Depth-Wise Attention (DWAtt): A Layer Fusion Method for Data-Efficient Classification

Muhammad ElNokrashy, [*†]
Independent
muhammad.nael@gmail.com

Badr AlKhamissi, [*‡]
Responsible AI, Meta
badr@khamissi.com

Mona Diab
Responsible AI, Meta
mdiab@fb.com

Abstract

Language Models pretrained on large textual data have been shown to encode different types of knowledge simultaneously. Usually, only the features from the last layer are used when adapting to new tasks or data. We put forward that in using or finetuning deep pretrained models, intermediate layer features that may be relevant to the downstream task are buried too deep to be used efficiently in terms of needed samples or steps. To test this, we propose a new layer fusion method: Depth-Wise Attention (DWAtt), to help re-surface signals from non-final model layers. We compare DWAtt to a basic concatenation-based layer fusion method (Concat), and compare both to a deeper model baseline—all kept within a similar parameter budget. Our findings show that DWAtt and Concat are more step- and sample-efficient than the baseline, especially in the few-shot setting. DWAtt outperforms Concat on larger data sizes. On CoNLL-03 NER, layer fusion shows $3.68 - 9.73\%$ F1 gain at different few-shot sizes. The layer fusion models presented significantly outperform the baseline in various training scenarios with different data sizes, architectures, and training constraints.

1 Introduction

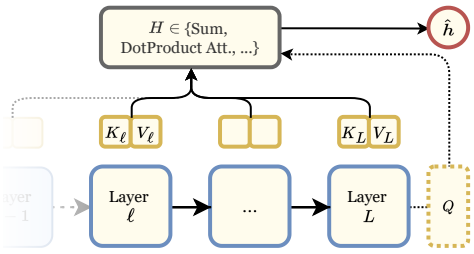


Figure 1: The Mixer H can be a Sum of Affines, a Dot-Product Attention module, etc. Different variants may define K_i, V_i, and Q (key, value, and query) and utilize them differently.

The Transformer architecture (Vaswani et al., 2017) and variants (Fan, Lavril, Grave, Joulin, & Sukhbaatar, 2020) have become a mainstream, reliable choice for a wide range of Natural Language Processing (**NLP**) tasks, such as sentence classification, token labeling, retrieval, and question answering (A. Wang et al., 2019, 2018). This is in part due to architectural properties that enable enhanced parallelization and better modeling of long-range dependencies. Several models have since

[*]Equal Contribution.
[†]Correspondence: muhammad.nael@gmail.com.
[‡]Work started independently then continued during residency at Meta AI, Seattle.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

surfaced, setting new records on different benchmarks (Lewis et al., 2020), (Brown et al., 2020), and (Devlin, Chang, Lee, & Toutanova, 2019). In the vanilla architecture, a stack of Transformer blocks are applied sequentially to refine the representation of an input sequence, which is then fed to a task-specific module, such as a classifier head.

Recent works have shown that the hidden representations from intermediate layers may benefit downstream tasks (Wallat, Singh, & Anand, 2021). Other works have tested the fusion of hidden representations in tasks such as sequence-to-sequence machine translation (F. Liu, Ren, Zhao, & Sun, 2020; X. Liu et al., 2021; Shen, Tan, He, Qin, & Liu, 2018).

In this work, we propose a method to combine the hidden representations of encoder layers to more easily utilize the full model. Moreover, we further investigate a simpler alternative as another compelling baseline. Some of the inspiration for this work is based on *Vertical Attention* introduced in AlKhamissi, Gabr, ElNokrashy, and Essam (2021).

First, we **motivate** the research in §2. We outline the utilized **tasks and datasets** in §3 (some details in §D). We describe the **method** in §4 (some details in §E). **Results and analysis** are in §5 (further analysis in §A). See §6 for the **experimentation setup**. We conclude in §7. We **discuss the proposed method** and possible hypotheses in §C, and some **related work** in §B.

2 Motivation & Hypothesis

A common goal when designing deeper networks is increasing their ability to represent longer chains of abstraction in different data domains and training objectives. As the model optimizes for a specific task, "unneeded" information is ignored. More general patterns that could benefit different downstream tasks are less likely to pass through to later layers and to the model's final representation.

Some properties of such base models (like large language models) may implicitly mitigate this effect—by leveraging the larger parameter capacity and increased width. The additional parameters enable memorizing more subtleties of the training data, while the increased model width allows for a less compact final representation, which may let through patterns less often used by the unsupervised language modeling objective of choice. A similar problem is the strength and clarity of gradients into earlier intermediate layers. Methods to alleviate this include skip connections (He, Zhang, Ren, & Sun, 2016), and alternatives to back-propagation (Nøkland, 2016). Some tasks may need such low-level knowledge across a large example space (X. Liu et al., 2021).

