WHAT'S THE PROBLEM WITH DEEP LEARNING WITH BIG AND SMALL DATA AND HOW IT CAN BE SOLVED

INDERPREET S. GANDA

Project Overview:

To investigate why Deep learning needs big data, what are the problems when we have small data and how it can be solved – The goal of this project was to determine why Deep learning needs big data, and why even with big data there is problem called Adverisal examples[1] and [2], even when there is big data we can still fool Deep neural networks with slight perturbations in the data.So in a nutshell to explore why there is problem with big and small data and what are possible ways of solving this problem.We are going to study this problem by applying Deep neural networks to Natural language processing task called Sentiment analysis. The architectures we use for this approach are discussed below.

Problem Statement:

We wanted to see why deep learning needs big data and what are problems with small data which makes deep learning ineffective. This work is a combination of both theoretical and practical work.We will use two deep networks, to solve this problem ,first one we will use is Convolution layers followed by a LSTM layer, followed by fully connected layer and then softmax layer to make predictions. The other network we use is fully convolutional neural network, it has four convolutional layers followed by one more convolutional layer for classification and then softmax for prediction. The reason behind these types of architectures is that language has lot of variance and there are many ways to express the same meaning with different set of words and by different set of word sequences. In the following sections we will discuss in more detail how these architectures attempt to best capture the variance in the language.We attempt to achieve high accuracy in this Sentiment classification task using less data. We did 8 way sentiment classification on large IMDB dataset(which has 50,000 movie reviews) and 8 labels (7,8, 9, 10) for positive reviews and (1, 2, 3, 4) for negative reciews.And the entire project is implemented using Google's Tensorflow[5] running on Amazon's Elastic compute.

Metrics:

The loss function we used for this project is cross-entropy loss which is defined as follows:

$$H(X) = -\sum p(x)\mathbf{log}q(x)$$

where p(x) is the true probability and q(x) is our estimate.
We choose cross entropy loss function because it is robust and it only gets minimized when when we are predicting each correct label with high accuracy and this does give very high emphasis on incorrect examples as other loss fuctions for instance Mean squared error etc.

The metric we used for this project to measure the performance of our model is accuracy which is defined as follows:

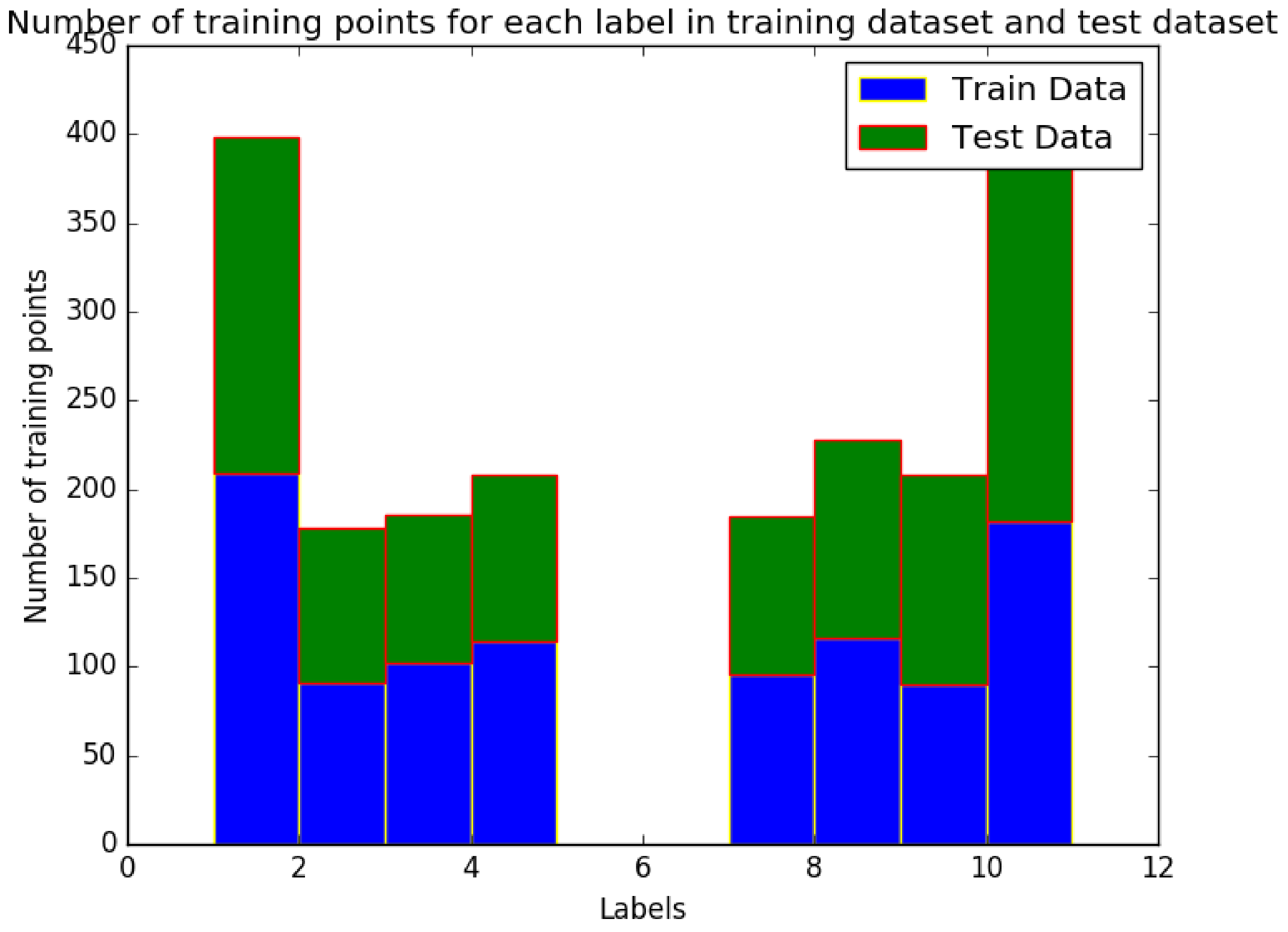
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

