DENSE AND CONVOLUTIONAL NEURAL NETWORKS FOR SENTIMENT ANALYSIS

Robert Jankowski
284516@pw.edu.pl

April 14, 2020

ABSTRACT

This work explores deep neural networks for sentiment analysis task. We tested three different neural network architectures: (i) character-level Convolutional Neural Network, (ii) word-level Convolutional Neural Network, (iii) a combination of Dense Neural Network and Convolutional Neural Network on word-level input on movie review dataset. Results show a slight improvement in merging networks for the binary classification task.

1 INTRODUCTION

Sentiment analysis (SA) is one of research area in text mining field. SA is the computational treatment of opinions, sentiments and subjectivity of text. It is widely used in analysis Twitter data [1], news, blogs [2] and movie reviews [3]. In this work, we mainly focus on two research papers: *Convolutional Neural Networks for Sentence Classification* [4] and *Character-level Convolutional Networks for Text Classification* [5].

Authors in [5] tested various neural networks architectures on different datasets (*e.g.* Yelp reviews, Yahoo! Answers dataset, Amazon reviews), the smallest one has above 120,000 train samples, the biggest 3,600,000, which is significantly more than in this work. They compared deep learning approach to classic methods i.e., bag of words, n-grams and found that on the largest datasets DL models perform better than "traditional" methods (*e.g.* n-grams TFIDF), which remain strong candidates for dataset of size up to several hundreds of thousands. They concluded that the choice of the proper alphabet is crucial, omitting lower and uppercase could lead to regularization effect.

In [4] author explored convolutional neural network (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. The experiments were conducted on multiple datasets with excellent results, giving competitive results against the more sophisticated deep learning models that utilize complex pooling schemes [6].

2 DATA

In our experiments, we use movie review dataset[1] (5331 positive and 5331 negative reviews). In preprocessing part each review was lowercased, we filter out unnecessary characters *e.g.* punctuation, tabs, newlines, also remove stop words, i.e., most common words in a language. In our work, we compare CNN on character and word-level with a properly prepared dataset.

2.1 Processing characters

The total number of unique characters (60 characters) – first dimension of the input matrix and maximum length of review (269 characters) – second dimension of the input matrix were calculated. Using one hot encoding the text was converted into matrix form and could be visualized as the white and black image (Fig. 1). We decide to cut the maximum length of review to 66 character, i.e., the median of the distribution of the number of characters (Fig. 2).

[1] http://www.cs.cornell.edu/people/pabo/movie-review-data/

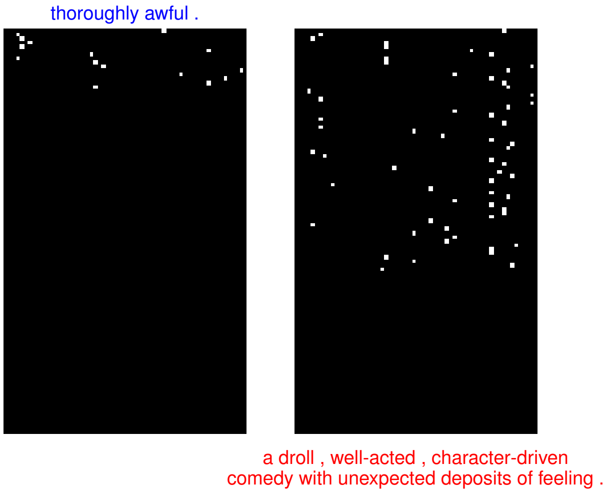

Figure 1: Examples of encoded texts using one-hot encoding; (left) negative review, (right) positive review.

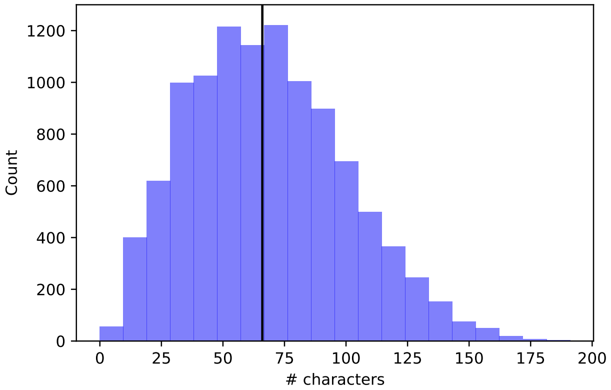

Figure 2: Distribution of the number of characters. The black line marks median – 66 characters.

2.2 Processing words

Using pre-trained `word2vec`[2] model the words were transformed into 300-dimensional vectors. When a word does not exist in model vocabulary, it is initialized as zeros. The second dimension for the input matrix is median number of words (Fig. 4).

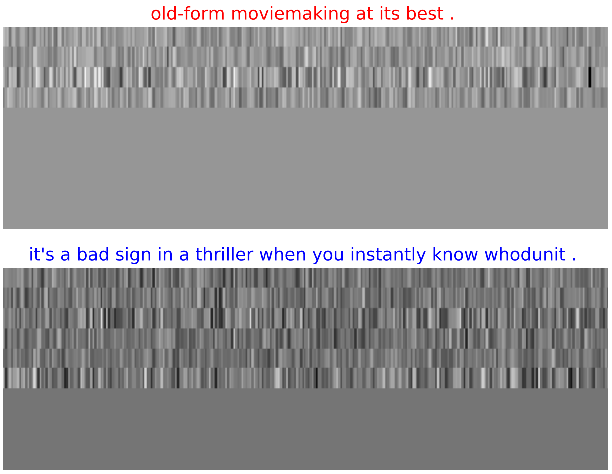

Figure 3: Examples of encoded texts using `word2vec` model. (top) Positive review, (bottom) negative review.

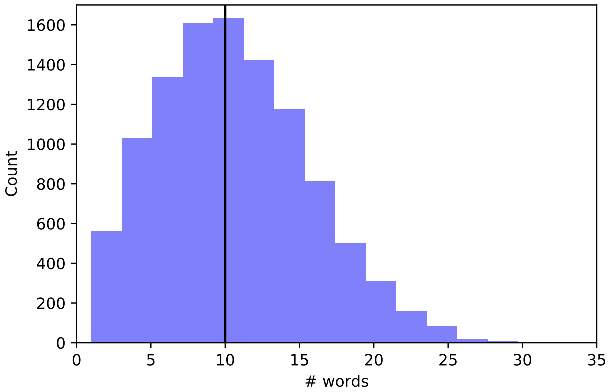

Figure 4: Distribution of the number of words. The black line marks median – 10 words.

3 Model

Models were built in `tensorflow`[3] library, with *RMSprop* optimizer and *binary cross-entropy* loss function. We use a cross-validation method (StratifiedKFold with $K = 3$) to evaluate models, then measure the following metrics: (i) F_1, (ii) AUC and (iii) accuracy. For regularization, the early stopping callback was used, with monitoring the validation accuracy and special callback that reduce the learning rate on a plateau. The architecture of the models are shown on Fig. 5 – 7. For *Dense & CNN* model the input of dense part was calculated as the sum of each word embedding vector.

[2] https://code.google.com/archive/p/word2vec/

[3] https://www.tensorflow.org/

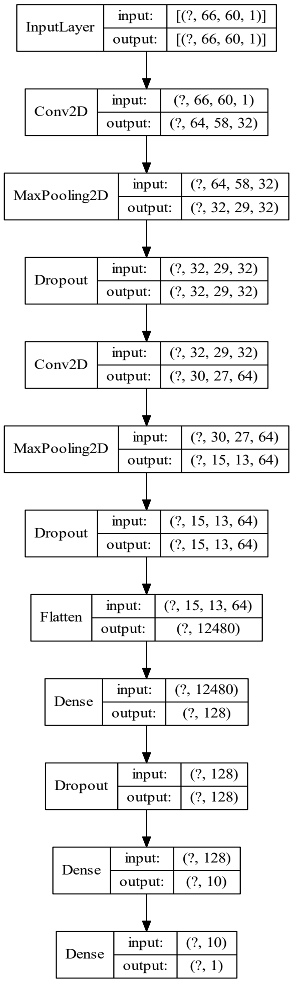
InputLayer input: [(?, 66, 60, 1)] output: [(?, 66, 60, 1)]
Conv2D input: (?, 66, 60, 1) output: (?, 64, 58, 32)
MaxPooling2D input: (?, 64, 58, 32) output: (?, 32, 29, 32)
Dropout input: (?, 32, 29, 32) output: (?, 32, 29, 32)
Conv2D input: (?, 32, 29, 32) output: (?, 30, 27, 64)
MaxPooling2D input: (?, 30, 27, 64) output: (?, 15, 13, 64)
Dropout input: (?, 15, 13, 64) output: (?, 15, 13, 64)
Flatten input: (?, 15, 13, 64) output: (?, 12480)
Dense input: (?, 12480) output: (?, 128)
Dropout input: (?, 128) output: (?, 128)
Dense input: (?, 128) output: (?, 10)
Dense input: (?, 10) output: (?, 1)

Figure 5: Architecture of character-level CNN.

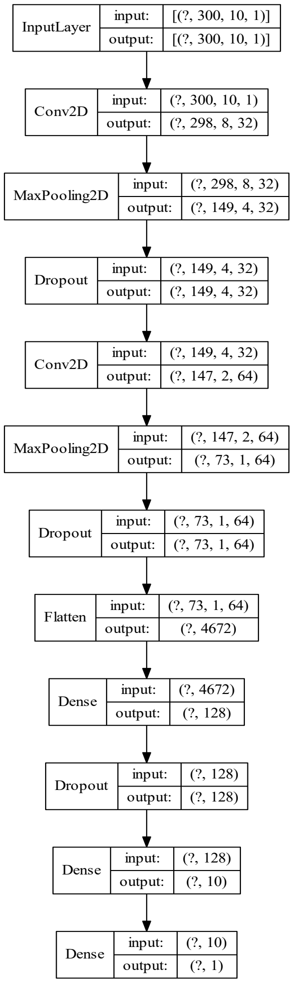
InputLayer input: [(?, 300, 10, 1)] output: [(?, 300, 10, 1)]
Conv2D input: (?, 300, 10, 1) output: (?, 298, 8, 32)
MaxPooling2D input: (?, 298, 8, 32) output: (?, 149, 4, 32)
Dropout input: (?, 149, 4, 32) output: (?, 149, 4, 32)
Conv2D input: (?, 149, 4, 32) output: (?, 147, 2, 64)
MaxPooling2D input: (?, 147, 2, 64) output: (?, 73, 1, 64)
Dropout input: (?, 73, 1, 64) output: (?, 73, 1, 64)
Flatten input: (?, 73, 1, 64) output: (?, 4672)
Dense input: (?, 4672) output: (?, 128)
Dropout input: (?, 128) output: (?, 128)
Dense input: (?, 128) output: (?, 10)
Dense input: (?, 10) output: (?, 1)

Figure 6: Architecture of word-level CNN.

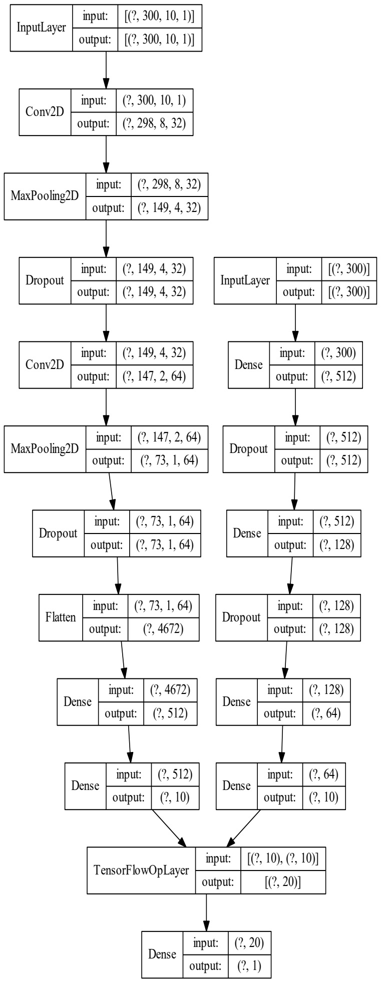
InputLayer input: [(?, 300, 10, 1)] output: [(?, 300, 10, 1)]
Conv2D input: (?, 300, 10, 1) output: (?, 298, 8, 32)
MaxPooling2D input: (?, 298, 8, 32) output: (?, 149, 4, 32)
Dropout input: (?, 149, 4, 32) output: (?, 149, 4, 32)
Conv2D input: (?, 149, 4, 32) output: (?, 147, 2, 64)
MaxPooling2D input: (?, 147, 2, 64) output: (?, 73, 1, 64)
Dropout input: (?, 73, 1, 64) output: (?, 73, 1, 64)
Flatten input: (?, 73, 1, 64) output: (?, 4672)
Dense input: (?, 4672) output: (?, 512)
Dense input: (?, 512) output: (?, 10)
InputLayer input: [(?, 300)] output: [(?, 300)]
Dense input: (?, 300) output: (?, 512)
Dropout input: (?, 512) output: (?, 512)
Dense input: (?, 512) output: (?, 128)
Dropout input: (?, 128) output: (?, 128)
Dense input: (?, 128) output: (?, 64)
Dense input: (?, 64) output: (?, 10)
TensorFlowOpLayer input: [(?, 10), (?, 10)] output: [(?, 20)]
Dense input: (?, 20) output: (?, 1)

Figure 7: Architecture of word-level Dense & CNN.

4 Results

Tables 1–3 summarise achieved results. Character-level models perform poorly, the maximum value of F_1 is 0.35 which indicates low performance on test data. Also, the accuracy (max 0.54) and AUC (max 0.56) show the possibility of the overfitting of the models on the dataset and their lack of ability to generalize acquired knowledge. The word-level models improve accuracy and AUC by over 0.2. The combination of dense and CNN slightly boosts some metrics (AUC and accuracy by 0.03), however, the F_1 stays invariable.

Model	Accuracy	AUC	F1
1	0.53	0.55	0.34
2	**0.54**	**0.56**	0.31
3	0.53	0.53	**0.35**

Table 1: Results on test data for character-level CNN.

Model	Accuracy	AUC	F1
1	**0.72**	**0.79**	0.42
2	0.69	0.77	0.39
3	0.70	0.78	**0.43**

Table 2: Results on test data for word-level CNN.

Model	Accuracy	AUC	F1
1	**0.75**	**0.82**	0.41
2	0.73	0.78	**0.43**
3	0.74	0.80	0.42

Table 3: Results on test data for word-level Dense & CNN.

We also test models on example sentences, Tab. 4 shows predictions of the best model for each architecture. All models capture that the first sentence is very positive. Interestingly, word-level CNN predicts this only with probability 0.67. The predictions for second positive sentence are similar. The third sentence is not as negative as the fourth one, but all models indicate it as the most negative. Unfortunately, Model A predicts fourth sentence as quite positive, which reveals its prediction limits.

Sentence	Model A	Model B	Model C
I really like this film, one of the best I've ever seen.	0.92	0.67	0.99
I like this film and recommend to everyone.	0.85	0.77	0.94
Sometimes boring with a simple plot twist	0.05	0.01	0.01
The movie was terrible, not worth watching once again.	0.92	0.05	0.15

Table 4: Example predictions for the best model from cross-validation. *Model A* – character-level CNN, *Model B* – word-level CNN, *Model C* – word-level Dense & CNN; **0** - negative, **1** - positive.

5 Conclusions

The work compares different convolutional neural network architecture for sentiment classification task. Using character-level CNN we experienced some difficulties, the models perform poorly on the prepared dataset. One solution for the above problem could be the choice of the more appropriate alphabet, as in [5] authors concluded that the alphabet plays a crucial role, or the change the sizes of the input matrix.

The word-level neural networks achieved better results, due to the pre-train `word2vec` model, which was able to learn words embedding, linguistic context of words, on a huge dataset. Using two networks can improve metrics a little bit i.e., information from dense layers may be important for classification.

The obtained results are significantly worse than in [4] and [5]. We did not copy models architectures, tried to create custom ones. Besides, the dataset was rather small (only 10,000 samples) comparing to other papers, wherein the authors also focused on data augmentation using Thesaurus [5].

Further work could be focused on experiments on more datasets and hyperparameters tuning of the models. Moreover, finding the proper model architecture could be also investigated.

References

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, pages 30–38, 2011.

[2] Namrata Godbole, Manja Srinivasaiah, and Steven Skiena. Large-scale sentiment analysis for news and blogs. *Icwsm*, 7(21):219–222, 2007.

[3] Vivek Kumar Singh, Rajesh Piryani, Ashraf Uddin, and Pranav Waila. Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification. In *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, pages 712–717. IEEE, 2013.

[4] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[5] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[6] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

A Training results

Figures 8–10 show training results of each model in cross-validation method. In Fig. 8 loss value for test data is raising, which is undesirable and indicates overfitting. Other metrics for train data are reaching the maximum value, however, on test data, they stay invariable. Figure 9 shows more appropriate training results, the loss value is decreasing in time. In Fig. 10 the loss value is increasing, but most metrics stay still.

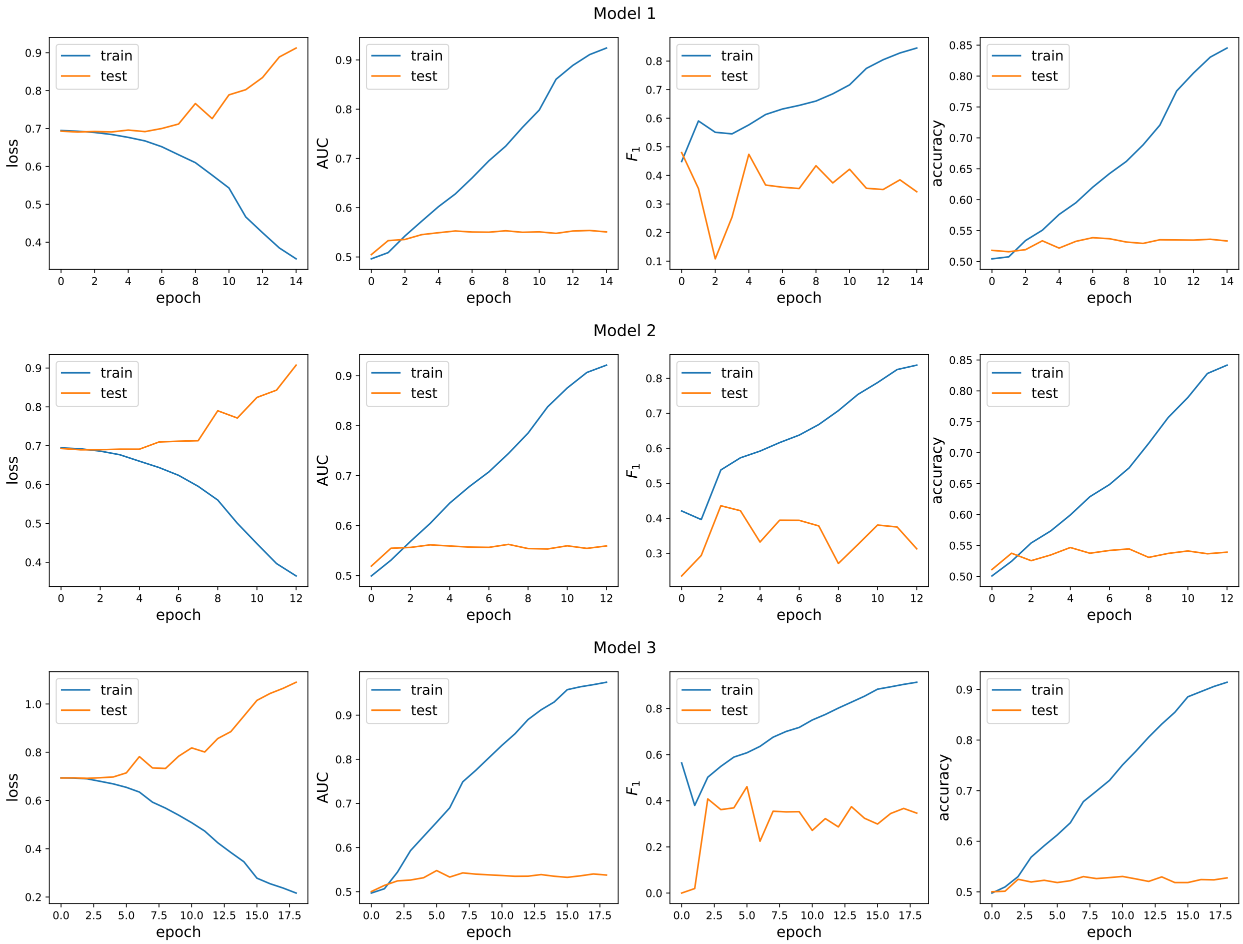

Figure 8: Results during training for each cross-validated model – **character-level CNN**.

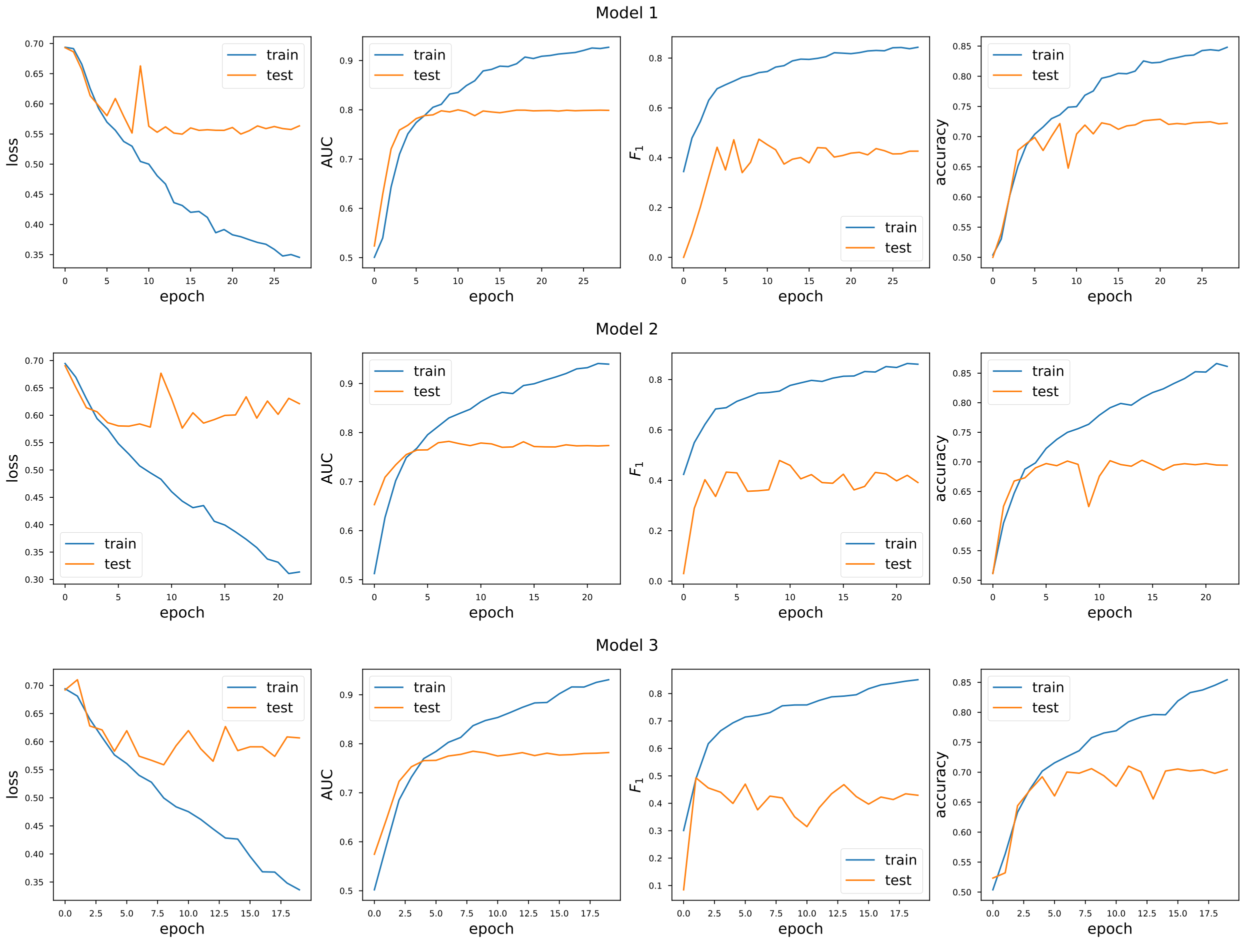

Figure 9: Results during training for each cross-validated model – **word-level CNN**.

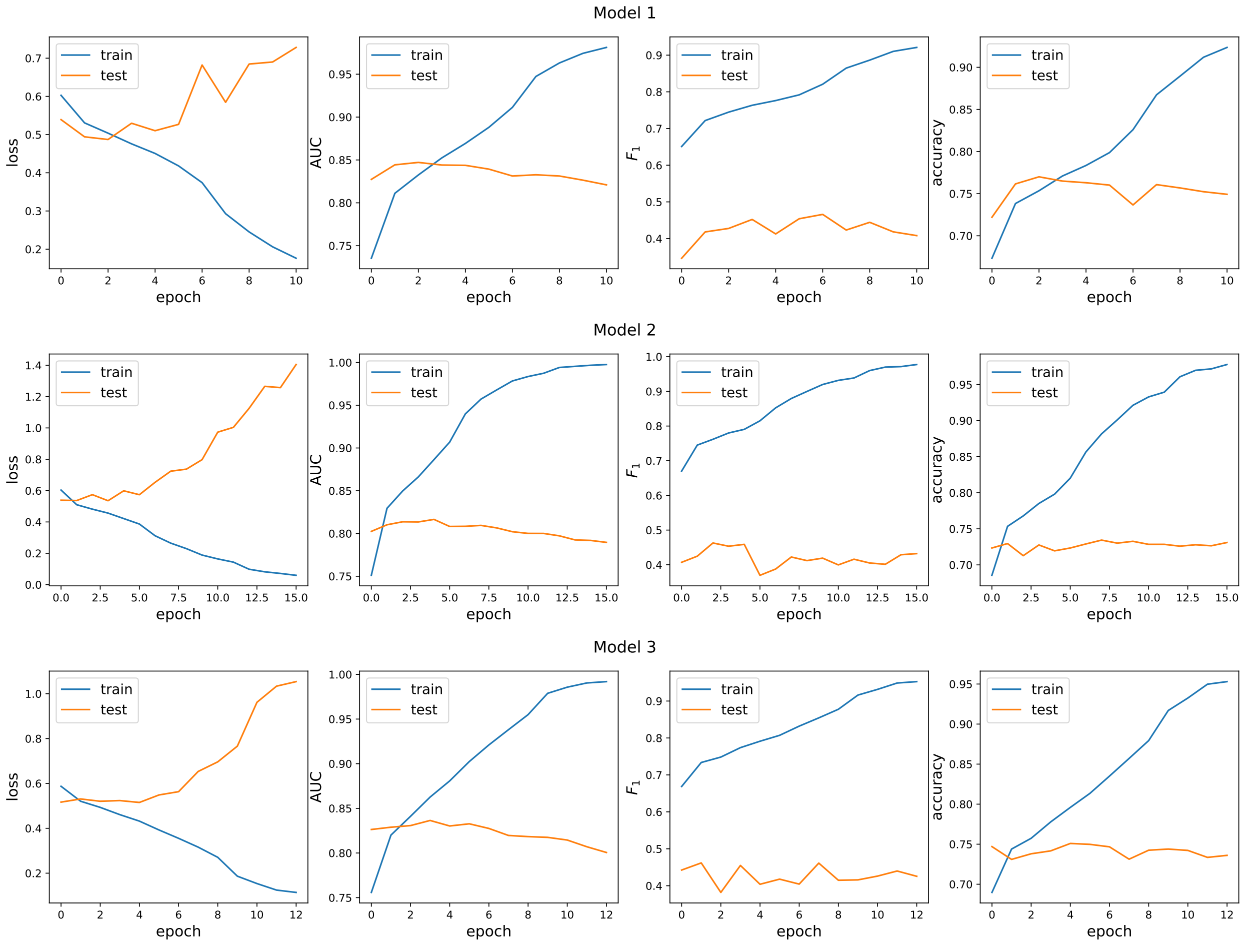

Figure 10: Results during training for each cross-validated model – **word-level Dense & CNN**.