CPEN455 2024 W2 Course Project: Conditional PixelCNN++ for Image Classification

Ranbir Sharma
Department of Electrical and Computer Engineering
University of British Columbia
ranbirs@student.ubc.ca

1 Model

We base our model architecture on PixelCNN++ (Salimans et al. 2017), a deep autoregressive model that factorizes the joint distribution of pixels using a conditional distribution for each pixel given the previous ones. To enable conditional generation, we augment the architecture to incorporate label-based conditioning using four fusion strategies: Early Fusion, Gated Residual Fusion, Late Fusion and Joint Fusion.

Each model shares the same core PixelCNN++ backbone with gated convolutions, down- and up-sampling paths, and discretized logistic mixture loss.

For our final submission, we have used Gated Residual Fusion and have implemeted other fusion strategies for the bonus points (Bonus 2, 4 and 5).

1.1 Model Description

We extend the **PixelCNN++** architecture for conditional image generation. Given an input image $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$ and a class label $y \in \{1, \ldots, K\}$, the goal is to model the conditional distribution $p(\mathbf{x} \mid y)$.

1.2 Conditioning Mechanism

To incorporate class label information, we embed the label using a learnable embedding matrix:

$$e_y = Embed(y), \quad e_y \in \mathbb{R}^d \tag{1}$$

This embedding is fused into the model through different strategies:

- **Early Fusion**: e_y is broadcast spatially and concatenated to the input image along the channel dimension. In general, Class embeddings are spatially broadcasted and concatenated to the input image along the channel dimension before any convolutions.
- **Gated Residual Fusion**: e_y is passed through a non-linear transformation and added to each residual block. In general, Class embeddings are injected into gated residual blocks throughout the network to modulate feature maps.
- **Late Fusion**: e_y is projected and added to the final feature map before the output layer. In general, Class embeddings are transformed via a linear projection and added to the final feature maps before the output layer.
- **Joint Fusion (Saponaro et al. 2023)**: Combines early, residual, and late fusion to enrich conditional information throughout the model. In general, early, Gated Residual, and late fusion, provides conditioning at all stages of the network.

1.3 Model Diagram

Figure 1 illustrates the computation graph of our Conditional PixelCNN++ model, adapted to support multiple fusion strategies (early, gated, late, joint). The architecture follows a U-Net structure composed of an up pass and a down pass, facilitating multi-scale context modeling.

Input: The input is a 3-channel RGB image of size 32×32. In conditional settings, the class label is embedded and fused into the network based on the selected fusion strategy.

Initial Layers:

- `u_init`: A down-shifted convolution that initializes the vertical stream (u-stream), responsible for capturing dependencies from pixels above.
- `ul_init`: A combination of down-shifted and down-right-shifted convolutions that initialize the diagonal stream (ul-stream), which considers context from pixels above and to the left.

Up Pass: The model processes input features through a series of gated residual blocks in both the u-stream and ul-stream (Downsize streams). At each stage:

- Gated residual blocks conditionally integrate label embeddings (in gated or joint fusion).
- Feature maps are downsampled through strided convolutions to build a hierarchical representation.

Down Pass: This path mirrors the up pass, input - Upsize streams:

- The u and ul feature streams are upsampled using deconvolutions.
- Residual blocks reuse features saved during the up pass via skip connections.
- Fusion strategies like late fusion add conditional vectors just before the final output layer.

Output Layer: After the final ul-stream output, an activation and 1x1 convolution projects the output to parameters of the discretized mixture of logistics (used for likelihood estimation or sampling).

1.4 Model Equations

Here, we are going to write down the equations used for fusion strategies:
Early Fusion:

$$\tilde{\mathbf{x}} = \text{Concat}(\mathbf{x}, \mathbf{1}, e_y^{\text{spatial}}) \tag{2}$$

Where $\mathbf{1}$ is a channel of ones (used in original PixelCNN++) and e_y^{spatial} is the class embedding broadcasted to spatial dimensions.