We used accuracy as metric to measure the performance of our model because it gives good measure of the performance of the model and it is standard in many machine learning and deep learning applications. And we are using the top-most prediction made by our model and comparing it to the correct label i.e. top-1 prediction, to measure the performance of our models.

Data Exploration:

The dataset we use for this problem is Large IMDB dataset. This dataset consists of 50,000 sentences and 8 labels, which are movie ratings given by users (1,2,3,4) for low ratings and (7,8,9,10) for high ratings.It has a total of 50,000 sample points, of which 25,000 are training points and 25,000 test points. The longest review is 2,445 tokens long and shortest is only 6 tokens. The average lenth of the review is 268 tokens long.

Exploratory Visualization:



Data visulization of 1000 reviews vs 8 labels for both training and test datasets and we can see that in this 1000 sample data set there is almost uniform distribution among all the eight labels for both training and test datasets except for labels 1 and 10(which are more in number in the dataset).

Algorithms and techniques:

We have used two deep neural nets for sentiment classifcation on this daatset. One fully Convolution neural net and other is Convolution connected to long short memeory networks which are then connected to fully connected layer. Both of these architectures are discussed below:

1.) **Convolutional Neural net:**

Convolutional neural networks have become standard for visual recognition tasks.Recurrent neural networks , particularly Lstms and GRUs have become quite popular in wide variety of NLP tasks,because of their ability to capture long term dependencies.

Typical Convolutional neural network is a deep neural network consisting of convolution layers which share weights spatially but not horizontally, max-pooling layers which squash the input keeping only the maximum value ,relu activation unit and then at the end fully connected layer and then softmax layer. Convolution neural networks are standard choice for visual recognition tasks like object detection, object segmentation and many more.For more information, refer to [8].

2.)**Recurrent neural network –** There are another popular types of deep neural nets called Recurrent neural nets which capture long term dependecies. Recurrent neural nets are described below:

$$
\begin{aligned}
&\mathbf{z}^t = g(\mathbf{W}_z\mathbf{X}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z)\\
&\mathbf{i}^t = \sigma(\mathbf{W}_i\mathbf{X}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i)\\
&\mathbf{f}^t = \sigma(\mathbf{W}_f\mathbf{X}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f)\\
&\mathbf{c}^t = \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1}\\
&\mathbf{o}^t = \sigma(\mathbf{W}_o\mathbf{X}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o)\\
&\mathbf{y}^t = \mathbf{o}^t \odot h(\mathbf{c}^t)
\end{aligned}
$$

They take input vector at every time step and they also take input from the previous time step in order to model long-term dependecy. They work very well on NLP tasks like speech recognition, sentiment classification and thay also have been used in attention models i.e. how much attention you need to pay to each input vector from previous hidden state. But they have a problem called vanishing/exploding gradient problem, i.e. when they model long term-term dependencies, their gradients either become very large or they become very very small. To solve this problem, new kind of RNNs were invented called LSTMs which we will use in this project. LSTMs work by using gates at every input, gates allow how much input is to be passed, how much output should be passed, basically these gates control how much data should be allowed to pass to the next unit. For further information on RNNs refer to [7].

Benchmark:

The benchmark we use for this sentiment classification task is by Wang and Manning[4], in which they evaluated number of SVM and Naive bayes classifiers and on variety of different datasets. They got 91.22% accuracy on Large IMDB dataset(which we use for this project). They reported 91.22% accuracy on binary sentiment classification but in this project we did 8-way sentiment classification. We haven't found any research, which did 8-way sentiment classification task and by using only 1000 training examples dataset. So, we will use score by Wang and Manning to compare the performance of our models.

Methodology:

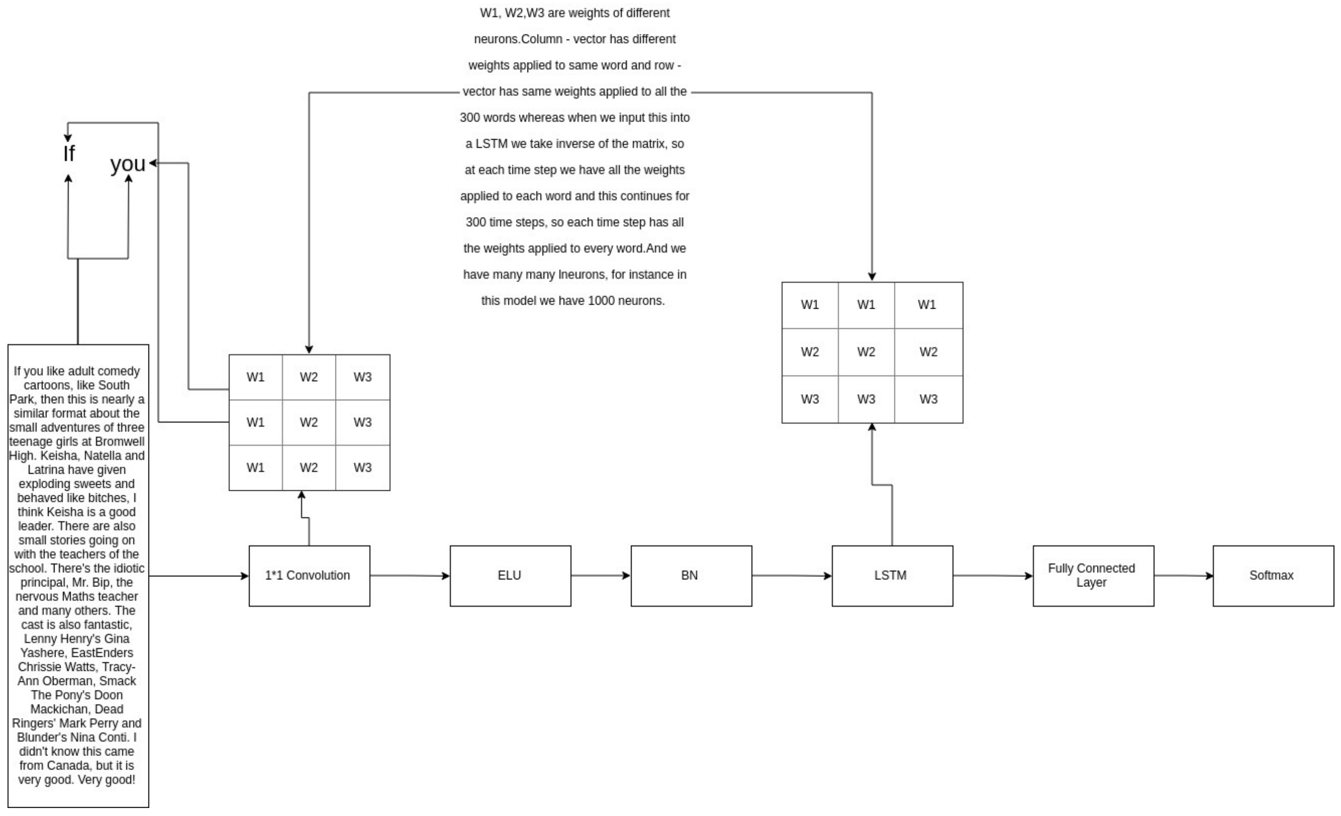
Data preprocessing:

Because we are using mini-batch so all the reviews have length of 300 tokens and for reviews shorter than 300 are padded with the keyword “pad” and reviews larger than 300 are cut down to 300. We used 300 length of reviews because average length of review is 268 and we want to capture more information about reviews which are longer than 268, so that's why we used 300 length. We used Pre-trained Glove Vectors[3] 50-dimensions each for each word trained on Wikiipedia and Gigaword 5 dataset and tokenization of reviews was performed using python NLTK library.We will feed review to our model and it has to classify it into 1 of 8 classes. If any word is not present in glove representation, then it is randomly sampled from the Gaussian distribution, with 0 mean and 1 standard deviation.

Implementation:

We implement two different types of deep neural nets, both of which are discussed as follows:

1.) **Convolutional neural network with LSTM**

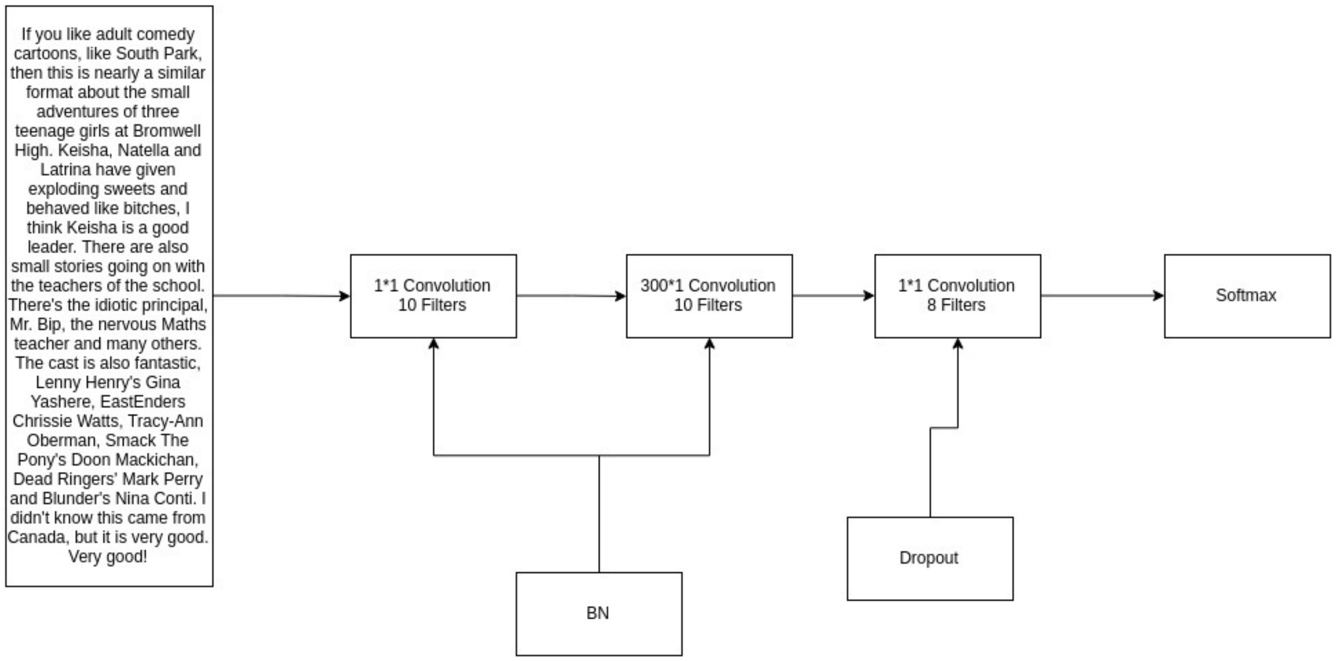


W1, W2,W3 are weights of different neurons.Column - vector has different weights applied to same word and row - vector has same weights applied to all the 300 words whereas when we input this into a LSTM we take inverse of the matrix, so at each time step we have all the weights applied to each word and this continues for 300 time steps, so each time step has all the weights applied to every word.And we have many many lneurons, for instance in this model we have 1000 neurons.

If you like adult comedy cartoons, like South Park, then this is nearly a similar format about the small adventures of three teenage girls at Bromwell High. Keisha, Natella and Latrina have given exploding sweets and behaved like bitches, I think Keisha is a good leader. There are also small stories going on with the teachers of the school. There's the idiotic principal, Mr. Bip, the nervous Maths teacher and many others. The cast is also fantastic, Lenny Henry's Gina Yashere, EastEnders Chrissie Watts, Tracy-Ann Oberman, Smack The Pony's Doon Mackichan, Dead Ringers' Mark Perry and Blunder's Nina Conti. I didn't know this came from Canada, but it is very good. Very good!

