Vesuvius Challenge using RepMode

Nicola Debole, Mateo Rodriguez

November 2023

1 Dataset

One of the main problem with this challenge is that the data is very sparse and there are few labeled examples. It is very hard to make predictions in the wild. In addition, the few labeled samples we have are **very heavy**: the training data is around 60 GB (and for reference consists only in 4 small fragments that contains in total around 90 letters). Our hardware has 16 GB of ram and 12 of VRAM which is not even close to be able to load a reasonable amount of samples in a batch (we were limited to 7). One full epoch was approximately one year long on our machine, so we had to do something about the dataset. After many iterations and experiments, the final recipe that gave us very good results was the following.

1.1 Edges

First, we gave priority to learning the boundary between ink and no ink. So we used a canny filter to detected the edges of the labels and extracted all the points on the border of the letters. This procedure reduce drastically the number of samples while still mantaining the ability to train the network. Still, the data was biased in a lot of ways since, for example, the center point would always be ink. So we added a random translation to counter some of the biases.

1.2 Buckets

Another bias present was the ratio between black (no ink) and white (ink) pixels in a given subset of the data. So what we did is to create 10 *buckets* that contains samples with similar ratios. For example, the first bucket will contain samples that have 90% black, the second 80%, and so on. Each bucket can be filled with maximum 1000 samples. In this way we also make a balanced dataset and we have control on the size (number of buckets and the size of each one are adjustable).

1.3 Void

Since with the above procedure there were still not that many completely black samples (without ink) we decided to add an arbitrary amount of totally black samples to let the model learn also how a black area is.

1.4 Final dataset

So the final dataset used was build using this recipe, with 24000 samples from the buckets

- 4000 samples from the first fragment
- 4000 samples from the second fragment (1st part)
- 6000 samples from the second fragment (2nd part)
- 6000 samples from the third fragment
- 4000 samples from the fourth fragment

and 12000 random samples from the areas without ink. In total, 36000 samples. This took 7hrs for a full epoch.

2 Tensor operations

Matrix multiplication using torch.einsum

2.1 Standard matrix multiplication

given two matrixes A,B of dimension $n \times k$ and $k \times m$ the product is the matrix P whose elements:

$$p_{i,j} = \sum_{l=1}^{k} a_{i,l} \cdot b_{l,j} \tag{1}$$

2.2 Einsum matrix multiplication

The above operation can be obtained by using:

$$einsum("il, lj \rightarrow ij", A, B) \tag{2}$$

2.3 Experts

2.4 Routing

In order to gate the different experts

$$torch.einsum("oidhw, o \rightarrow oidhw", expert\_conv5x5, g[n, 0, :]) \tag{3}$$

is equivalent to the element wise multiplication of the vector g with the first dimension of the tensor "expert_conv5x5" An example with:

- Tensor E of dimension $[3,2]$
$$\begin{bmatrix} 2 & 8 \\ 6 & 4 \\ 3 & 5 \end{bmatrix}$$

- Tensor G of dimension [3] [1,2,3]

The operation torch.einsum("ok,o$\rightarrow$ ok", E, G) is

$$\begin{bmatrix} 2 \cdot 1 & 8 \cdot 1 \\ 6 \cdot 2 & 4 \cdot 2 \\ 3 \cdot 3 & 5 \cdot 3 \end{bmatrix}$$

3 3D to 2D compression

To extract a 2D image from the 3D output of the original RepMoDE network we tested various approaches:

- Averaging Z-levels:

  This approach involved taking an average across all Z-levels to create a 2D representation, effectively collapsing the 3D volume into a 2D representation. While straightforward, this method seemed to overlook patterns in different Z-levels.

- 2D Convolution with Z as Channels:

  We then experimented using 2D convolution with the Z-dimension treated as channels. This technique allows the model to learn spatial hierarchies along each Z-level independently. By employing 2D convolutions, the model can capture complex features across spatial dimensions while considering information from different Z-levels. This approach offers more flexibility and capacity to extract intricate patterns compared to simple averaging. This method is the one that gave the best and most stable results on all our trainings

- Matmul for Tensor Product:

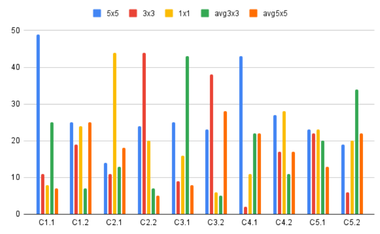
  Another attempt involved using matrix multiplication (matmul) to perform a tensor product between a 3D tensor and a 1D tensor along the Z-axis. There are 2 main problems with this application, first one is the need to apply the operation to each individual sample in the batch by looping, creating a bottleneck. The second problem is how to obtain the 1D tensor to be used in the operation. We attempted extracting it from the input by a series of reshapes and down-sampling, or flattening the bottleneck and using that as the 1D tensor. Regardless of the source of the 1D tensor, this method didn't seem to perform as well as the simpler 2D convolution.

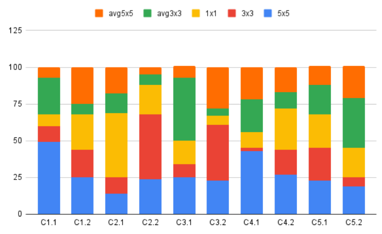
- 2D decoder:

  The final modification attempted was removing the 3D decoder and adopting a 2D one. This involved transforming the bottleneck and skip connections from a 3D representation to a 2D representation using the 2D convolutional method explained before, so that the data is directly decoded as a 2D image. This modification enhanced computational efficiency and retained the model's performance.

4 Experts analysis

After training the model we're able to load the weights and look at the gate value for each of the experts, meaning, we can see which experts the model considers more or less relevant for the task at hand.
The following figures were taken from the model with a 2D decoder, so the expert values are only from the side of the encoder where before each down sampling the data passes through 2 layers of experts:





The figures above show that in this particular model all the experts are being more or less used the same amount. The least used one is the average pool 5x5 expert with an average of 16.5%, while the most used is the 5x5 convolution expert coming at 27.2%.