Evaluation of Structural Hyperparameters for Text Classification with LSTM Networks

M. Frković*, N. Čerkez**, B. Vrdoljak* and S. Skansi***
*University of Zagreb Faculty of Electrical Engineering and Computing, Zagreb, Croatia
**College of Information Technologies (VSITE), Zagreb, Croatia
***University of Zagreb Faculty of Croatian Studies, Zagreb, Croatia
marija.frkovic@fer.hr
ninoslav.cerkez@vsite.hr
boris.vrdoljak@fer.hr
sskansi@hrstud.hr

***Abstract* - In natural language processing, most problems can be interpreted as text classification tasks, which makes this type of task a central one. A natural subdivision of more complex types of text classification can be made according to whether the classification is multilabel or multiclass, where both of these can be tackled with either a multiclass classifier or with a combination of several binary classifiers. An example of the problem which offers a natural way of comparing multiclass and binary classification on the same data, which makes the results comparable, is a classification of text author MBTI personality type. The dataset used is PersonalityCafe MBTI. We focus our comparison on structural hyperparameters, which are the hyperparameters pertaining to network structure. Structural hyperparameters are necessary for specifying the network itself, which makes them a primary concern in its construction. The hyperparameters investigated in this paper are the number of hidden layers and layer size. Through a number of experiments, we demonstrate the choice of hyperparameters and conclude with general hyperparameter selection recommendations based on our results.**

***Keywords* - natural language processing; LSTM; structural hyperparameters; multiclass classification; binary classification.**

I. INTRODUCTION

Deep neural networks can discover representations needed for detection or classification on multiple levels, starting from simple raw input to higher and more abstract level [1] An important application of deep neural networks can be found in natural language processing (NLP), which is an interdisciplinary field of computer science, artificial intelligence, and linguistics, based on tools and algorithms for understanding and processing naturally spoken and written language [3].

Long Short-Term Memory networks (LSTM networks) [11] are a type of recurrent neural network capable of learning long-term dependencies, initially invented to address the vanishing gradient problem [8]. The LSTM networks have a structure of linked memory cell blocks, where the information is stored. The memory cell has a recurrent connection to the previous time step, which is multiplicatively gated by another cell that learns to decide when to clear the content of the memory [4].

Problems in NLP, which can be interpreted as text classification tasks, have been successfully addressed with LSTM networks. A natural subdivision of more complex text classification tasks can be made according to whether the classification is multilabel or multiclass. Multiclass text classification can be tackled with either a multiclass classifier [14] or with a combination of several binary classifiers [12]. An example of the problem which offers a natural way of comparing multiclass and binary classification on the same data, which makes the results comparable, is a classification of text author MBTI personality type.

The primary motivation behind this paper is to make a clear comparison and guideline for hyperparameter selection based on a few experiments on the same dataset. This motivation has been inspired by the complexity of the neural networks, where a significant number of elements can affect the result of training and their ability to classify data correctly. We focus our comparison on structural hyperparameters, which are the hyperparameters pertaining to network structure. Structural hyperparameters are necessary for specifying the network itself, which makes them a primary concern in its construction. The hyperparameters considered in our paper are the number of hidden layers and layer size.

We approach the personality type classification problem in two different ways: through multiclass and binary classification. Through several experiments, we demonstrate the choice of hyperparameters and conclude with general hyperparameter selection recommendations based on our results. The obtained results will be used as a basis for later hyperparameter optimization to achieve higher accuracy in personality type prediction.

The paper is organized as follows: Section II describes text processing referring to the chosen dataset; Section III presents experimental setup with the emphasis on hyperparameters; The results and discussion for multiclass and binary classification are shown in section IV; and, section V concludes this paper.

II. TEXT PREPROCESSING

Before building an LSTM neural network model and performing hyperparameter analysis, we preprocess the data in the standard way.

Authorized licensed use limited to: University of Zagreb: Faculty of Electrical Engineering and Computing. Downloaded on March 20,2021 at 15:56:30 UTC from IEEE Xplore. Restrictions apply.

A. Dataset

The dataset used for this research is an MBTI Type dataset from Kaggle[1] which consists of 8.675 rows containing (a) textual posts from PersonalityCafe forum and (b) labels, which are MBTI personality types of their respective authors. Each author has an average of 50 posts. An MBTI dataset was selected because MBTI labels offer a rather complex yet well-defined multiclass text classification task, with 16 overlapping classes, which can also be broken down to binary classifications in a very natural way with four well-defined classifiers: E/I, N/S, T/F, P/J. Because the tasks are equally well defined for both the multiclass and binary setting, MBTI datasets are an excellent domain for comparing these two approaches. For more on the theory of MBTI personality types we refer the reader to [2]. The dataset used is quite large but imbalanced. Since our research does not focus on surpassing a benchmark but rather on hyperparameter evaluation of two approaches, no measure was taken to correct the imbalance as it would not influence the results of our analysis.

B. Word Embeddings

Data from the dataset was prepared and cleaned by removing links, numbers, punctuation, and special characters. Contractions were expanded to their full form. All uppercase characters were converted to lowercase. After the tokenization of sentences, where they were transformed into a list of words, stop words and one-letter words were removed. Since the posts contain a lot of MBTI type code words, they were also removed from the text, to prevent unnecessary noise in the text data. Lemmatization was performed using WordNetLemmatizer. The text was finally converted into word embeddings, using the Glove 100-dimensional word, pretrained on Twitter corpus of 2 billion tweets [9] .

III. Experiment Setup

The focus of this research is to compare different LSTM models based on structural hyperparameters.

Hyperparameters are parameters that are not learned during training but must be defined before training. As such, research in hyperparameters forms the only viable basis for selecting good hyperparameters. For the purpose of this paper, we will provisionally define two non-disjoint classes of hyperparameters. Backpropagation hyperparameters are hyperparameters controlling the optimization of the network parameters. These are, among others, regularization, learning rate, number of epochs, momentum, etc. In this paper, we are interested in a second class, namely the structural hyperparameters.

As the name suggests, these are parameters governing the neural network structure. The structural parameters we explore are the hidden layer size and the number of hidden layers.

Structural hyperparameters can be considered the dominant hyperparameters in learning since they are a necessary part of creating internal representations in neural networks, as opposed to optional parameters such as regularization.

In our experiments, we trained models of varying numbers of hidden layers and hidden layer size, which we will discuss below. To keep the comparison clear, we have opted not to use any regularization. Other necessary (non-structural) hyperparameters were kept invariant in all models: sigmoid activation for binary and softmax activation for multiclass classification, the batch size was set to 64, the number of epochs to 30, the optimizer was ADAM [10] with the learning rate of 0.001, and no regularization or dropout [5] was used. We have trained all models with both cross-entropy loss (C-E) and mean squared error loss (MSE). Modifying these parameters might improve results even further, but this will be left as a topic for future research. The architecture of the individual layers was Keras/Tensorflow CuDNNLSTM[2], trained on CUDA 10.1 GPUs.

By fixing the dataset and epochs, we deliberately limit the hyperparameters in question and focus only on the combination of layer size and number of neurons per layer (with two different loss functions), which to the best of our knowledge was not systematically explored for multi-layered LSTM networks with textual embeddings for multiclass text-based document classification. Results are evaluated by comparing multiple metrics, such as accuracy, F1-score, precision, recall, confusion matrix.

IV. Results

The results we obtained for multiclass text classification from the experiments conducted, in terms of accuracy, recall, precision, and F1 score are presented in TABLE I (we have bolded the best three results per column).

The results show that when it comes to accuracy, deeper neural networks tend to give better results. When looking at the average macro precision, recall, and F1 score, networks with 3 layers give similar or slightly worse results than the ones with 1 or 2 layers.

When it comes to the number of neurons per layer, in the networks with a single LSTM layer, a small number of neurons gives better results. In fact, models with 100, 150 and 500 neurons per layer have shown the signs of overfitting (Fig.1a), because after around 15th epoch there is a steep increase in training accuracy and decrease in training loss, but at the same time the validation accuracy is not changing or getting slightly worse, and validation loss is starting to increase as well, as shown in Fig.1b, Fig.1c and Fig.1d.

[1] https://www.kaggle.com/datasnaek/mbti-type

[2] https://keras.io/

Authorized licensed use limited to: University of Zagreb: Faculty of Electrical Engineering and Computing. Downloaded on March 20,2021 at 15:56:30 UTC from IEEE Xplore. Restrictions apply.

TABLE I. MULTICLASS CLASSIFICATION RESULTS

Model	NL	NN	Accuracy [%]		Precision		Recall		F1 score		Training time [s]	
			CE	*MSE*	*CE*	*MSE*	*CE*	*MSE*	*CE*	*MSE*	*CE*	*MSE*
model1	1	5	27,419	26,114	0,063	**0,099**	0,095	0,090	0,073	0,067	76,73	65,70
model2	1	10	19,201	**28,111**	0,039	**0,099**	0,060	**0,104**	0,036	**0,091**	78,80	64,67
model3	1	25	24,347	20,891	0,072	0,060	0,085	0,072	0,071	0,059	87,13	65,77
model4	1	50	22,657	19,201	0,093	0,048	0,080	0,063	0,068	0,048	94,49	74,46
model5	1	100	18,049	21,736	0,068	0,083	0,070	0,088	0,058	0,083	105,49	85,38
model6	1	150	20,046	20,046	**0,098**	0,080	**0,114**	0,082	**0,101**	0,078	143,38	104,37
model7	1	500	16,820	24,654	0,068	0,076	0,071	0,092	0,066	0,080	297,81	337,24
model8	2	5, 5	25,346	26,344	0,053	0,059	0,085	0,087	0,062	0,049	148,56	115,01
model9	2	10, 10	24,117	19,432	0,070	0,085	0,089	0,060	0,075	0,033	153,46	116,14
model10	2	25, 25	**28,034**	27,650	**0,106**	0,084	**0,110**	**0,101**	**0,101**	**0,086**	164,91	118,16
model11	2	50, 50	24,962	24,270	0,080	0,068	0,093	0,091	0,078	0,071	183,36	127,76
model12	2	100, 100	24,654	21,889	**0,136**	**0,098**	0,091	0,091	0,080	**0,088**	208,68	145,71
model13	2	150, 150	19,662	18,126	0,081	0,083	0,084	0,082	0,078	0,077	294,21	196,66
model14	2	500, 500	20,891	24,654	0,083	0,082	0,085	0,093	**0,081**	0,074	643,09	805,44
model15	3	5, 5, 5	21,121	20,968	0,054	0,042	0,067	0,069	0,036	0,050	183,54	158,84
model16	3	10, 10, 10	27,650	27,189	0,048	0,049	0,092	0,088	0,061	0,050	212,75	161,01
model17	3	25, 25, 25	22,657	21,813	0,029	0,014	0,066	0,062	0,030	0,022	234,74	157,26
model18	3	50, 50, 50	**28,725**	24,424	0,045	0,032	0,093	0,074	0,053	0,041	248,66	174,92
model19	3	100, 100,100	**29,416**	**29,339**	0,037	0,063	**0,096**	0,097	0,053	0,063	299,14	209,79
model20	3	150, 150, 150	27,727	**29,032**	0,061	0,066	0,095	**0,098**	0,073	0,066	416,82	277,95
model21	3	500, 500, 500	21,045	21,582	0,025	0,073	0,061	0,083	0,027	0,073	962,75	1245,97
model22	3	25, 50, 100	16,743	23,118	0,025	0,035	0,063	0,068	0,022	0,035	272,30	198,35
model23	3	100, 50, 25	27,573	24,424	0,041	0,070	0,088	0,088	0,053	0,070	276,73	186,82

NL-Number of Layers, NN-Number of Neurons per Layer, CE – Cross Entropy

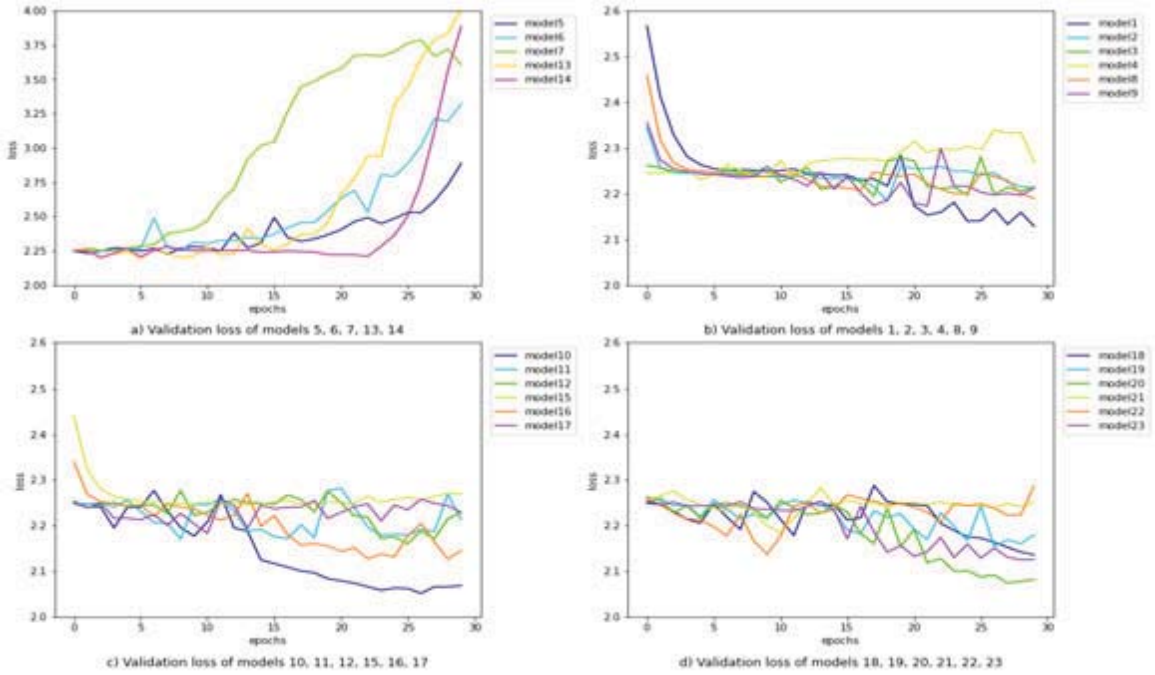

Figure 1. Validation cross-entropy loss

Networks with two LSTM layers also do not predict well with a high number of neurons. Unlike single-layer LSTM networks, they start to overfit when having 150 or 500 neurons. Networks with three LSTM layers have shown signs of overfitting when having 500 neurons per layer. Three-layered LSTMs with five neurons per layer produced poor results in terms of accuracy, but for the all layer sizes between 5 and 500 they gave satisfactory results (especially when considering training time)When comparing results for MSE loss and Cross-entropy loss, there is no significant difference. This can be seen in Fig. 2 and Fig. 3, which show 5-period simple moving averages (5-SMA) of validation accuracies of models trained with both cross-entropy (C-E) and mean squared error (MSE) loss functions. Two interesting exceptions are models 7 and 14 (one layer with 500 neurons, and two layers with 500 neurons each) where models trained with mean squared error performed quite well and have not shown any significant signs of overfitting, unlike models trained with cross-entropy.

In Fig. 4 both loss functions are shown for model 1, normalized to the interval between 0 and 1, in order to give better insight into the difference (or similarity) between them.

Authorized licensed use limited to: University of Zagreb: Faculty of Electrical Engineering and Computing. Downloaded on March 20,2021 at 15:56:30 UTC from IEEE Xplore. Restrictions apply.

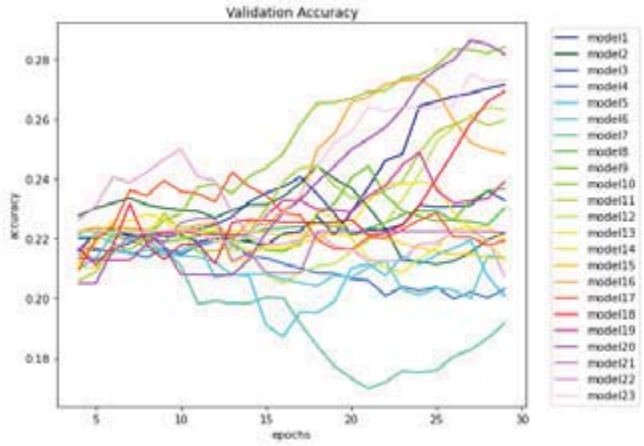

Figure 2. 5-SMA of validation accuracy with C-E loss

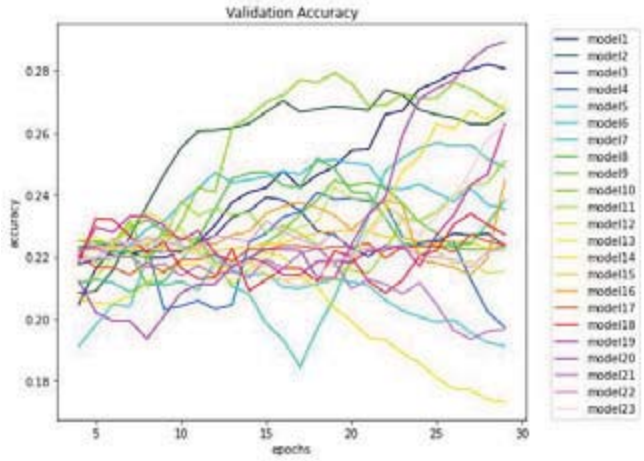

Figure 3. 5-SMA of validation accuracy with MSE loss

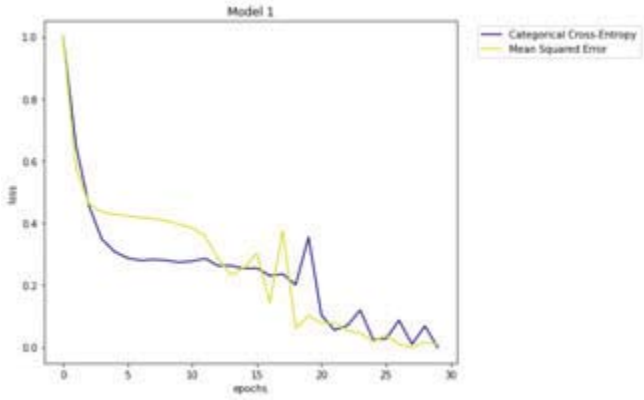

Figure 4. Comparison of C-E and MSE for Model 1

When it comes to average metrics values, they are very similar and we cannot determine which one is better, which is unexpected given the fact that cross-entropy is considered better performing in classification tasks.

As already mentioned, multiclass text classification can be tackled also with a combination of several binary classifiers. We have trained several models which use only binary classifiers for comparison with the multiclass models. Each model consists of four binary classifiers, one for each of personality preferences: introversion(I)/extraversion(E), sensing(S)/intuition(N), thinking(T)/feeling(F) and judging(J)/perceiving(P).

Across all models, the classificator T/F outperformed other classificators (I/E, S/N, J/P) which had problems with overfitting from epoch 5 onwards, as shown on Fig. 5 for model 8. For this reason, we have reduced the learning rate from 0.001 to 0.0001. Models with a higher number of layers as well as models with a higher number of neurons per layer showed a better F1 score.

LSTM network models for T/F prediction outperformed other models, but it is worthwhile to note that for T/F the dataset was quite balanced (3.981 posts for T and 4.693 posts for F), whereas all other classes were imbalanced[3] This is the reason the T/F classifier was chosen as the representative for binary classification models. Based on the results for T/F, we can presume all other classifiers would work similarly well on a larger and more balanced dataset in terms of relative number of binary labels.

The experiment results are shown in Table II. Models with 2 LSTM layers had the highest average accuracy, but models with only 1 layer showed the highest average F1 macro score. On average, simpler models outperformed other binary classificators. As an illustration we showcase the accuracy (Fig. 5) and loss (Fig. 6) for model 8 as an example of the typical behavior of binary classifiers.

However, the model which gave the best results on the test dataset was model 18, with F1 macro score of 0.716 and accuracy of 71.81%, followed by the model 7, with F1 macro score of 0.712 and accuracy of 72.04%. The number of neurons per layer had no significant impact. The 5-SMA of the loss for all T/F models is shown in Fig.7.

[3] I 6.675, E 1.999, P 5.240, J 3.434, N 7.477, S 1.197

Authorized licensed use limited to: University of Zagreb: Faculty of Electrical Engineering and Computing. Downloaded on March 20,2021 at 15:56:30 UTC from IEEE Xplore. Restrictions apply.

TABLE II. BINARY CLASSIFICATION RESULTS

Model	Number of layers	Number of neurons	Accuracy [%]	Precision	Recall	F1 score	Training Time [s]
model1	1	5	56,605	0,559	0,559	0,559	80,22
model2	1	10	66,052	0,656	0,653	0,653	74,35
model3	1	25	64,900	0,644	0,641	0,641	76,65
model4	1	50	53,763	0,509	0,506	0,469	84,01
model5	1	100	55,991	0,555	0,554	0,555	99,37
model6	1	150	58,986	0,675	0,621	0,567	135,18
model7	1	500	**72,043**	**0,720**	**0,711**	**0,712**	295,60
model8	2	5, 5	62,366	0,628	0,629	0,624	123,39
model9	2	10, 10	70,276	0,700	0,700	0,700	127,06
model10	2	25, 25	56,912	0,584	0,526	0,447	130,03
model11	2	50, 50	**70,353**	0,704	**0,706**	**0,703**	143,91
model12	2	100, 100	55,223	0,526	0,500	0,359	173,72
model13	2	150, 150	68,356	0,695	0,693	0,683	256,33
model14	2	500, 500	58,909	0,585	0,563	0,545	607,40
model15	3	5, 5, 5	66,206	0,680	0,675	0,661	177,29
model16	3	10, 10, 10	52,995	0,678	0,571	0,470	181,15
model17	3	25, 25, 25	55,376	**0,777**	0,502	0,360	183,98
model18	3	50, 50, 50	**71,813**	0,716	**0,717**	**0,716**	199,23
model19	3	100, 100, 100	56,068	0,547	0,541	0,532	255,29
model20	3	150, 150, 150	66,743	0,670	0,651	0,649	377,69
model21	3	500, 500, 500	**70,353**	**0,722**	0,683	0,681	945,07
model22	3	25, 50, 100	55,223	0,276	0,500	0,356	217,40
model23	3	100, 50, 25	59,601	0,652	0,555	0,492	234,87

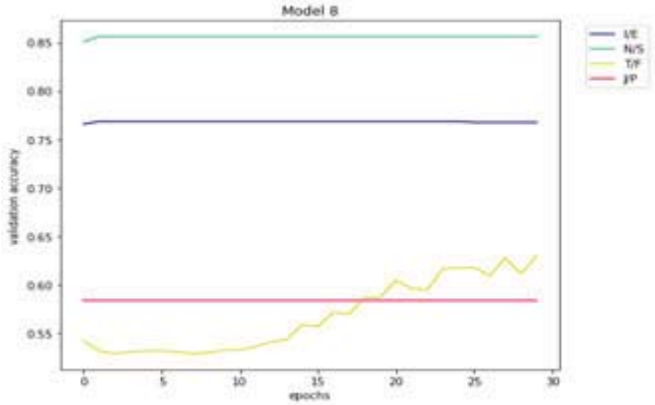

Fig 5. Validation accuracy of model 8 for 4 binary classification problems

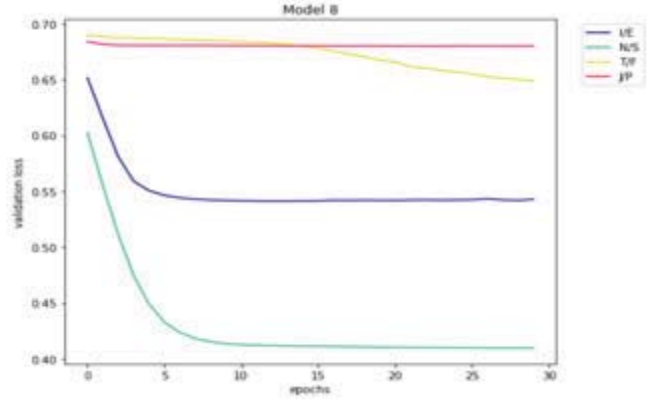

Fig 6. Validation loss of model 8 for 4 binary classification problems

Authorized licensed use limited to: University of Zagreb: Faculty of Electrical Engineering and Computing. Downloaded on March 20,2021 at 15:56:30 UTC from IEEE Xplore. Restrictions apply.

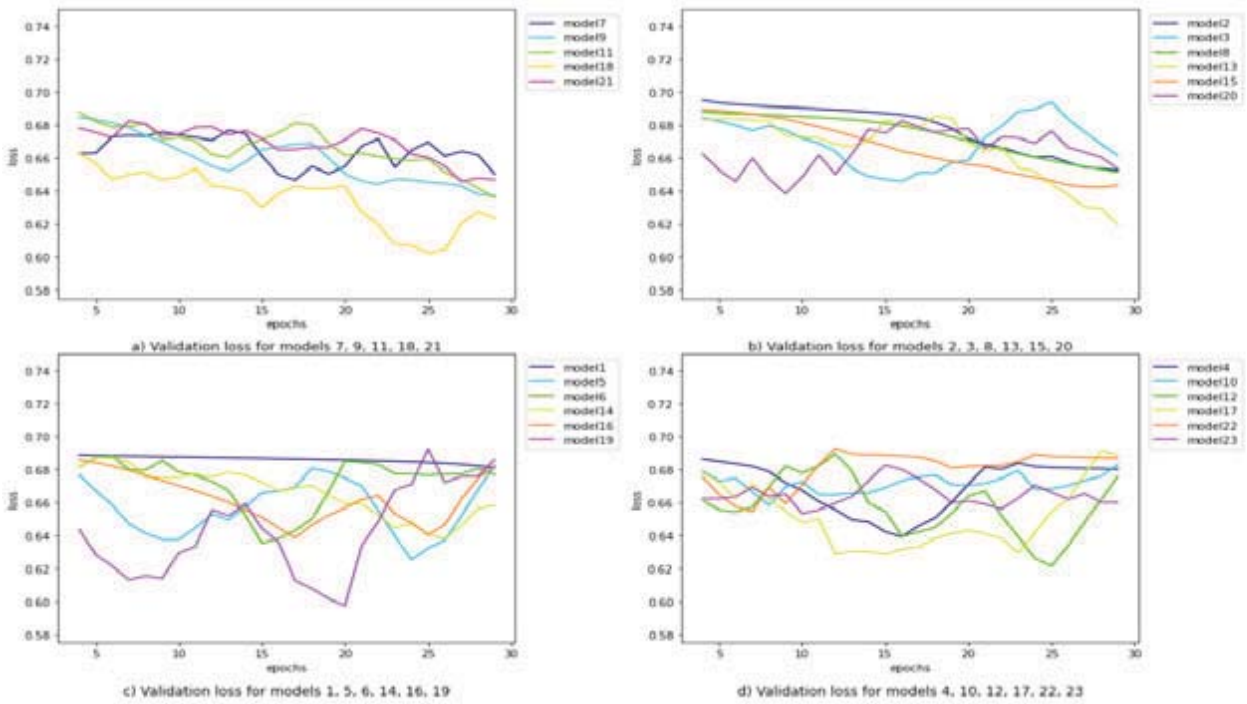

Fig. 7. 5-SMA of validation loss of different models for T/F prediction

V. CONCLUSION

Comparing the binary models with multiclass models, in both cases, simpler models gave better F1 scores, whereas models with a higher number of neurons had a higher tendency to overfit.

It is in line with our expectations that models with more layers and an increasing number of neurons per layer outperform models with a decreasing number of neurons per layer. However, model 22 did not behave according to that expectations, and such behavior can be a topic for future research.

The major issue we have detected was the imbalance of the dataset. A larger and more balanced dataset would lead to better results. However, the annotation of such a dataset would incur a significant cost as MBTI annotation requires highly skilled annotators, unlike sentiment analysis or similar tasks. Our results show that a more balanced dataset would lead to better results, as it was also emphasized by [6]. This was showcased by the T/F binary classifier, which operated on a well-balanced subset and achieved good accuracy results.

An alternative approach to the dataset imbalance problem would be to use class weights, as suggested in [7], [13]. We leave these ideas as topics for future research.

REFERENCES

[1] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning" in Nature, vol. 521., pp. 436-444, 2015.

[2] B. R. Strickland, "The Gale Encyclopedia of Psychology", 2nd edition, Gale Group, 2001

[3] D. Liu, Y. Li and M. A. Thomas, "A Roadmap for Natural Language Processing Research in Information Systems", Hawaii International Conference on System Sciences, 2017.

[4] A. Shertinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network" in the Elsevier journal "Physica D: Nonlinear Phenomena", vol. 404, 2020.

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" in Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014.

[6] Z. Zhou and X. Liu, "Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem", IEEE Transactions on Knowledge and Data Engineering, vol. 18, pp. 63-77, 2006.

[7] A. U. Rehman, A. K. Malik, B. Raza and W. Ali, "A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis", in Multimedia Tools and Applications, 2019.

[8] Y. Bengio, "Practical Recommendations for Gradient-Based Training of Deep Architectures", version 2., 2012.

[9] J. Pennington, R. Socher and C. D. Manning, "GloVe: Global Vectors for Word Representation", 2014.

[10] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization", 2015.

[11] Sepp Hochreiter and J¨urgen Schmidhuber. 1997. Long Short-Term Memory.Neural Computation, 9(8):1735–1780, November

[12] R. Hernandez and I. S. Knight, 'Predicting Myers-Briggs Type Indicator with Text Classification', https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6839354.pdf

[13] M. Ren, W. Zeng, B. Yang, R. Urtasun, "Learning to Reweight Examples for Robust Deep Learning,". https://arxiv.org/pdf/1803.09050.pdf

[14] A. Moreno, A. Esuli, F. Sebastiani, "Word-Class Cmbeddings for Multiclass Text Classifictaion", https://arxiv.org/pdf/1911.11506.pdf

Authorized licensed use limited to: University of Zagreb: Faculty of Electrical Engineering and Computing. Downloaded on March 20,2021 at 15:56:30 UTC from IEEE Xplore. Restrictions apply.