Deep Learning

CS-898BD

Y723u998

Assignment 3

Supervised by: Dr. Lokesh Das

I. Introduction

This assignment aims to give us an insight into auto encoders and machine translation. This assignment has two parts. In the first part are tasked to train three different architectures: a stacked two-layer RNN, a stacked two-layer LSTM, and a stacked two-layer GRU. The second part involves creating our own Auto encoder which is capable of denoising input images. This assignment provides a valuable opportunity for us to apply the concepts that we have learned in class.

The code of this assignment is published in the repository created by our professor under the branch named yassine.

https://github.com/lokesh-c-das/CS898BDDEEPLEARNING/tree/yassine

II. Methodology

In order to fulfill the requirements of this assignment, I have used different methodologies for each part of the assignment. In this section, I will give details concerning the methodology that I have used in each part.

2.1. Machine translation

For the machine translation part, I used the code provided by our professor as a reference to become familiar with the dataset. The code includes data loading, tokenization, preprocessing, and training a simple RNN. Initially, I attempted to develop my model using PyTorch, but I encountered an error: “Expected input batch_size (864) to match target batch_size (1504).” Despite adjusting the input dimensions, the error persisted. To overcome this issue, I decided to switch to TensorFlow.

I implemented three architectures: a stacked two-layer RNN, a stacked two-layer LSTM, and a stacked two-layer GRU. I used the following hyperparameters:

- Loss: sparse_categorical_crossentropy
- Optimizer: Adam(learning_rate)

The models were trained using the model.fit function with a batch size of 16 and over 100 epochs.

The dataset that we have used to train a model has to txt files the first one is in English and the second one is the parallel to this dataset containing the translation in French. The dataset has 137861 sentences with their translation

2.2. Image denoising

Moving to the image denoising phase, we used an autoencoder architecture, consisting of an encoder and a decoder. For the dataset, we utilized the Denoising Dirty Documents dataset, which contains

three main folders: the first folder contained the training data, the second contained the cleaned training data, and the third had the uncleaned test data. Since the training and cleaned training data were not directly mapped to each other, I loaded both folders using PyTorch in a way that ensured they were properly aligned.
Additionally, I needed a validation set, so I set aside 15% of the training data for validation. The professor required us to calculate the SSIM score on the testing dataset, but this was not possible because the test set did not contain ground truth labels, which are needed for calculating SSIM. Therefore, the SSIM score was calculated on the validation set that I extracted from the training data. The model was trained on 600 epochs.

III. Deep Learning Architecture

In this section I will go in details with the deep learning architectures that I have used to fulfill the requirements of this assignment.

3.1. Machine translation

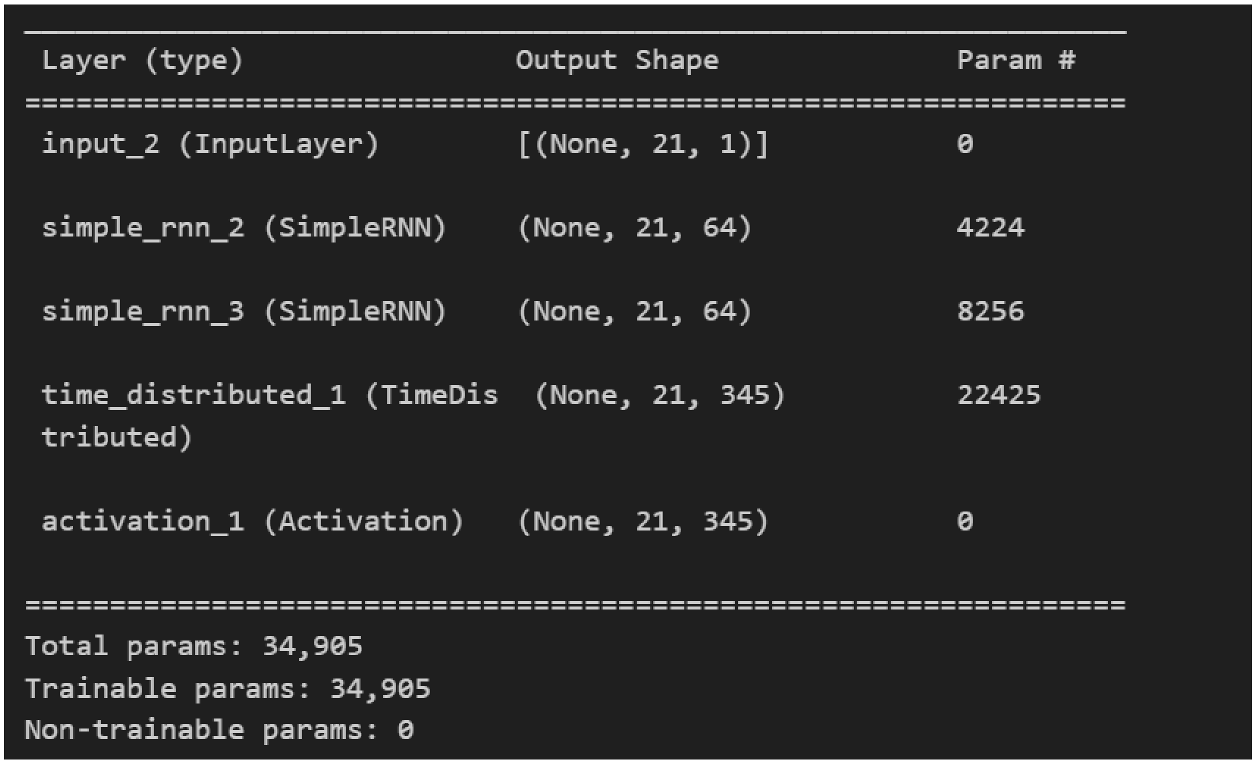
For the machine translation part, I have adopted three different architectures. In this section, I have explored a stacked two-layer RNN model, stacked two layer LSTM, and a stacked two layer GRU. The reason behind using these three architectures is to be able to compare the performance of each in our dataset. In this section I will give details about the architecture of each model.

3.1.1. Stacked RNN Model

The Stacked RNN model has the following building blocks:

- Input Shape: The input has a shape of (None, 21, 1), meaning it accepts sequences of 21 time steps, each with a single feature.
- Layer 1 (SimpleRNN): The first SimpleRNN layer has 64 units and processes the input sequence. It outputs a sequence of shape (None, 21, 64).
- Parameters: 4224 parameters (weights and biases) for this layer.
- Layer 2 (SimpleRNN): Another SimpleRNN layer with 64 units follows, processing the previous RNN layer's output.
- Parameters: 8256 parameters in this layer.
- Layer 3 (TimeDistributed): A fully connected layer applied to each time step. It outputs a sequence with 345 features for each time step.
- Parameters: 22,425 parameters.
- Activation Layer: An activation function is applied to the output from the fully connected layer, keeping the shape (None, 21, 345).

- Total Trainable Parameters: 34,905 parameters.



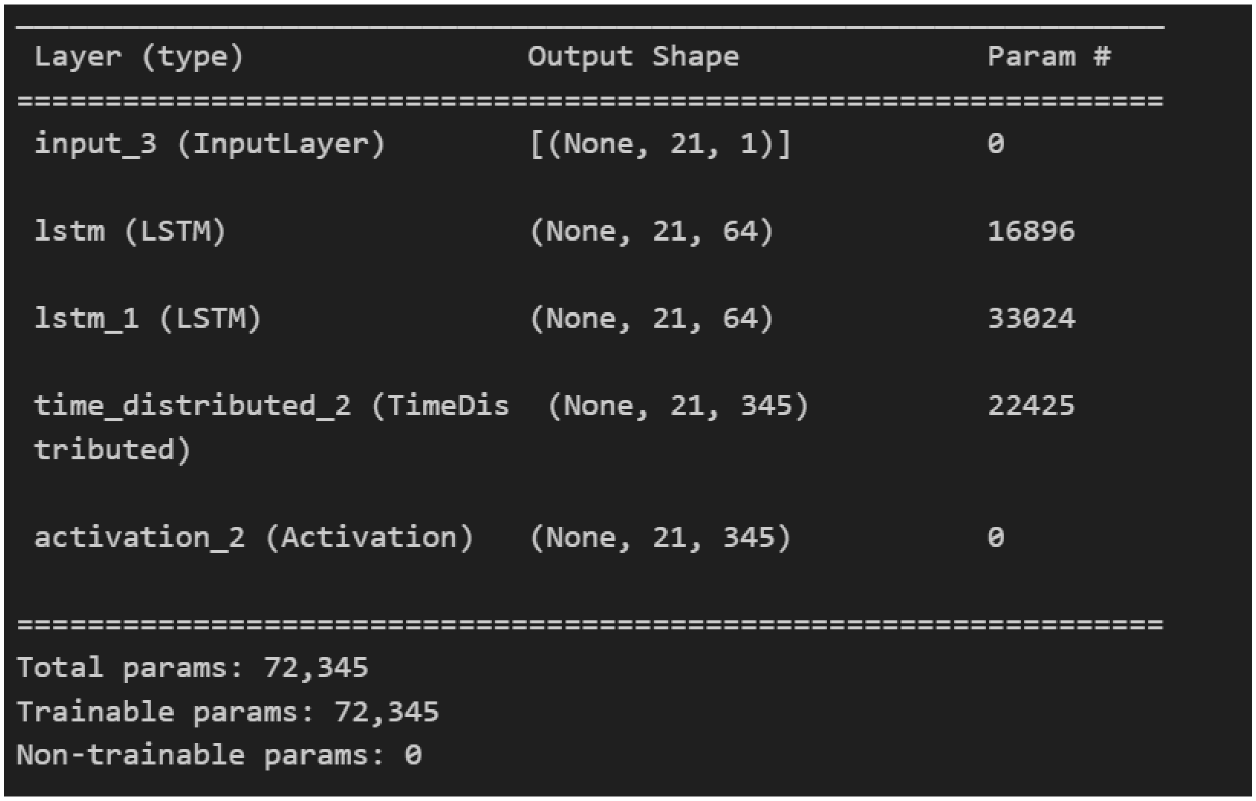
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 21, 1)]           0

 simple_rnn_2 (SimpleRNN)    (None, 21, 64)            4224

 simple_rnn_3 (SimpleRNN)    (None, 21, 64)            8256

 time_distributed_1 (TimeDis  (None, 21, 345)          22425
 tributed)

 activation_1 (Activation)   (None, 21, 345)           0

=================================================================
Total params: 34,905
Trainable params: 34,905
Non-trainable params: 0
```

Figure 1. Stacked RNN Model Architecture

3.1.2. Stacked LSTM model

The stacked LSTM model has the following building blocks:

- Input Shape: input shape of (None, 21, 1).
- Layer 1 (LSTM): The first LSTM layer has 64 units, outputs a sequence of shape (None, 21, 64).
- Parameters: 16,896 parameters (LSTM cells involve more parameters compared to SimpleRNN).
- Layer 2 (LSTM): Another LSTM layer with 64 units, outputs the processed sequence.
- Parameters: 33,024 parameters.
- Layer 3 (TimeDistributed): Similar to the SimpleRNN architecture, it outputs 345 features per time step.
- Parameters: 22,425 parameters.
- Activation Layer: Applies activation, maintaining the shape (None, 21, 345).
- Total Trainable Parameters: 72,345 parameters.



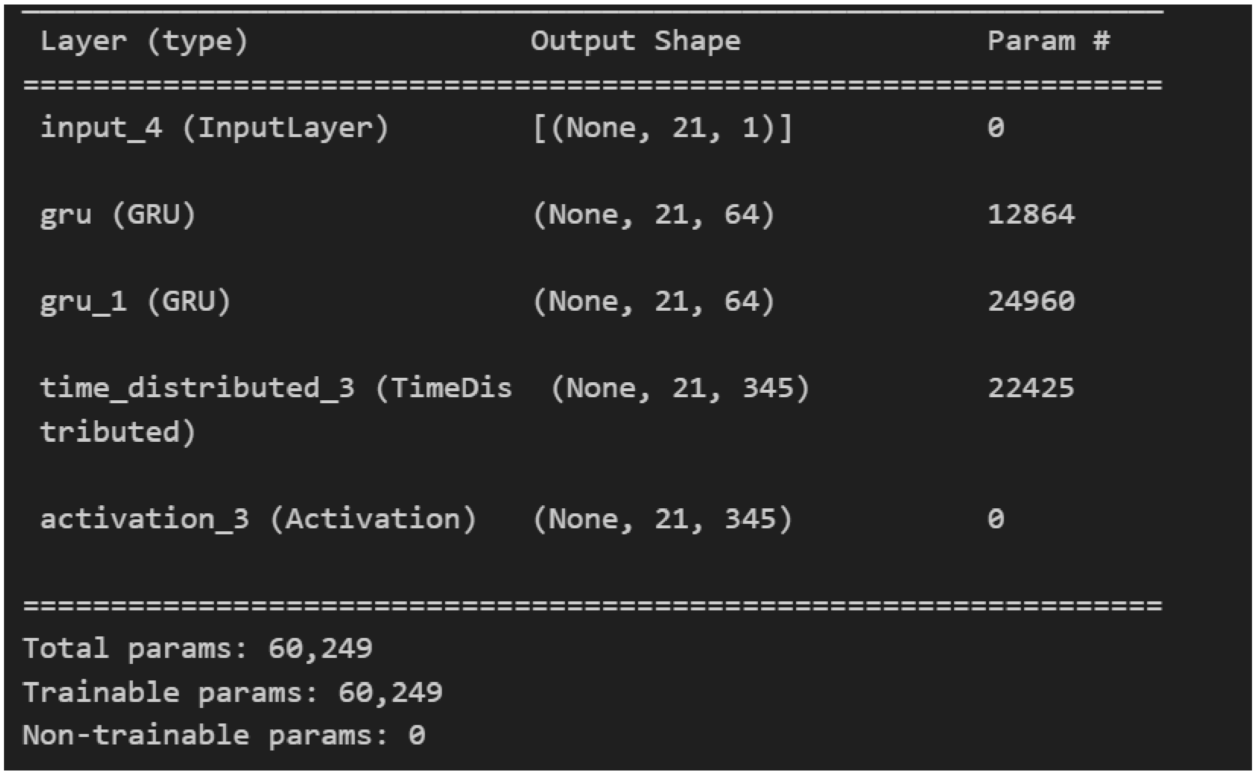
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 21, 1)]           0

 lstm (LSTM)                 (None, 21, 64)            16896

 lstm_1 (LSTM)               (None, 21, 64)            33024

 time_distributed_2 (TimeDis  (None, 21, 345)          22425
 tributed)

 activation_2 (Activation)   (None, 21, 345)           0

=================================================================
Total params: 72,345
Trainable params: 72,345
Non-trainable params: 0
```

Figure 2. Stacked LSTM Model Architecture

3.1.3. Stacked GRU Model

The stacked GRU model has the following Building blocks:

- Input Shape: (None, 21, 1).
- Layer 1 (GRU): The first GRU layer has 64 units and processes the input sequence.
- Parameters: 12,864 parameters.
- Layer 2 (GRU): A second GRU layer with 64 units processes the output of the first.
- Parameters: 24,960 parameters.
- Layer 3 (TimeDistributed): Outputs 345 features per time step like the other models.
- Parameters: 22,425 parameters.
- Activation Layer: Applies an activation function with no additional parameters.
- Total Trainable Parameters: 60,249 parameters.



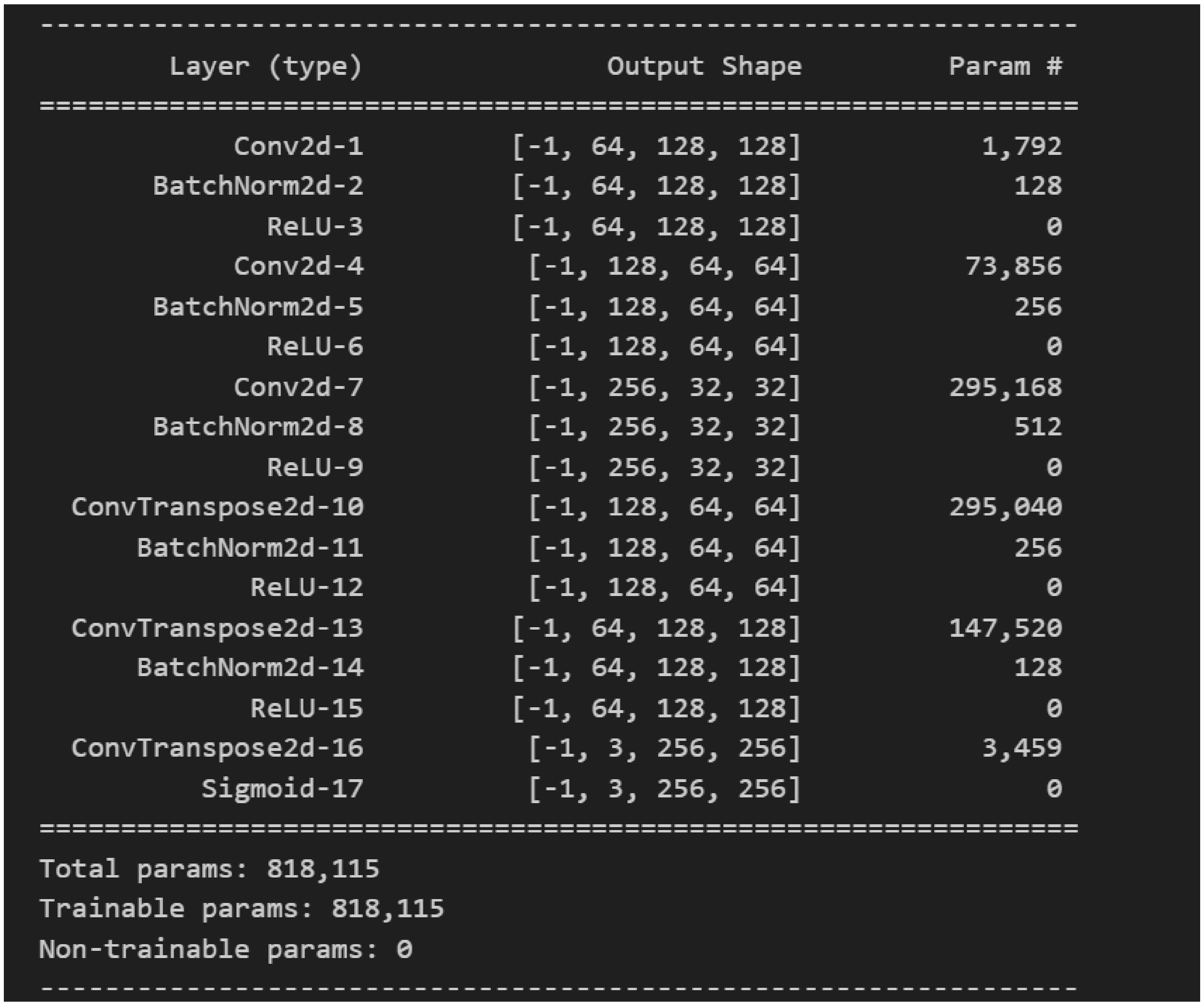
```
________________________________________________________________
 Layer (type)                Output Shape              Param #
================================================================
 input_4 (InputLayer)        [(None, 21, 1)]           0

 gru (GRU)                   (None, 21, 64)            12864

 gru_1 (GRU)                 (None, 21, 64)            24960

 time_distributed_3 (TimeDis  (None, 21, 345)          22425
 tributed)

 activation_3 (Activation)   (None, 21, 345)           0

================================================================
Total params: 60,249
Trainable params: 60,249
Non-trainable params: 0
```

Figure 3. Stacked GRU Model Architecture

3.2. Image Denoising

The image denoising model has the following building blocks:

- Conv2d Layers: These are standard 2D convolutional layers (Conv2d) that apply convolution operations to the input data. Each Conv2d layer is followed by:
- BatchNorm2d: Batch normalization to standardize the inputs to the following layer.
- ReLU Activation: Rectified Linear Unit (ReLU) activation functions are applied after each batch normalization layer.
- ConvTranspose2d Layers: These layers are transposed convolution layers, often used in upsampling operations for tasks such as image generation or segmentation. They help to increase the spatial dimensions (width and height) of the input.
- Sigmoid Layer: The final layer is a Sigmoid activation function, commonly used for binary classification tasks or when generating output in a constrained range (such as between 0 and 1).



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [-1, 64, 128, 128]           1,792
       BatchNorm2d-2        [-1, 64, 128, 128]             128
              ReLU-3        [-1, 64, 128, 128]               0
            Conv2d-4         [-1, 128, 64, 64]          73,856
       BatchNorm2d-5         [-1, 128, 64, 64]             256
              ReLU-6         [-1, 128, 64, 64]               0
            Conv2d-7         [-1, 256, 32, 32]         295,168
       BatchNorm2d-8         [-1, 256, 32, 32]             512
              ReLU-9         [-1, 256, 32, 32]               0
  ConvTranspose2d-10         [-1, 128, 64, 64]         295,040
      BatchNorm2d-11         [-1, 128, 64, 64]             256
             ReLU-12         [-1, 128, 64, 64]               0
  ConvTranspose2d-13        [-1, 64, 128, 128]         147,520
      BatchNorm2d-14        [-1, 64, 128, 128]             128
             ReLU-15        [-1, 64, 128, 128]               0
  ConvTranspose2d-16         [-1, 3, 256, 256]           3,459
          Sigmoid-17         [-1, 3, 256, 256]               0
================================================================
Total params: 818,115
Trainable params: 818,115
Non-trainable params: 0
----------------------------------------------------------------
```

Figure 4. Image Denoising Model Architecture

IV. Experiments and Results

4.1. Machine translation

In this subsection I will detail the Experiments conducted along with their results to build a model capable of translating from English to French.

Starting with the training and validation losses

4.1.1. Validation And training Loss

The figures below demonstrate the evolution of both training loss and validation loss over epochs for three models: Stacked RNN, Stacked LSTM, and Stacked GRU.

From the graphs, we observe that the validation loss decreases steadily as the models continue to train, indicating improvements in generalization performance. Among the models, the Stacked LSTM Model shows the best performance, achieving the lowest validation loss. This suggests that the LSTM effectively captures the long-range dependencies in the data.

In contrast, the Stacked RNN Model exhibits the highest validation loss, performing the worst among the three. This may be attributed to its limitations in handling long-term dependencies, making it less efficient in this task. The Stacked GRU Model performs better than the Stacked RNN but does not reach the performance level of the LSTM.

Overall, the Stacked LSTM stands out as the most effective model, while the Stacked RNN shows the least improvement

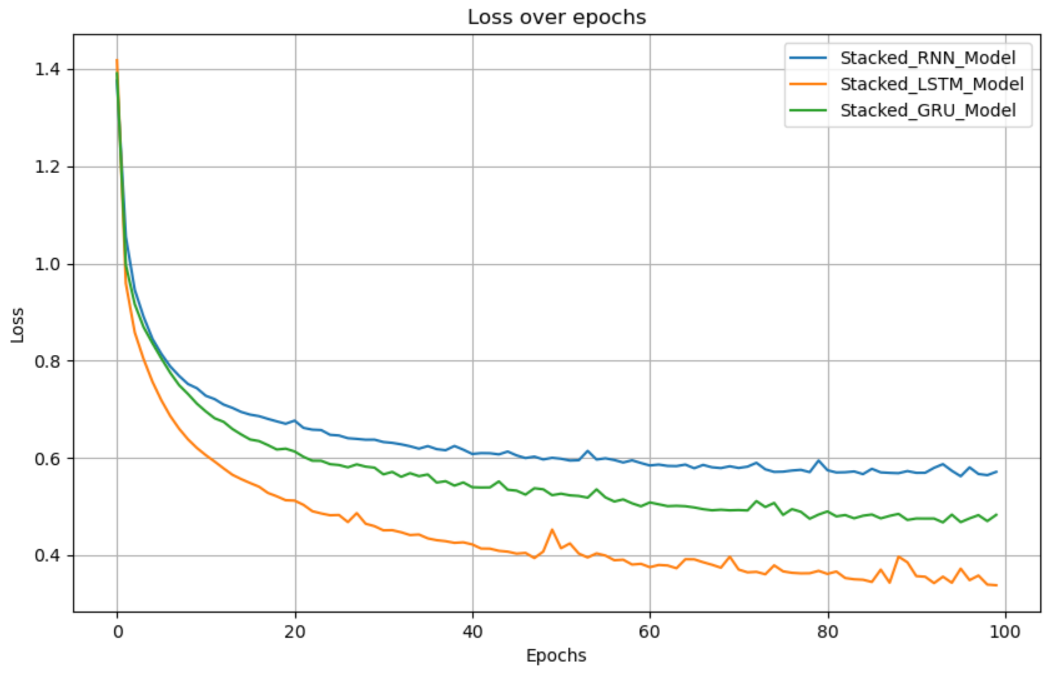


Figure 5. training loss

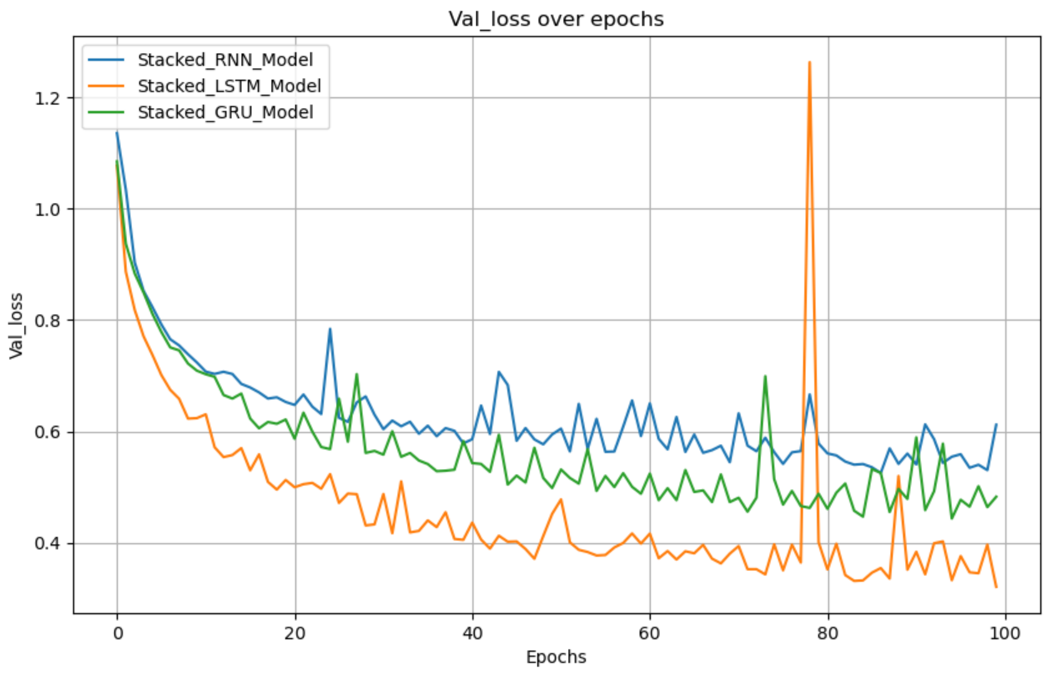


Figure 6. validation Loss

4.1.2. Validation and training accuracy

The figures below demonstrate the evolution of both training accuracy and validation accuracy over epochs for three models: Stacked RNN, Stacked LSTM, and Stacked GRU.

We observe that the validation accuracy improves progressively during training for all three models. The Stacked LSTM Model demonstrates the highest validation accuracy, consistently outperforming both the GRU and RNN models. This further solidifies LSTM's strength in capturing complex temporal dependencies, leading to better generalization.

The Stacked GRU Model shows better accuracy compared to the Stacked RNN Model, which lags behind both models throughout the epochs. While the RNN struggles with long-range dependencies, leading to less stable accuracy improvements, GRU balances performance and efficiency more effectively but does not match LSTM's performance.

The Stacked LSTM Model remains the best performer, while the Stacked RNN Model consistently performs worse in terms of validation accuracy.

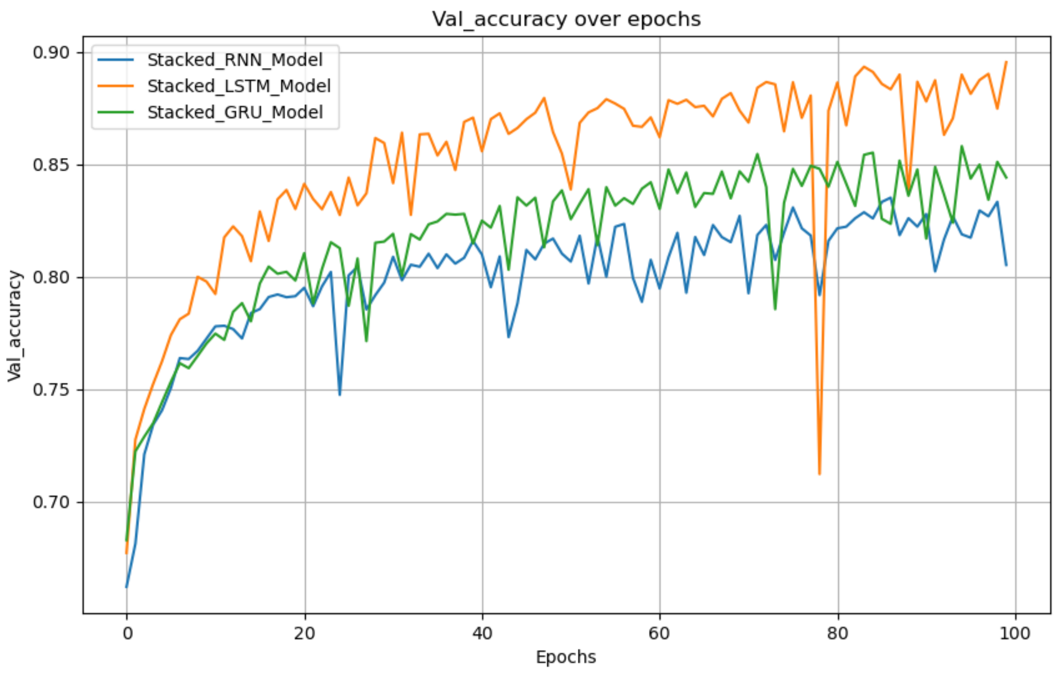


Figure 7. Validation accuracy

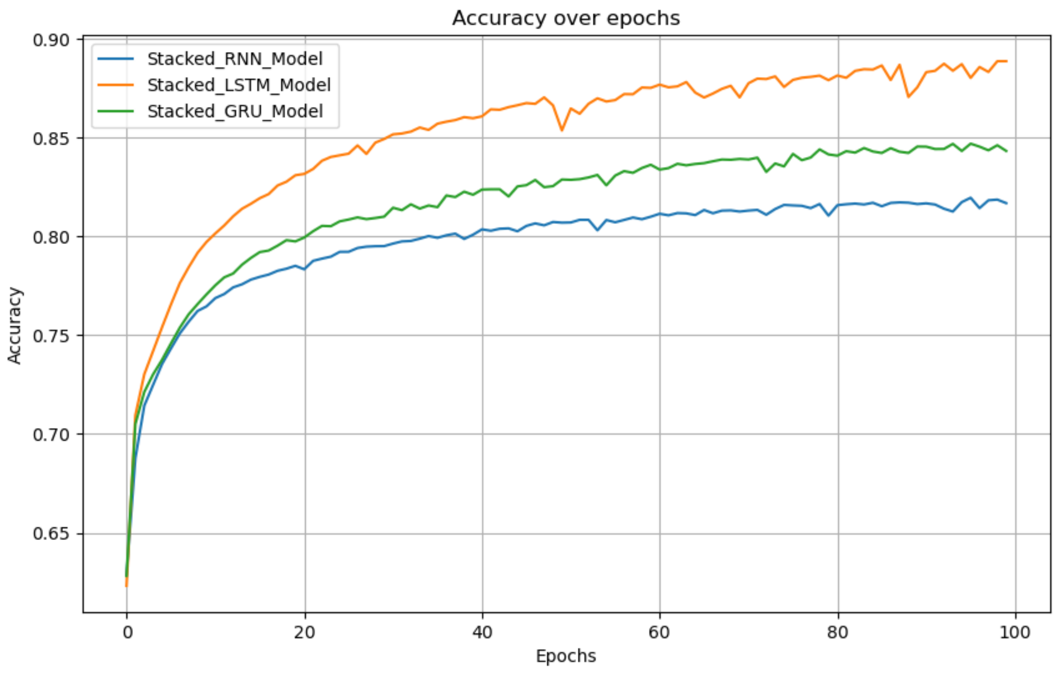


Figure 8. Training Accuracy

4.1.3. Sample outputs in each architecture

In this example, the model used is a GRU (Gated Recurrent Unit), which is a type of recurrent neural network (RNN). Here's a breakdown of the output provided:

- **GRU**
  - Original: new jersey is sometimes quiet during autumn , and it is snowy in april .
  - Predicted: new jersey est parfois calme en mois et et il est neigeux en juillet <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: the united states is usually chilly during july , and it is usually freezing in november .
  - Predicted: les états unis est généralement froid en juillet et il est généralement agréable en <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: california is usually quiet during march , and it is usually hot in june .
  - Predicted: californie est généralement calme en mars et il est généralement chaud en juin <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: the united states is sometimes mild during june , and it is cold in september .
  - Predicted: les états unis est parfois doux en juin et il est froid en septembre <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: your least liked fruit is the grape , but my least liked is the apple .
  - Predicted: votre fruit aimé moins aimé la pomme mais moins aimé est la pomme <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: his favorite fruit is the orange , but my favorite is the grape .
  - Predicted: son fruit préféré est la citron mais préféré préféré est la pomme <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: paris is relaxing during december , but it is usually chilly in july .
  - Predicted: paris est calme en décembre mais il est généralement froid en juillet <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: new jersey is busy during spring , and it is never hot in march .
  - Predicted: new jersey est sec en printemps et il est jamais chaude en mars <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: our least liked fruit is the lemon , but my least liked is the grape .
  - Predicted: notre fruit aimé des est la citron mais mon moins aimé est la pomme <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: the united states is sometimes busy during january , and it is sometimes warm in november .
  - Predicted: les états unis est parfois occupé en janvier et il est parfois chaud en novembre <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
- **LSTM**
  - Original: new jersey is sometimes quiet during autumn , and it is snowy in april .
  - Predicted: new jersey est parfois calme en mois automne et il est neigeux en avril <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: the united states is usually chilly during july , and it is usually freezing in november .
  - Predicted: les états unis est généralement froid en juillet et il est généralement en avril <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: california is usually quiet during march , and it is usually hot in june .
  - Predicted: californie est généralement calme en mars et il est généralement chaud en juin <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: the united states is sometimes mild during june , and it is cold in september .
  - Predicted: les états unis est parfois doux en juin et il est froid en septembre <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

  - Original: your least liked fruit is the grape , but my least liked is the apple .
  - Predicted: votre fruit aimé des fruits la raisin raisin mais moins aimé est est pomme pomme <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

    - Original: his favorite fruit is the orange , but my favorite is the grape .
    - Predicted: son fruit préféré est la mais son favori est la pêche <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

    - Original: paris is relaxing during december , but it is usually chilly in july .
    - Predicted: paris est relaxant en décembre mais il est généralement froid en juillet <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

    - Original: new jersey is busy during spring , and it is never hot in march .
    - Predicted: new jersey est occupé en printemps et il est jamais chaud en mars <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

    - Original: our least liked fruit is the lemon , but my least liked is the grape .
    - Predicted: notre fruit aimé des est la citron mais mon moins aimé est la raisin <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

    - Original: the united states is sometimes busy during january , and it is sometimes warm in november .
    - Predicted: les états unis est parfois occupée en janvier et il est parfois chaud en novembre <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
- **RNN**
    - Original: new jersey is sometimes quiet during autumn , and it is snowy in april .
    - Predicted: new jersey est parfois chaud en l' et il est est en en avril <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

    - Original: the united states is usually chilly during july , and it is usually freezing in november .
    - Predicted: les états unis est généralement froid en juillet et il est généralement en en <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

    - Original: california is usually quiet during march , and it is usually hot in june .

- Predicted: californie est généralement calme habituellement l' et il est généralement habituellement est en juin <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

- Original: the united states is sometimes mild during june , and it is cold in september .
- Predicted: les états unis est parfois doux en juin et il est froid en septembre <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

- Original: your least liked fruit is the grape , but my least liked is the apple .
- Predicted: votre fruit aimé aimé est la raisin mais mon moins aimé est la <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

- Original: his favorite fruit is the orange , but my favorite is the grape .
- Predicted: son fruit préféré est la mais mais moins préféré est la <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

- Original: paris is relaxing during december , but it is usually chilly in july .
- Predicted: paris est chaud au décembre mais il est généralement froid en juillet <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

- Original: new jersey is busy during spring , and it is never hot in march .
- Predicted: new jersey est occupé en printemps et il est jamais tranquille en mars <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

- Original: our least liked fruit is the lemon , but my least liked is the grape .
- Predicted: notre fruit moins aimé est la citron mais mon moins aimé est la raisin <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

- Original: the united states is sometimes busy during january , and it is sometimes warm in november .
- Predicted: les états unis est parfois occupé en janvier et il est parfois chaud en novembre <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>

4.2. Image Denoising

The graph below shows that the training loss starts at a high value around 0.22 and drops rapidly during the initial epochs. After this sharp decline, the loss stabilizes and fluctuates around 0.06. This indicates that the model has learned effectively in the early stages and maintains a steady performance during later epochs, suggesting minimal overfitting or underfitting.

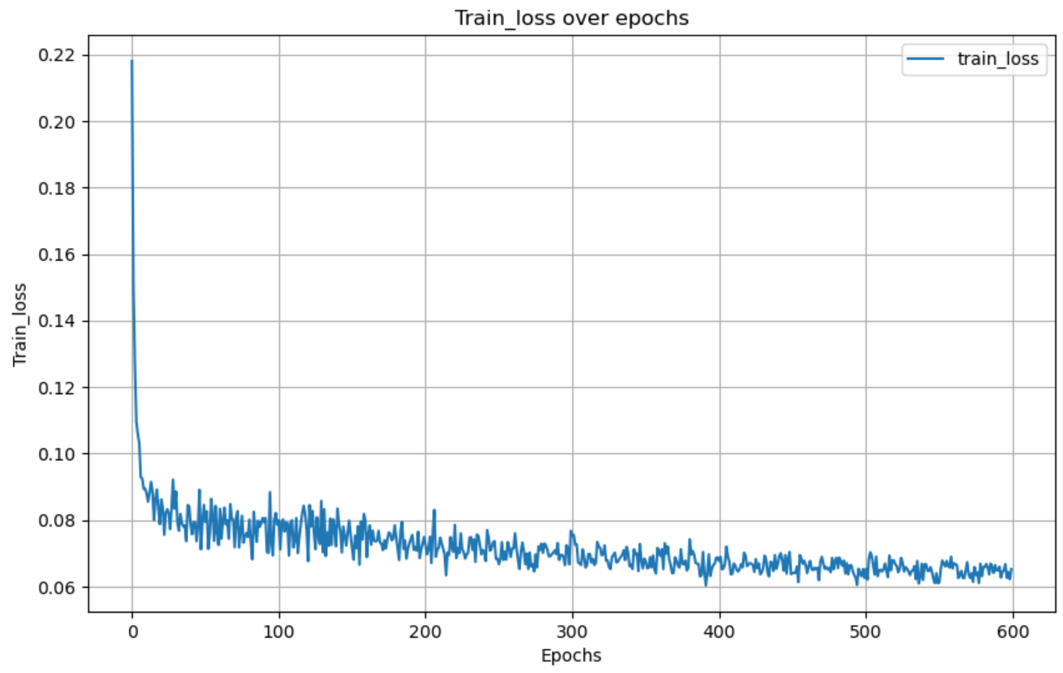


Figure 9. Training loss over epochs

The graph below depicts the validation loss, which follows a similar trend to the training loss. It starts high, decreases rapidly, and stabilizes around 0.06. The close correlation between the training and validation loss curves indicates that the model generalizes well to unseen data, without significant overfitting.

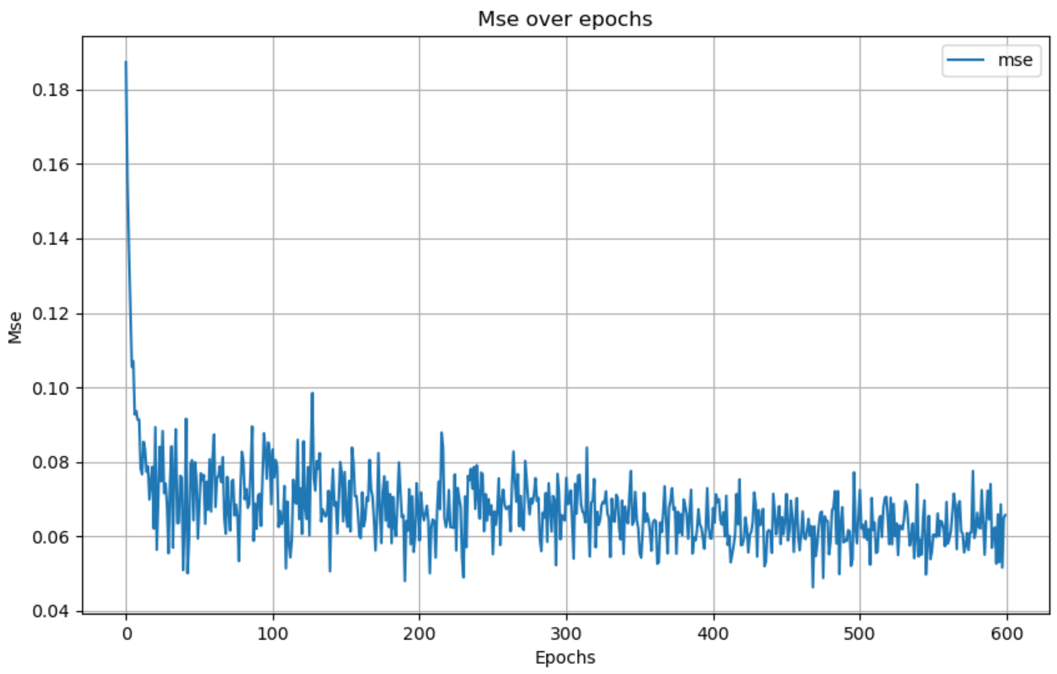


Figure 10. MSE Over epochs

The graph below highlights the SSIM score, which measures the perceptual quality of the denoised images. The SSIM starts low (around 0.1) and rapidly increases, stabilizing above 0.6 and frequently reaching values close to 0.7. This consistent improvement in SSIM shows that the model effectively enhances image quality, producing denoised images with better structural similarity to the original, clean images.

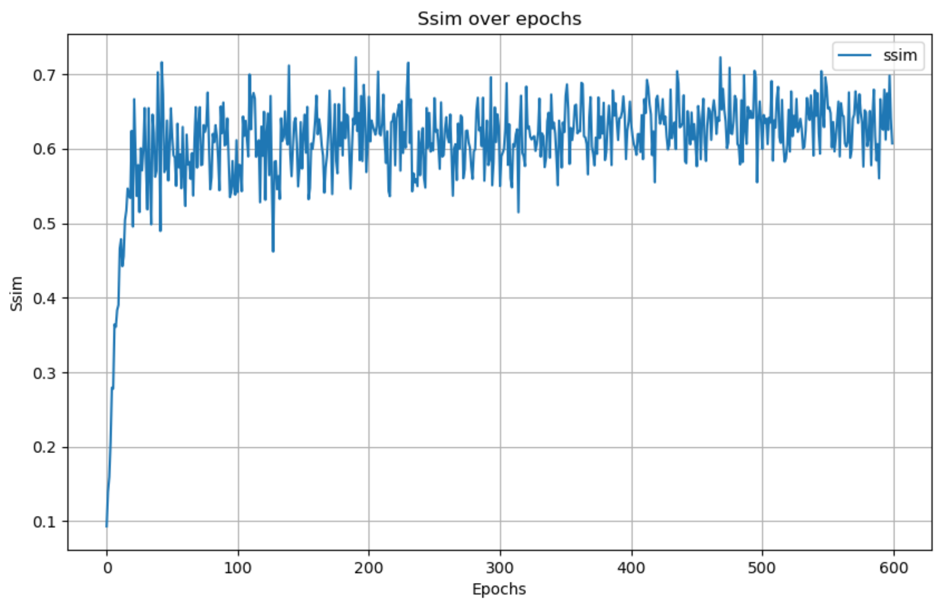
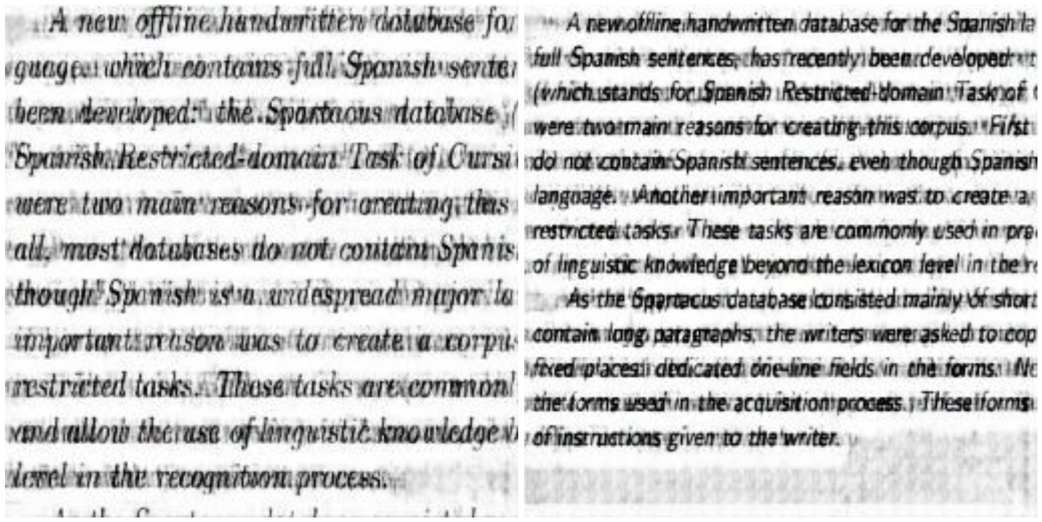


Figure 11. SSIM Score over epochs

After detailing the metrics achieved by the image denoising model, I am now sharing some sample outputs from the test set after running inference. We can see that the model successfully removed the background from the images.



A new offline handwritten database fo
guage which contains full Spanish sente
been developed: the Spartacus database (
Spanish Restricted-domain Task of Cursi
were two main reasons for creating this
all, most databases do not contain Spanis
though Spanish is a widespread major la
important reason was to create a corpu
restricted tasks. These tasks are common
and allow the use of linguistic knowledge
level in the recognition process.

A new offline handwritten database for the Spanish la
full Spanish sentences, has recently been developed
(which stands for Spanish Restricted-domain Task of
were two main reasons for creating this corpus. First
do not contain Spanish sentences, even though Spanish
language. Another important reason was to create a
restricted tasks. These tasks are commonly used in pr
of linguistic knowledge beyond the lexicon level in the r
As the Spartacus database consisted mainly of short
contain long paragraphs, the writers were asked to cop
fixed places: dedicated one-line fields in the forms. N
the forms used in the acquisition process. These forms
of instructions given to the writer.

Figure 12. Model Architecture

V. Conclusion

This assignment has provided a comprehensive understanding of two key areas: machine translation and image denoising, by exploring different architectures and applying them in practical scenarios. For the machine translation task, the comparative analysis of stacked RNN, LSTM, and GRU models highlighted the superior performance of LSTM, demonstrating its capacity to handle long-range dependencies, which is crucial for language tasks. The GRU model followed closely, balancing performance and computational efficiency, while the RNN model lagged due to its limitations with long-term dependencies.

The code of this assignment is published in the repository created by our professor under the branch named yassine.

https://github.com/lokesh-c-das/CS898BDDEEPLEARNING/tree/yassine

VI. References

[Denoising Dirty Documents | Kaggle]