Implementation and Evaluation of Conditional PixelCNN++ for Image Classification and Generation

Pedro Leite
Department of Electrical and Computer Engineering
University of British Columbia
pedrorgl@student.ubc.ca
Student Number: 61981213

Abstract

This report details the adaptation of the PixelCNN++ architecture to a conditional model, allowing controlled image generation based on specified labels. The implementation was tested using a diverse dataset including categories such as food and animals. This document covers the modifications made to the original architecture, the training procedures, and the evaluation metrics used to assess the model performance.

1 Introduction

Advancements in conditional image generation through deep learning have significantly impacted the field of computer vision, enabling the synthesis of images from complex data distributions. The PixelCNN architecture, introduced by van den Oord et al. van den Oord et al. [2016], marked a pivotal shift towards autoregressive generative models capable of producing diverse and coherent structures. Its capability to handle high-dimensional data and model the conditional probability distribution of image pixels has been a substantial step forward.

Following this trajectory, Salimans et al. Salimans et al. [2017] further refined this approach with PixelCNN++, enhancing the model's efficiency and sampling speed, thus addressing some of the computational challenges posed by its predecessor. The introduction of discretized logistic mixture likelihood and modified gating mechanisms exemplify the iterative nature of improvement in generative model architectures.

The present work builds upon these innovations, aiming to harness and extend the capabilities of the PixelCNN++ framework for improved image classification and generation. By integrating conditioned generation with labeled data, this study seeks to explore the boundaries of controlled synthesis and the interpretability of generated imagery, ultimately contributing to the robustness and applicability of generative models in practical scenarios.

2 Background and Related Work

Probabilistic Image Modeling is at the core of generative models such as PixelCNN. These models estimate the probability distribution of pixel values across an image, enabling the capture of complex inter-pixel correlations. The model predicts this distribution as a product of conditional distributions. This allows for a factorized likelihood of the image data, expressed as:

$$p_\theta(X_R, X_G, X_B) = \prod_i \Big(p_\theta(x_{R,i} | X_{R,<i}, X_{G,<i}, X_{B,<i}) \\ \cdot p_\theta(x_{G,i} | X_{R,i}, X_{R,<i}, X_{G,<i}, X_{B,<i}) \\ \cdot p_\theta(x_{B,i} | X_{R,i}, X_{G,i}, X_{R,<i}, X_{G,<i}, X_{B,<i}) \Big) \tag{1}$$

where p_θ denotes the probability under the model parameterized by θ, and $x_{R,i}, x_{G,i}, x_{B,i}$ represent the red, green, and blue sub-pixel values at the i-th location, respectively.

PixelCNN Structure

The PixelCNN architecture is a type of autoregressive model that generates images pixel by pixel, predicting each pixel conditioned on the pixels previously generated as seen in Figure 1 . The core

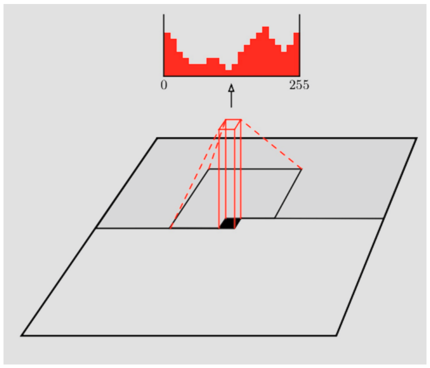

Figure 1: Visualization of the input-to-state and state-to-state mappings

structure of PixelCNN comprises two streams, as visualized in Figure 2, namely the vertical and horizontal streams . These streams are designed to ensure that the generation of a pixel can only be influenced by the pixels above it and to its left, adhering to the autoregressive property. The internal structure of these blocks are demonstrated in Figure 3, the feature maps used within the PixelCNN is bifurcated and thus the information processed by the network is split into two paths. This separation facilitates the management of different aspects of the image, such as texture and edges, and integrates them to form the final prediction.

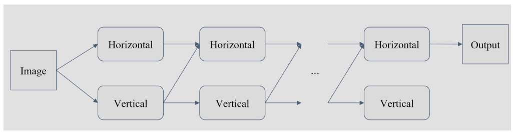

Figure 2: The vertical and horizontal streams of the PixelCNN architecture.

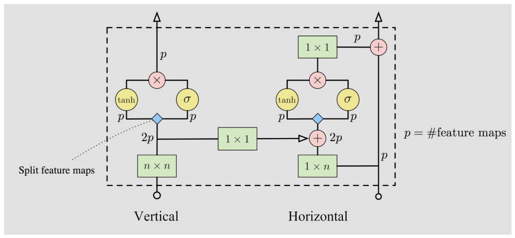

Figure 3: Demonstration of split feature maps in PixelCNN.

