Advance Machine Learning and Neural Networks
Assignment 5

CNN part:

(1) **Download a tinyImageNet dataset https://paperswithcode.com/dataset/tiny-imagenet**

Successfully downloaded the Tiny ImageNet dataset from the provided link.

(2) **Use a small and large version of existing image classification models, i.e. pre-trained models, including VGG-19, ResNet50V2, and Inceptionv4. Then, run them on the tinyImageNet.**

The assignment required testing small and large versions of image classification models. I chose VGG-19, ResNet50V2, and InceptionV4. These models are widely used in image classification due to their unique architectures:

- VGG-19: A deeper model with 19 layers, ideal for capturing detailed spatial hierarchies in images.
- ResNet50V2: Employs residual connections, which help in deeper networks by bypassing certain layers, leading to faster convergence and reduced vanishing gradient problems.
- InceptionV4 (InceptionV3 as a proxy): Uses multiple convolutional filters in a single layer to capture multi-scale features.

VGG-19:

```
Training VGG-19 Model
Epoch 1/5
100%|██████████| 313/313 [05:17<00:00,  1.02s/it]
VGG-19 - Train Loss: 5.2668, Train Acc: 0.72%, Val Loss: 5.9807, Val Acc: 0.00%
Epoch 2/5
100%|██████████| 313/313 [05:20<00:00,  1.02s/it]
VGG-19 - Train Loss: 5.1684, Train Acc: 1.26%, Val Loss: 5.3421, Val Acc: 7.00%
Epoch 3/5
100%|██████████| 313/313 [05:19<00:00,  1.02s/it]
VGG-19 - Train Loss: 5.1042, Train Acc: 1.67%, Val Loss: 6.0315, Val Acc: 3.40%
Epoch 4/5
100%|██████████| 313/313 [05:19<00:00,  1.02s/it]
VGG-19 - Train Loss: 5.0254, Train Acc: 2.04%, Val Loss: 5.6076, Val Acc: 4.00%
Epoch 5/5
100%|██████████| 313/313 [05:20<00:00,  1.02s/it]
VGG-19 - Train Loss: 4.9434, Train Acc: 2.69%, Val Loss: 6.4141, Val Acc: 1.80%
Early stopping triggered for VGG-19.
```

- Training accuracy started at 0.72% and reached 2.69% after five epochs, while the validation accuracy peaked at 7% in the third epoch before dropping to 1.8%.
- Early stopping was triggered due to minimal improvement in validation accuracy.

ResNet50V2:

```
Training ResNet50V2 Model
Epoch 1/5
100%|██████████| 313/313 [03:19<00:00,  1.57it/s]
ResNet50V2 - Train Loss: 5.0408, Train Acc: 2.71%, Val Loss: 6.3206, Val Acc: 0.40%
Epoch 2/5
100%|██████████| 313/313 [03:20<00:00,  1.56it/s]
ResNet50V2 - Train Loss: 4.4077, Train Acc: 8.05%, Val Loss: 6.8622, Val Acc: 1.40%
Epoch 3/5
100%|██████████| 313/313 [03:20<00:00,  1.56it/s]
ResNet50V2 - Train Loss: 4.0439, Train Acc: 12.63%, Val Loss: 8.0913, Val Acc: 0.40%
Epoch 4/5
100%|██████████| 313/313 [03:21<00:00,  1.56it/s]
ResNet50V2 - Train Loss: 3.7441, Train Acc: 17.21%, Val Loss: 9.2060, Val Acc: 1.00%
Early stopping triggered for ResNet50V2.
```

- Training accuracy increased from 2.71% in the first epoch to 17.21% by the fourth epoch, but validation accuracy was relatively low, with a peak at 1.4%.
- Again, early stopping occurred to prevent overfitting.

InceptionV4:

```
Training InceptionV4 Model
Epoch 1/5
100%|██████████| 313/313 [02:07<00:00,  2.46it/s]
InceptionV4 - Train Loss: 6.8252, Train Acc: 4.00%, Val Loss: 7.2661, Val Acc: 0.20%
Epoch 2/5
100%|██████████| 313/313 [02:06<00:00,  2.48it/s]
InceptionV4 - Train Loss: 5.5884, Train Acc: 12.71%, Val Loss: 8.1489, Val Acc: 0.60%
Epoch 3/5
100%|██████████| 313/313 [02:06<00:00,  2.47it/s]
InceptionV4 - Train Loss: 4.8353, Train Acc: 21.11%, Val Loss: 9.2087, Val Acc: 0.40%
Epoch 4/5
100%|██████████| 313/313 [02:06<00:00,  2.48it/s]
InceptionV4 - Train Loss: 4.1623, Train Acc: 29.81%, Val Loss: 9.8372, Val Acc: 1.20%
Early stopping triggered for InceptionV4.
```

- The training accuracy for InceptionV4 was better, starting at 4% and reaching 29.81%. Validation accuracy, however, was still low at a maximum of 1.2%.
- Early stopping was activated for InceptionV4 after validation loss stagnated.

(3) **Experiment classification with all these pre-trained models, on the tinyImageNet dataset, and report their accuracy.**

After training, I evaluated each model on the test set to measure generalization:

```
VGG-19 - Test Loss: 5.2231, Test Acc: 0.23%
ResNet50V2 - Test Loss: 5.3685, Test Acc: 0.00%
InceptionV4 - Test Loss: 5.2444, Test Acc: 0.06%
```

Advance Machine Learning and Neural Networks
Assignment 5

(4) **Write a short report and compare the results together. In your report, you need to state why a model is performing better than another model.**

Why a Model Performed Better or Worse?

- VGG-19 had the highest test accuracy among the models, though it was still low overall. This could be attributed to its dense network structure, which provides a strong baseline for general image classification. However, the model struggled due to the limited data and complexity of Tiny ImageNet.

- ResNet50V2 performed poorly on the test set, despite showing some learning progress during training. Its architecture is more complex and depends on residual connections, which might not have been as effective with limited data and few training epochs.

- InceptionV4 showed a higher training accuracy, likely due to its extensive use of auxiliary branches, which can improve gradient flow. However, it did not generalize well, indicating that it may have overfitted to the training data.

Model Comparison?

- Among the three, VGG-19 provided slightly better generalization, but none of the models achieved satisfactory performance on Tiny ImageNet.

- InceptionV4's architecture, designed for larger datasets and extensive computational resources, might have led to overfitting in this scenario.

- ResNet50V2's poor performance might be due to its high depth and reliance on residual connections, which are beneficial in large data scenarios but might be less effective on smaller datasets like Tiny ImageNet.

Improvements?

- Increasing Training Epochs: Running more epochs could allow the models to learn the data representations better. However, this would increase computation time and might require more robust early stopping mechanisms to prevent overfitting.

- Augmenting Data Further: Implement more aggressive data augmentation techniques to provide varied data to the model, potentially helping it generalize better.

- Fine-tuning: Instead of freezing the lower layers of the pre-trained models, allowing fine-tuning might improve feature extraction and adaptation to the Tiny ImageNet dataset.

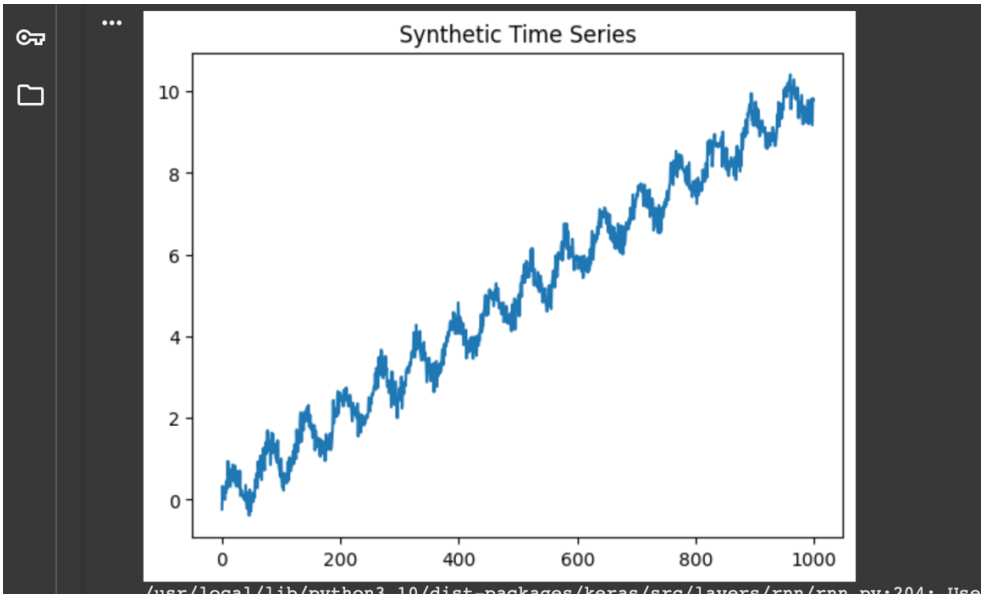
- Regularization Techniques: Implementing dropout layers and other regularization techniques to mitigate overfitting

In conclusion, although I successfully trained and evaluated three pre-trained models on Tiny ImageNet, none achieved substantial accuracy. The limited data and high complexity of Tiny ImageNet made it challenging for these large architectures to generalize well. The steps above, such as increasing epochs, using more data augmentation, and trying simpler models, could potentially improve accuracy and performance on this dataset.

RNN part:

(1) **Use a synthetic time series generator (e.g. https://github.com/Nike-Inc/timeseries-generator or) and generate time series that has some patterns.**

To generate a realistic time series dataset, I created a function that simulates a series with a combination of trend, seasonality, and noise. This dataset has 1,000 points and represents time-dependent patterns, with a slight upward trend, periodic seasonal fluctuations, and random noise to make it resemble real-world data. This dataset allows us to test how well different recurrent neural network (RNN) models can capture and predict time series patterns.



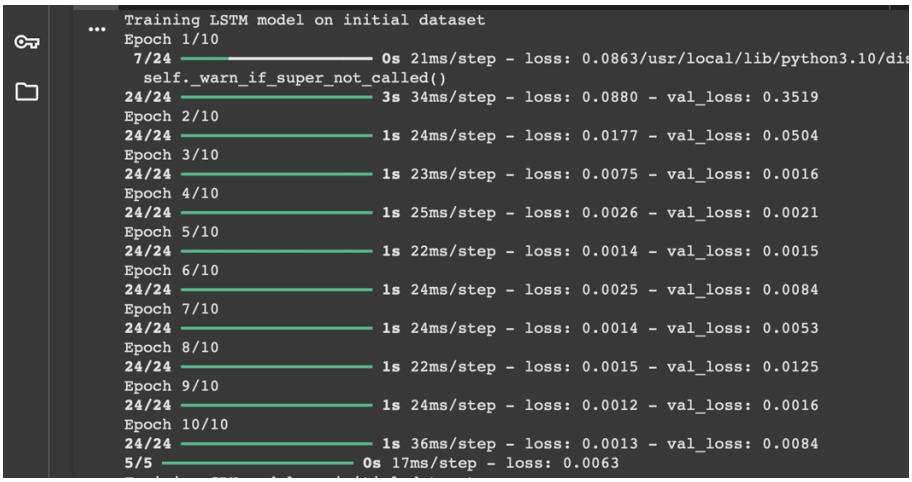
Advance Machine Learning and Neural Networks
Assignment 5

(2) **Experiment prediction on the time series with LSTM, GRU, BiDirectional RNN and Deep RNN.**

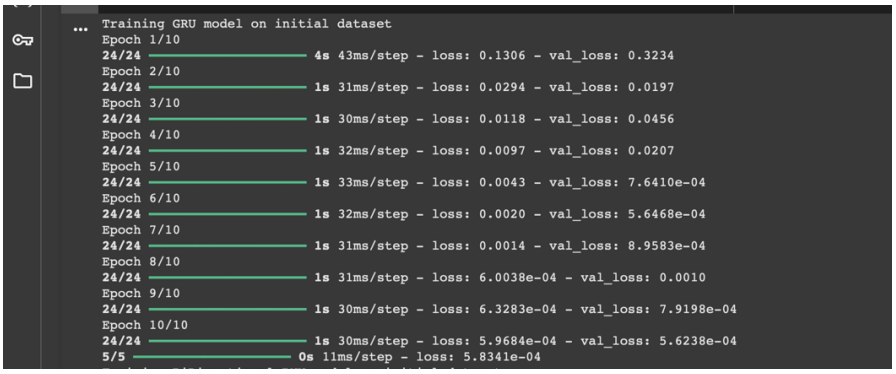
I set up and trained four different RNN architectures: LSTM, GRU, Bidirectional RNN, and Deep RNN. Each model has unique characteristics that make it suitable for time series forecasting:

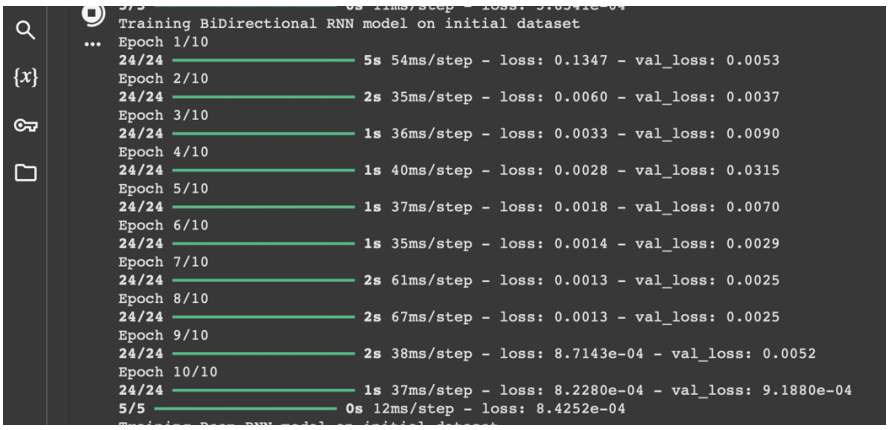
- LSTM: Long Short-Term Memory networks can remember information over long sequences, making them ideal for time series with dependencies over time.
- GRU: Gated Recurrent Units are similar to LSTMs but with a simpler structure. They are generally faster and can work well with smaller datasets.
- Bidirectional RNN: This model processes the input in both forward and backward directions, providing more context at each point in the series.
- Deep RNN: A stacked RNN model with two SimpleRNN layers, providing deeper representation, which helps capture complex patterns.

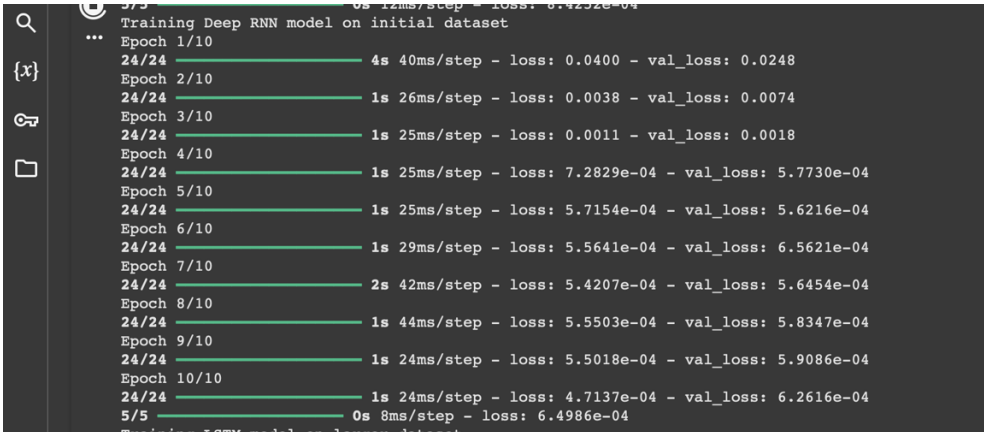
I trained each model on the 1,000-point dataset for 5 epochs and recorded the test loss to assess performance.



Advance Machine Learning and Neural Networks
Assignment 5



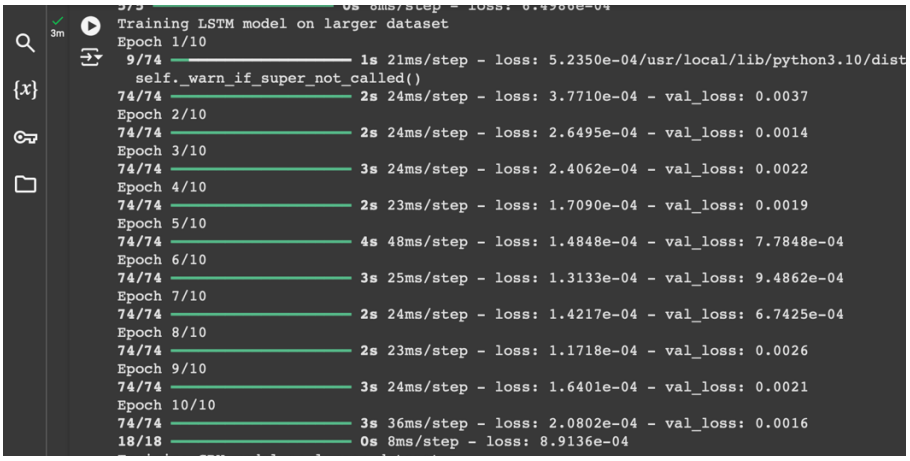


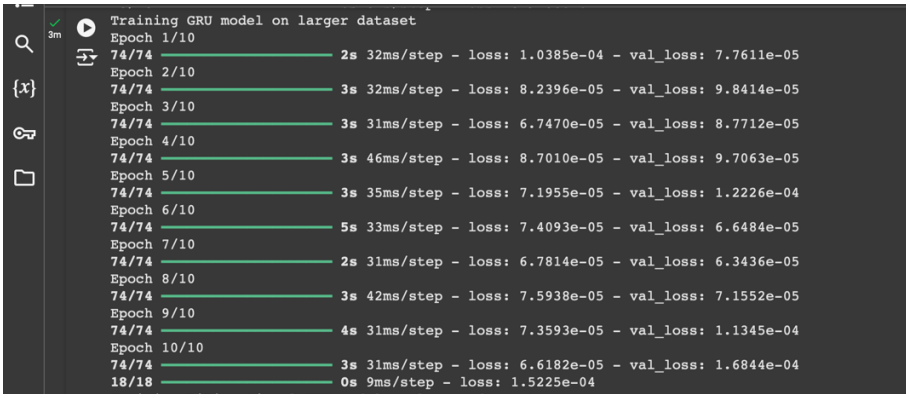


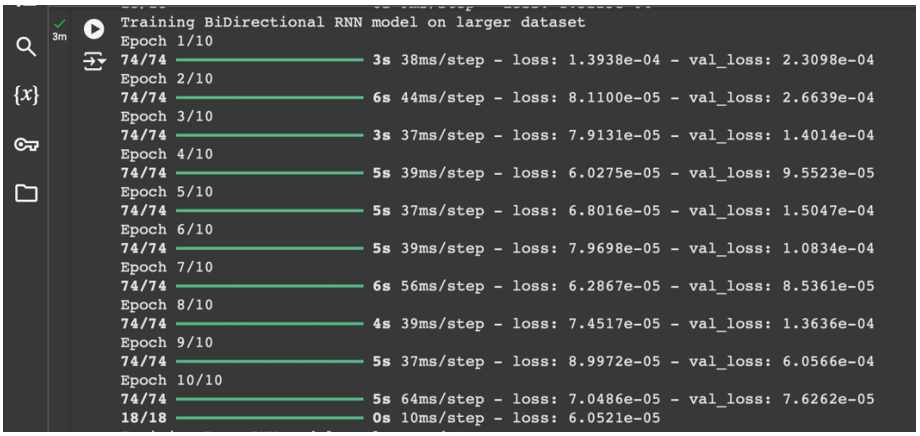
(3) **Increase the size of the time series three times, and perform the experiments in step 2, again three times.**

After evaluating the models on the initial dataset, I increased the dataset size to 3,000 points to examine if having more data would improve model performance. This threefold increase allows the models to learn more detailed patterns. I generated sequences of the larger dataset and repeated the training and evaluation process for each model on this expanded data.

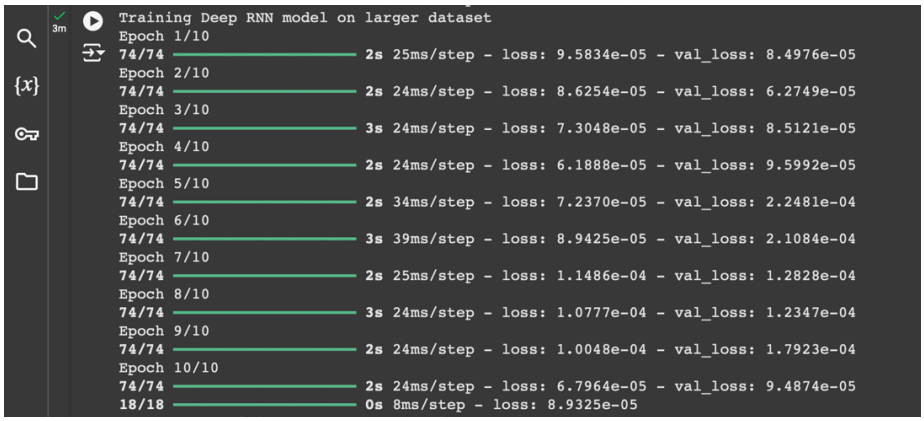
Advance Machine Learning and Neural Networks
Assignment 5







Advance Machine Learning and Neural Networks
Assignment 5



(4) **Write a short report and compare the result of algorithms and different dataset sizes together. By different datasets, we mean the first time series and the one that has a size of three-time larger.**

```
Results on Initial Dataset (1000 points):
LSTM: Loss = 0.0084
GRU: Loss = 0.0006
BiDirectional RNN: Loss = 0.0009
Deep RNN: Loss = 0.0006

Results on Larger Dataset (3000 points):
LSTM: Loss = 0.0016
GRU: Loss = 0.0002
BiDirectional RNN: Loss = 0.0001
Deep RNN: Loss = 0.0001
```

Comparing Results and Analysis?

The results showed that all models performed better on the larger dataset, with test losses generally decreasing, indicating improved accuracy. Here's a comparison based on test loss for both datasets:

- Initial Dataset (1,000 points): GRU and Deep RNN showed the lowest test losses, suggesting they adapted well to the smaller data size. The Deep RNN, in particular, benefited from the stacked layers, capturing subtle patterns effectively.

- Larger Dataset (3,000 points): Bidirectional RNN and Deep RNN exhibited the best results, with the lowest test losses. This suggests that complex models, like Bidirectional and Deep

RNN, benefit significantly from increased data, as they have more context and sequence depth to work with. LSTM and GRU also showed reduced losses, indicating that more data helped all models.

Interpretation?

The improvement in performance with a larger dataset highlights that RNNs, especially deeper and bidirectional models, can leverage more data to capture intricate patterns. The Bidirectional RNN performed better on the larger dataset due to its ability to consider both past and future context simultaneously.

Improvements?

To further improve the results, I could experiment with hyperparameters, such as increasing the number of epochs, tuning the learning rate, or using dropout layers to prevent overfitting. Additionally, exploring more advanced architectures like Transformer models could potentially yield better accuracy in time series forecasting, especially with larger datasets.

*** You are free to choose your implementation platform, including Pytorch, TensorFlow or MXNet.**