Speech Command Recognition with CNN

CS229 Final Project

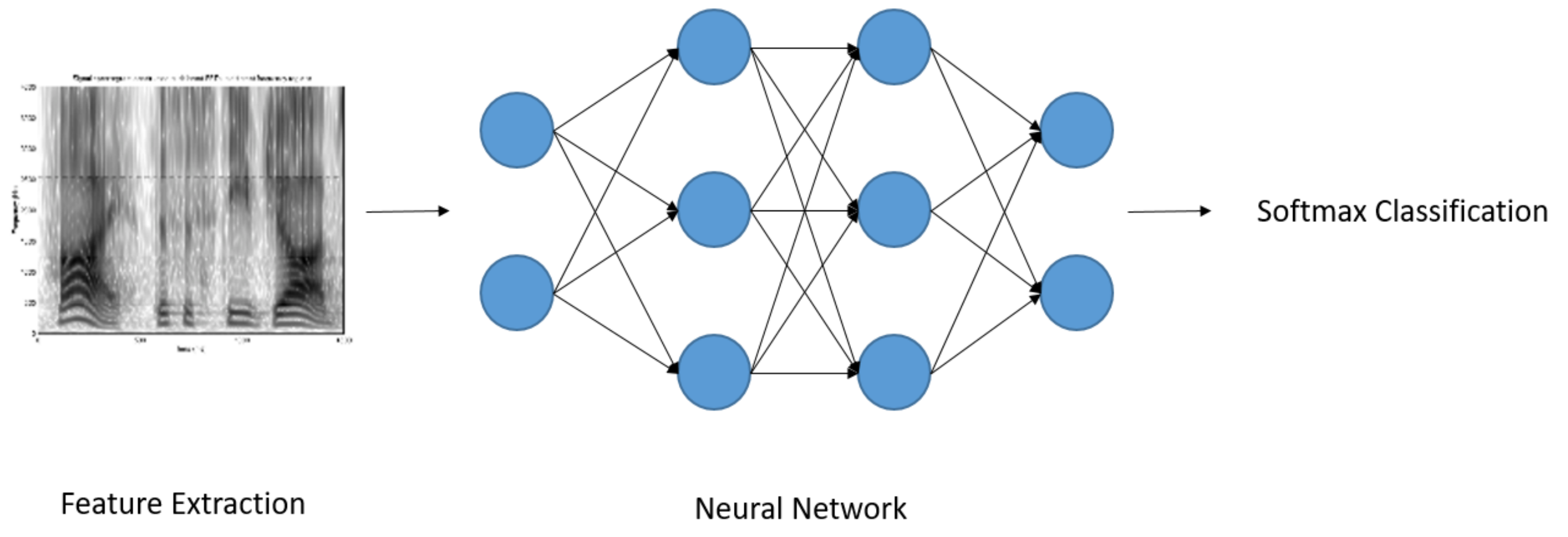
Xuejiao Li(xjli1013), Zixuan Zhou(zixuan95)

Introduction & Overview

- Thanks to the rapid development of mobile devices, speech recognition technology has become popular due to the increasing need of interaction with machines using voice commands.
- Goal of this project is building a light-weight keyword spotting system that can recognize multiple different short speech commands
- Dataset: Speech Commands Dataset provided by Google's TensorFlow and AIY teams

System Organization

- Feature Extraction with MFCC
- Neural Network model (Vanilla single layer, DNN, CNN)
- Softmax Classification

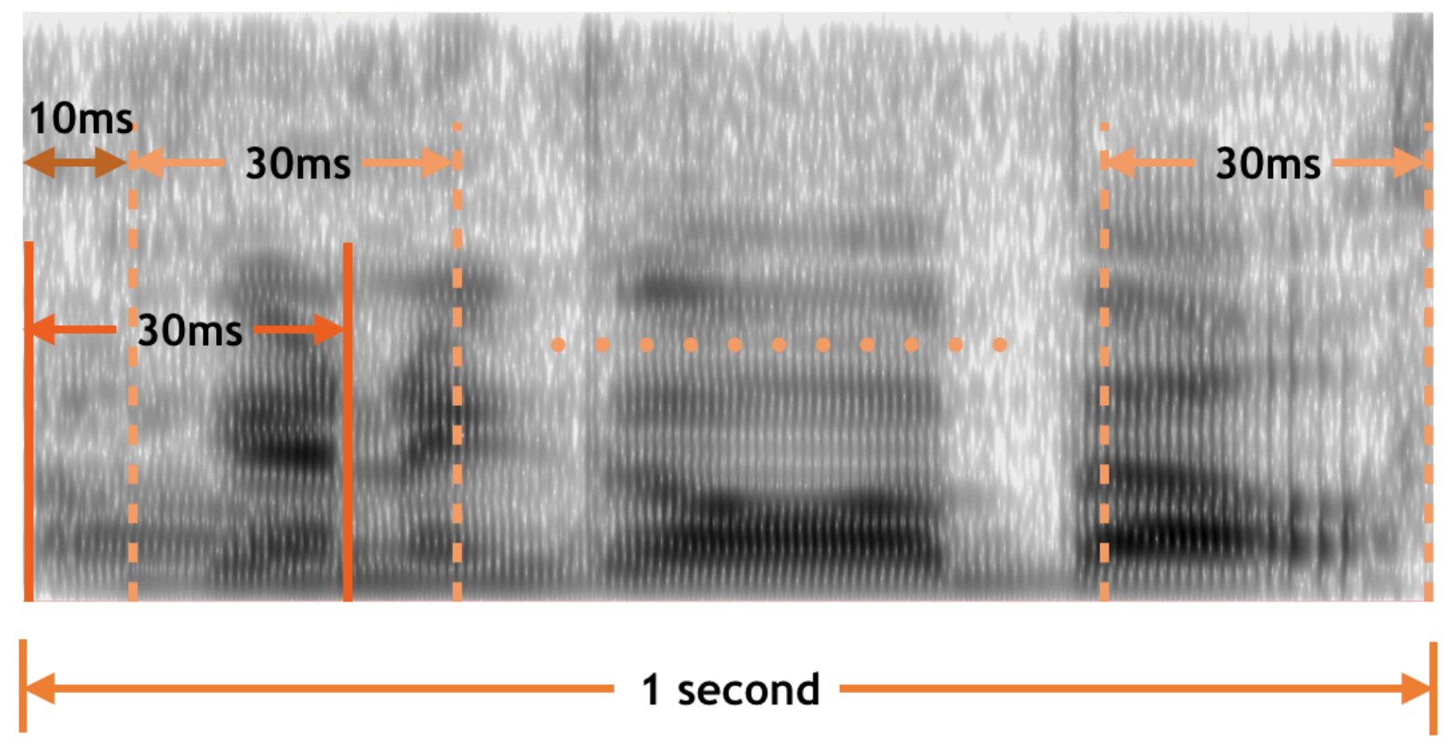


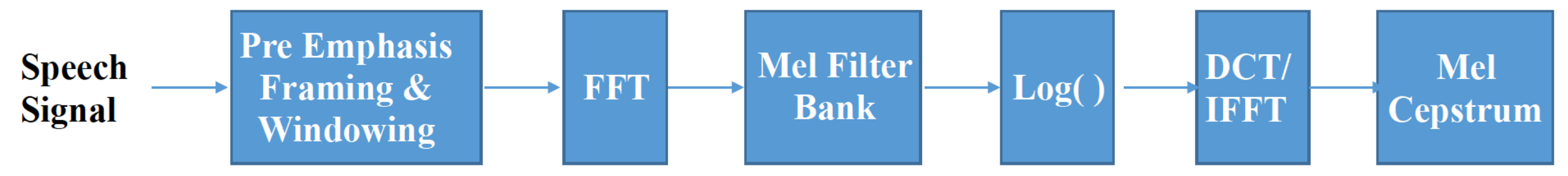
Feature Extraction

•We calculated Mel-Frequency Cepstral Coefficients (MFCC) to extract spectral features. Due to human perception experiments, Mel-Frequency analysis is employed to re-weight dimension of frequency and gain more perceptually-relevant representation of speech audio.

•Feature extraction is solved by firstly defining an analysis window, and dividing the speech signal into different time frames by shifting the window. Since audio signal sample is 1s each, we will have (1000-30)/10+1=98 time frames. After the windowing, Fast Fourier Transformation (FFT) is calculated for each frame and the logarithmic Mel-Scaled filter bank is applied to the Fourier transformed frame. The last step is to calculate Discrete Cosine Transformation (DCT) to obtain the coefficents vector.

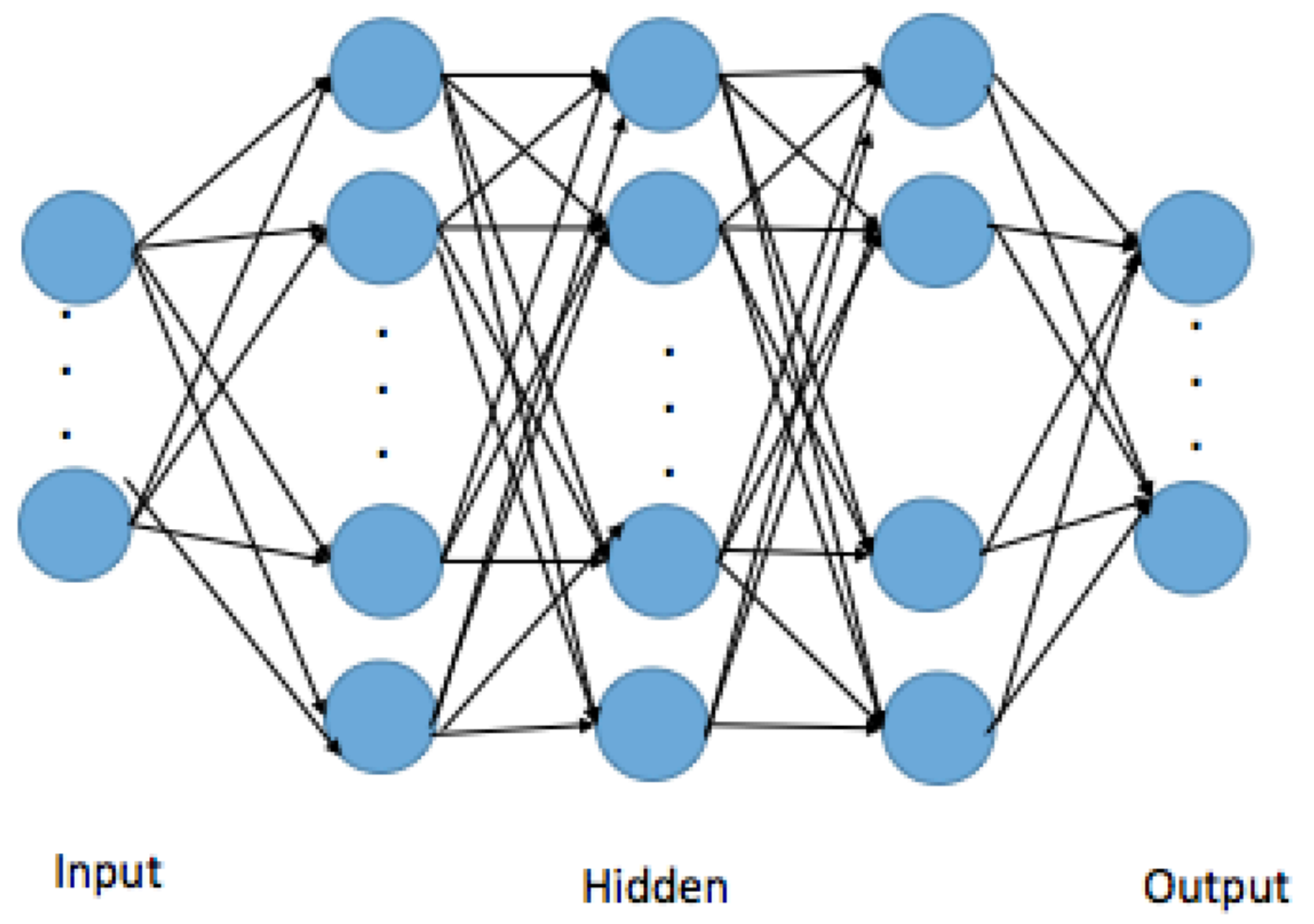
•In our project, we obtained 40-dimentional coefficents computed every 10ms over a a window of 30ms, and finally obtained a [98×40] 2D matrix desired to feed the successive neural network.





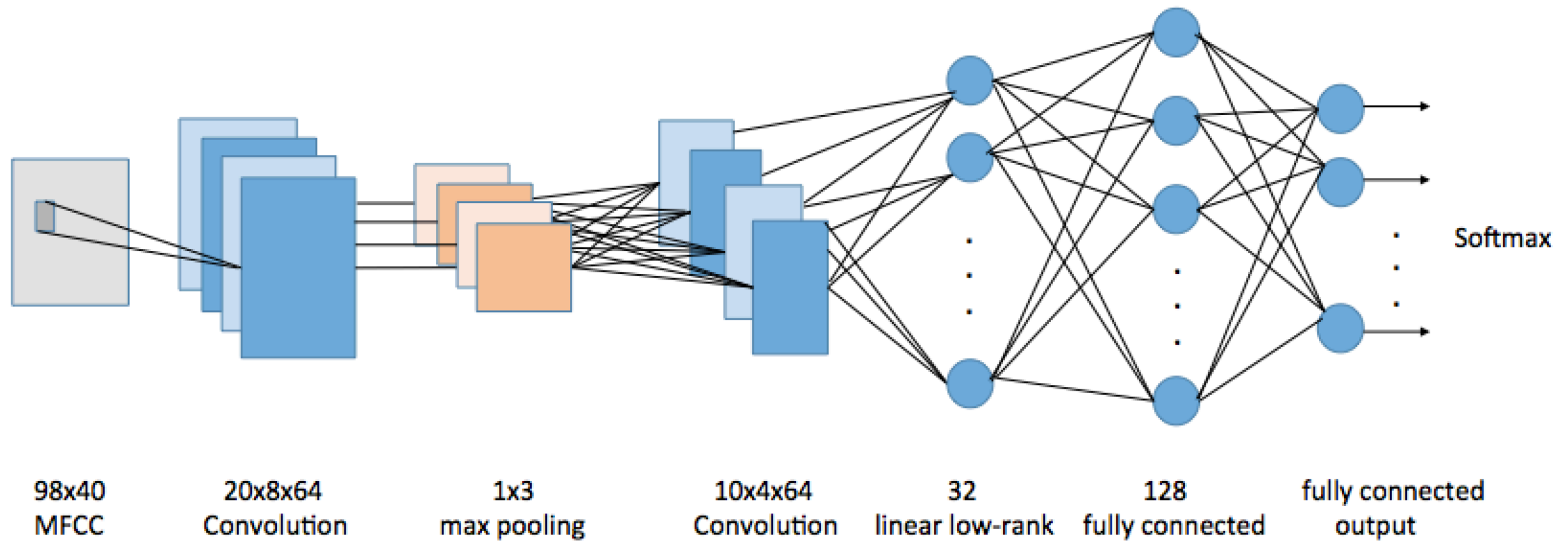
Baseline Models

- Vanilla Single Layer
  - We have firstly built a model with a single hidden fully-connected layer. This is a very simple model with just one matrix multiplication and bias.
  - As expected, it doesn't produce very accurate results, but it is very fast and simple.
- Fully Connected Deep Neural Network
  - We have built a standard feed-forward fully connected neural network with 3 hidden layers and 128 hidden nodes per layer, each computing a ReLU function of the weighted sum of the output of the previous layer.
  - Compared to Vanilla Single Layer, this model is expected to give a more accurate result at the cost of more memory footprint and higher computational cost.



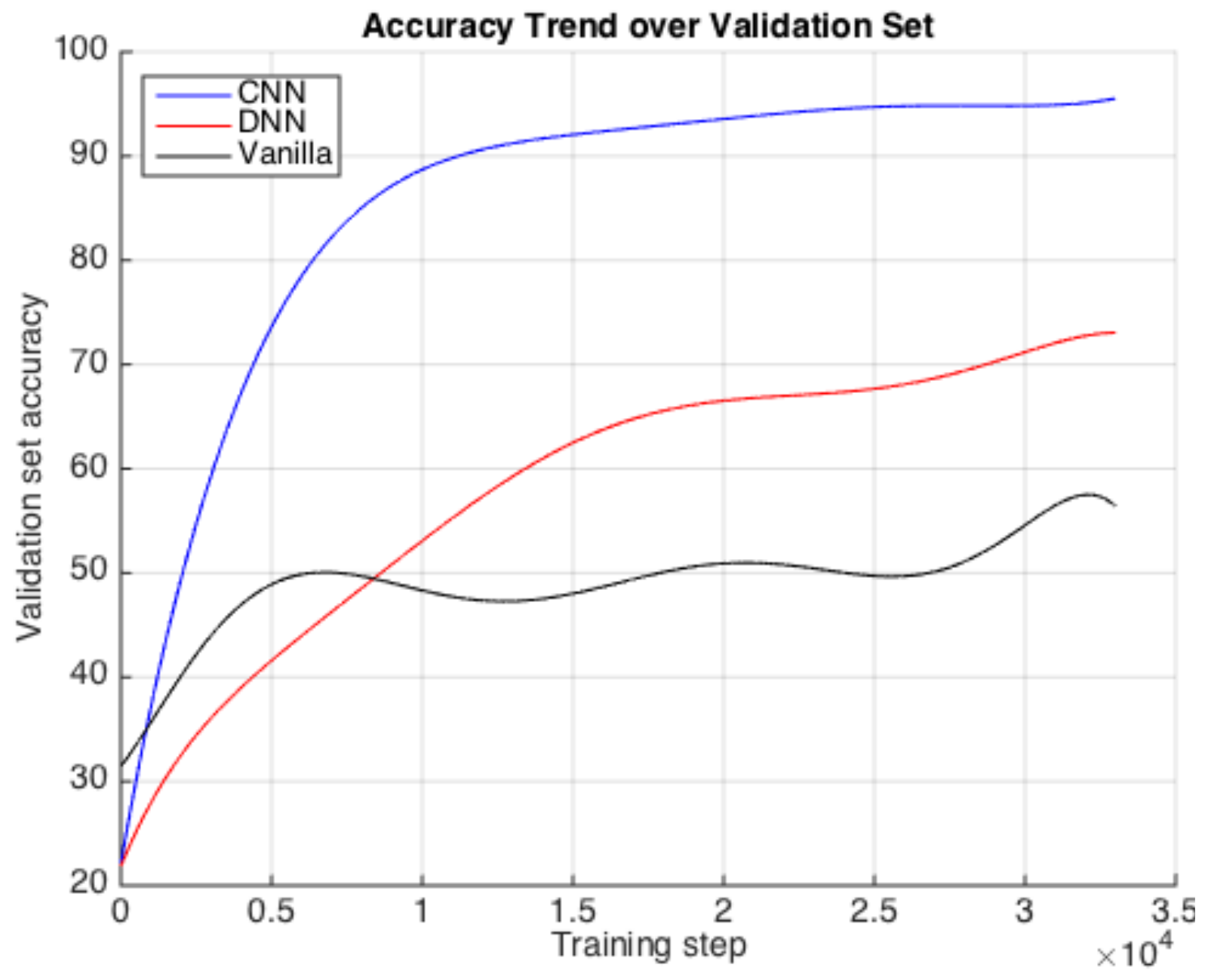
Convolutional Neural Network Model

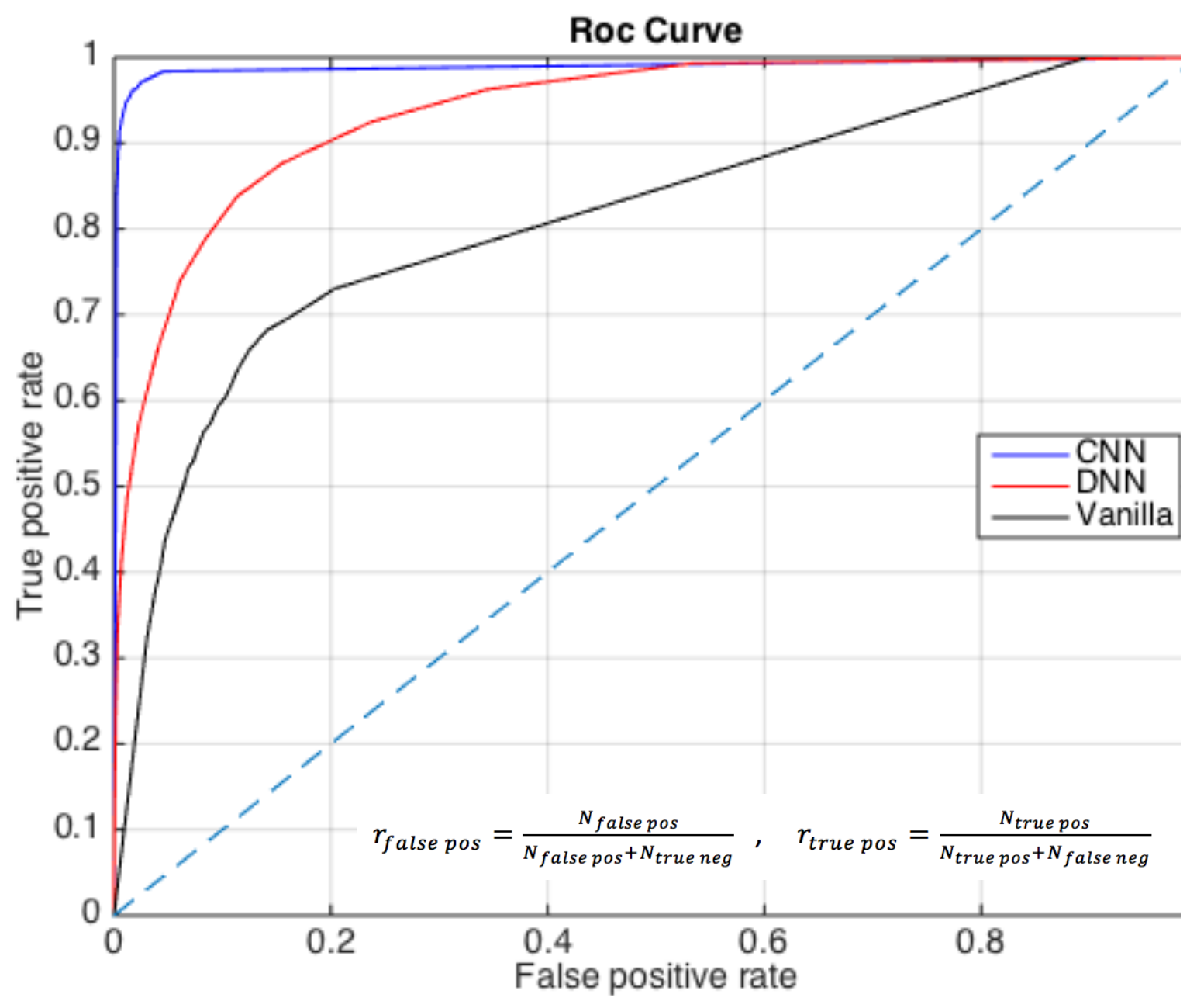
- Our CNN architecture uses two convolutional layers, one linear low-rank layer and one DNN layer.
- The first convolutional layer has a filter size of [20×8×64], which spans 2/3 of the overall input size in time. Convolutional multiplication is performed by striding the filter by stride = 1 in both time and frequency. Next, non-overlapping max-pooling in frequency only is performed, with a pooling region of 3.
- The second convolutional filter has a filter size [10×4×64] and no max-pooling is performed.
- A linear low-rank layer converting the output of the second convolutional layer into a 32-nodes output and a fully-connected DNN with 128 hidden nodes is implemented after 2 convolutional layers.



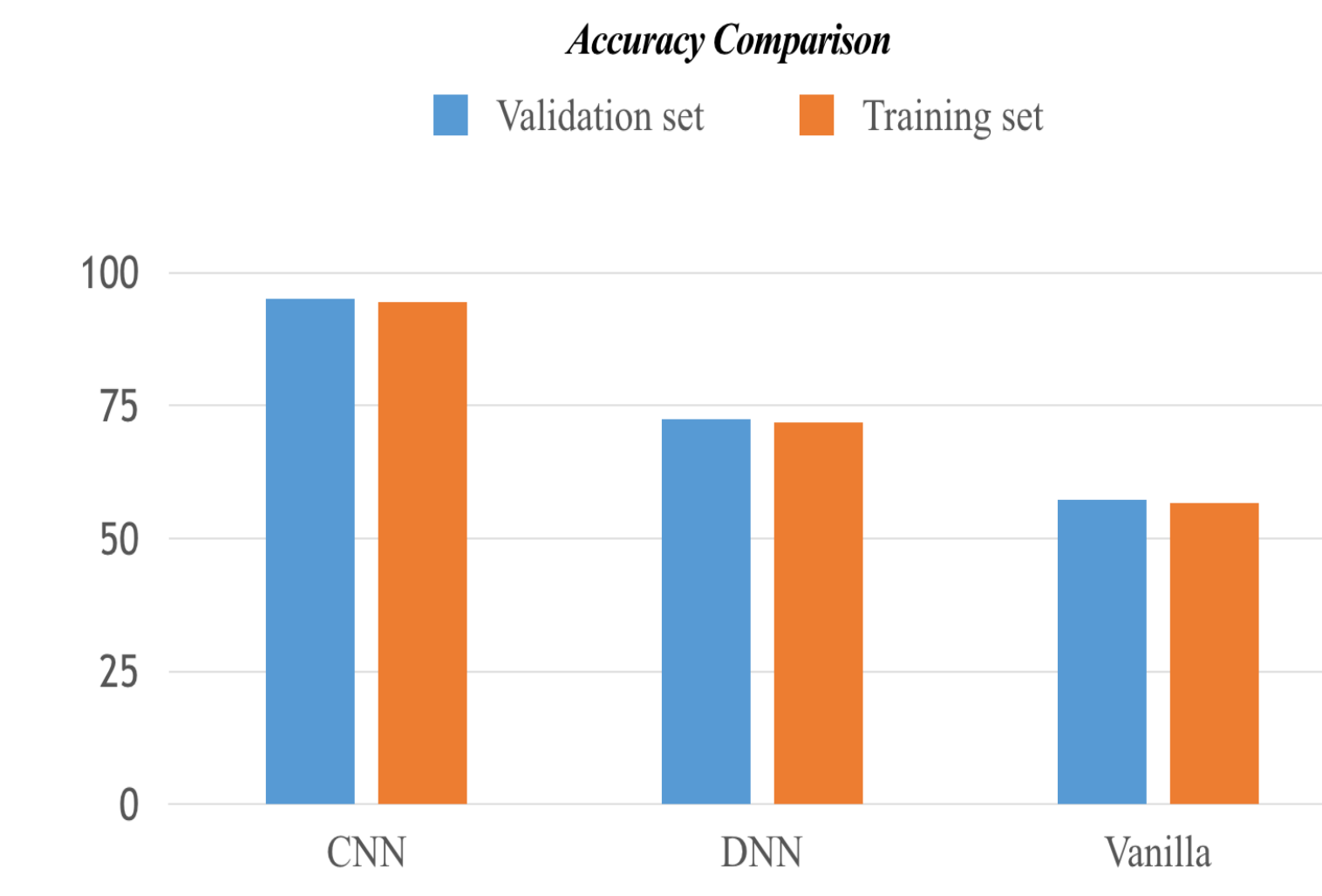
Type	M	R	N	P	Q	Params	Mult
Conv	20	8	64	1	3	10.2K	4.4M
Conv	10	4	64	1	1	164.8K	5.2M
Lin	-	-	32	-	-	65.5K	65.5K
Dnn	-	-	128	-	-	4.1K	4.1K
Softmax	-	-	6	-	-	0.7K	0.7K
Total	-	-	-	-	-	244.4K	9.9M

Results & Analysis





$r_{false\ pos} = \frac{N_{false\ pos}}{N_{false\ pos}+N_{true\ neg}}$, $r_{true\ pos} = \frac{N_{true\ pos}}{N_{true\ pos}+N_{false\ neg}}$



Test Confusion matrix for CNN

106	0	0	0	0	0
1	76	7	15	5	2
0	6	264	0	1	1
0	2	2	246	3	0
0	1	2	3	260	1
1	11	2	2	1	242

- We calculated the accuracy over validation-set every 400 steps and plot the accuracy trend (fitting) of 3 different models. It is clear that CNN model outperforms the baseline DNN and Vanilla with a higher accuracy and a smoother curve.
- ROC curve shows that CNN model achieves a much higher AUC, and gains dramatically at a very low false alarm rate, which is a desirable property for speech command system.
- The CNN model has the highest accuracy with 95.1% on validation-set and 94.5% on test-set; DNN model with 72.5% and 71.9%; Vanilla with 57.3% and 56.7%, respectively.
- The columns of the confusion matrix represent a set of samples that were predicted as "silence", "unknown", "up", "down", "left", "right" respectively. All of the entries are very small apart from the diagonal lines through the center, which indicates the CNN model makes very few mistakes.

Training Details

- We obtained the Speech Commands Dataset provided by Google's TensorFlow and AIY teams, which consists of 65,000 WAVE audio files.
- We split the data set into three part, including 80% training set, 10% validation set and 10% test set. Each subset is classified as either silence, unknown word or wanted word ("up", "down", "left", "right").
- We applied Nesterov Momentum update method with batch size equals to 100. The learning rate is 0.001 for the first 30,000 steps followed by 0.0001 towards the end to obtain both high efficiency and good convergence. The momentum value is 0.5, 0.9, 0.95, 0.99 with regard to the increasing steps.
- We applied dropout method as regularization technique with drop probability equals to 0.5.

Conclusion

We have implemented speech command systems based on Vanilla, DNN and CNN models respectively. The experiment results show that CNN model outperforms the other two baseline models and achieves 31.43% and 66.67% relative improvement with regard to DNN and Vanilla in test-accuracy.

RESEARCH POSTER PRESENTATION DESIGN © 2015
www.PosterPresentations.com