Why Skip If You Can Combine: A Simple Knowledge Distillation Technique for Intermediate Layers

Yimeng Wu[*] **Peyman Passban**[*] **Mehdi Rezagholizadeh** **Qun Liu**
Huawei Noah's Ark Lab
firstname.lastname@huawei.com

Abstract

With the growth of computing power neural machine translation (NMT) models also grow accordingly and become better. However, they also become harder to deploy on edge devices due to memory constraints. To cope with this problem, a common practice is to distill knowledge from a large and accurately-trained teacher network ($\mathcal{T}$) into a compact student network ($\mathcal{S}$). Although knowledge distillation (KD) is useful in most cases, our study shows that existing KD techniques might not be suitable enough for deep NMT engines, so we propose a novel alternative. In our model, besides matching $\mathcal{T}$ and $\mathcal{S}$ predictions we have a combinatorial mechanism to inject layer-level supervision from $\mathcal{T}$ to $\mathcal{S}$. In this paper, we target low-resource settings and evaluate our translation engines for Portuguese→English, Turkish→English, and English→German directions. Students trained using our technique have 50% fewer parameters and can still deliver comparable results to those of 12-layer teachers.

1 Introduction

Almost in all deep learning tasks, including neural machine translation (NMT), an ensemble of models outperforms a single model. In fact, ensemble modelling (training multiple models and ensemble decoding) is supported by most publicly available NMT frameworks (Klein et al., 2017; Junczys-Dowmunt et al., 2018; Vaswani et al., 2018; Ott et al., 2019). However, we know that dealing with multiple models could be challenging, especially in deep learning scenarios. To tackle the issue, one effective solution is to compress the knowledge in an ensemble into a single model through distillation (Buciluǎ et al., 2006; Hinton et al., 2015).

The core part of any knowledge distillation (KD) pipeline is a component that matches different models' predictions, which is usually implemented via multiple cost functions (see Section 2). Furthermore, we also need to take care of the architecture mismatch that may exist between student ($\mathcal{S}$) and teacher ($\mathcal{T}$) models. In KD, these two models can have different architectures (Jiao et al., 2019; Sun et al., 2019) and the motivation is to be able to compress a large teacher into a smaller student.

This research focuses on the aforementioned issue. If we distill from intermediate layers of a teacher that has more layers than its student, we have to select a subset of $\mathcal{T}$ layers and skip others as there are no peers for all of them on the $\mathcal{S}$ side. Clearly, we do not benefit from the skipped layers in this scenario. This type of KD introduces a problem of finding an optimal subset of $\mathcal{T}$ layers (to distill from). Although this might, to some extent, be mitigated via a search mechanism, our experimental results show that the problem is severe in NMT and each layer plays a unique role. Therefore, we prefer to keep all layers rather than skip them.

KD has recently become popular in NMT but, to the best of our knowledge, all NMT models (Kim and Rush, 2016; Tan et al., 2019) are still trained using the original idea of KD (Hinton et al., 2015), which is referred to as Regular KD (*RKD*) throughout this paper. *RKD* only matches $\mathcal{S}$ and $\mathcal{T}$ outputs, regardless of their internal architecture. However, there exist techniques such as Patient KD (*PKD*) (Sun et al., 2019) proposed for other tasks that not only match final predictions but also focus on internal components and distill their information too (Sun et al., 2020). In this research, we borrowed those ideas and adapted them to NMT. This is the first contribution of the paper.

PKD and other similar models suffer from the *skip* problem, which happens when $\mathcal{T}$ has more layers than $\mathcal{S}$ and some $\mathcal{T}$ layers have to be skipped in order to carry out layer-to-layer distillation. In this paper, we propose a model to distill from *all* teacher

[*]These authors contributed equally.

layers so we do not have to skip any of them. This is our second contribution by which we are able to outperform *PKD*. Moreover, for the first time we report experimental results for Transformer-based (Vaswani et al., 2017) models trained with a layer-level KD technique in the context of NMT. This set of results is our third and last contribution in this paper.

The remainder of the paper is organized as follows: In Section 2 we explain the fundamentals of KD. Section 3 discusses the methodology. We describe the advantages of our model and accompany our claims with experimental results in Section 4. Finally, in Section 5, we conclude the paper with our future plan.

