Conditional PixelCNN++ for Image Classification

Juhee Han[*]
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada
`jhan57@student.ubc.ca`

1 Model

1.1 Architecture

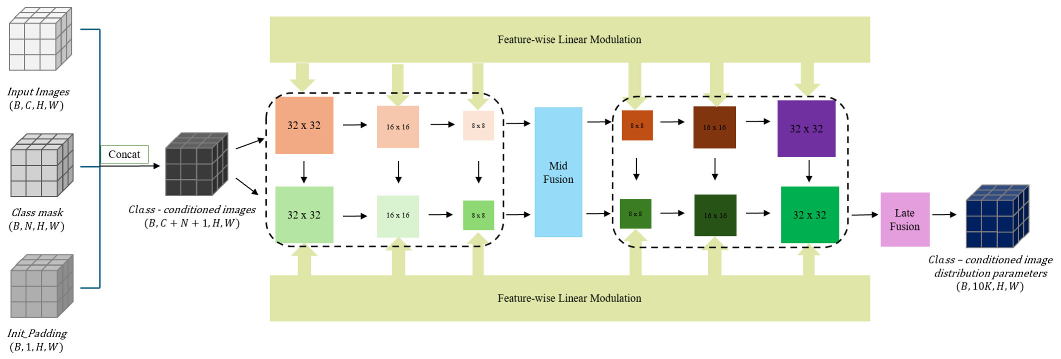


Figure 1: Overview of the class-conditional PixelCNN++ architecture with early, mid, and late fusion mechanisms. The input image $\mathbf{x} \in \mathbb{R}^{B\times C\times H\times W}$ is concatenated with a one-hot class mask and an init-padding channel, forming the *class-conditioned input images*. This tensor flows through two streams: the vertically masked stream (u) and the vertically+horizontally masked stream (ul), each processed across multiple spatial resolutions (from 32×32 to 8×8). Feature-wise Linear Modulation (FiLM) injects class embedding as channel-wise modulation into every residual block throughout both streams. Mid-fusion applies a 1×1 convolution and adds it to both u and ul in the bottleneck (8×8) layer. Finally, late fusion injects the class embedding once again before producing the output parameters of a discretized logistic mixture model, yielding the class-conditional distribution $\hat{x} \in \mathbb{R}^{B\times 10K\times H\times W}$.Here, N denotes the number of classes, and K is the number of logistic mixture components (`nr_logistic_mix`). The black dashed boxes indicate residual blocks, each repeated M times where $M =$ `nr_resnet`.

1.2 Algorithm

[*]This work was conducted as part of the CPEN455 course project at the University of British Columbia (UBC). For inquiries, please contact jhan57@student.ubc.ca

Algorithm 1 Forward Pass of Conditional PixelCNN++

```
Require: Image tensor $x \in \mathbb{R}^{B\times C\times H\times W}$, class label $y \in \{0, \ldots, N-1\}$
 1: # Early fusion
 2: $m \leftarrow$ one_hot$(y)$
 3: $M \leftarrow$ repeat$(m, H, W)$
 4: $P \leftarrow$ ones$(1, H, W)$
 5: $x' \leftarrow$ concat$(x, M, P)$
 6: # FiLM conditioning
 7: $e \leftarrow$ Embed$(y)$
 8: $\gamma \leftarrow W_\gamma e + b_\gamma, \quad \beta \leftarrow W_\beta e + b_\beta$
 9: $\Gamma \leftarrow$ broadcast$(\gamma, H, W), \quad B \leftarrow$ broadcast$(\beta, H, W)$
10: # Up-pass (three scales)
11: $(u_0, ul_0) \leftarrow$ InitStreams$(x')$
12: for $s = 1$ to $3$ do
13:     for $j = 1$ to $M$ do
14:         $u_{s,j} \leftarrow$ GatedResNet$(u_{s,j-1};\ \Gamma,\ B)$
15:         $ul_{s,j} \leftarrow$ GatedResNet$(ul_{s,j-1};\ u_{s,j} + \Gamma,\ B)$
16:     end for
17:     if $s < 3$ then
18:         $(u_{s+1,0}, ul_{s+1,0}) \leftarrow$ DownScale$(u_{s,M}, ul_{s,M})$
19:     end if
20: end for
21: # Mid fusion (at bottleneck 8×8)
22: $\hat{e} \leftarrow$ Conv$_{1\times 1}(e)$
23: $(u, ul) \leftarrow (u_{3,M} + \hat{e},\ ul_{3,M} + \hat{e})$
24: # Down-pass with skip connections
25: for $s = 3$ to $1$ do
26:     for $j = 1$ to $M$ do
27:         $u \leftarrow$ GatedResNet$(u;\ \Gamma,\ B,$ skip $= u_{s,M-j})$
28:         $ul \leftarrow$ GatedResNet$(ul;\ \Gamma,\ B,$ skip $= ul_{s,M-j})$
29:     end for
30:     if $s > 1$ then
31:         $(u, ul) \leftarrow$ UpScale$(u, ul)$
32:     end if
33: end for
34: # Late fusion and output layer
35: if fusion_type = "add" then
36:     $ul \leftarrow ul + \Gamma$
37: else
38:     $ul \leftarrow$ concat$(ul, \Gamma)$
39: end if
40: $\hat{x} \leftarrow$ nin_out(ELU$(ul)$)
41: return $\hat{x} \in \mathbb{R}^{B\times 10K\times H\times W}$
```

