# Deep Learning Homework 2

By: Todd DeLuca

Class: UVM CSYS 395 Deep Learning, Spring 2018, taught by Prof. Safwan Wshah.

This homework explores the training and use of character-to-character sequence models. All the code for the project can be found at https://github.com/todddeluca/uvm_deep_learning_homework2.

## Part 1: Data and Representation

- *Select you training dataset. Keep in mind, first of all, that your dataset has to be big in order to learn the structure, typically RNNs have been trained on highly diverse texts such as novels, Eminem lyrics, programming code, etc. (so be creative here!) Easy option, Gutenberg Books is a source of free books where you may download full novels in a .txt format.*
- *You need to use a character-level representation for this model. Convert the chosen training set to the extended ASCII of 256 characters, read characters one at a time and convert it into a one-hot-encoding. Each character will map to a vector of ones and zeros, where the one indicates which of the characters is present. Your RNN will read in binary vector of length-256.*

### Datasets

The following dataset was used in this project:

- Pride and Prejudice by Jane Austen
    - 4671 sentences, 121567 words.
    - Source: The Gutenberg Project at http://www.gutenberg.org/ebooks/1342
    - Size: 682,302 chars

The dataset was downloaded as described in the notebook 'download'. Nothing fancy here. Just downloading and storing files to a data directory.

### Preprocessing

The extended ASCII encoding, ISO-8859-1, aka ISO Latin 1, was chosen to encode dataset characters as integers in the range [0, 255]. In the preprocessing code, every character in the text of a dataset was converted to a one-hot encoded representation of the ISO-8859-1 integer value of the character.

# Part 2: Training

*Train your recurrent neural network using the dataset you created in Part 1. You are free to choose learning parameters (sequence length, learning rate, stopping criteria, etc.), to make it easy for you, you can use build in function from deep learning toolkits to implement and train RNN architectures. Complete the following task:*

- *Report your training procedure. Plot the training loss vs. # of training epochs.*
- *During training, choose 5 breaking points (for example, you train the network for 100 epochs and you choose the end of epoch 20,40,60,80,100) and show how well your network learns through time. You can do it by feeding in the network a chunk of your training text and show what is the output of the network. Report your result.*

## Training Procedure

The network was trained for 100 epochs on a dataset of 50,000 examples, 25% of which were reserved for validation. After every epoch, performance was evaluated and the model was saved.

The following simple RNN architecture was used for this **baseline model**:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 40, 256)           0
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, 128)               49280
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
=================================================================
Total params: 82,304
Trainable params: 82,304
Non-trainable params: 0
_________________________________________________________________
```

Other hyperparameters:

- Training set size: 50000 examples
- Input sequence length: 40 characters
- Batch size: 128
- Hidden Units: 128
- Optimizer: Adam
- Learning Rate: 0.001
- Learning rate decay: None
- Beta_1: 0.9
- Beta_2: 0.999

- Stopping Criteria: 100 epochs

## Performance

The **baseline model** achieves a training accuracy of 66% and a validation accuracy of 43% after 100 epochs. Validation accuracy peaks around epoch 30 at 46%.

The following table shows performance across epochs. The columns are acc (training accuracy), loss (training loss), val_acc (validation accuracy), and val_loss (validation loss).

| | epoch | acc | loss | val_acc | val_loss |
|---|---|---|---|---|---|
| 0 | 0 | 0.180027 | 3.019693 | 0.23072 | 2.741361 |
| 10 | 10 | 0.457307 | 1.840933 | 0.43320 | 1.939085 |
| 20 | 20 | 0.520133 | 1.597155 | 0.45776 | 1.854289 |
| 30 | 30 | 0.560480 | 1.443088 | 0.45952 | 1.870289 |
| 40 | 40 | 0.598187 | 1.320244 | 0.45264 | 1.939490 |
| 50 | 50 | 0.618453 | 1.240908 | 0.45368 | 1.996428 |
| 60 | 60 | 0.634987 | 1.180191 | 0.44928 | 2.045738 |
| 70 | 70 | 0.644720 | 1.142096 | 0.44272 | 2.108088 |
| 80 | 80 | 0.654640 | 1.108643 | 0.44616 | 2.150113 |
| 90 | 90 | 0.658987 | 1.090908 | 0.44624 | 2.179227 |
| 99 | 99 | 0.660507 | 1.081073 | 0.43216 | 2.226593 |

The following graphs plot accuracy and loss for training and validation datasets:


## Generated Text

As seen in the performance plot above, the training accuracy of the trained network rose slowly over 100 epochs. The validation accuracy plateaued early and never rose much above 45%.

Models from the following epochs were chosen to generate text: 20, 40, 60, 80, 100. These models were initialized with a sequence chosen randomly from the training text and then used to predict the next 40 characters. Each character was randomly sampled according to the probability distribution output by the network and then incorporated into the seed for predicting the next character.

The subjective quality of the generated text improved from epoch 20 to epoch 100.

Results:

Epoch 20:
nd ald; i vore whin choce the rooll the astringsmograiser a verve the vas i not
Epoch 40:
there it manner, hak for here do wack_ mich mest fart falk, a derpeas mighind on
Epoch 60:
be a pleascade" at really megom stours is of their rifuced my sighat. erurebye
Epoch 80:
d for twe say enden, it withe treyetr if the comether of her comirget mism hom;
Epoch 100:
uctionso to intshaby wich would nom hom simins her growe hed soon heved, in tigh

In [ ]:

# Part 3: Experiments

*Experiment with different Network Structures. Discuss your findings if you change these parameters:*

- *Number of hidden units. Try doubling and halving your number of hidden units. plot the training loss vs. the # of training epochs and show your text sampling results.*
- *Sequence length. Try doubling and halving your length of sequence that feeds into the network. plot the training loss vs. the #of training epochs and show your text sampling results.*
- *Replace your RNN with LSTM and GRU. plot the training loss vs. the #of training epochs and show your text sampling results.*

The following experiments were performed:

- Using an LSTM
- Using dropout regulation
- Using shorter sequences (with dropout)
- Using longer sequences (with dropout)
- Using more training data (with shorter sequences and dropout)
- Using 2 RNN layers (with shorter sequences, dropout, and more training data)
- Using a Convolution before the recurrent layer
- Using a Conv layer, 2 GRU layers, shorter sequences and more training data

The following is a summary of the performance of models trained for 100 epochs:

| model | train_acc | train_loss | val_acc | val_loss | params |
|---|---|---|---|---|---|
| baseline | 0.660507 | 1.081073 | 0.43216 | 2.226593 | 82304 |
| lstm | 0.64312 | 1.155526 | 0.50912 | 1.761593 | 230144 |
| dropout | 0.4956533333 | 1.699878918 | 0.4385599999 | 1.954402937 | 82304 |
| dropout_short | 0.4957 | 1.6999 | 0.4386 | 1.9544 | 82304 |
| dropout_long | 0.4897 | 1.7149 | 0.4414 | 1.9543 | 82304 |
| dropout_short_data | 0.4982 | 1.7125 | 0.4896 | 1.7499 | 82304 |
| dropout_2rnn_short_data | 0.562053 | 1.481656 | 0.53208 | 1.618062 | 115200 |
| conv | 0.742933 | 0.811836 | 0.494 | 2.176098 | 106944 |
| conv_2gru_short_data | 0.836587 | 0.488136 | 0.55056 | 2.326647 | 255040 |

The following observations can be made:

- Using an LSTM seems to help performance
- Using more training data seems to help performance and generalization
- Using dropout seems to improve generalization
- Using multiple recurrent layers seems to improve performance

Future experiments:

- Vary the number of hidden units
- Try multi-task learning

## Experiment 1: Using an LSTM

Using an LSTM layer instead of a SimpleRNN layer.

### Architecture and Hyperparameters

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 40, 256)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               197120
_________________________________________________________________
dense_1 (Dense)              (None, 256)               33024
=================================================================
Total params: 230,144
Trainable params: 230,144
Non-trainable params: 0
_________________________________________________________________
```

The changes from the **baseline model** are:

- An LSTM layer was used, not a SimpleRNN

All other hyperparameters and training methods were kept the same.

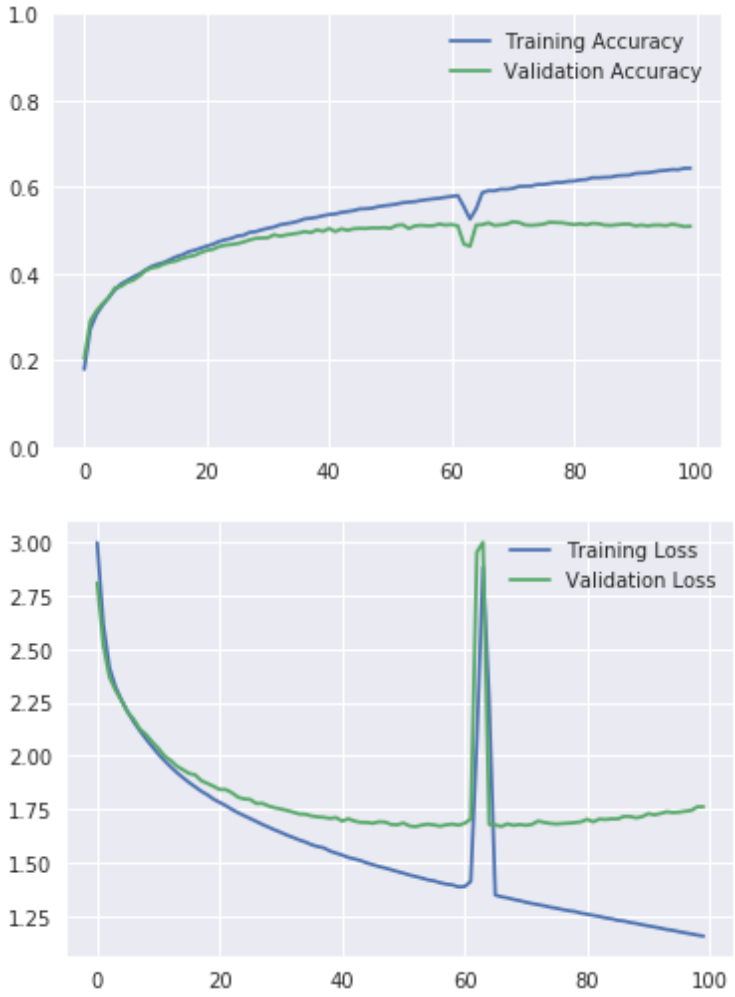
### Performance

The LSTM model achieved a higher validation accuracy at a cost to training accuracy than the **baseline model**.

Performance Table:

| | epoch | acc | loss | val_acc | val_loss |
|---|---|---|---|---|---|
| 0 | 0 | 0.178933 | 2.998935 | 0.20344 | 2.810523 |
| 10 | 10 | 0.408133 | 2.008093 | 0.40656 | 2.035114 |
| 20 | 20 | 0.463333 | 1.781776 | 0.45336 | 1.842651 |
| 30 | 30 | 0.504560 | 1.641285 | 0.48256 | 1.751044 |
| 40 | 40 | 0.536187 | 1.535481 | 0.50408 | 1.694658 |
| 50 | 50 | 0.558267 | 1.450485 | 0.50440 | 1.685249 |
| 60 | 60 | 0.578133 | 1.387810 | 0.51312 | 1.684092 |
| 70 | 70 | 0.596480 | 1.314403 | 0.51952 | 1.674533 |
| 80 | 80 | 0.613493 | 1.258118 | 0.51296 | 1.702674 |
| 90 | 90 | 0.630987 | 1.203318 | 0.51000 | 1.729475 |
| 99 | 99 | 0.643120 | 1.155526 | 0.50912 | 1.761593 |

Performance Graphs:



### Generated Text

Epoch 20:
ld freaod tuch her it hat to soone with had shigr had so thas mone my derk and s
Epoch 40:
 ton the racarter his some in undausioven." 's tholiet. it the fistion only to b
Epoch 60:
hem." socling ould had give to for her, for wand that dintle aty fiss said i ano
Epoch 80:
ssem's riddude, hope a paite it uphavouse, and why dares, whech which you mear,
Epoch 100:
 expresside repolf, _heretions exaute it cainsirned a recomject in grvertions co

## Experiment 2: Using dropout regularization

This experiment modifies the **baseline model** by:

- Using a dropout layer after the RNN layer (using a dropout rate of 0.5)

All other hyperparameters and training methods were the same.

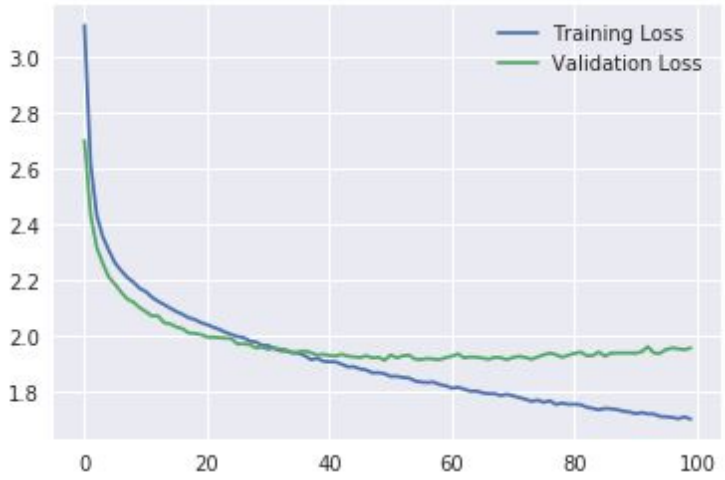
### Architecture

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | (None, 40, 256) | 0 |
| simple_rnn_1 (SimpleRNN) | (None, 128) | 49280 |
| dropout_1 (Dropout) | (None, 128) | 0 |
| dense_1 (Dense) | (None, 256) | 33024 |

Total params: 82,304
Trainable params: 82,304
Non-trainable params: 0

## Performance

Final training accuracy was around 49% and validation accuracy around 44%.  Validation loss and accuracy plateau around epoch 30 and improve only slightly thereafter.


## Generated Text

Generated text improved somewhat from epoch 20 to epoch 100, as seen here:

Epoch 20:
muvery toas buthinn, was rodly s! hourd devirgait ileditt whot and siled rivady
Epoch 40:
ininelon thopondiplo. in the hathes but wiswind, pethennercasima! is nedineirged
Epoch 60:
dound arn ouct wo ctheat har. che bi gloughe; "no thes alk, ant m then ilizabeeh

Epoch 80:
certing, at ono more thimr. "i had werer intele with ingte  hsiver how hopeilt-e
Epoch 100:
tt of popt of insiided it cale te thy gave ams, ould montoffen ther in thet oon.

## Experiment 3: Using Shorter Training Sequences

This experiment modifies the RNN from **experiment 2** by using training sequences of length 20 instead of length 40.  The idea here is that RNNs are not great at using long distance information, so shortening the sequence should not hurt and might improve performance.

### Architecture

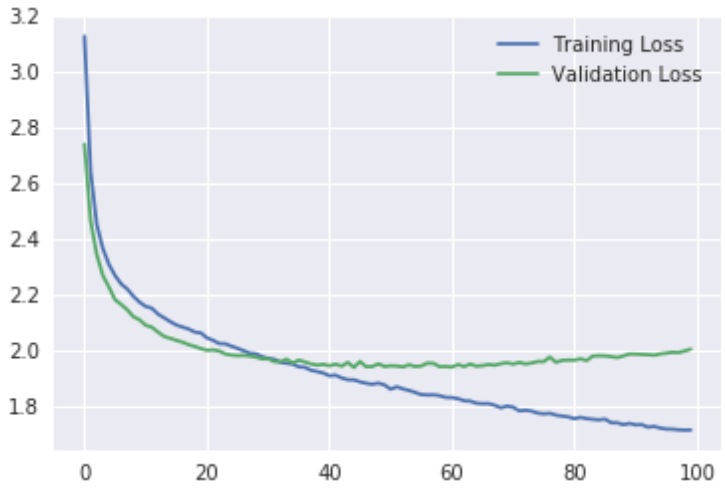
The same as experiment 2.

### Performance

Like experiment 2, validation loss and accuracy plateau around epoch 30 and improve only slightly thereafter.  Epoch 100 performance was:

Training loss: 1.6999
Training accuracy: 0.4957
Validation loss: 1.9544
Validation accuracy: 0.4386

These numbers are very similar to experiment 2.  Using a shorter input sequence length did not improve performance.




### Generated Text

Generated text improved slightly from epoch 20 to epoch 100.

Epoch 20:
passer tha ruppny on hevi io nestwas saks non ny efrjane fowins rxasing lange fi
Epoch 40:
e sode what ht ivery deelligede to beco be ne was beomet he hem prel ads oury mo
Epoch 60:
y all-jaee oo thaok, ad homsste aid miget rojant. s kond prtem!" fam om and,d ab
Epoch 80:
uc liss begpeiss; but i tugho had beingve mastengerssughisery bilkly thee gither
Epoch 100:
ther have nat fing ol yoc limss. aidsmerno gill an surheey ble sunt to hipwhetiv

## Experiment 4: Using Longer Training Sequences

This experiment modifies the RNN from **experiment 2** by using training sequences of length 80 instead of length 40.  Will this improve performance?

