Conditional PixelCNN++ for Image Classification

Phi Lam
Department of ECE
University of British Columbia
philam@student.ubc.ca

Abstract

In this report, I am adapting an unconditional PixelCNN++ to a conditional PixelCNN++ model for image classification. In this report, I describe the methodology used to add class conditioning into the architecture, including different iterations and fusion methodologies. I will discuss the experimental setup and how the model is evaluated using a few metrics, more specifically Fréchet Inception Distance (FID) and classification accuracy, to discuss the performance results. I conclude with an analysis of the model's generative quality, performance, and next steps.

1 Model and Methodology

1.1 Overview

I adapt an existing PixelCNN++ model to support class-conditional image generation by incorporating class information via a combination of early and middle fusion strategies. I inject class-related signals via a one-hot encoded label map at the input, and a learned, scaled class embedding throughout the network's layers.

1.2 Class Conditioning

I explore two fusion strategies for conditioning the model on class labels. I initially only had middle fusion as part of my model, but as the accuracy was not high enough, came to understand that maybe the conditional signal was not strong enough. Therefore, I added early fusion, multiple injections throughout the network and a learnable scalar parameter.

Early Fusion: Each label condition $y \in \{0, \ldots, C-1\}$, where C is the number of classes (in our case $C = 4$), is converted into a one-hot encoded spatial map. This map is then concatenated along the channel dimension with the input image and a constant padding channel. This fusion input is then processed by the initial convolutional layers to inject class information early on.

Middle Fusion: A learned class embedding is scaled by a single trainable scalar parameter α and reshaped to match the spatial dimensions. This embedding is then added to the feature maps after each residual block in both the upward and downward passes. This should thus reinforce class-conditioning information throughout the network.

$$\mathbf{e}_y = \alpha \cdot \text{Embed}(y), \quad \mathbf{e}_y \in \mathbb{R}^D$$

This embedding is then added directly to intermediate feature tensors:

$$\mathbf{f}_i \leftarrow \mathbf{f}_i + \mathbf{e}_y \quad \text{for all layers } i$$

Preprint. Under review.

2 Experiments and Results

2.1 Setup and Training Strategy

I used Kaggle, Google Colab, and upgraded to buying my own GPUs for accelerated training as well as training in the background. I trained the model on our given dataset and evaluate the model's generative and classification capabilities by:

- **Accuracy** — classification performance on the validation set
- **Bits per Dimension (BPD)** — compression efficiency of the model for both validation and training set
- **Fréchet Inception Distance (FID)** — quality of generated images

Accuracy and FID were the primary metrics relevant for our evaluation, so I tuned mostly based off of that. To optimize performance, I conducted a hyperparameter sweep using the `wandb` sweep utility. Initially, I manually tuned and observed trends in the performance metrics (accuracy and FID). So, I kept some hyperparamets consistent(e.g. epochs=200) and didn't add it into the sweep to optimize time. The full sweep configuration is provided in the appendix as Listing 1. The performance were visualized via an accuracy sweep (Figure 1) and a FID score sweep (Figure 2).
I used visual indicators from the hyperparameter sweep to understand the influence of each hyperparameter on improving the model's validation accuracy and decreasing the FID score.

- **Green Indicators:** indicates a positive correlation, therefore increasing this value generally improves performance.
- **Red Indicators:** indicates a negative correlation, implying that higher values tend to reduce performance.

By examining these indicators, I was able to come to understand how each hyperparameter can affect the model. The final hyperparameter configuration in Listing 2 was selected based on these insights that yield the best tradeoffs for a satisfactory performance for accuracy and FID.

2.2 Ablation Study

To assess the impact of our design choices, I performed an ablation study for the addition of the early fusion, multiple injections, and learnable scalar. Unfortunately, due to code-mix up, I can't evaluate based on the accuracy metric for both models as I don't have it recorded. Furthermore, time constraints limited the scope of testing to a comparison between the *combined effect* (early + middle fusion) and *middle fusion* alone.

Figure 3 shows key training and validation metrics from this study. The **middle fusion** configuration achieved a lower BPD, indicating a more efficient representation in terms of likelihood. However as FID is the more important metric we are looking at here, the **combined effect** demonstrates a better FID score and decreasing trend shown in Figure 4. This implies integrating the class conditioning at multiple stages of the network enhances the clairty of the generated samples. Incorporating class information in both early and middle layers reinforces the model's ability to learn the conditional features and therefore increases performance.

2.3 Quantitative Analysis

The quantitative evaluation involved:

- **Validation Accuracy:** Measures how well the model captures class information
- **Fréchet Inception Distance (FID):** Quality of generated images

Figures 1 and 2 visually capture these performance trends. Table 1 shows the best resulting performance of my combined effort model.

Table 1: Classification performance on the validation set.

Metric	Value
Validation Accuracy	78.42003853564548%
FID Score	15.880702676263667

2.4 Qualitative Analysis

I also perform a qualitative analysis by visually inspecting generated samples. The conditional model produced images that almost reflect the target classes (e.g. pizza, etc.). Comparisons between samples generated with only fusiong and combined efforts really showcases the importance of both early and middle fusion mechanisms in enhancing image quality and class identification.

3 Conclusion

In this report, I learned a lot from expanding the PixelCNN++ model for conditional image generation and classification. My results demonstrates a good start into this conditional classifier, achieving a validation accuracy $> 75\%$ and FID $< 30\%$. My next steps would be to explore different fusion methods like FILM and seek to incorporate it into an application for personal usage.

Acknowledgements

I thank Professor Renjie Liao for a good last semester of my electrical engineering degree, and Qi Yan, Sadegh Mahdavi, Qihang Zhang, and Felix Fu for their support and guidance.

References

Lukas Biewald. *Experiment Tracking with Weights and Biases*. Available at `https://www.wandb.com/`, 2020.

Appendix

Listing 1: WandB hyperparameter sweep configuration

```
method: bayes
metric:
  goal: maximize
  name: accuracy
parameters:
  batch_size:
    values: [16, 32, 64]
  lr_decay:
    values: [0.99995]
  max_epochs:
    values: [200]
  nr_filters:
    values: [40, 60, 80, 100]
  nr_logistic_mix:
    values: [5, 10, 15, 20, 25]
  nr_resnet:
    values: [1, 2, 3]
  sample_batch_size:
    values: [8, 16, 24]
  en_wandb:
    values: [true]
program: pcnn_train.py
```

Listing 2: Training Script for Conditional PixelCNN++

```
python pcnn_train.py \
  --batch_size 16 \
  --sample_batch_size 16 \
  --sampling_interval 25 \
  --save_interval 25 \
  --dataset cpen455 \
  --nr_resnet 2 \
  --nr_filters 80 \
  --nr_logistic_mix 10 \
  --lr_decay 0.999995 \
  --max_epochs 200 \
  --en_wandb True
```

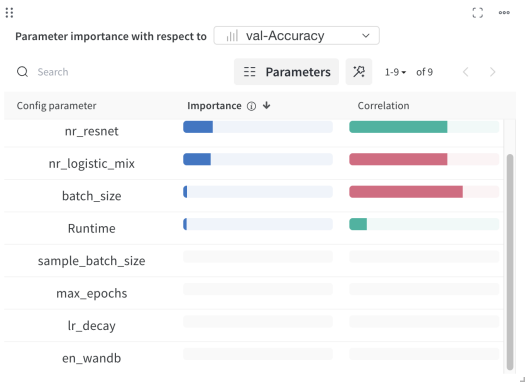

Figure 1: Accuracy sweep results for various hyperparameter configurations.

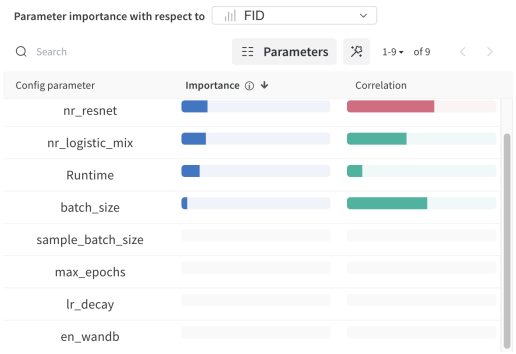

Figure 2: FID score sweep for generative performance across different configurations.

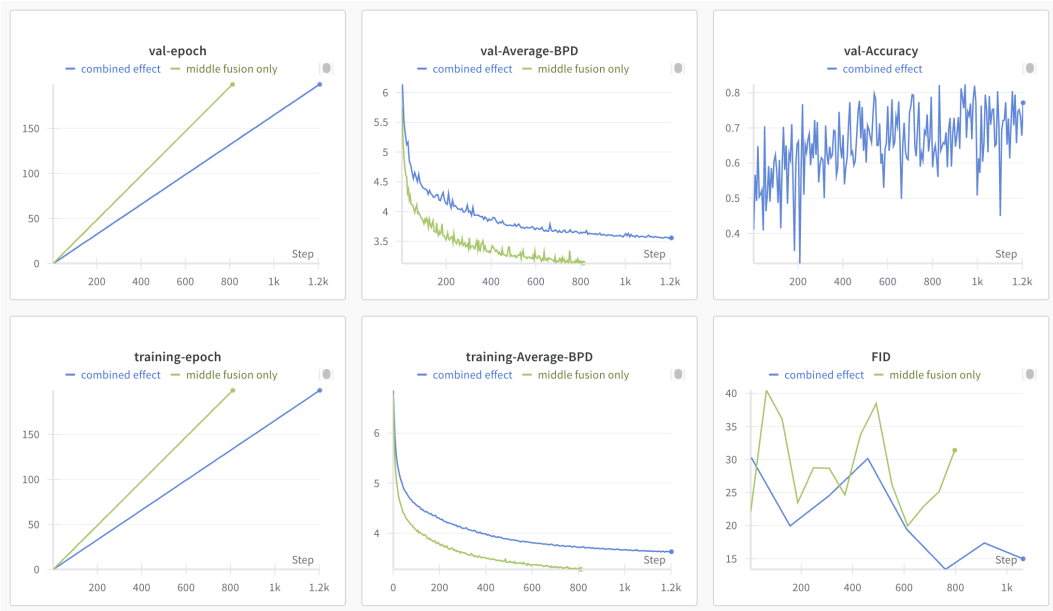

Figure 3: Comparing combined conditioning approach with middle fusion

Figure 4: FID comparison between the combined effect and the middle fusion configurations