2 Background

Usually, in multi-class classification scenarios the training criterion is to minimize the negative log-likelihood of samples, as shown in Equation 1:

$$\mathcal{L}(\theta) = -\sum_{v=1}^{|V|} \mathbb{1}(y=v) \times \log p(y=v|x;\theta) \quad (1)$$

where $\mathbb{1}(.)$ is an indicator function, (x, y) is an input-output training tuple, and θ and $|V|$ are the parameter set of the model and the number of classes, respectively. There is no feedback returned from the network for misclassified examples as $\mathbb{1}(y \neq v) = 0$. This issue is resolved in KD with extending $\mathcal{L}$ with an additive term (Kim and Rush, 2016; Tan et al., 2019), as shown in Equation 2:

$$\mathcal{L}_{KD}(\theta_{\mathcal{T}}, \theta_{\mathcal{S}}) = \\ -\sum_{v=1}^{|V|} q(y=v|x;\theta_{\mathcal{T}}) \times \log p(y=v|x;\theta_{\mathcal{S}}) \quad (2)$$

where there is a student model with the parameter set θ_S whose predictions are penalized with its own loss as well as $\mathcal{T}$ predictions given by $q(y = v|x;\theta_T)$. In KD, the first component of the loss (q) is usually referred to as the *soft* loss and the $\mathcal{S}$ model's loss is known as the *hard* loss. This form of training provides richer feedback compared to the previous one and leads to high(er)-quality results. KD for NMT also follows the same principle where V is a target-language vocabulary set and $\mathcal{L}_{KD}$ is computed for *each* word during decoding.

With the matching strategy proposed in KD, $\mathcal{S}$ learns to mimic its $\mathcal{T}$. A teacher could be a deep model trained on a large dataset but we do not necessarily need to have the same complex architecture for $\mathcal{S}$. We can distill teacher's knowledge into a smaller model and replicate its results with fewer resources.

Kim and Rush (2016) studied this problem and proposed a sequence-level extension of Equation 2 for NMT models. They evaluated their idea on recurrent, LSTM-based models (Hochreiter and Schmidhuber, 1997) and could run the final model on a cellphone. Freitag et al. (2017) extended the original two-class idea (one $\mathcal{S}$ with one $\mathcal{T}$) to distill from multiple teachers. They trained an attention-based recurrent model (Bahdanau et al., 2015) for their experiments.

Tan et al. (2019) proposed a setting to train a multilingual Transformer for different language directions. In order to have a high-quality multilingual model they distill knowledge from separately trained bilingual models. Their work is one of the few papers that reports KD results for NMT on Transformers. However, their results are not directly comparable to ours as they benefit from rich, multilingual corpora.

Wei et al. (2019) introduced a pipeline where a student model learns from different checkpoints. At each validation step, if the current checkpoint is a better model than the best existing checkpoint, $\mathcal{S}$ learns from it, otherwise the best stored checkpoint is considered as the teacher.

In all models discussed so far, *i*) $\mathcal{S}$ usually has the same architecture as its teacher(s) but we know that recent NMT models, particularly Transformers, are deep models which makes them challenging to run on edge devices. Moreover, *ii*) the training criterion in the aforementioned models is to combine final predictions. Transformers have new components (e.g. self-attention) and multiple (sub-)layers that consist of valuable information (Clark et al., 2019) and we need more than an output-level combination to efficiently distill for/from these models. Therefore, a new technique that is capable of addressing *i* and *ii* is required.

Authors of *PKD* spotted the problem and focused on internal layers (Sun et al., 2019). They studied the limitations of *RKD* for BERT (Devlin et al., 2019) models and introduced a layer-to-layer cost function. They select a subset of layers from $\mathcal{T}$ whose values are compared to $\mathcal{S}$ layers. They also showed that different internal components are important and play critical roles in KD.

The layer-level supervision idea was successful for monolingual models but so far, no one has tried it in the context of NMT. In this paper, we investigate if the same idea holds for bilingual models or if NMT requires a different type of KD. Moreover, we address the *skip* problem in *PKD* (shown in Figure 1). It seems in deep teacher models we do not need to skip layers and we can distill from *all layers*.

3 Methodology

In *RKD*, distillation only happens at the output level whereas *PKD* introduces layer-wise supervision. This idea is illustrated in Figure 1.

