CPEN 455 Project Report - Conditional PixelCNN++ for Image Classification

1. Model

1.1 Overview

PixelCNN++ is an autoregressive generative model that factorizes the joint distribution of an image into a product of conditional probabilities.

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, x_2, \ldots, x_{i-1})$$

This project extends PixelCNN++ to a conditional version, enabling both image generation and classification based on class labels [1].

$$p(\mathbf{x}|\mathbf{h}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1}, \mathbf{h})$$

1.2 Model Architecture

The model follows the original U-Net-inspired PixelCNN++ structure, incorporating:

- **Masked convolutions layers** to enforce the autoregressive property by ensuring that the prediction of a pixel does not depend on future pixels in the spatial order. A mask is used in the first layer to block all future pixel values, including the current pixel, while the other is used in subsequent layers to allow the network to process previously generated pixels while maintaining autoregressive constraints [2].
- **Mixture of logistics** to provide a more expressive and stable way to model pixel dependencies compared to a simple Gaussian. They allow the model to capture sharp edges and multimodal distributions more effectively, leading to higher-quality image generation. Additionally, the discretized nature of the distribution aligns well with image data, improving both training stability and sampling efficiency [2] [3].
- **Gated Residual Blocks** to combine input features with learned features using a gating mechanism. The residual connections stabilize training and allow deeper networks [2].

1.3 Middle Fusion for Conditional PixelCNN++

The original unconditional PixelCNN++ is modified to support middle fusion, where class conditioning is injected at the input level. This is achieved by adding an embedding layer that maps class labels to a continuous vector representation.

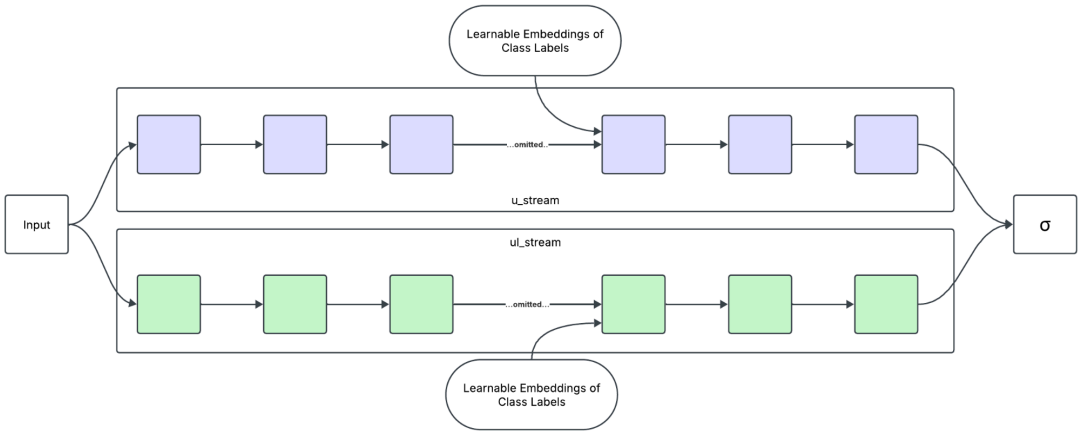

Figure 1: PixelCNN++ Model Architecture with Middle Fusion

1.4 Loss Functions

Two loss functions are utilized:

- **Discretized Mixture of Logistic Loss**: Used for likelihood-based training of the generative model.
- **Negative Log-Likelihood for Classification**: Used to assign labels based on maximum likelihood.

2. Experiments

2.1 Training Setup

- **Dataset**: The model is trained on the provided dataset with four classes.
- **Experiments**
    - Fusion Strategies
    - Data Augmentation
    - Hyperparameter Tuning
- **Optimization**: Adam optimizer with weight decay.
- **Evaluation Metrics**:
    - **FID Score** and **BPD** for image generation.
    - **Classification Accuracy** for conditional image classification.

2.2 Experimental Results

2.2.1 Fusion Strategies

I compared different fusion strategies, including early, middle, and late fusion, and observed that their performance did not differ much in BPD and FID scores but middle fusion wins in classification accuracy. Refer to Figure 2 to 4 in the appendix for the comparison.

2.2.2 Data Augmentation

I applied random horizontal and vertical flips, and random color jitter to test if they help improve performance, but unfortunately none of the metrics showed an improvement.

2.2.3 Hyperparameter Tuning

I tried with different values of `nr_resnet`, `nr_filters`, and `nr_logistic_mix`. The provided command uses 1, 40, and 5 respectively but didn't provide good results even after longer training, but the training speed was fast, e.g. 300 epochs took about 2.5 hours. I observed that increasing `nr_resnet` and `nr_filters` improves the performance obviously, while changing `nr_logistic_mix` didn't affect as much. Therefore taking performance and training speed into account, I settled for 3, 160, and 10. I trained this model with 400 epochs and it took around 36 hours, achieving FID around 30 and a 85% accuracy. Refer to Figure 5 to 7 in the appendix to see the results.

3. Conclusion

3.1 Key Findings

- Class embedding placement, or fusion strategy, significantly affects the result. Middle fusion in conditional PixelCNN provides better results by balancing the integration depth of embeddings and preserving representational capacity. Unlike early fusion, which forces the network to learn modality interactions from scratch, or late fusion, which limits conditioning to final layers, middle fusion injects embeddings at intermediate depths [1].
- The more residual blocks and filters, the better the performance, but also increases training time by a huge amount. Adding more residual blocks optimize gradient flow, expand the receptive field for modeling long-range dependencies. Increasing filter count improves detail preservation and latent space mapping, enhancing texture generation and spatial relationship modeling [2].

3.2 Limitations

- Autoregressive models are computationally expensive for generation.
- Training is extremely slow compared to alternative generative models like GANs or VAEs. I was able to train this large model since I trained exclusively with my MacBook Pro with 16 core GPU, but it still took 1 and a half days.
- Performance is highly dependent on hyperparameter tuning, so conducting experiments takes a great amount of time.

3.3 Future Work

- Implementing attention mechanisms to enhance global dependencies.
- Exploring different conditioning mechanisms (e.g., FiLM layers).
- Training on larger datasets to improve generalization.
- Train with an even bigger model.

References

[1] van den Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., & Kavukcuoglu, K. (2016). Conditional Image Generation with PixelCNN Decoders. *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/1606.05328

[2] Salimans, T., Karpathy, A., Chen, X., Kingma, D. P., & Bulatov, Y. (2017). PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1701.05517

[3] Jubb, T. (2019). Autoregressive Generative Models in Depth: Part 2. Thomas Jubb's Blog. https://thomasjubb.blog/autoregressive-generative-models-in-depth-part-2/

