Application of Long Short-Term Memory (LSTM) Autoencoder for Image Reconstruction Using CIFAR-10 Dataset

by

Muhammad Ayob

20442409

Submitted in partial fulfilment of the requirements for the degree

Hons BIT (Information Systems)

in the

FACULTY OF ENGINEERING, BUILT ENVIRONMENT AND INFORMATION TECHNOLOGY

at the

UNIVERSITY OF PRETORIA

Study leader:

Dr. Mike Nkongolo Wa Nkongolo

Date of submission

7 October 2024

TABLE OF CONTENTS

LIST OF FIGURES

Application of Long Short-Term Memory (LSTM) Autoencoder for Image Reconstruction Using CIFAR-10 Dataset

ABSTRACT

This study explores the application of a Long Short-Term Memory (LSTM) based autoencoder for reconstructing blurred images from the CIFAR-10 dataset. Image quality improvement is a task which works well with autoencoders being used on tasks like image compression, noise reduction as well as reconstruction. We evaluate the effectiveness of the LSTM autoencoder at recovering relevant image features on the CIFAR-10 dataset of 60,000 32x32 pixel images from 10 different classes. The dataset was subject to Gaussian blur to simulate real world image degradation and trained LSTM autoencoder to reconstruct clear images from the blurred inputs. The LSTM autoencoder was trained and able to capture both spatial and temporal dependencies within the image data and produce high quality reconstruction of these are key image details. Final values of the Mean Squared Error (MSE) loss were small, indicating successful reconstruction with low error. The results show the model denoising images efficiently, and future improvements with more advanced architectures, such as variational autoencoders, or Generative Adversarial Networks (GANs), may even better sharpen the reconstructed images.

Keywords: LSTM Autoencoder; Image Processing; Image Reconstruction; CIFAR-10; Machine Learning; Denoising Autoencoder

1. INTRODUCTION

Machine learning and deep learning methods have developed so rapidly in the past years that they have brought considerable improvement in several tasks including image reconstruction, denoising and compression. Autoencoders hold a key role among these approaches because they allow efficient representations of input data that are learned. Unsupervised neural networks are autoencoders that attempts to minimize the difference between the original input and its reconstruction in a latent space, and are called autoencoders (Vincent et al., 2008). Traditional Convolutional Neural Networks (CNNs) have been effective for image data, but in this experiment, we employ an LSTM-based autoencoder, which is typically used for sequential data, to reconstruct blurred images from the CIFAR-10 dataset.

When referring to computer vision tasks, the CIFAR-10 dataset is extensively used: it includes 60,000 32x32 pixel images distributed to 10 classes (airplanes, automobiles, birds etc.) (Krizhevsky, 2009). However, it offers a range of images to run through machine learning models. For this project, we needed to run Gaussian blur on CIFAR-10 images to simulate degradation, and train an LSTM autoencoder for reconstruction of the CIFAR-10 images. The goal is to assess the effectiveness of this model in recovering important image features, such as edges and textures, from degraded images. In this approach we use the LSTM's ability to exploit spatial and temporal dependencies in image data.

2. METHODOLOGY

Dataset Preparation and Exploratory Data Analysis (EDA)

We loaded CIFAR-10 dataset using TensorFlow's Keras datasets module. There is a train set of 50,000 images and a test of 10,000 images. The first step to do was to carry out Exploratory Data Analysis (EDA) to understand better what we had. Then, we examined the shape of the data, confirming that the training data has a shape of (50000, 32, 32, 3), and the test data has a shape of (10000, 32, 32, 3). The number of images, height, width, three RGB (red, green, blue) colour channels are represented by these shapes.

