Analysis of patch aggregation methods in ink detection

Louis Schlessinger and Arefeh Sherafati

Introduction

Most 3D-to-2D ink detection methods that use the surface volumes utilize a grid-based sliding window inference to reconstruct a prediction image. Each window consists of a subvolume and its associated binary ink labels. This is used to produce an ink prediction of the same spatial region. The ink predictions are then aggregated by combining the possibly spatially overlapping patches to produce the full-size image.

Here, we use four different patch aggregation methods (averaging, Gaussian, cropping, and Hanning) and compare their resulting prediction images both qualitatively and quantitatively with respect to the ground truth ink labels on an example holdout fragment (fragment 4). We also show how the choice of patch aggregation method impacts the appearance and perhaps legibility of the predicted characters.

Methods

Models

We used two different models to produce the patch predictions. Model 1 is used for Figures 1, 2, and panel D of Figure 3. Model 2 is used for panel E of Figure 3, and Figure 4.

- **Model 1:**
    - Architecture: UNETR-SegFormer [1]
    - Window size: 512
    - Batch size: 6
- **Model 2:**
    - Architecture: UNETR-SegFormer
    - Window size: 128
    - Batch size: 96

Due to computational resource limitations, we increased the training batch size of model 2 to 96.

Data

Fragments 2 and 3 were used for training and fragment 1 was used for validation for both models. We use the fragment 4 surface volumes (54 keV, 3.24 μm) as the holdout set. We used the same inference code as used by the winning Kaggle ink detection model [3] and used slices 32 ± 8 for inference. Quantitative and qualitative comparisons were made possible through the use of the ink labels of this holdout fragment. Note that quantitative comparisons were not possible with scroll surface volumes as there is no exposed surface for a measure of ground truth.

Patch aggregation methods

We compare four different patch aggregation techniques. In particular, two classical signal processing window functions (Gaussian and Hann) and two types of patch overlap averaging (averaging and cropping). Each method except averaging is designed to emphasize the center of the patch more than its edges.

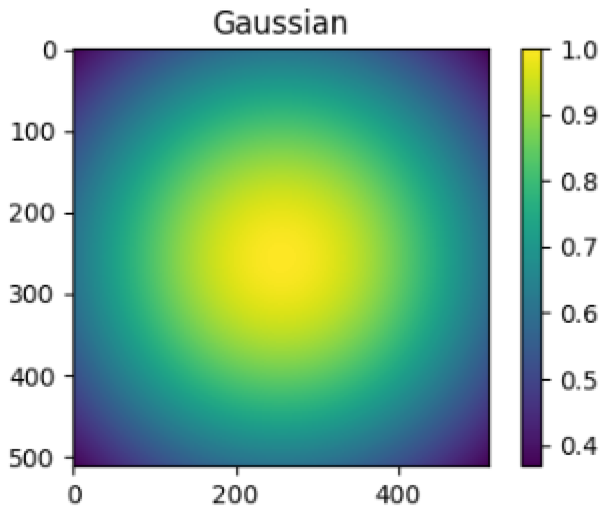
Each 2-dimensional window function is multiplied by each patch of ink probabilities and then summed at the absolute locations for each patch. Note that for the Gaussian and Hanning methods we didn’t handle the edge and corner cases of the full-size image in our implementation.

Averaging

Averaging is a straightforward method for patch aggregation, where each patch contributes equally to the final reconstructed image. Overlapping patches are weighted by an equal-weighted average window. That is, for each patch prediction, a uniform weight is applied, and the sum of these patches is normalized by the number of overlapping patches at each pixel. This method ensures that all parts of the image contribute equally, reducing bias towards any particular pixel.

Gaussian

In this method, a 2-dimensional Gaussian window is used to weight the patches during aggregation. This approach emphasizes the central part of each patch more than the edges, effectively reducing the impact of boundary artifacts and smoothing the transition between overlapping patches. In our case, we use the standard deviation of half the spatial window size.



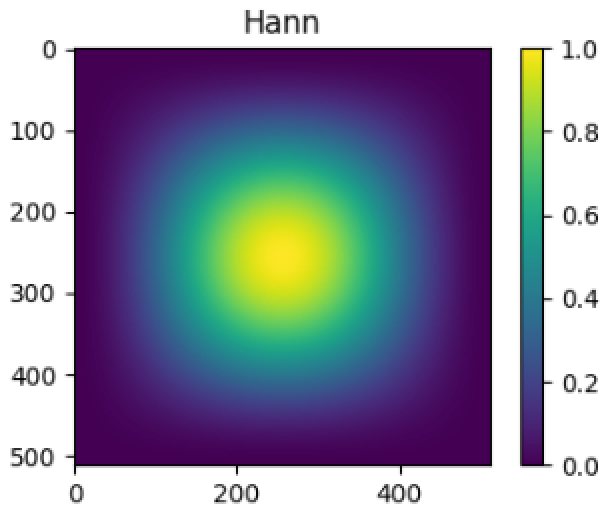
2D Gaussian window

Cropping

The cropping method involves center-cropping the overlapping patch predictions and then averaging. This method also reduces the impact of the window edges and corners. In our case, we crop one-fourth of the image from each side of the patch.

Hanning

In this method, a 2-dimensional Hann window is used to weight the patches during aggregation. The Hann window, like the Gaussian kernel, emphasizes the center of each patch with a sharper but smoother drop at the edges compared to the Gaussian kernel. It is a taper formed by using a raised cosine.



2D Hann window

Reproducibility

You can reproduce the figures and tables by running the notebooks in the following order:

1. generate and save patch aggregation
2. qualitative patch aggregation analysis
3. quantitative patch aggregation analysis

Results

We first compared the effects of these four patch aggregation methods on the patch predictions produced using model 1 (window size = 512). Our results are shown in Figure 1 for the entirety of fragment 4 (Figure 1A) and two example letters from the first row of the same fragment (Figure 1B-C) using a fixed stride = window size / 4 = 128.

The effects of different patch aggregation methods

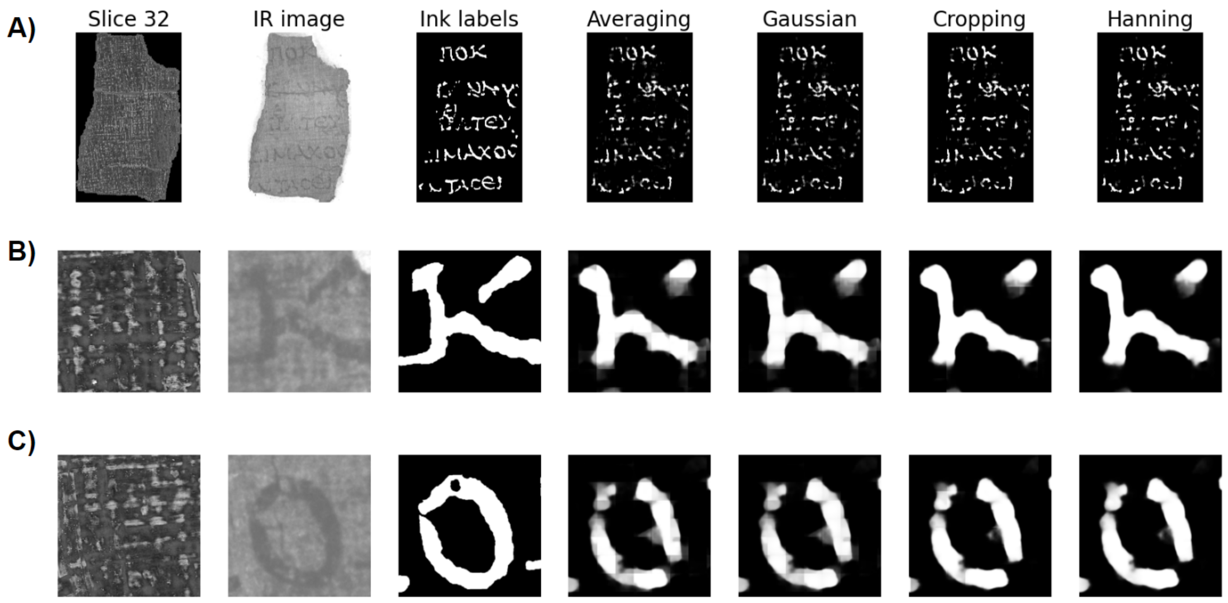


Figure 1 | The effects of different patch aggregation methods on the same patch predictions made using model 1 (window size = 512) on fragment 4. From left-to-right for each column are shown: slice 32 of the TIF image, IR image, ink labels, and following those four different patch aggregation methods, namely, averaging, Gaussian, cropping, and Hanning methods all made using a stride = window size / 4 = 128. A) The predictions on all of fragment 4. B) Example letter kappa from the first row. C) Example letter omicron from the first row.

The effects of stride on patch aggregation methods

We then evaluated the effects of different strides in the patch aggregation inference step using the prediction images produced using model 1 (window size = 512). Our results are shown in Figure 2 for the letter kappa from row 1 in fragment 4 for three different strides, namely, window size / 2 = 256, window size / 4 = 128 (same as Figure 1), and window size / 8 = 64. The slice 32 TIF input image, the IR image, and ink labels associated with the same region are added for better visual comparison. Figure 2D shows the baseline prediction image which simply tiles the patches, which is equivalent to window size = stride = 512.

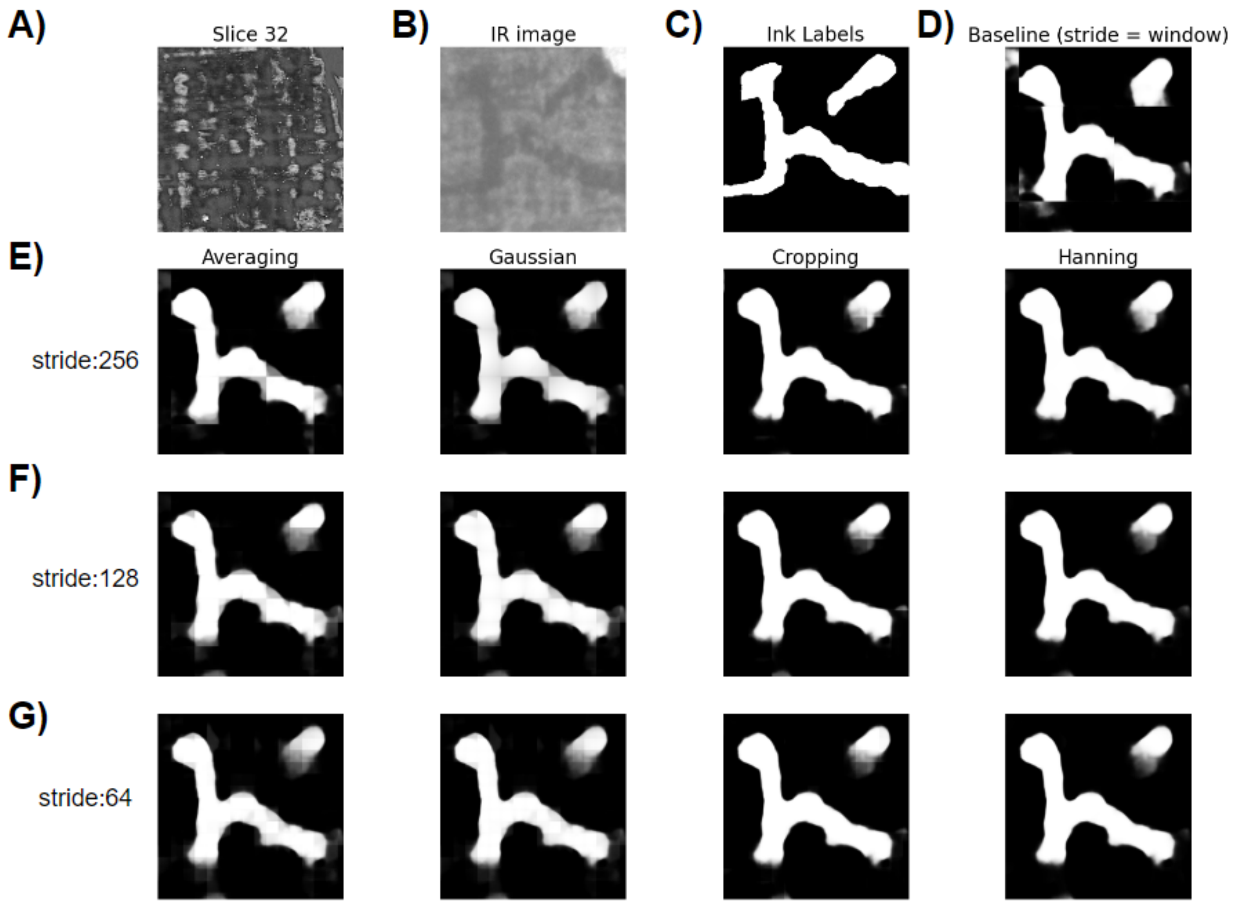


Figure 2 | The effects of different strides on the patch aggregation method are shown for the letter kappa from fragment 4 with the same model as Figure 1 (window size 512). A) slice 32 of the TIF image. B) IR image. C) ink labels. D) baseline prediction image made by simply tiling the patches of size 512 with no overlap (stride = window size). Prediction images are shown for four different patch aggregation methods, namely, averaging, Gaussian, cropping, and Hanning using E) stride = window size / 2 = 256. F) stride = window size / 4 = 128 (same as figure 1). G) stride = window size / 8 = 64.

The effects of model window size on patch aggregation methods

Patch aggregation methods demonstrate different effects when applied to models using different window sizes. Figure 3 shows the prediction images for the same letter kappa selected from the first row of fragment 4, using two different models, one with a window size of 512 (Figure 3D) and another with a window size of 128 (Figure 3E).

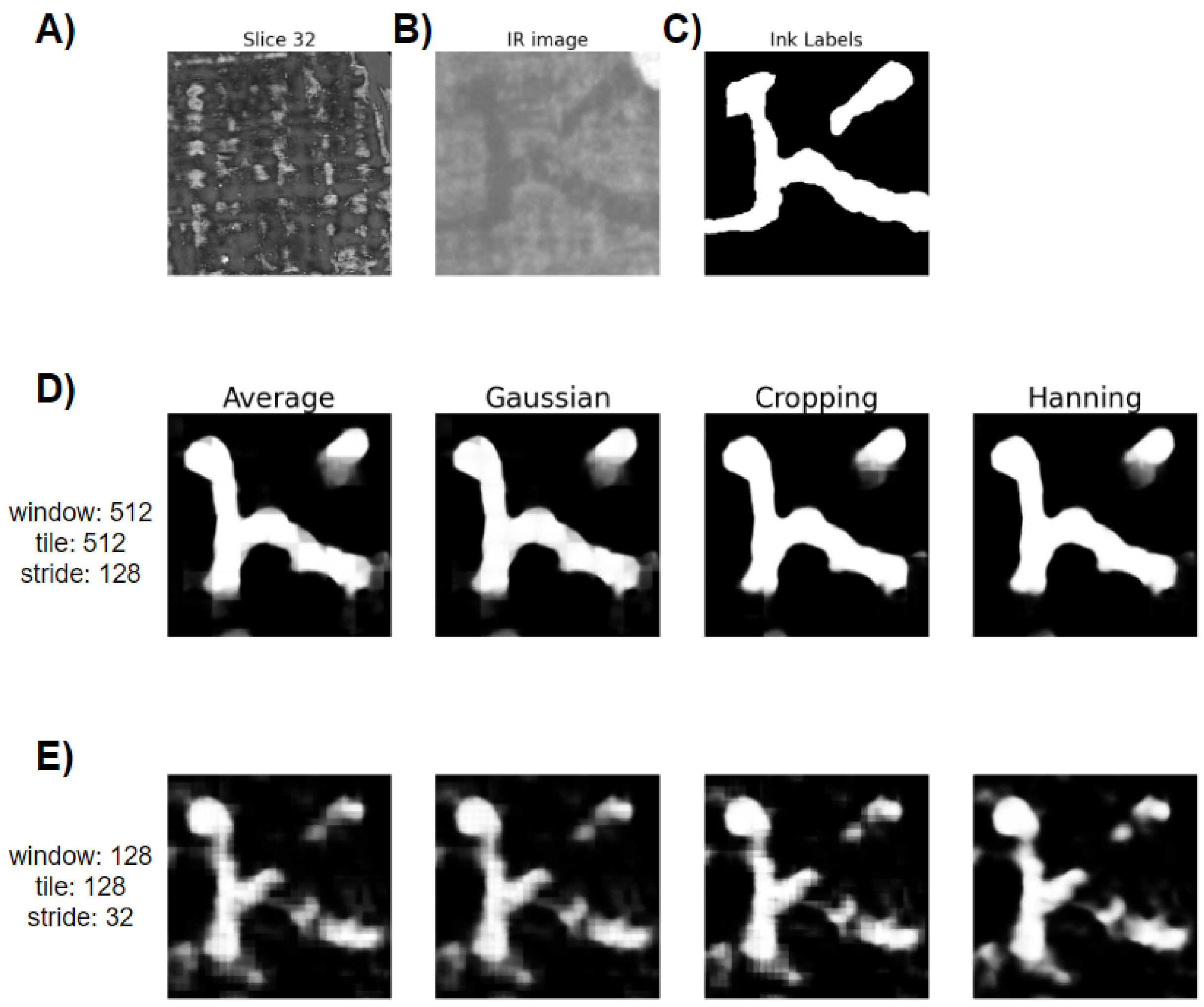


Figure 3 | The effects of model window size on patch aggregation methods. A) slice 32 TIF image. B) IR image. C) ink labels. Prediction images are shown for four different patch aggregation methods, namely, averaging, Gaussian, cropping, and Hanning using stride = window size / 4 for D) model 1 with window size = 512 (same model as Figure 1), E) model 2 window size = 128.

The effects of smaller model window size on patch aggregation methods with respect to stride

We replicated Figure 2 but with model 2, which uses a smaller window size of 128. Figure 4 shows the effect of different patch aggregation methods using a smaller window size with respect to stride.

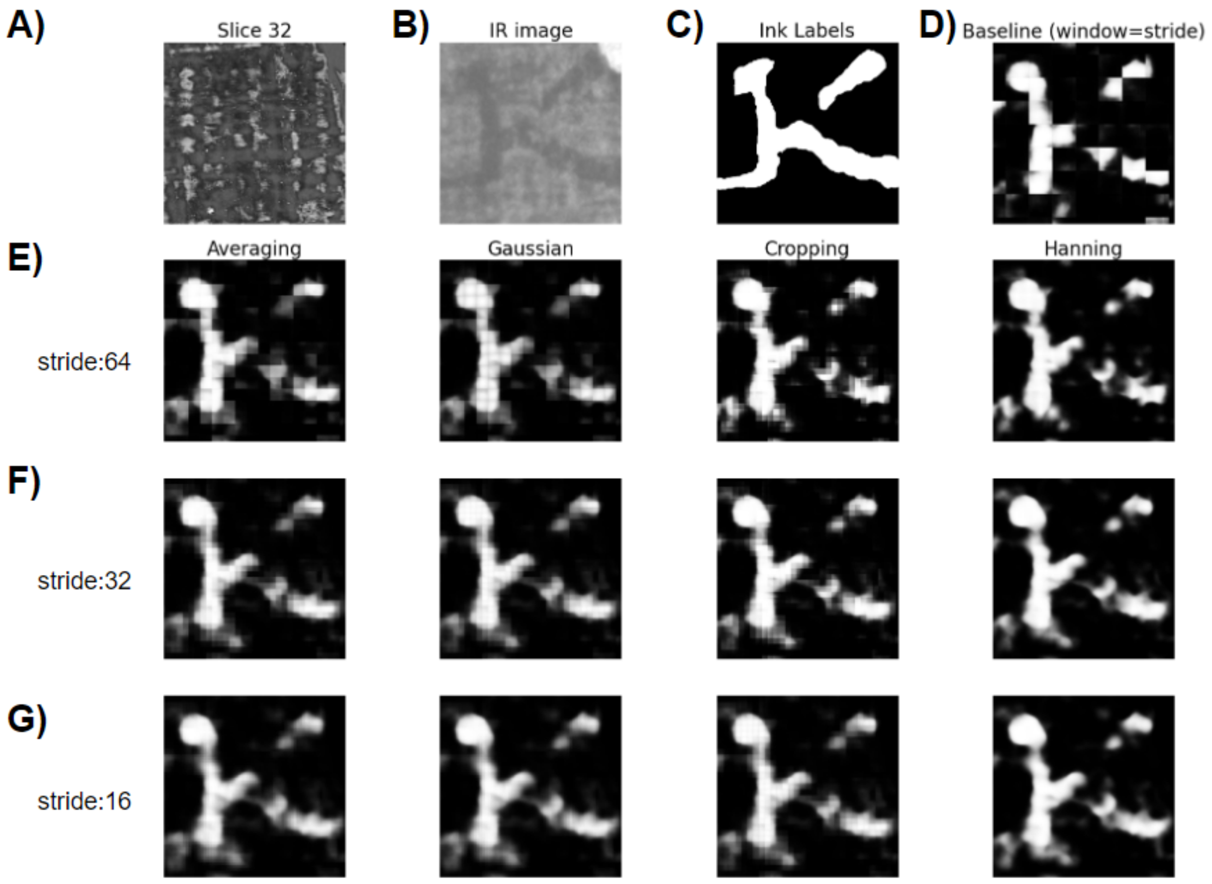


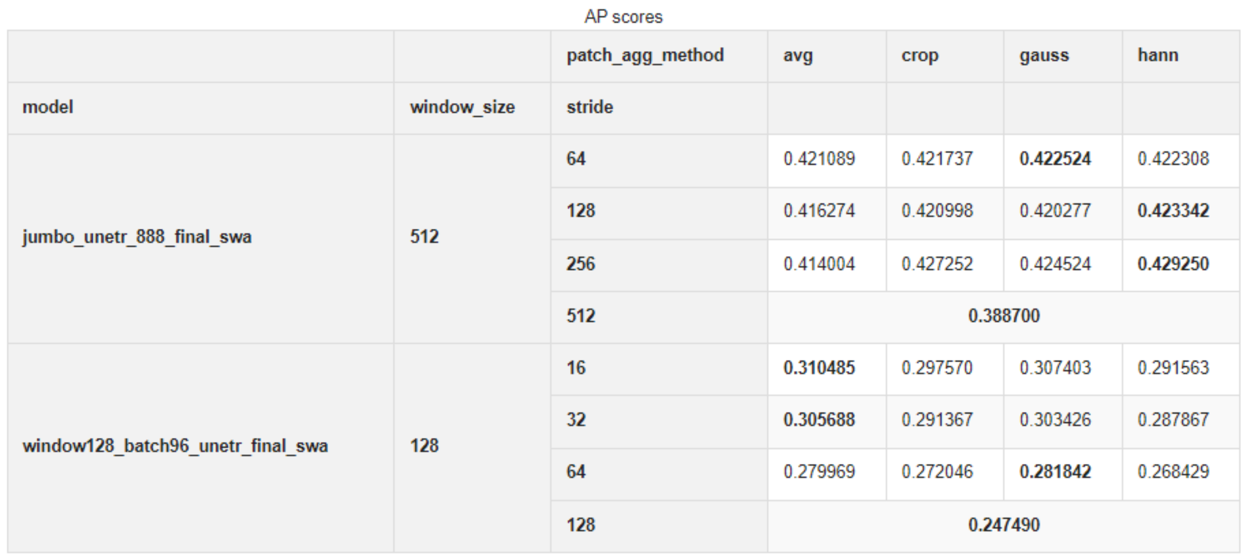
Figure 4 | This is a replication of Figure 2 with the difference that patch predictions were made using a model with a smaller window size (UNETR-SegFormer, window size 128) showing the effects of different strides on the patch aggregation methods. A) slice 32 of the TIF image. B) IR image. C) ink labels. D) baseline prediction image made by tiling the patches of size 128 with no overlap (stride = window size). Prediction images are shown for four different patch aggregation methods, namely, averaging, Gaussian, cropping, and Hanning using E) stride = window size / 2 = 64. F) stride = window size / 4 = 32. G) stride = window size / 8 = 16.

Quantitative analysis

Besides the qualitative analysis that is presented in the figures above, we used two performance measures to make quantitative comparisons between the prediction images and the ground truth ink labels. Please note that in this preliminary analysis, we only calculated these performance measures using full-size prediction images of the holdout fragment.

Average precision

Average precision (AP) is a widely used performance measure in object detection and image segmentation tasks. It combines the precision and recall values at various threshold levels to provide a single performance measure. Precision is the ratio of true positive predictions to the sum of true positive and false positive predictions, while recall is the ratio of true positive predictions to the sum of true positive and false negative predictions. AP calculates the area under the precision-recall curve, offering a comprehensive evaluation of the model's ability to correctly identify the ink labels while minimizing false detections.



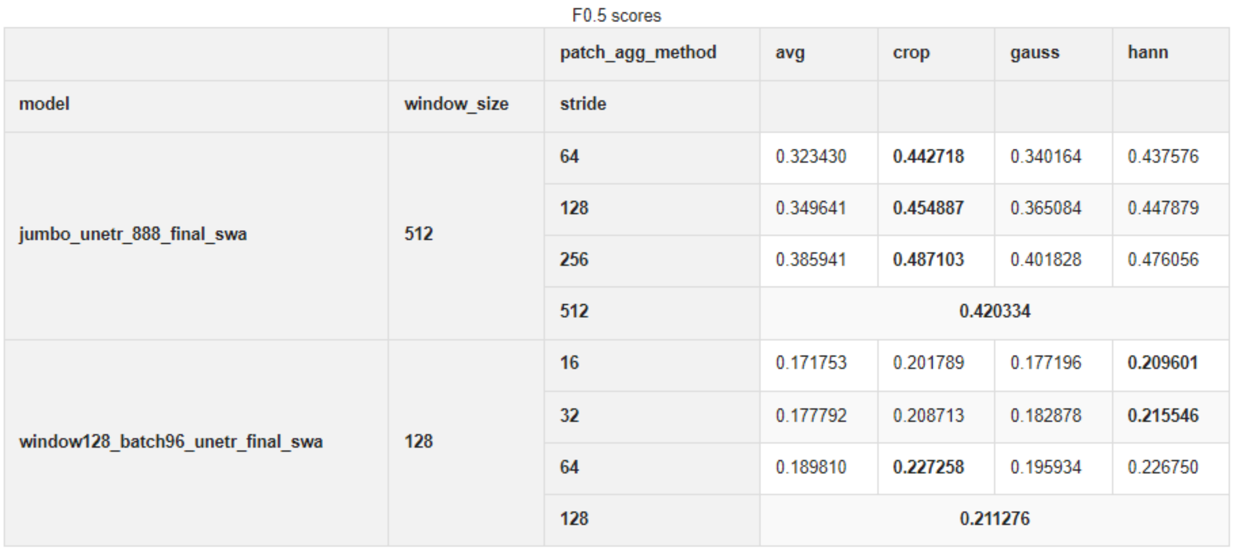
AP scores

		patch_agg_method	avg	crop	gauss	hann
model	**window_size**	**stride**				
jumbo_unetr_888_final_swa	**512**	**64**	0.421089	0.421737	**0.422524**	0.422308
		128	0.416274	0.420998	0.420277	**0.423342**
		256	0.414004	0.427252	0.424524	**0.429250**
		512	**0.388700**			
window128_batch96_unetr_final_swa	**128**	**16**	**0.310485**	0.297570	0.307403	0.291563
		32	**0.305688**	0.291367	0.303426	0.287867
		64	0.279969	0.272046	**0.281842**	0.268429
		128	**0.247490**			

Table 1 | AP scores for two models with a window size of 512 and window size of 128 and four aggregation methods calculated at strides equal to window size, window size/2, window size/4, and window size/8. Bold numbers show the best score for each row. Model 1 is “jumbo_unetr_888_final_swa” and model 2 is “window128_batch96_unetr_final_swa”.

$F_{0.5}$ score

The $F_{0.5}$ score is a variation of the F_β score that gives more weight to precision than recall. It is defined as the harmonic mean of precision and recall, with precision being more influential in the calculation. The $F_{0.5}$ score is particularly useful in applications where false positives are more critical than false negatives (as it was emphasized by the Vesuvius Challenge Kaggle Ink Detection Competition). By assigning a higher weight to precision, the $F_{0.5}$ score helps in evaluating the model's performance in scenarios where correctly predicting the presence of ink is more important than missing some actual ink.



F0.5 scores

		patch_agg_method	avg	crop	gauss	hann
model	window_size	stride				
jumbo_unetr_888_final_swa	512	64	0.323430	**0.442718**	0.340164	0.437576
		128	0.349641	**0.454887**	0.365084	0.447879
		256	0.385941	**0.487103**	0.401828	0.476056
		512	**0.420334**			
window128_batch96_unetr_final_swa	128	16	0.171753	0.201789	0.177196	**0.209601**
		32	0.177792	0.208713	0.182878	**0.215546**
		64	0.189810	**0.227258**	0.195934	0.226750
		128	**0.211276**			

Table 2 | $F_{0.5}$ scores for two models with a window size of 512 and window size of 128 and four aggregation methods calculated at strides equal to window size, window size/2, window size/4, and window size/8. Bold numbers show the best score for each row. Model 1 is “jumbo_unetr_888_final_swa” and model 2 is “window128_batch96_unetr_final_swa”.

Discussion

Here, we compare the prediction images obtained using four different patch aggregation methods. We make this comparison both qualitatively and quantitatively using two performance measures (AP and $F_{0.5}$).

In general, we observe striking differences in the boundaries of characters across different patch aggregation methods. Further, grid artifacts are more evident in images produced using Gaussian and averaging methods and less evident in images produced using Hanning and cropping methods. We also observed that smaller window sizes result in more grid artifacts across all patch aggregation methods, except for the Hanning method, which shows the least grid artifacts at smaller window sizes (Figure 3, Figure 4).

Similarly, we found that images produced using Hanning and cropping methods look more similar across many strides. However, images produced using Gaussian and averaging methods improve with smaller strides and show fewer false positives. For example, at stride = window size / 8, the prediction images from all four patch aggregation methods look the most similar to each other (Figure 3E, Figure 4G).

Interestingly, like the qualitative observations, we found that AP scores for smaller windows seem to be grouped into two clusters across strides. Hanning and cropping in the first and Gaussian and averaging in the second. This grouping also can be seen in $F_{0.5}$ scores for both window sizes.

In general, our quantitative results show that AP scores have less variance for larger windows across strides for any given patch aggregation method. This trend is also in agreement with qualitative observations.

Additionally, we found that both quantitative measures, AP and $F_{0.5}$ scores, were higher for larger window sizes across all patch aggregation methods. This result is in agreement with the reports from the contestants of the Vesuvius Challenge Kaggle Ink Detection Competition that $F_{0.5}$ scores were higher for larger window sizes. However, despite these higher quantitative scores, we think that: 1) these scores are not perfectly correlated with legibility, and 2) as noted by papyrologists, models with smaller window sizes tend to have fewer stroke-like hallucinations (i.e., false positives around character boundaries). Thus, these scores should be interpreted with caution depending on the goal.

Furthermore, we expect that lower strides yield more accurate results with diminishing returns at an increasing computational cost. This occurs because pixels that are more centrally located within a patch window tend to have more contextual information. However, we found that across all patch aggregation methods, AP scores were higher for lower strides, but $F_{0.5}$ scores were not. This also underscores the importance of interpreting the scores with caution.

Limitations

Although we trained the same model as the one with a window size of 512 using a smaller window size of 128, we should note that these two models were not trained with the same batch size due to limitations in computational resources. However, we do not think that this affects features such as grid artifacts and character boundaries, but this might have affected $F_{0.5}$ and AP measures.

Additionally, $F_{0.5}$ score is threshold dependent, and we did not tune its threshold here. Thus, $F_{0.5}$ scores need to be carefully interpreted.

Conclusions

In summary, we analyzed the prediction images of four patch aggregation methods. We explored the effects of model window size and inference stride parameters for each method by comparisons to the ground truth fragment ink labels. We quantified our observations using two performance measures ($F_{0.5}$ and AP), but more work needs to be done to find measures that correlate better with legibility as assessed by papyrologists. We hope that sharing our analysis and code highlights the importance of the choice of image reconstruction method in patch-based pipelines and its implications for the legibility of the characters.

References

1. jumbo_unetr_unetr_888_final_swa_all_train_long.pth
2. window128_batch96_unetr_final_swa.pth (available upon request)
3. Vesuvius-InkDetection/inference_notebook.ipynb
4. Introducing Hann windows for reducing edge-effects in patch-based image segmentation
5. Patch-based pipelines
6. Monai sliding window inference
7. Patch-based analysis
8. Quantifying Efficiency of Sliding-Window Based Aggregation Technique by Using Predictive Modeling on Landform Attributes Derived from DEM and NDVI