# Paper

Andrej Karpathy, Justin Johnson, Li Fei-Fei. “Visualizing and Understanding Recurrent Networks”.

# Abstract

In our project, we tried to reproduce most of the paper’s figures (elaborated in part 04) using our own implementation of the paper’s Recurrent Neural Networks with the help of ‘PyTorch’.  
We managed to reproduce the test set cross-entropy loss and accuracy table for all models (figure 1), graphic cell and gate visualizations (figure 2 & 3) and lastly, venn diagrams of three studied networks (figure 4).

Given the lack of source code which wasn’t supplied by the researches and the fact that all the figures depend on the data from figure 1, we think that most of our figures represent our results from figure 1 reliably. We do believe, that if we managed to achieve the results that the researches got, our plots would be consistent with the paper.

# Introduction

The paper “Visualizing and Understanding Recurrent Networks” tries to shed a light on the source of performance and the limitations of Recurrent Neural Networks (RNNs) along with their variants (LSTM and GRU). In order to analyze these models, the researchers used character-level language models (LM’s) as a testbed. In particular, the experiments revealed the existence of interpretable cells that keep track of long-range dependencies. They compared their analysis against finite horizon n-gram models which supposedly lack long range structural dependencies. The paper presents those results and suggests areas for further study in the subject.

# Methodology

The researchers didn’t supply any of their source code for the networks or their analysis which meant the we needed to write it on our own. In order to mimic the experiments, we did use most of the parameters that were mentioned in the paper (part 3.3) and added some of our own to achieve best results and overcome technical difficulties.

In addition, we only used the enormous book “war and peace” by Leo Tolstoy as a dataset since the Linux Kernel source code has overblown since the paper was written and was too big for our machine.

We implemented the paper’s experiments using Python with two main frameworks, ‘PyTorch’ and ‘PyTorch-Lightning’.

For the training procedure we used ‘PyTorch-Lightning’ and for each RNN architecture we used ‘PyTorch’s own implementation. We will explore LSTM, GRU and RNN each with hidden size of 32, 64, 128, 256 with 1 to 3 layers. The hyper parameters for the training phase are mostly the same as the paper.

The hyper parameters we used:

* 50 Epochs
* Batch size of 100
* Sequence length of 100
* Dataset splits percentages of (80, 10, 10) for (training, validation, test)
* Learning rate of with gamma equals 0.95 and learning rate decay of 10.
* We decided to omit truncated back propagation because the training phase with it took too long and with basic hardware it just wasn’t feasible.

The analysis part of our project consists of two parts:

* Network simulation with gates and cells extraction: implement our own “forward pass” with additional gate extraction.
* Visualization methods for graphic data presentation.

We trained the networks using “Google Collaboratory” environment with the free GPU

hardware supplied by them. The analysis part ran on our CPU-only machine.

# Experiments

Firstly, we trained each model, layer and hidden size on the training set of the “War and Peace” dataset. For each model, when the training phase has ended, we saved the network locally in order to produce the following analysis.

Figure 1: Accuracy (top table) and Cross-entropy loss (bottom table) for the test set on each model. Models in each row have a nearly equal number of parameters. We took each saved network and ran the test set on the network and kept the mean loss and mean accuracy.

Figure 2: Cell Visualization – Helps us to visualize cells with interpretable activations in our “War and Peace” LSTMs and GRUs where the text color corresponds to , where -1 is red and +1 is blue. This experiment enables us to identify whether our networks can detect high-level patterns. We used a web interface supplied by “[huanghao-code](https://github.com/huanghao-code)” to showcase cell activation values for a selected cell in the entire network on a small section of the test set.

Figure 3: Saturation plots for 3-layered, 64-hidden-size and 128-hidden-size LSTM and GRU models. For the LSTM figure, the plots correspond to forget, input and output gates, while for the GRU figure, the plots correspond to update and reset gates. Each circle in the figure is a gate and its position is determined by the fraction of times it’s left or right saturated. We consider a gate to be left or right saturated if its activation is less than 0.1 or higher than 0.9 respectively. Red color represents the first layer, green- the second and blue- the third.

Figure 4: Venn diagrams – In the paper, the researchers tested the overlap between test set errors. We decided to test the overlap between test set correctness of our best RNN, GRU and LSTM networks. We loaded the three networks and for each one we saved its correct examples in a set data structure. Then, we compared each network’s set against the others in order to seek for similarities between the three.

# Results

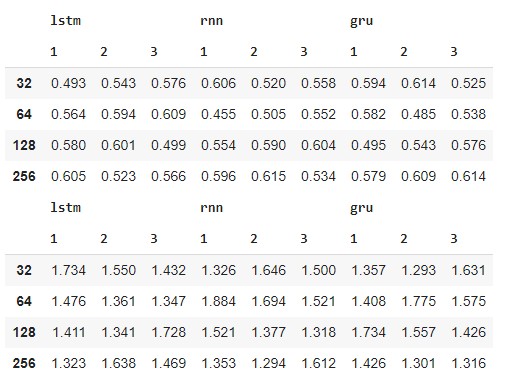


Figure 1

Our figure 1 reproduction attempt resulted in mixed results. We got better cross-entropy loss on LSTM models with hidden size of 64 and on RNN models with hidden size of 256 but on the others, we mostly got higher loss. In our project, we used the same parameters as the researchers detailed. This fact leads us to the conclusion that the difference between the results lays on other parameters that were not specified or different models’ implementation.

As in the paper, we also got to the conclusion that depth of at least two improves the results, highly beneficial and usually between depth 2 and 3 the results are mixed. Furthermore, in most cases the LSTM models outperform the RNN models.

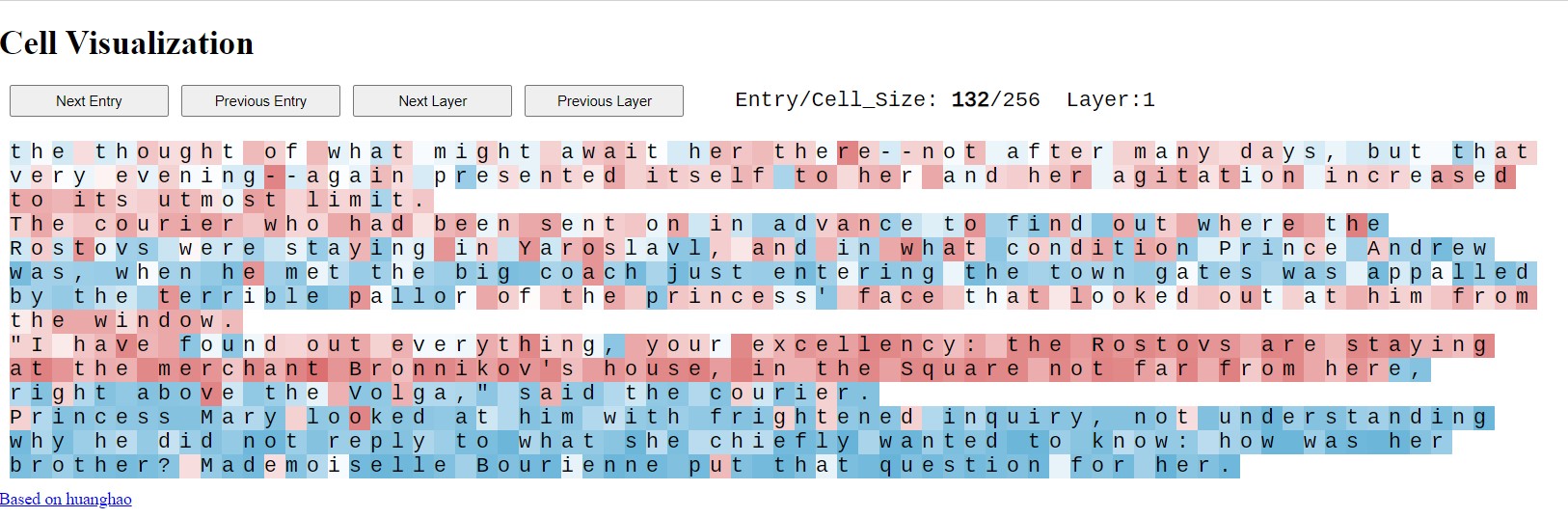


Figure 2

In the paper, the researchers have found some cells which are “responsible” for identifying specific patterns, like line length counter, parenthesis and more. They stated that these types of cells are exceedingly rare, since almost most of the cells don’t have a specific pattern. Due to the large data set, we couldn’t find those unique interpretable cells that represent those patterns. However, we are quite sure that some of these cells do exists in our LSTM models.

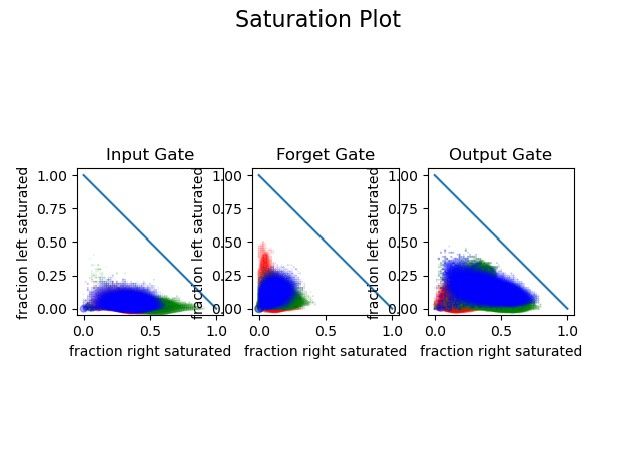
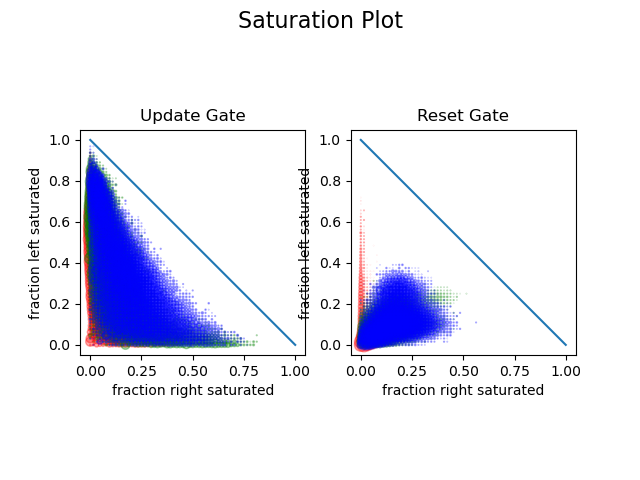
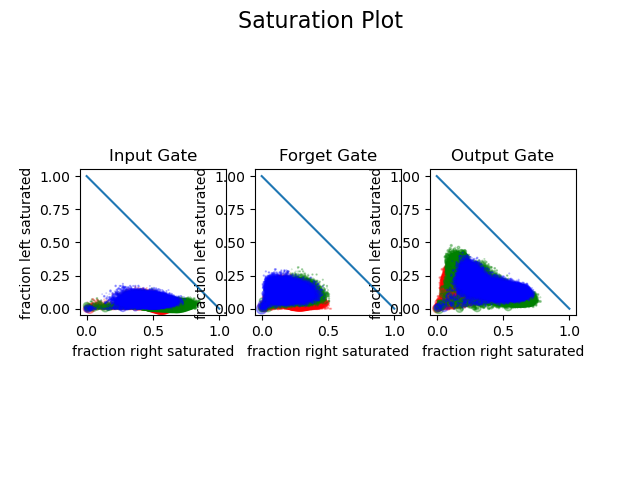
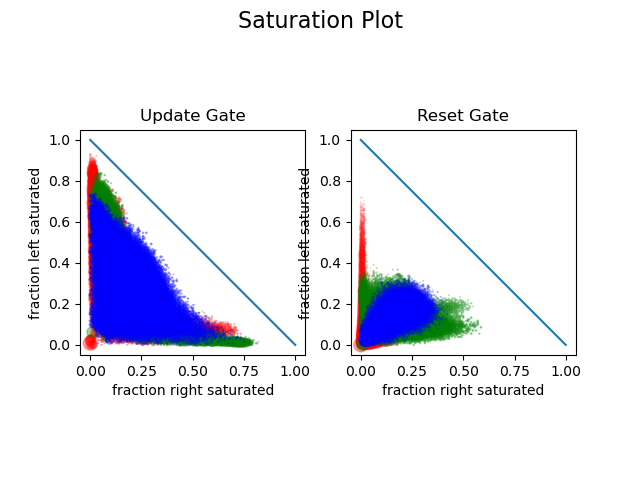


Figure 3.4 - GRU 128 Hidden Size Gate Visualization

Figure 3.3 – LSTM 128 Hidden Size Gate Visualization

Figure 3.2 – GRU 64 Hidden Size Gate Visualization

Figure 3.1 – LSTM 64 Hidden Size Gate Visualization

Firstly, for the LSTM: According to the paper, right and left saturated forget gates are particularly interesting. The almost-always right saturated gates correspond to cells that remember their values for very long time period. While almost-always left-saturated correspond to feed-forward cells.

The forget gates of the 128-hidden-size LSTM were more right-saturated than the forget gates of the 64-hidden-size LSTM. This means that the 128-LSTM gates remembered more information from previous calculations. But, in both networks, there were no forget-gate cells which always remembered previous calculations, and there were no cells who acted as completely feed-forward.

The output gates plots reveal that for either of the LSTMs, there are no cells that get consistently revealed or blocked to the hidden state.

Secondly, for the GRU: As the layer progresses, the update gate cells are less left-saturated and less right-saturated, meaning they keep information from previous calculations more. The reset gates are “indecisive” in layer 2 and 3, like the forget-gate cells of the LSTM.

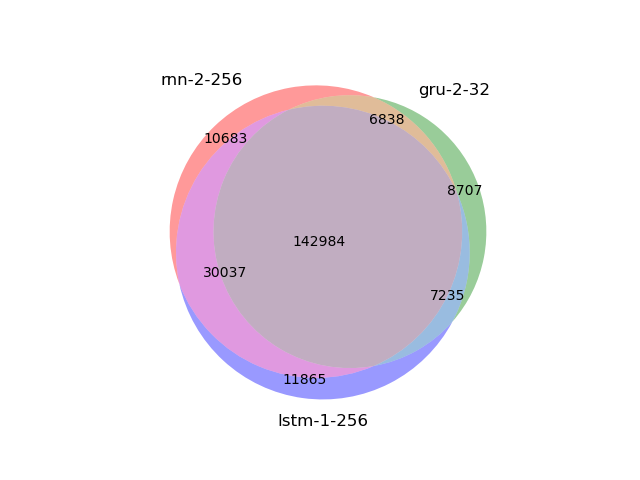
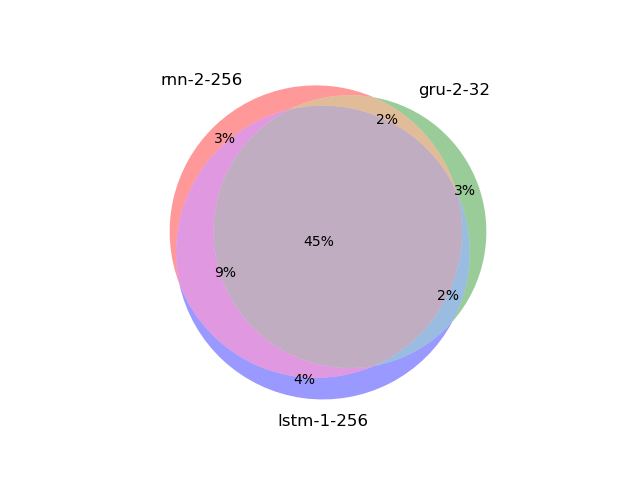


Figure 4.2 – Model Accuracy Comparison by Count

Figure 4.1 – Model Accuracy Comparison by Percentage

In our experiments we’ve dealt with recurrent neural networks which hold common principles. Under this knowledge, we can expect that these networks will perform mostly the same, which translate to common predictions on most of the examples (with slight variations). The above venn diagrams show that our networks behave as expected, since most of the correct predictions were common between all three models.