Fact or Fake News: True, Partly True, and False News Classification

Anonymous ACL submission

1 Introduction

For better or for worse, social media has had a significant impact on content and information distribution. Since anyone has the ability to post anything at anytime, there comes the cost of an increase in the spread of fake news articles and claims. Detecting whether an article is promoting fake news or not is difficult since the claim's goal is often to mislead readers to some false conclusion. Therefore, being able to fact check claims by analyzing relevant articles can be beneficial as it would allow systems to promote true and relevant news. Ideally, this would reduce public exposure to fake news and deter its spread. In this project, we examine this challenge and evaluate how natural language processing (NLP) models can be used for the task of fake news classification. Our goal is to train a variety of linear, ensemble, and neural network models and use pre-processing techniques that we've seen in class to achieve high classification accuracy. By training and evaluating models on a news dataset provided by the 2019 online competition 'Leaders Prize: Fact or Fake News', we demonstrate several NLP models' effectiveness at classifying false, partly true, and true news. We show that our methods can successfully detect fake news but would require more work in true news identification.

2 Related Work

News and headline agreement. In a workshop paper by Riedel et al. [1], the authors were concerned with news and headline agreement. Instead of predicting whether a claim is true or not, their challenge was to identify whether a headline agrees, disagrees, discusses or is unrelated to a body of text. However, they do not address their data imbalance problem. Given a headline and a body text, they predict 4 labels, ('agree', 'disagree', 'discuss', or 'unrelated'). Their system uses three input features: TF and TF-DF features of a title and its body, and a cosine similarity score between the two, and a fully connected layer from a neural network with RELU activations [2]. They achieved 81% accuracy.

Fake news detection. Esmaeilzadeh et al. explore a family of neural network based text summarization models and use the extracted features from the networks to train a fake news detection system [3]. Their classifier architecture contains a simple, uni-directional Long Short-Term Memory (LSTM) network. The authors achieve 93% accuracy. However, this task was only for binary classification (true vs false), unlike [1] or our current challenge.

Karimi et al. present a 'Hierarchical Discourse-Level Structure' for fake news detection [4]. The authors develop a Bidirectional-LSTM (Bi-LSTM) that builds sentence vectors which represent each sentence in the corpus. They achieve an accuracy of 82.19% for classifying fake from true news in their dataset. The authors develop a discourse parse tree that models the dependencies between sentences using a dynamic programming computational linguistics method, similar to the probabilistic parsing presented briefly in class. The collection of sentence probabilities in the discourse tree are used to construct embedding vectors of a body of text. Afterwards, the sentence vectors are used as inputs to a Bi-LSTM classification network. The experiment compares different embedding vectors and feature vectors, and concluded that their new model surpasses classic features, such as N-gram and Linguistic Inquiry and Word Count (LIWC).

Model	F1 Score
Naive Bayes	0.382
Bi-LSTM with Attention	0.494
BERT	**0.511**

Table 1: Competition Baseline Model Results

2.1 Competition Baselines

This is a novel dataset with little previous work done on it. The only previous work involves the competition's committee releasing baseline measures for the dataset. The competition's committee released three baseline model descriptions and their respective F1 scores (a classification measure of both recall and precision). The scores are available within Table 1. While there are no specific details about their models' parameters, we intend to compare our trained models' effectiveness to their baselines. We use several linear, ensemble and custom neural network models that differ from theirs.

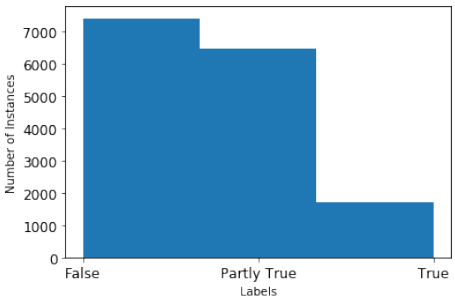

Figure 1: Labels Histogram

3 Data Description

The source of our labelled dataset is from the 2019 online competition called 'Leaders Prize: Fact or Fake News?' [5]. The competition involves the task of classifying news claim within its provided dataset as either: False, Partly True, or True. The dataset is comprised of labelled claims (i.e. statements), claimants, and related article content. Claimants are related to claims, however not all claims have a claimant. The data contains a total of **15,555 claims**, and **64,974 related articles**. Therefore, there is a total of 15,555 examples to train and evaluate our models with. The competition has a hidden test set that can only be evaluated on when making a submission. We did not make use of this test set because the competition submission closed before we had the chance to make any.

3.1 Data Imbalance

Upon further analysis of our data, we noticed a severe class imbalance as illustrated in Figure 1. The majority of our data is of False and Partly True claims. More precisely our examples are labelled: 48% False, 41% Partly True and 11% True. We believe if this imbalance is not addressed, we expect to see a significantly low recall within our results. We discuss a method to circumvent this imbalance in Section 4.7.

4 Methodology

Here we explain the different pre-processing techniques, feature extraction methods and models that we used and evaluated. For this project, we are using the Python 3 programming language and several external Python packages: Pandas [6], SciPy [7], Matplotlib [8], Tqdm [9], Scikit-Learn [10], Tensorflow [11], Keras [12], PyTorch [13] and XGBoost [14].

4.1 Data Split

Since the competition's test set is hidden from us, we made use of splitting the training data that was provided. The split is a ratio of 70%, 15% and 15% for the training, development and test sets respectively.

4.2 Data Pre-processing

Next we describe the techniques we've used for pre-processing our data. The methods used are the following: square bracket removal, non-ASCII character removal, punctuation removal, character lowercasing, numerical character replacement, stemming and lemmatization.

These techniques are applied on claims and related article content. The majority of claimants are proper nouns, so we didn't apply pre-processing on claimants. We applied non-ASCII character removal to clean our corpus from unordinary characters. Character lowercasing is applied because we don't want our models to differentiate between words of the same spelling but different capitalization. Stemming and lemmatization are both performed to make our feature space dimensionally lower, with the intent of making our models faster to train and more generalizable.

4.3 Training and Evaluation

We tuned our models using the training and development sets to determine optimal hyperparameter configurations. Then we trained the optimally parameterized model on the training and validation sets, and finally evaluated it on the held-out test set. By doing so, we can empirically determine optimal hyperparameter configurations by detecting overfitting and use the test set to have an unbias evaluation of our models.

4.4 Most Frequent Label Classifier Baseline

For comparison purposes, we have tested a Most Frequent Label Classifier baseline. Given a set of test examples, the predictor classifies each example to that of the most frequently occurring label. This was implemented using SciKit-Learn's DummyClassifier application programming interface (API) . This is not meant to be used as an effective predictor, yet a critical baseline for an unbalanced dataset.

4.5 Linear Model Baselines

We trained and evaluated three linear models as baselines to compare with more complex models that we eventually train and evaluate.

We trained and evaluated three different multi-class linear models: Logistic Regression, Support Vector Machine (SVM) and Naive Bayes. Logistic Regression and SVM are discriminative models whereas Naive Bayes is a generative model. Generative models estimate the joint distribution of features and labels whereas discriminative models estimate the posterior probability of some label given its features. The difference between Logistic Regression and SVM is the different objective functions used at training. They both estimate an hyperplane that can separate the training labels' categories. The subtle difference is that SVMs attempt to maximize the margin (i.e. distance) between its estimated hyperplane and the observations from each class [15] whereas Logistic Regression performs Maximum Likelihood Estimation (MLE) to maximize the likelihood that a random point gets classified correctly [16]. The models are trained on Term Frequency-Inverse Document Frequency (TF-IDF) representations

of the concatenation of the claim, claimant, and related article content.

We fine tuned each linear models' parameters with Grid Cross Validation of Fold 3. The model configurations, validation F1 scores, and test F1 scores are available within Table 3. We trained, fine-tuned and evaluated these models using SciKit-Learn's relevant predictor, Grid Cross Validation and metric APIs.

4.6 Ensemble Models: Random Forest and Extreme Gradient Boosting (XGBoost)

Ensemble models are popularly used in these sorts of classification challenges and are often considered the state-of-the-art solution [17]. The main premise is that by combining multiple models, the errors of a single base-learner will likely be compensated by other base-learners, as a result the overall prediction performance would be better than that of a single model.

A Random Forest is decision-tree-based ensemble model [18]. There are two main components to this ensemble. The first is that each tree is trained on a random sample from the training data. The second is that a random subset of features are selected to generate a split for each node in a tree.

Similarly to a Random Forest, XGBoost is a decision-tree-based ensemble model, however it uses a gradient boosting framework [19]. XGBoost trains an ensemble of base-learners and boosts weak learners using gradient descent. It also uses some system optimizations and algorithmic enhancements to make it faster to train compared to its sister model: Gradient Boosting Machine (GBM) [19].

Similarly to the linear models, we fine tuned the ensemble models using a Grid Cross Validation of Fold 3. Their results are available within Table 3. The models are trained on TF-IDF representations of the concatenation of the claim, claimant, and related article content. We used SciKit-Learn's and XGBoost's relevant predictor APIs.

4.7 Fully Connected MLP Neural Network Baseline

We re-implement the simple fully connected neural network, or multi-layer perceptron (MLP) developed by Riedel et al. [1] using PyTorch, in order to verify whether their method would transfer to our fake news dataset.

The input layer samples from our customized PyTorch `Dataset` class, which concatenates a 5,000 dimensional TF vector from claim, and another 5,000 dimensional TF vector from related texts, and a cosine similarity score of the TF-IDF vectors of the two, according to Riedel et al. [1]'s hyperparameters. The paper's architecture only yields around 35% validation accuracy.

Upon examining the TF and TF-IDF vectors, only around 500 to 1,000 are non-zero. We then decide to limit the dimensions to 500 in the TF and TF-IDF functions. Then, we apply principal component analysis (PCA) to the 500 dimensional TF vectors, and obtain 64 principle components from claim and text.

In order to address the data imbalance problem, we further modify a plain PyTorch `Dataset` using a `WeightedRandomSampler`. This ensures that each batch samples the rare class with a higher probability, and each batch contains approximately equal number of samples from each class.

We also experiment with various loss metrics utilized by computer vision experiments in handling data imbalance, such as dice loss [20] and focal loss [21]. However, the metrics that are suitable for vision tasks do not transfer to NLP applications. Binary cross entropy loss is the best out of all metrics we tried. Weighted batch sampling and binary cross entropy loss improve our validation set unweighted F1 score from 35% to 45%.

4.8 Sequential Neural Network Model: Bidirectional Long Short-Term Memory (Bi-LSTM)

The final model we test is a custom neural network model inspired from the Bi-LSTM architecture. The idea behind this model is to leverage simpler model for metadata features and use more complex model such as LSTM to extract relevant information from more complex features like claim and related articles.

Metadata features such as the claimant, the number of articles supporting the claim are features already in a format that can be passed to the model and are passed directly to the model via a standard MLP. For the claim and related articles, we first preprocess the text like described in section 4.2, then convert the preprocessed text to a fixed length vector using the text to sequence and padding implementations of Keras. Then, we pass both the claim resulting vector and the related article resulting vector to two embedding layers. The goal of using embedding layers is convert sparse vector to dense vector. Finally, we concatenate the two dense vectors and pass the resulting concatenated vector to a bi-lstm model. To obtain a final prediction, we concatenate the output of the bi-lstm and of the MLP and pass everything through a softmax layer. Model architecture is shown in figure 2.

We expect that this model, if trained properly, should be the most performing one due to the sequential nature of the model. To train the model, we used the Nadam optimizer presented in [22]. This optimizer is a mix between two well known optimizers: Nesterov and Adam.

5 Results

To evaluate our models' performance, we are concerned with Weighted Average F1 and Recall for the 'True' class. We are particularly interested by the Recall of the True category since it tells us how much of the truthful news our system was able to detect within

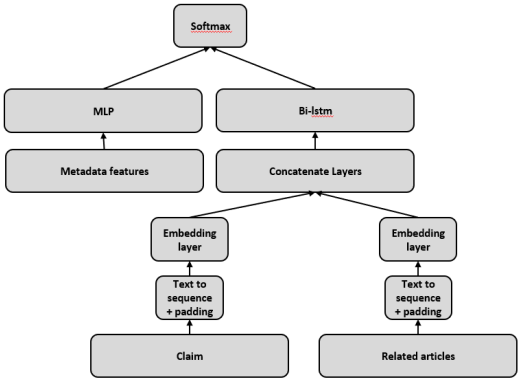

Figure 2: Custom Bi-LSTM Model Architecture

Model	Weighted Average F1 Score (%)	Recall of 'True' Class (%)
Most Frequently Occurring Label Classifier	31.0	0.0
Logistic Regression	58.0	1.0
Linear SVM	57.0	0.0
Naive Bayes	53.0	0.0
Random Forest	60.0	2.0
XGBoost	56.0	1.0
MLP	40.1	**17.0**
Bi-LSTM Custom	**60.0**	0.0
Bi-LSTM Rebalanced	57.0	5.0

Table 2: Experiment F1 Scores

the dataset. If some system like this were to be deployed, we would want it to be able to detect 'True' news from 'False' ones with a high success rate. We would not simply censor all news because it was incorrectly deemed as 'False.' A more detailed look at precision, recall and F1 scores are available in Table 2. It is also the measure that highlights a lack of performance in handling an unbalanced dataset.

Next, we compare our different methods and discuss potential reasons behind their performance.

5.1 Linear Model Baselines

With respect to our linear model baselines, although they all performed similarly, Logistic Regression ranks best with a Weighted Average F1 of 58%. We believe this is due because it makes less assumptions as opposed to Naive Bayes and its conditional independence between output categories, and is less complex than an SVM.

5.2 Ensemble Models

As for the ensemble models, surprisingly Random Forest outranks XGBoost by Weighted Average F1 Scores. XGBoost has a higher number of parameters than the Random Forest ensemble, so one possible reason why it was outranked was because it wasn't fine-tuned enough given its higher complexity in hyperparameter configuration.

5.3 Neural Network: MLP Baseline

Upon close examination of results presented in Riedel et al. [1], we note that they also have a data class imbalance problem. However, they solely report a reweighted accuracy score and accuracy, which are what their competition measure. From their results table, we calculate that their method yields roughly 57% precision and 65% recall overall. Only the more frequent classes achieve around 80% precision, whereas the less frequent class has only 6.60% precision.

The 80% accuracy metric alone is extremely misleading. The precision and recall score is however, close to what we obtain using a similar model structure.

Despite overall lower F1 score, MLP proves to be robust in handling the rare 'true' class instances, compared to other methods with higher F1 scores. Our reweight data sampler helps in training MLP handling this rare class. Another justification would be the universal approximation property of MLP, where a hidden layer of size n is capable of representing continuous functions in $\mathbb{R}^n$ [23]. Compared to the linear models, this property makes MLP much more robust in predicting rare instances.

5.4 Bi-LSTM

The custom bi-LSTM model is one of our best performing model with a weighted average F1 score of 60%. Similarly to other models, the model has difficulty predicting the 'True' class. The good overall performance of the model comes from a by class F1 accuracy of 70% and 60% for the False and 'Partly False' category. In order to improve the F1 score for the 'False' category, we tried training the same model but with some epochs on a rebalanced dataset. This improved the 'False' F1 score from 0% to 5% but did decrease the performance on other class for a total macro average F1 score of 57%.

5.5 Comparison with competition baselines

As shown in Table 1, the competition baseline reach numbers a lot lower than our numbers. They do not give access to the validation set they used to obtain those numbers nor give details on this set. We hypothesis that the validation set they used has to be harder to get good results than the one we had. One reason might be that their validation set is more proportional between classes so that results are lower due to poor performance for the 'True' class.

6 Discussion

Our models are able to detect fake news ('False' claims) and classify 'False' claims from 'Partly True'

ones. The MLP model can achieve 17% 'True' recall score, but lacking on other classes compared to the rest. Other models are more robust than MLP at detecting 'False' from 'Partly True' ones, but their 'True' Recall scores are less than 5%. We struggle to train a single model that can handle both cases above. For a fake news classifier system like this to be deployed, we believe there is more work to be done in developing systems to detect truthful news from fake.

Another major challenge is handling unbalanced corpora. The experiment in Section 4.7 shows that metrics used in computer vision tasks with unbalanced dataset do not transfer well in NLP tasks. This problem needs a tailored metric, such as adaptive softmax and loss in Grave et al. [24], to address NLP specific challenges instead of using purely information theory (entropy) or statistics.

7 Conclusion

To conclude, we train a variety of linear, ensemble, and neural network models to perform multi-class classification on the unbalanced dataset from 2019 Leader's Prize Data Cup. The best performing models in terms of class weighted F1 score is Bi-LSTM, and the best in rarest class F1 score is MLP, achieving 60% and 17% respectively.

Our proposed methods can classify false claims from partially-true ones, the two most frequent classes, with up to 70% accuracy. Although our proposed methods struggle to detect the rarest class, all our models beat the most frequent classifier baseline in predicting this extremely unbalanced dataset, judging from weighted F1 score.

We show that one cannot straightforwardly use some out-of-the-box machine learning model, such as Logistic Regression, Random Forest, or Bi-LSTM, for this problem and expect it to perform significantly well. Though our models underperformed in this work, the above-baseline performance still holds potential in addressing the pressing challenge of fake news detection.

8 Statement of Contributions

All team members contributed equally on all phase of the project. Violet wrote MLP in Pytorch, conducted literature review, and edited the report. Matthew worked on pre-processing, trained ensemble models, trained linear models, and edited the report. Etienne worked on the data pre-processing, training of linear models and the bi-lstm model and also edited the report.

Appendices

Model	Parameters Tested	Best Parameters	Validation F1	Test F1
Logistic Regression	Penalty = ['l1', 'l2'] C = [1, 2, 5]	Penalty = 'l2' C = 1	0.580	0.580
Linear SVM	C = [1, 5, 10]	C = 1	0.580	0.580
Naive Bayes	Alpha = [0.25, 0.5, 0.75, 1.0]	Alpha = 0.25	0.560	0.560
Random Forest	Estimators = [100, 200, 500]	Estimators = 500	0.590	0.600
XGBoost	Estimators = [100, 200, 500]	Estimators = 200	0.570	0.560

Table 3: Linear Models: Grid Cross Validation Configurations and Scores

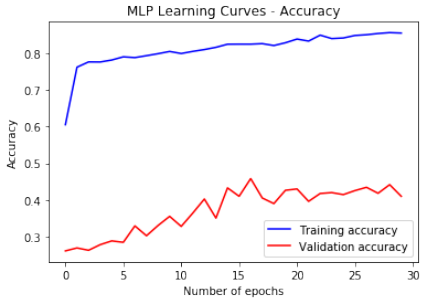

Figure 3: Our MLP using TF vectors and cosine similarity, accuracy curve

Logistic Regression			
Class	Precision	Recall	F1
False	0.63	0.73	0.68
Partly True	0.60	0.64	0.62
True	0.67	0.01	0.02
Weighted Average	0.62	0.62	0.58
SVM			
Class	Precision	Recall	F1
False	0.63	0.71	0.67
Partly True	0.59	0.65	0.62
True	0.00	0.00	0.00
Weighted Average	0.54	0.61	0.57
Naive Bayes			
Class	Precision	Recall	F1
False	0.56	0.85	0.68
Partly True	0.62	0.42	0.50
True	0.00	0.00	0.00
Weighted Average	0.61	0.61	0.53
Random Forest			
Class	Precision	Recall	F1
False	0.64	0.77	0.70
Partly True	0.62	0.63	0.63
True	0.28	0.02	0.04
Weighted Average	0.59	0.63	0.60
XGBoost			
Class	Precision	Recall	F1
False	0.63	0.71	0.66
Partly True	0.58	0.64	0.61
True	1.00	0.01	0.02
Weighted Average	0.65	0.60	0.57
MLP			
Class	Precision	Recall	F1
False	0.49	0.42	0.45
Partly True	0.42	0.41	0.42
True	0.13	0.23	0.17
Weighted Average			0.40
Bi-LSTM Custom			
Class	Precision	Recall	F1
False	0.63	0.69	0.70
Partly True	0.56	0.64	0.60
True	0.00	0.00	0.00
Weighted Average			0.60
Bi-LSTM Re-balanced			
Class	Precision	Recall	F1
False	0.57	0.85	0.68
Partly True	0.61	0.36	0.46
True	0.17	0.05	0.08
Weighted Average			0.57

Table 4: Test Set Scores

References

[1] Benjamin Riedel et al. "A simple but tough-to-beat baseline for the Fake News Challenge stance detection task". In: *CoRR* abs/1707.03264 (2017). arXiv: 1707.03264. URL: http://arxiv.org/abs/1707.03264.

[2] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[3] Soheil Esmaeilzadeh, Gao Xian Peh, and Angela Xu. *Neural Abstractive Text Summarization and Fake News Detection*. 2019. arXiv: 1904.00788 [cs.CL].

[4] Hamid Karimi and Jiliang Tang. "Learning Hierarchical Discourse-level Structure for Fake News Detection". In: *CoRR* abs/1903.07389 (2019). arXiv: 1903.07389. URL: http://arxiv.org/abs/1903.07389.

[5] *Data Cup Leaders Prize: Fact or Fake News?* https://www.datacup.ca/main/competitions/leadersprize2019/about. Accessed: 2019-10-21.

[6] Wes McKinney. "pandas: a Foundational Python Library for Data Analysis and Statistics". In: ().

[7] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 51–56.

[8] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[9] Casper O. da Costa-Luis. "tqdm: A Fast, Extensible Progress Meter for Python and CLI". In: *Journal of Open Source Software* 4.37 (May 2019), p. 1277. DOI: 10.21105/joss.01277. URL: https://doi.org/10.21105/joss.01277.

[10] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[11] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[12] François Chollet. *keras*. https://github.com/fchollet/keras. 2015.

[13] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[14] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[15] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[16] Raymond E Wright. "Logistic regression." In: (1995).

[17] Omer Sagi and Lior Rokach. "Ensemble learning: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1249.

[18] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[19] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM. 2016, pp. 785–794.

[20] Ken C. L. Wong et al. "3D Segmentation with Exponential Logarithmic Loss for Highly Unbalanced Object Sizes". In: *Lecture Notes in Computer Science* (2018), pp. 612–619. ISSN: 1611-3349. DOI: 10.1007/978-3-030-00931-1_70. URL: http://dx.doi.org/10.1007/978-3-030-00931-1_70.

[21] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2017. arXiv: 1708.02002 [cs.CV].

[22] Timothy Dozat. "Incorporating Nesterov Momentum into Adam". In: (2016). URL: https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ.

[23] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(91)90009-T. URL: http://www.sciencedirect.com/science/article/pii/089360809190009T.

[24] Edouard Grave et al. "Efficient softmax approximation for GPUs". In: *CoRR* abs/1609.04309 (2016). arXiv: 1609.04309. URL: http://arxiv.org/abs/1609.04309.