In this spirit, we propose an add-on module for pretrained deep sequence models to combine the representations of intermediate layers to adapt better to novel tasks. To benchmark our proposal by experimenting on the following task: Named Entity Recognition (**NER**) in the few-shot setting on the CoNLL-03 and WikiAnn datasets in two settings: Finetuning (**FT**) and Feature Extraction (**FE**).

3 Tasks, Datasets, and Raised Questions

On many common benchmarks, state-of-the-art performance is often near saturation. We consider some useful synthesizable variants of the benchmarks that test specific training settings or aspects of performance. The following experiments aim to analyze different aspects using each dataset: *adaptability* (finetuning versus feature extraction), *sample efficiency* (few-shot training), *training step efficiency* (time to convergence), and *effect of model depth*. See Table 2 for more details.

CoNLL-2003 The CoNLL-03 dataset (Tjong Kim Sang & De Meulder, 2003) provides a now mainstream NER benchmark in the English language.

WikiAnn The WikiAnn dataset (Rahimi, Li, & Cohn, 2019) is a multilingual NER benchmark which we use to test the effect of resource availability in the pretraining phase. For few-shot experiments, we sample a fixed training set of size $N = 100$ and train for each language separately. We also compare performance on the English subset between RoBERTa and the multilingual XLM-R.

Table 1: RoBERTa$_{\text{LARGE}}$-based configurations, the core new layer added in each, and their *extra* parameter count. The enhanced baseline with additional layers is chosen to be close in size to DWATT and CONCAT. For example, for BASE and LARGE-sized models, only n=1 and n=2 layers are added.

Name	New Layer	+ $O(\cdot)$ Params	+ # Params
Base	-	-	-
Base$_{+n}$	$n\times$ Base	$7n\ (d + d^2)$	25.19M
Average	MeanPool	-	-
Concat	Affine	Ld^2	25.18M
DWAtt	DWAtt	$Ld^2 + d^2 + Ld$	26.38M

4 Models

Let L denote a deep network's layer stack. Then H^4 is a learned function that mixes the layers' intermediate per-token representation vectors $\{\mathbf{z}_n \mid n \in |L|\}$ into $\hat{\mathbf{h}}$; the final representation.[5]

$$\hat{\mathbf{h}} = H\left(\cdots, \left\{(n, \mathbf{z}_n) \mid n \in |L|\right\}\right) \tag{1}$$

Then H is a function of the layer indices $\{n\}$ and the representation vectors $\{\mathbf{z}_n\}$. It can take other signals, like $\mathbf{z}_L \mapsto \mathbf{q}$ for DWAtt's query. We consider the following layer fusion models.

Layer Concatenation. A sum of linear transforms. Note that this is equivalent to concatenating all $\{\mathbf{z}_n\}$ then transforming the concatenation into the model width $d_{\mathbf{Z}}$.

$$\hat{\mathbf{h}} = \sum_n \mathbf{W}_n(\mathbf{z}_n) \tag{2}$$

Depth-Wise Attention. DWATT uses dot-product attention with keys $\mathbf{k}_n$, values $\mathbf{v}_n$, and query $\mathbf{q}$.

$$\mathbf{k}_n = \mathsf{PE}(n), \qquad \mathbf{v}_n = \mathsf{LN}_n\left(f_n^V(\mathbf{z}_n)\right), \qquad \mathbf{q} = 1 + \mathsf{elu}\left(\mathbf{z}_L + f^Q(\mathbf{z}_L)\right). \tag{3}$$

Where $\mathsf{PE}(n)$ is a learned positional embedding vector for layers n. LN is LayerNorm (Ba, Kiros, & Hinton, 2016).

$$f(\mathbf{z}) = \mathbf{W} \cdot \mathsf{LN}(\mathsf{gelu}(\mathbf{U}\mathbf{z})) \tag{4}$$

f^Q, f_n^V are MLPs with a bottleneck $1/2$ the model width $d_{\mathbf{Z}}$. elu from Clevert, Unterthiner, and Hochreiter (2015). Each MLP is comparable to a single $d_{\mathbf{Z}} \times d_{\mathbf{Z}}$ linear layer. Details in Appendix E.

