אוניברסיטת חיפה
UNIVERSITY OF HAIFA
جامعة حيفا

Pixel Recurrent Neural Network

Experiments

Team:

Abdallah Massarwe Removed ID

Tayma Ibrahem Removed ID

Course:

Deep Learning

Data and Resources

Dataset:

MNIFST

CIFAR-10 (32×32 color images)

Model:

We used the **Pixel RNN Row LSTM** based model to train and evaluate on CIFAR-10 Dataset

We used **Pixel RNN Diagonal BILSTM** to train and evaluate on MNIFST Dataset

Hardware:

We Used the A100 GPU provided by the google collab

Time Constraints:

1 epoch takes ~25 minutes – ROW LSTM – CIFAR (After using A100) so its typically took us 1:30 Hours for 1 epoch even more on complex models

Batch Sizes:

Followed the paper, they used batch size of 16 for training on CIFAR-10 Dataset, also batch size 16 for MNIFST Dataset

Total Epochs:-

Started 20 epochs then added gradually to compare generated images and improvement. But depends on the compution power since we are using google collab and we have limited resources it changes.

Evaluation Metrics used:

We followed the paper and they used

- **Negative Log-Likelihood:** NLL This is the main loss function they used to evaluate the performance of the Model. And bits per dimension [BPD]

Experiments Plan and Description

In the original *Pixel Recurrent Neural Networks* paper, the authors provided specific hyperparameters for training Pixel RNN on the CIFAR-10 dataset. The model architecture they used for CIFAR-10 included:

- **Number of Layers: 12**
- **Hidden Dimensions: 128**

Baseline Performance

The authors reported a Negative Log-Likelihood (NLL) score of 3.06 as their performance metric, which we also adopt as our loss function. This provides a benchmark to compare our implementation's effectiveness and accuracy.

5 right). For CIFAR-10 the Row and Diagonal BiLSTMs have 12 layers and a number of $h = 128$ units. The Pixel-

# layers:	1	2	3	6	9	12
NLL:	3.30	3.20	3.17	3.09	3.08	3.06

Experiment Plan We aim to reproduce the baseline model using the Row LSTM of Pixel RNN and evaluate its performance on the CIFAR-10 dataset. Our plan consists of the following steps:

Model Setup and Training

- Implement the Row LSTM architecture as described in the paper.
- Set hyperparameters exactly as specified:
    - 12 LSTM layers
    - 128 hidden dimensions
    - ReLU activations
    - RMSProp optimizer with learning rate scheduling
- Use the Negative Log-Likelihood (NLL) loss function for evaluation.
- Train the model on CIFAR-10 with the same preprocessing and data augmentation techniques where applicable.

Baseline Experiment Results and performance evaluation CIFAR-10

1. Training Progress and Test Performance

We trained the **Row LSTM** variant of **PixelRNN** on the CIFAR-10 dataset following the exact architecture and hyperparameters specified in the original paper. Below are the key results obtained after **4 epochs** of training:

Epoch 4: 100%|[redacted]| 3125/3125 [18:18<00:00, 2.85it/s, BPD=5.1600, NLL=3.5767]

- **Test Loss (Negative Log-Likelihood, NLL): 3.5767**
- **Learning Rate at Epoch 4: 0.0001**
- **Checkpoint Saved:** checkpoint_epoch4.pt

Comparison with Paper Baseline

The original paper reported an **BPD score of 3.06**, whereas our implementation currently achieves **5.16** at epoch 4. This suggests that further training is needed for our model to converge to the reported performance. Possible factors affecting this difference include:

- Insufficient training epochs (the original paper likely trained for more epochs).
- Slight variations in implementation or optimization settings.

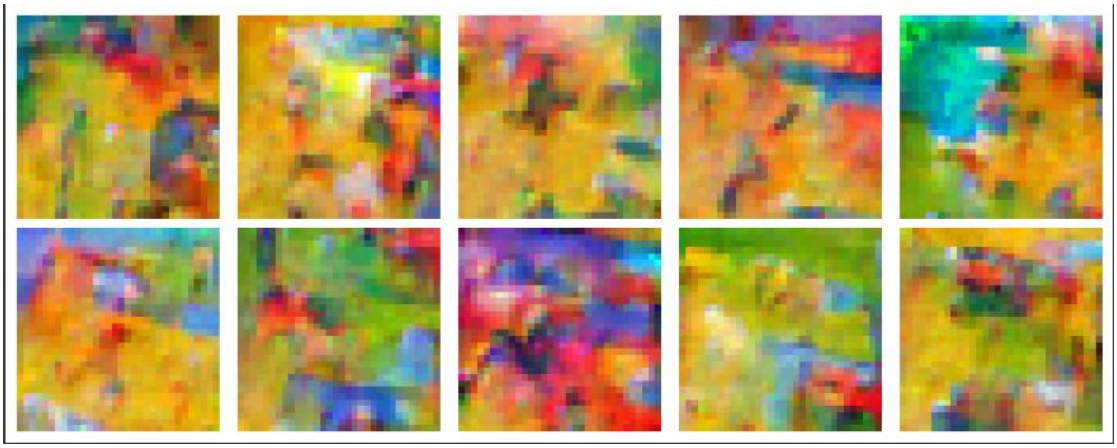
Generated Samples

- We also examined the samples generated by the trained model. Below is an overview of the visual quality and structure of the generated images:
- The image clearly has a structure and still some noise but this indicates that the model need more training epochs

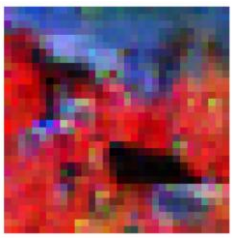
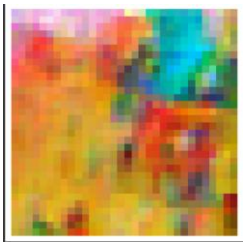
Time taken for 4 epochs on A100 GPU: 2H 28M

Negative Log-Likelihood Loss: 3.735

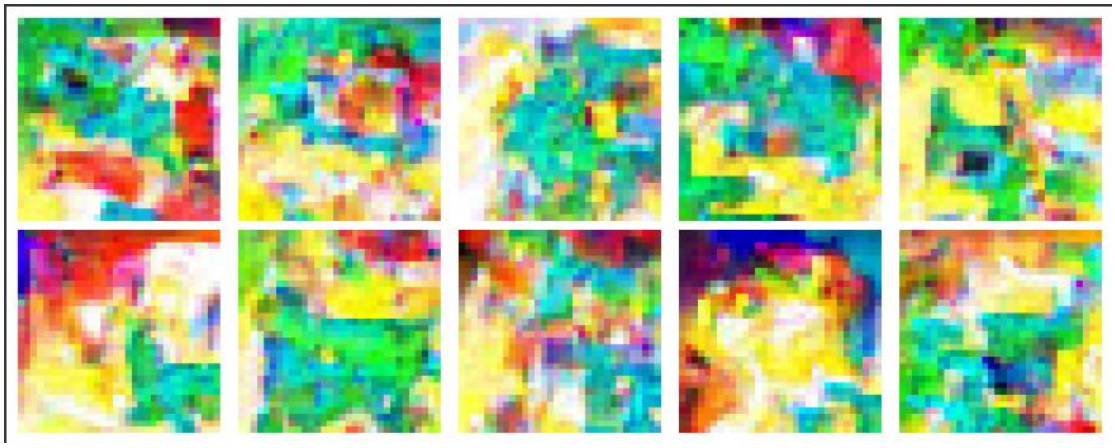
More generated samples at epoch 4



If we look closely, we can start to see emerging shapes and structures, indicating that the model is beginning to learn meaningful patterns. However, to achieve better results and generate clearer images, we need to continue training the model for more epochs.

Metric	Epoch 1	Epoch 4
Bits per dimension (BPD)	6.91	5.2
Negative Log-likelihood (NLL)	4.94	3.6
Generated sample		
Performance	The generated images contain significant noise with no recognizable objects, indicating that the model has not yet learned meaningful representations or captured dependencies effectively.	While the images still appear noisy, there are signs of improvement, suggesting that the model is gradually learning patterns in the data.

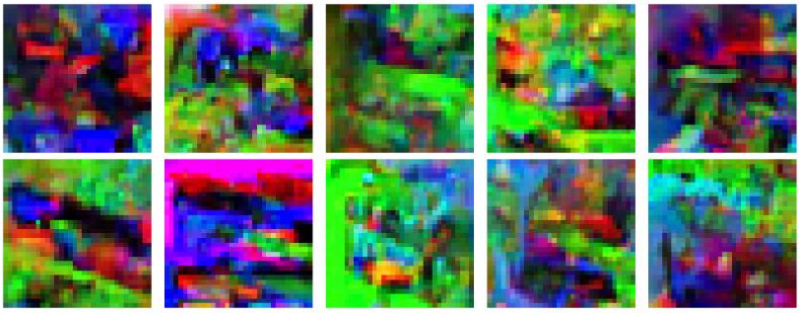
Epoch 5:



Epoch 5: 100%|[redacted]| 3125/3125 [18:22<00:00, 2.84it/s, BPD=5.0970, NLL=3.5330]
Negative Log-likelihood:- 3.53
Bits per dimension:- 5.097

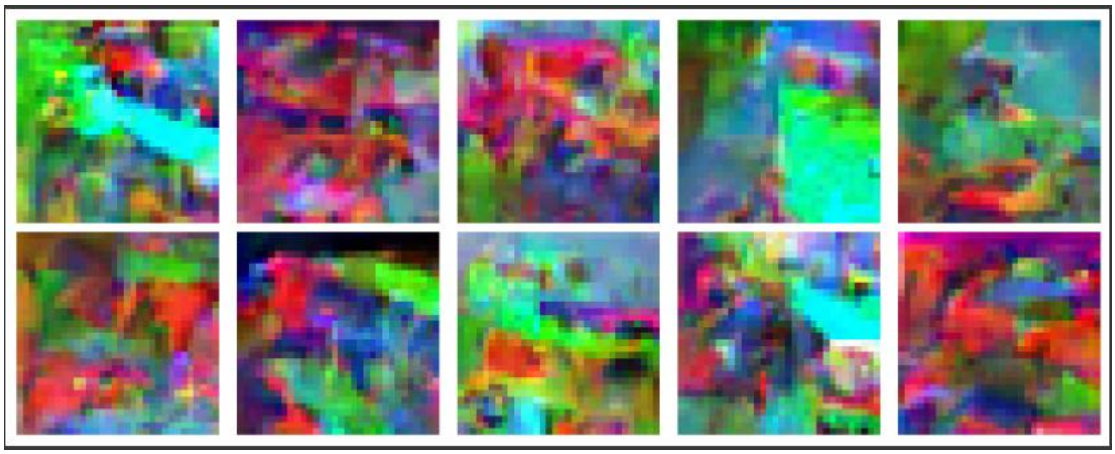
There are still noise in the samples. We are still early.
Now we will tain the model 20 epochs at a time and track the performance and quality of the generated samples

Epoch 15
Generated Samples



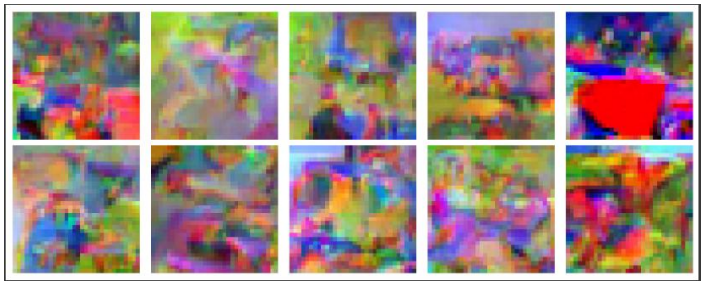
BPD=4.9213, NLL=3.4112

Epoch 20 – Loss BPD=4.8678, NLL=3.3741



Epoch 30

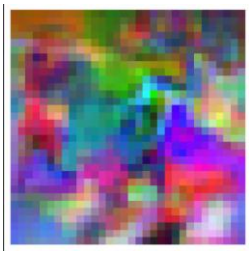
BPD=4.8269, NLL=3.3458

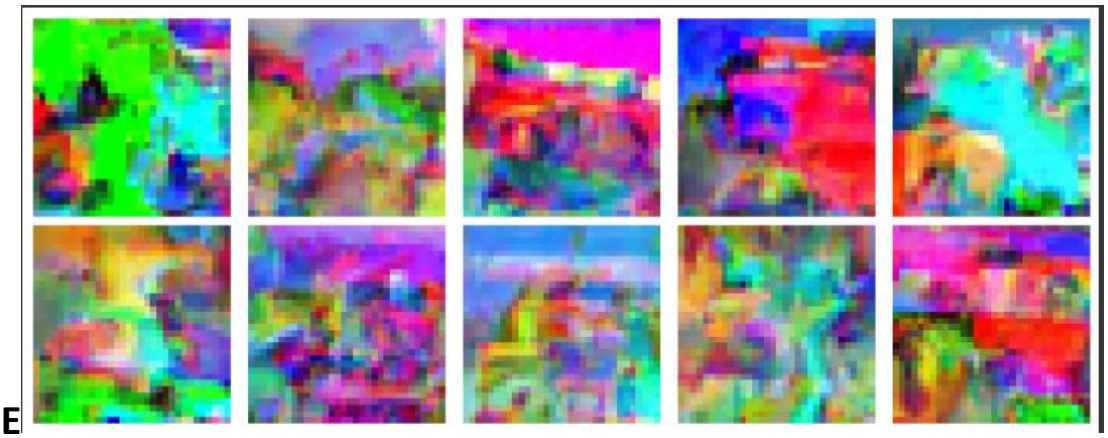


<u>Analysis After 30 Epochs Baseline Experiment:</u>

After 30 epochs of training, the PixelRNN outputs display abstract, highly saturated color patches we can see shapes and objects and still some noise we need to train for more .

Epoch 40:

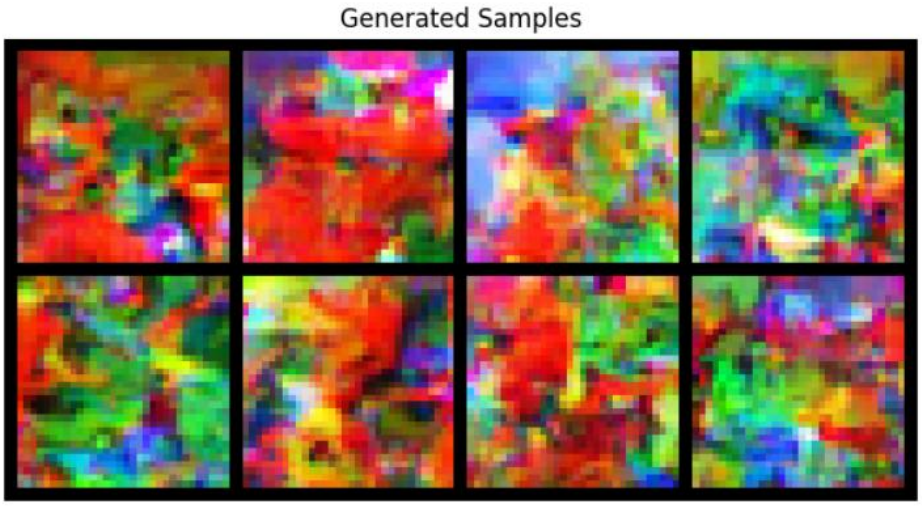




BPD=4.7799, NLL=3.3132

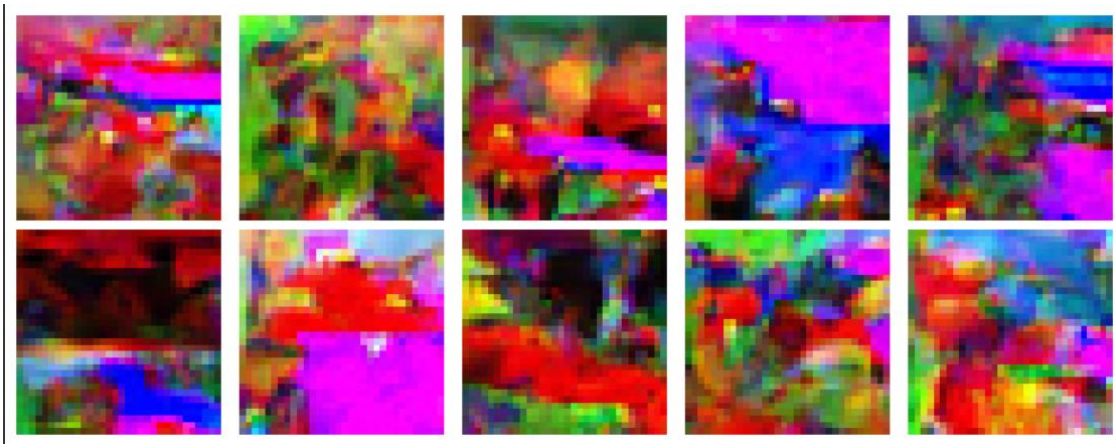
We also ran a training on a the same dataset same model but we wanted to experiment with lower layers 6 instead of 12

Epoch 40 – Lower Layers model



Comparison: We can clearly see that the images are not smooth and they lack structure they aren't so different from the epoch 40 more layers but we can clearly notice that.

Epoch 45:



BPD=4.7567, NLL=3.2971

Analysis:

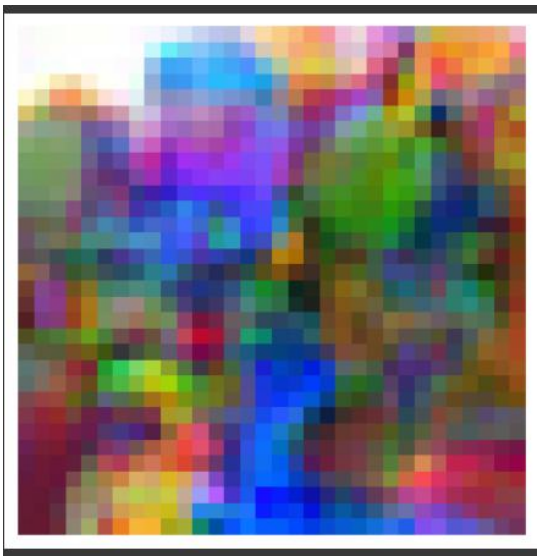
After 45 now the mode is slowly learning we can notice clear shapes but the noise still there the samples are not very clear to the eye but if we look closely we can notice objects which is very promising.

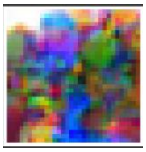
We will continue to train for more epochs and compare.

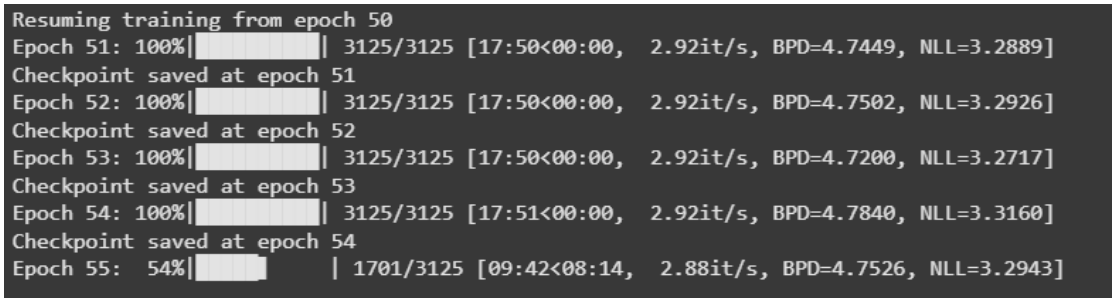
We reached 55 Epochs the model can't learn more the loss kept going up and down for many epochs

We can clearly notice the object in the middle of the generated sample image, and less noise which is a very good sign that the model has learned and converged, but it reached a point where there is no improvement in the loss reduction.

Started to fluctuate up and down meaning the model reached a plateau







We notice that the model has reached the plateau and doesn't learn more at this stage of the training which indicates that more tuning required and optimiziation

Experiment Variants

We will try to improve the current model to reach better results and performance

Plan:-

Since we know the model converges at 3.1~3.3 Negative Log likelihood loss

(4.7~4.6 BPD) We need to enhance its performance by

Enhancing PixelRNN Training and Sampling Quality

Summary

the model struggled to converge beyond a loss of 3.3 and generated poor-quality samples. To improve training stability, generalization, and sampling quality, we implemented a series of optimizations.

We modified these:

Hidden dimensions: 128 (baseline) ➔ 256

Layers: 12

Along with other optimization described below:-

---

1. Optimizer Change: Switching from RMSProp to AdamW

Modification

- Replaced RMSProp with AdamW, which adds decoupled weight decay to improve generalization.

Why?

- AdamW improves weight regularization, reducing overfitting**.**
- Helps avoid exploding gradients in LSTMs, stabilizing training.

Expected Impact

- Faster convergence
- Better generalization → higher-quality sampled images

---

2. Learning Rate Scheduling: Switching from no scheduler to ReduceLROnPlateau

Modification

- Added ReduceLROnPlateau, which adjusts the learning rate dynamically when loss stagnates.

Why?

- ReduceLROnPlateau only lowers LR when loss stops improving, preventing premature slowdown.

Expected Impact

- Prevents early convergence & underfitting
- Ensures learning rate decreases at the right time
- Leads to better image generation stability

---

3. Gradient Clipping for Stability

Modification

- Added gradient clipping to prevent exploding gradients**.**
- Clipping ensures gradients don't exceed a certain norm, improving convergence.

Expected Impact

- Prevents training crashes
- More stable and structured generated images

---

4. Data Augmentation to Improve Generalization

Modification

- Added data augmentation to introduce variability in the dataset.

Why?

- Prevents the model from memorizing training images.
- Helps model learn robust pixel dependencies.

Expected Impact

- Diversity in generated samples
- Better generalization → more realistic images

Final Summary

Modification	Why We Did It?	Expected Benefit
RMSProp → AdamW	Improve weight decay & generalization	Better sampling results
NoScheduler → ReduceLROnPlateau	Prevents premature LR decay	More stable training
Gradient Clipping	Avoids exploding gradients in LSTMs	Prevents training crashes
Data Augmentation	Improve robustness of training	More realistic generated images

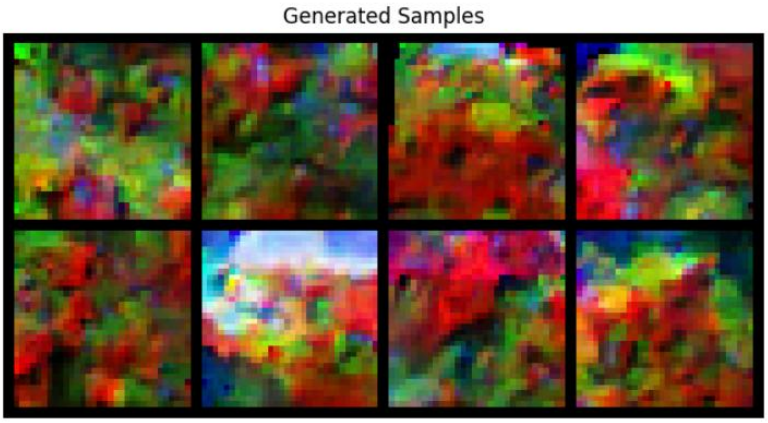
---

Next Steps

- We are going to Re-train the model with these optimizations
- Generate new samples
- Evaluate if samples have improved color consistency & quality
- Compare results before & after the modifications

Epoch 4:

Generated Samples:

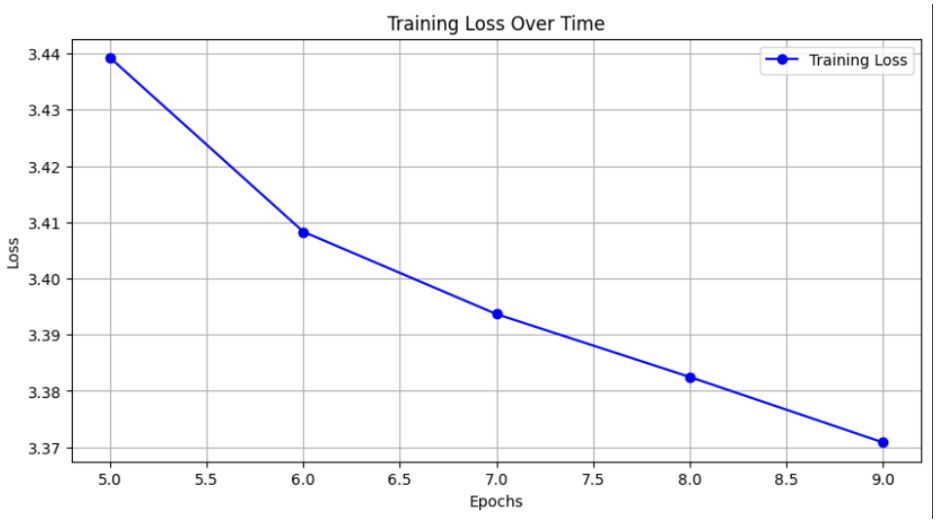


Loss: 3.4933 nats
BPD: 5.0398 bits/dim

Comparing With Epoch 4 results from the previous model (before the optimization and tuning)

Loss: 3.6 nats
BPD: 5.2 bits/dim

Analysis:-
We can clearly see the improvement at epoch 4
Improved Model had 0.1 less NLL loss and ~0.17 less BPD compared to the baseline model which indicates that the model performance is very promising
We will continue to train the model and compare.
the samples at this stage exhibit noticeably more coherent color transitions compared to previous epochs. While still abstract, these images suggest the model is beginning to capture more structured patterns rather than purely random noise.



Graph Analysis:

Downward Trend: The training loss is steadily decreasing over epochs, indicating that the model is successfully learning from the data.

Consistent Improvement: No sudden fluctuations or spikes, suggesting stable convergence.

Smooth Slope: The loss decrease is gradual, meaning the learning rate is likely well-tuned.

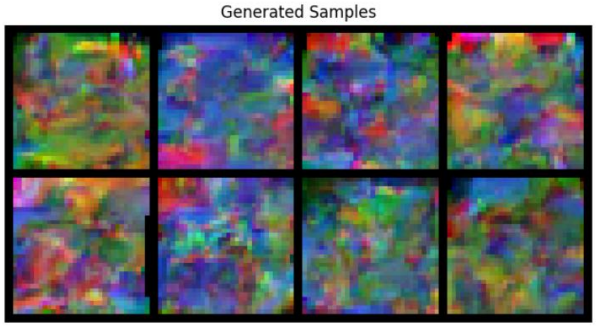
Next Steps:

Check if BPD follows a similar trend.

Continue training to see if loss stabilizes or if additional improvements are possible.

Monitor if loss plateaus too early, which might indicate the need for further hyperparameter tuning.

Images Generated at epoch 10



Analysis of Training Progress and generated samples.

Loss & BPD Values:

Average Loss = 3.3709 nats

BPD = 4.8631 bits/dim

Observation:

The loss continues to decrease, indicating ongoing learning.

Blurry samples suggest that while the model is capturing overall shapes, finer details are still missing.

The BPD is relatively high, meaning the model hasn't yet fully learned to represent the data compactly.

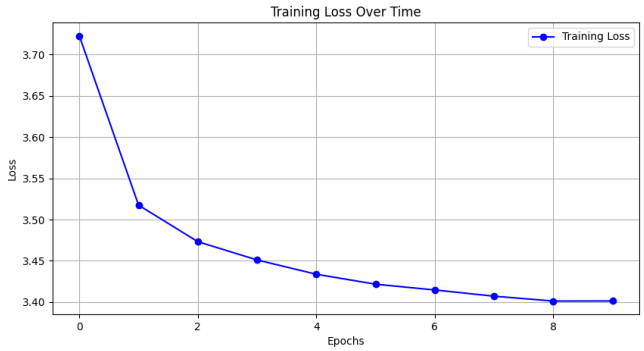
Analysis of Training Loss Curve

- **Rapid Initial Decrease:**

  The loss drops significantly in the first few epochs, indicating that the model is learning fast initially.
  This suggests that the optimizer is effectively updating weights early on.

- **Slower Convergence After Epoch 4:**

  The loss continues to decrease but at a much slower rate after epoch 4.
  This could indicate that the model is approaching a local minimum.



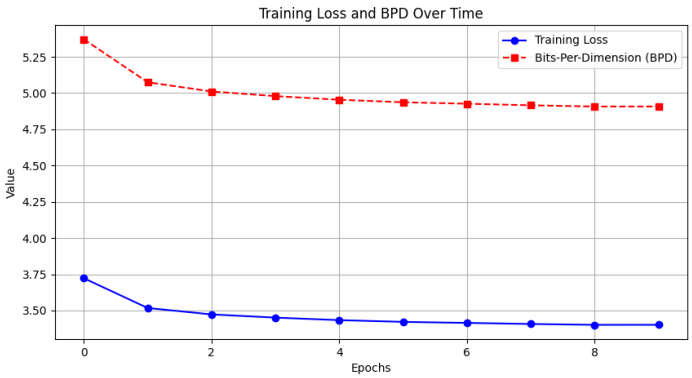
Bits-Per-Dimension (BPD) (Red Dashed Curve)

BPD is also decreasing, but at a slower rate compared to the loss.

This means the model is learning to encode data more efficiently, but there is room for further improvement. Around epoch 6, BPD plateaus, which might suggest that more training or architectural improvements are needed.

Observations:-

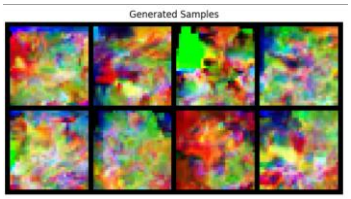
The decreasing loss aligns well with the decreasing BPD, showing that the model is improving overall.

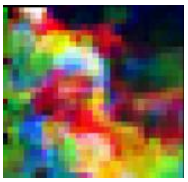


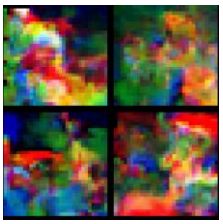
Epoch 20: Early Training Insights

At **Epoch 20**, we begin to notice structured object formations in the generated samples.

- **Observations:**
    - A **duck** is clearly visible in the **first sampled image**.
    - Other **recognizable objects** appear in the remaining three samples.







Loss Analysis at Epoch 30:

By **Epoch 30**, we see that the training loss **fluctuates**, indicating that the model is reaching a **plateau point**. This suggests that **learning progress has slowed down** due to the dataset size limitation.

- **Observations:**
    - Loss decreases initially but begins **fluctuating after epoch 27**.
    - The model is **struggling to improve further**, likely due to the **small dataset size**.

Findings

- The model shows **early-stage learning success** with clear object formations.
- However, it **fails to generalize well** beyond a certain point due to **data limitations**.

Next Experiment: Enhancing Performance + More Layers

1. Optimizing RMSprop & Learning Rate

We train the model 5 epochs at a time so we can compare and evaluate

Current Issue

- The model is struggling to converge at lower loss values.
- AdamW did not yield better results in previous trials.

Plan

1. **Learning Rate Tuning**
    - Start with a baseline LR (1e-3) for RMSprop change to 5e-4
2. **RMSprop Parameter Tuning**
    - **alpha**: The smoothing constant (defaults to 0.99). We put 0.9
    - **weight_decay**: We typically use 1e-4. Will check if this tuning helps.
    - **eps**: 1e-6.
3. **Scheduler Usage**
    - Continue using ReduceLROnPlateau to dynamically lower the LR when the loss plateaus.
    - Monitor the training loss and bits-per-dimension (BPD) closely to determine if the LR adjustments are helping.

```
model = PixelRNNRowLSTM(in_channels=3, hidden_dim=128, num_residual_blocks=15, height=32, width=32).to(device)
```

```
optimizer = optim.RMSprop(model.parameters(), lr=5e-4, alpha=0.9, eps=1e-6, weight_decay=1e-4)
```

Now we run the model training 5 epochs at a time with saving checkpoints

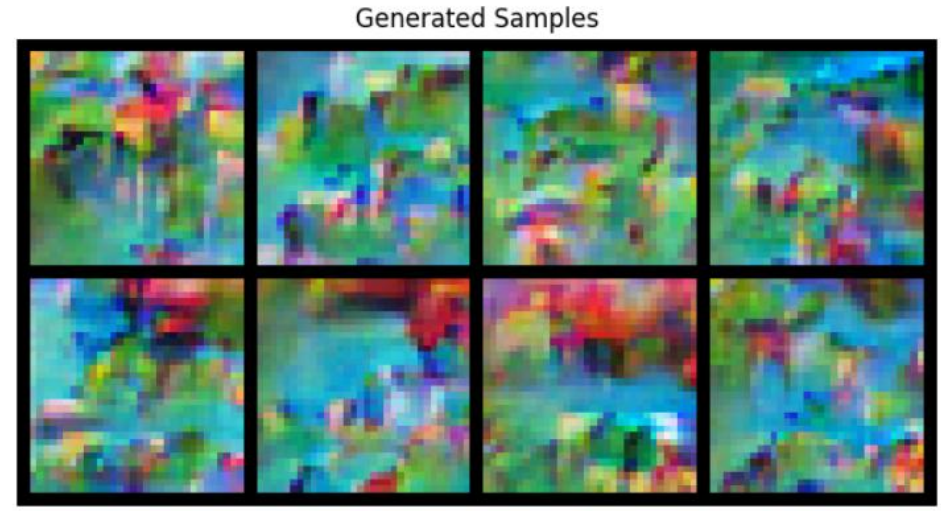
So we can evaluate the model and performance.

Progress:-

Epoch 5:

Average Loss = 3.5582 nats, BPD = 5.1334 bits/dim

Generated Images:



The generated samples from the first 5 epochs show **some promising progress**, but they are still in the **early stages of learning**. Here are some observations:

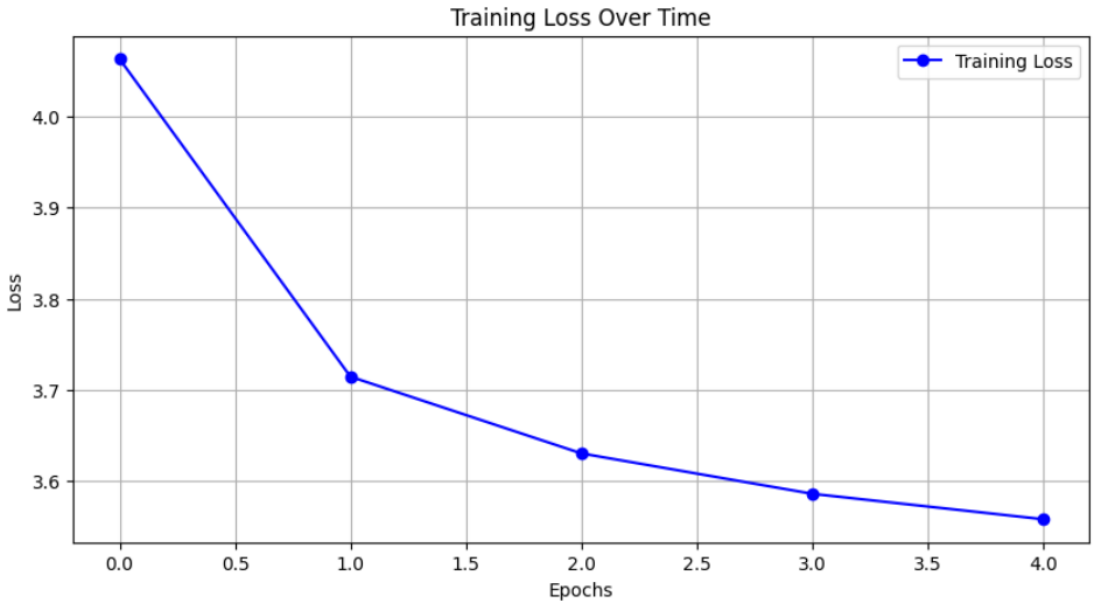
Positives

- **Color Diversity**: The model is capturing a wide range of colors, which suggests it is learning the distribution of pixel intensities well.
- **Rough Structure Formation**: There are hints of emerging patterns, though they are still abstract and blurry.

Challenges

- **Lack of Sharpness**: The images remain heavily pixelated, indicating that the model is still struggling to capture fine details.
- **No Clear Object Formation**: At this stage, the samples are mostly **blobs of color** without recognizable features.
- **Mode Collapse Risk**: Some images have **similar color patterns**, which could indicate that the model is not yet learning diverse structures properly.

Loss Graph:



Observations

Steady Decrease in Loss

- The loss is **consistently decreasing**, which is a positive sign that the model is learning.
- It starts above **4.0** and drops to **around 3.5** by the 4th epoch.

Fast Initial Drop

- The sharp drop between **epoch 0 and epoch 1** indicates that the model is quickly learning the basic data distribution.
- This is expected in many deep learning models, where the **biggest improvements happen early** before refinement slows down.

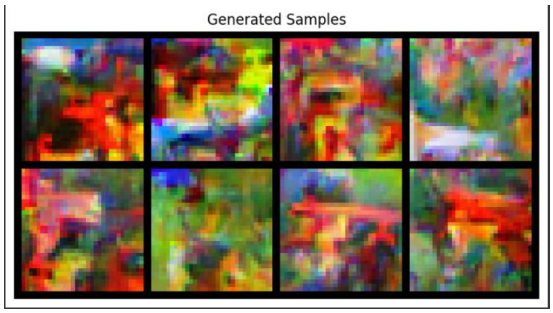
Next?

We will keep training for another 5 epochs and compare.

Progress:

Epoch 10:

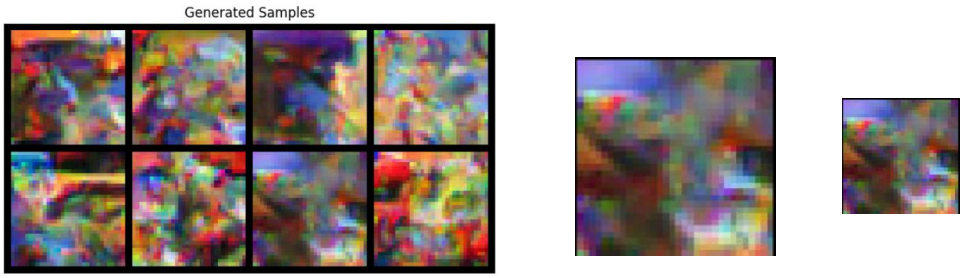
Average Loss = 3.4811 nats, BPD = 5.0222 bits/dim



From the training logs, We can see that by epoch 10 the model is still around **5.02 bits/dim** on CIFAR-10, which is better than random (8 bits/dim) but still fairly high compared to typical state-of-the-art PixelRNN/CNN results (which can go down toward ~3.0 bits/dim). The generated samples reflect that: they're fairly abstract "color blotches" rather than coherent objects.

We will train till convergence

Epoch 15:-



Analysis of Generated Samples

The sampled images shows eight image samples produced by our 15-block PixelRNN model trained on CIFAR-10. While the samples exhibit some local color consistency—evident in patches of similar hues, they are more smoother and visible.

Local Correlations:

- The model learns to place similar or smoothly transitioning colors next to each other.
- Small clusters of color indicate that the network captures short-range dependencies well.

Increased Local Detail:

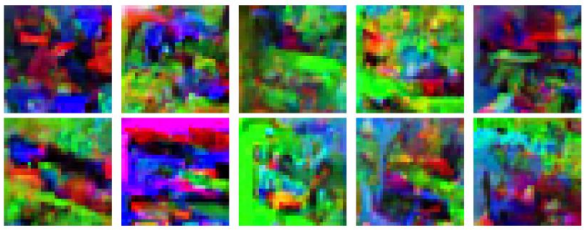
- We observe tighter clusters of color and more pronounced edges or patches, indicating that the network has improved its ability to capture small-scale structures.

Using this 15 Blocks Variant has improved the results.

Comparison

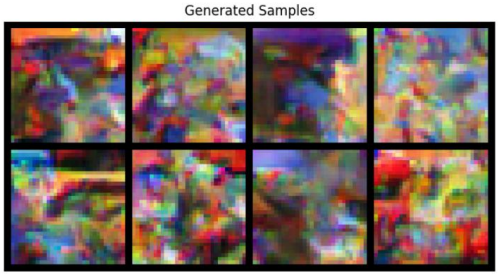
Baseline Experiment

Epoch 15:-



Variant experiment – improved model.

Epoch 15:-



Baseline Experiment (Epoch 15)

- **Visual Characteristics:**
    - The generated samples exhibit large, blotchy patches of color with only vague transitions.
    - While the model has learned some local color correlations, it has not formed recognizable shapes or coherent objects.
    - Bright, saturated regions dominate the images, suggesting the network is capturing broad color distributions but struggling to assemble them into meaningful patterns.
- **Possible Causes:**
    - Limited model depth or fewer LSTM/convolutional blocks can reduce representational power.

Variant Experiment – Improved Model (Epoch 15)

- **Visual Characteristics:**
    - The samples show more refined local details compared to the baseline. Color transitions are slightly smoother, and patches appear less uniformly blotchy.
    - Some subtle hints of structure are emerging, though the images are still largely abstract.
    - The overall color palette is somewhat more balanced, indicating that the model is better at distributing color intensities.
- **Model Improvements:**
    - Additional blocks or layers, residual/skip connections, or gated mechanisms can help the network capture more complex patterns.
    - A refined training strategy (e.g., longer training, mixed-precision, or better learning-rate scheduling) might also have contributed to a lower loss/bits-per-dim.

Although the training takes a lot of time and added blocks add to that, the model can capture better dependancies and cocntext while being more deep and complex.

increasing the number of blocks significantly extends training time, a deeper and more complex model generally captures richer dependencies and context. This is evident in the improved local detail and more coherent color transitions in the generated samples. Despite the computational cost, deeper architectures typically achieve lower bits-per-dimension and higher sample quality, demonstrating the trade-off between training efficiency and modeling capacity.

Trade Off

More complex model -> More layers - > more hidden dimensions -> more computation cost -> Better Quality samples ->More context.

Less complex model -> less layers - > less hidden dimensions -> lower computation cost -> lower Quality samples ->lower context.

Whats next?

We will keep training the model and track the results.

Conclusion

In this project, We explored two different autoregressive models for image generation on the CIFAR-10 dataset: a **Baseline model** and an **Improved model**. The Baseline version showed that a PixelRNN-style network can indeed learn local color correlations, but the samples it produced were still quite “blotchy” and didn’t form any recognizable shapes.

In contrast, the Improved model, which added more layers and refinements—led to samples with smoother color transitions and slightly better local structure. However, training took significantly longer, and the generated images still weren’t completely coherent. This underscores a central trade-off: **while deeper, more complex models often learn richer dependencies and context, they also demand more computational resources and time**.

Overall, these experiments confirm that increasing model depth and complexity can yield more realistic samples, but practical considerations like training duration and hardware constraints play a big role in deciding how far we can push the model. In the future, We plan to experiment with even more advanced architectures (Gated PixelCNN or Diagonal BiLSTM), longer training schedules, and mixed-precision techniques to see if we can reduce bits-per-dimension further and produce sharper, more recognizable images.