CS 291K: Introduction to Deep Learning

David Weinflash
Department of Computer Science
University of California, Santa Barbara
dweinflash@ucsb.edu

ABSTRACT

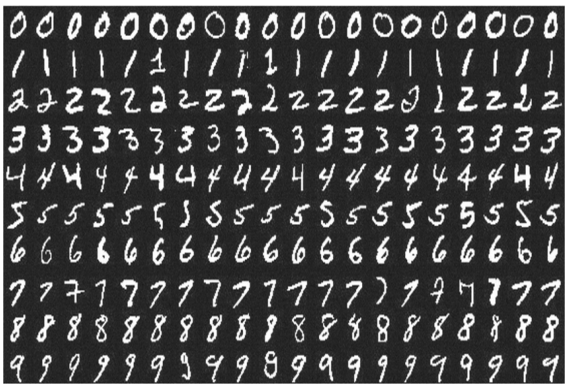
This project focuses on the performance of recurrent neural networks in the context of image generation and text classification. Specifically, multiple recurrent neural networks are trained in order to classify movie reviews from the Internet Movie Database and generate images from the MNIST dataset. The open-source software library TensorFlow and the processing power provided by Google Colab's GPU make up the main technologies of the project. Academic publications such as Francois Chollet's *Deep Learning with Python* and O'Reilly's *Using TensorFlow to generate images with PixelRNNs* are relied upon in order to generate neural networks. Models are evaluated in light of training and test data performance and also via manual verification methods. Finally, a discussion of the main takeaways from the project concludes the paper.

1 Data

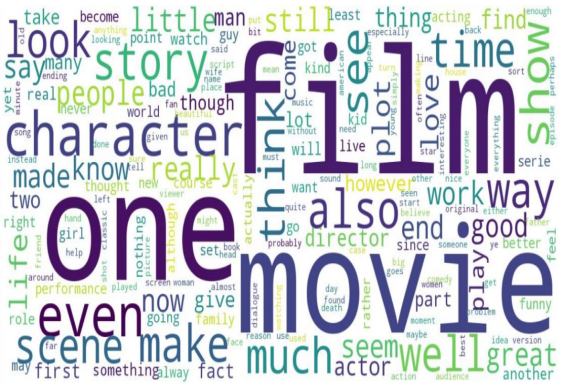
The MNIST and IMDB datasets were both used in order to train the recurrent neural networks. The MNIST dataset, well known in the machine learning community, is made up of handwritten digits (28x28 pixels) belonging to one of ten different categories (0-9). The IMDB dataset, on the other hand, consists of 25,000 movie reviews (encoded as a sequence of word indices) labeled based on sentiment.

Both the MNIST and IMDB datasets were imported from the Python library Keras. The handwritten MNIST images were encoded as Numpy arrays, stored as shape (60,000, 28, 28) of type `uint8` with values in the [0,255] interval. Each IMDB movie review, on the other hand, is represented by a single Numpy array of type `uint8` where each value corresponds to a word based on a key-value relationship in a dictionary. Indices in the array represent word prevalence, where a lower index corresponds to a higher word frequency.

In order to accommodate the recurrent neural networks, a small amount of pre-processing was necessary. In particular, the MNIST training and test images were transformed into `float32` arrays of shape (60,000, 28*28), where values were normalized so that all entries lied between 0 and 1. Likewise, the IMDB data set was transformed into a 2D Numpy array of shape (10,000, 500), where 10,000 total reviews were considered utilizing the 500 most common words from each review.



(a) The MNIST Dataset



(b) The IMDB Dataset Word Cloud

Figure 1: RNN Datasets

2 Network

In order to quantitatively measure the performance of a recurrent neural network, the IMDB dataset was analyzed first. Utilizing the SimpleRNN model from Keras, movie reviews were analyzed and then fed back into the six layer fully connected neural network to classify the text as either positive or negative. To increase the representational power of the network, several recurrent layers were stacked one after the other, with intermediate layers returning a full sequence of successive outputs from each timestamp. The last layer thus returned output corresponding to not only the last timestamp but all previous timestamps as well.

In order to teach the network, the *categorical_crossentropy* loss function was utilized. Used as a feedback signal for tuning the weight tensors during the training phase, *categorical_crossentropy* helped to minimize the value of the loss function during stochastic gradient descent. Additionally, the *RMSprop* algorithm was implemented as an optimizer to guide the gradient descent procedure and achieve the smallest possible loss function.

After numerically determining the accuracy of a recurrent neural network, model performance was next evaluated in the context of image generation. Specifically, a Pixel Recurrent Neural Network was implemented in order to model the distribution of the MNIST data set and consequently infer the pixels of an occluded hand written digit. Using a neighboring order from left-to-right and top-to-bottom, the PixelRNN model learned the conditional probabilities of each pixel through a series of LSTM layers and bounded convolutions. Relying on the premise that the intensity value of each pixel is dependent on all previously traversed pixels, the PixelRNN model inferred the values of occluded pixels based upon a bounded receptive field around each unknown pixel.

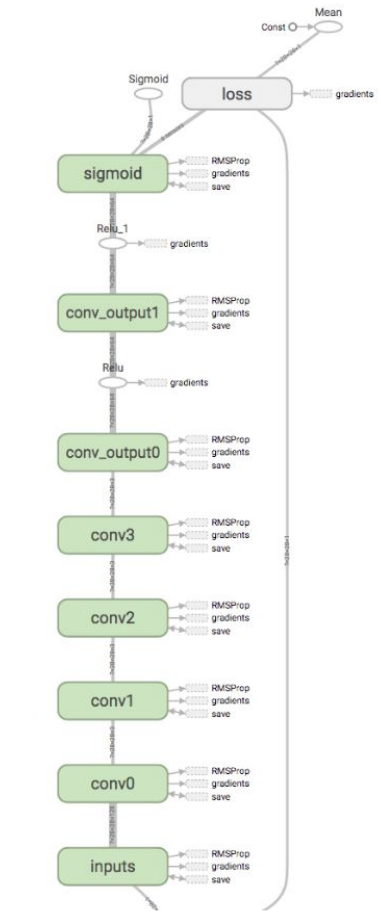


Figure 2: The PixelRNN Network

Perhaps the most powerful feature of the PixelRNN was its ability to preserve learned information using residual connections, where output from shallow layers in the network are copied and concatenated with the output from deeper levels. Such a design strategy allowed the model to increase its depth while maintaining a relatively high level of accuracy, an accomplishment the SimpleRNN model was unable to achieve when predicting movie reviews.

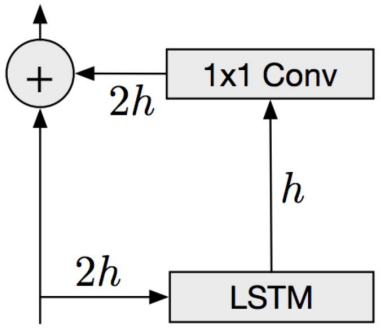
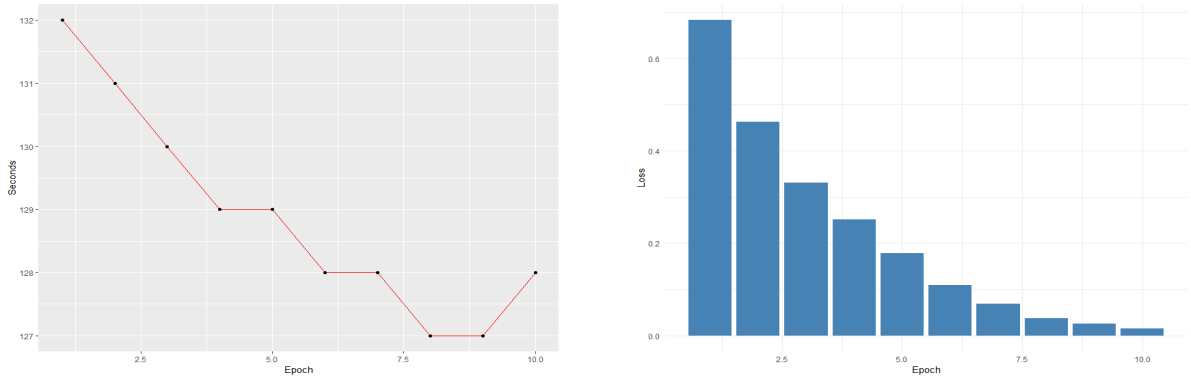


Figure 3: The Residual Connection

3 Training

During SimpleRNN training, the network iterated over the training data (10,000 reviews and 10,000 labels) in mini-batches of 128 samples, repeating the procedure for a total of 10 epochs. For each iteration, the network computed the gradients of the weights in reference to the loss on the batch and updated the weights accordingly. Gradient updates were performed during each epoch (469 updates per epoch), accumulating to 4,690 total gradient updates over the course of the training procedure. See the figure below for the timing and loss measurements for each training epoch.



(a) Time (b) Loss

Figure 4: Time and loss as a function of training epoch

During PixelRNN training, analysis of the handwritten digits and pixel prediction occurred in parallel. Similar to the SimpleRNN model, the *cross entropy* loss function was utilized to minimize the loss between predictions and actual pixel values. The *RMSProp* optimizer was also used, again optimizing the gradient descent procedure to achieve the smallest possible loss function and stabilize learning.

4 Validation

In order to monitor the accuracy of the SimpleRNN model on unfamiliar data, 5,000 samples (or 20% of the original data) were collected and set aside to act as a validation set. The network was then trained for 10 epochs in mini-batches of 128 samples, measuring the classification performance of the model on the training data in conjunction with the performance on the validation set.

Based on the loss and accuracy of the model on the validation set, it is clear that the model began overfitting the data after the *fifth* epoch. In other words, after the fifth epoch, the model was over optimized on the training data, or dependent on representations that were specific to the training data but do not generalize well to data outside of the training set. In terms of statistics, training performance peaked at epoch 10 with 99.51% accuracy and 1.6% loss, while validation performance peaked at epoch 5 with 83.6% accuracy and 2.2% loss.

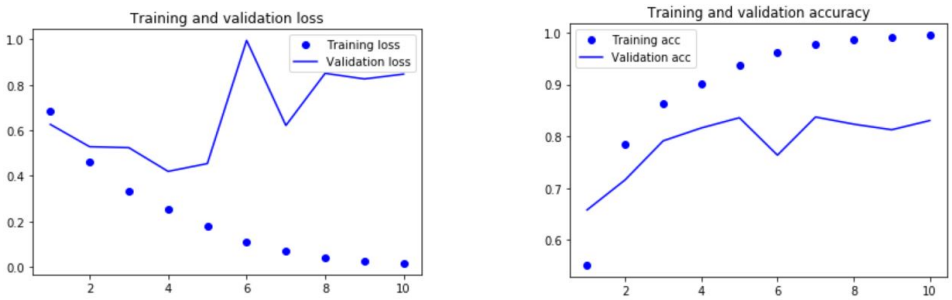


Figure 5: Loss and accuracy of the network on training and validation sets

5 Visualization

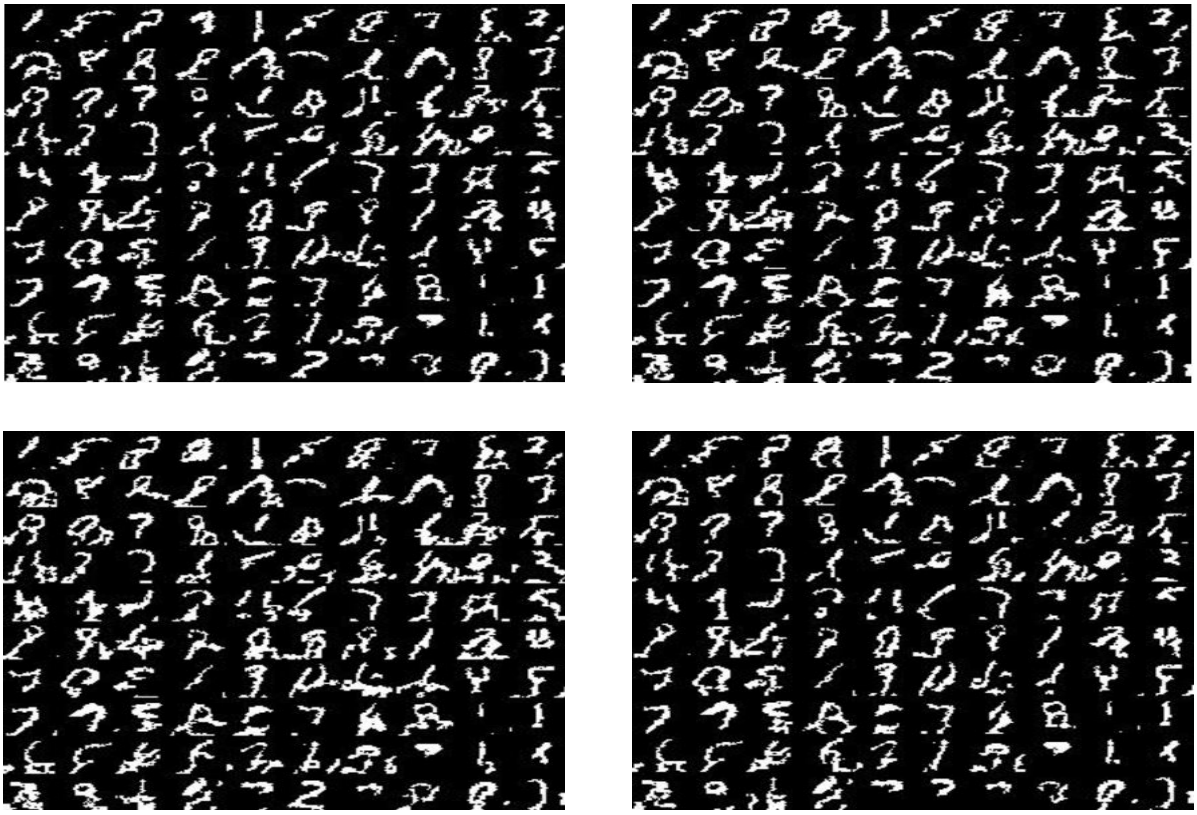


Figure 7: MNIST Image Generation

6 Discussion

In both the quantifiable and observational sense, the recurrent neural network does not perform well in either accurately predicting the sentiment of movie reviews or generating hand-written digits. In terms of the IMDB dataset, the SimpleRNN model achieved a maximum validation accuracy of only 83.6%. The model also struggled with image generation, as an estimated 30% of all images above are illegible. Indeed, the RNN model may be too simplistic to be of real use, as it is incapable of maintaining and learning long-term dependencies. In other words, the vanishing gradient problem may prevent an RNN model from ever becoming truly successful at generating images. Future iterations may experiment with Varational Autoencoders and Generative Adversial Networks in order achieve more accurate results.

7 References

Deep Learning with Python - Francois Chollet - Manning Publications

Using TensorFlow to generate images with PixelRNNs - Phillip KuznetsovNoah and Meyer Golmant - O'Reilly Media