The PixelCNN utilizes gated activations, a concept captured by the formula:

$$y = \tanh(W_f x) \circ \sigma(W_g x) \tag{2}$$

This methose is used to control the flow of information. The $\tanh$ non-linearity captures the primary features while the sigmoid gate σ determines the contribution of these features to the next layer.

Advancements with PixelCNN++

PixelCNN++ constitutes a significant improvement over the original PixelCNN model, implementing strategic modifications that enhance its efficiency and expressiveness in image synthesis. One of the principal advancements is the adoption of a discretized logistic mixture likelihood for the conditional distribution of sub-pixel values. This model replaces the exhaustive 256-way softmax distribution with a logistic mixture, which not only curtails memory usage but also expedites the convergence of the model during training. It models the continuous univariate distribution as a mixture of logistic distributions, significantly simplifying the probability calculation for a sub-pixel value x, as shown in Equation (4). This approach ensures precise handling of the probability values, except for the edge cases where the pixel value x is either 0 or 255:

$$\nu \sim \sum_{i=1}^{K} \pi_i \text{logistic}(\mu_i, s_i) \tag{3}$$

$$P(x|\pi, \mu, s) = \sum_{i=1}^{K} \pi_i \left[\sigma\left(\frac{x + 0.5 - \mu_i}{s_i}\right) - \sigma\left(\frac{x - 0.5 - \mu_i}{s_i}\right) \right] \tag{4}$$

Here, σ denotes the logistic sigmoid function. In the special case of the edge values, the sub-pixel value $x-0.5$ is replaced by $-\infty$ for 0, and $x+0.5$ is replaced by $+\infty$ for 255. This numerical approach is elegantly handled in PixelCNN++ and is supported by a stable implementation that computes the log of the probability for the observed pixel values efficiently. The choice of a discretized logistic distribution smoothens the predictive distribution for pixel values, adeptly accommodating the edge pixel values that manifest more frequently in images.

Conditional PixelCNN++

The architecture of PixelCNN++ was augmented to facilitate conditional image generation and classification. Central to this enhancement was the incorporation of class embeddings, which were integrated at various points in the network to guide the generation process conditionally.

Conditional Gating: Gating mechanisms within the convolutional layers were modified to accept conditional vectors derived from class embeddings. This modification enables the network to produce outputs that are contingent on the specified class, thus tailoring the generation process to produce class-specific images. The conditional gating is represented mathematically as follows:

$$y = \tanh(W_{f,k} * x + V_{f,k}^T h) \circ \sigma(W_{g,k} * x + V_{g,k}^T h), \tag{5}$$

where $W_{f,k}$ and $W_{g,k}$ are the convolutional weights for the filter and gate, $V_{f,k}$ and $V_{g,k}$ are the class-conditional biases, and h represents the class embedding vector.

Location-Dependent Conditioning: In addition to class-conditional biases, we explored location-dependent conditioning to introduce variability based on the position within the image. This was particularly useful for datasets with strong location-specific features. The location-dependent conditioning allowed for the adjustment of feature map activations based on both the class embedding and the pixel's location, providing a more nuanced control over the generative process. The adjusted gating mechanism is given by:

$$y = \tanh(W_{f,k} * x + V_{f,k} * s) \circ \sigma(W_{g,k} * x + V_{g,k} * s), \tag{6}$$

where s is the spatial representation mapped from the class embedding h, and $V_{f,k}$ and $V_{g,k}$ are now transformed to provide a location-specific bias.

These architectural advancements were essential for improving the versatility and applicability of PixelCNN++ in conditional image synthesis tasks. The conditional vectors played a pivotal role in guiding the model to learn and recreate class-specific features, thus enhancing both the quality of generated images and the accuracy of classification.

3 Results

In assessing the performance of the enhanced PixelCNN++ models, we carried out extensive experiments over numerous epochs. The standard PixelCNN++ underwent 500 epochs of training, while the variant with spatial representation was subjected to 250 epochs. The training was conducted with a batch size of 32, which emerged as the most effective balance between computational efficiency and learning progress. The learning rate was set to 0.00005, following a decay rate of 0.99995 to ensure gradual adjustments to the weights during optimization.

For the architecture, we chose 30 filters to capture complex image features without excessive computational costs, ensuring a swift and efficient training process. With 15 logistic components in the mixture, we struck a balance between detailed pixel intensity modeling and computational simplicity, essential for realistic image generation. The addition of two residual blocks was to deepen the network's understanding of image hierarchies, facilitating the learning of intricate patterns while mitigating the vanishing gradient problem, thus improving the model's generative and discriminative capabilities.

Quantitative assessments revealed comparable accuracy figures for both the standard and spatial representation models, with no significant divergence observed. This similarity also extended to the Fréchet Inception Distance (FID) scores, which serve as a benchmark for the quality of generated images in comparison to a real dataset.

