What is a Neural Network?

Simple Definition

- **A Neural network is essentially a series of mathematical algorithms designed to recognize patterns within data.** When interpreting data, they can classify or cluster it. Although they can be used to recognize patterns in real-world data such as images, sound, and text, they can only recognize patterns in numbers within vectors, so the real-world data will have to converted or translated.

Behavior

- Neural networks are able to find correlations within data. When finding correlations, they can either classify and cluster the data.
- Classification is when a neural network groups certain data under a set of labels. These labels will have to be defined by humans in order for the neural network to determine the correlation between these labels and the data. This is also referred to as supervised learning.
  - For example, you can provide a neural network with a set of geometric shapes as labels, such as squares, triangles, hexagons, etc. Each shape will have their own set of features, like how a square has 4 sides and 4 corners. Knowing the characteristics of these shapes, the neural network can then group the data into those shapes/labels.
- In some cases, the labels used in classification can be used to label events happening in time. You can assign labels to world events that took place in the past and future. Based on these events/labels and a sufficient set of a data, the neural network can now make correlations between past, present, and future events.
  - For example, if you have a set of recorded events that have happened to a person suffering from a medical disease, you can have the neural network train on this data and predict health breakdowns.
- Clustering is essentially the detection of similarities within data. When clustering, a neural network isn't classifying data based on labels or predefined characteristics. Instead, it is trying to find raw similarities and differences in the data. This is also referred to as unsupervised learning.
  - For example, in this case, you wouldn't provide the neural network with any of the characteristics or labels of a given list of geometric shapes; instead, the neutral network will point out what similarities and generalizations in the data IT sees.

Structure

- A neural network is made up of layers of nodes. These nodes are where the computations take place.
- When receiving inputs, a node will take these inputs and combine it with their corresponding weights to either amplify or dampen it. These weights are based on their input's relative importance to the other inputs, which could be adjusted by humans to favor different inputs. The combined values are then summed up to be passed through the node's activation function. The activation function will determine if and to what certain extent the signal should be sent down the neural network. If the signal is sent out, then the node has been "activated".
- These nodes then make up node layers, with each layer's output also being the input of the next layer.
- With this in mind, Deep Neural Networks or Deep Learning are defined literally as neural networks with 3 or more hidden layers (the layers that are neither the input or output layers). To keep things simple, Deep Learning concepts will not be directly discussed or defined.

Recurrent Neural Networks and Feed-Forward Networks

Explaining Feed-Forward Networks

- Before going over Recurrent Neural Networks, it is crucial that you understand what Feed-Forward Networks are first.
- A Feed-Forward Network (FFN) is a neural network where the nodes feed information straight through the network: the information will never touch a given node twice. To contrast, Recurrent Neural Networks (RNN) cycles information in a loop, hence the word “recurrent” in the name. This will be elaborated upon soon.
- FFN’s transform their input examples to create an output, and in the case of supervised learning or classification, that output would be a label to categorize the input.
  - For example, going back to the example with geometric shapes mentioned earlier, the FFN will map it’s input or raw incoming data to categories, so when receiving raw data that relates to shapes (e.g. # of sides, corners, etc.), it will try to label the incoming data based on the it’s previous knowledge of the characteristics of certain shapes.
  - A Feed-Forward Network will train on labeled data until it minimizes the error made when guessing the categories of that data. When minimizing the error, it will try to adjust the weights of its nodes based on the correctness of its guesses.
- Once the FFN is trained on a dataset, like the example data relating to a set of specific geometric shapes, it can then be used to categorize data that it has never seen before.
  - However, when categorizing that data, an FFN will not attempt to classify a piece of data based on how it classify another. For example, when regarding geometric shapes, if it classifies a piece of data as a triangle, it will not classify another piece as a rectangle, square, or circle because of the triangle. Due to this, a Feed-Forward Network does not recognize the order of the data based on time, hence the “feed-forward” part of its name.

Explaining Recurrent Neural Networks

- To contrast, a Recurrent Neural Network doesn’t just input the current piece of data given to them, but it also inputs their previous classifications/decisions; therefore, a decision made on one moment will affect the decision made on the next moment. This loop of cycling information is the main difference between FFN’s and RNN’s.
- The sequential pieces of data are kept in the hidden state of an RNN, retaining relevance for several “time steps” as it affects the decision made for each new piece of data.

What is a Virtual Machine?

- Before considering what a virtual machine is, first consider a typical personal computer, such as a laptop, gaming computer, etc. Now, on that computer, picture an app running that has a little virtual, mini computer running within a window.
  - That is essentially what a virtual machine is: a virtual system that is being emulated by a host system. A portion of the host system’s resources, such as the CPU, GPU, RAM, Storage Medium, and so on will be used to power the virtual system. The virtual system is isolated from any other program, process, or service running on the host system.
- To create, manage, and use virtual machines, a virtual machine “hypervisor” program needs to be installed on your operating system.
  - Using the hypervisor, the specifications of a virtual machine can be tweaked and adjusted, similar to modifying a real computer. You can decide exactly how many CPU threads will be used, or RAM, or GPU video memory, etc. Furthermore, you can install whatever operating system (e.g. Windows 7, Windows 10, Mac OS X, Android) you please on that virtual machine.
  - **In the case of the scientific experiment for the science fair, the CPU execution cap will be adjusted between two virtual machines, which is basically the maximum amount of time that the host machine’s CPU will spend emulating the virtual machine’s CPU.**

What is a Neural Network?

Before the Experiment...

During the Experiment...

After the Experiment...

Question

Will the specifications of the processor of a computer training a Recurrent Neural Network (RNN) with a text-based database affect the rate at which the RNN will train at?

Hypothesis

The specifications of the processor of a computer training an RNN with a text-based database affects the rate at which the RNN will train at.

Variables

Independent Variable

- A virtual machine with a certain list of specifications that has a 100% execution cap for it’s CPU. These specifications will be elaborated upon later.

Dependent Variable

- The rate at which the RNN’s running on the virtual machines will be trained. This will be determined by the delay in seconds between each checkpoint created by the RNN’s, which will be elaborated upon later.

Control Variable

- The virtual machine that is the same as the virtual machine in the independent variable, but will have a 50% execution cap instead of 100%.

Prediction and Answer to Hypothesis

Prediction

- Prior to starting the experiment, I predicted that the specifications of the processor of a computer training an RNN with a text-based database will significantly affect the rate at which the RNN will train. Therefore, the virtual machine with the greater execution cap will be able to train the RNN much quicker than the other virtual machine.

Answer to Hypothesis

- Prior to starting the experiment, this is my answer to the hypothesis: The specifications of the processor of a computer training an RNN with a text-based database does affect the rate at which the RNN will train at.

Procedure with Materials

The following materials are listed in order of when they were set up to be used in the experiment.

1. **A personal computer**
   - The computer that will be used has an Intel i7-6700K CPU, a total of 12 GB for DDR4 RAM, an nVidia GeForce 980 Ti GPU, and a Samsung 256 GB SSD (SATA 6Gb/s) as a main OS drive with Windows 10 (along with a 1TB Seagate HDD over SATA 6Gb/s)
2. **A virtual machine manager program**
   - Oracle VM-Virtualbox will be the virtual machine hypervisor/manager program that will be used for this experiment.
   - Once Virtualbox is installed, two virtual machines will be set up.
     - For their specs, each virtual machine will have 2 CPU threads and 4 GB of RAM.
     - As for more specific specifications, each virtual machine has PAE/NX disabled, VT-x/AMD-V enabled, Nested Paging enabled, 3D video acceleration disabled, 2D video acceleration disabled, USB disabled, emulated PS/2 keyboard and mouse peripherals enabled, and a total of 17 MB of video memory (VRAM).
     - The Ubuntu Mate 18.04.2 Linux operating system will be installed on each virtual machine, along with its latest updates for the system and built-in programs.
3. **Important software**
   - For the sake of putting an RNN on the virtual machines and convenience, the following Linux software packages were installed: pip, virtaulenv, git, cmake, readline dev libraries, and samba. Everything but samba was mandatory for the RNN to work.
4. **An RNN program**
   - The actual RNN related software comes in the form of a modified version of the RNN system, TORCH-RNN, that was coded with the LUA and Python programming languages. The RNN program can train RNN models on using a GPU, but we are going to train in CPU mode, due to the nature of the hypothesis.
   - The installation process of TORCH-RNN and it's required software took place on both virtual machines.
   - There were numerous programming related dependencies and packages that had to be installed for the program, but the complete details for the full installation can be found on *https://github.com/sharpie7/torch-rnn-4-rfc*.
5. **All of Shakespeare's Tragedies compiled into a text file.**
   - To be used as the training dataset for the RNN's, a text file containing the entirety of William Shakespeare's tragedies was created. These tragedies include Antony and Cleopatra, Coriolanus, Hamlet, Julius Caesar, King Lear, Macbeth, Othello, Romeo and Juliet, Timon of Athens, and Titus Andronicus.
   - The text file would then be copied over onto the virtual machines.
6. **A custom Python program written by me**
   - To monitor the checkpoint files created by the RNN's when training, a custom Python program was written by me to record the time stamps of those checkpoints.
   - The python program would then be copied over onto the virtual machines.

Procedure with Experiment

The following steps are listed in order of when they were executed in the experiment.

1. **The pre-processing of the dataset (the text file of Shakespeare's tragedies)**
   - To reiterate, an RNN cannot directly input anything that isn't raw numerical data. Due to this, the file containing Shakespeare's tragedies, which entirely contains raw text, has to be translated on each of the virtual machines.
   - The result of this translation was a set of two files on each virtual machine that were then directly inputted into the RNN's.
2. **The training of the RNN**
   - Once all the steps involving materials and adjustments were done, the RNN's were now ready to be trained.
   - To reduce the potential bias of running the virtual machines on the host machine simultaneous, the RNN's on their respective virtual machines were trained one at a time.
   - The RNN's were trained with a configuration of 256 nodes and 3 node layers, including a checkpoint interval of 25 iterations. This means that when an RNN model updates 25 times, a checkpoint and time stamp is made. **Furthermore, the RNN's were ran on CPU mode, not GPU mode.**
     - Although there were other adjustments made before training, such as the 50 sequences used in a minibatch, a max epoch of 50 and so on, the main focus is on the number of nodes and node layers.
     - Almost immediately after the RNN's began their training, the custom python programs on the virtual machines were executed to monitor the training process and record their findings in their own text files.
   - The training of the RNN's was the last major step in the experimental process, but technically wasn't the last step in the entire science fair project, for an analysis, conclusion, etc. had to be made afterwards.
3. **IMPORTANT NOTE: the concern over the number of trials this experiment would have was excused by Mrs. Woods, due to the nature of the experiment. This means that only one trial was done.**

Analysis

- **The RNN within the virtual machine without the handicapped CPU performed better than the handicapped virtual machine.**
  - The virtual machine with the 100% execution capped CPU performed better than the virtual machine with the handicapped, 50% execution capped CPU, and here are the pieces of data that show why.
    - Based on the tables representing the delay in seconds between every checkpoint and the checkpoint prior, no delay value of the handicapped RNN is less or equal to <u>any</u> of the delay values of the non-handicapped RNN. This means that the handicapped RNN always had a greater latency between checkpoints than the non-handicapped RNN; therefore, the non-handicapped RNN trained faster and with less latency, for the lower the latency/delay, the faster the training.
    - Based on the graph representing the delay values of the tables with their corresponding checkpoint, no bar of the handicapped virtual machine has a less or equal length to <u>any</u> of the bars of the non-handicapped virtual machine. This means that the handicapped virtual machine always had a greater latency between checkpoints than the non-handicapped virtual machine. This brings us to the same conclusion as the tables: the non-handicapped virtual machine trained faster.
- **The percentage of the CPU execution cap of the virtual machines seemed to have directly affected the performance of the RNN's.**
  - As it was already stated, the non-handicapped virtual machine performed better than the handicapped machine; however, the handicapped RNN's delay values represented by the table and graph were always almost twice the value of those from the non-handicapped RNN.
  - To reiterate, the lower the latency/delay, the faster the training of the RNN. If the delay values of the handicapped RNN were twice the values of the non-handicapped RNN, then the non-handicapped RNN was twice as fast of the handicapped RNN.
  - Furthermore, the 100% execution cap of the non-handicapped virtual machine was twice the percentage of the 50% execution cap for the handicapped virtual machine.
  - Therefore, the percentage of the execution caps for the CPU's directly affected the percentage of the performance of the RNN's (for example, if the execution cap for a hypothetical virtual machine was 75%, then the performance of the RNN training would be 75% the performance of a 100% execution cap.