As we can see,In this model we have one convolution layer and then a LSTM and then a softmax. Convolution layer opertate on every word up to 300 words. We have 1000 convolutional filters each having different weights.Convolutional layer is then connected to Exponential Linear unit and then we have Batch Normalization layer to ensure that gradients are not zero and error is backpropagating fully and weights are updated as expected.Dropout is not used. Then we have a LSTM which is then connected to fully connected layer which is connected to softmax to make predictions.And this model is trained using Adam optimizer.

2.) **Fully convolutional network**



If you like adult comedy cartoons, like South Park, then this is nearly a similar format about the small adventures of three teenage girls at Bromwell High. Keisha, Natella and Latrina have given exploding sweets and behaved like bitches, I think Keisha is a good leader. There are also small stories going on with the teachers of the school. There's the idiotic principal, Mr. Bip, the nervous Maths teacher and many others. The cast is also fantastic, Lenny Henry's Gina Yashere, EastEnders Chrissie Watts, Tracy-Ann Oberman, Smack The Pony's Doon Mackichan, Dead Ringers' Mark Perry and Blunder's Nina Conti. I didn't know this came from Canada, but it is very good. Very good!
1*1 Convolution 10 Filters
300*1 Convolution 10 Filters
1*1 Convolution 8 Filters
Softmax
BN
Dropout

As we can, this model is fully convolutional neural network. We have first have convolutional layer operating on every word up to 300 words. Then we have another convolutional layer which is operating on full sentence and then at last we have classification convolutional layer and then a softmax to make predctions. And after every convolutional layer we have implemented Batch normalization to ensure that training goes well and dropout for regularization is used.And this model is trained using RMSProp optimizer.

Refinement:

We've implemented two differeent deep neural networks and choosen hyperparameters settings for both of them are as follows:
After testing models many times, so that they don't overfit and generalize well on unseen data hyperparamters of both the models were adjusted. Below are final hyperparameter settings for both the models:

Hyperparameter settings for model 1.

Batch size	100
Learning rate and Decay	1e-4 and 90% after every 3 epochs
LSTM Units	384
Convolutional Units	1000
Fully Connected Units	8

Hyperparameter settings for model 2.

Batch Size	100
Learning Rate and Decay	1e-4 and 96% after every 5 epochs
Dropout Probability	94.5%
Convolutional Units	10 + 10 + 8 = 28