Gated Residual Block:

$$h = \text{GRUBlock}(h_{in}) + \text{NIN}(e_y) \tag{3}$$

The gated residual block(GRU) modulates input features using class-specific information projected through a NIN layer.

Late Fusion:

$$h_{out} = h_{out} + W_{late} e_y \tag{4}$$

Where $W_{late} \in \mathbb{R}^{d \times F}$ projects the embedding to the feature map dimension.

1.5 Loss Function

We model each pixel using a discretized mixture of logistic distributions:

$$p(x \mid y) = \sum_{k=1}^{M} \pi_k(y) \cdot Logistic(x; \mu_k(y), s_k(y)) \tag{5}$$

where M is the number of mixture components, and each component is parameterized by a mean μ_k, scale s_k, and mixture coefficient π_k.

The model is trained using the negative log-likelihood:

$$\mathcal{L}_{NLL} = -\log p(\mathbf{x} \mid y) \tag{6}$$

This loss encourages the model to learn the conditional distribution of image pixels given the class label.

2 Experiment

2.1 Training Methods and Techniques

All conditional PixelCNN++ models were trained using the Adam optimizer with an initial learning rate of 0.002 and a batch size of 32. Each model used 3 ResNet blocks, 80 filters and 5 logistic mixture components. Each model was trained for 75 epochs, with early stopping applied based on validation loss to prevent overfitting.However, the main model was trained on the same hyperparameters but was trained until 300 epochs.

During training, PixelCNN++ tries to minimize the negative log-likelihood (NLL) of the training images — more specifically, it's optimizing a discretized mixture of logistic likelihood for the pixel values. All models were trained on a single NVIDIA GPU (T4), and intital training convergence was typically observed around 70 epochs. For consistent evaluation, the same training hyperparameters and random seed were used across all experiments.

2.2 Quantitative Analysis:

We evaluate the generative quality of our model using the Fréchet Inception Distance (FID), and classification accuracy using a classifier.

Table 1: Comparison of fusion strategies

Fusion Strategy	**Train Acc (%)**	**Val Acc (%)**	**FID Score**
Early Fusion	41.7	44.28	33.7
Gated Residual	**60.5**	**68.7**	34.14
Late Fusion	46.6	46.28	**27.04**
Joint Fusion	52.9	50	33.54

Here, due to limited GPU resources, early, late, gated residual and joint fusion were trained only until 75 epochs (Bonus 2 and 4). However, another Gated Residual fusion model was trained until 300 epoch as a part of the main submission. The main model has the training accuracy of 92.5%, validation accuracy of 77.6% and testing accuracy of 78.19%, with FID score of 28.

You can view each of the model's graphs at the Appendix Section 3. In Summary, gated resnet performed the best in terms of training accuracy and validation accuracy, whereas late fusion performed the best at FID.

2.3 Qualitative Analysis:

Figures in Appendix Section 3 presents a visual comparison of samples generated using four fusion strategies: Early, Late, Gated, and Joint Fusion.

Early Fusion tends to introduce low-level conditioning signals early in the network, resulting in diverse yet occasionally chaotic outputs. Some images display strong visual patterns, but semantic structure is often lacking.

Late Fusion, while better at preserving high-level structure, often suffers from blurry or less vibrant outputs. It relies heavily on the decoder's capacity to integrate late-stage information, which can cause reduced image diversity.

Gated Fusion introduces conditioning selectively, leading to improved clarity and localized coherence. The outputs appear more refined and balanced than early or late alone.

Joint Fusion, which combines Early, Gated, and Late strategies, produces the most visually rich and diverse samples. However, it occasionally results in overly saturated regions and some semantic ambiguity. Despite these issues, Joint Fusion strikes a good balance between detail and high-level structure, making it a strong candidate for conditional image generation.

