explore how image differences might better portray the fusings. It might also be prudent to selectively fuse certain image layers, though conditions for side output dropout might change some of the class-balancing the loss functions for HED try to establish given that the side-outputs were described as being constant. The learning rate was chosen adaptively, in hopes of better coasting down in speed as our SGD optimizer trained the network on the BSDS initial train data. The SGD optimizer choice moreso due to the stochasticity of the batchsize of one that is explored further below. Note that while the fogging here is quite concering, this is still a

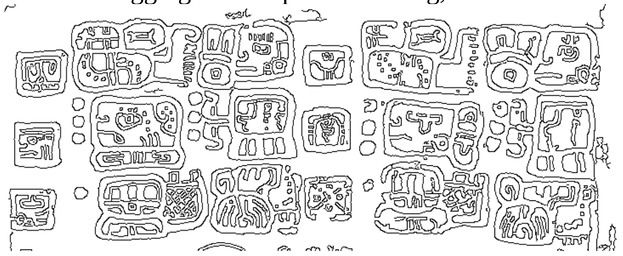
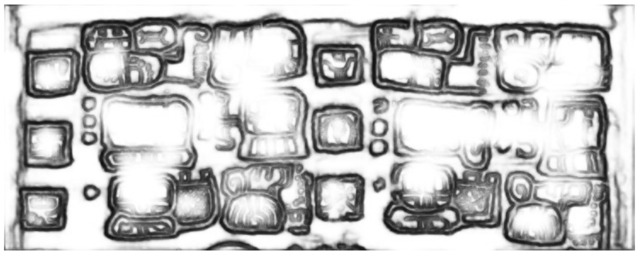



Figure 6: Uncanny Edge Ground Truth (above) and HED First Train Output (epoch3) (beneath) on Codex Dresdensis p.37

result gleaned from before our transfer learning requisite of a fine-tuned train cycle. Given the lack of data, and time-constraints for a sizeable foray into augmentation, I chose to save checkpoints for a full 15 epoch which equaled the amount of epochs used on the base implementation of the HED. The mini-batch unfortunately was kept at 1, and was due to some highly irregular buggy output and segmentation faults that were occurring on batches greater than 1. It is unknown whether indeed this is a result of PyTorch's tensoring into GPU, or actual malformed image header data from having used a third party application for original image collection.

Results varied to some degree when comparing the later epochs8-epochs14. As training increased in the epoch cycles completed, one would expect there to be a more attuned weight scaling for the fused, layer. A cursory testing was done on three random glyphs from the training set to check the side output layers for admissibility and consistency. There seems to be an issue in the bias correction parameter. The deepest convolution block, conv5, represents the highest dimensional perception and also has the largest receptive field of all the chunks, it also has the worst resolution. In this side output, instead of a nicely haloed and blurry effect, we see a very harsh, bitmap like pixelation effect that is seen to a lesser extent on the layer preceding it, conv4. The effect, however, is drastic on the fused image as we see a superposition of the pixeling throughout the image. The VGGNet did impose stride 1, and only after having trained and seeing this input did I see that stride 1 may not have been a default value to the convolution layer as I had passingly assumed. What makes this conjecture tenuous though, is the increasing strength of this pixelation as the different epoch checkpoints , from epoch0 – epoch14 , were cross checked.

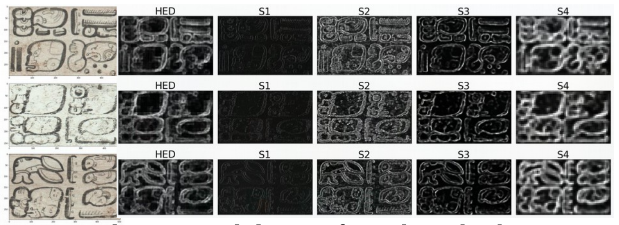


Figure 7: Preliminary validation of epoch14 checkpoint on Glyphs

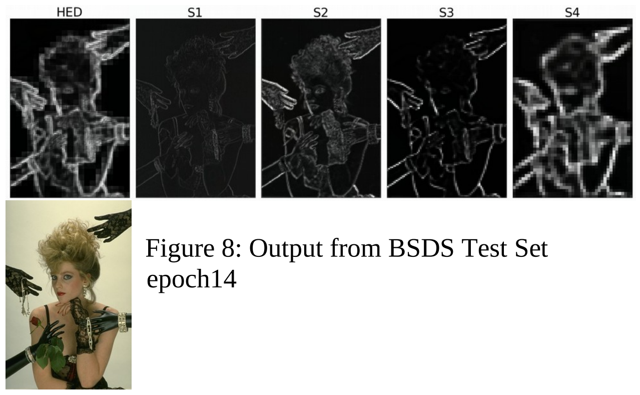


Figure 8: Output from BSDS Test Set epoch14

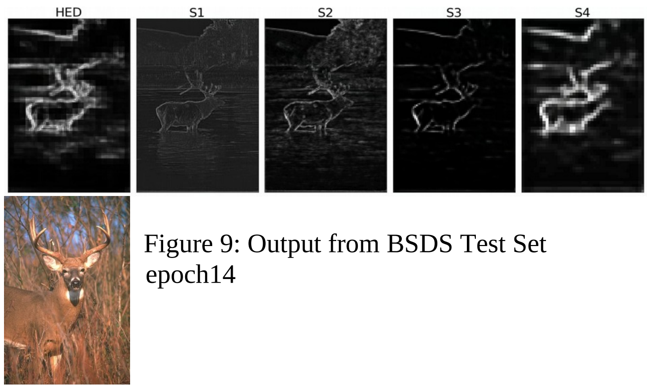


Figure 9: Output from BSDS Test Set epoch14

Results seen using the post transfer learning fine tuning training cycles were a whole other matter however. The results were quite surprising to say the least, given the heavy pixelation seen in side out-puts s4, s5 (corresponding to conv4 and conv5 chunks respectively) from the last checkpoint trained using the standard BSDS dataset for HED training. HED fused and side output below

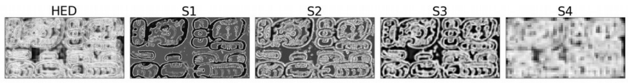


Figure 10: Tested using epoch5 checkpoint

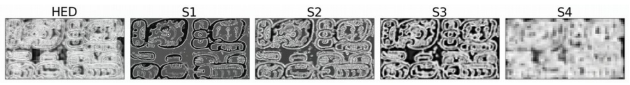


Figure 11: Tested using epoch10 checkpoint

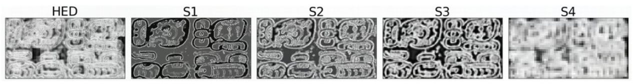


Figure 12: Tested using epoch14 checkpoint

Upon first glance, what was surprising was the absolutely horrendous edge mapping of the final fused output. The fact that this pixelation extends here might be more of a vestige of using epoch14 as starting point for all glyph training. This strategy assumed that set of weights that are near-optimal were converged upon at later checkpoints. This again, speaks more to an issue with the implementation of the class-balancing bias term that was discussed previously. Out of the side outputs produced (Note: s5 not displayed anywhere due to its unintelligibilty), the task generalized the best across the epochs in side output 2, with consistent clarity across th board. Aside from s2 (conv2 chunk) being the most generalized across the epochs trained upon, the most exciting part however, is this finding in the first output layer s1 corresponding to the conv1 chunk. Recall in the end of §3.1 that a edges are a low dimensional feature. This result is proof of concept for that claim. This s1 is roughly equivalent to an image derivative, and given its the first of such layers, it amounts to a first order method. The success of producing such clean and distinct dark bounds as non edges can effectively make this output layer the defining feature for MayaNet. This was explored in MayaNet's sister module MayaScribe for the course CS221 and serves as companion to this CNN based module. There, the fused HED was explored as a useful masking tool when paired with the saliency mapping. In the exploration of other HED output layers other than fuse, it might even serve as the best layer to use in combination with saliency for clean glyph extraction, as originally sought by this project

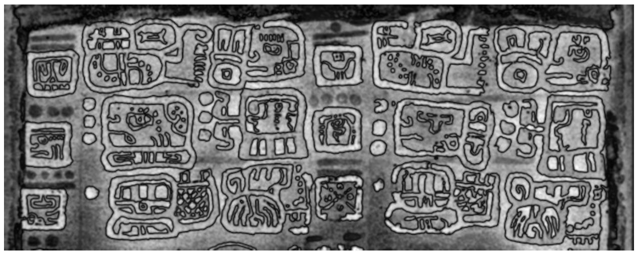


Figure 13: Saliency (Fine Grained) + Canny Edge Ground Truth

6.0 Future Works

Future explorations will include differences in strata placement, as well as the further annealment of the image fusing process to try and abate the loss of detail caused by the highest dimensional convolution blocks. Another similar approach could be to maintain two different networks, one with side-output, and another without, with the goal of having one produce a fused image as in the HED model, and have the other traditional CNN be bootstrapping of a pre-trained network, though one trained exclusively on edge/contour corpi, to provide a decent enough contrast to determine if the image classification pre-training is the main culprit behind the encountered loss of detail or if it is indeed the structuring of the implementation. Attempting a change in variable regarding the HED loss function is another avenue to explore post preliminary findings. BinaryCrossEntropy loss was the loss function used and discussed but another loss function that scales down the deeper convoultional side outputs that correspond to the highest dimensional features might make up for the bitmap like effect seen in the majority of the results. Structural Similarity is also another metric that could be used though might prove costly, computationaly speaking. While the original authors Deng, Shen, et al. only explored the binary case of edge versus non-edge, the modifications considered during my approach to a tertiary cased problem of edge/nonedge/fuzzy might be worth a full venture (see

§3.2)[3]. While the end focus of this project may not be to explore an efficient loss function for Crisp Edge Extraction, it produced enough results in their findings that it is absolutely relevant in the established goal of producing healthy, contiguous glyph inscriptions.

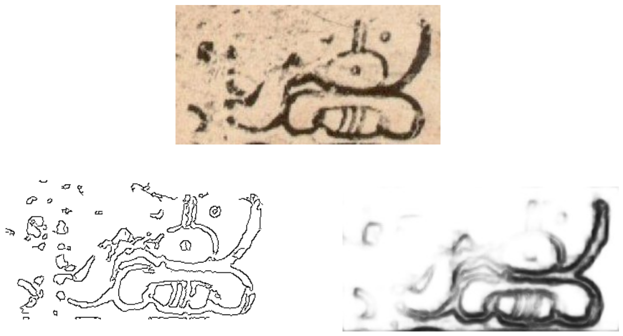


Figure 7: Single degraded glyph input (above) vs. Uncanny Ground Truth (lower left) and HED output (lower right) on First Train Output (epoch3) on Codex Dresdensis

Contributions:

HED Implementation was leveraged using @Buntyke's pyTorch implementation of the HED Operator
https://github.com/buntyke/pytorch-hed

Sobel Opertor implementation was modified from Nika Tsankashvili's Medium article on the comparison of Edge Detectors
https://medium.com/@nikatsanka/comparing-edge-detection-methods-638a2919476e

Dataset Preparation was possible thanks to SLUB https://digital.slub-dresden.de/data/kitodo/codedrm_280742827/codedrm_280742827_tif/jpegs/

MayaNet is hosted on my github

https://github.com/csalguer/MayaNet/upload/master

References

[1] Shen, Wei, et al. "DeepContour: A Deep Convolutional Feature Learned by Positive-Sharing Loss for Contour Detection." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, doi:10.1109/cvpr.2015.7299024.

[2] Xie, Saining, and Zhuowen Tu. "Holistically-Nested Edge Detection." *CoRR*, Apr. 2015, arxiv.org/abs/1504.06375. arXiv:1504.06375v2 [cs.CV]

[3] Deng, Ruoxi, et al. "Learning to Predict Crips Boundaries." *European Conference on Computer Vision EECV*, 26 July 2018, arxiv.org/pdf/1807.10097.pdf.

[4] D. Marr and E. Hildreth. Theory of edge detection.Proceedings of the Royal Society of London. Series B. BiologicalSciences, 207(1167):187–217, 1980.

[5] Reuter, M. and Biasotti, S. and Giorgi, D. and Patane, G. and Spagnuolo, M." (2009). "Discrete Laplace-Beltrami operators for shape analysis and segmentation". Computers & Graphics. 33 (3): 381–390df. CiteSeerX 10.1.1.157.757

[6] Dim, Jules R.; Takamura, Tamio (2013-12-11). "Alternative Approach for Satellite Cloud Classification: Edge Gradient Application

[7] J. Canny (1986) "A computational approach to edge detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 8, pages 679–714.

[8] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation.PAMI,33(5):898–916, 2011

[9] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Sta-tistical edge detection: Learning and evaluating edge cues.PAMI, 25(1):57–74, 2003.

[10] P. Dollár and C. L. Zitnick, "Structured Forests for Fast Edge Detection," *2013 IEEE International Conference on Computer Vision*, Sydney, NSW, 2013, pp. 1841-1848.

[11] *MAAYA*, www.idiap.ch/project/maaya/.

[12] Famsi. "Search the FAMSI Databases." *Search the FAMSI Databases*, research.famsi.org/.

[13] "Deciphering Maya Glyphs | Unravel Magazine." *Unravel*, unravellingmag.com/articles/deciphering-maya-glyphs/.