Voice to Image Generation

David Ekvall dekvall@kth.se
Sabeen Nawaz sabeen@kth.se
Qingyan Ben qingyan@kth.se

Abstract

Generating images with only your voice would be helpful for artistic creation and even film making. We propose such a method where the model is consist of three different sequential parts: audio to text, text to feature space embedding and text embedding to image. For audio to image part, a k-layer neural network trained with minibatch gradient descent is used. For the second part, we use a text embedding model with places similar words near each other in feature space. Finally, a conditional Generative Adversarial Network (GAN) is used to generate the image. We experiment on two datasets: The Speech Command dataset and the `flickr_30k` dataset. The audio network recalled 12.31 % of the words and the conditional GAN produced images of poorer quality than expected. Supposedly because of the complex scene structure of the data in the `flickr_30k` dataset.

1 Introduction

Automatic image generation is a field in which you produce an image for a specific purpose. There exists methods to generate images from natural language descriptions like [1][2]. We would like to combine the image generation with and audio encoder so that you would be able to dictate your image. With the rise of smart-speakers it could be a fun way for both artist and amateurs to create art.

With the rise of smart-speakers people are less keen to use their keyboard as an input device, they will simply use their voice. Another focus of research nowadays is to automatically generate captions from images. We propose a way to combine these two while flipping the latter the other way around.

One can consider this model to be three separate parts combined: audio to text, text to feature space embedding and finally text embedding to image. We must develop a method for combining these three parts together.

For the audio to text part, a k-layer neural network trained with minibatch gradient descent is implemented with a preproccessed audio input, which is "transformed" into feature vectors containing Mel Frequency ceptstral coefficients (mfccs). To convert the text to a text embedding, we use a word2vec approach[3] in which we ignore the order and look at each word separately. In this project we aim to generate real images with many objects and relationships, so employ a Conditional Generative Adversarial Network (GAN) structure. It has two parts, the Generator and the Discrimnator. During training, the Discriminator is optimized to notice whether or not an image is created by the Generator and the Generator is optimized to create as realistic images as possible. Both of these are updated continuously in order to improve network performance. Since we have a limited amount of captions we also perform a simplified version of condition augmentation.

We experiment on two datasets: The Speech Command dataset [4] was used for the training and testing of the deep neural network, which contain 65000 audio file of people all around the world and the `flickr30k dataset` is used to generate the images, it has 30000 images with 5 captions for each image. In audio to text training, 80 percent was selected as training set and the remaining 2 percent was used as validation set for calculating the cost. In the end we were able to generate images, although the quality was worse than expected, from the network. Our audio encoder also performed worse than expected with only a 12 % recall.

2 Related Work

There have been are many attempts on Text to Image generation before, For example, Variational autoencoder(VAE), Generative Adversarial Network(GAN) and Seq2seq. However, the image produced by VAE is of low resolution that is too vague. For Seq2seq, although it can produce image with more details, it can only be used in generating abstract scenes from textual descriptions, instead of real image. Thus, we decide to using GAN to generate image from text.

General adversarial networks are a fairly new approach to image generation. One such approach is the Deep-Convolutional Adversarial Network(DC-GAN)[5]. The basis of a GAN is to train two components, one generator and one discriminator. Basically the Discriminator is optimized to notice whether or not an image is created by the Generator and the Generator is optimized to create as realistic images as possible for the generator. In essence they play the game $V(G, D)$ where

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[log(D(x))] + \mathbb{E}_{z \sim p_z(z)}[1 - log(D(G(z)))] \tag{1}$$

Which has a global optimum where $p_g = p_{data}$ and with sufficient data converges to that optimum [6]. In the beginning of training though, samples from G will be poor and has a high chance of being rejected by D so it has been found out that in practice one should maximize $log(D(G(z)))$ instead of minimizing $log(1 - D(G(z)))$[2].

DC-GAN does use any information about the scene when creating images, since that is something we are intersted in, we look at an approach called ConditionalGAN [7] where we condition the Generator, G and Discriminator D with a conditioning value c along with the latent vector z. This conditioning variable is associated with the text which describes the image. Furthermore, in [8] the authors report using a technique called conditioning augmentation where instead of taking the text representation directly they sample the condition variables from a normal distribution around the true c.

The most common approach for speech recognition is Recurrent neural network, which have been successfully implemented in [9] and [10]. Convolution neural networks have also been used for speech recognition in [4]. The model is developed by Tensorflow and is specifically developed for their Speech Command dataset.

3 Data

3.1 Datasets for Speech Recognition

The Speech Command [4] dataset was used for the training and testing of the deep neural network. The dataset contains 65000 audio files of people saying 30 different words. Each audio file is 1 second long. The samples have been collected using crowdsourcing, from people all over the world. There were no special requirements on the quality of the audio file meaning that background noise can occur in the samples. 18 percent of the dataset was used as the test set, 80 percent was selected as training set and the remaining 2 percent was used as validation set for calculating the cost. The audio samples were processed and extracted to feature vectors using Mel Frequency Cepstral Coeffecients (mffc). Each audio file was extracted to five mfccs, mainly because of computational advantage with a smaller size of the feature vector. Mffcs were computed using functions from LibROSA, which is a python package for audio analysis. [11]

3.2 Datasets of Image Generation

To generate the images the `flickr_30k` dataset is used. It contains 30 000 images with 5 captions for each image. The images are resized to 64×64 in order to reduce the size of the network since training a GAN is a very time-consuming process. Most images are in a 4:3 format so images will not be subject to motif-altering distortions. The captions can sometimes describe different parts of the image and cropping would risk removing important elements of the depicted scene. Before the images are fed into the network, each one of the three color channels are normalized. The text embedding for each caption were generated beforehand and added to the dataset instead of re-generating it each training-cycle.

4 Method

4.1 Feature extraction of audio samples

One very commonly used feature extraction method of audio files is Mel Frequency ceptstral coefficients (mffcc). The mffcs of a signal is a feature vector which describe the frequency components the signal is comprised of. Mffcs are used to describe audio signals and are commonly used when building deep networks for speech recognition. The main idea of mffcs is to squash the frequencies in a signal into Mel scale. The audio signal is firstly transformed into a discrete signal sampled at some frequency and the power spectrum (Discrete Fourier Transform) is computed for the signal. Mel filterbank is then applied to the power spectrum of the signal. Mel filterbank is a set of triangular shaped filters, usually 20-40 filters. The filter gives a response which resembles the Mel scale (1). (1) F(Mel)=[2595*log10[1+f]700] The filtered signal is then transformed to the time domain using discrete cosine transform, and we get the mel frequency cepstral coefficients. The final output is k number of vectors where k is the number of mel filterbanks.[12]

4.2 Speech recognition model

The speech recognition is done with a k-layer neural network. The training of the network is done with mini-batch gradient descent and batch normalization is applied at each layer. During the training, cyclic learning rates [13] are used when updating the weights and biases of each layer. The cost is computed with cross-entropy loss. ReLu activation function was used at each layer. The weights are initialized with random numbers sampled from $\mathcal{N}(0, 1/\sqrt{m^{l-1}})$ where m^{l-1} is the number of nodes in the previous layer. The biases are initialized to vectors containing zeros. When training with cyclic learning rates, the learning rate varies linearly between a specified lower and upper boundary for a specified number of cycles. The table below describes the hyperparameters of minibatch gradient descent along with cyclic learning rates.

Parameter	Description
η_{min}	the lower boundary for cyclic learning rate
η_{max}	the upper boundary for cyclic learning rate
n_s	is the stepsize of the cyclic learning rate. $2n_s$ corresponds to a whole cycle with $2n_s$ update steps
λ	is the degree of regularization that is used
n_{batch}	is the size of the batch in minibatch gradient descent
l	is the number of cycles for learning rate, i.e. the network is trained for in total $l * 2 * n_s$ update steps

Table 1

4.3 Text Encoding

The text embedding, $\varphi(t)$, of the text description is generated by a pre-trained encoder [14]. We used a word2vec [3] model trained on 3.3 billion Google news articles. Every caption is fed word for word into this model to yield a 300-dimensional feature space text embedding. Because of the captions' different length we calculate the max and min elements row-wise in the sentence. These vectors are then stacked into a text embedding of length 600. This yields an adequate feature space representations for short sentences [15] although the sentence structure is lost. In other work the output has been averaged [14] or only used a simple bag of words model, thus order is deemed non-essential for the success of the network.

4.4 Image Generation

The image generation is done with a Conditional GAN [7] with small perturbation to the condition variable. In [8], condition augmentation is used in which instead of just using the fixed text embedding $c = \varphi(t)$, they sample the latent variables $\hat{c}$ from a normal distribution $\mathcal{N}(\mu(c), \Sigma(c))$, where $\Sigma(c)$ is the diagonal covariance matrix. This sampling mitigates discontinuity in the latent data manifold and

therefore makes training process more manageable. We approximate this behaviour by adding a small perturbation p, to the condition variable which is sampled from a normal distibution $\mathcal{N}(0, 0.001)$ for which $\hat{c} = c + p$. Since a property of the text embedding is that words with similar meaning are similar in feature space this should also help generalize the output of the network.

The latent vector z is sampled from $\mathcal{N}(0, 1)$ and has a length of 64.

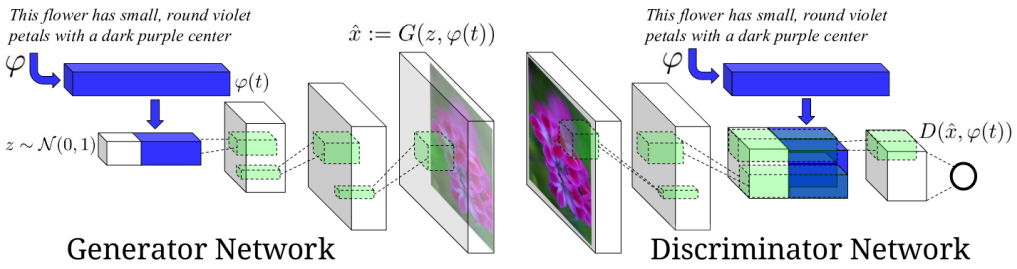

Figure 1: Overall network structure of the image generation. Image from [2]

4.4.1 Generator

As seen in the left part of Figure 1, the latent vector z is appended to $\hat{c}$ and fed into the network. The input vector is upscaled to $\mathbb{R}^{512\times4\times4}$ with transposed convolution and batch normalization with a Relu activation function. This is repreated for dimensions $\mathbb{R}^{256\times8\times8}$, $\mathbb{R}^{128\times16\times16}$, $\mathbb{R}^{64\times32\times32}$ and $\mathbb{R}^{3\times64\times64}$ with tanh as activation function for the latter. The output is our 64 image with three color channels. Every convolution layer has a kernel size of 4 and stride of either 1 or two, depending on the layer in question.

4.4.2 Discriminator

The Discriminator can be seen to the right of Figure. 1. It consists of 4 convolution layers for the input image. There is also a leaky-ReLu after every layer with an incline of 0.2. These convolution transform the 64×64 image into $\mathbb{R}^{64\times32\times32}$, $\mathbb{R}^{128\times16\times16}$, $\mathbb{R}^{256\times4\times4}$. Kernel size is 4 and stride is 2. At this point the discriminator does an 1d convolution of the text embedding and reshapes it to append to the current state of the image convolution. This stack is then fed into a convolution layer with 1 as output dimension. The activation function for this layer is a sigmoid function.

4.4.3 Training process

The discriminator needs to be able to distinguish generated images from real ones, but it also need to be able to determine if the text embedding is representative of the depicted scene. The discriminator will thus be given three different kinds of input during training: Real image with real text (s_r), Real image with wrong text (s_w) and fake image with real text(s_f). The generator thus has the following loss function.

$$\mathcal{L}_\mathcal{D} = log(s_r) + \frac{1}{2}\left(log(1 - s_w) + log(1 - s_f)\right) \tag{2}$$

The generator simply has to create the best fake images possible and has the loss function

$$\mathcal{L}_\mathcal{G} = log(s_f) \tag{3}$$

The training was done in minibatches with 64 images in each using the ADAM optimizer with a step size of 0.0002. A simplified view of the training algorithm can be seen in Table 1. The image generation was written from scratch in PyTorch and trained on a NVIDIA Tesla K80 GPU for approximately 28 hours, or 100 epochs, on the `flickr_30k` dataset. The GAN has thus been trained on around 15 million different text and image combinations.

Algorithm 1: The training algorithm for the image generation, using Mini batch SGD with step size α for simplicity.

Input : Minibatch images x, Minibatch text embeddings $\varphi(t)$, number of training batch steps S

for $i = 1$ *to* S **do**

- $z \sim \mathcal{N}(0,1)^Z$ {Generate latent vector}
- $p \sim \mathcal{N}(0, 0.001)^\phi$ {Generate perturbation vector}
- $w \sim \mathcal{N}(0,1)^\phi$ {Generate wrong text embedding}
- $\hat{c} \leftarrow \varphi(t) + p$ {Perturb text embedding}
- $\hat{x} \leftarrow G(z, \hat{c})$ {Generate fake image}
- $s_r \leftarrow D(x, \hat{c})$ {Real image, Real text}
- $s_w \leftarrow D(x, w)$ {Real image, Wrong text}
- $s_f \leftarrow D(\hat{x}, \hat{c})$ {Fake image, Real text}
- $\mathcal{L}_\mathcal{D} \leftarrow log(s_r) + (log(1 - s_w) + log(1 - s_f))/2$
- $D \leftarrow D - \alpha \partial \mathcal{L}_\mathcal{D} / \partial D$ {Update discriminator}
- $\mathcal{L}_\mathcal{G} \leftarrow log(s_f)$
- $G \leftarrow G - \alpha \partial \mathcal{L}_\mathcal{G} / \partial G$ {Update generator}

end

5 Experiments

5.1 Audio to text

In order to find good values for the hyperparameters (described in table 1) for the network, a coarse search was done. η_{min} and η_{max} was explored in the range of 1e-7 to 1e1. λ was explored in the range 1e-3 to 1e-1. The network was also trained with varying n_s and number of cycles l. n_s was explored in the range of $1 * 100$ to $10 * 100$, while l varied between 5 and 15. We also experimented with different numbers of layers in the network.

5.2 Text to image

The captions in the dataset `flickr_30k` can be considered to be fairly complex as the scene consists of many parts and a multitude of classes. In [2] the datasets used were of fairly uniform data such as a flower or a bird. Our model had to work with a multitude of different words.

6 Result and Discussion

6.1 Result

In Figure 2 the result of the loss function can be seen for the Discriminator D and Generator G. The plot is the result of a moving average with a window size of 1000 data points to identify the trend of the loss function. Looking specifically at the discriminator loss we see that the loss seem to converge towards 0 which means the generator isn't able to deliver good enough images. There is an apparent negative correlation between the Generator and Discriminator loss which it should be since they are playing a min-max game, though, the generator shouldn't be losing this bad. We believe that is because of the complex captions in `flickr_30k` dataset. We might have had better success to use dataset with less classes, such as `Oxford 102 Flowers` or the like. Datasets which handles far less object classes in their captions. Unfortunately there was no time to train such a network when we realised that. We see in Figure 3, 5, 4 that even after 100 epochs the scene depicted in the generataed image is not apparent.

In Figure 6 we can see how the validation and training cost is converging towards 2.5. The best test set accuracy achieved with the Speech Command dataset on 8-layer network was 12.31 %. The table 2 below displays the top results achieved with this network. Different number of layers with different amount of hidden nodes were tested but the top results were achieved with 8-layer network where the nodes settings were: 100,100,100,100,70,50,50,30.

Figure 2: The loss of the generator and discriminator during training

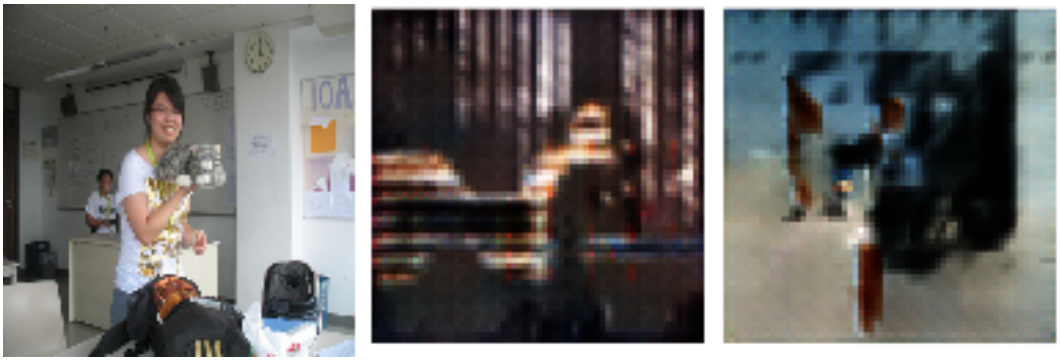

Figure 3: Caption: *Girl holding cat stuffed animal.*
From left to right; Captioned Image, Output after 50 epochs, Output after 100 epochs

η_{min}	η_{max}	ns	l	λ	$nbatch$	accuracy
1e-3	1e-1	6*110	5	0.0001	100	12.31
1e-3	1e-1	5*110	5	0.0001	100	11.94
1e-3	1e-1	6*110	3	0.001	100	11.30

Table 2

6.2 Future Work

Time limitation: Since there is no enough time to training audio2text, we could not covering all words. Thus, we just training limit number of word and assume that audio2text is done. With more time on hands we would also consider trying out implementing Convolutional Neural Network for speech recognition with the Speech Command dataset. We would also try to preprocess the speech data differently using more amount of mffcs and see whether that affects how well the network learns to label the audio files.

The above image generation in pixel level based on GAN is currently focused on the generation of a single object. When we want to generate complex scenes with multiple interactive objects in the image based on the text, it is very difficult to understand this high-level semantic information from the pixel level.

Figure 4: Caption: *A little boy shows off his suitcase full of toys.*
From left to right; Captioned Image, Output after 50 epocs, Output after 100 epochs

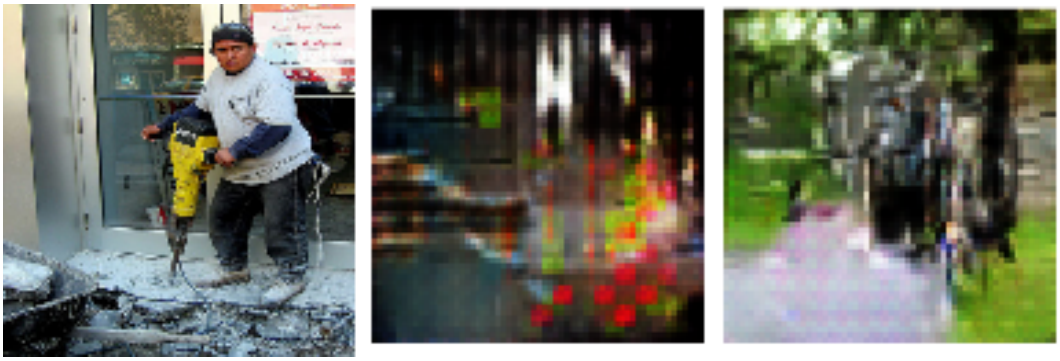

Figure 5: Caption: *A man with a jackhammer demolishing cement.*
From left to right; Captioned Image, Output after 50 epochs, Output after 100 epochs

Extensions of this work include using the condition augmentation presented in [16] instead of the perturbation approach. Another extension would be to actually use the scene information that lies in the ordering of the words instead of just taking the minimum and maximum elements and thus ignoring the order. Another improvement to our approach is to stack discriminators and generators on top of each other [16] which could yield in improved quality and thus the creation of more realistic images.

7 Conclusion

This has been a massive project and we aimed far too high. In hindsight, we should have chosen one of the three parts and put our full focus on that. Even if the results are worse than expected we still feel like we have learnt a lot because we implemented the majority of the networks from scratch. Future expansions would be to make the text embedding context aware and increase the quality of the output images as was done in [8]. Future implementations could be creating videos from a music track in order to create a new kind of music aware screensaver instead of the typical Windows Media Player one.

Source code for the image generation and speech recognition is available at:
`https://github.com/dekvall/dd2424_image_generation`

References

[1] S.Reed et al. “Whatwhere: Learning what and where to draw”. In: *arXiv In NIPS* (2016).

[2] S. Reed et al. “Generative adversarial text-to-image synthesis.” In: *In ICML* (2016).

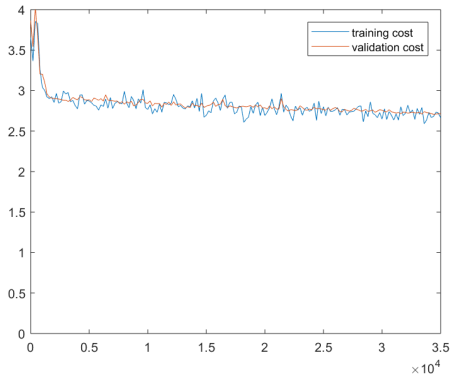

Figure 6: Training and validation set loss every 200th update.

[3] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[4] P. Warden. "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition". In: *ArXiv e-prints* (Apr. 2018). arXiv: 1804.03209 [cs.CL]. URL: https://arxiv.org/abs/1804.03209.

[5] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[6] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[7] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[8] Han Zhang et al. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5907–5915.

[9] Awni Hannun et al. "Deep speech: Scaling up end-to-end speech recognition". In: *arXiv preprint arXiv:1412.5567* (2014).

[10] Haşim Sak, Andrew Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". In: *Fifteenth annual conference of the international speech communication association*. 2014.

[11] Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th python in science conference*. 2015, pp. 18–25.

[12] Jorge Martinez et al. "Speaker recognition using Mel frequency Cepstral Coefficients (MFCC) and Vector quantization (VQ) techniques". In: *CONIELECOMP 2012, 22nd International Conference on Electrical Communications and Computers*. IEEE. 2012, pp. 248–251.

[13] Leslie N. Smith. "No More Pesky Learning Rate Guessing Games". In: *CoRR* abs/1506.01186 (2015). arXiv: 1506.01186. URL: http://arxiv.org/abs/1506.01186.

[14] Scott Reed et al. "Learning deep representations of fine-grained visual descriptions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 49–58.

[15] Cedric De Boom et al. "Representation learning for very short texts using weighted word embedding aggregation". In: *Pattern Recognition Letters* 80 (2016), pp. 150–156.

[16] Han Zhang et al. "Stackgan++: Realistic image synthesis with stacked generative adversarial networks". In: *arXiv preprint arXiv:1710.10916* (2017).

8 Appendix

9 Learning Outcomes

9.1 David

- Learning the PyTorch framework. I have never worked with PyTorch before and it was a joy to learn.
- Learning about GANs. Before this project i had no idea that GANs even existed. I think it's brilliant how it utilizes the min-max principle you often see in AI for games such as chess etc.
- Learning about feature space representations. I have delved into different approaches of how you can get something meaningful from words and explored bag of words models, word2vec and structured scene graphs. I have learned that the concept of context is not an easy one to transfer into machine learning algorithms.

9.2 Qingyan

- Enjoying the process of team work and brain storming.
- Learning some text to image methods. This experience make me addicted to this field. I find another method called Image Generation from Scene Graphs presented by Justin Johnson. From the paper, this method can produce image with more complex relationship. I will try it in the summer holiday.
- Through the project, we learn how to complete a research, from idea finding, paper searching, coding to report writing. This will be really helpful for my master thesis and future work.

9.3 Sabeen

- I learned how to process speech data using Mel frequency cepstral coefficients. Prior to this project, I had no idea about how to extract features from audio files.
- I also learned how to implement a k-layer network with batch normalization and training it using cyclic learning rate.
- Learning about GANs was also very interesting.
- I also read up alot on RNNs with lstm cells because we were initially thinking of using those for speech recognition but we were not able to implement it successfully so we had to switch to a more simple network in a very late stage of the project. However, I did learn alot about RNNs.