2 Experiments

2.1 Training Techniques and Conditioning Strategies

To improve both generation quality and classification accuracy in our conditional PixelCNN++ model, I integrate a combination of architectural, optimization, and regularization techniques. Below, I describe each in detail.

Class Conditioning via Multi-stage Injection. I inject class information through three complementary mechanisms:

- **Early fusion:** A one-hot mask $m \in \mathbb{R}^{N\times H\times W}$ (with N being the number of classes) is concatenated to the input image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$, forming a class-aware input.

- **FiLM:** Inside every Gated ResNet block, the class embedding $\mathbf{e} \in \mathbb{R}^D$ is passed through learned linear layers to produce modulation parameters (γ, β), which modulate intermediate features x as:
$$\tilde{x} = \gamma(\mathbf{e}) \cdot x + \beta(\mathbf{e}).$$
- **Mid-fusion and Late-fusion:** The class embedding is also fused into the bottleneck and the final features before output via either channel-wise addition.

Label Dropout. To enhance generalization and enable semi-conditional behavior, I apply label dropout during training with probability $p = 0.1$:
$$\tilde{\mathbf{e}} = m \odot \mathbf{e} + (1 - m) \odot \epsilon, \quad m \sim \text{Bernoulli}(1 - p), \;\; \epsilon \sim \mathcal{N}(0, 0.01^2),$$
where ϵ adds Gaussian noise to dropped-out embeddings.

Data Augmentation. To prevent overfitting and improve sample diversity, I employ a multi-stage augmentation pipeline:

- Basic augmentations: `RandomCrop(32, padding=4)` and `RandomHorizontalFlip()`.
- Strong content-preserving: `RandAugment(N=2, M=9)`, Cutout (erase 16×16 patches).
- Style perturbation: `ColorJitter`, `GaussianBlur`.

Dropout Regularization. I use spatial dropout ($p = 0.1$) inside Gated ResNet blocks to regularize the model and mitigate overfitting in the deep autoregressive structure.

Optimizer. Training is performed using AdamW with weight decay 1×10^{-4}, and a learning rate scheduler: `StepLR` with decay $\gamma = 0.999995$ per epoch.

Exponential Moving Average (EMA) and Warm-up. To stabilize training and reduce noise in evaluation metrics such as FID, I maintained an Exponential Moving Average (EMA) of the model weights θ. The EMA weights θ_{EMA} are updated at each step t via:
$$\theta_{\text{EMA}}^{(t)} = \lambda \theta_{\text{EMA}}^{(t-1)} + (1 - \lambda)\theta^{(t)}, \quad \lambda = 0.999,$$
where λ is the decay rate. The EMA weights are used for validation and sampling rather than the raw training weights.In later ablation studies, I also disabled EMA entirely to assess its direct effect on model convergence, sample quality, and classification accuracy.

2.2 Ablation Study

1. Exponential Moving Average (EMA). I investigated the impact of EMA and its variants on training stability and performance:

- **With EMA:** Validation accuracy exhibited more oscillations, but eventually achieved a final accuracy of 87%.
- **With EMA Warm-up (5 epochs):** I applied a warm-up strategy to delay EMA updates at early training stages. While it stabilized the early updates, the FID convergence was extremely slow, taking over 200 epochs to decrease from 250 to 199. Therefore, I discarded this strategy in later experiments.
- **Without EMA:** Accuracy curves were more stable overall, with fewer fluctuations, reaching a final accuracy of 80%.