Note: the scoring step in Attend reduces to a single, vector-by-static matrix multiplication as the keys $\{\mathbf{k}_n\}$ need no input. Assume a time step t, then let $\mathbf{K} = \{\mathbf{k}_n\}$ and $\mathbf{V} = \{\mathbf{v}_n\}$ be the matrix forms of the keys and values for $n \in |L|$. Then:

$$\mathsf{Attend}(\mathbf{q}, \{\mathbf{k}_n\}, \{\mathbf{v}_n\}) = \mathsf{softmax}_n\left(\mathbf{q} \cdot \mathbf{K}^\top\right) \cdot \mathbf{V}, \tag{5}$$

$$\hat{\mathbf{h}} = \mathbf{z}_L + \mathsf{Attend}\left(\mathbf{q}, \{\mathbf{k}_n\}, \{\mathbf{v}_n\}\right). \tag{6}$$

Base Model. The base model (RoBERTa or XLM-RoBERTa in BASE or LARGE sizes) is used as-is. The task module is changed where needed. In **FE** only the task module is trained.

Extra Transformer Layers. On top of the base model, we add 2 more Transformer layers before the classification head. In **FE** mode, we train only the added layers. We refer to this model as **R$_{26}$** (RoBERTa) or **XLM-R$_{26}$** (XLM-RoBERTa).

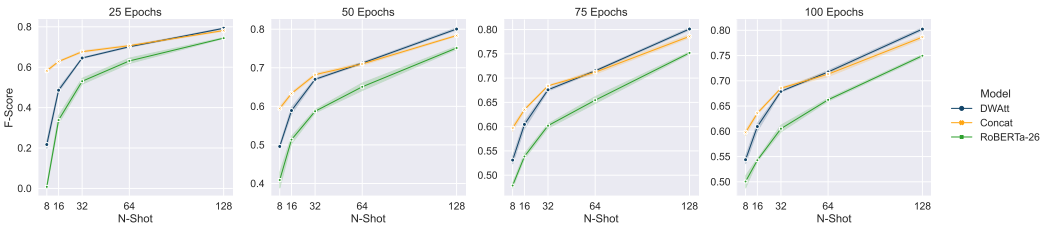


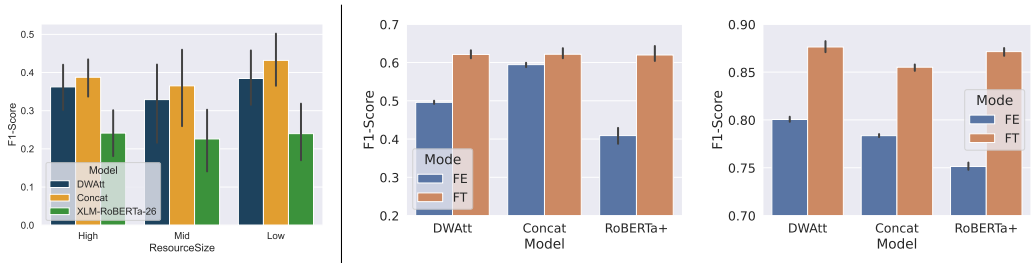
Figure 2: **F1-Score on the CoNLL-03 devset.** All pretrained weights are frozen (**FE**). In each chart from left to right, training is constrained to 25, 50, 75, and 100 max epochs. For each *N-Shot* experiment, NC samples ($C = 4$ classes) are randomly selected and trained on for the entire experiment. The scores are averaged across 5 trials with random initialization of weights and data sampling. We report the best observed dev score from the full training of each experiment and trial.

5 Results and Analysis

5.1 CoNLL: Few-Shot Adaptation

Micro-F1 is reported at each *N-Shot* in $\{8, 16, 32, 64, 128\}$. Each model in Figure 2 adds the specified module on top of a pretrained $\text{RoBERTa}_{\text{LARGE}}$. CONCAT (orange) noticeably outperforms R_{26} at all few-shot settings, while DWATT (blue) outperforms CONCAT at the higher data sizes.

5.2 Step and Sample Efficiency



(a) F1-Score on the WIKIANN. (b) $N = 8$ (c) $N = 128$

Figure 3: *(a)* Languages are grouped into three tiers of resource size. See Figure 7 and Table 2 for details. *(b & c)* **Feature Extraction (*FE*) versus Finetuning (*FT*).** Best validation F1-Score on CoNLL-03 within $N_{\text{epochs}} = 50$ training epochs. Finetuning consistently outperforms alternatives, but CONCAT and DWATT approach its performance even in **FE** training at lower data sizes.

In Figure 2, CONCAT outperforms noticeably at the lowest end (N=8 at N_{epochs}=25), improving on R_{26} by $\mathbf{58}\%$—possibly owing to its simpler connectivity and gradient path. While DWATT's improvement given increased data and training time may be explained by its being more selective through the attention module. In the largest setup (N=128, N_{epochs}=100; Figure 2), CONCAT improves on R_{26} by 3.68%, while DWATT improves on it by $\mathbf{5.28}\%$. R_{26} demonstrates the difficulties of extracting the patterns needed by a downstream task from a feature extractor that has already fit its last layer representation (L=24 for $\text{RoBERTa}_{\text{LARGE}}$) for its pretraining task.

Feature Extractor Adaptability Figure 3b & 3c show the effect of finetuning (**FT**) all model parameters for each of the same configurations used in other CoNLL-03 experiments. At $N = 8$, CONCAT in feature extraction (**FE**) training already gets most of the improvement observable from full **FT** across the board. At $N = 128$ the effect is less pronounced. While layer fusion methods help close the gap—with DWATT giving nearly half the improvement of FT using only FE, at $+5.28\%$—FT still manages to overshadow all FE configurations even with just R_{26}.

[4] Symbols in red have learned parameters.
[5] See diagram in Figure 1.

5.3 WIKIANN: Adapting to High, Mid, and Low Resource Languages

Figure 3a shows average performance grouped by the token count of pretraining data according to Conneau et al. (2019). Low-resource languages had 10–100M tokens, Mid-resource languages had 200–300M tokens , while High-resource languages had 2–20G tokens. While there is a slight lead for layer fusion methods in the Low-resource column, the difference does not translate to a similar lead in Mid-resource scores over High-resource. Training for Figure 3a was done similarly to Figure 2 in **FE** mode.

6 Experimentation Setup

6.1 Base Models

Monolingual experiments using CONLL-03, the English subset of WIKIANN all build on RoBERTa (**R**) (Y. Liu et al., 2019) via the pretrained `roberta-large` model. Multilingual experiments using WIKIANN build on XLM-RoBERTa (**XLM-R**) via the pretrained `xlm-roberta-large` model. Experiments using other variants are explicitly specified. Pretrained models are accessed via the HuggingFace package (Wolf et al., 2019).

6.2 Training

In figures where the epoch count is reported: Each chart refers to an experiment with the indicated N_{epochs} max epochs, which starts LR at max then decays it linearly to zero.

Few-shot experiments sample NC points uniformly at random from the full training set. Sampling is not stratified, so $N = 8$ Shot for $C = 4$ classes means 32 points in total sampled without replacement.

6.3 Evaluation

NER experiments report Micro-Average F1 using seqeval (Nakayama, 2018). Where applicable, experiments are given 5 trials whose average and confidence interval are reported by `seaborn` (Waskom, 2021). For easier reading, we report scores that lie in $[0, 1]$ as percentages $[0, 100]\%$.

7 Conclusion & Future Work

We present DWATT—a new method of reusing the latent representations of a deep neural network. We analyze DWATT and a similar, simpler method—CONCAT—from multiple aspects of performance and scaling on NER tasks. Results suggest similar layer fusion methods can be a robust tool for downstream adaptation. Performance gains from 1% to 6% and as high as 30% can be seen in various experiments for different few-shot sizes and training times, under Finetuning or Feature Extraction, and on base models of different depths. DWATT and CONCAT have shown improved performance even in Feature Extraction training against full Finetuning. We believe this effect may extend to other tasks besides sequence labeling and to other sequence modeling architectures besides the Transformer—and propose such analysis for future work. We believe *addon*-style additions to pretrained models, such as adapters and depth-wise mixing, to be a fertile ground for research into low-cost adaptation of large models that has not been saturated yet.

References

AlKhamissi, B., Gabr, M., ElNokrashy, M., & Essam, K. (2021, Apr). Adapting marbert for improved arabic dialect identification: Submission to the nadi 2021 shared task. In *Proceedings of the sixth arabic natural language processing workshop* (p. 260–264). Association for Computational Linguistics. Retrieved from https://aclanthology.org/2021.wanlp-1.29

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). *Layer normalization.* arXiv. Retrieved from https://arxiv.org/abs/1607.06450 doi: 10.48550/ARXIV.1607.06450

Bapna, A., Chen, M. X., Firat, O., Cao, Y., & Wu, Y. (2018). Training deeper neural machine translation models with transparent attention. *CoRR, abs/1808.07561*. Retrieved from http://arxiv.org/abs/1808.07561

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 1877–1901). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). *Fast and accurate deep network learning by exponential linear units (elus).* arXiv. Retrieved from https://arxiv.org/abs/1511.07289 doi: 10.48550/ARXIV.1511.07289

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *CoRR*, *abs/1911.02116*. Retrieved from http://arxiv.org/abs/1911.02116

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Kaiser, L. (2019). Universal transformers. *ArXiv*, *abs/1807.03819*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. Retrieved from https://aclanthology.org/N19-1423 doi: 10.18653/v1/N19-1423

Dou, Z., Tu, Z., Wang, X., Wang, L., Shi, S., & Zhang, T. (2019). Dynamic layer aggregation for neural machine translation with routing-by-agreement. *CoRR*, *abs/1902.05770*. Retrieved from http://arxiv.org/abs/1902.05770

Fan, A., Lavril, T., Grave, E., Joulin, A., & Sukhbaatar, S. (2020). Accessing higher-level representations in sequential transformers with feedback memory. *ArXiv*, *abs/2002.09402*.

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. *ArXiv*, *abs/1410.5401*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.

Hendrycks, D., & Gimpel, K. (2016). Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv*, *abs/1606.08415*.

Huang, G., Liu, Z., & Weinberger, K. Q. (2016). Densely connected convolutional networks. *CoRR*, *abs/1608.06993*. Retrieved from http://arxiv.org/abs/1608.06993

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... Zettlemoyer, L. (2020, July). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 7871–7880). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2020.acl-main.703 doi: 10.18653/v1/2020.acl-main.703

Li, J., Liu, C., & Gong, Y. (2018, Aug). Layer trajectory lstm. *arXiv*, *abs/1808.09522*. Retrieved from http://arxiv.org/abs/1808.09522

Liu, F., Ren, X., Zhao, G., & Sun, X. (2020). Layer-wise cross-view decoding for sequence-to-sequence learning. *CoRR*, *abs/2005.08081*. Retrieved from https://arxiv.org/abs/2005.08081

Liu, X., Wang, L., Wong, D. F., Ding, L., Chao, L. S., & Tu, Z. (2021). Understanding and improving encoder layer fusion in sequence-to-sequence learning. In *International conference on learning representations.* Retrieved from https://openreview.net/forum?id=n1HD8M6WGn

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv*, *abs/1907.11692*.

Loshchilov, I., & Hutter, F. (2017). *Decoupled weight decay regularization.* arXiv. Retrieved from https://arxiv.org/abs/1711.05101 doi: 10.48550/ARXIV.1711.05101

Nakayama, H. (2018). *seqeval: A python framework for sequence labeling evaluation.* Retrieved from https://github.com/chakki-works/seqeval (Software available from https://github.com/chakki-works/seqeval)

Nøkland, A. (2016). *Direct feedback alignment provides learning in deep neural networks.* arXiv. Retrieved from https://arxiv.org/abs/1609.01596 doi: 10.48550/ARXIV.1609.01596

Rahimi, A., Li, Y., & Cohn, T. (2019, July). Massively multilingual transfer for NER. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 151–164). Florence, Italy: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/P19-1015

Shen, Y., Tan, X., He, D., Qin, T., & Liu, T. (2018). Dense information flow for neural machine translation. *CoRR*, *abs/1806.00722*. Retrieved from http://arxiv.org/abs/1806.00722

Tjong Kim Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003* (pp. 142–147). Retrieved from https://www.aclweb.org/anthology/W03-0419

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st international conference on neural information processing systems* (p. 6000–6010). Red Hook, NY, USA: Curran Associates Inc.

Wallat, J., Singh, J., & Anand, A. (2021, September). BERTnesia: Investigating the capture and forgetting of knowledge in BERT. *arXiv:2106.02902 [cs]*. Retrieved 2021-12-03, from http://arxiv.org/abs/2106.02902 (arXiv: 2106.02902)

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., ... Bowman, S. R. (2019). Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, *abs/1905.00537*. Retrieved from http://arxiv.org/abs/1905.00537

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, *abs/1804.07461*. Retrieved from http://arxiv.org/abs/1804.07461

Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D. F., & Chao, L. S. (2019). Learning deep transformer models for machine translation. *CoRR*, *abs/1906.01787*. Retrieved from http://arxiv.org/abs/1906.01787

Wang, Q., Li, F., Xiao, T., Li, Y., Li, Y., & Zhu, J. (2018). Multi-layer representation fusion for neural machine translation. In *Proceedings of the 27th international conference on computational linguistics* (pp. 3015–3026). Retrieved 2021-06-25, from https://www.aclweb.org/anthology/C18-1255.pdf

Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. Retrieved from https://doi.org/10.21105/joss.03021 doi: 10.21105/joss.03021

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, *abs/1910.03771*. Retrieved from http://arxiv.org/abs/1910.03771

A More Analysis

A.1 Scaling by Depth

In Figure 4, we compare the same add-on configurations on the BASE and LARGE variants of RoBERTa. At $N = 8$, CONCAT is an outlier in its performance gain over the alternatives. In all $N \in \{8, 128\}$, both CONCAT and DWATT match or beat the performance gain observed from RoBERTa+[6] when changing from BASE to LARGE.

