Theme classification for texts using Naive Bayes and Neural Networks

Albin Larsson Forsberg
Department of Electrical Engineering
Stanford University
albinfor@stanford.edu

Rémy Zawislak
Department of Aeronautics and Astronautics
Stanford University
remzawi@stanford.edu

***Abstract*—The motivation behind this project is to find a way of classifying and summarizing the theme of a text. We do so by building a model that learns from labelled quotes to detect the theme of a sentence or group of sentences. In this paper, we focus on the classification of a few number of sentences at a time. It could then be extended to full text classification. Three models are tested and compared: Multinomial Naive Bayes, Dense Neural Network and Convolutional Neural Network. The Convolutional Neural Network outperforms the other two models and achieve good accuracy on both the training and test set, providing encouraging results in the matter of theme classification.**

I. Introduction

Being able to understand and convey feelings, emotions, and subtle messages is something that we as humans are really good at. We are emotional creatures that don't only feel emotions, we also base many of our decisions on what feels right. A problem then appears in the digital age where we rely increasingly more on computers in our daily life. Computers are in almost everything that we use today, and the day where the need of having them seamlessly integrate with our life is probably not that far into the future. Human computer interaction is thus going to become more frequent and deep, but computers can not communicate like humans do. A step in the right direction is for the computer to be able to pick up and distinguish themes in sentences to be able to understand the deeper message and context in which a sentence is used in. As such, training algorithms to capture the theme of a sentence is of great interest. In this paper, we focus on three different models to tackle this problem: a **Naive Bayes classifier(NB)**, a **Dense Neural Network (DNN)**, and a **Convolutional Neural Network (CNN)**. We focus on sentence classification to study independently sentences of a text. All the implementations can be found in the following github repository: https://github.com/remzawi/CS229-theme-classification.

II. Related work

In [3] the authors investigate how to classify different document types. The approach they used were about using a stacked version of Deep learning networks that they chose to call Hierarchical Deep Learning for Text.

[4] chose to take a divide and conquer approach where they first where they classify the texts before they do sentiment analysis. For the actual sentiment analysis they used a one dimensional (CNN). They achieved state-of-the-art results on benchmark datasets.

A group that tried to solve a similar problem to ours is [5]. They used a Recurrent Neural Network (RNN) model with Long Term Short Term memory. They achieved good results and significantly outperforms other more traditional methods.

[2] comes up with a method where they manage to achieve a per document normalization for the features to improve the performance of NB on classifying texts.

[1] suggests methods that could be applied to NB to correct for the assumptions that the model is based on and make it useful and comparable in performance to State-of-the-art classifiers.

The method suggested in [5] and the adjustments done in [1] are sophisticated and more complex than the models implemented in this paper. They are more complicated to implement than the ones in this paper, but perhaps they would perform better on the problem at hand.

Finally, in [11] the author tries three different structures of neural networks to perform a similar task and compare their effectiveness: than the one studied in this paper: a CNN, a RNN and a Hierarchical Attention Network (HAN). The author observes that even though in some cases the HAN can outperform the CNN, the CNN provides consistently good accuracy which makes it a good candidate for sentence classification.

III. Dataset and Features

The dataset that was acquired for this project came from [6]. It is consisting of 76 000 data points containing text, author and sentiment labels. In total the dataset has 117 different labels An example set of how the data is available is seen in table I

TABLE I
Example of three data points from the dataset

Text	Author	Label
We shape our buildings thereafter they shape us	Winston Churchill	art
Every business and every product has risks. You can't get around it	Lee Iacocca	work
Within 10 years it will be impossible to travel to the North Pole by dog team. There will be too much open water	Will Steger	nature

However we noticed as the project went on, that the quality of the dataset was lacking. There was a lot of overlap between different classes. We therefore also used a subset of the data consisting of 37 413 data points and 16 labels. The more specific labels were merged into more general ones to make it easier for the classifier to be trained by providing more data per class, and also to make the classes more separable. A first reduction on the number of labels was based on human observation and knowledge, and a second one on the analysis of the confusion matrix to determine conflicting classes or classes hard to classify.An example of two classes that are Religion, Faith, and God, into a common class called Religion. Reducing the specificity creates a more simple dataset. The author label wasn't used in this paper and was as such discarded.

To generate the features from the dataset two different preprocessing approaches were needed. One approach was needed for the NB and DNN models, and another one for the CNN.

A. Naive Bayes and Dense Neural Network

The NB and DNN were fed with a feature vector containing the frequency of words in a dictionary. The dictionary was created by first stemming the words using the Natural Language Toolkit (or NLTK) package [7]. This reduces the words to their basic forms and makes words such as run, runs, and running, appear as the same word. This effectively reduces the size of possible candidates to the dictionary. The frequency of the stemmed words was recorded and if the frequency was high enough it got added to the dictionary. A stop list was also used to avoid many common words that does not add much value to the classification problem. The threshold was set to 150 requiring the frequency of words to be relatively high for it to be added to the dictionary. It was arbitrarily chosen, but empirically showed to be a good trade-off between removing insignificant words while keeping enough information to actually perform classification.

B. Convolutional Neural Network

The CNN was fed with a feature vector containing positional information instead of quantitative. Every word was first converted to lower case and then into a number representing the index that word is in the dictionary. The feature vector was padded to make sure all feature vectors were the same length. Word embedding was used for the CNN with pretrained features from [10] (the 6B tokens set with 200 features was used), with a randomly initialized embedding for unknown words.

IV. Methods

A. Naive Bayes

NB is a classification method that works well with text based problems. It works with the naive assumption that the probability that one feature is true is independent of other features. This lets us do a simplification that greatly simplifies the calculations.

$$p(C_k|x_1,...,x_n) = p(x_1|x_2,...,x_n,C_k)...p(x_n|C_k)p(C_k)$$
$$p(x_i|x_{i+1},...,x_n,C_k) = p(x_i|C_k)$$
$$p(C_k|x_1,...,x_n) = p(C_k)\prod_{i=1}^{n} p(x_i|C_k)$$

The NB classifier was chosen as a more simple method that we could use as a baseline and compare the other methods to. The implementation of the NB is the multivariate version. The design parameter θ_k is calculated by $\theta_k = \frac{x_i+\alpha}{N+\alpha d}$.

N is the total number of test cases and d is the total number of features. The parameter α can be tuned to change the sensitivity to previously not seen words. Each class will have one parameter trained for itself and the numbers can then be fed into a softmax function to calculate the most probable class.

$$\textit{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

NB works by finding the probability that a word indicates that a sentence belongs to a class and then selects whichever class has the highest probability in the end. Empirical experiments seemed to demonstrate that a value of 10 for α produced great results. The classifier was implemented with the library sci-kit learn [9].

B. Dense Neural Network

The Dense Neural Network, also known as Feedforward Fully-Connected Network, is a feedforward network in which each neuron from a layer is connected to all the neurons from the following layer. The formula for the forward propagation in each node is the following.

$$a^{[j]} = f\left(\sum_i w_i a^{[j-1]} + b\right)$$

The activation function in each of the layers before the output layer is the ReLU (Rectified Linear Unit) function, while the output layer has the softmax function as activation function.

$$\textit{ReLU}(x) = \begin{cases} x & \textit{if} \;\; x > 0 \\ 0 & \textit{otherwise} \end{cases}$$

The DNN consists of three hidden layers with 96, 64 and 32 neurons respectively. After each layer BatchNormalization [12] is done to improve the speed of training, performance and accuracy of the network.

The Adam optimizer [13] was used for training. It has proven in the literature to offer good convergence performance on a number of different tasks. The DNN was implemented with Keras [8] with Tensorflow [15] backend.

The structure of the DNN is given in figure 1

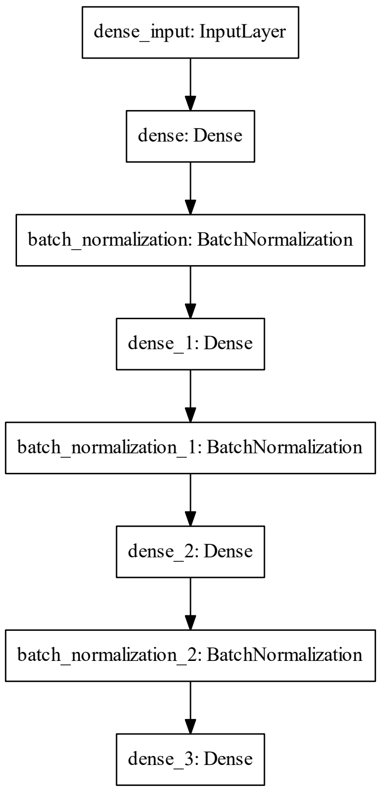
dense_input: InputLayer
dense: Dense
batch_normalization: BatchNormalization
dense_1: Dense
batch_normalization_1: BatchNormalization
dense_2: Dense
batch_normalization_2: BatchNormalization
dense_3: Dense

Fig. 1. DNN model

C. Convolutional Neural Network

The CNN consists of a trainable embedding layer followed by three convolution and pooling layers, and two dense networks for classification at the end. The architecture was inspired by [11]. Batch normalization was performed after each convolution layer. DropOut [14] was used between the dense layers to limit overfitting. For the CNN, the RMSprop optimizer [16] was used, as instabilities seemed to arise with Adam. Padding was used in the convolution layers to preserve information on the border which is important for sentence classification (we noticed a noticeable improvement with padding).

CNN works similarly to DNN as they are both neural networks. However, unlike our implementation of the DNN, the CNN can use the position of words in the sentence, so we can expect better result with this architecture. The CNN was also implemented with Keras.

The structure of the CNN is given in figure 2

V. Results

The models were trained, tested and on the dataset with an 80/10/10 split. Training and evaluating on the original dataset did not perform as well as intended with accuracy under 60%. We assumed this was due to the large number of

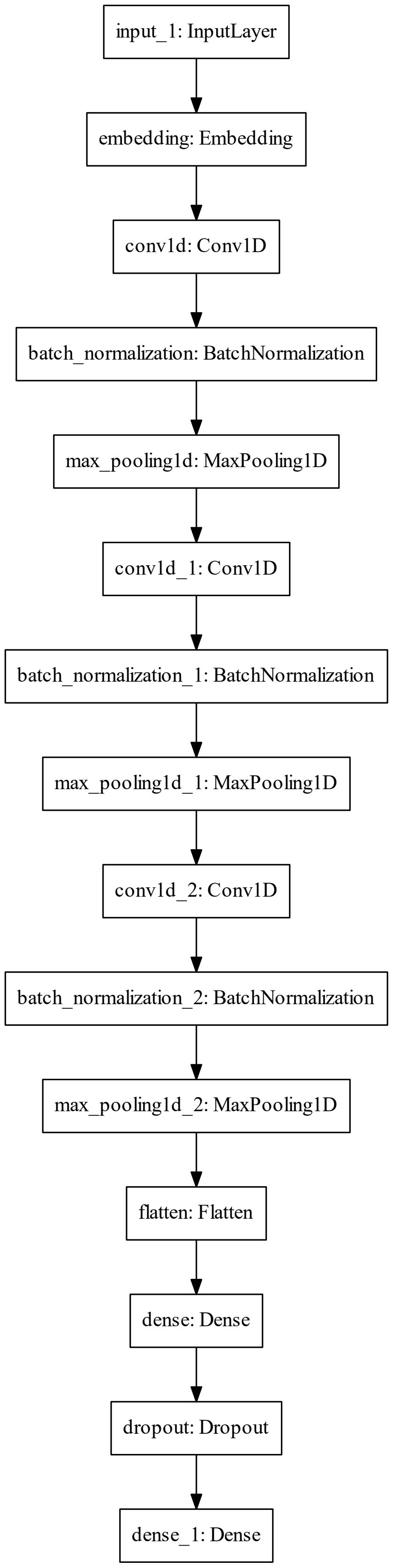
input_1: InputLayer
embedding: Embedding
conv1d: Conv1D
batch_normalization: BatchNormalization
max_pooling1d: MaxPooling1D
conv1d_1: Conv1D
batch_normalization_1: BatchNormalization
max_pooling1d_1: MaxPooling1D
conv1d_2: Conv1D
batch_normalization_2: BatchNormalization
max_pooling1d_2: MaxPooling1D
flatten: Flatten
dense: Dense
dropout: Dropout
dense_1: Dense

Fig. 2. CNN model

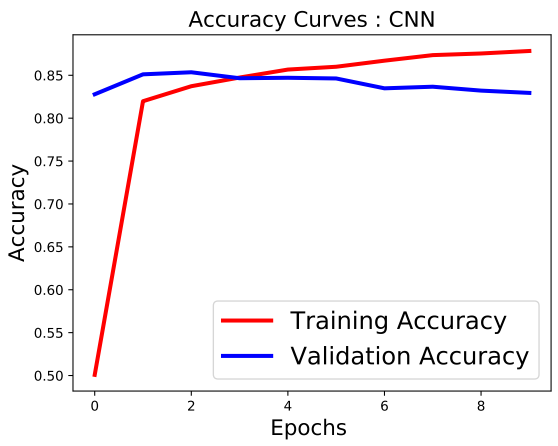

Fig. 3. Accuracy for CNN

labels with overlapping between different classes, and also the fact that some classes had orders of magnitudes more training examples. Classes containing a small amount of data points and vague classes were then merged into more general classes reducing the total number of labels from 117 to 16. Training on these more general classes yielded much better results. The results from the training of models can be seen in Table II.

TABLE II
ACCURACY RESULTS ON TRAIN AND TEST SETS

Model	Train	Test
Naive Bayes	0.80518	0.79853
DNN	0.82673	0.82343
CNN	0.85764	0.86394

Since the dataset was large enough we chose not to do any cross validation and relied on the split dataset instead. Several runs indicate that cross validation for evaluation was not needed as the results were consistent with only a small variance.

Plotting the results on a confusion matrix also helped identify classes that were hard to distinguish between. As an example, a class named ”work/business” was removed due to being hard to classify for all models. On the other hand, two classes named ”society” and ”politics” were merged due to high similarity in meaning and vocabulary creating confusion for the models. The final confusion matrices for the three different models is seen in figure 7, 8, and 9.

The training accuracy over time for the DNN and CNN can be seen in figure 3 and 5. Similarly, the training loss over time is seen in figure 4 and 6.

VI. Discussion

Overall the classification of the theme went well and with the CNN, an 85 percent accuracy with 16 classes is remarkable. However, we are lacking information about how it would perform against a human in the same task. This would be useful an useful addition to do a complete evaluation of the performance.

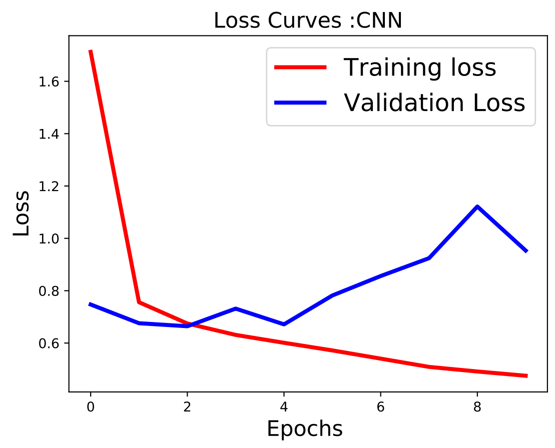

Fig. 4. Loss for CNN

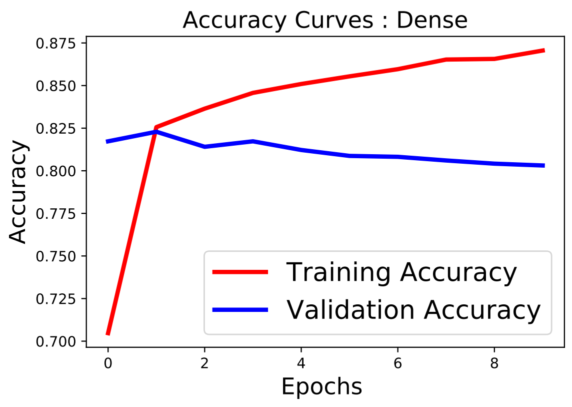

Fig. 5. Accuracy for DNN

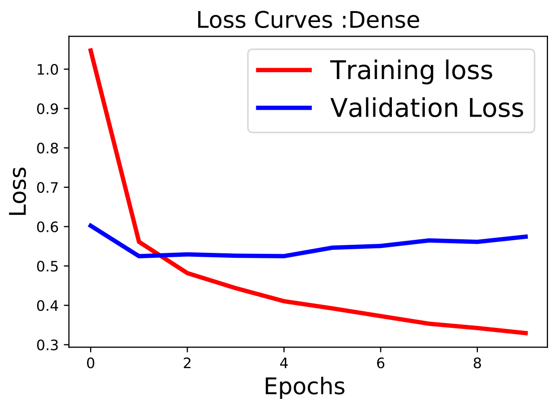

Fig. 6. Loss for DNN

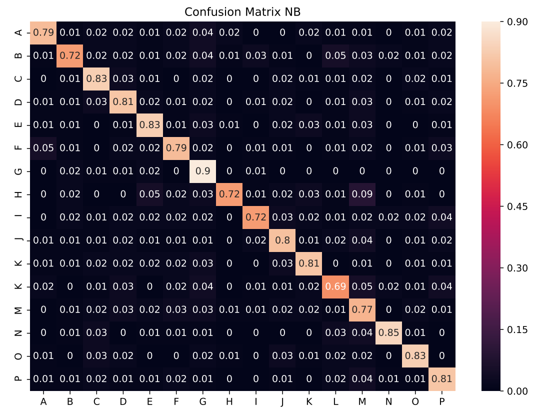

Fig. 7. Confusion matrix for NB

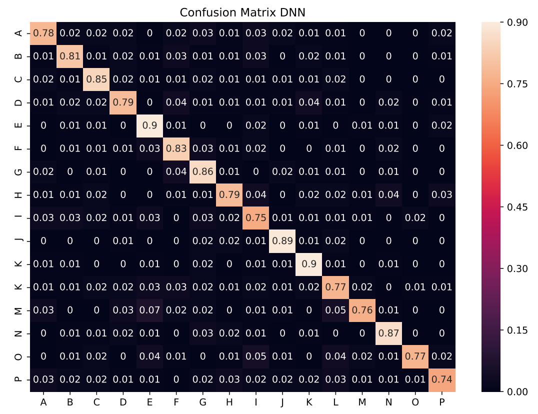

Fig. 8. Confusion matrix for DNN

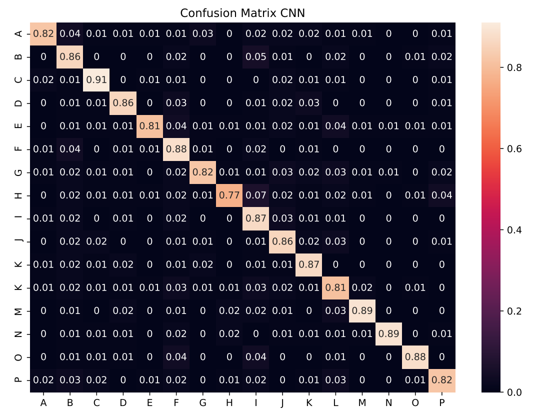

Fig. 9. Confusion matrix for CNN

One thing that was noted was that the quality of the dataset was not the greatest. Several of the sentences in it could have easily been classified as something else rather than the one that was provided. This could be the reason why the classifiers did not perform that well. Also to be noted is that the complexity of the neural networks had to be kept low, as the models started to overfit to the data after only three to five epochs. This can be explained by either the dataset or the model or both. The overfitting was even faster when ran on the original dataset and the model started to overfit already after one epoch. To try to reduce the overfitting of the CNN, we tried to reduce the complexity. However, while it did reduce overfitting a little, it came at a too high cost on the loss and accuracy of the model, so we preferred to use early stopping to stop training when the validation stops improving.

The confusion matrices expose that the overall accuracy on a per class basis is consistent, without any class showing significantly worse results. However, we can notice the overal better performance across classes of the CNN, with a minimal accuracy on the diagonal of 0.77 which is the only value under 0.8, compared to 0.74 for the DNN and 0.69 for the NB, but with multiple classes under 0.8 for these two.

Comparing the three algorithms, we observe results in accordance with expectations: more complexity improves the performance, so the CNN performs better than the DNN which itself performs better than the NB model. However, the improvements are of a few percent of accuracy, which is important if the goal is to maximize accuracy, but the added complexity can also not be worth it in applications where such difference in accuracy is not significant.

VII. Conclusion

From this research, we can conclude that training a model to perform text classification can provide great accuracy. However, the performance of the algorithm will depend on the model. From the three tested model which are NB, DNN and CNN, CNN performs better, probably due to its ability to take advantage of the order of the words in the sentence. However, the difference in performance when compared with the increase in complexity remains contained. To conclude, our result demonstrate the ability of models to perform correctly on the task of theme classification, which is of great omen for the future of the field.

VIII. Future work

To pursue this research, it would represent a good addition to study how the models fair against a human in a similar task. Moreover, our results showed potential weaknesses due to the potential poor quality of our dataset, so more work on the date or the utilization of a better dataset could improve significantly the results. Also, other models could be tried, especially Recurrent Neural Networks which have demonstrated remarkable results in multiple natural language processing tasks. Finally, one could implement a specific model to find the overall theme of an entire text which would not be based on the individual study of the sentences of the text.

Contributions

- Albin Larsson Forsberg: Implemented the Naive Bayes Classifier and dictionary generation
- Rémy Zawislak: Implemented the Dense NN and Convolutional NN.

References

[1] Kim, Sang-Bum, et al. "Some effective techniques for naive bayes text classification." IEEE transactions on knowledge and data engineering 18.11 (2006): 1457-1466.

[2] Rennie, Jason D., et al. "Tackling the poor assumptions of naive bayes text classifiers." Proceedings of the 20th international conference on machine learning (ICML-03). 2003.

[3] Kowsari, Kamran, et al. "Hdltex: Hierarchical deep learning for text classification." 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2017.

[4] Chen, Tao, et al. "Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN." Expert Systems with Applications 72 (2017): 221-230.

[5] Hassan, Abdalraouf, and Mahmood, Ausif. "Deep learning for sentence classification." 2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT). IEEE, 2017.

[6] Madadipouya, Kasra. Csv dataset of 76,000 quotes, suitable for quotes recommender systems or other analysis., 07 2016.

[7] Bird, Steven, Klein, Ewan, and Loper, Edward. (2009), Natural Language Processing with Python, O'Reilly Media.

[8] Chollet, François. (2015), Keras, Github: https://github.com/fchollet/keras

[9] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[10] Pennington, Jeffrey, Socher, Richard, Manning, Christopher D. GloVe: Global Vectors for Word Representation. 2014

[11] Maheshwari, Akshat. Report on text classification using cnn, rnn han. https://medium.com/jatana, 2018.

[12] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[13] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[14] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[16] Geoffrey Hinton (2014) https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf