October Report

This report will present the advances made since the publication of the previous one. The aim of the previous report was to propose an approach to extend ink detection to other scrolls, mainly by (i) **pretraining** an encoder on large amounts of **unlabeled** segment data and then (ii) leveraging the **learned representations** to detect initial traces of ink so that they can be used as the starting point of an iterative labeling process similar to the one used in the Grand Prize.

Therefore, the aim of this month has been on **finding a way to evaluate the usefulness of the learned representations in the task of ink detection.** This has been done by:

- Implementing a supervised baseline so that the quality of the learned representations can be compared.
- Writing code to automatically download segments, pretrain an encoder and fine-tune models to perform ink detection.
- Performing an initial set of experiments to evaluate the impact of pretraining on ink prediction.

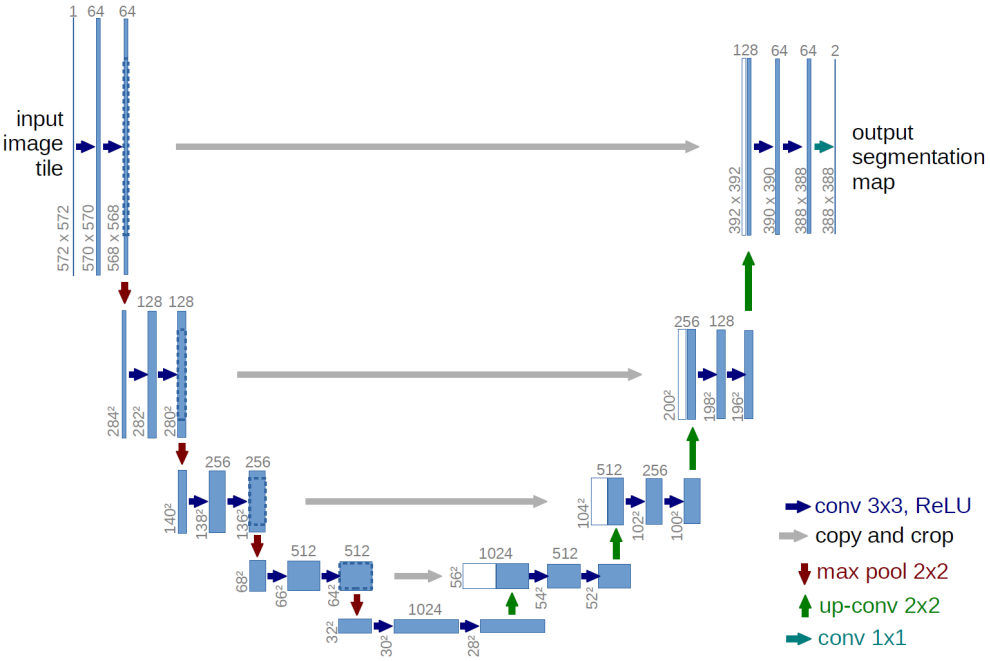
How to evaluate the pre-trained encoder

As mentioned in the previous report, the main idea was to pretrain a ResNet-50 encoder via a Self-Supervised Training (SSL) procedure named SparK. The idea of this procedure is to obscure large patches of the initial image (here, a crop of the segment) and train the encoder to reconstruct it. With sufficient data and training, the encoder has been shown to learn useful features that can be used on many different specific tasks, even without fine-tuning.

Now, we require a way to evaluate the pretrained encoder on the task of ink detection, as well as compare it to a supervised baseline.

Baseline Model

The UNet is a popular model for image segmentation tasks. Specifically, it is a really popular option for ink detection. Its name comes from the U-shape that its diagram has:



Roughly, the UNet is composed by an **encoder**, which captures context and reduces spatial dimensions while increasing the number of feature channels, and a **decoder**, which takes this features, and restores the spatial dimensions, predicting a segmentation mask.

Hence, the idea here is to adapt the UNet architecture so that a pretrained ResNet-50 can be integrated. Therefore, the UNet will be modified so that the **encoder is a ResNet-50 without fully connected layers**. We will call this architecture **ResNet-UNet** (e.g. UNet with ResNet backbone/encoder)

Experiments

This baseline model will enable us to evaluate the quality of the representations learned by the pretrained encoder by replacing the weights of the ResNet-UNet by the ones obtained during pretraining. If the representations are useful, we expect the performance of the model to be improved if we use the pretrained encoder instead of the randomly initialized.

Thus, the following experiments will be performed:

1. **Randomly initialize the ResNet-UNet, freeze the weights of the encoder and train only the decoder**. This will be our baseline and we

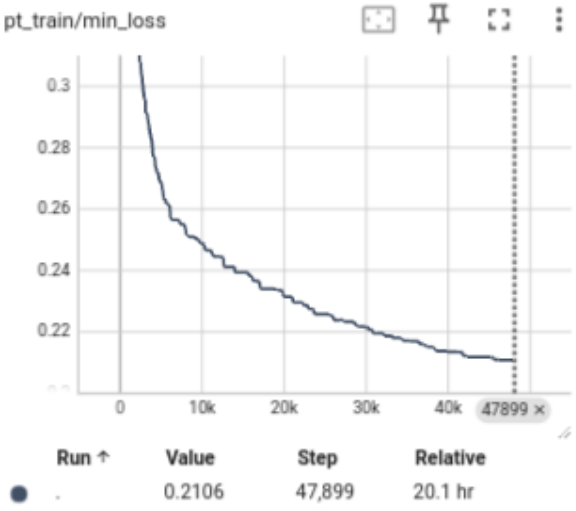
expect to obtain really poor performance.

2. **Load the pretrained encoder, freeze it and only train the decoder.** We can think about this as the decoder taking the learned representations during pretraining and using them to learn to detect ink. If the performance is better than the previous experiment, it implies that the representations learned during pretraining are indeed relevant.
3. **Randomly initialize the model, train the whole model:** This would be the standard procedure to perform ink detection, i.e. normal supervised training.
4. **Load the pretrained encoder, train the whole model:** The idea here is to also evaluate the representations, but enable the encoder to fine-tune them to the specific task of ink detection. It is expected to obtain similar or better performance than (3)

Again, it is important to remark that the aim of these experiments is to enable us to evaluate and compare the representations of the encoder vs. a supervised baseline, but the final objective of the approach is to be able to obtain a sufficiently good encoder that is able to detect traces of ink in an unsupervised manner. This evaluation scheme will help us in the evaluation of such pretrained encoder so that it can be improved.

Results

The experiments were performed in a single, relatively small segment, with ID `20230827161847`. The ResNet-50 encoder was pretrained for ~20 hours on a RTX 4090:

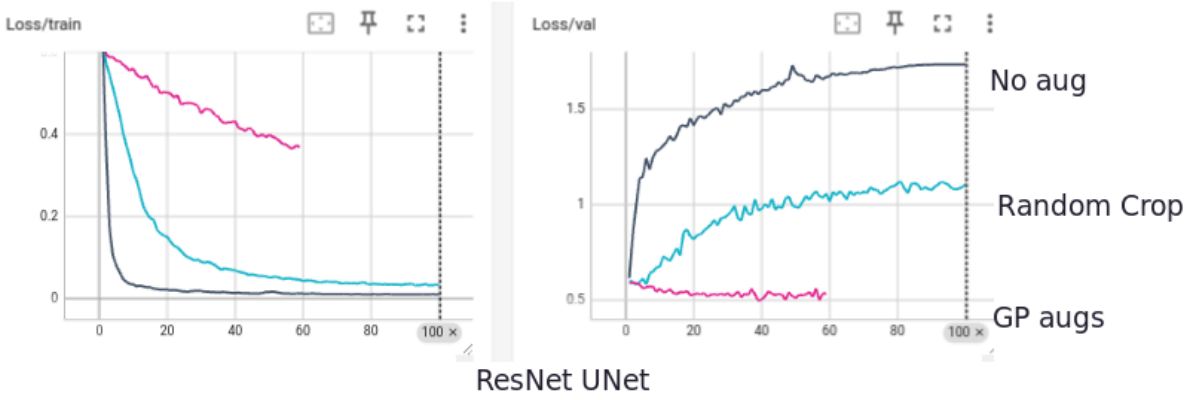


Notice that the encoder could have been trained for longer as it still had a

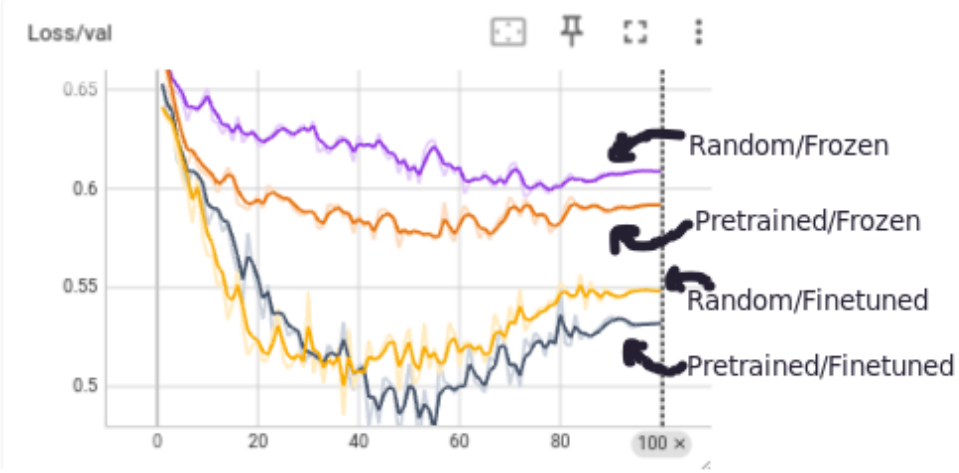
small margin of improvement, but it will be enough for our purpose, which is to show how to evaluate the pretrained encoder.

The ResNet-UNet will also be trained on the same segment. As previously-mentioned, it will be trained in a supervised manner. The details are as follows:

- The segment will be divided in crops of size 20x320x320 (i.e. we only take the 20th centermost layers), and **only the segments containing at least 5% of ink are taken**. This is done to provide the model with a balanced amount of ink/no ink regions. Then, the 320x320 crop will be randomly cropped and resized to 224x224 resolution.
- The first 90% of the previous crops are used for training, whereas the remaining 10% is used for validation purposes.
- The model is trained for 100 epochs
- The augmentations used in the Grand Prize model will also be applied. As shown in the following picture, data augmentation is key, specially when working with small segments.



- The loss will be a 50/50 combination of Binary Cross Entropy (BCE) and DICE.



The previous image shows the validation loss obtained in each training run for each experiment. Here, a lower loss implies better results.

- As expected, the worst results are obtained in the random/frozen experiment. Here, the decoder has to learn from random representations/features given by the frozen encoder.
- Loading the pretrained model and freezing it yields better results. This step was **crucial** to show that the features learned by the encoder are indeed useful to perform ink prediction.
- Randomly initializing the UNet and training the whole model from scratch gives considerable better results than the previous two runs.
- Again, loading the pretraining encoder and training the whole model boosts the performance when comparing to the previous experiment.

In summary, the results show that the features learned by the encoder are relevant for ink prediction. One might think that the performance boost might not be enough, but it is important to remark that these experiments have been performed on a single small segment (only ~3000 crops were used in training). SSL procedures are trained in the order of **millions of images** and have been shown to provide better results as the amount of training data increases, so we expect this performance boost to be larger as the approach is scaled up to more segments.

We can also visualize the ink predictions in the different experiments:

Again, it is important to remark that the model has only been trained on ~3000 images, therefore the results are expected to not be ideal. Despite this, the improvement in the predictions can clearly be seen when using the pretrained encoder vs. the randomly initialized one.

Conclusions

The previous report set out the motivation and design choices for the unsupervised ink detection method, as well as a roadmap of experiments to iteratively build and extend it. Following this, efforts this month have been devoted to establishing a way to evaluate the quality of the representations learned with the SSL procedure. For this, a modified UNet architecture with a ResNet encoder is used, so that the encoder weights can be replaced for those obtained during SSL.

The results show that using a pretrained encoder clearly boosts the performance, both when the encoder is frozen and unfrozen. These experiments have been performed in a single, relatively small segment (training was performed with ~3000 crops or images) but the results are promising and are expected to improve as the approach scales up. Therefore, the natural next step is to scale pretraining and fine-tuning to more segments.