A.2 Training Behavior

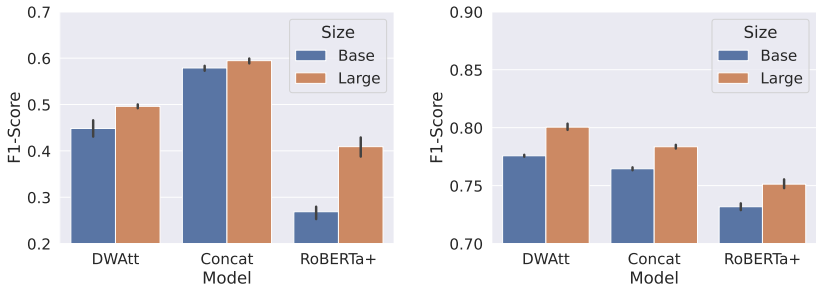
In Figure 5, we focus on model validation behavior during training with N_{epochs}=50. At N=8, only CONCAT manages to adapt well, and rapidly, converging in 30% of training time. At N=128, DWATT has enough data to reach similar performance at a similar pace, and quickly pulls ahead.

We repeat in Figure 6 the experiments from Figure 2 but on English only from WIKIANN, on R_{26}. Yet again CONCAT performs better with less data, while DWATT is better at the higher end (N=128, N_{epochs}=100) by +2.08% and +0.92% over CONCAT and R_{26}, respectively.

A.3 Problems from Multilinguality

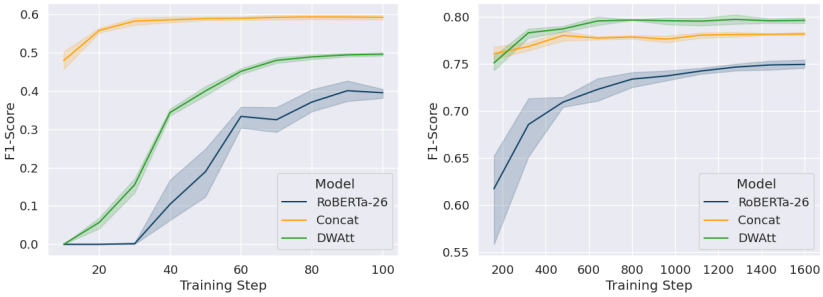
We hypothesize that the consistent gain of CONCAT over DWATT is due to the multilingual nature of the pretrained XLM-R model. By design, the input space of the model is shared by all languages—any needed language differentiation may take place only deep into the encoder stack. A layer fusion

[6] RoBERTa+ BASE is L=12 + 1 layers, while LARGE is L=24+2.



(a) $N = 8$ (b) $N = 128$

Figure 4: **BASE versus LARGE Pretrained Models.** Validation F1-Score on CoNLL-03 on the DWATT, CONCAT, and enhanced (+layers) configurations of RoBERTa BASE and LARGE. *(a)* At $N=8$, $\text{CONCAT}_{\text{BASE}}$ shows a clear lead even on $\text{RoBERTA+}_{\text{BASE}}$. *(b)* At $N=128$, $\text{CONCAT}_{\text{BASE}}$ outperforms $\text{RoBERTA+}_{\text{LARGE}}$, while $\text{DWATT}_{\text{LARGE}}$ outperforms $\text{CONCAT}_{\text{LARGE}}$. *(c)* This also supports the claim for higher data and training efficiency from CONCAT and DWATT in **FE** training compared to traditional last-layer fitting.



(a) Few-shot at $N = 8$. (b) Few-shot at $N = 128$.

Figure 5: **Training Behavior.** Validation F1-Score across steps in **FE** training on CoNLL-03. CONCAT generalizes more readily at smaller N compared to DWATT. At both sizes, layer fusion methods are able to extract more from pretrained models than traditional last-layer fitting.

method like DWATT, which is incentivized to choose only one or a few layers per token, may have a poorer fit compared to a less constrained method like CONCAT.

In Figure 6 we repeat the experiments from Figure 2 on the WIKIANN English subset. Compare the scores in both Figures 6 and 7. Starting from N_{epochs}=25 and only N=32 samples, performance using $\text{RoBERTa}_{\text{LARGE}}$ as the base model beats the corresponding run on $\text{XLM-R}_{\text{LARGE}}$ at N_{epochs}=25 with 100 samples.

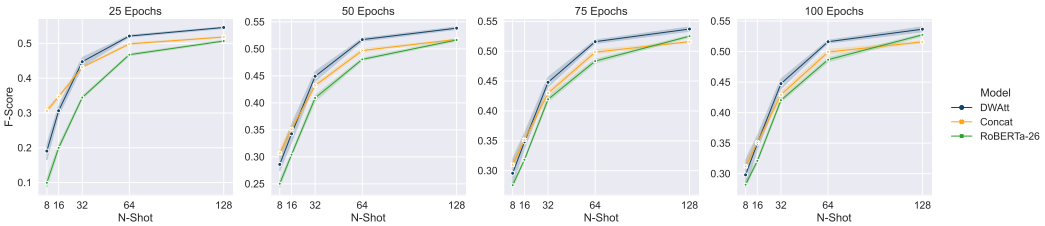


Figure 6: F1-Score on the WIKIANN *English* devset on the *RoBERTa$_{LARGE}$* base model.

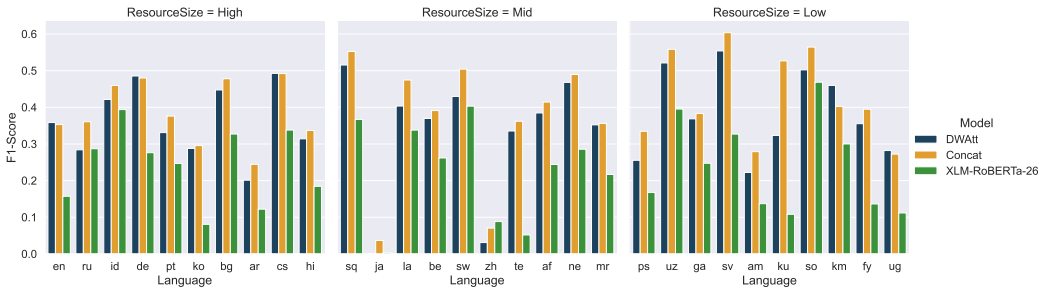


Figure 7: F1-Score on the WIKIANN devset. Each language was trained for $N_{\text{epochs}} = 25$ and tested separately. Each model was initialized from the pretrained XLM-R_{LARGE} weights, augmented with the specified module (DWAtt, Concat, or additional layers), then trained on exactly 100 uniformly-sampled training points for each language. The languages are sorted by token count of pretraining data according to Conneau et al. (2019).

B Related Work

Layer Aggregation (static weights). Bapna, Chen, Firat, Cao, and Wu (2018) defines, for each decoder layer, a trainable softmax-normalized vector of weights to get the weighted sum of the encoder intermediates. X. Liu et al. (2021) provides one static, learned vector for each intermediate layer's representation which acts as element-wise scaling. Both works are similar to CONCAT in applying a static, learned mixing transform to all layers. For each layer: The first provides a scalar weight, the second provides a vector for element-wise multiplication, while CONCAT applies a linear transform.

In Shen et al. (2018), the **DenseNMT** is an encoder-decoder NMT architecture densely connected in the style of **DenseNet** (Huang, Liu, & Weinberger, 2016). Each encoder layer takes a concatenation of all previous layer representations. Similarly for decoder layers. See also Q. Wang et al. (2019). These methods compare structurally to CONCAT, which is applied only once on the full layer stack.

Q. Wang et al. (2018) generates the weight for an encoder layer via an MLP on the layer's representation, irrespective of the rest of the model: $w_i = f^a(\mathbf{z}_i)$ then $\hat{\mathbf{h}} = \sum_i (w_i \mathbf{z}_i)$.

Dynamic Layer Mixing. The following methods use signals from the input to change the transformation itself dynamically. See also DWATT which uses a Dot-Product Attention module for comparison. Li, Liu, and Gong (2018) applies an LSTM depth-wise on the intermediate vectors of the stack of LSTM cells applied on input sequences, as expected. We attempted adding an LSTM cell applied depth-wise to the Transformer encoder stack but observed lacking performance. Note that the referenced work utilizes a non-basic cell with peep-hole expressions, and some architecture connectivity that complicates experiments in the Feature Extraction context. Dou et al. (2019) utilizes a dynamically-weighted routing mechanism to mix transformations of each intermediate representation, then concatenate all such.

C Discussion

C.1 Levels of Abstraction

For non-recurrent deep neural networks, there exists a functional limit to the depth of abstraction obtainable, which is proportional to the depth of the network. Abstraction here refers to the depth of composed rules that a model may learn to apply on low-level stimulus, such as pixels or tokens. For feed-forward models like an MLP or transformers, this corresponds to the depth in terms of stacked nonlinear layers. As an example, to handle program-like systematic inputs, depth-recurrent and memory-augmented architecture were utilized in works such as Dehghani, Gouws, Vinyals, Uszkoreit, and Kaiser (2019); Graves, Wayne, and Danihelka (2014). For traditional Transformer models, like $\text{RoBERTa}_{\text{LARGE}}$, we can say that the limit is proportional to the number of layers in the network (e.g. $O(|L|)$; $|L| = 24$).

Hiding Within Scale. In practice, this is seldom an obvious problem because large networks would have enough width-wise parameter capacity to directly encode "intuition", i.e. shortcuts to knowledge and abstraction. They may do so by tying the low-level representation of some inputs to intermediate signals for the high-level concepts they tend to manifest. As an artificial example: A small, shallow model for classifying sentiment may tie a token such as `scary` to become a strong signal for negative sentiment, say in a movie review setting. Sensible in the domain of kids movies; but may in fact signify a positive sentiment instead when observed in reviews for horror movies. The key point is that it is an early shortcut, not whether it is accurate.

Intuition as Shortcuts From Raw Input. Thus, earlier layers can encode information at a higher level than $|L|$-steps of abstraction would suggest. This would be needed, and expected, for models of a reasonable depth and width to be able to satisfy the feature extraction needs of the *pretraining* task at the last layer. Should a *downstream* task require a different signal from what the *pretraining* task exposed, it may have to *shift* a large subset of the weights to relearn or resurface that information from the shallower levels.

Proposition. The methods we've discussed may enable downstream tasks to query the model for hidden but useful information. For the downstream task to make use of such features, it would likely need to transform them further. By applying the 2-layer MLPs f_n^V on these intermediate features (in DWATT), and a linear transform in CONCAT, the task can extract a more useful representation from each level/layer.

C.2 Modeling Capacity

The two models presented and highlighted—DWATT and CONCAT—are aggregate views of the features of all intermediate Transformer layers $\{\mathbf{z}_n \mid n \in |L|\}$ (see Section 4). DWATT's added module requires access to the last layer's $\mathbf{z}_L$ to form the query, while CONCAT does not. Neither method makes any use of external signals such as, for example, a task embedding vector.

These three points together present an underlying property of the modeling capacity of DWATT and CONCAT: *The depth-wise layer mixing arrives at a model that is, at most, as expressive as the underlying sequence-modeler*. See Figure 3b & 3c where layer fusion under FE approaches but does not exceed FT, while all methods are similar under FT.

Modeling Dimensions. DWATT and CONCAT operate *depth-wise* over a sequence-modeling model. By that very nature, it may not be the best option when the task at hand requires increased or improved *spatial* abstraction—the ability to learn connections on a spatial axis (e.g. between tokens in sequences in a text Transformer). Then, adding extra Transformer layers or full finetuning may be better options.

D Tasks

Table 2: Train and Dev subset sizes in sentence count. For WIKIANN, we list the average size of a language in the corresponding resource tier, determined by train size.

Dataset	Classes	Resources	Train	Dev
CONLL-03	4	High	14k	3250
WIKIANN	3	High	20k	10k
		Mid	1k–5k	1k
		Low	100	100

Table 3: Architecture Parameters

Model	Param	Value
All$_{\text{Large}}$	Transformer Layers	24
All	Learning Rate	1e-5
DWAtt	γ^Q, query bottleneck	0.5
DWAtt	γ^V, values bottleneck	0.5
DWAtt	d_{pos}, keys latent	24

E Design Details

E.1 Training

All trainings use the AdamW (Loshchilov & Hutter, 2017) optimizer with a linear decay learning rate (LR) schedule. Training on WIKITEXT-2 uses a batch size of 8 samples, and a max LR of 5×10^{-5}. Training on CONLL-03 and WIKIANN uses a batch size of 16 and a max LR of 5×10^{-5}.

E.2 Layer Index Embedding

$\mathbf{k}_n^{\text{pos}} \in \mathbb{R}^{d_{\text{pos}}} \sim \mathcal{U}(0,1)$ are static positional embedding vectors of the layer index n. $\mathbf{W}^K$ is a learned affine transform.

$$\mathbf{k}_n = \mathsf{PE}(n) = \mathbf{W}^K \mathbf{k}_n^{\text{pos}} \tag{7}$$

E.3 MLP Modules

For each role $\in \{Q, V\}$ that uses an MLP, one is defined for a layer n as:

$$f_n(\mathbf{x}) = \mathbf{W}_n \cdot \mathsf{LN}\left(\mathsf{gelu}\left(\mathbf{U}_n\ \mathbf{x}\right)\right) \tag{8}$$

Where $\mathbf{U}_n, \mathbf{W}_n$ are the down and up projections of the bottleneck ($d_{\mathsf{z}} \mapsto \gamma d_{\mathsf{z}} \mapsto d_{\mathsf{z}}$), respectively. This means, for example, that for value transforms one set of weights is assigned to each layer n with nothing shared. gelu is the activation function from Hendrycks and Gimpel (2016).