The Journal of Open Source Software

pytorch-widedeep: A flexible package for multimodal deep learning

Javier Rodriguez Zaurin[1] and **Pavol Mulinka**[2]

1 Independent Researcher, Spain **2** Centre Tecnologic de Telecomunicacions de Catalunya (CTTC/CERCA), Catalunya, Spain

DOI: 10.21105/joss.05027

Software

- Review
- Repository
- Archive

Editor: Øystein Sørensen

Reviewers:

- @siboehm
- @makoeppel

Submitted: 13 November 2022
Published: 15 June 2023

Summary

In recent years datasets have grown in size and diversity, combining different data types. Multimodal machine learning projects involving tabular data, images and/or text are gaining popularity (e.g. Garg et al. (2022)). Traditional approaches involved independent feature generation from every data type and their combination in the later stage before passing them to an algorithm for classification or regression.

However, with the advent of "*easy-to-use*" Deep Learning (DL) frameworks such as Tensorflow (Abadi et al., 2015) or PyTorch (Paszke et al., 2019), and the subsequent advances in the fields of Computer Vision, Natural Language Processing or Deep Learning for Tabular data, it is now possible to use state-of-the-art DL models and combine all datasets early in the process. This has two main advantages: (i) we can partially or entirely skip the feature engineering step, and (ii) the representations of each data type are learned jointly. This means that such representations contain information: (i) related to the target if the problem is supervised; and (ii) how the different data types relate to each other.

Furthermore, the flexibility inherent to DL approaches allows the usage of techniques primarily designed only for text and/or images to tabular data, e.g., transfer learning or self-supervised pre-training.

With that in mind, we introduce `pytorch-widedeep`, a flexible package for multimodal deep learning designed to facilitate the combination of tabular data with text and images.

Statement of need

There is a small number of packages available to use DL for tabular data alone (e.g., pytorch-tabular (Joseph, 2021), pytorch-tabnet or autogluon-tabular (Erickson et al., 2020)) or that focus mainly on combining text and images (e.g., MMF (Singh et al., 2020)). With that in mind, our goal is to provide a modular, flexible, and "*easy-to-use*" framework that allows the combination of a wide variety of models for all data types.

`pytorch-widedeep` is based on Google's Wide and Deep Algorithm (Cheng et al., 2016), hence its name. The original algorithm is heavily adjusted for multimodal datasets and intended to facilitate the combination of text and images with corresponding tabular data. As opposed to Google's *"Wide and Deep"* and *"Deep and Cross"* (R. Wang et al., 2017) architecture implementations in Keras/Tensorflow, we use the wide/cross and deep model design as an initial building block of PyTorch deep learning models to provide the basis for a plethora of state-of-the-art models and architecture implementations that can be seamlessly assembled with just a few lines of code. Additionally, the individual components do not necessarily have to be a part of the final architecture. The main components of those architectures are shown in Figure 1.

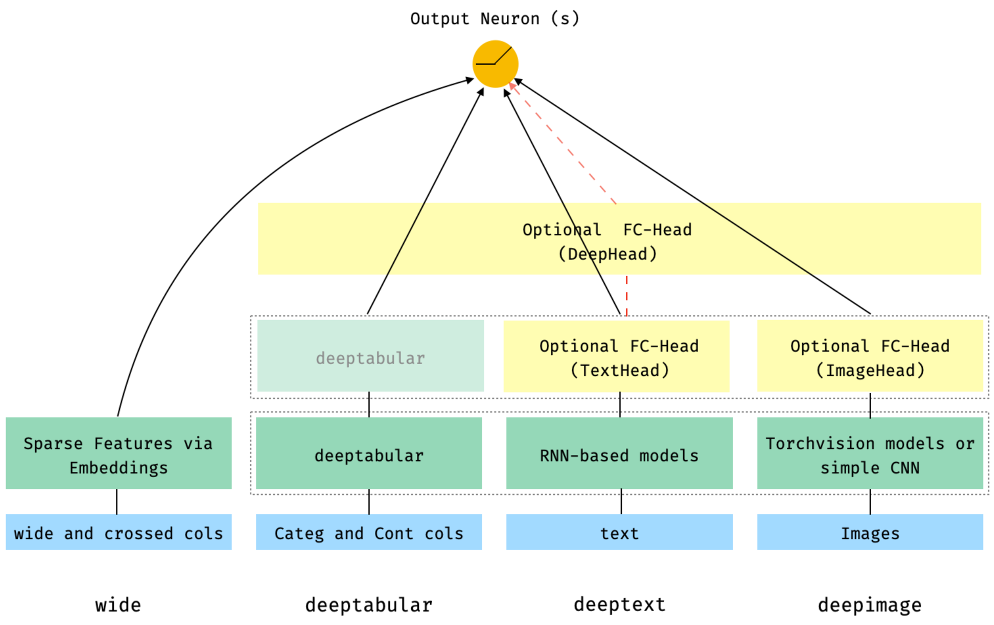

Figure 1: Main components of the pytorch-widedeep architecture. The blue and green boxes in the figure represent the main data types and their corresponding model components, namely `wide`, `deeptabular`, `deeptext` and `deepimage`. The yellow boxes represent *so-called* fully-connected (FC) heads, simply MLPs that one can optionally add on top of the main components. These are referred to in the figure as `TextHead` and `ImageHead`. The dashed-line rectangles indicate that the outputs from the components inside are concatenated if a final FC head (referred to as `DeepHead` in the figure) is used. The faded-green `deeptabular` box aims to indicate that the output of the deeptabular component will be concatenated directly with the output of the `deeptext` or `deepimage` components or with the FC heads if these are used. Finally, the arrows indicate the connections, which of course, depend on the final architecture that the user chooses to build.

Following the notation of (Cheng et al., 2016), the expression for the architecture without a deephead component can be formulated as:

$$pred = \sigma(W^T_{wide}[x, \phi(x)] + \sum_{i \in \mathcal{I}} W^T_i a^{l_f}_i + b)$$

Where $\mathcal{I} = \{deeptabular, deeptext, deepimage\}$, σ is the sigmoid function, W are the weight matrices applied to the wide model and to the final activations of the deep models, a are these final activations, $\phi(x)$ are the cross-product transformations of the original features x, and b is the bias term.

If there is a deephead component, the previous expression turns into:

$$pred = \sigma(W^T_{wide}[x, \phi(x)] + W^T_{deephead} a^{l_f}_{deephead} + b)$$

At this stage, it is worth mentioning that the library has been built with a special emphasis on flexibility. That is, we want users to easily run as many different models as possible and/or use their custom components if they prefer. With that in mind, each and every data type component in the figure above can be used independently and in isolation. For example, if the user wants to use a ResNet model to perform classification in an image-only dataset, that is perfectly possible using this library. In addition, following some minor adjustments described in the documentation, the user can use any custom model for each data type – mainly, a custom model is a standard PyTorch model class that must have a property or attribute called `output_dim`. This way, the `WideDeep` collector class knows the size of the incoming activations and is able to construct the multimodal model. Examples of how to use custom components can be found in the repository and documentation.

The Model Hub

This section will briefly introduce the current model components available for each data type in the library. Remember that the library is constantly under development, and models are constantly added to the "model-hub".

The `wide` component

This is a linear model for tabular data where the non-linearities are captured via cross-product transformations. This is the simplest of all components, and we consider it very useful as a benchmark when used on its own.

The `deeptabular` component

Currently, `pytorch-widedeep` offers the following models for the so-called `deeptabular` component:(i) TabMlp, (ii) TabResnet, (iii) TabNet (Arik & Pfister, 2021), (iv) ContextAttentionMLP (Z. Yang et al., 2016), (v) SelfAttentionMLP (X. Huang et al., 2020), (vi) TabTransformer (X. Huang et al., 2020), (vii) SAINT (Somepalli et al., 2021), (viii) FT-Transformer (Gorishniy et al., 2021), (ix) TabFastFormer: our adaptation of the FastFormer (Wu et al., 2021), (x) TabPerceiver: our adaptation of the Perceiver (Jaegle et al., 2021), (xi) BayesianWide and (xii) BayesianTabMlp (both based on Blundell et al. (2015)).

The `deepimage` component

The image-related component is fully integrated with the newest version of torchvision (TorchVision maintainers & contributors, 2016) (0.13 at the time of writing). This version has Multi-Weight Support. Therefore, a variety of model variants are available to use with pre-trained weights obtained with different datasets. Currently, the model variants supported by `pytorch-widedeep` are (i) Resnet (He et al., 2016), (ii) Shufflenet (Zhang et al., 2018), (iii) Resnext (Xie et al., 2017), (iv) Wide Resnet (Zagoruyko & Komodakis, 2016), (v) Regnet (Xu et al., 2022), (vi) Densenet (G. Huang et al., 2017), (vii) Mobilenet (A. G. Howard et al., 2017), (viii) MNasnet (Tan et al., 2019), (ix) Efficientnet (Tan & Le, 2019) and (x) Squeezenet (Iandola et al., 2016).

The `deeptext` component

Currently, `pytorch-widedeep` offers the following models for the `deeptext` component: (i) BasicRNN, (ii) AttentiveRNN and (iii) StackedAttentiveRNN. The library will be integrated with the Huggingface transformers library (Wolf et al., 2019) in the near future. However, it is worth mentioning that although transformer-based models are not natively supported by our library, these can be used easily with `pytorch-widedeep` as a custom model (please, see the documentation for details).

Forms of model training:

Training single or multi-mode models in `pytorch-widedeep` is handled by the different training classes. Currently, `pytorch-widedeep` offers the following training options: (i) "*Standard*" Supervised training, (ii) Supervised Bayesian training, and (iii) Self-supervised pre-training.

Contribution

pytorch-widedeep is being developed and used by many active community members. Anyone can join the discussion on Slack.

Acknowledgements

We acknowledge the work of other researchers, engineers, and programmers from the following projects and libraries:

- the Callbacks and Initializers structure and code is inspired by the torchsample library (TorchSample maintainers & contributors, 2017), which in itself partially inspired by Keras (Chollet & others, 2015)
- the TextProcessor class in this library uses the fastai (J. Howard & Gugger, 2020) Tokenizer and Vocab; the code at utils.fastai_transforms is a minor adaptation of their code, so it functions within this library; to our experience, their Tokenizer is the best in class
- the ImageProcessor class in this library uses code from the fantastic Deep Learning for Computer Vision (DL4CV) (Adrian, 2017) book by Adrian Rosebrock
- we adjusted and integrated ideas of Label and Feature Distribution Smoothing (Y. Yang et al., 2021)
- we adjusted and integrated ZILNloss code written in Tensorflow/Keras (X. Wang et al., 2019)

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

Adrian, R. (2017). *Deep learning for computer vision with python*. PyImageSearch.com.

Arik, S. Ö., & Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*, 6679–6687.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. *International Conference on Machine Learning*, 1613–1622.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., & others. (2016). Wide & deep learning for recommender systems. *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 7–10. https://doi.org/10.48550/arXiv.1606.07792

Chollet, F., & others. (2015). *Keras*. https://keras.io.

Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2020). *AutoGluon-tabular: Robust and accurate AutoML for structured data*. arXiv. https://doi.org/10.48550/ARXIV.2003.06505

Garg, M., Wazarkar, S., Singh, M., & Bojar, O. (2022). Multimodality for NLP-centered applications: Resources, advances and frontiers. *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, 6837–6847.

Gorishniy, Y., Rubachev, I., Khrulkov, V., & Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, *34*, 18932–18943.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. https://doi.org/10.1109/cvpr.2016.90

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv Preprint arXiv:1704.04861*.

Howard, J., & Gugger, S. (2020). Fastai: A layered API for deep learning. *Information*, *11*(2). https://doi.org/10.3390/info11020108

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708.

Huang, X., Khetan, A., Cvitkovic, M., & Karnin, Z. (2020). Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv Preprint arXiv:2012.06678*.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and$<$ 0.5 MB model size. *arXiv Preprint arXiv:1602.07360*.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., & Carreira, J. (2021). Perceiver: General perception with iterative attention. *International Conference on Machine Learning*, 4651–4664.

Joseph, M. (2021). *PyTorch tabular: A framework for deep learning with tabular data*. https://arxiv.org/abs/2104.13638

maintainers, TorchVision, & contributors. (2016). TorchVision: PyTorch's computer vision library. In *GitHub repository*. https://github.com/pytorch/vision; GitHub.

maintainers, TorchSample, & contributors. (2017). TorchSample: Lightweight pytorch functions for neural network featuremap sampling. In *GitHub repository*. https://github.com/ncullen93/torchsample; GitHub.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Singh, A., Goswami, V., Natarajan, V., Jiang, Y., Chen, X., Shah, M., Rohrbach, M., Batra, D., & Parikh, D. (2020). *MMF: A multimodal framework for vision and language research*. https://github.com/facebookresearch/mmf.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., & Goldstein, T. (2021). Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv Preprint arXiv:2106.01342*.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2820–2828. https://doi.org/10.1109/CVPR.2019.00293

Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*, 6105–6114.

Wang, R., Fu, B., Fu, G., & Wang, M. (2017). Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17* (pp. 1–7). https://doi.org/10.48550/arXiv.1708.05123

Wang, X., Liu, T., & Miao, J. (2019). A deep probabilistic model for customer lifetime value prediction. *arXiv Preprint arXiv:1912.07753*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., & others. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv Preprint arXiv:1910.03771*.

Wu, C., Wu, F., Qi, T., Huang, Y., & Xie, X. (2021). Fastformer: Additive attention can be all you need. *arXiv Preprint arXiv:2108.09084*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1492–1500. https://doi.org/10.1109/CVPR.2017.634

Xu, J., Pan, Y., Pan, X., Hoi, S., Yi, Z., & Xu, Z. (2022). RegNet: Self-regulated network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2022.3158966

Yang, Y., Zha, K., Chen, Y., Wang, H., & Katabi, D. (2021). Delving into deep imbalanced regression. *International Conference on Machine Learning*, 11842–11851.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). *Hierarchical attention networks for document classification*. 1480–1489. https://doi.org/10.18653/v1/N16-1174

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. *arXiv Preprint arXiv:1605.07146*. https://doi.org/10.5244/c.30.87

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6848–6856. https://doi.org/10.1109/CVPR.2018.00716