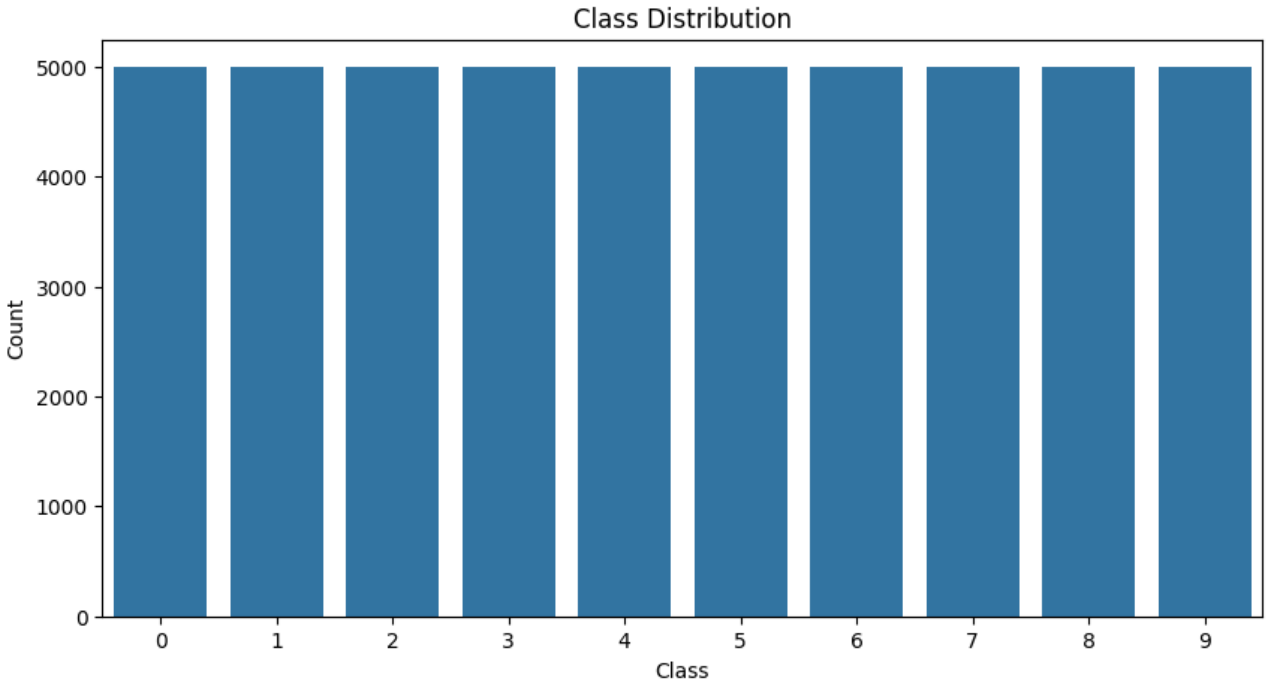

Figure 1: Class Distribution of dataset

Figure 1 displays the class distribution of the dataset. The class distribution was plotted and revealed that the dataset is balanced, with approximately 5,000 images in each of the 10 classes.

Sample images were also visualized to better understand the nature of the data. We selected 10 random images from the training set, representing each class, and displayed them to ensure a thorough understanding of the dataset's content.

Figure 2: Sample Images from the dataset

Data Preprocessing

The next step was to preprocess the images. In order to simulate image degradation, gaussian blur was used to both the training and test sets, using the gaussian effect library in SciPy. Gaussian blur is an inverse operation of a filter which averages out pixel values thus simulating real world noise or distortion (Quanlin et al., 2022; Sankaran et al., 2017). The autoencoder was trained on blurred versions (automatically created for the study by this step) of the original images used as input data.

Figure 3: Comparison of the original vs blurred images before training

Pixel values were normalized to the range [0, 1] to increase the convergence rate of neural networks as uniformity of data distribution is ensured. Next the images were then reshaped (from 32x32x3 (height, width, RGB channels) to 32x96. Since LSTMs process data in a sequential manner, which the flattened pixel rows represent in the model's input sequence, the model needed to reshaped in a way.

Pixel Intensity Distribution and Summary Statistics

```
Mean per channel (R, G, B): [125.30691805 122.95039414 113.86538318]
Standard deviation per channel (R, G, B): [62.99321928 62.08870764 66.70489964]
```

To get to know the dataset, we calculated the mean and standard deviation of pixel intensity values about the three colour channels (Red, Green and Blue). Approximately 125.3, 122.9 and 113.9 mean pixel intensities were for red, green and blue channel respectively. These channels had a standard deviation between 62.9 and 66.7, indicating a somewhat varying distribution of the intensity among the channels of the dataset. We then plotted pixel intensity distributions of each colour channel to

better visualize this: how pixel values 'spread out' across the images in the dataset.

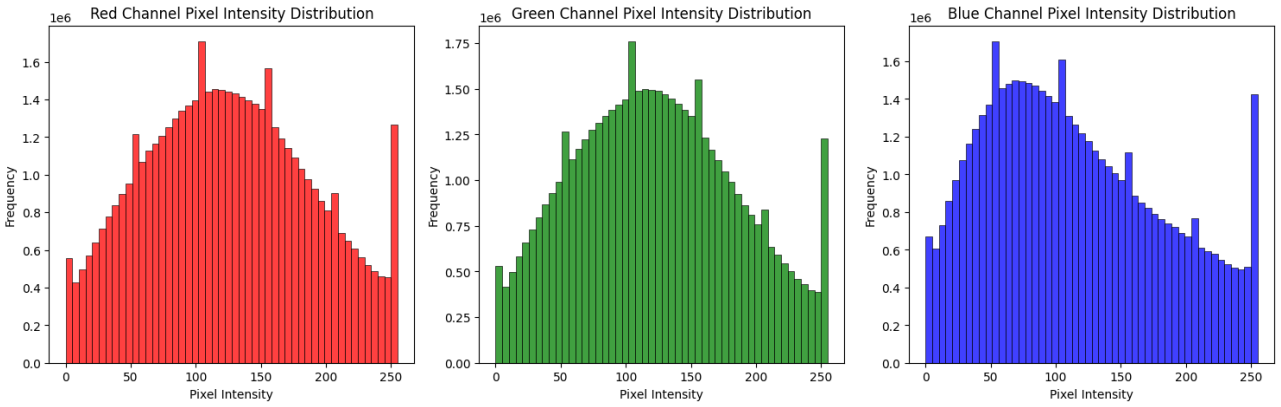

Figure 4: Histograms for the RGB channel intensity ditributions

Shown in figure 4 is the histograms of Red, Green and Blue channel intensity distributions. It helps to illustrate how these pixel intensities vary relative to one another, and also how balanced they are.

3. MODEL AND CODE EXPLANATION WITH RESULTS

LSTM Autoencoder Architecture

For this task an LSTM autoencoder is developed with an encoder-decoder architecture, in which the LSTM layers can be seen as processing sequential image data. Although LSTMs are traditionally used for temporal sequence data (such as time series or natural language), we have adapted them to image data by treating the rows of the image as sequences of pixel intensities. It allows the model to learn spatial dependencies (within rows) and inter-row correlations thus critical for reconstruction of more detailed images structures.

In the autoencoder, we have one part (encoder) which operates to compress the input data (blurred image) into a low dimensional self-representation in latent space. The resulting encoding runs dramatically faster compared to traditional encoding, significantly reducing the dimensionality of the input data; while simultaneously preserving the most important 'features' that bear on classifying the data: What discards noise and redundant info and what retains edges, textures, and contours. Here, the encoder consists of two stacked LSTM layers:

128 units at LSTM layer 1 with return sequences on. The first layer processes the input pixel rows sequentially, capturing dependencies inside each row, as well as across rows adjacent to each other. 128 units are large enough to capture and compress high order spatial patterns in the image.

The representation learned in the first LSTM layer is further refined then with a **LSTM layer 2 (64 units).** Its potential in this construction is that it also returns sequences, so it can in turn pass on a slightly revised form of the encoded sequence to the decoder. This indicates the progressive reduction in dimensionality as image is encoded into small form, by the smaller number of units.

The autoencoder also has one part called decoder, which looks just like the encoder and reconstructs the original image from the latent space representation. Decoder is constructed of TimeDistributed Dense layer that outputs pixel values sequence per each row of the reconstructed image. The TimeDistributed layer lets us apply the dense layer one row at a time (or pixel row), in sequence, until we reconstruct the image row by row. The autoencoder outputs the same shape as the input (32x96) so the image reconstructed has valid dimensions and can be reshaped back to 32x32x3 (height, width and RGB channels).

Model Summary for deeper understanding:

```
Model: "sequential_3"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ lstm_12 (LSTM)                       │ (None, 32, 128)             │         115,200 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ lstm_13 (LSTM)                       │ (None, 32, 64)              │          49,408 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ time_distributed_3 (TimeDistributed) │ (None, 32, 96)              │           6,240 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 170,848 (667.38 KB)
 Trainable params: 170,848 (667.38 KB)
 Non-trainable params: 0 (0.00 B)
```

This model contains 170,848 trainable parameters, much lighter than more complex architectures, such as GANs. The resultant model is very compact (and therefore easy to train) while still being able to reconstruct images with high fidelity.

For this task, we picked the Mean Squared Error (MSE) loss function for it directly measures the difference between the pixel intensities of the original and reconstructed images. The better the reconstruction will be the closer the MSE is to zero. During training, the model was trained using the Adam optimizer which permitted efficient handling of sparse gradients and dynamically controlling of the learning rate. The combination of these will offer convergence and will avoid vanishing or exploding gradients observed in deep networks.

Model Training and Loss Evaluation

The LSTM autoencoder was trained by passing blurred images to the model and original images as a target output. We trained the model for 20 epochs, which seemed appropriate with the observed convergence of the training and validation losses. We chose a batch size of 64 in order to allow the model to train on batches of images as this helps reduce the amount of time that each epoch takes without overloading memory.

In training, the MSE was decreasing over time. Validation loss and training loss were recorded for each epoch during training. The metric used to train your model will be training loss which shows how well the model is learning the training data. This validation loss is a measure for how good the model generalizes to data it hasn't seen before (the test set in this case).

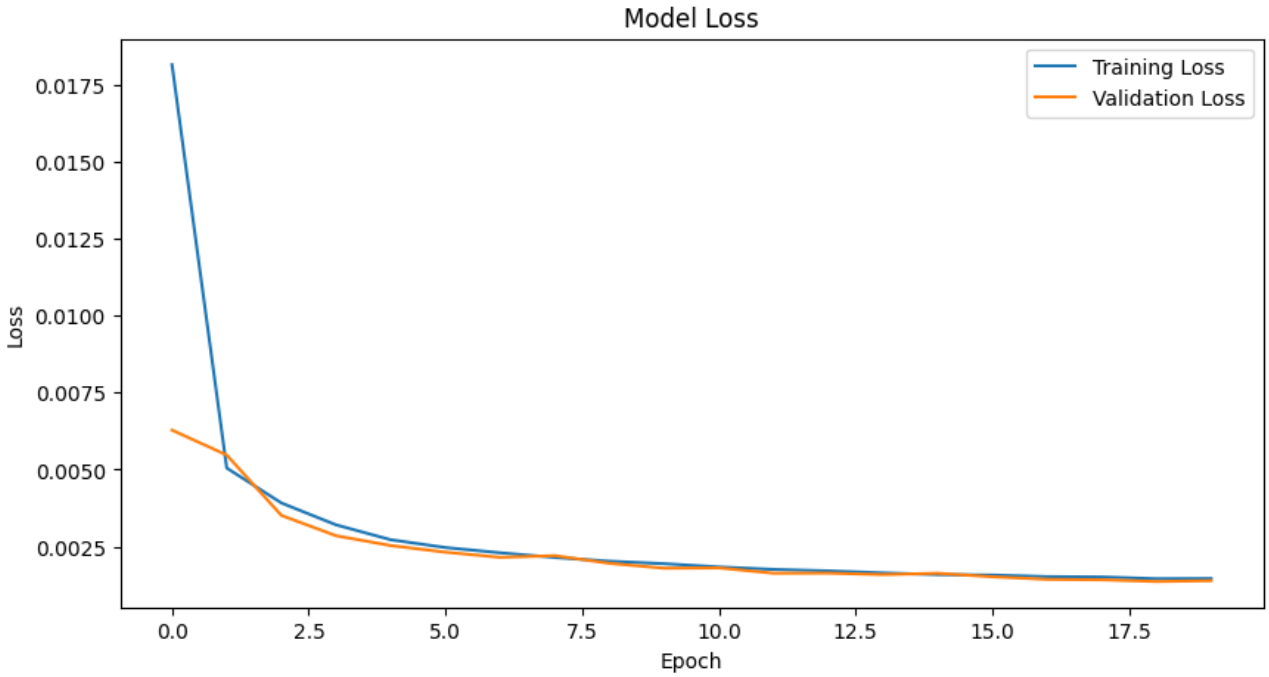

Figure 5: Model Loss

In Figure 5, we have loss curves which depicted an initial training loss of 0.0175 and then steadily decreasing to approximately 0.0015. Similarly, the validation loss dropped from around 0.0063 to 0.0014. We observe that the training and validation loss are near similar with no significant overfitting observed during training, indicating the model is generalizing well to unseen data.

Model Training output:

```
Epoch 1/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 82s 124ms/step - loss: 0.0431 - val_loss: 0.0063
Epoch 2/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 85s 130ms/step - loss: 0.0056 - val_loss: 0.0055
Epoch 3/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 80s 127ms/step - loss: 0.0041 - val_loss: 0.0035
Epoch 4/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 80s 124ms/step - loss: 0.0033 - val_loss: 0.0028
Epoch 5/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 79s 126ms/step - loss: 0.0028 - val_loss: 0.0025
Epoch 6/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 77s 123ms/step - loss: 0.0025 - val_loss: 0.0023
Epoch 7/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 82s 124ms/step - loss: 0.0023 - val_loss: 0.0021
Epoch 8/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 82s 124ms/step - loss: 0.0022 - val_loss: 0.0022
Epoch 9/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 83s 127ms/step - loss: 0.0021 - val_loss: 0.0020
Epoch 10/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 77s 122ms/step - loss: 0.0020 - val_loss: 0.0018
Epoch 11/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 82s 123ms/step - loss: 0.0019 - val_loss: 0.0018
Epoch 12/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 82s 123ms/step - loss: 0.0018 - val_loss: 0.0016
Epoch 13/20
625/625 ━━━━━━━━━━━━━━━━━━━━ 84s 126ms/step - loss: 0.0017 - val_loss: 0.0016
```

```
Epoch 14/20
625/625 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82s 127ms/step - loss: 0.0017 - val_loss: 0.0016
Epoch 15/20
625/625 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79s 123ms/step - loss: 0.0016 - val_loss: 0.0016
Epoch 16/20
625/625 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82s 123ms/step - loss: 0.0016 - val_loss: 0.0015
Epoch 17/20
625/625 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82s 123ms/step - loss: 0.0015 - val_loss: 0.0014
Epoch 18/20
625/625 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 83s 125ms/step - loss: 0.0015 - val_loss: 0.0014
Epoch 19/20
625/625 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 82s 125ms/step - loss: 0.0015 - val_loss: 0.0014
Epoch 20/20
625/625 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81s 124ms/step - loss: 0.0015 - val_loss: 0.0014
```

Both the training and validation losses show consistency, which implies that the LSTM autoencoder was able to learn a meaningful representation of the image data, and that the structure of the learnt representation was such that it did not overfit to the training data.

Why MSE was Chosen:

As the loss function, MSE was chosen to be more sensible to image reconstruction tasks considering that minimal change in the pixel values to the image can lead to noticeable degradation in image quality. Because MSE is minimized it will attempt to best reconstruct the image as close as possible to the original with a pixel to pixel basis.

Results and analysis of reconstruction:

After training the model was evaluated with the test set, using it to assess the model's reconstruction ability. Please see figure 6 below:

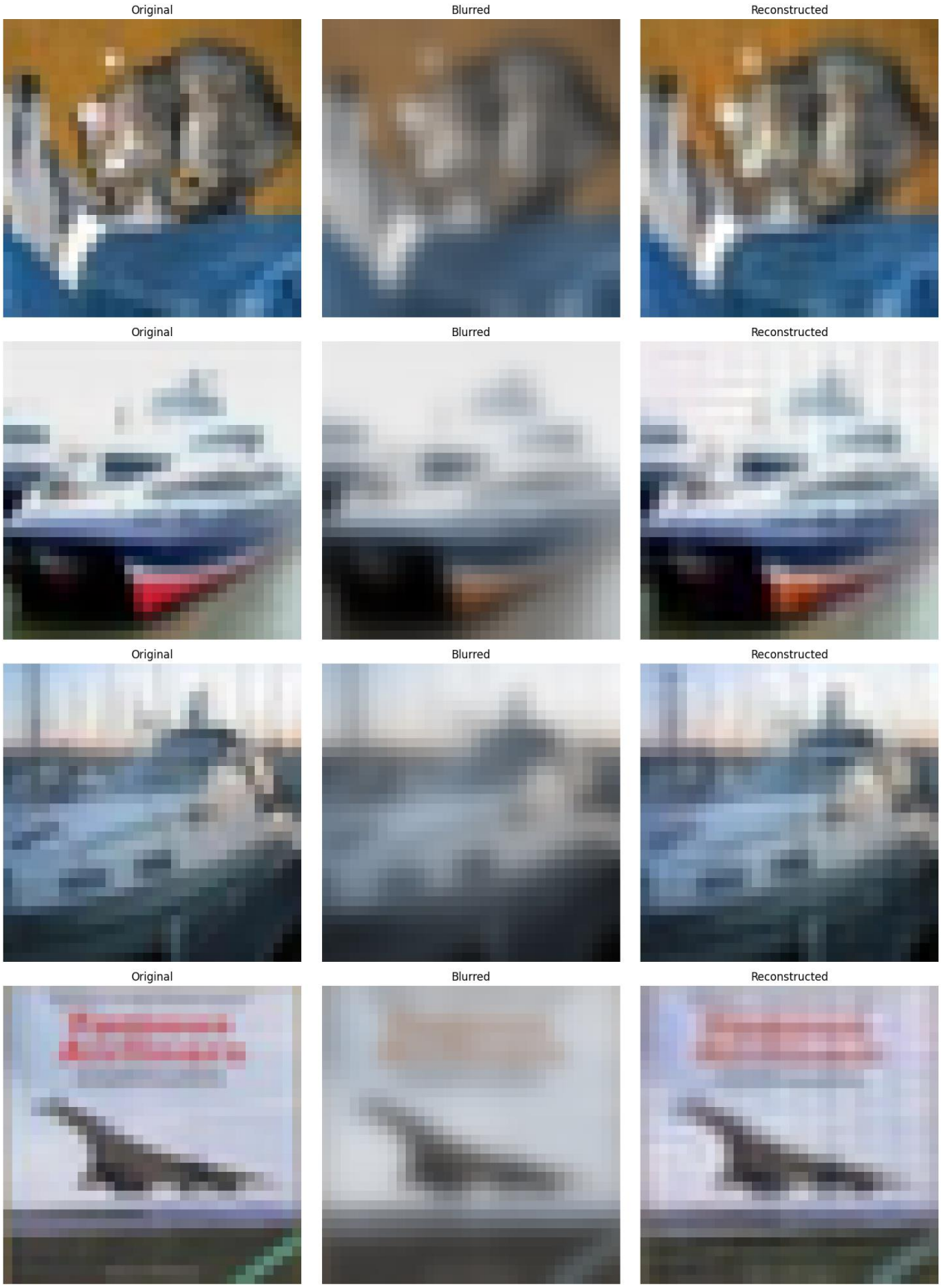

Figure 6: Side-by-side comparison of original, blurred, and reconstructed images

Figure 6 shows the results of the reconstruction: the original, the blurred and the reconstructed images are compared side by side. The qualitative results indicate that the LSTM autoencoder was

able to recover key features, such as edges and textures, which are often lost when images are blurred.

In particular, edges and textures were particularly well reconstructed, as these are important since they provide structure and form of objects contained in the image. However, some images — with more detailed textures and even more fine edges — still showed some residual blurring, suggesting that the autoencoder may have had a hard time handling the most complex patterns. The overall quality of the reconstructed images was improved significantly compared to the blurred inputs. Figure 6 compares original, blurred and reconstructed images clearly indicating that the autoencoder is capable of restoring most of the lost image detail.

Denoising the input images with the LSTM autoencoder was performed well. The model achieves visual restoration even in the presence of the inherent noise induced by Gaussian blur.

4. ANALYTICS USE CASES, RECOMMENDATIONS, AND CONCLUSION

The implementation of an LSTM-based autoencoder for image reconstruction in this study demonstrated the model's capability to denoise and recover key image features from blurred inputs. The LSTM performed well with regard to both spatial and temporal dependencies present in image data, which it used to reconstruct the image, especially in preserving structure, e.g. edges and textures.

Real world degradation was simulated using Gaussian blur and the model was able to recover most image details as evidenced by low Mean Squared Error (MSE) and high quality reconstructed images. The model successfully reconstructed key image features with a low mean squared error.