A notable observation was the quicker convergence rate of the spatial representation model, as depicted in the training bits-per-dimension (BPD) graph. This metric, representing the model's uncertainty or entropy about the data, showed a steeper decline for the spatial model. Below are the comparative visualizations of the model performance:

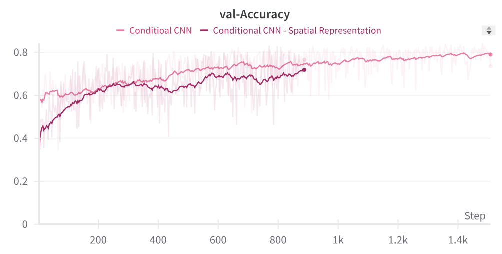

Figure 4: Accuracy comparison between the standard and spatial representation models.

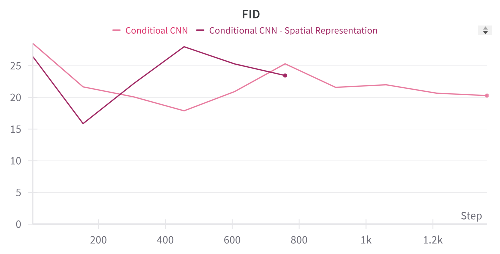

Figure 5: FID score comparison over training epochs.

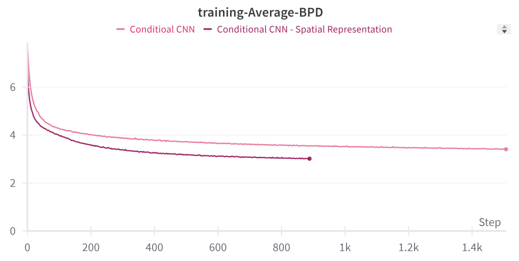

Figure 6: Training bits-per-dimension (BPD) for the models.

Additionally, qualitative analysis through visual inspection of generated image samples corroborated these quantitative findings, reflecting the models' efficacy in synthesizing high-fidelity images while maintaining class-specific attributes. The images can be found on the appendix.

References

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

Tim Salimans, Andrej Karpathy, Xi Chen, Diederik P Kingma, and Yaroslav Bulatov. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

A Appendix

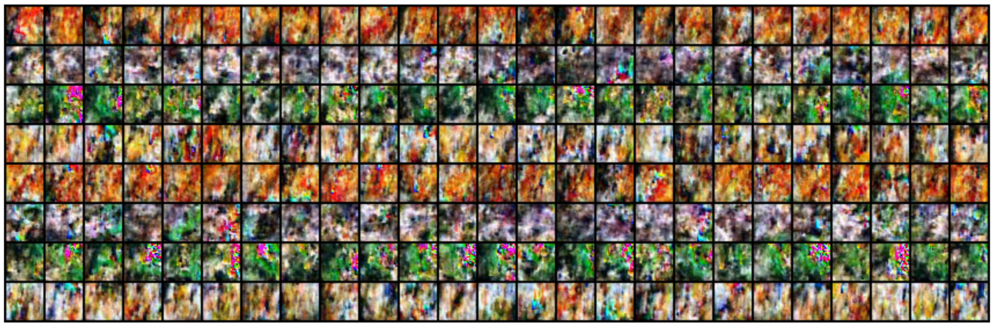

Figure 7: Rotated Image

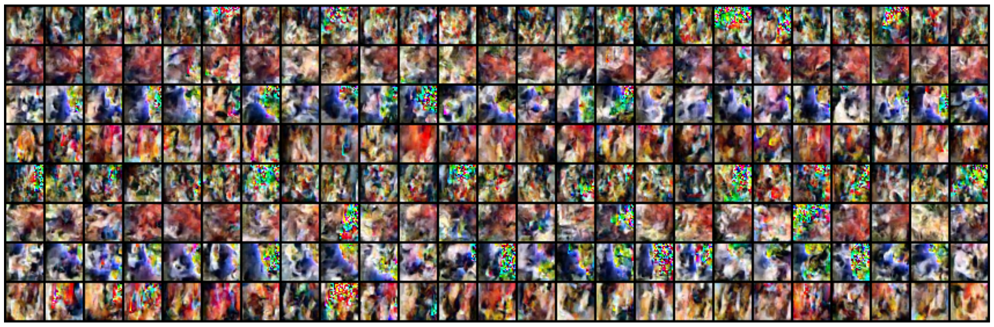

Figure 8: Rotated Image

A.1 ChatGpt chats

https://chat.openai.com/share/67d13846-6774-474f-a38a-6d6c34047f9c
https://chat.openai.com/share/8de9d789-ef0c-4a42-943e-51f6ddd3d1c5