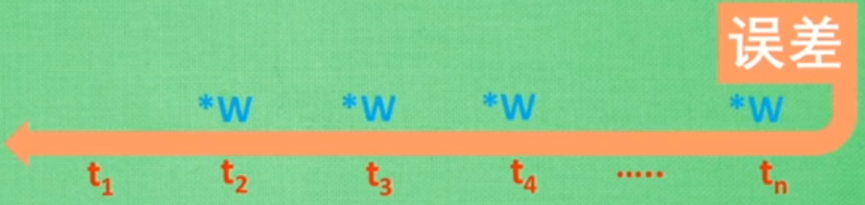
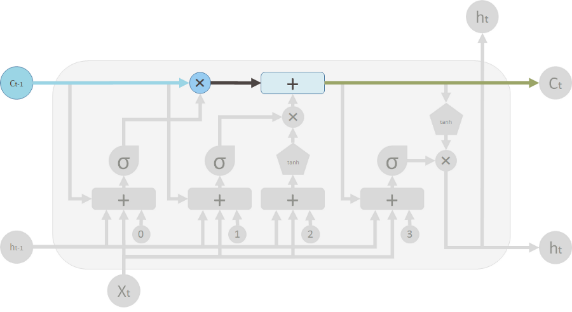
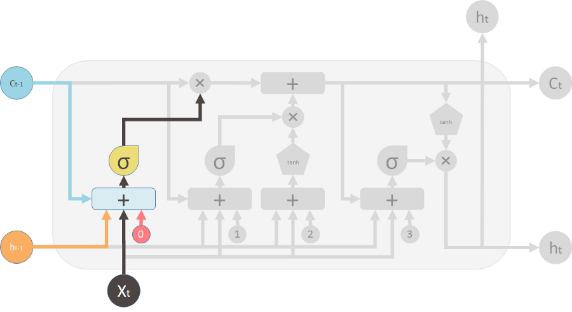
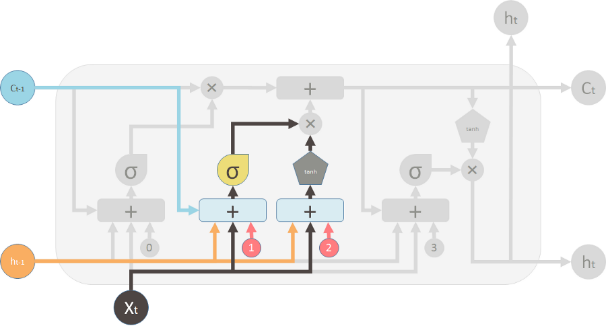
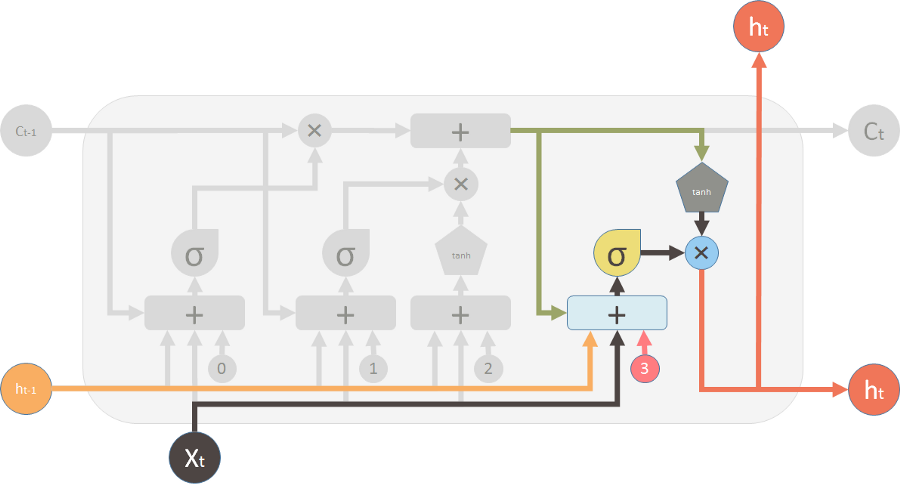
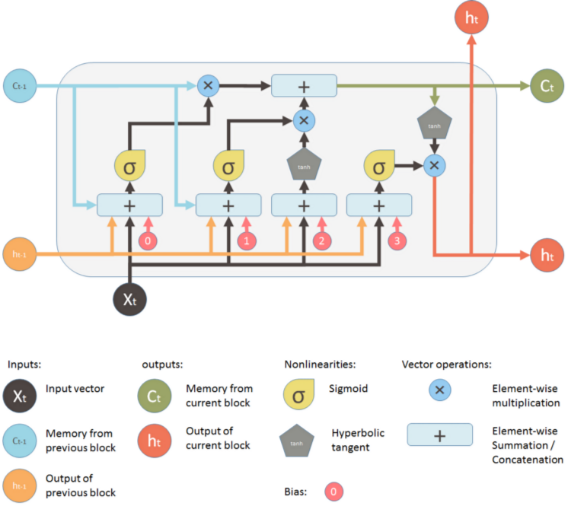
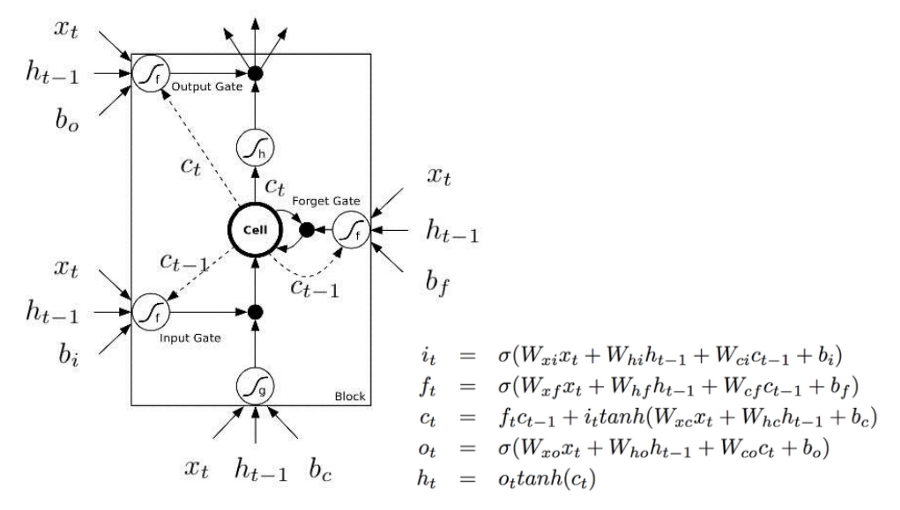
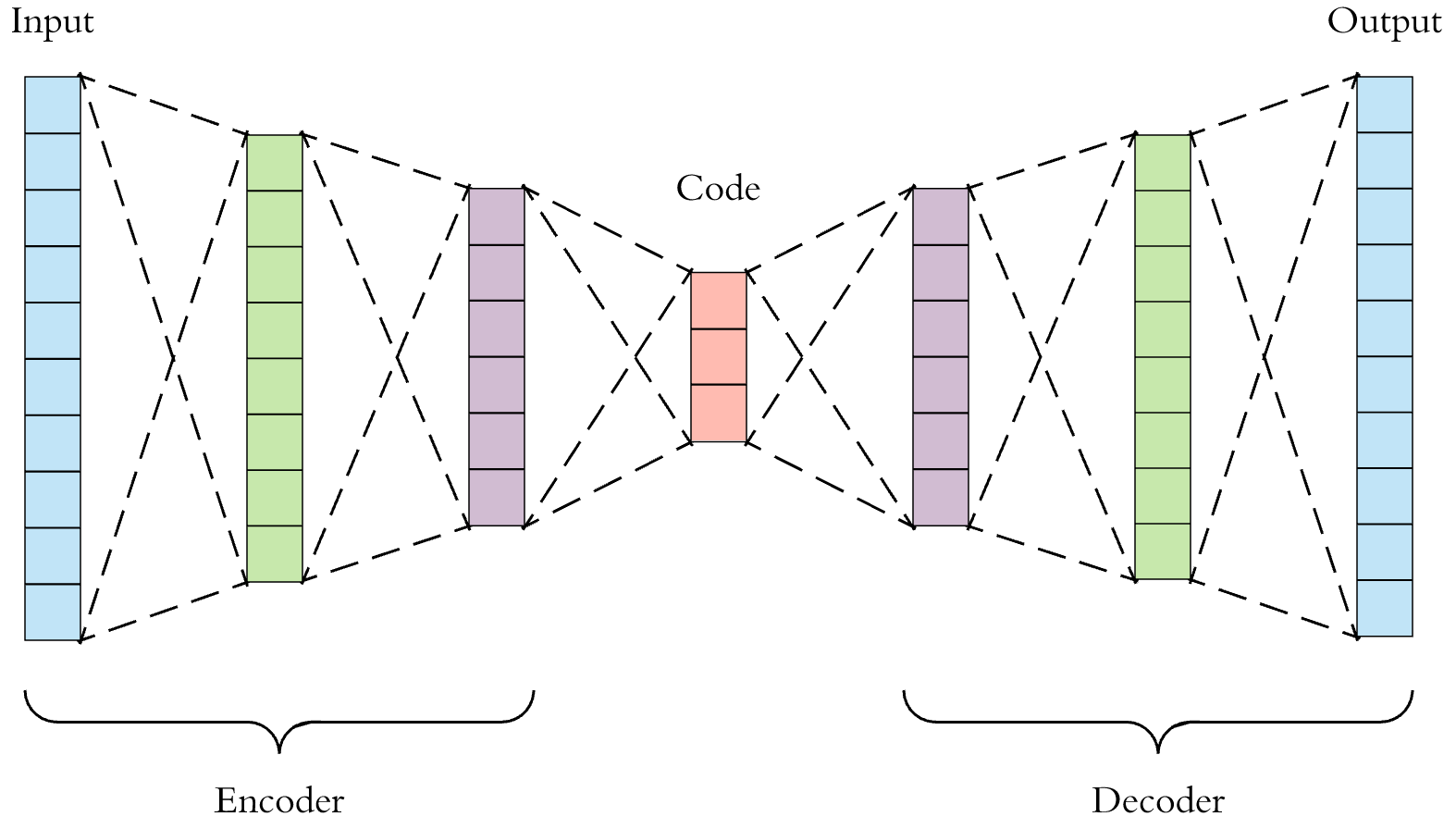
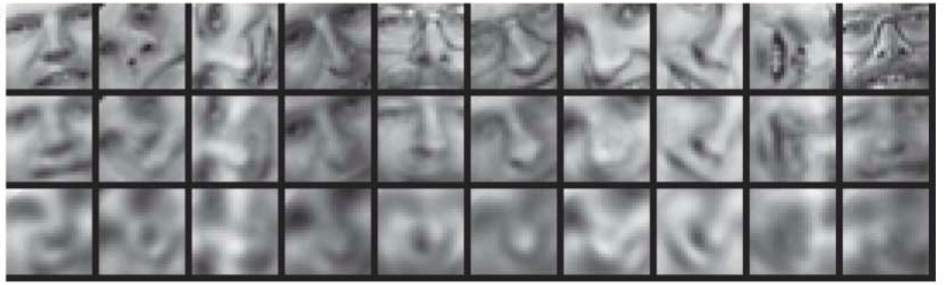
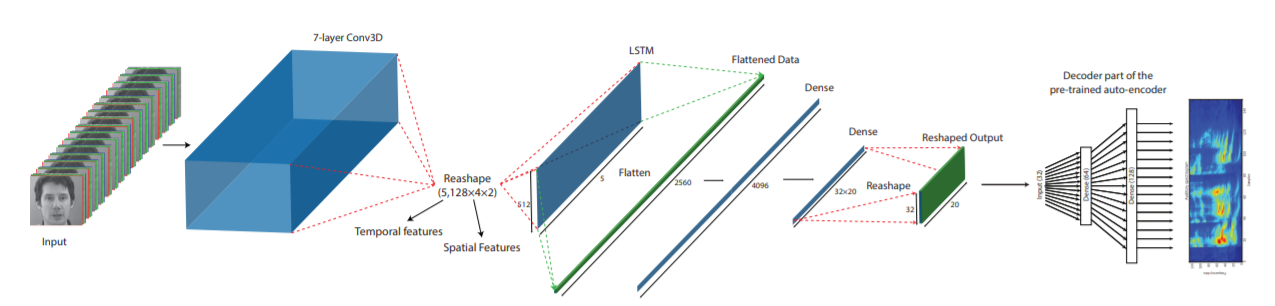
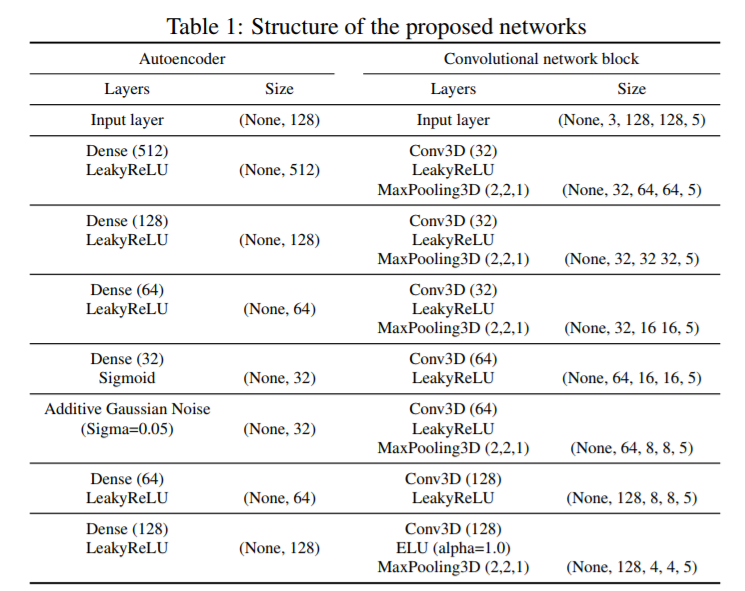
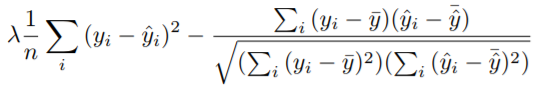
**Machine Learning Final Project**

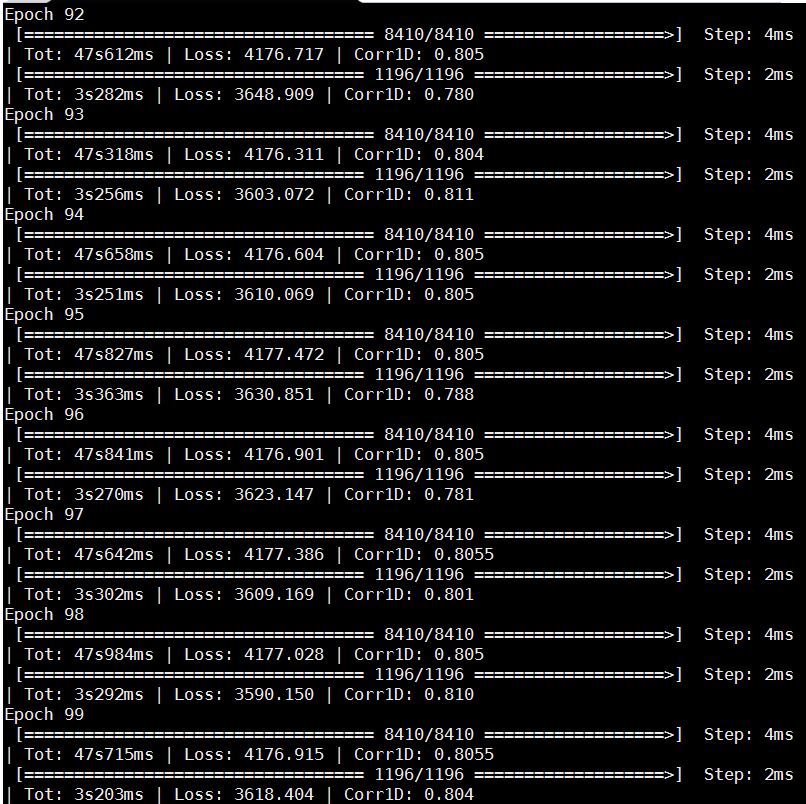
* **Paper title:**   
  Lip2AudSpec: Speech reconstruction from silent lip movements video
* **Authors:**  
  Hassan Akbari, Department of Electrical Engineering, Columbia University  
  Himani Arora, Department of Electrical Engineering, Columbia University  
  Liangliang Cao, Department of Electrical Engineering, Columbia University   
  Nima Mecgarani, Department of Electrical Engineering, Columbia University
* **Published:**  
  ICASSP2018
* **Significance, Motivations, and Contributions of the research in that paper:**   
  作者希望從影片中，判讀裡面說話的人的影像，**從唇語或是臉部變化，轉成語音輸出**。重建語音的困難是音調或語氣train完會不好辨識，聽起來就像機器人說話一樣，所以這篇論文提出了一種end-to-end的訓練方式，發現效果蠻好的，不僅重建後仍可聽出語音訊息，還能恢復發音的風格。
* **Background description of the core technique:**  
  以下介紹架構會用到的觀念  
  **(1) CNN**  
    
  CNN包含convolution、pooling、fully connected layer。convolution代表、pooling能等比例縮小數據量、fully-connected則是把神經元和上一層全部連接。  
  **(2) LSTM**  
  這是RNN的改版，所以這裡先介紹RNN：  
    
  RNN(Recurrent Neural Network)會把整個句子、所有的data都餵進neural network中，連接多層的NN，每一層負責1time step。他的缺點是模型會忘記前面說的話，而且在誤差反向傳遞時，一個<1的數或一個很大的數，一層一層往前傳乘上權重時，可能導致梯度消失或梯度爆炸！  
  LSTM(Long Short Term Memory)改善了這個問題，**在每一個time step餵data，而不是一次把全部丟進去**。同時，它**新增了forget valve和new memory valve來控制權重**，需要忘記多少比例的舊資訊、以及需要採納多少比例的新資訊。  
     
  **首先，介紹Forget Valve：**這是其中一層NN，input有X\_t (input of current step)、h\_t-1 (input of previous LSTM)、C\_t-1 (memory of prevoius unit)，b\_0則表示bias vector，經過sigmoid activation function後輸出。  
    
  **接著介紹New Memory Valve：**經過tanh activation function後輸出  
    
  **最後Output**會根據前面求得的資料配權重，混合出結果，整體架構圖與公式如下所示：  
    
  **(3) autoencoder**  
  這是一種unsupervised learning，將資料經過壓縮、降維，使用更少的信息量來代表這個資料，再**透過這關鍵的code重建出原本的資料**。通常可以用在壓縮高清圖片，只留下重要的code部分，讓model train起來更簡單輕便。  
    
  使用autoencoder降維，儲存code部分後，重新decode回去，結果如下。第一頁是原始圖片，第二列是autoencoder降維結果，第三列是PCA降維結果。因為PCA找eigenvalue大的維度，所以被限制成是linear的，而autoencoder則可以是非線性的encode/decode，**通常autoencoder的表現會超過PCA**：  
  
* **Objective of the research and Problem Formulation:**  
  以往音訊的重建，往往會使說話者的個人風格(如：音調、語氣)喪失，為了解決這類問題，作者設計了不一樣的架構，  
  作者提出結合影像和其對應的音訊，一起丟到model內(放在不同的部分)train，最後透過影像重建出音檔，結果顯示**這種帶有聽覺的聲音模型，比傳統只用聲音spectrum的model好很多**。使用2-D correlation去算accuracy，autoencoder的部分甚至可以達**98%**那麼高，在未來很有潛力！
* **Proposed method in the paper:**1. **先用音訊pre-train我們的auto-encoder**，而autoencoder會放在整個model的最末端，train好autoencoder後保留其參數。現在，model後面的部分做好了！  
  2. **接著要從前面開始train所有的model**，把圖片丟進去，分別經過7層NN、reshape、LSTM、3層NN、reshape出32\*20的值，32代表latent space(core)的dimension，20則是因為音訊有20ms，**後面一共會接20個autoencoder去train**。  
  此外，**每一層NN都要做batch normalization**，前面7層NN的部分每兩層做dropout(p=0.25)，後面LSTM和MLP的部分則是p=0.3做dropout。  
    
  3. **Error function:** 使用mean square error(MSE)加correlation:  
  
* **Experimental settings – Data Preparation:**1. 首先介紹dataset:  
  這篇論文使用了GRID dataset(The GRID audiovisual sentence corpus)，這個dataset由34個speaker一起完成，**每個speaker會講1000句話**，一句話六個字，而這些字是從有限的字典庫來的。  
  影片的樣子都大概是長這樣：  
    
  2. 接著介紹data(vedio frame)的處理方式：  
  會用到vedio和audio的data，這裡先介紹前者，audio的部分在第3點。  
  因為model主要是丟圖片進去，去重建語音，所以我們從影片以25fps抓出臉部的變化，存成一張張的圖片，放在不一樣的資料夾歸類好。  
    
  而抓臉的部分使用Ada-boost演算法，haarcascade\_frontalface\_alt.xml幫助我們把臉分類出來，從影片中標記出臉的影像，如下圖所示：  
     
  **一共有34個speaker，每個speaker的1000句話放在1000個資料夾，沒個資料夾又有75個紀錄臉部變化的frame。**  
  ◎從左邊的影片可以抓出右邊一張張臉的frame，所以這裡我想稍微介紹一下Ada-boost演算法：  
  AdaBoost全名為Adaptive Boosting。 AdaBoost是一種疊代算法，其核心思想是針對同一個訓練集(training set)訓練不同的分類器(弱分類器)，然後把這些弱分類器集合起來，構成一個更強的最终分類器(強分類器)。  
  AdaBoost結合Rectangle feature在人臉辨識的algorithm：首先會有一堆training的image，分別標示著人臉m張以及非人臉l張。對於每一張image 我們一開始分別給他們 1/m或是 1/l的weight，端看它們是人臉或是非人臉。 接著從一大堆 rectangle feature 裡面取出 T 個 feature 來，重複 T 次下面的步驟。  
  (1) 首先將所有的 weight normalize 成加起來為 1。   
  (2) 根據hypotheses function，選出一個error值最小的feature。  
  (3) 紀錄可以使 error 值最小的參數。   
  (4) Update weights，最後我們求出來的 classifier 就是由 T 個 weak feature 所組成的。因此對於丟入的 一張 image，就會由這 T 個 feature 來投票，每一個 feature 的投票的權重不太一 樣。但只有當加權值大於一半以上的所有分數時，才會認可這一張 image為人臉。  
  3. 生成data(audio)的方式：  
  剛才有提到，每個人會講1000句話，所以我們把1個speaker切出1000段影片，以44kHz做down-sample，得出音訊檔。  
  一共有兩種生成方式，matlab或python，兩者處理的方式不一樣，比較如下：

|  |  |  |
| --- | --- | --- |
|  | Matlab | Python |
| 轉頻率域spectrum | 使用模擬人耳的wav2aud function (<http://www.isr.umd.edu/Labs/NSL/Software.htm> 下載第三個檔案有function的詳細說明，由交大教授提出的，超猛) | Fast Fourier Transform ãfast fourier transformãçåçæå°çµæ |
| 維度 | 84 | 128 |
| 資料量 (9 : 1) | Train: 5568416 Test: 618713 | Train: 17045997 Test: 1894000 |

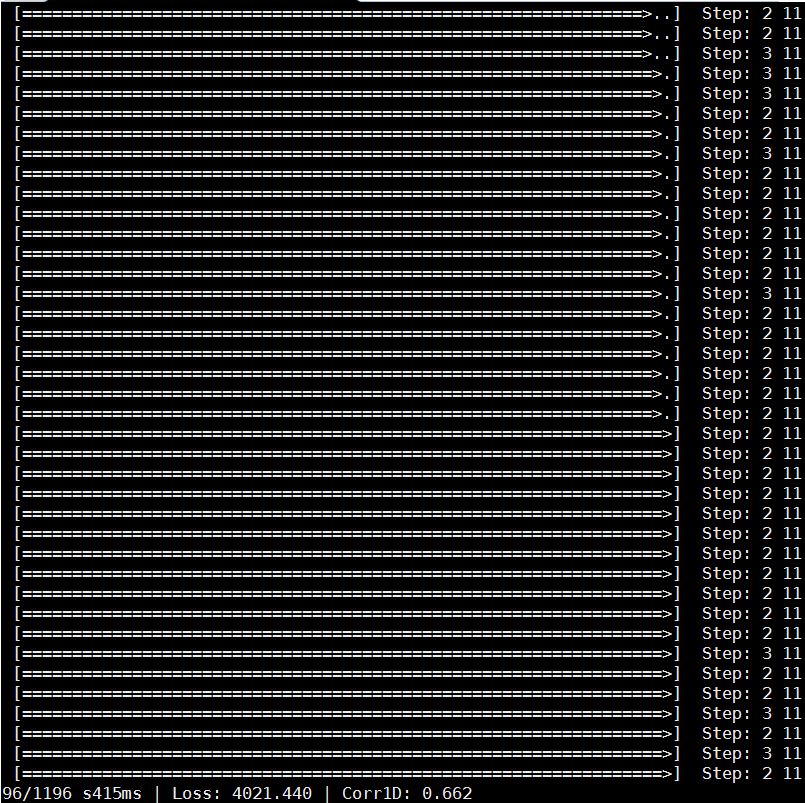
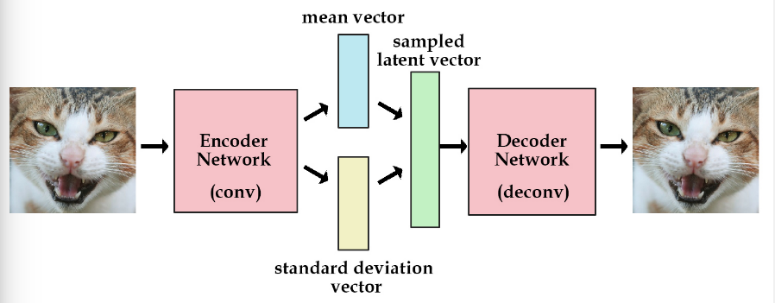
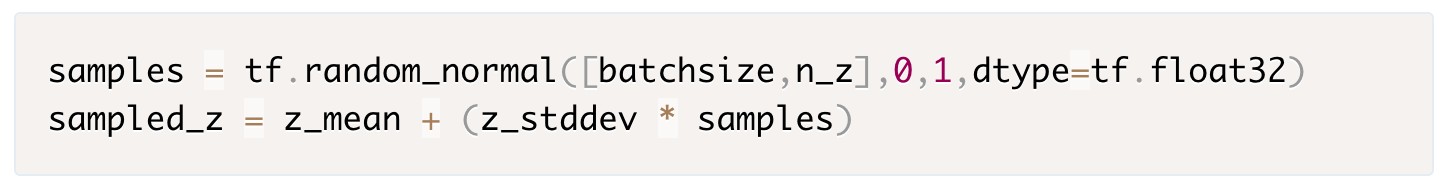
4. 餵進去的data長怎樣？  
可以從前面tabel發現input維度是(None, 3, 128, 128, 5)   
3: Derivative，包含和前/後面圖片同一個pixel的相關性，三個維度分別代表0階、1階(後面減自己)、2階(後面加前面減掉兩倍的自己)  
5: 五張圖片(frame)放一起  
128: frame的長跟寬

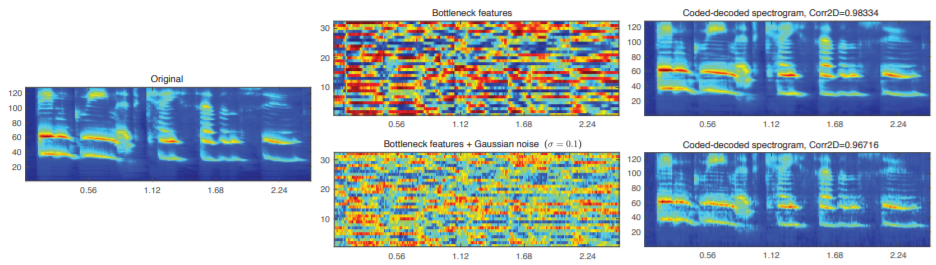
* **Experimental results:**1. 用聲音pre-trained autoencoder，重建後的聲音與原本的聲音Accuracy，根據paper可達98%  
  後來我**照著paper架構implement，發現最高只能達94%**  
  **推測是因為以下幾個原因：**  
  (1)GRID dataset一共有34個speaker，而paper只從裡面擇優挑了4個speaker，所以出來的數據比較漂亮。  
  (2)GRID官網提供50kHz和25kHz，因為50kHz才是原始音檔，25kHz是從50kHz來的，因為不是原始資料，所以train的結果跟真實情況可能有些微差異。  
  (3)作者只train了40個epochs，但我train了100個，可能有over fitting。  
  2. train autoencoder  
  把34個speaker全部丟進去train，從原本的維度一層一層降到latent space 32維，再一層一層還原回原本的維度，訓練一次是100個epochs，matlab data要超過20個小時，python data則耗時超過10小時。

|  |  |
| --- | --- |
| Loss | Corr1D(Accuracy) |
| -0.989(使用matlab data) | **0.948** |
| -0.989(使用matlab data) | **0.942** |
| 111.488(使用python data) | **0.939** |
| 570.845(使用python data) | **0.854** |
| 234.956(使用python data) | **0.904** |
| 655.275(使用python data) | **0.838** |

Loss算出來之所以會差那麼多是因為這兩種data生成出來後，一個有虛數一個沒有，且生成方式導致數字的量尺不同，loss因而有差異，但train出來的結果Accuracy蠻接近的！  
  
3. **此外，我寫了variational auto-encoder(VAE)，把原本的auto-encoder換掉**，但train出來的Accuracy變低了，只有60幾趴，因為把34個speaker訓練完一次要很久，matlab data超過20個小時，python data則超過10小時，我一共只跑了三次，結果如下：

|  |  |
| --- | --- |
| Loss | Corr1D(Accuracy) |
| -0.385(使用matlab data) | **0.659** |
| -0.666(使用matlab data) | **0.682** |
| 4021.440(使用python data) | **0.662** |

  
這裡介紹我了解的**VAE：**  
VAE修正了autoencoder的缺點，autoencoder的缺點是它有keycode(代表圖片的隱藏向量)，而我們不能自己改變隱藏向量，所以在重建data時不能任意生成圖片。  
我們在編碼時會加上條件，當原始data映射到低維分布時，讓中間的core部分能依循Gaussian常態分佈。  
  
**可以看到latent vector由mean和std配出來**，公式如下：

4. Autoencoder – fine-tune：  
增加node數 -> correlation和quality增加；使用dropout -> 反之  
如果**在中間code的部分增加一點gaussian noise**，可以發現重建結果更接近原始音訊，能更穩健地恢復頻譜。  
  
5. 整體model：  
我用pytorch把整個model寫好了，在network.py那個檔案，試過shape是可以跑的，但是這裡很麻煩的部份是**要算好圖像和音檔，時間要對上才能求error**，這部分我還沒有做出來。  
現在就差把資料的時間對應關係搞清楚，寫好error function和accuracy的計算方式，便能往下一階段train。

* **Conclusion:**  
  使用corr2D當作loss function，效果確實比其他常用的loss function(PESQ)好，且透過連續圖片重建後的音檔，成功保留了說話者的語調與口氣，且與原音訊意思接近。  
  未來將會蒐集更多數據，除了臉部frame的pixel變化之話，作者提到**還可以加入情緒判讀，使重建後的音訊意思、語氣都更準確**，並建立end-to-end的structure，因為現在autoencoder和整體model是分兩階段分開train的，第一階段的最佳化並不能保證第二階段的最優，end-to-end則是把兩者直接放在一起，系统中不再有獨立的聲音模型，input丟進去後直接train、做優化，直到output。
* **Describe how this paper can inspire you in any aspects:**  
  這是我第一次把那麼多種model結合在一起，以前從沒想過**唇語真的能訓練出音檔(雖然這裡仍需要藉著pre-train autoencoder的幫助)**，這種作法對我來說是非常新奇的！  
  此外，因為這個GRID dataset的影片，音檔部分很流暢，但影像有時候會卡住，換言之，這個dataset有缺陷，遇到這種有缺陷的dataset，除了可以像作者一樣只挑選其中四個speaker，也能有其他的處理方式。例如卡住時就用上一個時刻抓到的臉，因為短期之內臉部變化不會太大，所以我們這樣做可以盡量接近原始狀況。  
  最後，**我也沒想過autoencoder可以一次接20個，語音的20ms每1ms依序接其中一個autoencoder去train**，這對我來說也是第一次接觸到。  
  這篇paper作者使用Keras去實作，他雖然有釋出部分程式碼在github上，但根本就不能跑，所以後來我們用了Pytorch照著paper的架構實作，我體會到學術界有時候不會是真的open-source，在研讀這一篇paper時也體會到做研究的難處，實在比寫作業困難太多了！如果要選擇的話，我想我絕對不會簽博。
* **Reference:**  
  http://spandh.dcs.shef.ac.uk/gridcorpus/  
  <http://www.isr.umd.edu/Labs/NSL/Downloads.html>  
  <https://www.google.com.tw/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwiB5ImbgfnbAhUJE7wKHZ22D1UQjRx6BAgBEAU&url=http%3A%2F%2Fmriquestions.com%2Ffourier-transform-ft.html&psig=AOvVaw2sjUgugprH9fFwXK7DMB8U&ust=1530366377004626>  
  <https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714>
* **Code:**1. Data preparation  
  **python prepare\_crop\_files.py  
  python prep\_data.py  
  或用matlab跑genSpec.m**2. Train autoencoder  
  **python train\_autoencoder.py**3. Train whole model  
  因為時間對應還沒算好  
  所以只打出架構，已確認model shape是對的  
  **network.py**4.其他會用到的modol或utility，另外放在model資料夾和utils資料夾