

Attention Based Models for Key-Word Spotting

Human Data Analytics Project

Riccardo Mazzieri

September 6th 2021

Overview

1 Introduction

2 Attention in Machine Learning

3 Experiments and Results

Introduction

Key-Word Spotting Task

Definition (Key-Word Spotting Task)

The Key-Word Spotting (KWS) task consists in the detection of a predetermined limited set of keywords from a stream of user utterances.

- Systems are required to run on mobile devices, which are typically constrained in terms of computational capabilities
- Tradeoff between model performance and memory/computational footprint must be optimized
- Examples of successful deep neural architectures for KWS are CNNs and RNNs.

Key-Word Spotting Task

Definition (Key-Word Spotting Task)

The Key-Word Spotting (KWS) task consists in the detection of a predetermined limited set of keywords from a stream of user utterances.

- Systems are required to run on mobile devices, which are typically constrained in terms of computational capabilities
- Tradeoff between model performance and memory/computational footprint must be optimized
- Examples of successful deep neural architectures for KWS are CNNs and RNNs.

Key-Word Spotting Task

Definition (Key-Word Spotting Task)

The Key-Word Spotting (KWS) task consists in the detection of a predetermined limited set of keywords from a stream of user utterances.

- Systems are required to run on mobile devices, which are typically constrained in terms of computational capabilities
- Tradeoff between model performance and memory/computational footprint must be optimized
- Examples of successful deep neural architectures for KWS are CNNs and RNNs.

Key-Word Spotting Task

Definition (Key-Word Spotting Task)

The Key-Word Spotting (KWS) task consists in the detection of a predetermined limited set of keywords from a stream of user utterances.

- Systems are required to run on mobile devices, which are typically constrained in terms of computational capabilities
- Tradeoff between model performance and memory/computational footprint must be optimized
- Examples of successful deep neural architectures for KWS are CNNs and RNNs.

Attention in Machine Learning

Bahdanau et al. (2014) [2]

To solve the information bottleneck problem in Encoder-Decoder networks, the authors model the Decoder's output conditional distribution as follows:

$$p(y_t \mid \{y_1, \dots, y_{t-1}\}, \mathbf{x}) = g(y_{t-1}, s_t, \mathbf{c}_t)$$

where:

$$\begin{cases} s_t = f(s_{t-1}, y_{t-1}, \mathbf{c}_t) \\ c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j \\ \alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \\ e_{tj} = \text{align}(s_{t-1}, h_j) \end{cases}$$

- *align* is an “alignment model”, which scores how well the inputs around position j and the output at position t match;
- The authors parametrize it with a single layer FFNN
- This is called **additive attention** in the literature

Bahdanau et al. (2014) [2]

To solve the information bottleneck problem in Encoder-Decoder networks, the authors model the Decoder's output conditional distribution as follows:

$$p(y_t \mid \{y_1, \dots, y_{t-1}\}, \mathbf{x}) = g(y_{t-1}, s_t, \mathbf{c}_t)$$

where:

$$\begin{cases} s_t = f(s_{t-1}, y_{t-1}, \mathbf{c}_t) \\ c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j \\ \alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \\ e_{tj} = \text{align}(s_{t-1}, h_j) \end{cases}$$

- *align* is an “alignment model”, which scores how well the inputs around position j and the output at position t match;
- The authors parametrize it with a single layer FFNN
- This is called **additive attention** in the literature

Bahdanau et al. (2014) [2]

To solve the information bottleneck problem in Encoder-Decoder networks, the authors model the Decoder's output conditional distribution as follows:

$$p(y_t \mid \{y_1, \dots, y_{t-1}\}, \mathbf{x}) = g(y_{t-1}, s_t, \mathbf{c}_t)$$

where:

$$\begin{cases} s_t = f(s_{t-1}, y_{t-1}, \mathbf{c}_t) \\ c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j \\ \alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \\ e_{tj} = \text{align}(s_{t-1}, h_j) \end{cases}$$

- *align* is an “alignment model”, which scores how well the inputs around position j and the output at position t match;
- The authors parametrize it with a single layer FFNN
- This is called **additive attention** in the literature

Bahdanau et al. (2014) [2]

To solve the information bottleneck problem in Encoder-Decoder networks, the authors model the Decoder's output conditional distribution as follows:

$$p(y_t \mid \{y_1, \dots, y_{t-1}\}, \mathbf{x}) = g(y_{t-1}, s_t, \mathbf{c}_t)$$

where:

$$\begin{cases} s_t = f(s_{t-1}, y_{t-1}, \mathbf{c}_t) \\ c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j \\ \alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})} \\ e_{tj} = \text{align}(s_{t-1}, h_j) \end{cases}$$

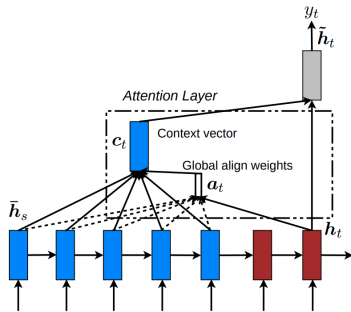
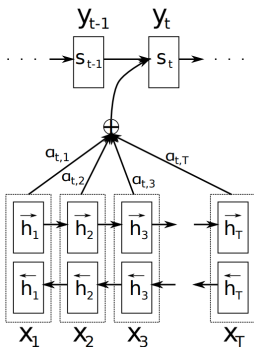
- *align* is an “alignment model”, which scores how well the inputs around position j and the output at position t match;
- The authors parametrize it with a single layer FFNN
- This is called **additive attention** in the literature

Luong et al. (2015) [4]

- Here the authors propose different ways to compute the alignment score. They introduce what is commonly called **dot-product attention**:

$$e_{tj} = \text{align}(s_t, h_j) = s_t^\top h_j$$

- This approach is faster and more memory efficient (no learnable parameters).



Vaswani et al. (2017) [5]

- One of the most influential recent papers in deep learning
- The authors introduce the Transformer (not used for the project)
- They also introduce new terminology for Attention mechanism and Multi-Head Attention layer

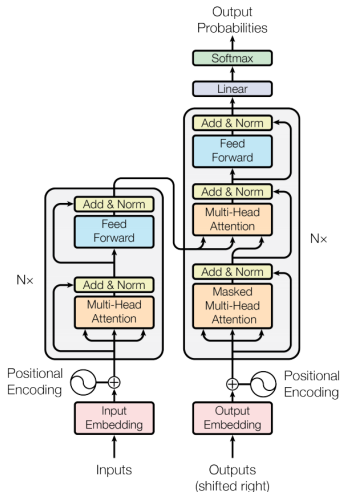


Figure: Transformer Architecture

Vaswani et al. (2017) [5]

- One of the most influential recent papers in deep learning
- The authors introduce the Transformer (not used for the project)
- They also introduce new terminology for Attention mechanism and Multi-Head Attention layer

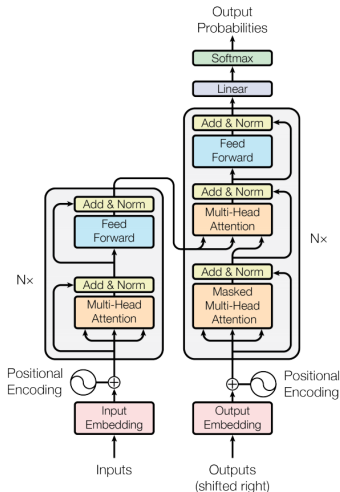


Figure: Transformer Architecture

Vaswani et al. (2017) [5]

- One of the most influential recent papers in deep learning
- The authors introduce the Transformer (not used for the project)
- They also introduce new terminology for Attention mechanism and Multi-Head Attention layer

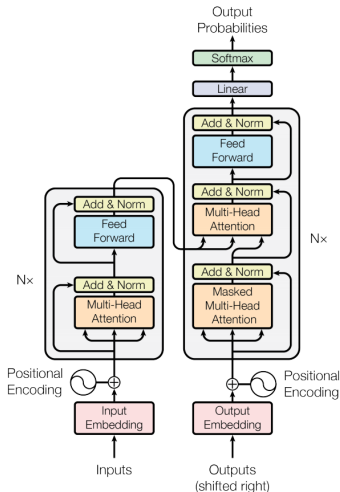


Figure: Transformer Architecture

Query, Keys and Values

The authors provide an alternative definition for Attention, which suggests an analogy with relational databases:

“An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a **compatibility function** of the query with the corresponding key.”

Query, Keys and Values

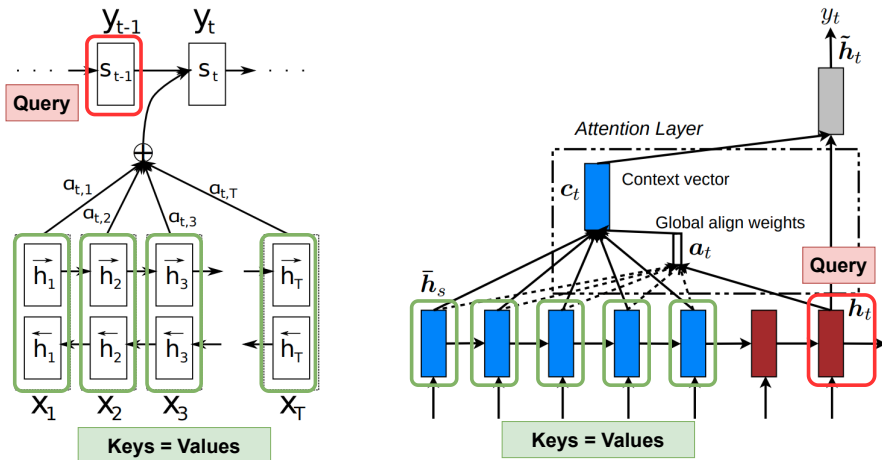
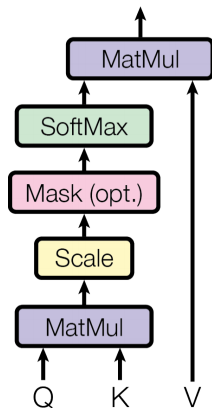


Figure: Examples of Query, Keys and Values in Additive and Dot-product attention

Scaled Dot-Product Attention

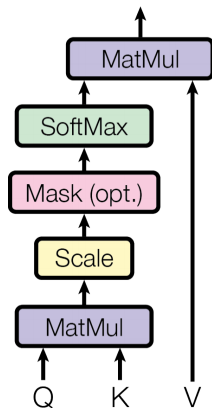


- Query vectors are the rows of a matrix Q ; In the same way, keys and values are denoted as matrices K and V respectively;
- New attention score is computed as:

$$A(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

where d_k is the dimension of the keys and query vectors (they must coincide);

Scaled Dot-Product Attention



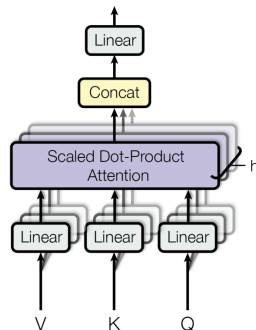
- Query vectors are the rows of a matrix Q ; In the same way, keys and values are denoted as matrices K and V respectively;
- New attention score is computed as:

$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

where d_k is the dimension of the keys and query vectors (they must coincide);

Multi-Head Attention

- Consists in computing Scaled Dot-Product attention h times in parallel (different *heads*);
- For each head, Q, K , and V are projected with a dense linear layer to dimensions d_k, d_k and d_v respectively; each head uses different projection matrices;
- Output of h heads consists in h d_v -dimensional vectors: those are concatenated and projected back with a linear dense layer:

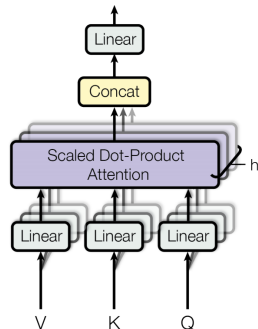


$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

$$\text{where } \text{head}_i = A \left(QW_i^Q, KW_i^K, VW_i^V \right)$$

Multi-Head Attention

- Consists in computing Scaled Dot-Product attention h times in parallel (different *heads*);
- For each head, Q, K, and V are projected with a dense linear layer to dimensions d_k, d_k and d_v respectively; each head uses different projection matrices;
- Output of h heads consists in h d_v -dimensional vectors: those are concatenated and projected back with a linear dense layer:

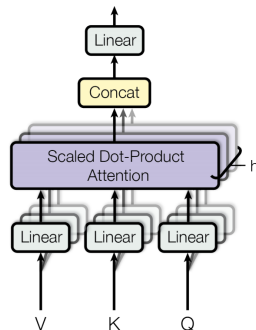


$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

$$\text{where } \text{head}_i = A(QW_i^Q, KW_i^K, VW_i^V)$$

Multi-Head Attention

- Consists in computing Scaled Dot-Product attention h times in parallel (different *heads*);
- For each head, Q, K , and V are projected with a dense linear layer to dimensions d_k, d_k and d_v respectively; each head uses different projection matrices;
- Output of h heads consists in h d_v -dimensional vectors: those are concatenated and projected back with a linear dense layer:



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

$$\text{where } \text{head}_i = A(QW_i^Q, KW_i^K, VW_i^V)$$

Self-Attention

- When Q , K and V come from the same sequence, we perform **Self-Attention**: the output is a different representation of the items associated with Q ;
- Self attention was first proposed in Cheng et al. “*Long short-term memory-networks for machine reading.*” (2016) [3]

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

Experiments and Results

The Dataset and the tasks

Google Speech Commands V2 [6]

- Total of 105829 user utterances. Each one contains one of 35 keywords.
- Each sample is a one second long .wave file with a sample-rate of 16KHz.

The Tasks

- **12kws** task: the model must discriminate between 10 different keywords, a “unknown” class and a “silence” class; In order to balance the dataset, the “unknown” samples are randomly extraced from the pool of all the remaining 25 keywords in order to have roughly the same amount of samples per class.
- **35kws** task: the model must discriminate between all the 35 different keywords;

The Dataset and the tasks

Google Speech Commands V2 [6]

- Total of 105829 user utterances. Each one contains one of 35 keywords.
- Each sample is a one second long .wave file with a sample-rate of 16KHz.

The Tasks

- **12kws** task: the model must discriminate between 10 different keywords, a “unknown” class and a “silence” class; In order to balance the dataset, the “unknown” samples are randomly extraced from the pool of all the remaining 25 keywords in order to have roughly the same amount of samples per class.
- **35kws** task: the model must discriminate between all the 35 different keywords;

Splits, features and Data augmentation

- For each task, the dataset is split in 80% for training, 10% for validation and 10% for testing.
- The actual features used as input for the models are 40 dimensional Mel Frequency Cepstral Coefficients (MFCC). Those are obtained by using a window size of 25ms and a hop size of 10ms. → Each audio sample is converted in a 98×40 feature matrix.
- Training set undergoes data augmentation process during training. Validation and Test remain fixed.

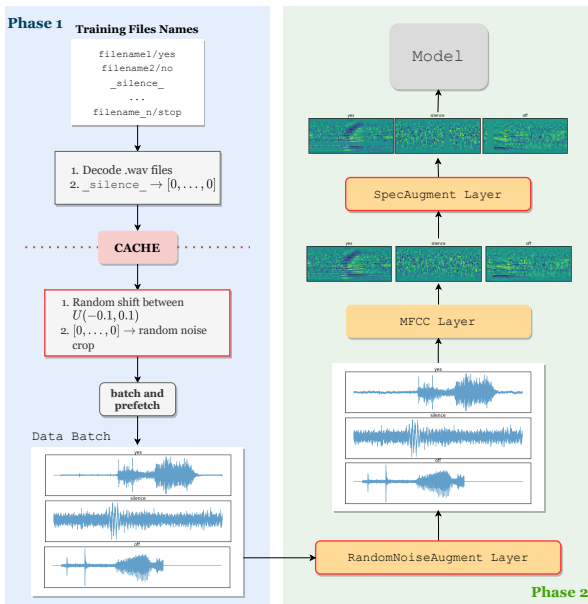
Splits, features and Data augmentation

- For each task, the dataset is split in 80% for training, 10% for validation and 10% for testing.
- The actual features used as input for the models are 40 dimensional Mel Frequency Cepstral Coefficients (MFCC). Those are obtained by using a window size of 25ms and a hop size of 10ms. → Each audio sample is converted in a 98×40 feature matrix.
- Training set undergoes data augmentation process during training. Validation and Test remain fixed.

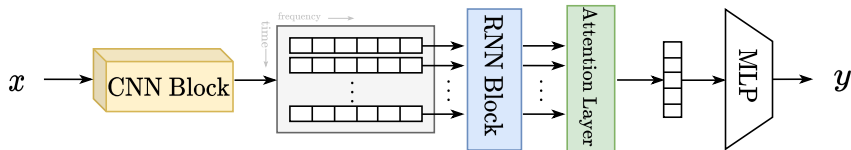
Splits, features and Data augmentation

- For each task, the dataset is split in 80% for training, 10% for validation and 10% for testing.
- The actual features used as input for the models are 40 dimensional Mel Frequency Cepstral Coefficients (MFCC). Those are obtained by using a window size of 25ms and a hop size of 10ms. → Each audio sample is converted in a 98×40 feature matrix.
- Training set undergoes data augmentation process during training. Validation and Test remain fixed.

Input Pipeline

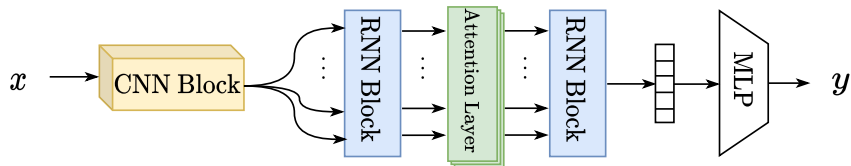


Att-RNN (de Andrade et al. (2018) [1])



layer type	n_{RNN}	n_{MLP}	n_{F}	F_{w}	F_{h}	Act.
Conv			10	5	1	ReLu
Conv			1	5	1	ReLu
RNN (LSTM)	128					tanh
RNN(LSTM)	128					tanh
Attention						
Dense		64				ReLu
Dense		m				Softmax

SQAtt-RNN



layer type	nRNN	nMLP	nF	Fw	Fh	Act.
Conv			10	5	1	ReLu
Conv			1	5	1	ReLu
RNN (GRU)	128					tanh
RNN (GRU)	128					tanh
QAttention						
RNN (GRU)	64					tanh
Dense		64				ReLu
Dense		m				Softmax

Using Multi-Head Attention

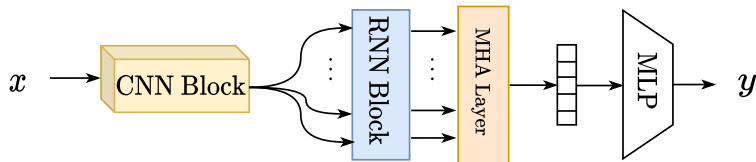


Figure: MHAtt-RNN k , k is the number of heads. Proposed in Rybakov et al. (2020)

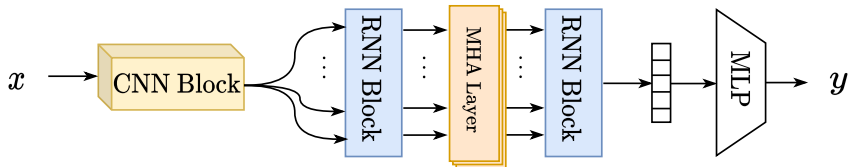


Figure: SQMHAtt-RNN k , k is the number of heads.

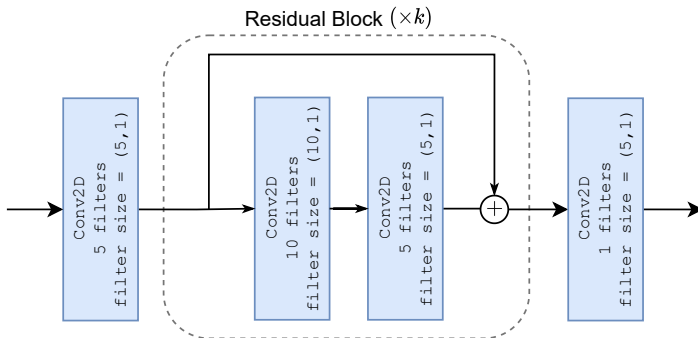
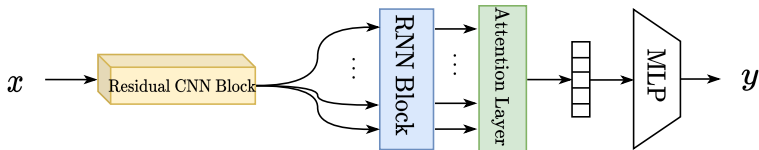


Figure: Residual CNN Block

Results

- Each model was trained for 30 epochs, using the Adam optimizer and shrinking the learning rate of a factor of 0.1 with a patience of 2.

Table: Results in term of top-1 accuracy for both tasks.

Model	Param.	Acc. 12kw	Acc. 35kw
SimpleAtt	84 K	92.9%	92.8%
Att-RNN	180 K	94.7%	95.2%
SQAtt-RNN	169 K	94.9%	94.6%
SQAtt-noCNN	169 K	94.8%	94.8%
MHAtt-RNN2	208 K	95.1%	94.8%
MHAtt-RNN3	241 K	95.2%	95.3%
MHAtt-RNN4	274 K	94.4%	95.3%
MHAtt-RNN5	307 K	95.5%	95.1%
SQMHAAtt-RNN2	235 K	95.3%	95.4%
SQMHAAtt-RNN3	268 K	95.4%	94.4%
SQMHAAtt-RNN4	301 K	94.7%	94.3%
SQMHAAtt-RNN5	334 K	95.7%	95.0%
Res-AttRNN3	182 K	95.1%	94.6%
Res-AttRNN4	183 K	94.4%	94.7%
Res-AttRNN5	184 K	94.0%	94.8%

Accuracy VS Parameters

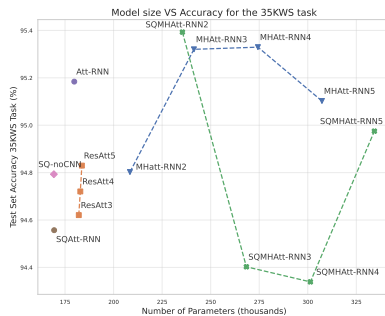
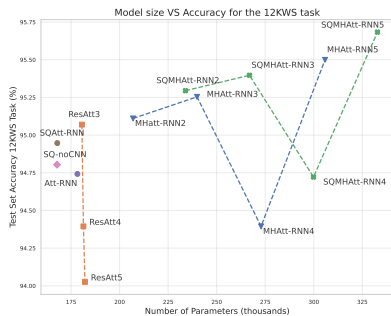
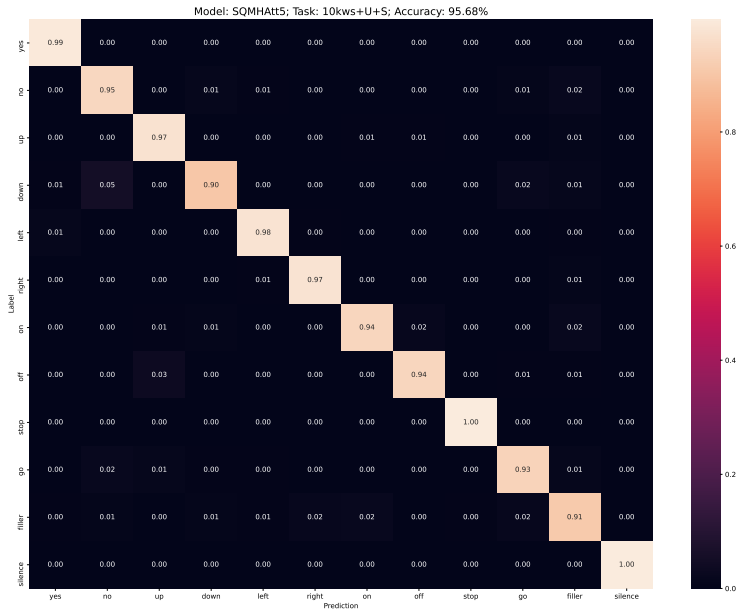
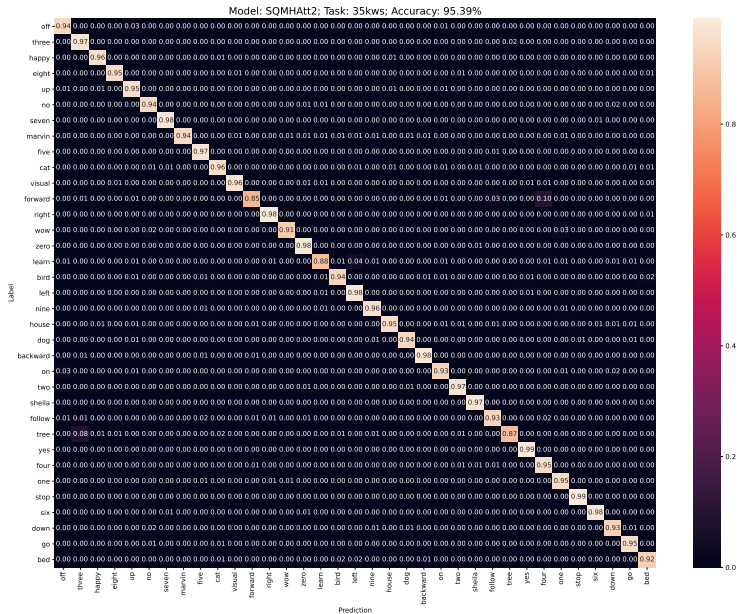


Figure: Models size vs. test set accuracy, for the 12kws (left) and 35kws task (right).

Confusion Matrix 12kws Task (Best model)



Confusion Matrix 35kws Task (Best model)



Attention Scores

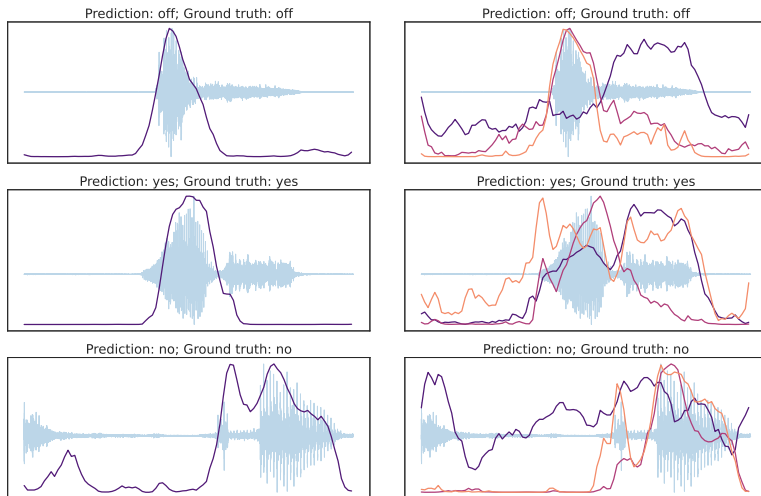


Figure: Comparison between attention scores from Att-RNN model (left) and MHAtt-RNN3 (right), on the words *off*, *yes* and *no*.

Conclusions and future work

- Multi head attention increases results in accuracy, even if it comes at the price of higher parameter count. In our case, it is still manageable since we are using just one MHA layer with not too many heads.
- More complex Residual CNN block didn't provide consistently better results: more investigations could be made on the design of the residual block
- More evaluation tests should be made (e.g. streaming tests)
- To have more statistically relevant results, one should retrain each model multiple times, and take an average of the test set accuracies, together with a confidence interval.

Conclusions and future work

- Multi head attention increases results in accuracy, even if it comes at the price of higher parameter count. In our case, it is still manageable since we are using just one MHA layer with not too many heads.
- More complex Residual CNN block didn't provide consistently better results: more investigations could be made on the design of the residual block
- More evaluation tests should be made (e.g. streaming tests)
- To have more statistically relevant results, one should retrain each model multiple times, and take an average of the test set accuracies, together with a confidence interval.

Conclusions and future work







- Multi head attention increases results in accuracy, even if it comes at the price of higher parameter count. In our case, it is still manageable since we are using just one MHA layer with not too many heads.
- More complex Residual CNN block didn't provide consistently better results: more investigations could be made on the design of the residual block
- More evaluation tests should be made (e.g. streaming tests)
- To have more statistically relevant results, one should retrain each model multiple times, and take an average of the test set accuracies, together with a confidence interval.

Conclusions and future work

- Multi head attention increases results in accuracy, even if it comes at the price of higher parameter count. In our case, it is still manageable since we are using just one MHA layer with not too many heads.
- More complex Residual CNN block didn't provide consistently better results: more investigations could be made on the design of the residual block
- More evaluation tests should be made (e.g. streaming tests)
- To have more statistically relevant results, one should retrain each model multiple times, and take an average of the test set accuracies, together with a confidence interval.

Thank you

References

-  Douglas Coimbra de Andrade et al. “A neural attention model for speech command recognition”. In: *arXiv preprint arXiv:1808.08929* (2018).
-  Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).
-  Jianpeng Cheng, Li Dong, and Mirella Lapata. “Long Short-Term Memory-Networks for Machine Reading”. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, 2016, pp. 551–561.
-  Thang Luong, Hieu Pham, and Christopher D. Manning. “Effective Approaches to Attention-based Neural Machine Translation”. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, 2015, pp. 1412–1421.
-  Ashish Vaswani et al. “Attention is All You Need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Long Beach, California, USA, 2017, pp. 6000–6010.
-  Pete Warden. “Speech commands: A dataset for limited-vocabulary speech recognition”. In: *arXiv preprint arXiv:1804.03209* (2018).