Conclusion

- Based on the Analysis, the specifications of the processor of a virtual machine, more specifically the CPU execution cap, directly affected the rate at which the RNN's under those virtual machines trained at.
- Therefore, t**he specifications of the processor of a computer training an RNN with a text-based database does affect the rate at which the RNN will train at.**

Discussing the possibility of Error

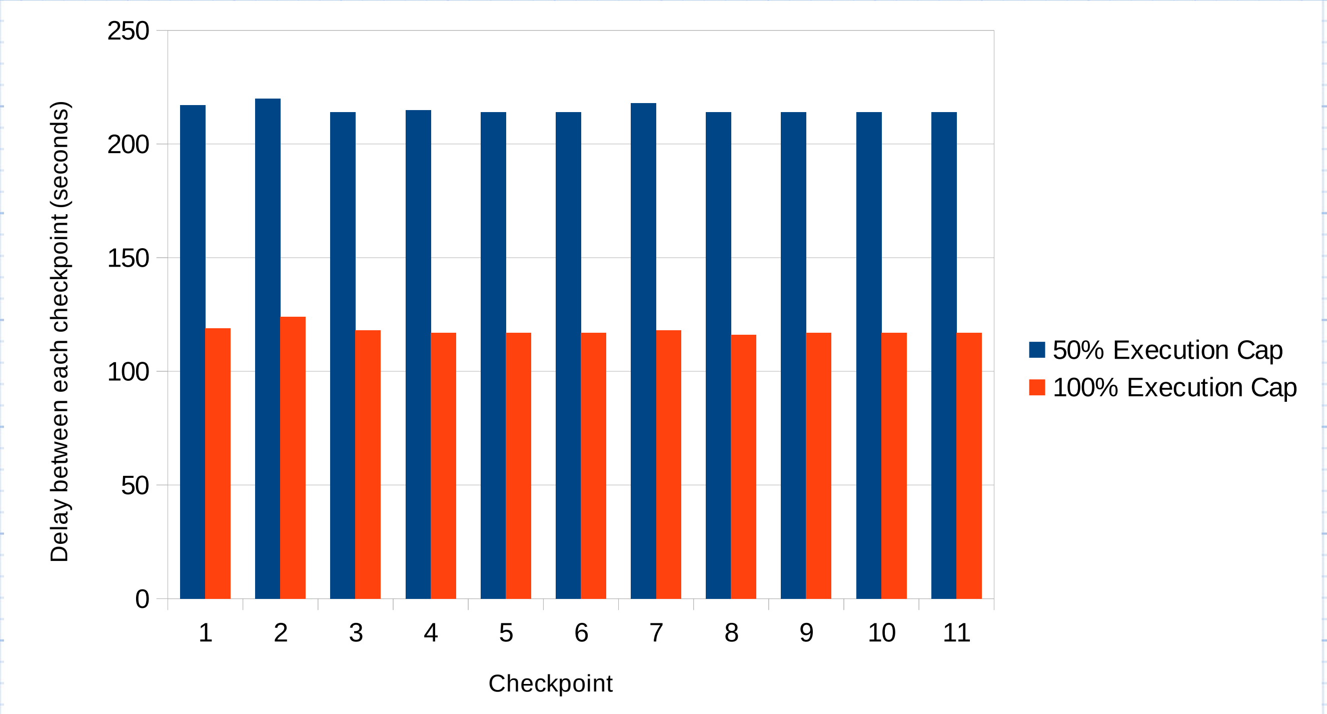
- When regarding the materials, dependent variable, and execution of the experiment, there is a possibility for a somewhat significant error.
  - If someone were to replicate this experiment with a different physical hardware, especially a host computer with different specifications, their execution of the experiment may yield different results. More specifically, the performance difference between the handicapped RNN and the non-handicapped RNN may be different. This is not necessarily an error, but an important point to make.
  - When regarding the software that was installed on the virtual machines, there was a custom python program that was written. Due to how I, the creator of this experiment and project, wrote that program myself and is therefore custom to the chosen RNN program, an unintentional bias to the results of the experiment may have been made.
  - The dependent variable, more specifically the delay value between checkpoints that was being tested for, may not accurately tell the whole story when comparing the performance between the RNN's, and a better variable may exist.
  - The background processes on the host computer may have varied between the training sessions of the virtual machines' RNN's, which would result in unintentional bias between the results of the experiment.