```
Epoch 20/20
625/625 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81s 124ms/step - loss: 0.0015 - val_loss: 0.0014
```

While the results were generally positive, some limitations were observed:

Residual Blurring: In some cases, we did not fully restore fine details, resulting in slight blurring of the reconstructed image. One potential reason that this occurs is if the LSTM model is not good at capturing highly complicated spatial patterns, or simply that the latent space is too small.

Room for Improvement: Methods in addition to using a deeper autoencoder, or in a sense attending to parts of the image map rather than absolutely recovering the whole image, should be developed to further help the model, if needed, concentrate on and reconstruct finer image details.

The LSTM autoencoder's strengths in dealing with these image reconstruction tasks are these, but also these are the points where it can still be improved. Futher exploring more advance architectures as variational autoencoders (VAEs) or generative adversarial networks (GANs) can show better images. Though we saw some limitations in reconstructing high detail textures, the performance as a whole was strong, considering the simplicity of the LSTM autoencoder architecture used in this work. This approach provides a compelling approach to image denoising, compression and reconstruction for applications in medical imaging, satellite imaging and real time video processing. This work serves as a starting point that can be extended in future work for more complex models, such as variational autoencoders or attention based mechanisms, to enhance reconstruction quality and robustness.

5. REFERENCES

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *University of Toronto*. Available at: http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf

Quanlin, W., Ye, H., Gu, Y., Zhang, H., Wang, L., & He, D. (2022). Denoising masked autoencoders are certifiable robust vision learners. *ArXiv*. Available at: https://arxiv.org/pdf/2210.06983

Sankaran, A., Vatsa, M., Singh, R., & Majumdar, A. (2017). Group sparse autoencoder. *Image and Vision Computing*, *60*, 64-74. Available at: https://doi.org/10.1016/j.imavis.2017.01.005

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, 1096-1103. Available at: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=d8004ccce5ab7dc89f968ca0226d6be5c8697bdf