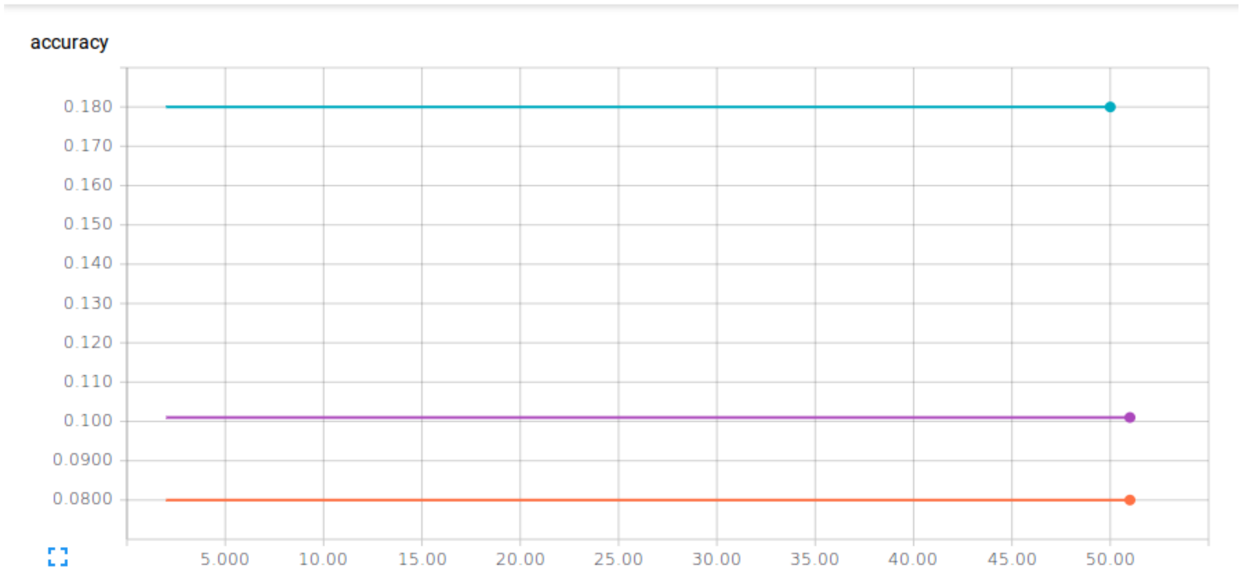
Model Evaluation:

Both of these models suffer from problems. Before going into the detail about these problems. We will list results that we got for both the models. Both the models were trained for 50 epochs.

Results for Model 1:

Training Error	5.18
Training Accuracy	10.1%
Validation Error	5.52
Validation Accuracy	8%
Test Error	4.91
Test Accuracy	18%

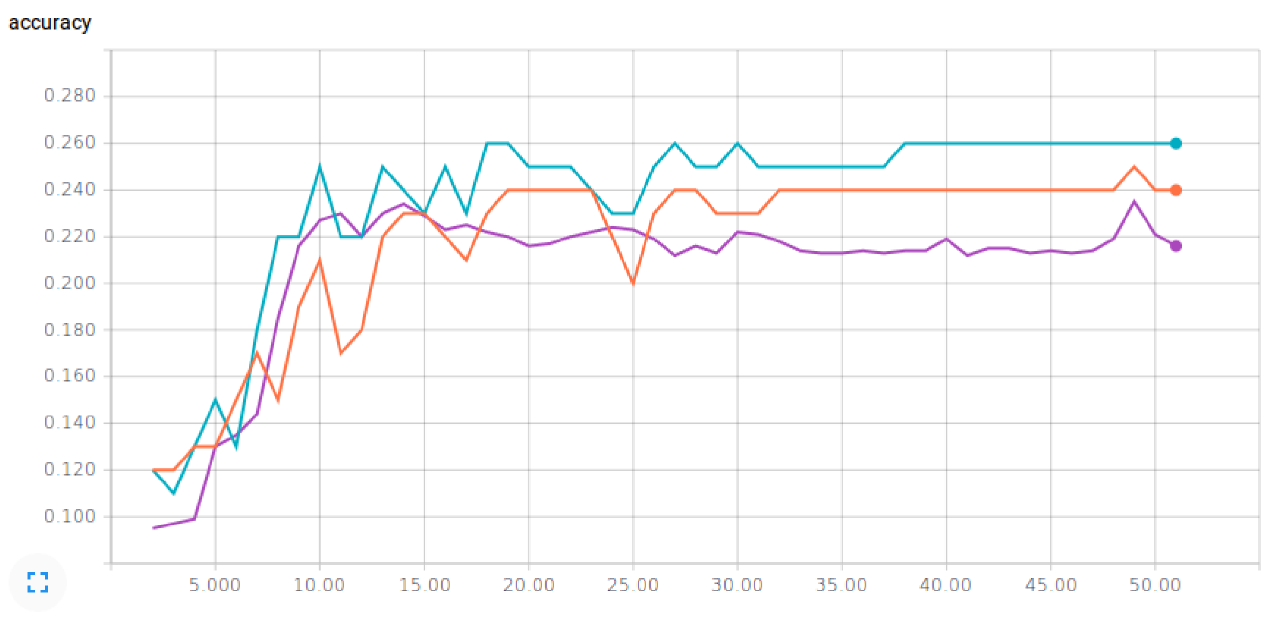
Graphical visualization of Results of Model 1:



Results for Model 2:

Training Error	1.90
Training Accuracy	21.6%
Validation Error	2.03
Validation Accuracy	24%
Test Error	2.12
Test Accuracy	26%

Graphical visualization of Results of Model 2:



As from the results, we can see that Model-1 didn't performed well,inspiration behind model-1 and problems with Model-1 are discussed below. Model-2 performed well compared to Model-1 and produced good accuracy by learning from only 1000 training examples on a high variance NLP data set with 8 labels, but this architecture also suffers from problems which are discussed below.

Note(Model 2) – Due to low computational budget and high computational cost of this task and it has several hyperparameters to be tuned, we couldn't scale this model to be trained on entire dataset and compare it's performance to, when trained on 1000 training points.

Inspiration behind 1st Model – The inspiration behind this model is to capture all the variance in the data. As language has lot of variance i.e. There are many ways to express the same meaning by using different set of words and by putting same words at different places. And everyone has different way of representing the meaning of same thing. So to capture this huge amount of variance has led to the design of this model. Now back to the question of why deep learning needs more data and even if we have more data there still problems with deep learning e.g. adversarial examples. To give little bit of background on adversarial examples, Anh Nguyen, Jason Yosinski and Jeff Clune took the famous Alex net trained on 1.3 million images on Imagenet and added small pertubations in the images and Alex net failed to recognize the images, for instance by making small changes in the image of the dog, when we feed this image to the network it thinks it is goldfish not a dog.

Now in detail why this happens, let's say we have input X and weights W, these weights are trained on the input X, and if during test time our input is slightly changed our network will make different prediction. Because to make prediction our network does a linear multiplixation of inputs X and weights W for e.g Y = W*X + B if our input gets slightly changed it will make a different prediction.

This is what happens in the famous “Deep Neural Networks are Easily Fooled” paper. That's the reason we need big data, if we have big data our network's weights get adjusted according to the big data, so new data is almost always similar to data our model is trained on so that's why it makes very few errors when we have big data and so it's harder to get good predictions from a deep neural network trained on small data.

And when we have NLP task like sentiment classification, it becomes even more difficult to train a good model that generalizes well on unseen data. Because language has lot of variance, many ways to express a smilar meaning, many combinations of different words can be used, so to have a good model that generalizes we need a very big dataset. So to capture the variance of language we designed this model.

How this model tries to capture variance in the language?
As we have already talked about this model has convolutional layer and then a LSTM followed by Fully connected layer and then a softmax to make a prediction. Convolutional network operates on every single word in the review, words itself in the review cannot be chnaged, but we can change the order of the words but we cannot put perturbations in the words itself, and then the output of the filters of convolutional layer gets multiplied by single layer of LSTM, we know LSTM share weights and every word gets multiplied by constant value in the LSTM so it is invariant to the variance in the data and then output of the LSTM is fed to the classification layer which is fully connected it is not invariant to the variance in the data but it has only 8 neurons so it has not much control over the results of the model, as the model will get trained FC weights will be adjusted, so in a way they will behave like a LSTM weights. So in this way, this model tries to be invariant to the variance in the data.

Why this model didn't perform as expected?
1st Model – In this model, we have convolutional layer connected to a LSTM. There is problem with model, it is hard to train both convolutional and LSTM connected network for this dataset. We have even tried to train this network using only 2 and 3 examples, it is unable to overfit 2 examples and we have also increased number of filters and number of LSTM units but network wasn't able to fit 2 examples perfectly. But we still trained the network on more examples and we got the results as above. But if we remove LSTM layer or convolutional layer the model works fine.