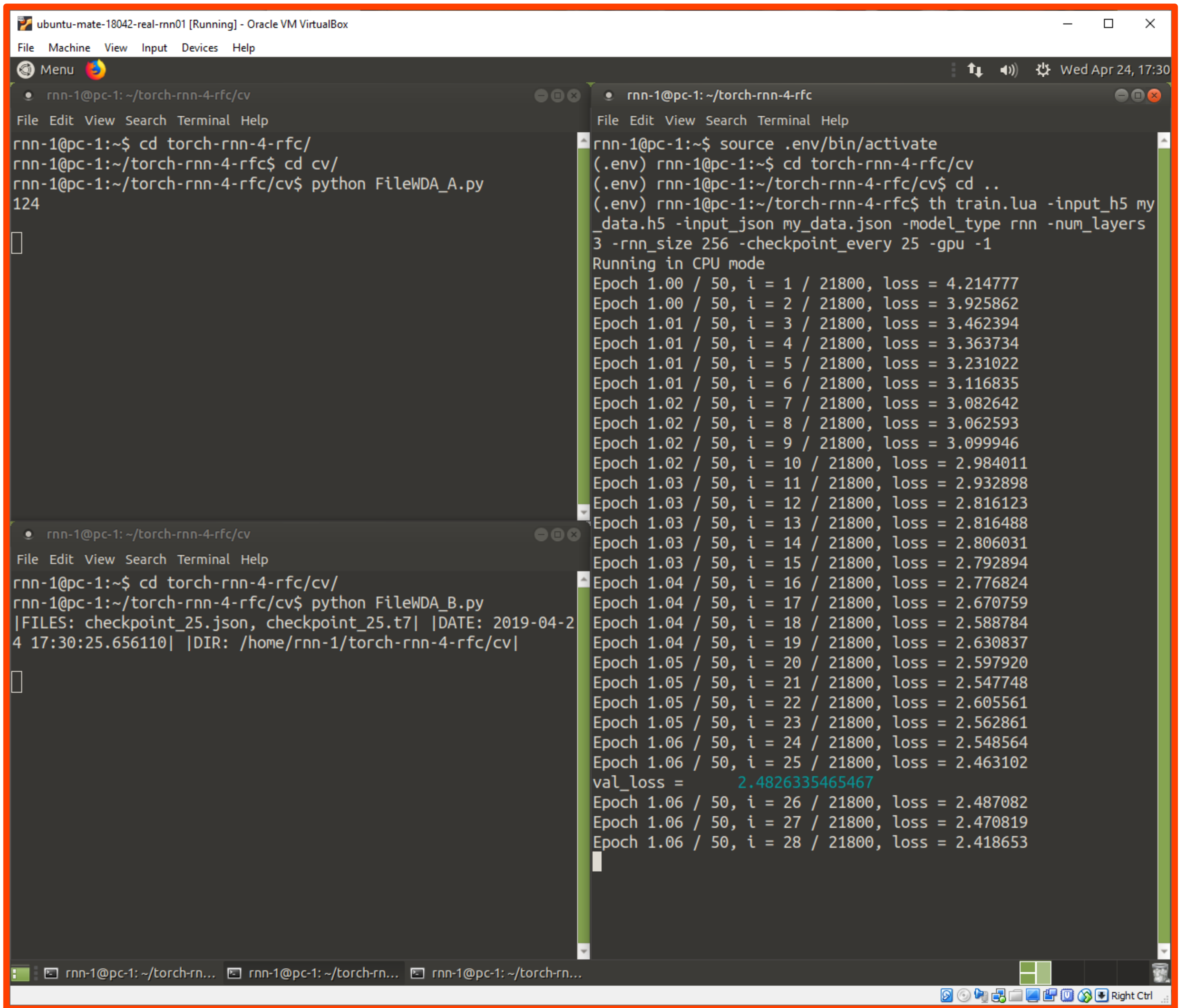
Tables

50% Execution Cap	
Checkpoint	**Delay (sec)**
1	217
2	220
3	214
4	215
5	214
6	214
7	218
8	214
9	214
10	214
11	214

100% Execution Cap	
Checkpoint	**Delay (sec)**
1	119
2	124
3	118
4	117
5	117
6	117
7	118
8	116
9	117
10	117
11	117

Chart




ubuntu-mate-18042-real-rnn01 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Menu
Wed Apr 24, 17:30
rnn-1@pc-1: ~/torch-rnn-4-rfc/cv
File Edit View Search Terminal Help
rnn-1@pc-1: ~/torch-rnn-4-rfc
Right Ctrl

```
rnn-1@pc-1:~$ cd torch-rnn-4-rfc/
rnn-1@pc-1:~/torch-rnn-4-rfc$ cd cv/
rnn-1@pc-1:~/torch-rnn-4-rfc/cv$ python FileWDA_A.py
124
```

```
rnn-1@pc-1:~$ cd torch-rnn-4-rfc/cv/
rnn-1@pc-1:~/torch-rnn-4-rfc/cv$ python FileWDA_B.py
|FILES: checkpoint_25.json, checkpoint_25.t7| |DATE: 2019-04-2
4 17:30:25.656110| |DIR: /home/rnn-1/torch-rnn-4-rfc/cv|
```

```
rnn-1@pc-1:~$ source .env/bin/activate
(.env) rnn-1@pc-1:~$ cd torch-rnn-4-rfc/cv
(.env) rnn-1@pc-1:~/torch-rnn-4-rfc/cv$ cd ..
(.env) rnn-1@pc-1:~/torch-rnn-4-rfc$ th train.lua -input_h5 my
_data.h5 -input_json my_data.json -model_type rnn -num_layers
3 -rnn_size 256 -checkpoint_every 25 -gpu -1
Running in CPU mode
Epoch 1.00 / 50, i = 1 / 21800, loss = 4.214777
Epoch 1.00 / 50, i = 2 / 21800, loss = 3.925862
Epoch 1.01 / 50, i = 3 / 21800, loss = 3.462394
Epoch 1.01 / 50, i = 4 / 21800, loss = 3.363734
Epoch 1.01 / 50, i = 5 / 21800, loss = 3.231022
Epoch 1.01 / 50, i = 6 / 21800, loss = 3.116835
Epoch 1.02 / 50, i = 7 / 21800, loss = 3.082642
Epoch 1.02 / 50, i = 8 / 21800, loss = 3.062593
Epoch 1.02 / 50, i = 9 / 21800, loss = 3.099946
Epoch 1.02 / 50, i = 10 / 21800, loss = 2.984011
Epoch 1.03 / 50, i = 11 / 21800, loss = 2.932898
Epoch 1.03 / 50, i = 12 / 21800, loss = 2.816123
Epoch 1.03 / 50, i = 13 / 21800, loss = 2.816488
Epoch 1.03 / 50, i = 14 / 21800, loss = 2.806031
Epoch 1.03 / 50, i = 15 / 21800, loss = 2.792894
Epoch 1.04 / 50, i = 16 / 21800, loss = 2.776824
Epoch 1.04 / 50, i = 17 / 21800, loss = 2.670759
Epoch 1.04 / 50, i = 18 / 21800, loss = 2.588784
Epoch 1.04 / 50, i = 19 / 21800, loss = 2.630837
Epoch 1.05 / 50, i = 20 / 21800, loss = 2.597920
Epoch 1.05 / 50, i = 21 / 21800, loss = 2.547748
Epoch 1.05 / 50, i = 22 / 21800, loss = 2.605561
Epoch 1.05 / 50, i = 23 / 21800, loss = 2.562861
Epoch 1.06 / 50, i = 24 / 21800, loss = 2.548564
Epoch 1.06 / 50, i = 25 / 21800, loss = 2.463102
val_loss =      2.4826335465467
Epoch 1.06 / 50, i = 26 / 21800, loss = 2.487082
Epoch 1.06 / 50, i = 27 / 21800, loss = 2.470819
Epoch 1.06 / 50, i = 28 / 21800, loss = 2.418653
```

Image - This is a virtual machine training an RNN under the specifications of the experiment, more specifically a 100% CPU execution cap; however, the training session in the image is a test, not the real deal.

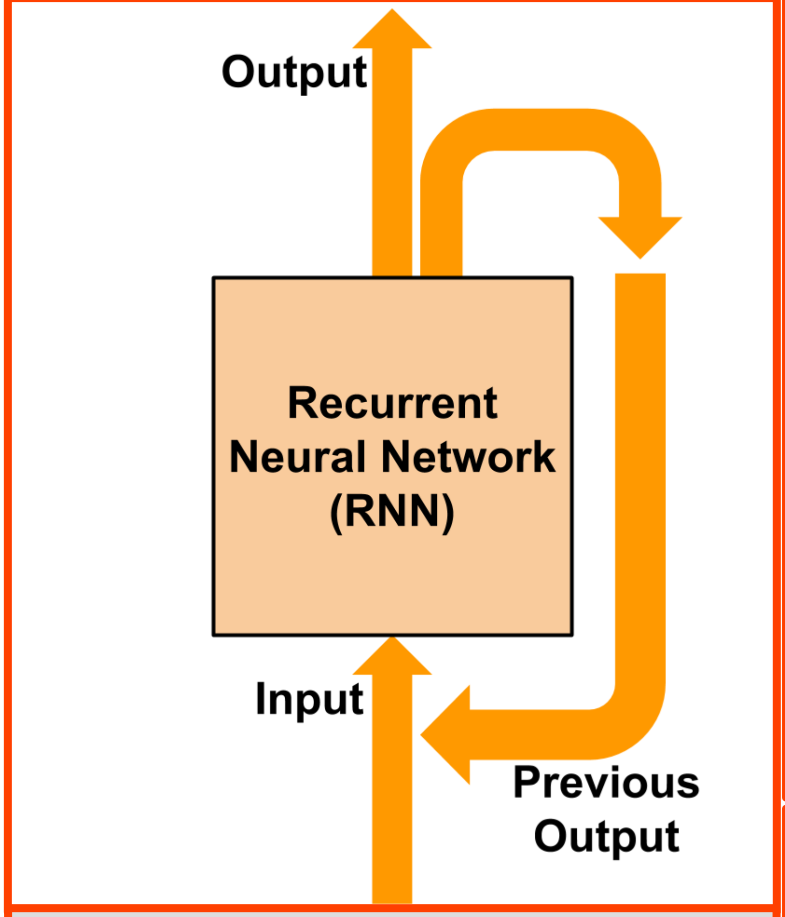


Image - This is a basic visual representation on how data flows through an RNN.

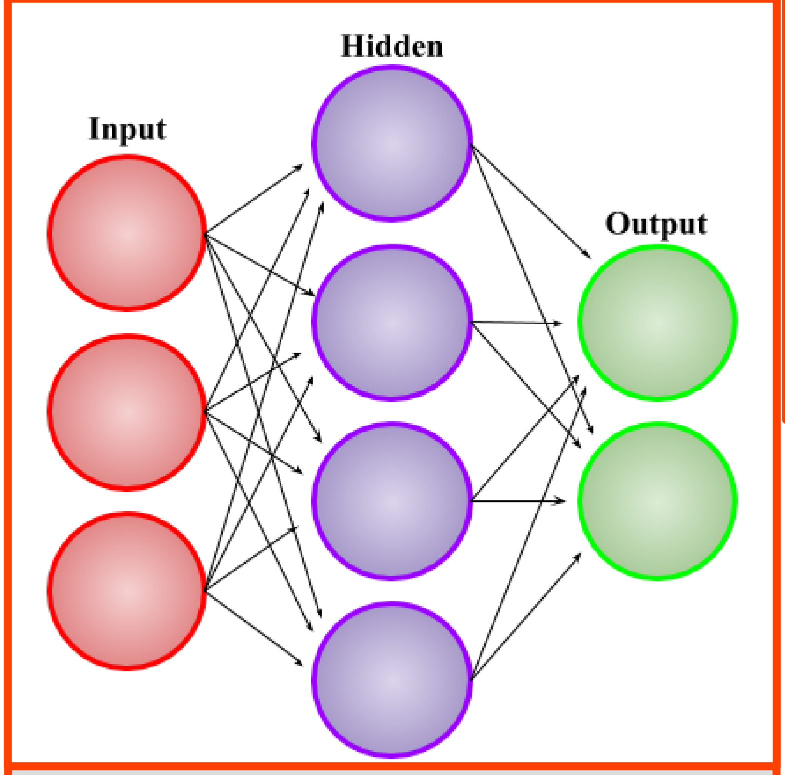


Image – This is a visual representation of a neural network.

Machine Learning On Virtual Machines

A Project by Eiza Stanford of Mrs. Wood's 3rd Hour Class, 9th Grade