2.4 Bonus 5: Comparsion with another classifier

Table 2: Accuracy comparison between ResNet18 and different fusion strategies of PixelCNN++

Model	Train Accuracy	Validation Accuracy	Test Accuracy
ResNet18	98.23%	85.93%	87.07%
PixelCNN++ (Gated)	92.5%	77.26%	78.19%

We compare the performance of our conditional PixelCNN++ model with a dedicated CNN-based classifier, ResNet18, trained on the same CPEN455 dataset and hyperparameter using the Adam optimizer. While PixelCNN++ is generative in nature and models the joint distribution of pixels, ResNet18 is discriminative and optimized solely for classification tasks.
Advantages of conditonal PixelCNN++ over Resnet18

- Can generate class-conditional images.
- Allows architectural flexibility through conditioning fusion strategies.
- Offers fine-grained control over generation, allowing deeper insights into the data distribution.

Disadvantages of conditonal PixelCNN++ over Resnet18

- Slower in training and generation.
- Requires more memory and compute resources
- Not as effective as CNNs like ResNet for pure classification.

3 Conclusion

In this project, we explored conditional image generation using PixelCNN++ and investigated various conditioning fusion strategies—including early fusion, gated residual fusion, late fusion, and joint fusion. Among these, gated residual fusion demonstrated superior performance in classification accuracy.

We further compared our generative model against a ResNet18 classifier trained on the same dataset. While ResNet18 outperformed PixelCNN++ in pure classification tasks, our model offered generative capabilities, class-conditional sampling, and interpretable outputs, highlighting its versatility beyond classification.

However, our work has several limitations. PixelCNN++ is computationally intensive, especially for high-resolution image generation, and sampling is autoregressive, which slows down inference. Additionally, we observed a drop in performance with certain fusion strategies, particularly early and late fusion, which calls for deeper architectural tuning or normalization techniques.

For future work, we plan to:

- Integrate attention mechanisms for long-range dependency modeling.
- Explore diffusion-based generative models for better scalability.
- Investigate hybrid architectures that combine discriminative and generative pathways to improve both classification and sample quality.

This project demonstrates how conditional PixelCNN++ can bridge the gap between generative modeling and classification, offering a foundation for further research in multimodal learning and robust visual understanding.

Acknowledgment

This project was completed with the help of ChatGPT. In order to see the usages of the ChatGPT, please take a look at this link - Link

References

Salimans, Tim et al. (2017). *PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications*. arXiv: 1701.05517 [cs.LG]. URL: https://arxiv.org/abs/1701.05517.

Saponaro, Sara et al. (Sept. 2023). *Deep Learning based Joint Fusion approach to exploit anatomical and functional brain information in Autism Spectrum Disorders*. DOI: 10.21203/rs.3.rs-3372317/v1.

Appendix

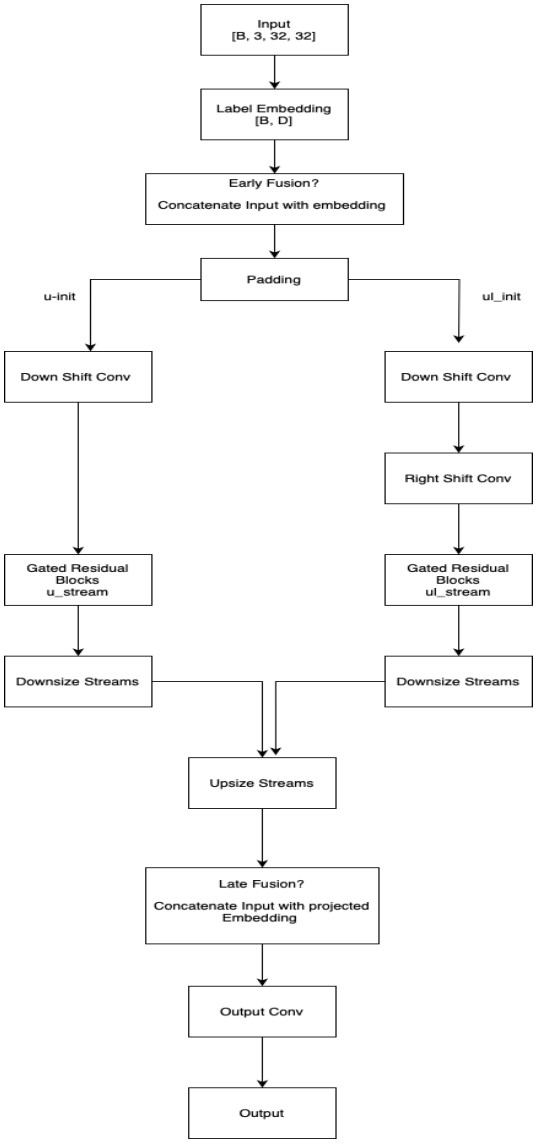

Figure 1: Architecture diagram of PixelCNN++ with conditional fusion.

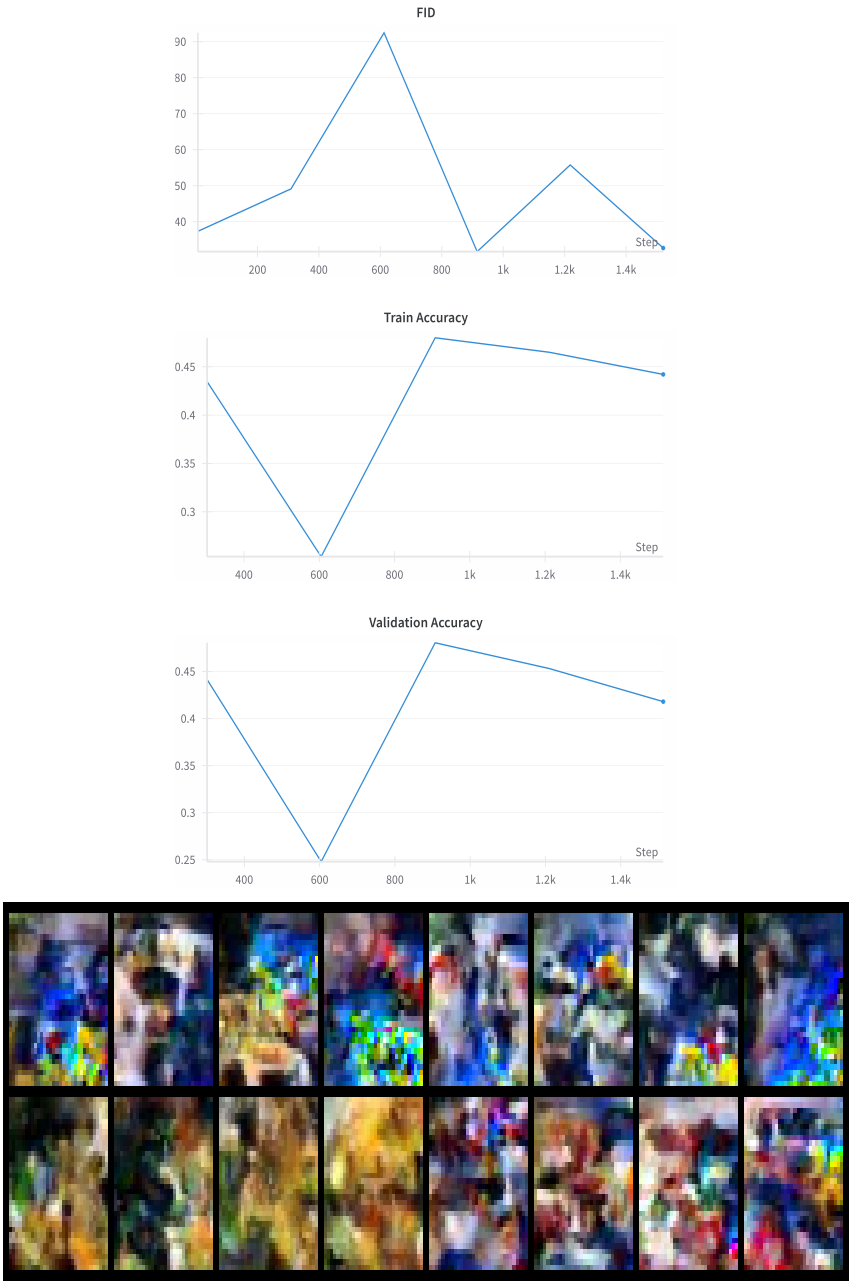

Figure 2: Training and validation accuracy with FID and sample for Early Fusion Model

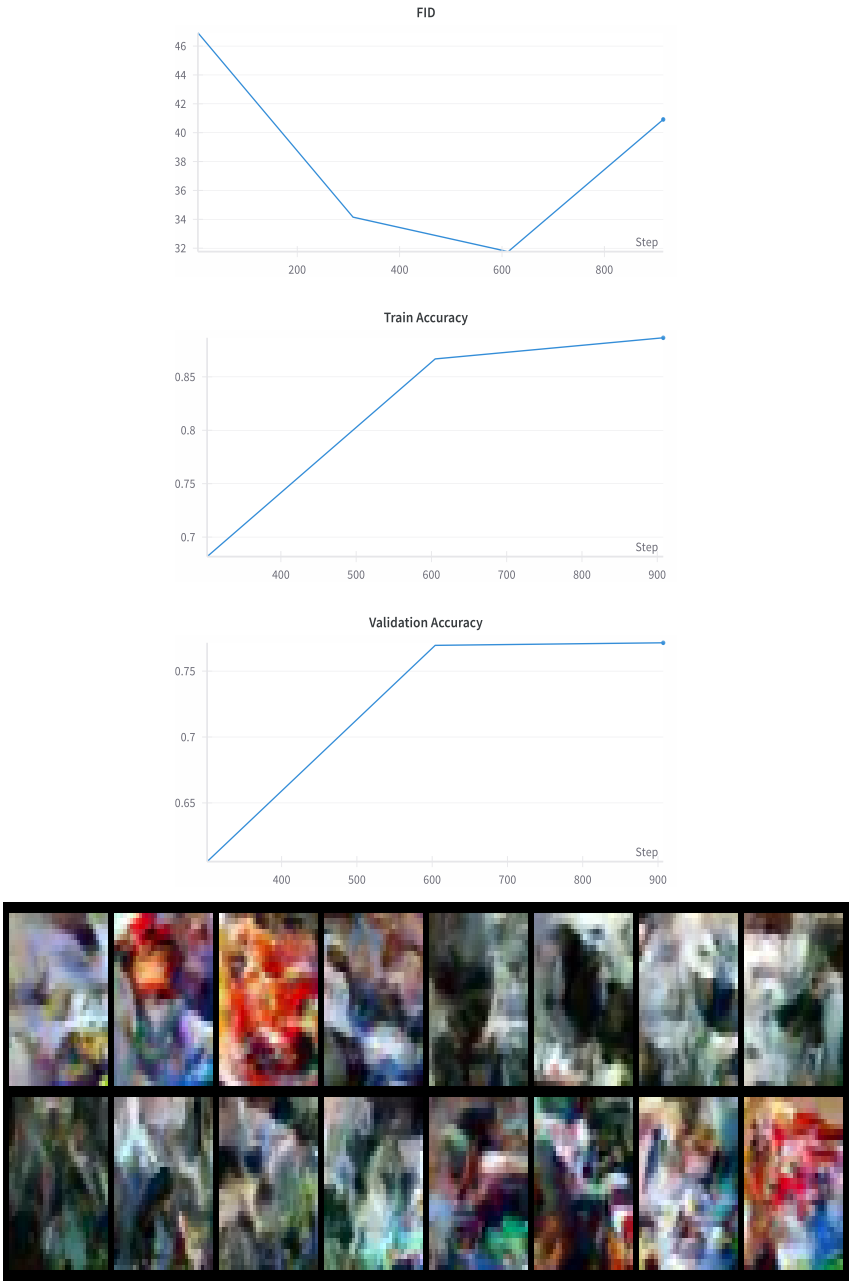

Figure 3: Training and validation accuracy with FID and sample for Gated Residual Fusion Model

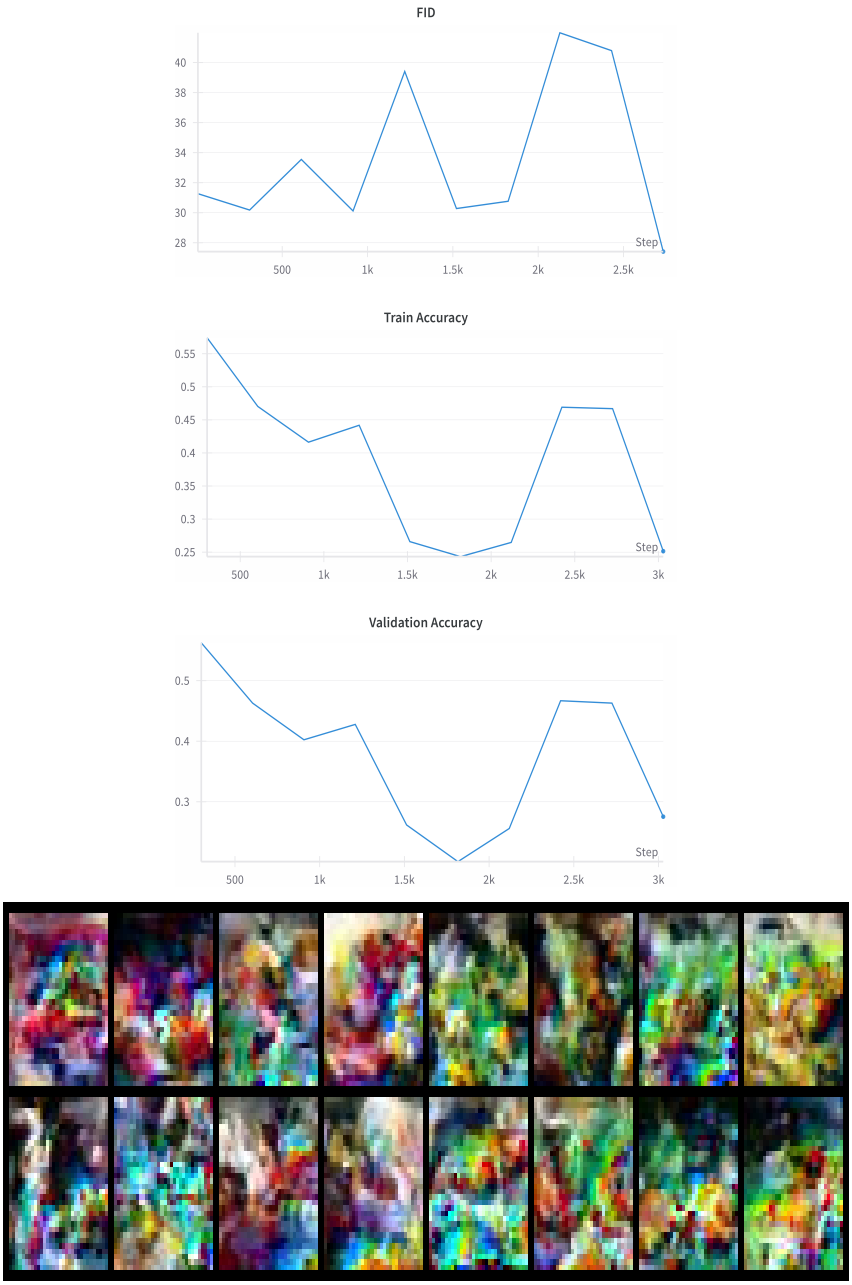

Figure 4: Training and validation accuracy with FID and sample for late Fusion Model

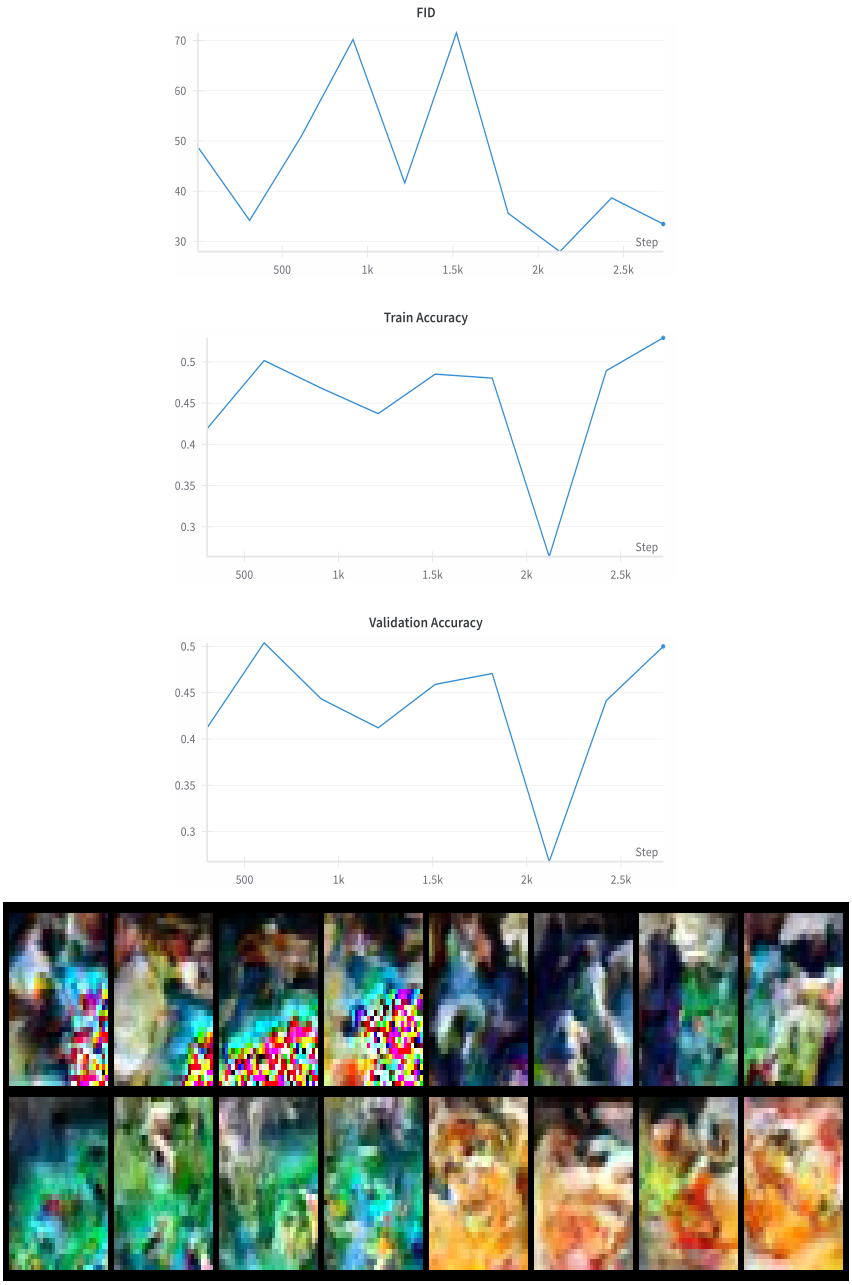

Figure 5: Training and validation accuracy with FID and sample for Joint Fusion Model

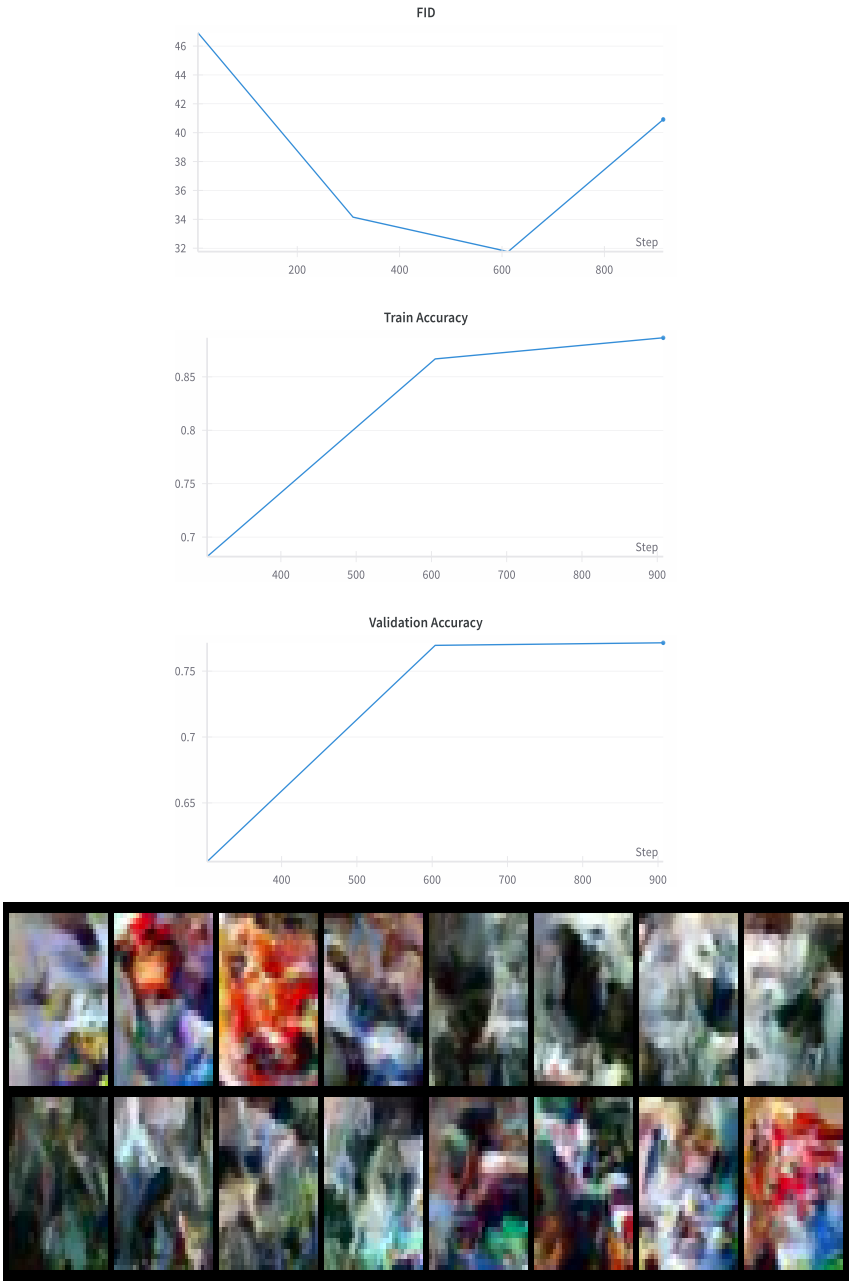

Figure 6: Training and validation accuracy with FID and sample for Main Model (Gated Fusion)

In order to execute the two scripts follow this guide

```
# To run the classification evaluation script
python classification_evaluation.py

# To run the generation evaluation script
python generation_evaluation.py

# To run the resnet model for classification(train, test and validation)
python resnet_18_val.py
```

Figure 7: Guide for running the scripts