### Architecture

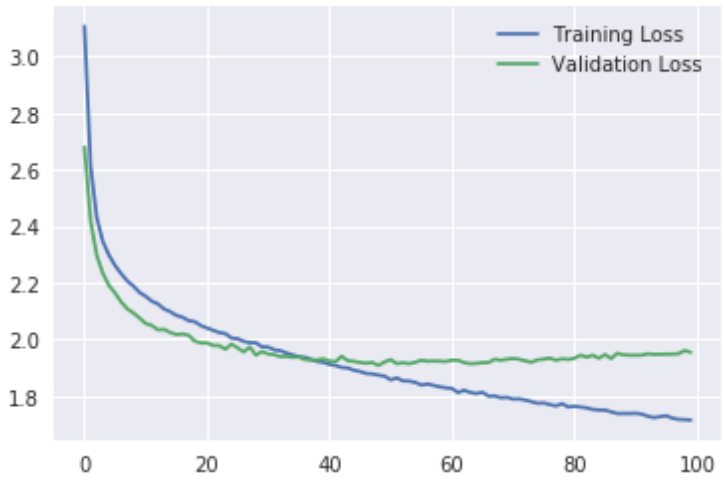
The same as experiment 2.

## Performance

Like experiment 2, validation loss and accuracy starts to plateau around epoch 30 and improves only slightly thereafter.  Epoch 100 performance was:

Training loss: 1.7149
Training accuracy: 0.4897
Validation loss: 1.9543
Validation accuracy: 0.4414

These numbers are very similar to experiment 2.  Using a longer input sequence length did not improve performance, though it did make training much slower.

### Generated Text

Epoch 20:
ad yoh hir the ald if eluhiverto rosenthing mf ros akd oo masuke same gact acret
Epoch 40:
 tuln no negingrseda issess  tell comthe the, byer elyzabeth, wast stave ys once
Epoch 60:
t and reatithe porsof ey hom, her the conladt,ry. erow_ har organd wich athas de
Epoch 80:
licular, with sero tha titurit wan have with. me. bbinded her ofer athing to ere
Epoch 100:
very mand is parstiol wot the exelpnosed the comle wever t osis, mase chereed ti

## Experiment 5: Using More Training Data

This **experiment 2** by using shorter training sequences, like experiment 3, and using twice as many training sequences, 100,000.

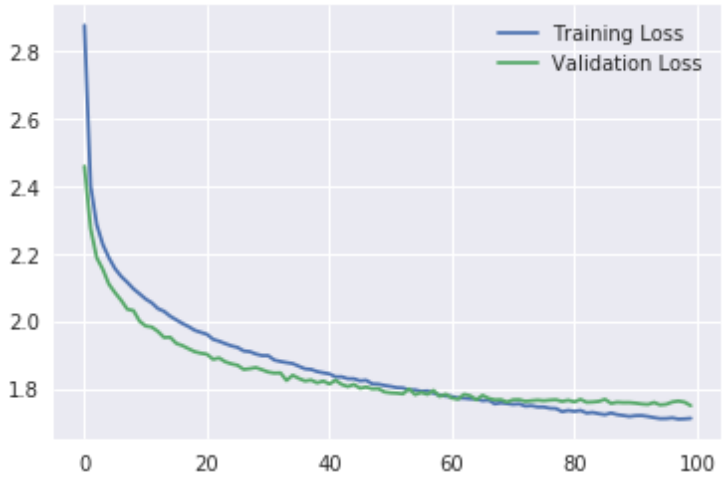
### Architecture

The same as experiment 2.

### Performance

This experiment shows a significant improvement in validation performance over experiment 2 (and 3 and 4).  Training and validation accuracy and loss continue to improve for 100 epochs. More training epochs would likely improve performance even more.  Epoch 100 performance was:

Training loss: 1.7125
Training accuracy: 0.4982
Validation loss: 1.7499
Validation accuracy: 0.4896


## Generated Text

Generated text improved slightly from epoch 20 to epoch 100.

Epoch 20:
den lad lart, endemothel yould be the sere the sse. de cenn-mite aull monessa, p
Epoch 40:
ce, wherher withou the been the buthen say she of her ktow--urceos bo ho wad bon
Epoch 60:
o nos im nsters inmarseece, he adaen not and resveud ald med, "of t ay his assio
Epoch 80:
s's pertaed, both was have ureathered chirccemcther whith my matiog." "she lomel
Epoch 100:
been migsthef it wat in she shes, in the joinadayr." "enthend then are could nut

## Experiment 6: Using more RNN layers

The architecture extends the RNN architecture of experiments 2-5 by adding an additional RNN layer. The shorter sequences (length 20) of experiment 3 are used. The larger amount of training data (100k sequences) of experiment 5 is used. All other parameters are the same as experiment 2.

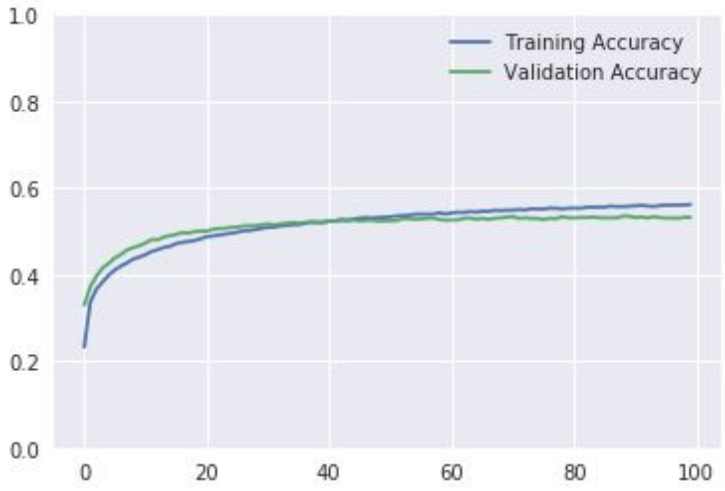
### Architecture

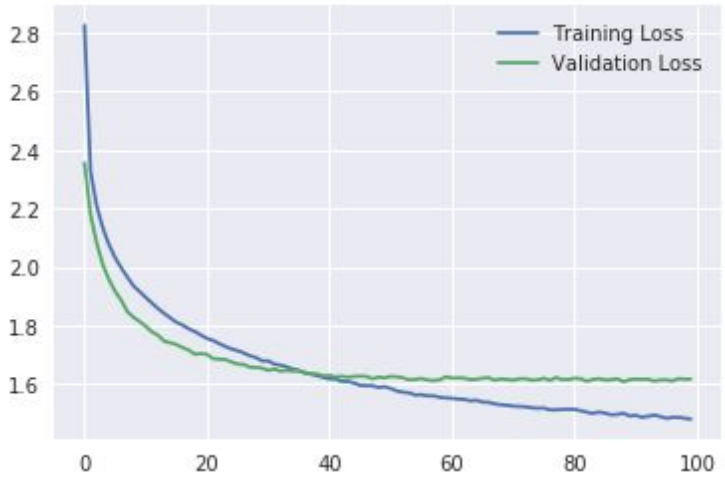
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 20, 256)           0
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 20, 128)           49280
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               33024
=================================================================
Total params: 115,200
Trainable params: 115,200
Non-trainable params: 0
_________________________________________________________________
```

### Performance

The performance of this model is

```
epoch       99.000000
acc          0.562053
loss         1.481656
val_acc      0.532080
val_loss     1.618062
```




## Generated Text

I was struck by how word-like the generated text appeared to be. There is an improvement in word quality from epoch 20 to epoch 100. Epoch 100 even outputs several words in a row: “the the fortune of the move than she”.

Epoch 20:
wele mil-ely mike whoke had bith were that o ever mits livey, and prestexs to b
Epoch 40:
isgrimen to to mo faon him to oustente, as she atpeas of simingd her to lingders
Epoch 60:
rest acifors. "yow was not a pporst bi as carsi at is aokne the everone her own
Epoch 80:
ft quit of ilatadits maeagem sthereve for they follott noty as that of was mianl
Epoch 100:
mrs. der-worte." "i ks wowtikn the the fortune of the move than she habte. anty

## Experiment 7: Convolution before Recurrence

This experiment extends the baseline model by adding a 1D convolution layer between the Input layer and the recurrent layer. The 1D convolution has a filter size of 3, 64 filters and a ReLU activation, so its resulting shape is (m, seq_len - 2, 64).

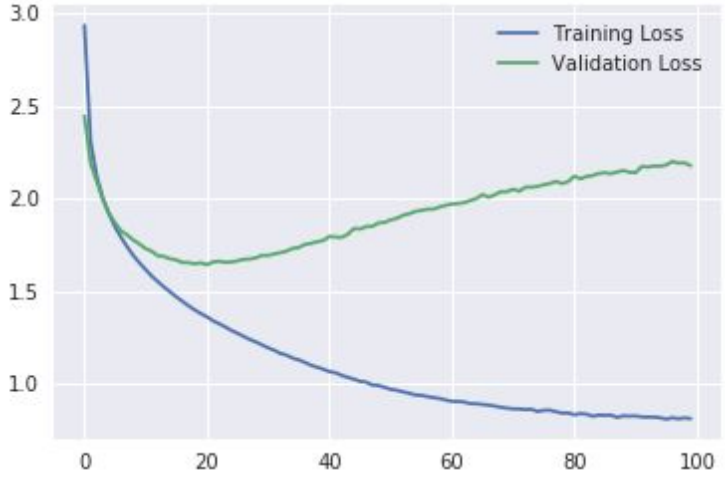
### Architecture

```
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         (None, 40, 256)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 38, 64)            49216
_________________________________________________________________
simple_rnn_3 (SimpleRNN)     (None, 128)               24704
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
=================================================================
Total params: 106,944
Trainable params: 106,944
Non-trainable params: 0
_________________________________________________________________
```

In every other way the hyperparameters and training methods are the same as the baseline model.

### Performance

| | epoch | acc | loss | val_acc | val_loss |
|---|---|---|---|---|---|
| 0 | 0 | 0.215040 | 2.932857 | 0.32392 | 2.443766 |
| 10 | 10 | 0.520107 | 1.615437 | 0.49936 | 1.728377 |
| 20 | 20 | 0.587387 | 1.360980 | 0.52784 | 1.641755 |
| 30 | 30 | 0.633893 | 1.193242 | 0.52136 | 1.691847 |
| 40 | 40 | 0.672427 | 1.065106 | 0.51624 | 1.793999 |
| 50 | 50 | 0.699867 | 0.969641 | 0.50688 | 1.883673 |
| 60 | 60 | 0.717333 | 0.904971 | 0.49952 | 1.969273 |
| 70 | 70 | 0.729333 | 0.864408 | 0.50176 | 2.049192 |
| 80 | 80 | 0.738907 | 0.832751 | 0.49136 | 2.119259 |
| 90 | 90 | 0.736720 | 0.826248 | 0.49840 | 2.138940 |
| 99 | 99 | 0.742933 | 0.811836 | 0.49400 | 2.176098 |


## Generated Text

Epoch 20:
if they it have antertase of mas's, hrraves to be hapted the such the reon of mu
Epoch 40:
t, solving the pister, and the party in mareing, had forpot compored to much sev
Epoch 60:
he sholl herself. with hir sets june of it. i must be ady convers was at oup com
Epoch 80:
what darcy dight tronge, room herselved to easier." he jase ters _bout; it wend
Epoch 100:
 retifnectinnsing to exted the; wille, he hone then to entlied; and its enivanio

## Experiment 8: The Kitchen Sink

This experimental model varies from the baseline model in the following ways:

- A stack of 2 GRU layers is used in place of the recurrent layer
- A Conv1D layer is used after the input layer and before the GRU layers.
- Shorter, 20 char, input sequences are used.
- 150k training samples are used, 3x more than the baseline.

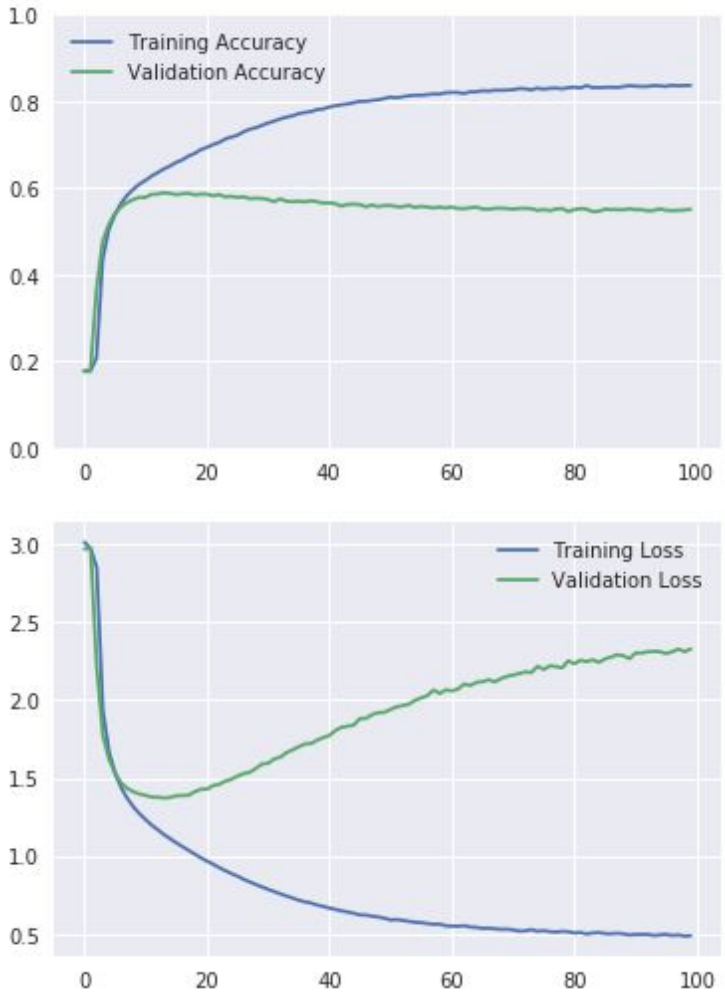
### Architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         (None, 20, 256)           0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 18, 64)            49216
_________________________________________________________________
gru_1 (GRU)                  (None, 18, 128)           74112
_________________________________________________________________
gru_2 (GRU)                  (None, 128)               98688
_________________________________________________________________
dense_2 (Dense)              (None, 256)               33024
=================================================================
Total params: 255,040
Trainable params: 255,040
Non-trainable params: 0
_________________________________________________________________
```

### Performance

| | epoch | acc | loss | val_acc | val_loss |
|---|---|---|---|---|---|
| 0 | 0 | 0.177387 | 3.010902 | 0.176240 | 2.968116 |
| 10 | 10 | 0.617218 | 1.232163 | 0.578160 | 1.388015 |
| 20 | 20 | 0.693476 | 0.966526 | 0.585707 | 1.427865 |
| 30 | 30 | 0.749742 | 0.785069 | 0.573120 | 1.594487 |
| 40 | 40 | 0.786702 | 0.665133 | 0.565627 | 1.771952 |
| 50 | 50 | 0.809796 | 0.588748 | 0.559413 | 1.939893 |
| 60 | 60 | 0.820400 | 0.550093 | 0.555520 | 2.058456 |
| 70 | 70 | 0.826658 | 0.526138 | 0.551013 | 2.160227 |
| 80 | 80 | 0.832756 | 0.505789 | 0.549787 | 2.232697 |

```
90      90  0.834862  0.496111  0.549760  2.301567
99      99  0.836587  0.488136  0.550560  2.326647
```




## Generated Text

Epoch 20:
bingley to an ehe, "you were catched by the arrownecas of your interpby they had
Epoch 40:
it, her come our such earner to me to make might be soon oppordingting, in that
Epoch 60:
mble, that he smokeds more in fanning reading her by impossible that it know. an
Epoch 80:
 was confined with such a mame fewcister which reneys her into, before chaity as
Epoch 100:
say that the laugas. with diffeer in me, i shall have engaged on the cause party

## Experiment 9: The Other Kitchen Sink

After looking more at the results of the previous experiments, I decided to try this other kitchen sink. It includes more convolutions, LSTMs for longer sequences, dropout for regularization, and more training data.

This experimental model varies from the baseline model in the following ways:

- 2 LSTM layers are used in place of the recurrent layer
- 2 Conv1D layers are used after the input layer and before the recurrent layers.
- 100k training samples are used.
- Dropout (rate=0.2) is used after the recurrent layers and before the Dense layer.

### Architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         (None, 40, 256)           0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 38, 64)            49216
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 36, 64)            12352
_________________________________________________________________
lstm_7 (LSTM)                (None, 36, 128)           98816
_________________________________________________________________
lstm_8 (LSTM)                (None, 128)               131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               33024
=================================================================
Total params: 324,992
Trainable params: 324,992
Non-trainable params: 0
_________________________________________________________________
```

## Performance

Pending...

## Generated Text

Pending...