2nd Model – The second model we implemented is fully convolutional network. It has three convolutional layers and then a softmax to make prediction. This model trains well but it suffers from same problem as every other deep network. We have first layer which is invariant to the variance in the data as before but second convolutional layer which operates on output of first convolutional layer and output of first convolutional layer can change corresponding to the word in the senetence and can result in second convolutional layer to generate false output, so it is not invariant to the variance in the data. We trained this model just to see how well this performs on this dataset using very few examples i.e. 1000 in this case.

Results:

As we saw, we did not get the results as we expected. Our models didn't performed well as compared to the results of Wang and Manning. As we saw how our models performed on this particular dataset using very examples and we also saw what's the problem with deep learning.We will discuss the possible solutions in the next section. Deep learning is only effective when we have big data and when we have dataset which has high variance we must have millions of examples for our model to generalize well on unseen data. When we have data with high variance, other machine learning methods like SVM and Naive bayes can be also useful as shown by Manning and Wang. Possible solutions are unsupervised learning and most important, which we believe can solve the problem is Deep Rreinforcement Learning.This work started out as supervised learning problem and how to design models in a way that can capture variance in the dataset and generalize well on unseen data but ended up combination of theoretical and practical work which discussed problems of deep learning and what are possible solutions to this problem which are discussed in the next section in detail. Attention models and Deep LSTM models also suffer from the same problem, although it's a bit more complicated to show that.

Conclusion and future work:

We have seen that what are the problems with deep learning when we have small data and when we have big data. In this last section we will talk about the possible solutions. First one is unsupervised learning, we first train the network in an unsupervised way, we can train the network to represent the every sentence by a fixed set of features, in our case we have 300 words, we can train the network to represent every senetence by fixed set of 300 features and then those features can be fed into supervised deep neural network for sentiment classification and this is one way we can capture variance in the data. The other way is by using reinforcement learning[6] which is more important and together with deep learning it becomes deep reinforcement learning. In reinforcement learning we have notion of reward,action and state. The biggest benefit of reinforcement learning is that we can encode domain knowledge into the system. We can give reinforcement learning algorithm an environment, in which it can act on a sentence and learn how traverse a sentence and then make prediction about it. And depending how it learns to traverse the sentence we can further improvements in the algorithm and this technique can be generalized to data that it has not seen, very well. We can also give reinforcement learning algorithm a game environment in which words in a sentence can be encoded as actions in the game and it get rewards based on actions and underlying state. We can also train above discussed network(convolutional network with LSTM) seprately, first we can train Convolutional layer and then we can keep the weights of the convolutional layer fixed and train then LSTM on top of that.

Acknowledgements:

We would like to thank Nando de Freitas,Andrej Karapathy,Fei-Fei Li,Justin Johnson and Richard Socher for putting excellent video lectures and course materials online, because without these materials it would have been very hard for us to do this Deep Learning project.

References:

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever. (2014) Intriguing properties of neural networks.
https://arxiv.org/pdf/1312.6199v4.pdf
[2] Anh Nguyen, Jason Yosinski, Jeff Clune. (2015) Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. https://arxiv.org/pdf/1412.1897v4.pdf
[3] Jeffrey Pennington, Richard Socher, Christopher D. Manning (2014) GloVe: Global Vectors for Word Representation.
http://nlp.stanford.edu/pubs/glove.pdf
[4] Sida Wang, Christopher D. Manning. (2012) Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. http://www.aclweb.org/anthology/P12-2018
[5] Abadi et al. (2015) TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. http://download.tensorflow.org/paper/whitepaper2015.pdf
[6] Volodymyr Mnih Koray Kavukcuoglu David Silver Alex Graves Ioannis Antonoglou Daan Wierstra Martin Riedmiller. (2015) Playing Atari with Deep Reinforcement Learning.
https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf
[7] Hochreiter, S. , Schmidhuber, J. (1997) Long Short-Term Memory.
http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf
[8]Alex Krizhevsky,Ilya Sutskever,Geoffrey E. Hinton. (2012). ImageNet Classification with Deep Convolutional Neural Networks. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf