Data Note

- In the dataset, a single pair composed of:
  INPUT: obfuscated text such as **“fgxkibqc…fnjgrieb”**, without any space
  LABEL: totally 12 classes, distribution shown as below chart.

Class	0	1	2	3	4	5	6	7	8	9	10	11
Distribution	8.67%	22.5%	3.6%	9.5%	5.6%	5.5%	10.2%	12.3%	8.8%	2.4%	7.4%	3.4%

  I can’t share the dataset to public yet, but I believe similar tasks can share the experiment results here.

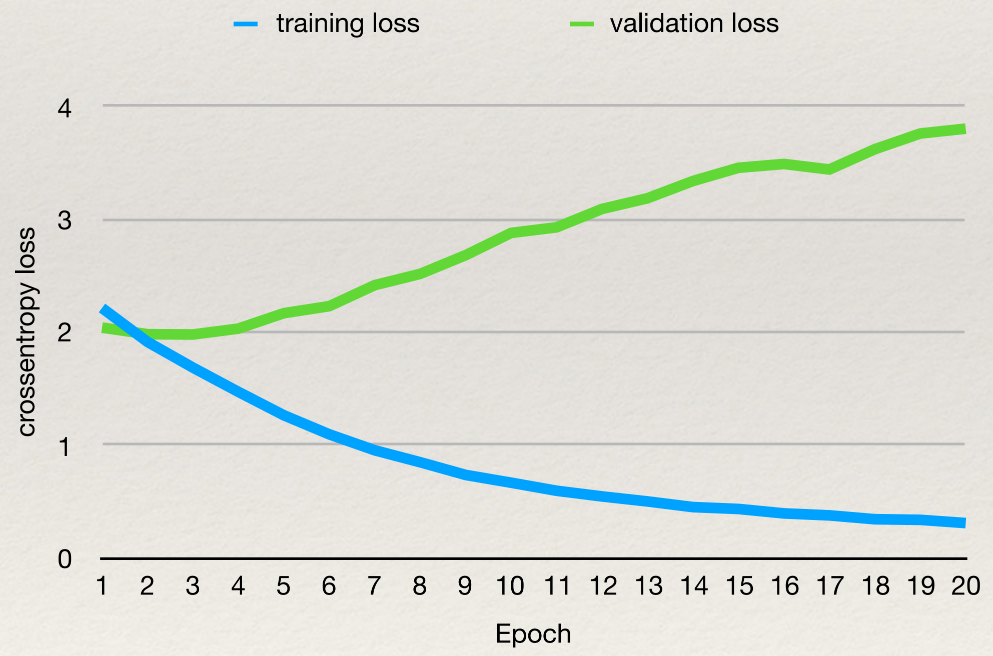
“fgxkibqc…fnjgrieb” → Array([6, 7, 24, 11, 9,…, 5, 2, <u>0, …, 0</u>]) → **Embedding Layer**

padding zero to the maximum length in the dataset.

- The input Array[Integer] is expected to be stored as numpy memmap in my implementation.
- 20% of standalone validation set is split to judge the performance.

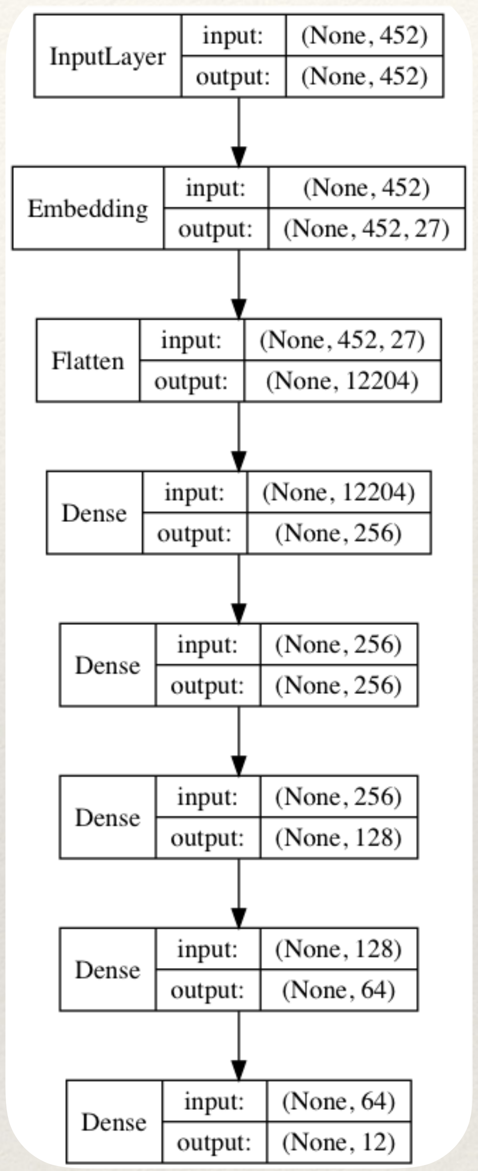
1_Baseline Model

- Expanding the full sequence vectors, pumped all in to a fully connected neural networks as a simple baseline. The neural network
- Best standalone validation accuracy: 32.45% (epoch 4)
- Overfitting start after epoch 4



Class	0	1	2	3	4	5	6	7	8	9	10	11
Precision	0.148	0.377	0.207	0.282	0.413	0.556	0.232	0.310	0.579	0.000	0.190	0.293
Recall	0.036	0.205	0.109	0.336	0.333	0.338	0.487	0.410	0.465	0.000	0.187	0.043

* All prediction based on best model (lowest validation loss), overall validation accuracy = 32.45%

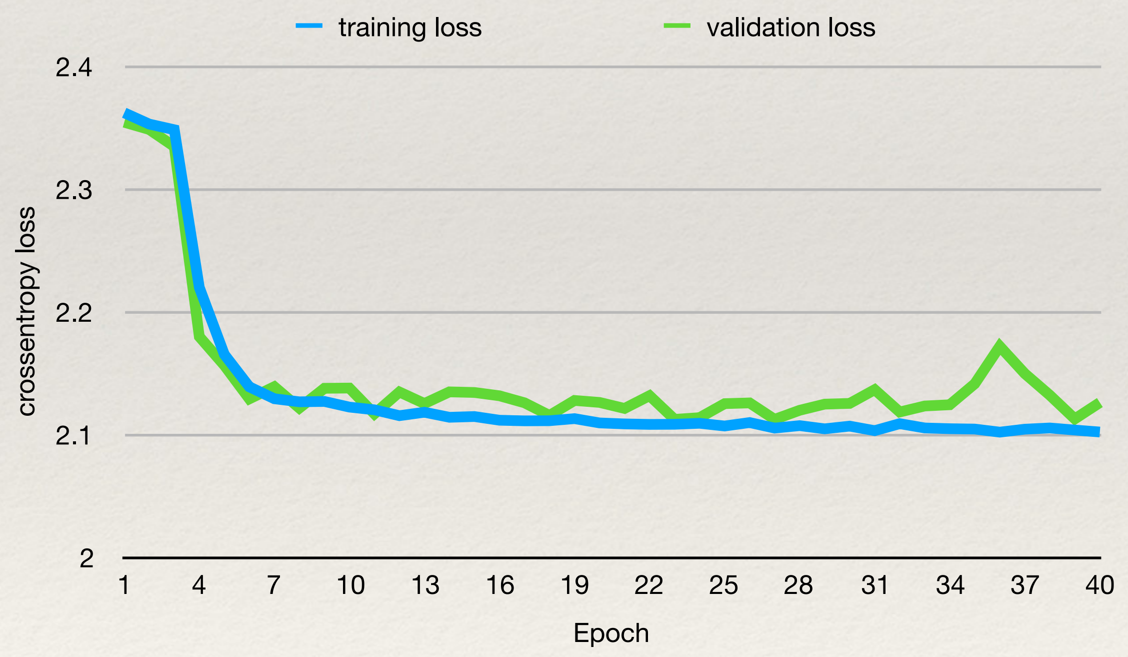


InputLayer | input: (None, 452) | output: (None, 452)
Embedding | input: (None, 452) | output: (None, 452, 27)
Flatten | input: (None, 452, 27) | output: (None, 12204)
Dense | input: (None, 12204) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 128)
Dense | input: (None, 128) | output: (None, 64)
Dense | input: (None, 64) | output: (None, 12)

Activation: LeakyReLU
Last Activ.: softmax
Loss: crossentropy
Optimizer: Adam
Batch size: 100
Dropout: 0.2
* Details please refer to 1_Baseline.py

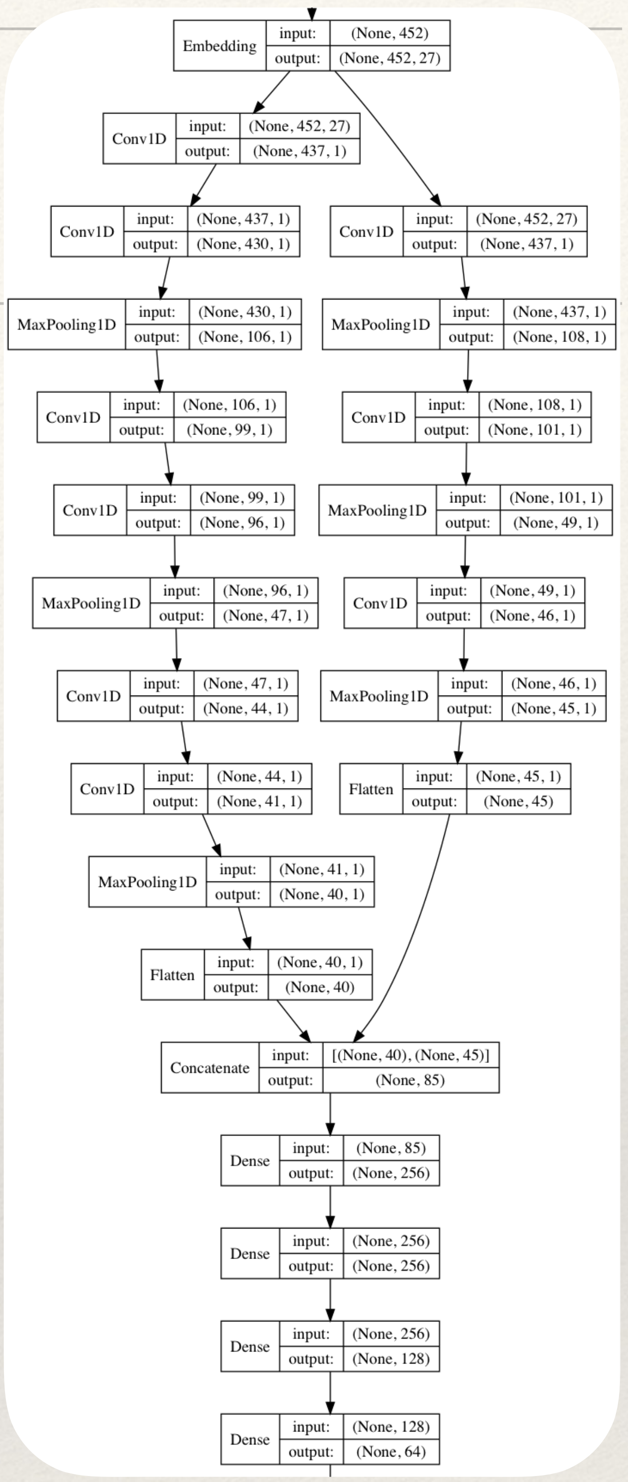
2_CNN Model

- Hypothesis here:
  1. Convolution serves as a special mask to quantify a sequence characters, here I choose 16.
  2. MaxPooling serves as choosing and filtering only the most possible "words".
- According to above, right branch here assumes 16 characters form a words, 8 most possible words make a sentence, 4 sentences reside valuable info.
- Left branch mimics Pyramid-CNN which has proved to be affective dealing with character-level classifications. [8]
- Best standalone validation accuracy: 23.93% (epoch 23)



Class	0	1	2	3	4	5	6	7	8	9	10	11
Precision	0.000	0.389	0.000	0.000	0.000	0.000	0.372	0.183	0.919	0.000	0.000	0.000
Recall	0.000	0.084	0.000	0.000	0.000	0.000	0.262	0.997	0.384	0.000	0.000	0.000

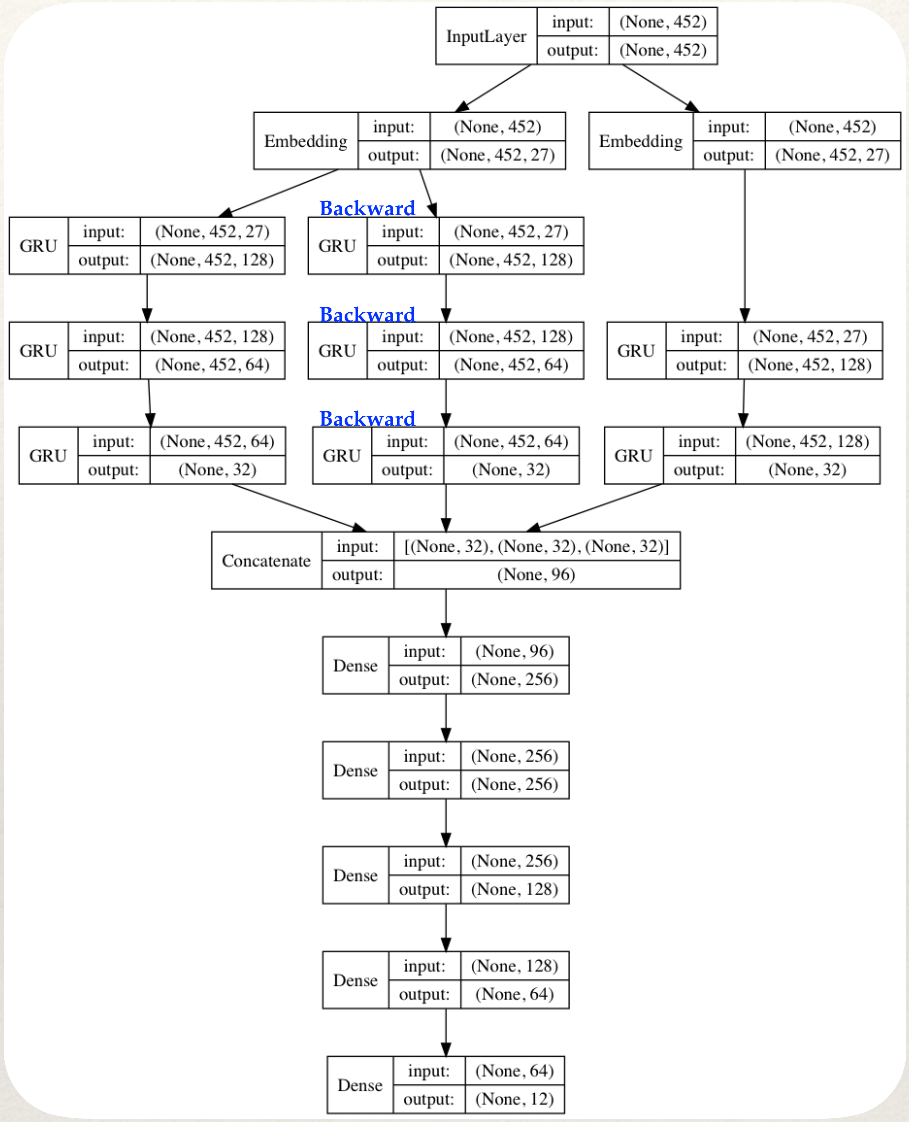
* All prediction based on best model (lowest validation loss), overall validation accuracy = 23.93%

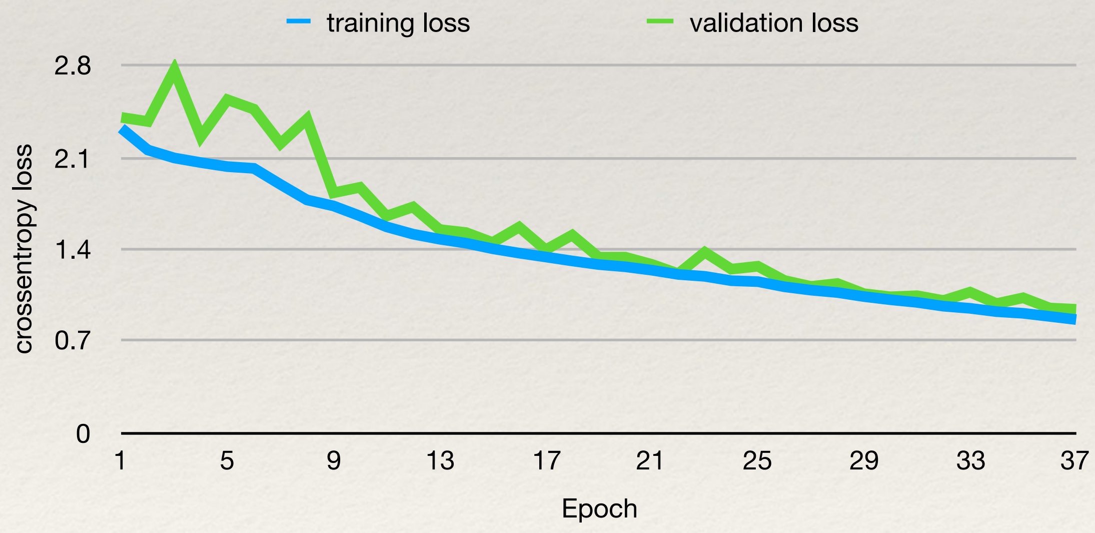

Embedding input: (None, 452) output: (None, 452, 27)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
Conv1D input: (None, 437, 1) output: (None, 430, 1)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
MaxPooling1D input: (None, 430, 1) output: (None, 106, 1)
MaxPooling1D input: (None, 437, 1) output: (None, 108, 1)
Conv1D input: (None, 106, 1) output: (None, 99, 1)
Conv1D input: (None, 108, 1) output: (None, 101, 1)
Conv1D input: (None, 99, 1) output: (None, 96, 1)
MaxPooling1D input: (None, 101, 1) output: (None, 49, 1)
MaxPooling1D input: (None, 96, 1) output: (None, 47, 1)
Conv1D input: (None, 49, 1) output: (None, 46, 1)
Conv1D input: (None, 47, 1) output: (None, 44, 1)
MaxPooling1D input: (None, 46, 1) output: (None, 45, 1)
Conv1D input: (None, 44, 1) output: (None, 41, 1)
Flatten input: (None, 45, 1) output: (None, 45)
MaxPooling1D input: (None, 41, 1) output: (None, 40, 1)
Flatten input: (None, 40, 1) output: (None, 40)
Concatenate input: [(None, 40), (None, 45)] output: (None, 85)
Dense input: (None, 85) output: (None, 256)
Dense input: (None, 256) output: (None, 256)
Dense input: (None, 256) output: (None, 128)
Dense input: (None, 128) output: (None, 64)