2. Data Augmentation. The following augmentation techniques were evaluated:

- **Standard:** `RandomCrop(32, padding=4)` and `RandomHorizontalFlip()`.
- **Content-preserving:** `RandAugment(N=2, M=9)`, `Cutout(16×16)`.
- **Style-perturbing:** `ColorJitter`, `GaussianBlur`.

While these augmentations improved data diversity, their contribution to accuracy was marginal compared to the gains provided by class conditioning.

3. Label Dropout. Applying label dropout ($p = 0.1$) degraded both classification accuracy and generation quality. In comparison experiments, models trained without label dropout consistently outperformed their counterparts with dropout, suggesting that reliable class conditioning is critical to stable training.

4. Class Conditioning Strategy. I compared different fusion strategies while keeping the base conditioning mechanisms fixed (FiLM, one-hot mask, and Gated ResNet). The setup using both mid- and late-fusion outperformed configurations using only mid-fusion, demonstrating that multi-stage injection of class information enhances both classification and generation performance.

More specific results and graphs are provided in Appendix A.

2.3 Results and Analysis

I evaluate the architecture illustrated in Figure 1, which incorporates the following design choices:

- **Class conditioning:** FiLM layers (via γ, β modulation), gated ResNet blocks, and a one-hot mask concatenated to the input (early fusion).
- **Fusion strategies:** Both mid-fusion and late-fusion.
- **Regularization:** No data augmentation or dropout was applied.

The model was trained using the configuration: `-nr_resnet 1 -nr_filters 160 -batch_size 16 -sample_batch_size 16 -nr_logistic_mix 5`.

Final performance. At epoch 130, the model achieved the following results:

- **Test Accuracy:** 88.03%
- **F1 Score:** 87.99%
- **FID:** 28.95

This configuration demonstrates that even with a shallow network (`nr_resnet=1`), the combination of FiLM conditioning and fusion mechanisms is effective for both classification and image generation, yielding strong results without requiring data augmentation or dropout regularization.

3 Conclusion

In this project, I extended the unconditional PixelCNN++ model into a class-conditional generative model by integrating multiple conditioning mechanisms. The final architecture combines early fusion via one-hot mask, FiLM-based channel modulation in every gated ResNet block, mid-fusion at the 8×8 bottleneck, late-fusion before output prediction,and also EMA. These mechanisms allow class information to be injected at multiple depths and resolutions, improving both classification and generation performance.

Key contributions.

- I systematically implemented and compared various fusion strategies (early, mid, late), demonstrating that combining all outperforms single-point conditioning.
- Ablation studies revealed that data augmentation and label dropout offer minor improvements compared to embedding-based conditioning.
- EMA enhanced the performance and finally achieved strong results (Test Acc: 88.03%, FID: 28.95) without data augmentation or dropout.

Limitations and future work. While the model performs well in terms of accuracy and FID, there are still avenues for improvement:

- **Generation fidelity:** The generated images occasionally lack fine-grained texture details. Exploring more expressive output distributions could improve visual quality.

References

[1] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshops*, pages 702–703, 2020.

[2] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. In *arXiv preprint arXiv:1708.04552*, 2017.

[3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[4] Jonathan Ho, Xi Chen, and Aravind Srinivas. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning (ICML)*, pages 2722–2730, 2019.

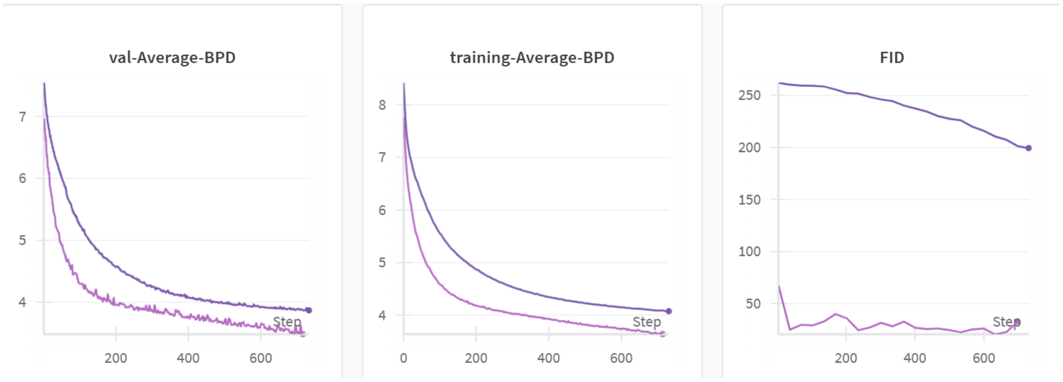
[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[6] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, pages 3942–3951, 2018.

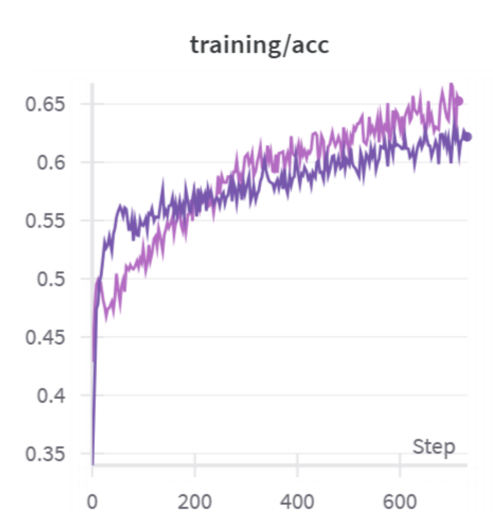
[7] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations (ICLR)*, 2017.

[8] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning (ICML)*, pages 1747–1756, 2016.

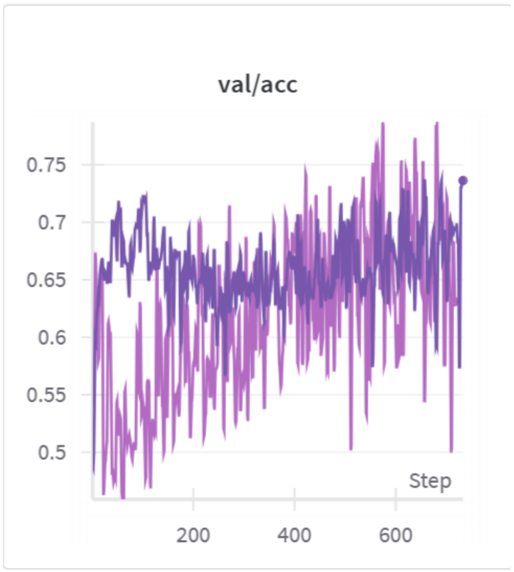
A Additional Training Curves and Ablation Comparisons



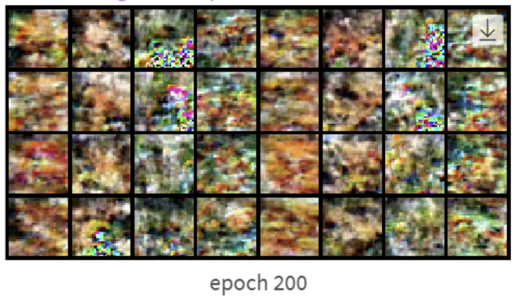
(a) BPD and FID



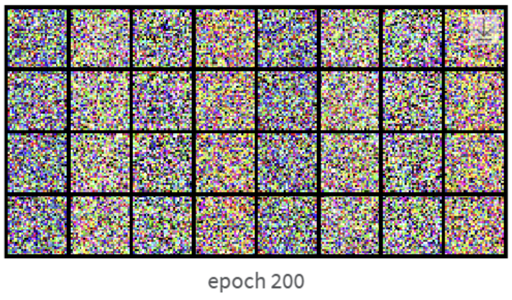
(b) Train Accuracy



(c) Validation Accuracy

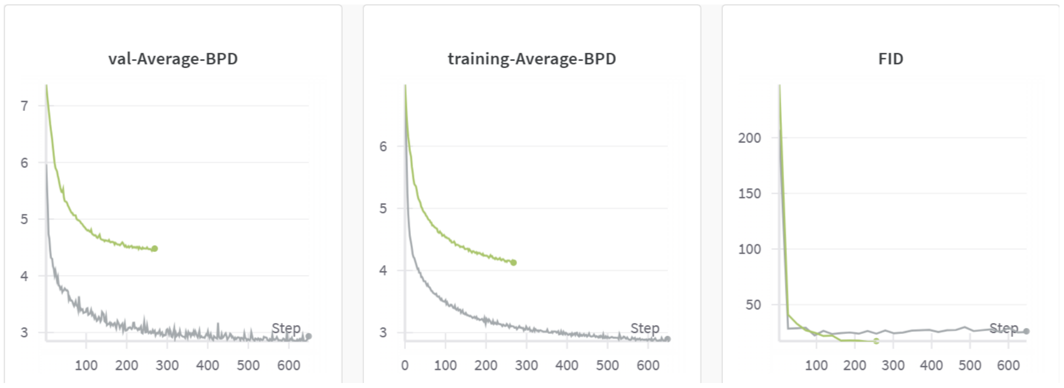


(d) Sampled Images (No EMA)

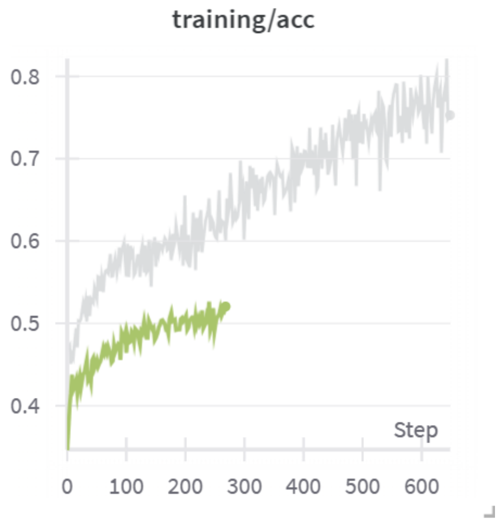


(e) Sampled Images (Warmup EMA)

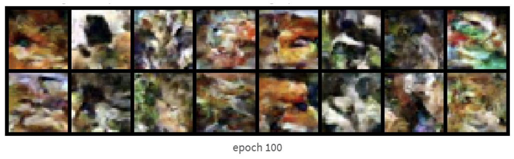
Figure 2: **EMA Ablation.** The dark purple line represents the model trained with EMA and 5-epoch warm-up. The light purple line corresponds to the model trained without EMA. The absence of EMA leads to more stable accuracy curves, while EMA requires warm-up to stabilize early training. Due to warm-up, even after 200 epochs, FID is just below 200 and sampled images remain noisy. Without warm-up, EMA training leads to oscillating validation accuracy, as seen in the gray curve of Figure 4.



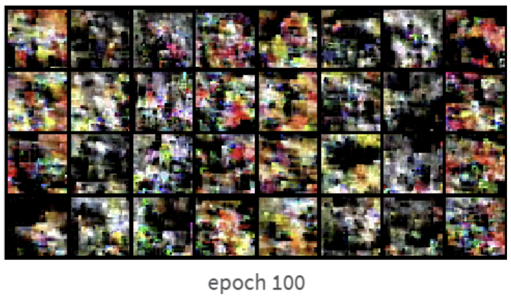
(a) BPD and FID



(b) Train Accuracy

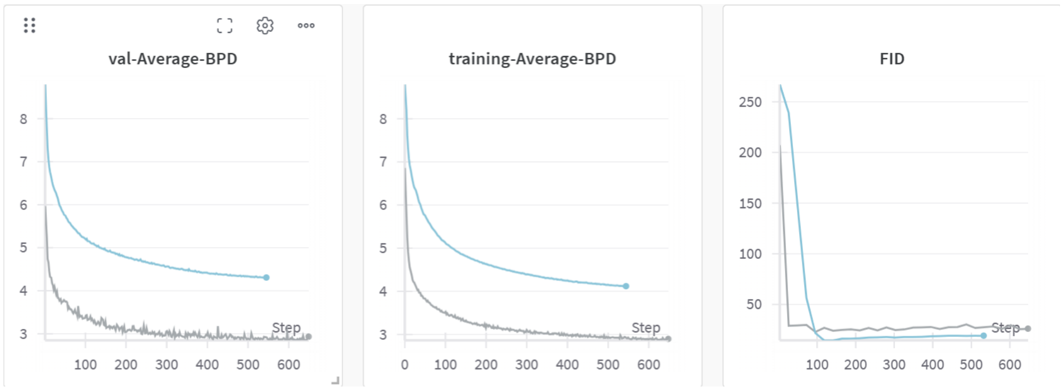


(c) Sampled Images (No Aug/Dropout)

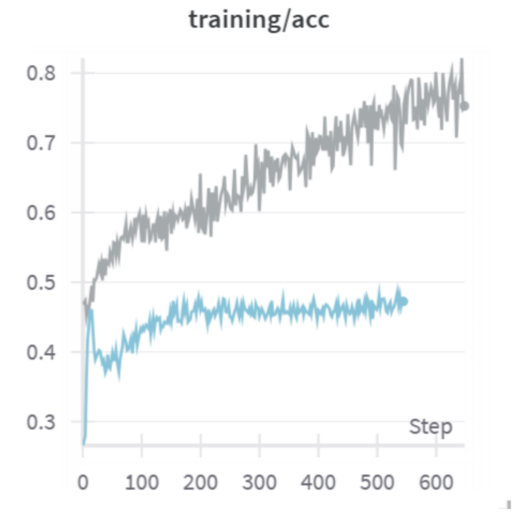


(d) Sampled Images (With Aug/Dropout)

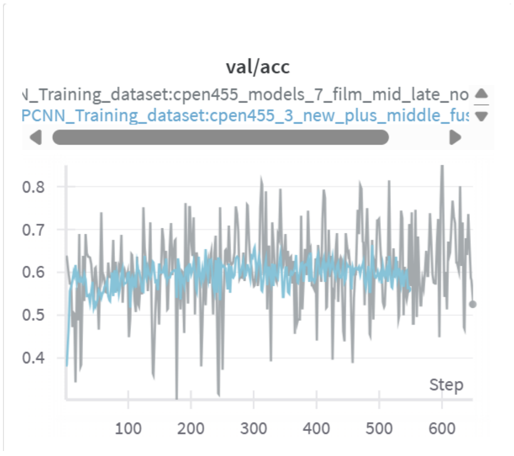
Figure 3: **Data Augmentation and Dropout Ablation.** The light green line indicates the configuration with full augmentation (RandomCrop, RandAugment, Cutout, Jitter, GaussianBlur) and dropout ($p = 0.1$). The dark blue line corresponds to a model without augmentation or dropout. Augmented and dropouted images are visually distinguishable due to jitter and cutout. However, these augmentations and dropout did not improve model.



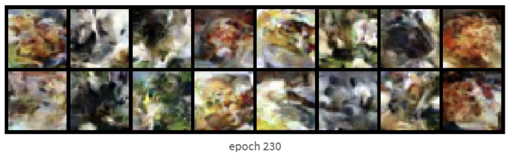
(a) BPD and FID



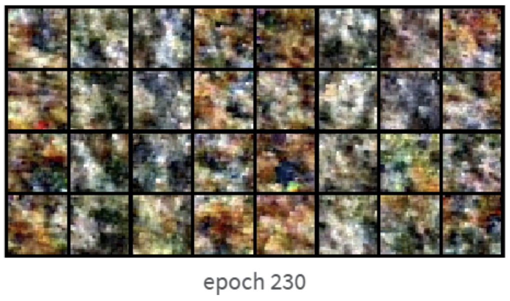
(b) Train Accuracy



(c) Validation Accuracy

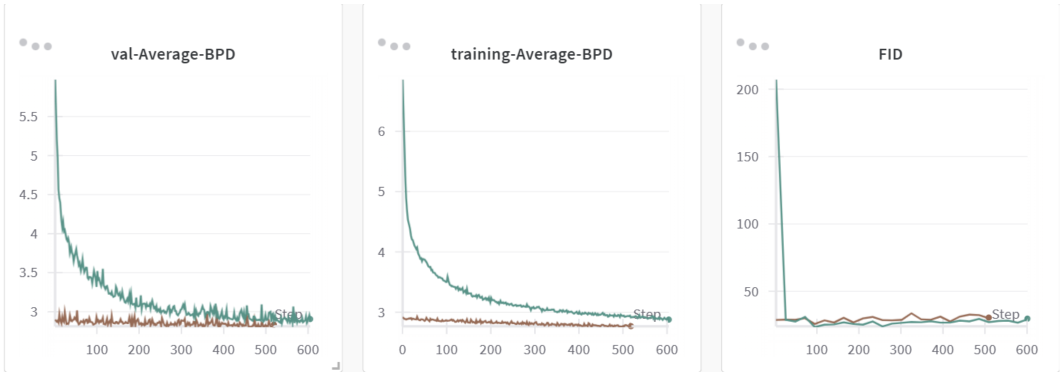


(d) Sampled Images (w/ Late Fusion)

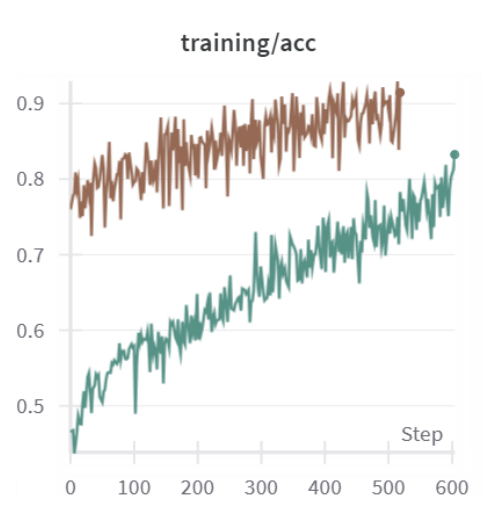


(e) Sampled Images (w/o Late Fusion)

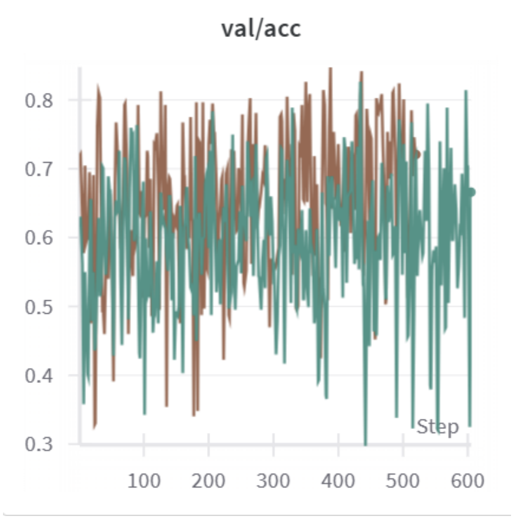
Figure 4: **Fusion Strategy Ablation.** The gray line shows the model using both mid-fusion and late-fusion. The light blue line corresponds to the model with mid-fusion only. Adding late-fusion improves class-conditional control, especially in validation accuracy and image sharpness.



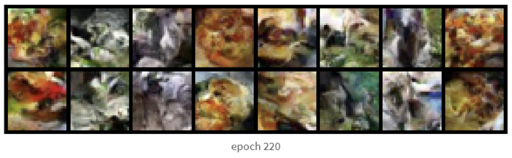
(a) BPD and FID



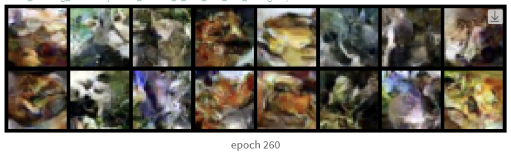
(b) Train Accuracy



(c) Validation Accuracy

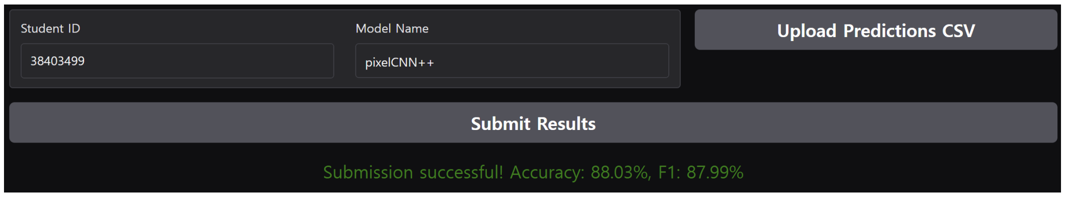


(d) Sampled Images (Epoch 0–220)



(e) Sampled Images (Epoch 230–490)

Figure 5: **Best Performance Model.** The green curve represents the training trajectory from epoch 0 to 220, while the brown curve corresponds to epochs 230 to 490. Throughout the training process, the sampled images appear increasingly sharp, and the FID consistently remains below 30. Additionally, the bits-per-dimension (BPD) steadily decreases as training progresses, indicating improved log-likelihood modeling.



Student ID
38403499
Model Name
pixelCNN++

Submission successful! Accuracy: 88.03%, F1: 87.99%

(a) Test Accuracy

Figure 6: **Best Performance Model.** The final test accuracy reached 88.03% with an F1 score of 87.99%.