# Lab 8 : Analysis of out-of-distribution elements in autoregressive models

Pau Hidalgo Pujol - APRNS @ GIA-UPC

November 28, 2024

## 1 Introduction

This work analyzes autoregressive generative models like MADE, PixelCNN, and ImageGPT. After training, this kind of model could, in theory, be used to detect out-of-distribution elements. Using the Negative Log-Likelihood, we'll evaluate how well these models perform on that task, which are their limitations, and observe the differences between all the classes on the MNIST dataset.

## 2 Methodology

We are will the three models on the MNIST dataset, only using the 0 class. After that, we'll have some models that output a probability distribution related to that class. Loading the other classes and calculating the NLL with those will allow us to get a number indication how good are those models at predicting other classes.

We will also perform some other tests to assess our results: we'll analyze the closest samples from each class by NLL, to see how similar are those images to 0, and we will also measure directly the similarities between the digits using MSE and Structural Similarity Index (SSI), to see if they match with what the models learn. After that, we will also perform some experiments with other classes.

In 1 we can observe the training losses.

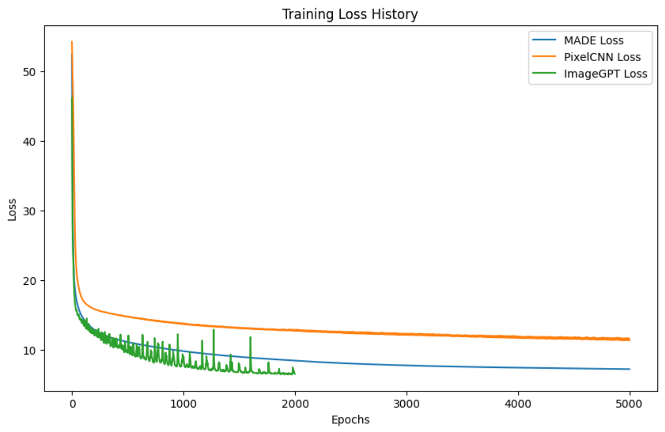


Figure 1: Training Losses of MADE, PixelCNN and ImageGPT

Implementing the first comparison (simply observing the NLL values of other classes) was done by looping through the classes. To look at the closest samples, we used a modified NLL function that works with only one sample and computed the score for each. We then sorted and saved the index of the lowest 5, to be able to visualize them.

Measuring the similarity from classes directly (without the model) was done by computing MSE and the skimage implementation of SSI. We used the average of each class similarity. Finally, performing experiments with other classes meant executing the existing code modifying the initial line where we selected class 0.

## 3 Results

The main results are the NLL comparison for the models trained with the 0 class, plotted at figure 2 and available at table 1.

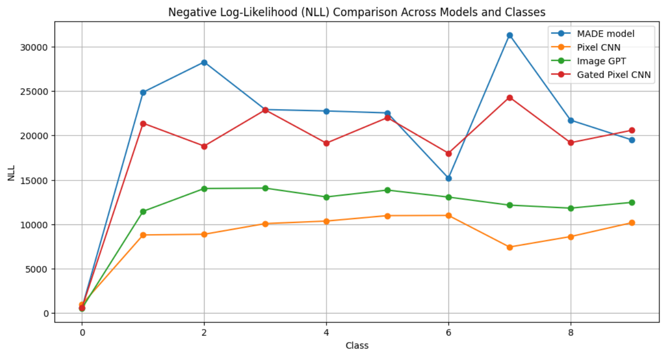


Figure 2: NLL comparison between classes of models trained with class 0

| **Model** | **Class 0** | **Class 1** | **Class 2** | **Class 3** | **Class 4** | **Class 5** | **Class 6** | **Class 7** | **Class 8** | **Class 9** |
|---|---|---|---|---|---|---|---|---|---|---|
| MADE | 609.33 | 24850.56 | 28251.91 | 22904.40 | 22745.77 | 22527.24 | 15191.99 | 31312.09 | 21713.87 | 19496.69 |
| Pixel CNN | 991.16 | 8797.52 | 8871.70 | 10073.49 | 10358.88 | 10971.43 | 10992.98 | 7439.12 | 8613.91 | 10169.76 |
| Image GPT | 559.29 | 11455.29 | 14020.82 | 14064.42 | 13073.48 | 13843.12 | 13056.70 | 12151.58 | 11812.81 | 12458.40 |
| Gated Pixel CNN | 624.25 | 21365.08 | 18800.04 | 22862.44 | 19128.04 | 21991.76 | 17999.58 | 24289.60 | 19177.77 | 20568.77 |

Table 1: NLL Values for Different Models and Classes

We can observe that digit 0 has a very low NLL compared to the others. MADE has a very important drop in NLL score in class 6. However, the other models seem to be more consistent, and they don't exhibit this behavior as much.

Interestingly, the model that performed worse on the 0 class, PixelCNN, is the best when generalizing. That seems to indicate that the model didn't learn as well as the others the distribution of 0, which makes it better in those new cases. ImageGPT, on the other hand, was the best on class 0 but also is pretty good on the other digits. That could be explained by the fact that ImageGPT was trained on fewer epochs.

Let's look at the most similar images for each model. To start, let's see which zeros are the most similar to the learned distribution 3

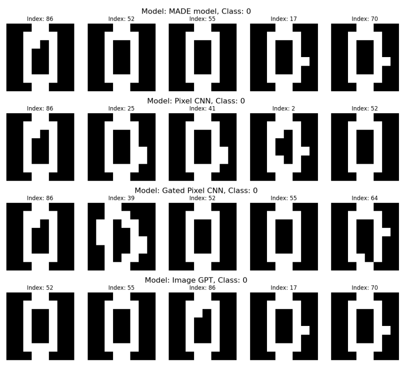
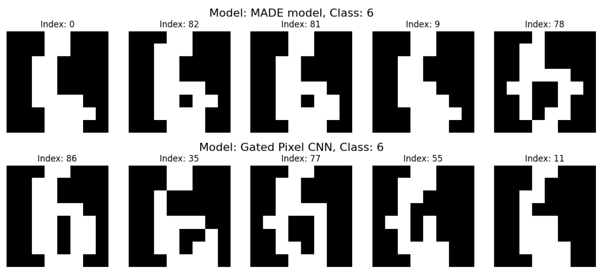


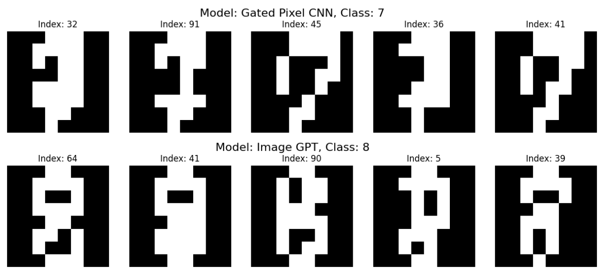
Figure 3: Samples of 0 with the lowest NLL score

They all look like normal zeros. The only difference that we could maybe see is that MADE and ImageGPT learned zeros that were almost "perfect", while PixelCNN and Gated PixelCNN ones are a bit more different.

We can also observe the most similar samples of the most similar classes for each model:



(a) Examples of the lowest NLL classes (excluding 0) for MADE and Gated Pixel CNN



(b) Examples of the lowest NLL classes (excluding 0) for Pixel CNN and ImageGPT

Both MADE and Gated Pixel CNN lowest NLL class (excluding 0) was 6. However, for Pixel CNN and ImageGPT that was 7 and 8 respectively. Looking at the images, it's understandable that the models would get some pixels correctly. The binarized digits, in many cases, lose their original form and end up being pretty weird. If they occupy the same pixels that the 0 used, the models get lower NLLs.

It's interesting to see that, even though MADE and PixelCNN both had 6 as their most similar class, the most similar samples are a bit different: for MADE, they are a bit more "dense", with some not even having a hole, while PixelCNN most similar samples are thinner. This just shows that the models have learned different distributions, and that also affects how they perceive the other classes and what similarities they find.

The more different classes, looking at the results, seem to be 2, 7, and 1.

It's important to note that the results vary from execution to execution since the models learn different things. For example, a replica of the same plot as before looks like 5.