Appendix

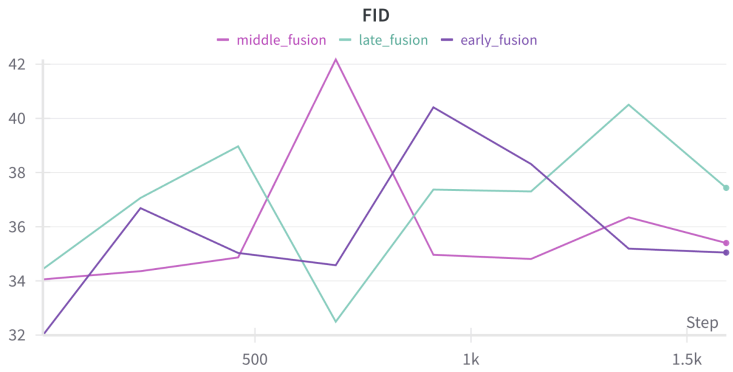

Figure 2: FID Comparison for Different Fusion Strategies

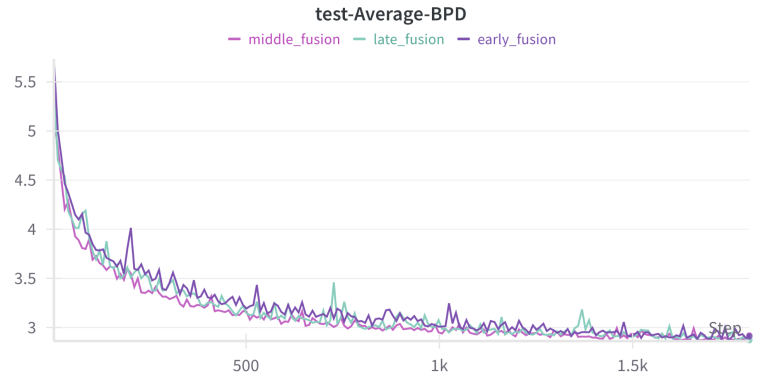

Figure 3: BPD Comparison for Different Fusion Strategies

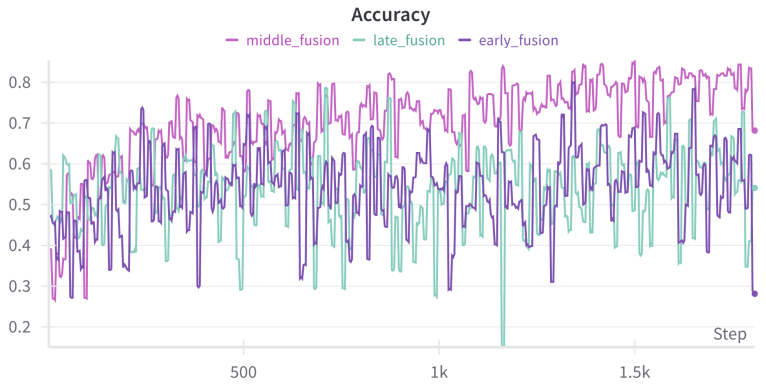

Figure 4: Accuracy Comparison for Different Fusion Strategies

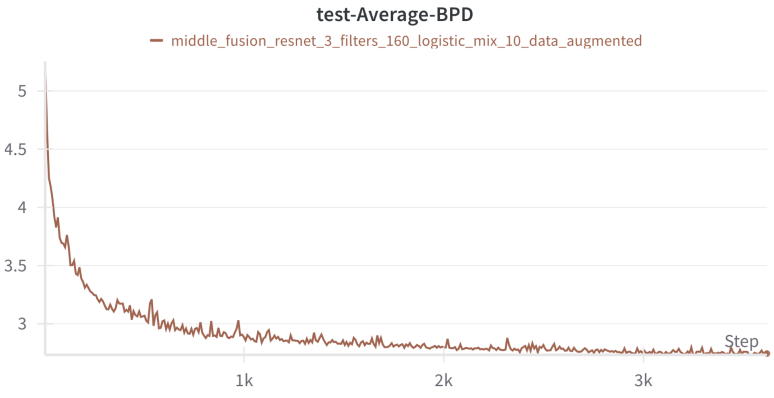

Figure 5: BPD for the Final Model

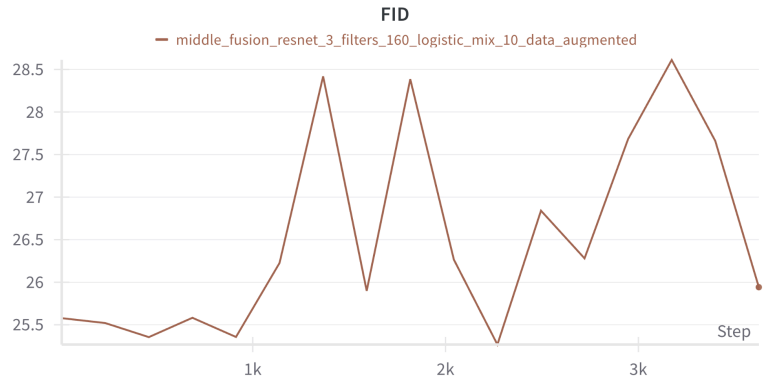

Figure 6: FID for the Final Model

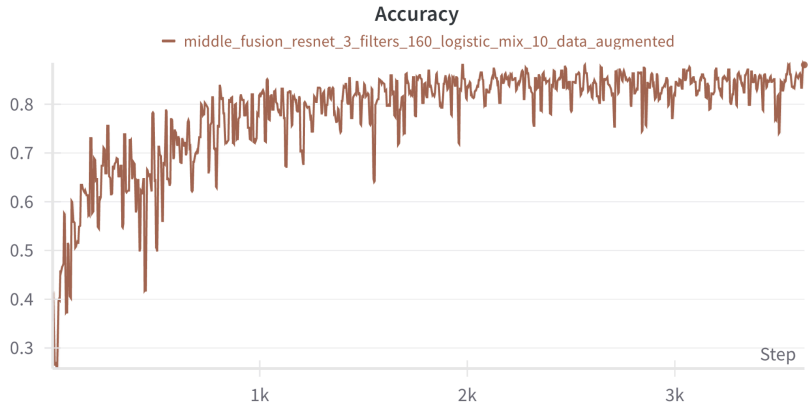

Figure 7: Accuracy for the Final Model