CNN Activation: ReLU
DNNActivation: LeakyReLU
Last Activ.: softmax
Loss: crossentropy
Optimizer: Adam
Batch size: 100
Dropout: 0.2
* Structure details please refer to 2_CNN.py

3_GRU Model

- Tested and validated that LSTM has same performance as GRU, choosing GRU due to its efficiency. Stacking GRU together to compose Pyramid-GRU, which proved to be powerful with seq2seq problem.
- A small Pyramid-GRU on the right validated to be essential to boost the accuracy up. And its embedding must be separate to interpret characters differently.
- Bidirectional GRU, the additional Backward branch here, gain no performance margin at all. But it only prove the backward reading is not informative in this specific obfuscation only, which might still works in another kind of obfuscation.
- Best standalone validation accuracy: 73.95% (epoch 32)
- Validation loss seems very promising. Would be valuable to keep training to test the real limit.


InputLayer | input: (None, 452) | output: (None, 452)
Embedding | input: (None, 452) | output: (None, 452, 27)
Embedding | input: (None, 452) | output: (None, 452, 27)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
Backward
GRU | input: (None, 452, 27) | output: (None, 452, 128)
GRU | input: (None, 452, 128) | output: (None, 452, 64)
Backward
GRU | input: (None, 452, 128) | output: (None, 452, 64)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
GRU | input: (None, 452, 64) | output: (None, 32)
Backward
GRU | input: (None, 452, 64) | output: (None, 32)
GRU | input: (None, 452, 128) | output: (None, 32)
Concatenate | input: [(None, 32), (None, 32), (None, 32)] | output: (None, 96)
Dense | input: (None, 96) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 128)
Dense | input: (None, 128) | output: (None, 64)
Dense | input: (None, 64) | output: (None, 12)



GRU Activation: tanh
DNN Activation: LeakyReLU
Last Activ.: softmax
Loss: crossentropy
Optimizer: Adam
Batch size: 100
Dropout: 0.2

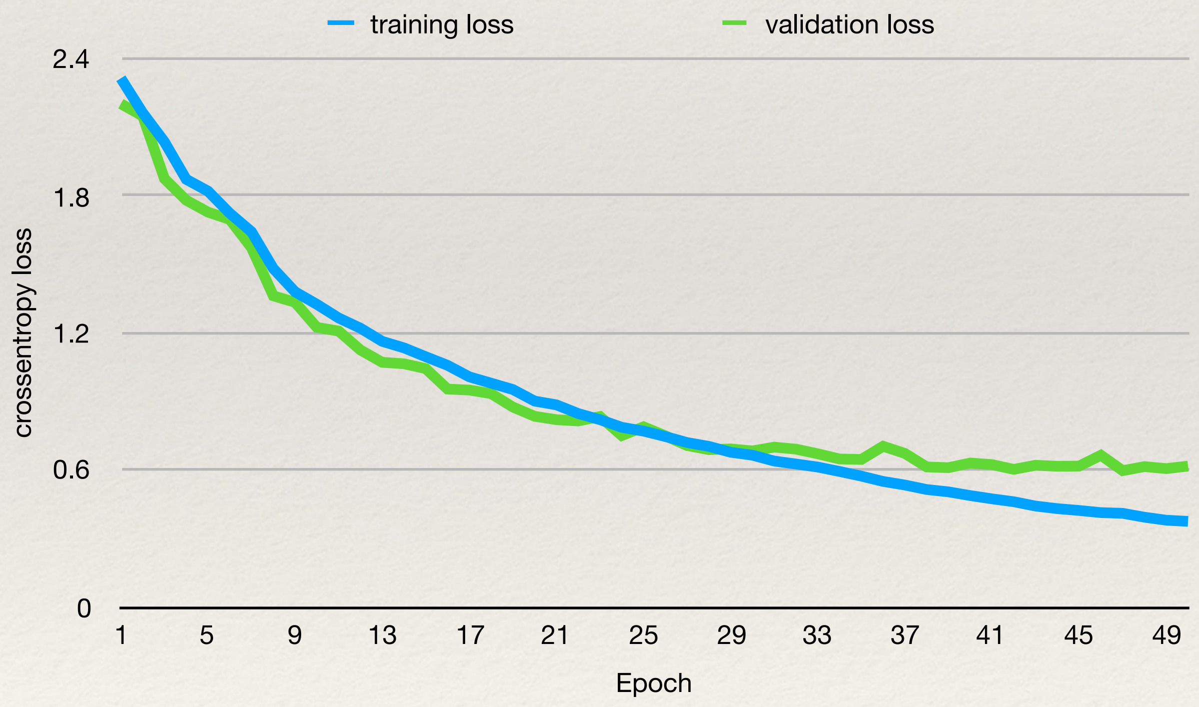
* Structure details please refer to 3_GRU.py

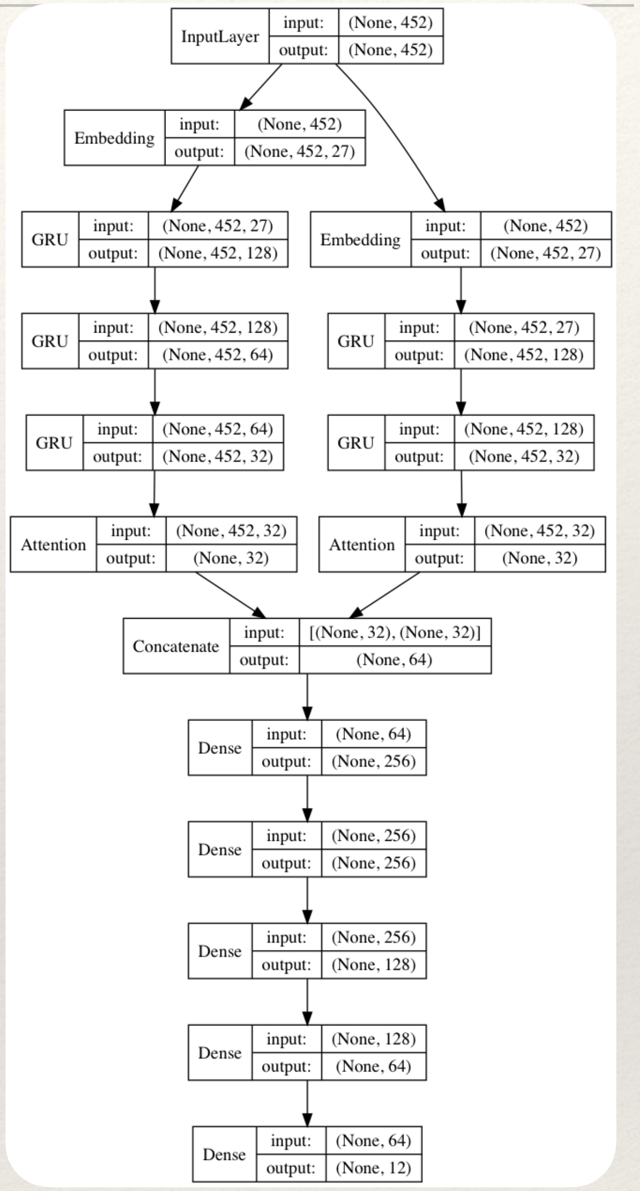
Class	0	1	2	3	4	5	6	7	8	9	10	11
Precision	0.815	0.697	0.766	0.729	0.891	0.898	0.695	0.746	0.858	0.760	0.620	0.508
Recall	0.682	0.757	0.562	0.727	0.741	0.869	0.764	0.812	0.797	0.477	0.622	0.657

* All prediction based on best model (lowest validation loss), overall validation accuracy = 73.95%

4_GRU_Attention Model

- The attention used here is Hierarchical Attention Networks proposed by Zichao Yang et. al. [9]
- The attention proved to be a big breakthrough. And the pyramid-GRU (stacking GRU that still returning time sequences) proved to be critical too. However, the model can be further simplified as the next page shown.
- Best standalone validation accuracy: 81.59% (epoch 47)



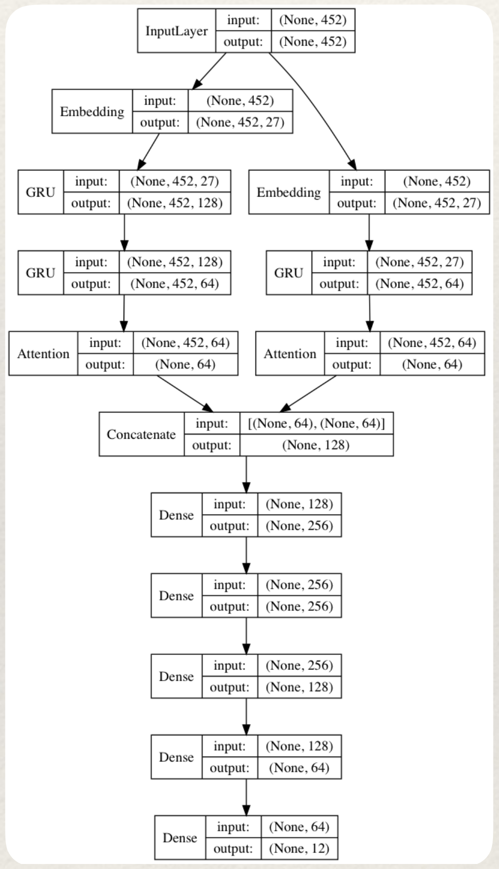

InputLayer input: (None, 452) output: (None, 452)
Embedding input: (None, 452) output: (None, 452, 27)
GRU input: (None, 452, 27) output: (None, 452, 128)
Embedding input: (None, 452) output: (None, 452, 27)
GRU input: (None, 452, 128) output: (None, 452, 64)
GRU input: (None, 452, 27) output: (None, 452, 128)
GRU input: (None, 452, 64) output: (None, 452, 32)
GRU input: (None, 452, 128) output: (None, 452, 32)
Attention input: (None, 452, 32) output: (None, 32)
Attention input: (None, 452, 32) output: (None, 32)
Concatenate input: [(None, 32), (None, 32)] output: (None, 64)
Dense input: (None, 64) output: (None, 256)
Dense input: (None, 256) output: (None, 256)
Dense input: (None, 256) output: (None, 128)
Dense input: (None, 128) output: (None, 64)
Dense input: (None, 64) output: (None, 12)

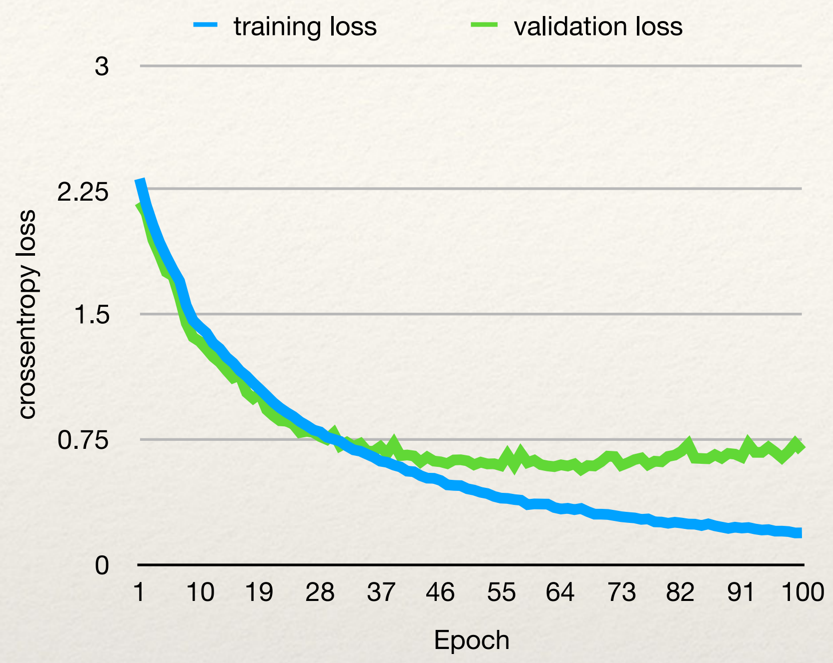
GRU Activation: tanh
DNN Activation: LeakyReLU
Last Activ.: softmax
Loss: crossentropy
Optimizer: Adam
Batch size: 100
Dropout: 0.2
* Structure details please refer to 4_GRU_Attention.py

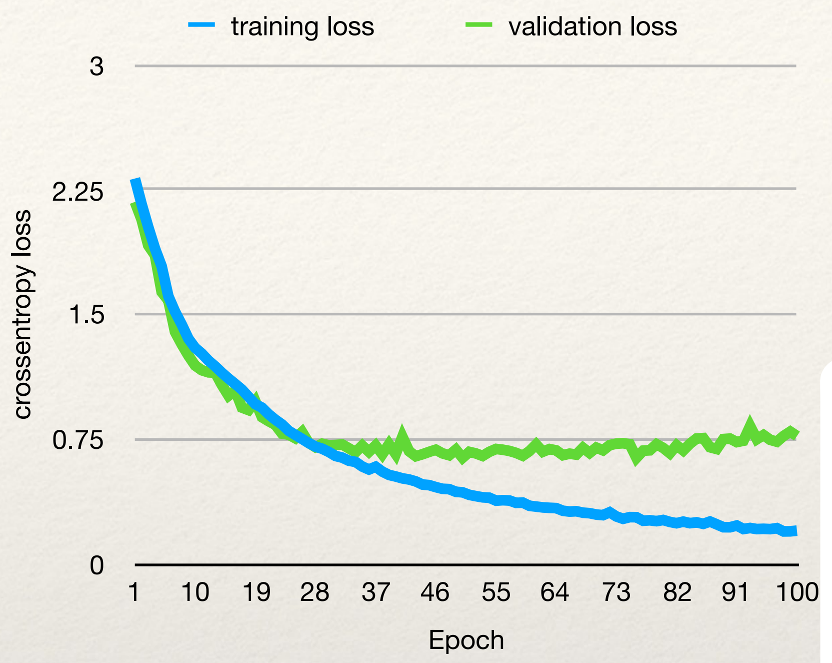
Class	0	1	2	3	4	5	6	7	8	9	10	11
Precision	0.803	0.797	0.720	0.810	0.877	0.874	0.727	0.884	0.947	0.869	0.734	0.706
Recall	0.855	0.797	0.788	0.790	0.814	0.953	0.888	0.855	0.827	0.653	0.710	0.679

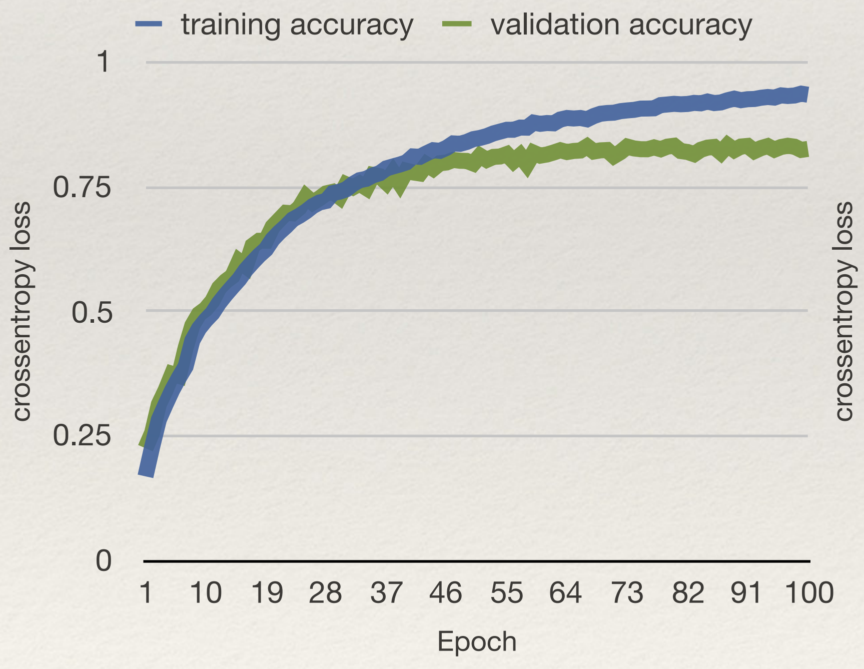
* All prediction based on best model (lowest validation loss), overall validation accuracy = 81.59%

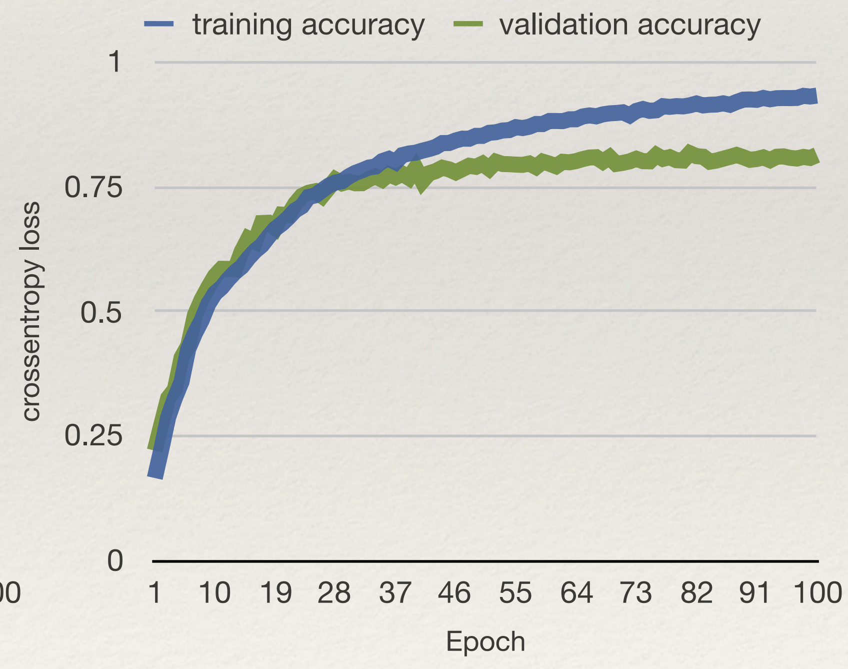
4_GRU_Attention Model
Compare single v.s. multiple embedding

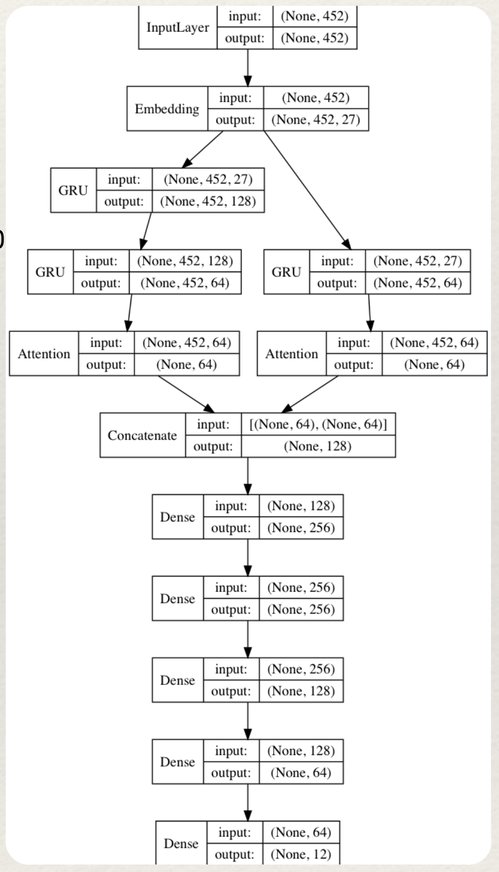

InputLayer | input: (None, 452) | output: (None, 452)
Embedding | input: (None, 452) | output: (None, 452, 27)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
Embedding | input: (None, 452) | output: (None, 452, 27)
GRU | input: (None, 452, 128) | output: (None, 452, 64)
GRU | input: (None, 452, 27) | output: (None, 452, 64)
Attention | input: (None, 452, 64) | output: (None, 64)
Attention | input: (None, 452, 64) | output: (None, 64)
Concatenate | input: [(None, 64), (None, 64)] | output: (None, 128)
Dense | input: (None, 128) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 128)
Dense | input: (None, 128) | output: (None, 64)
Dense | input: (None, 64) | output: (None, 12)







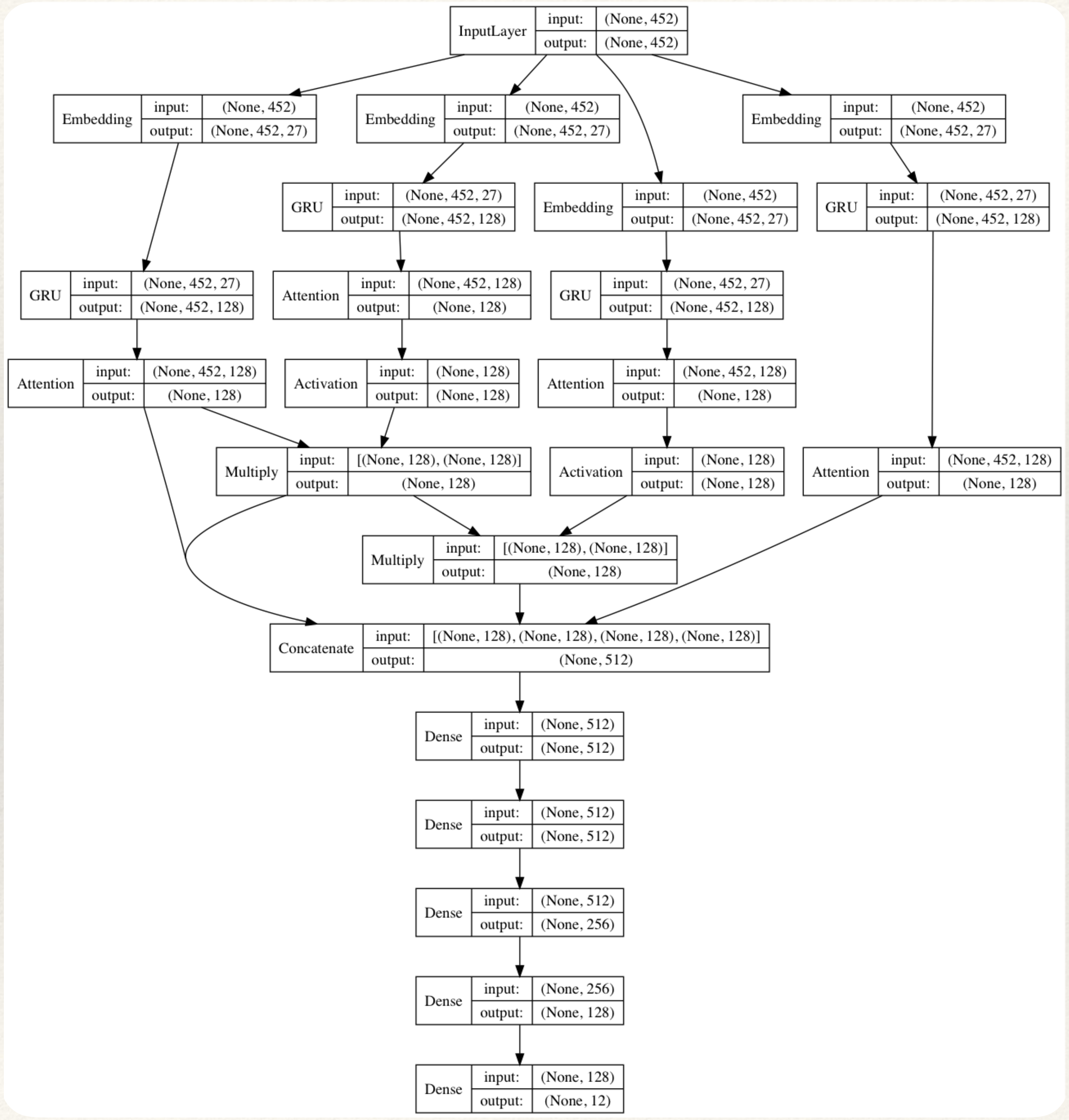



InputLayer | input: (None, 452) | output: (None, 452)
Embedding | input: (None, 452) | output: (None, 452, 27)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
GRU | input: (None, 452, 128) | output: (None, 452, 64)
GRU | input: (None, 452, 27) | output: (None, 452, 64)
Attention | input: (None, 452, 64) | output: (None, 64)
Attention | input: (None, 452, 64) | output: (None, 64)
Concatenate | input: [(None, 64), (None, 64)] | output: (None, 128)
Dense | input: (None, 128) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 128)
Dense | input: (None, 128) | output: (None, 64)
Dense | input: (None, 64) | output: (None, 12)

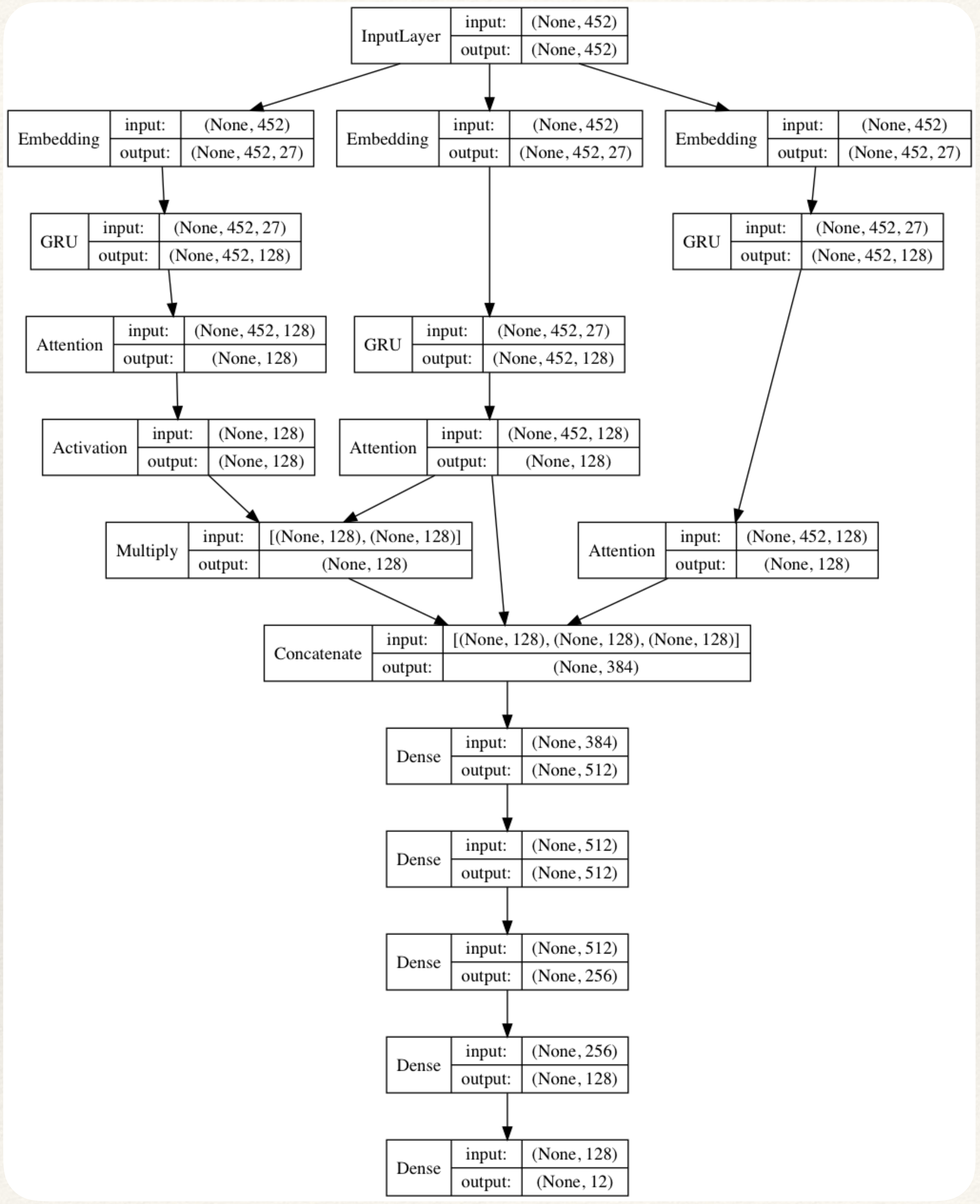
* Standalone validation accuracy = **83.12%** (epoch 67)

* Standalone validation accuracy = 81.39% (epoch 76)

4_GRU_ Attention_Multihop Model



InputLayer | input: (None, 452) | output: (None, 452)
Embedding | input: (None, 452) | output: (None, 452, 27)
Embedding | input: (None, 452) | output: (None, 452, 27)
Embedding | input: (None, 452) | output: (None, 452, 27)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
Embedding | input: (None, 452) | output: (None, 452, 27)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
Attention | input: (None, 452, 128) | output: (None, 128)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
Attention | input: (None, 452, 128) | output: (None, 128)
Activation | input: (None, 128) | output: (None, 128)
Attention | input: (None, 452, 128) | output: (None, 128)
Multiply | input: [(None, 128), (None, 128)] | output: (None, 128)
Activation | input: (None, 128) | output: (None, 128)
Attention | input: (None, 452, 128) | output: (None, 128)
Multiply | input: [(None, 128), (None, 128)] | output: (None, 128)
Concatenate | input: [(None, 128), (None, 128), (None, 128), (None, 128)] | output: (None, 512)
Dense | input: (None, 512) | output: (None, 512)
Dense | input: (None, 512) | output: (None, 512)
Dense | input: (None, 512) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 128)
Dense | input: (None, 128) | output: (None, 12)



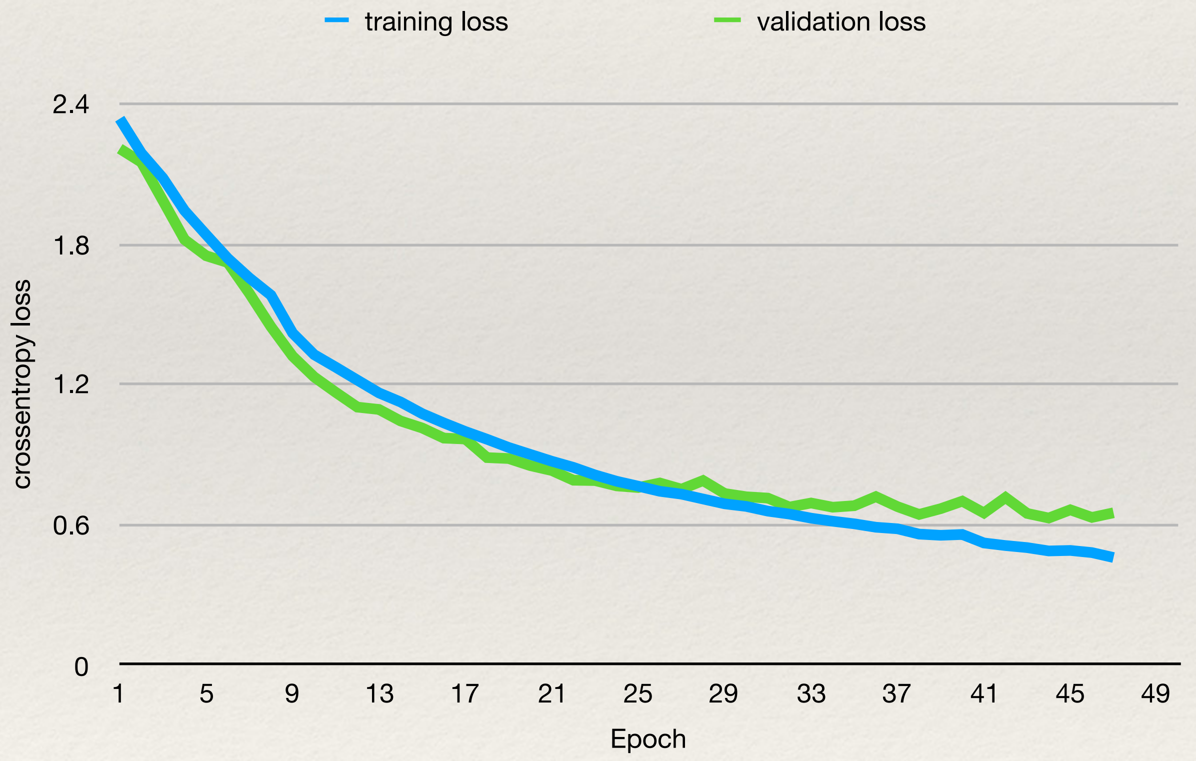
InputLayer | input: (None, 452) | output: (None, 452)
Embedding | input: (None, 452) | output: (None, 452, 27)
Embedding | input: (None, 452) | output: (None, 452, 27)
Embedding | input: (None, 452) | output: (None, 452, 27)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
Attention | input: (None, 452, 128) | output: (None, 128)
GRU | input: (None, 452, 27) | output: (None, 452, 128)
Activation | input: (None, 128) | output: (None, 128)
Attention | input: (None, 452, 128) | output: (None, 128)
Multiply | input: [(None, 128), (None, 128)] | output: (None, 128)
Attention | input: (None, 452, 128) | output: (None, 128)
Concatenate | input: [(None, 128), (None, 128), (None, 128)] | output: (None, 384)
Dense | input: (None, 384) | output: (None, 512)
Dense | input: (None, 512) | output: (None, 512)
Dense | input: (None, 512) | output: (None, 256)
Dense | input: (None, 256) | output: (None, 128)
Dense | input: (None, 128) | output: (None, 12)

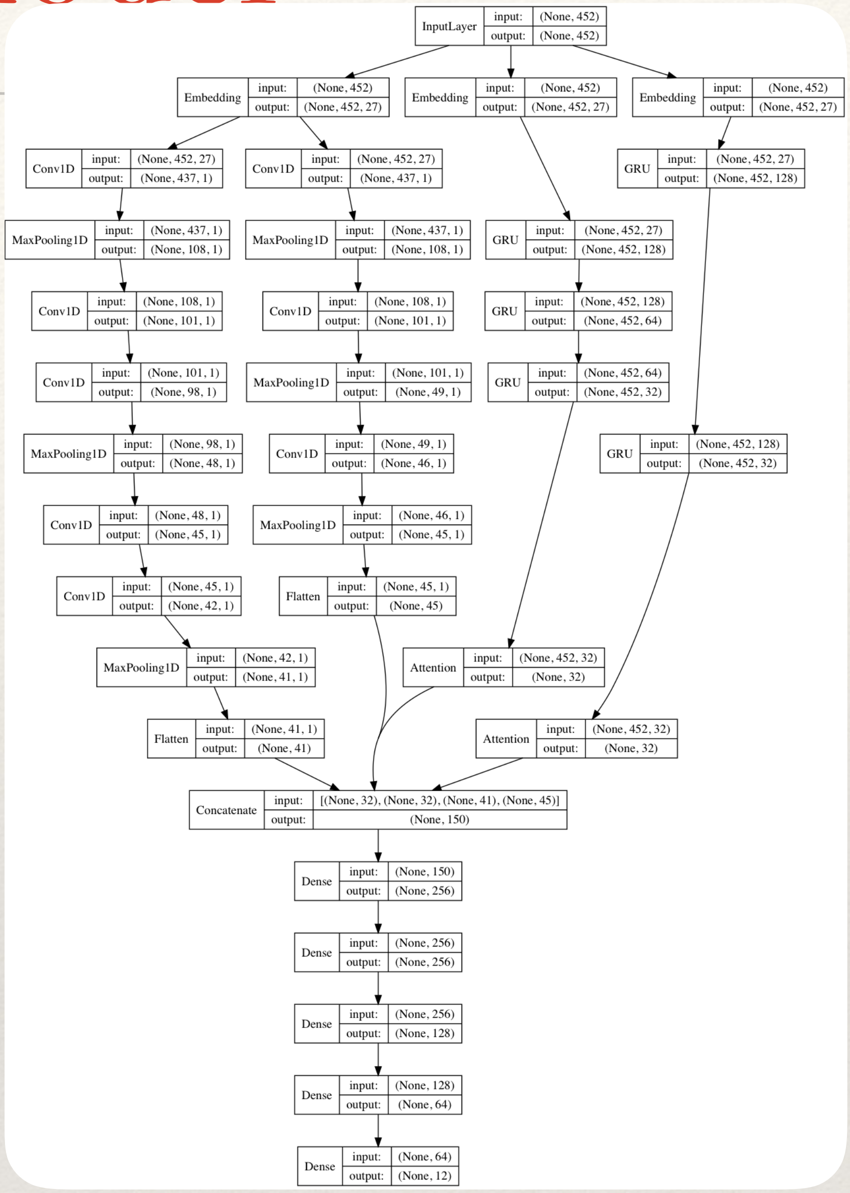
* Standalone validation accuracy = 73.81% (epoch 24)

* Standalone validation accuracy = 76.75% (epoch 33)

5_Hybrid_Parrallel Model

- CNN here is not helping as expected. More study and tuning need to boost this up.
- Best standalone validation accuracy: 80.07% (epoch 44)
- The more is not necessarily better.





InputLayer input: (None, 452) output: (None, 452)
Embedding input: (None, 452) output: (None, 452, 27)
Embedding input: (None, 452) output: (None, 452, 27)
Embedding input: (None, 452) output: (None, 452, 27)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
GRU input: (None, 452, 27) output: (None, 452, 128)
MaxPooling1D input: (None, 437, 1) output: (None, 108, 1)
MaxPooling1D input: (None, 437, 1) output: (None, 108, 1)
GRU input: (None, 452, 27) output: (None, 452, 128)
Conv1D input: (None, 108, 1) output: (None, 101, 1)
Conv1D input: (None, 108, 1) output: (None, 101, 1)
GRU input: (None, 452, 128) output: (None, 452, 64)
Conv1D input: (None, 101, 1) output: (None, 98, 1)
MaxPooling1D input: (None, 101, 1) output: (None, 49, 1)
GRU input: (None, 452, 64) output: (None, 452, 32)
MaxPooling1D input: (None, 98, 1) output: (None, 48, 1)
Conv1D input: (None, 49, 1) output: (None, 46, 1)
GRU input: (None, 452, 128) output: (None, 452, 32)
Conv1D input: (None, 48, 1) output: (None, 45, 1)
MaxPooling1D input: (None, 46, 1) output: (None, 45, 1)
Conv1D input: (None, 45, 1) output: (None, 42, 1)
Flatten input: (None, 45, 1) output: (None, 45)
MaxPooling1D input: (None, 42, 1) output: (None, 41, 1)
Attention input: (None, 452, 32) output: (None, 32)
Flatten input: (None, 41, 1) output: (None, 41)
Attention input: (None, 452, 32) output: (None, 32)
Concatenate input: [(None, 32), (None, 32), (None, 41), (None, 45)] output: (None, 150)
Dense input: (None, 150) output: (None, 256)
Dense input: (None, 256) output: (None, 256)
Dense input: (None, 256) output: (None, 128)
Dense input: (None, 128) output: (None, 64)
Dense input: (None, 64) output: (None, 12)

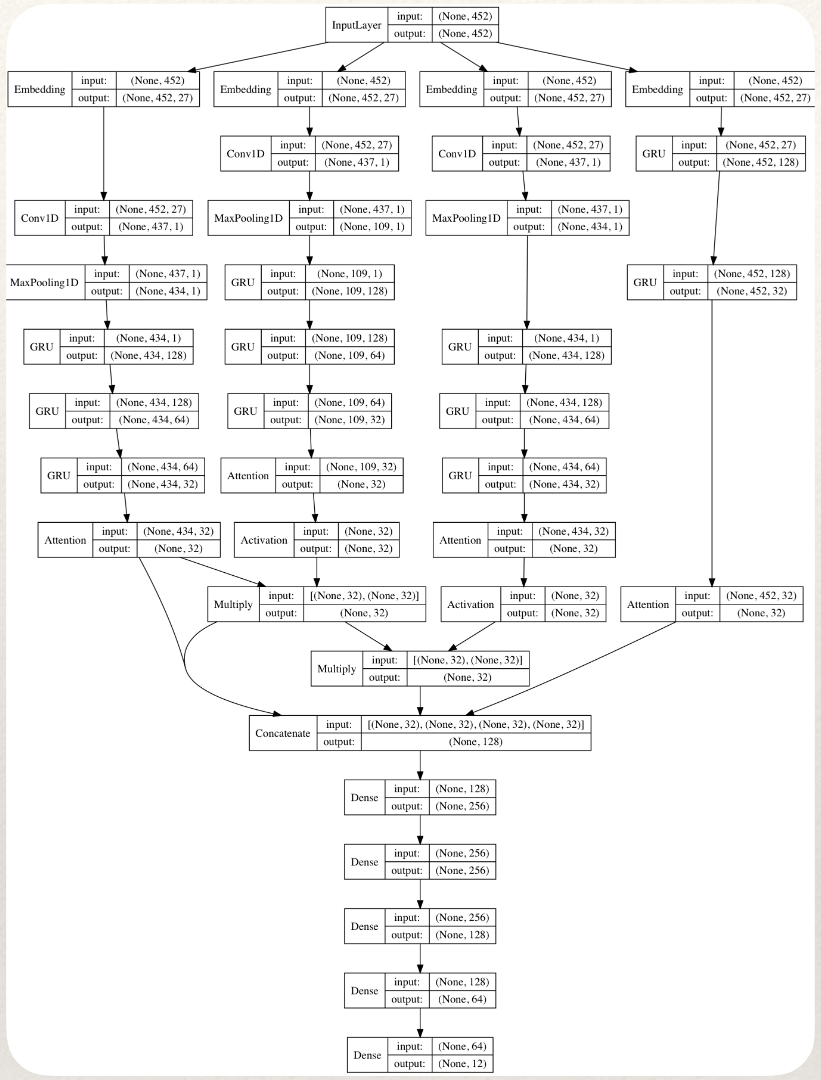
Class	0	1	2	3	4	5	6	7	8	9	10	11
Precision	0.840	0.778	0.810	0.806	0.933	0.880	0.755	0.802	0.832	0.794	0.712	0.718
Recall	0.809	0.763	0.686	0.791	0.766	0.920	0.837	0.889	0.884	0.699	0.655	0.629

* All prediction based on best model (lowest validation loss), overall validation accuracy = 80.07%

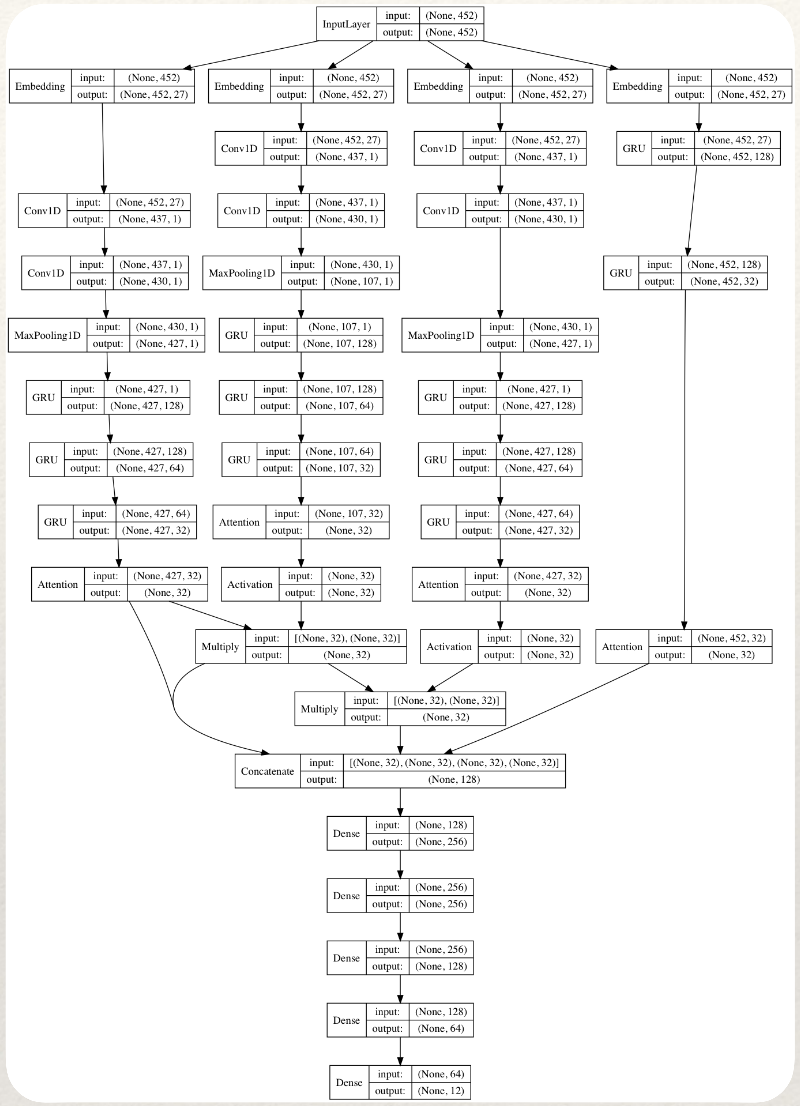
GRU Activation: tanh
CNN Activation: ReLU
DNN Activation: LeakyReLU
Last Activ.: softmax
Loss: crossentropy
Optimizer: Adam
Batch size: 100
Dropout: 0.2

* Structure details please refer to 5_Hybrid.py

6_Hybrid_Concat Model

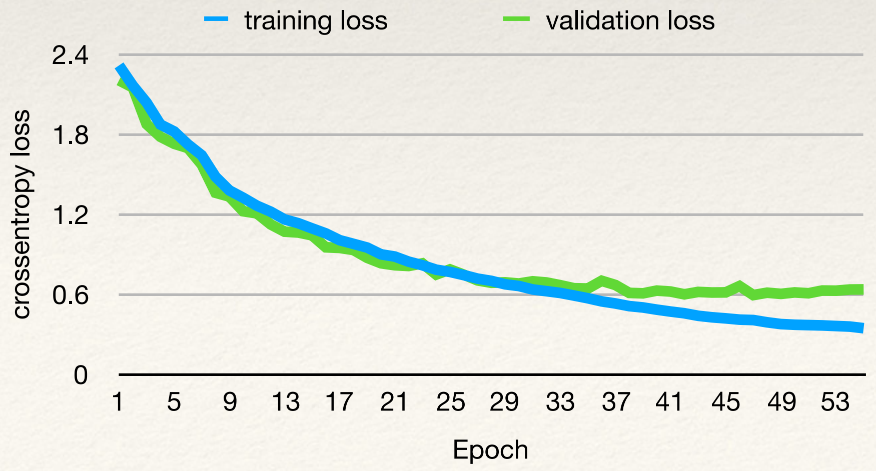


InputLayer input: (None, 452) output: (None, 452)
Embedding input: (None, 452) output: (None, 452, 27)
Embedding input: (None, 452) output: (None, 452, 27)
Embedding input: (None, 452) output: (None, 452, 27)
Embedding input: (None, 452) output: (None, 452, 27)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
GRU input: (None, 452, 27) output: (None, 452, 128)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
MaxPooling1D input: (None, 437, 1) output: (None, 109, 1)
MaxPooling1D input: (None, 437, 1) output: (None, 434, 1)
MaxPooling1D input: (None, 437, 1) output: (None, 434, 1)
GRU input: (None, 109, 1) output: (None, 109, 128)
GRU input: (None, 452, 128) output: (None, 452, 32)
GRU input: (None, 434, 1) output: (None, 434, 128)
GRU input: (None, 109, 128) output: (None, 109, 64)
GRU input: (None, 434, 1) output: (None, 434, 128)
GRU input: (None, 434, 128) output: (None, 434, 64)
GRU input: (None, 109, 64) output: (None, 109, 32)
GRU input: (None, 434, 128) output: (None, 434, 64)
GRU input: (None, 434, 64) output: (None, 434, 32)
Attention input: (None, 109, 32) output: (None, 32)
GRU input: (None, 434, 64) output: (None, 434, 32)
Attention input: (None, 434, 32) output: (None, 32)
Activation input: (None, 32) output: (None, 32)
Attention input: (None, 434, 32) output: (None, 32)
Multiply input: [(None, 32), (None, 32)] output: (None, 32)
Activation input: (None, 32) output: (None, 32)
Attention input: (None, 452, 32) output: (None, 32)
Multiply input: [(None, 32), (None, 32)] output: (None, 32)
Concatenate input: [(None, 32), (None, 32), (None, 32), (None, 32)] output: (None, 128)
Dense input: (None, 128) output: (None, 256)
Dense input: (None, 256) output: (None, 256)
Dense input: (None, 256) output: (None, 128)
Dense input: (None, 128) output: (None, 64)
Dense input: (None, 64) output: (None, 12)

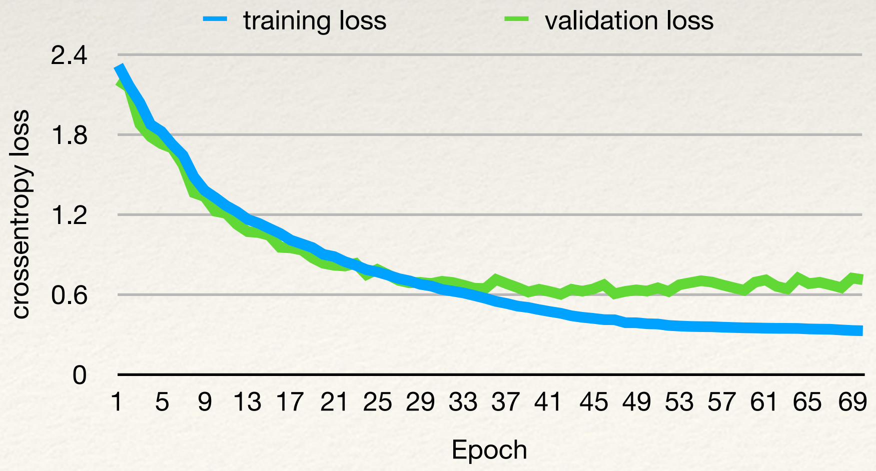


InputLayer input: (None, 452) output: (None, 452)
Embedding input: (None, 452) output: (None, 452, 27)
Embedding input: (None, 452) output: (None, 452, 27)
Embedding input: (None, 452) output: (None, 452, 27)
Embedding input: (None, 452) output: (None, 452, 27)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
GRU input: (None, 452, 27) output: (None, 452, 128)
Conv1D input: (None, 452, 27) output: (None, 437, 1)
Conv1D input: (None, 437, 1) output: (None, 430, 1)
Conv1D input: (None, 437, 1) output: (None, 430, 1)
Conv1D input: (None, 437, 1) output: (None, 430, 1)
MaxPooling1D input: (None, 430, 1) output: (None, 107, 1)
GRU input: (None, 452, 128) output: (None, 452, 32)
MaxPooling1D input: (None, 430, 1) output: (None, 427, 1)
GRU input: (None, 107, 1) output: (None, 107, 128)
MaxPooling1D input: (None, 430, 1) output: (None, 427, 1)
GRU input: (None, 427, 1) output: (None, 427, 128)
GRU input: (None, 107, 128) output: (None, 107, 64)
GRU input: (None, 427, 1) output: (None, 427, 128)
GRU input: (None, 427, 128) output: (None, 427, 64)
GRU input: (None, 107, 64) output: (None, 107, 32)
GRU input: (None, 427, 128) output: (None, 427, 64)
GRU input: (None, 427, 64) output: (None, 427, 32)
Attention input: (None, 107, 32) output: (None, 32)
GRU input: (None, 427, 64) output: (None, 427, 32)
Attention input: (None, 427, 32) output: (None, 32)
Activation input: (None, 32) output: (None, 32)
Attention input: (None, 427, 32) output: (None, 32)
Multiply input: [(None, 32), (None, 32)] output: (None, 32)
Activation input: (None, 32) output: (None, 32)
Attention input: (None, 452, 32) output: (None, 32)
Multiply input: [(None, 32), (None, 32)] output: (None, 32)
Concatenate input: [(None, 32), (None, 32), (None, 32), (None, 32)] output: (None, 128)
Dense input: (None, 128) output: (None, 256)
Dense input: (None, 256) output: (None, 256)
Dense input: (None, 256) output: (None, 128)
Dense input: (None, 128) output: (None, 64)
Dense input: (None, 64) output: (None, 12)

* Standalone validation accuracy = 78.39% (epoch 51)



* Standalone validation accuracy = 77.04% (epoch 63)



References

(1) Attention layer is modified from:
https://github.com/richliao/textClassifier/blob/master/textClassifierHATT.py

(2) Inspirations of trying CNN on raw character sequences:
https://arxiv.org/pdf/1502.01710.pdf

(3) Normal dropout v.s. recurrent dropout on RNN:
https://arxiv.org/pdf/1512.05287.pdf

(4) GRU v.s. LSTM:
https://arxiv.org/pdf/1412.3555.pdf

(5) seq2seq using Attention:
https://arxiv.org/pdf/1508.04025.pdf
https://www.cs.cmu.edu/%7Ediyiy/docs/naacl16.pdf
https://arxiv.org/pdf/1706.03762.pdf

(6) CNN parameter tuning: How to choose filter size, kernel size, strides and pooling
https://arxiv.org/pdf/1510.03820.pdf

(7) CNN dropout location:
https://arxiv.org/pdf/1512.01400.pdf

(8) Pyramid-CNN
http://ai.tencent.com/ailab/media/publications/ACL3-Brady.pdf

(9) Hierarchical Attention Networks
https://www.cs.cmu.edu/%7Ediyiy/docs/naacl16.pdf

Tools & Environment

- Python 2.7.13
- Keras (2.1.3), TensorFlow (1.5.0), Theano (1.0.1), numpy (1.14.0), scikit-learn (0.19.1)