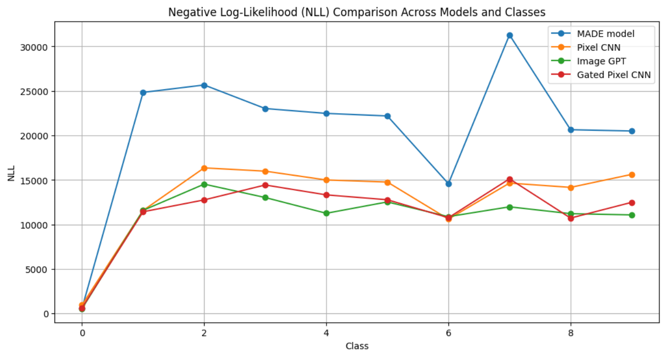


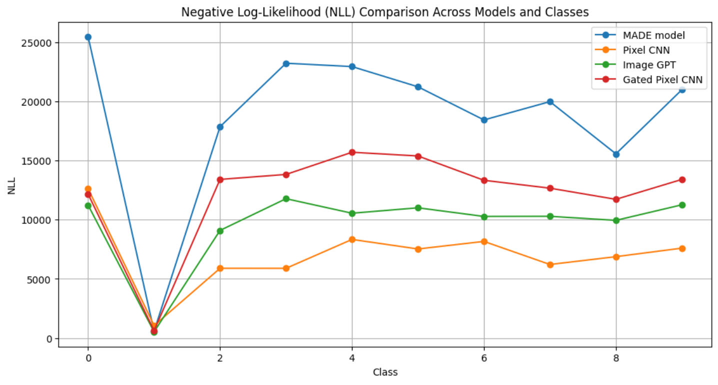
Figure 5: Replica of NLL comparison between classes of models trained with class 0

In general, however, the same trends found apply: MADE has the highest NLL with other classes, and 6 is the most similar one (with 1, 2, and 7 being the most different). These results are also consistent with the comparisons of the samples directly, at table 2.

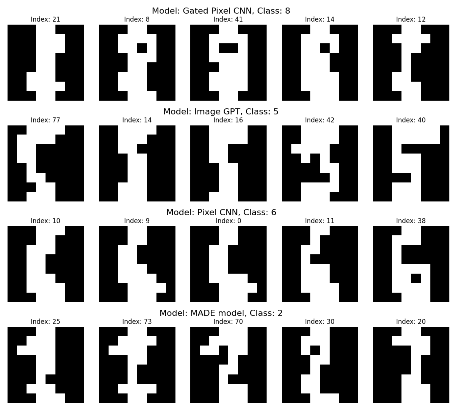
| Comparison | MSE | SSIM | Avg NLL |
|---|---|---|---|
| 0 vs 0 | 0.0000 | 1.0000 | 35.8140 |
| 0 vs 1 | 0.3106 | 0.2616 | 46.9140 |
| 0 vs 2 | 0.2969 | 0.2690 | 46.3540 |
| 0 vs 3 | 0.2791 | 0.3002 | 45.9740 |
| 0 vs 4 | 0.2759 | 0.3411 | 45.9940 |
| 0 vs 5 | 0.2722 | 0.3349 | 45.2540 |
| 0 vs 6 | 0.2372 | 0.4253 | 43.9940 |
| 0 vs 7 | 0.3047 | 0.2602 | 46.8140 |
| 0 vs 8 | 0.2437 | 0.4119 | 43.6740 |
| 0 vs 9 | 0.2428 | 0.3992 | 44.3740 |

Table 2: Comparison of MSE, SSIM, and Avg NLL

As we can see, the similarities we found from sampling the models also hold on the real dataset: the most similar class to 0 is 6 (followed by 8), and the most different ones are 2 and 7. If we train with other classes, we can also see other patterns. For example, training with class 1 we see the following 6a:



(a) NLL comparison between, trained with class 1



(b) Samples from other classes similar to 1

In this case, the general NLL is a bit lower, indicating that there are more samples similar to 1 in the other classes. We can see that 0 is very different, and also that MADE is still the model with the highest NLL. We can also find some of those samples in other classes which are very similar to ones 6b.

## 4 Conclusions

The analysis highlights the behavior of the autoregressive generative models when trained on a single class and evaluated on out-of-distribution data.

MADE exhibited the highest NLL values in those different classes, indicating that it was the model that "best" distinguished between the training class and the others. MADE's design as a masked autoencoder imposes strict constraints and allows it to learn precise distributions, which makes it pretty accurate but also makes it generalize poorly. PixelCNN, which relies on convolutional filters, had the lowest performance on class 0. This weaker ability to specialize in the class it was trained with allowed it to generalize better to out-of-distribution classes, suggesting that its learned distribution may be too "smooth" and not as specific to the training class. Its gated version exhibited an intermediate behavior, performing better on class 0 while generalizing more than MADE. Finally, ImageGPT seemed to be the most versatile model: it achieved the lowest NLL in the training class and also a reasonably low NLL on the other classes. This generalization, in this case, could be linked to fewer training epochs which prevented overfitting to class 0.

With our other experiments, we saw that the model's performances in other classes are aligned with human intuition and with the similarities of the different digits. Training with other classes also allowed us to reaffirm our findings and showed that the results weren't specific to class 0.

In conclusion, if your goal is to generate samples similar to your class, your best model is going to be ImageGPT, since it has the lowest NLL. However, its fewer training epochs make it not the best model to distinguish out-of-distribution samples. If that's your objective, you should use the MADE model, which seems to be able to capture details very specific to the class it's trained on.