Denoising Images by Deep Learning Models

The architecture and Design Choices

The implemented architecture in this project is combined by several components to achieve the goal of de-noising images from MNIST dataset.

Firstly, forward diffusion model played a role on progressively adding Gaussian noise over multiple time steps. The noisy image (x_t) at time step t is calculated by the closed form formula:

$$x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$$

where x_0 is the original clean image and ϵ is the Gaussian noise. The signal scaling factor ($\sqrt{\bar{\alpha}_t}$) and noise scaling factor ($\sqrt{1 - \bar{\alpha}_t}$) are defining how much image is preserved and how much noise is added respectively. The forward diffusion process allows the model to learn how the image became noisy in order to learn the way to de-noise reversely. 100 time steps are used for considerable denoising performance without heavy loading.

After adding noise by forward process, we need reverse process for denoising. By considering the formula above, if we want to get the de-noised image (closely the original image), we need to get the noise. To achieve this, a simple U-Net architecture is used for noise prediction. In the U-Net model, there is encoder with 2 downsampling layers to reduce the resolution with double convolutional layers to increase number of channels which increases feature depth. In the bottleneck, transformer layer is added to refine features. It is followed by the decoder which contains 2 upsampling layers to restore the original resolution. Skip connections is involved to merge encoder features with decoder features to recover fine details. With the final convolutional layer, the output return predicted noise which is useful for reconstructing de-noised image in the testing process. U-Net is chosen because of its strength on image to image handling. The number of 2 downsampling and upsampling layers are used to keep the balance between quality and efficiency which is enough to capture the features and also minimize training speed.

As mentioned above, transformer is included in the bottleneck of U-Net for feature refinement. The transformer is formed by several layers. The first one is the positional encoding which provides positional information to the transformer to tell where the features are located. Then, the core part of transformer consists of multi-head attention and position-wise feedforward layer. Multi-head attention splits input data into smaller vectors and each of them learns different patterns. The final concatenated heads can capture diverse contexts. It helps model learning in long-range dependencies, parallelisation and interpretability. Besides, position-wise feedforward used non-linear activation which output context-rich representations of the input. 2 layers of transformer layers are applied in this model to keep the model light-weighting.

In the training loop, batches of clean images from MNIST dataset are input and became noisy images using forward diffusion. At the same time, the real noise is returned from the diffusion process as well. Putting the noisy images into the designed model (U-Net + transformer), we can get the predicted noise. By comparing the real noise and the predicted noise, the MSE loss is calculated. After that, Back propagation is involved to

update optimiser parameter to improve the model training. 50 epochs is chosen to balance speed and performance which is meaningful enough but not overloading.

On the other hand, in the testing loop, we generate noisy images by diffusion with only half of the full time steps. It means these images with moderate level of noise only which is appropriate level for testing the denoising ability. Again we input the noisy images for model to do noise prediction. Now, there are noisy images and predicted noise, the de-noised images can be calculated by the reverse of diffusion formula. For each testing, matrices mean squared error MSE and structural similarity index measure SSIM are calculated for better evaluation.

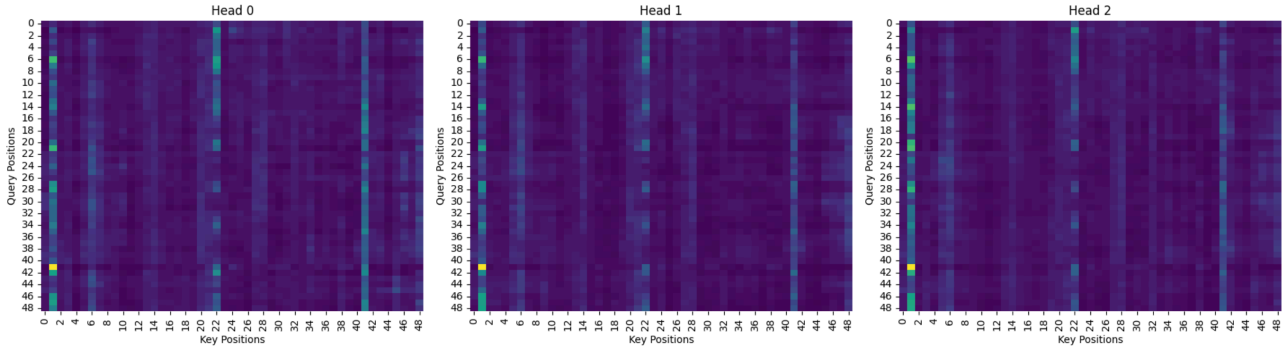
Training and Evaluation Results

Here is a summarizing table on the training and evaluation results:

Epoch	MSE	SSIM	Description
1	0.0067	0.7634	Training just started
2-22	0.0058 - 0.0069	0.7568 - 0.8039	Fluctuation
23	0.0057	0.8083	The highest SSIM is reached and MSE is satisfactory
24-40	0.0055 - 0.0061	0.7825 - 0.7925	Smaller fluctuation which is relatively stable with improvement
41	0.0053	0.8062	The lowest MSE is reached
42-49	0.0056 - 0.0060	0.7661 - 0.7989	Maintain good and stable performance
50	0.0058	0.7760	Training ended

From the above table, we can see the model performance is gradually enhancing throughout 50 epochs training. The similarity index measure matrix reached the peak at epoch 23 while the mean squared error matrix is the lowest at epoch 41. Comparing other range of epochs, i.e. epochs 2 - 22, epochs 24 - 40 and epochs 42 - 49, we can see the range of MSE is decreasing and around 0.0056 - 0.0060 finally. This indicates the model is trained to output more accurate noise prediction. On the other hands, the SSIM range is increasing and around 0.7661 - 0.7989 finally. This depicted the de-noised image is nearly 80% similar to the original image. The figures proved that the training is useful for model improvement on de-noising capability. However, from the above result, we can see the evaluation matrices became quite stable and satisfactory after epoch 23. If we concern the training time and cost (e.g. GPU is needed for running), we can consider to minimize the number of epochs for training at around 20 - 25. It is also an acceptable but optional approach.

Insights from Attention Maps



From the above attention maps from head 0 - 2, the brightness of certain columns are obvious, i.e. column 1, 22, 41. It indicates that these key positions have high attention with many query positions so they should be informative and having important features such as edges. Besides, each head has specific distribution and intensity throughout the map. It means they focus and attend to diverse areas which is beneficial from the multi-head attention. In conclusion, the transformer helps learning in long-range dependencies and play an important role on feature refinement.