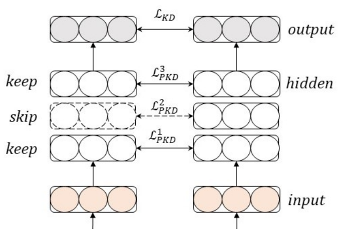

Figure 1: The network on the left-hand side is $\mathcal{S}$ and the other one is $\mathcal{T}$. In this example, $\mathcal{T}$ has 3 hidden layers and KD for intermediate layers can be applied using all layers or a subset of them, e.g. the second layer can be skipped.

In *PKD*, finding a skippable layer is the main challenge. Accordingly, we propose a combinatorial idea, *CKD*, by which we are able to fuse layers and benefit from *all* information stored in *all* layers. Our idea can be formulated as follows:

$$\mathcal{L}_{CKD}(L_s, L_t) = \sum_{l_s^i \in L_s} MSE(l_s^i, f_t^i) \qquad (3)$$
$$f_t^i = F(l_t^j); j \in M(i)$$

where L_s and L_t indicate the set of all hidden layers of $\mathcal{S}$ and $\mathcal{T}$, respectively. $MSE()$ is the mean-square error function and l_s^i is the i-th hidden layer of $\mathcal{S}$. In *PKD*, f_t^i is the teacher's i-th layer whereas in our case f_t^i is the result of a fusion applied through the function $F()$ to a particular subset of $\mathcal{T}$ layers. This subset is defined via a mapper function $M()$ which takes an index (pointing to a layer on the student side) and returns a set of indices from the teacher model. Based on these indices, teacher layers are combined and passed to the distillation process, e.g. if $M(2) = \{1, 3\}$ that means F is fed by the first (l_t^1) and third (l_t^3) layers of $\mathcal{T}$ and the distillation happens between l_s^2 and f_t^2 (result of fusion).

For $F()$, a simple concatenation followed by a linear projection provided the best results in our experiments, so in the previous example:

$$f_t^2 = F(l_t^1, l_t^3) = W[l_t^1 \bullet l_t^3]^T + b$$

where $\bullet$ indicates concatenation, and $W \in \mathbb{R}^{d \times 2d}$ and $b \in \mathbb{R}^d$ are learnable parameters of KD. All l_t^1, l_t^3, l_s^2, and f_t^2 are d-dimensional vectors.

The mapper function $M()$ defines our combination strategy for which we have 4 different variations of regular combination *(RC)*, overlap combination *(OC)*, skip combination *(SC)*, and cross combination *(CC)*. Figure 2 visualizes these variations. As the figure shows, *PKD* is a particular case of our model, but *CKD* gives us more flexibility in terms of distilling from different teacher configurations.

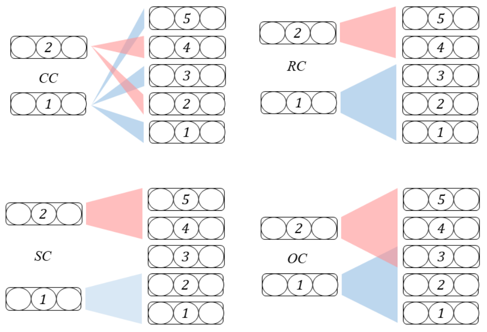

Figure 2: Different variations of *CKD*. $\mathcal{T}$ has 5 and $\mathcal{S}$ has 2 hidden layers. For the *CC* case $M(1) = \{1, 3, 5\}$.

4 Experimental Study

Although our proposed model is a general KD technique and can be applied in different settings, we narrow down the scope of this paper to low-resource, NMT settings. The incentive idea behind our project was to train NMT models for small datasets, so we report experimental results accordingly.

To evaluate *CKD*, we trained multiple models to translate from English (En) into German (De), and from Portuguese (Pt) and Turkish (Tr) into English (En). For the Pt|Tr→En directions we use the `IWSLT-2014` dataset, and the En→De experiment uses the `WMT-2014` dataset.

In Pt→En, we use the original split of datasets from IWSLT[1] with 167K, 7590, and 5388 sentences for training, development, and test sets, respectively. For Tr→En, the split is 142K, 1958,

[1] `https://wit3.fbk.eu/`

	Models	Pt→En	Tr→En	En→De$_1$	En→De$_2$
Baselines	*Teacher*	43.69	26.44	18.57	27.03
	No-KD	42.12	24.60	17.04	16.09
	Regular KD	42.26	25.16	17.66	16.99
	PKD	42.27	26.88	17.84	21.06
CKD (Ours)	*Regular Comb. (RC)*	43.43	26.75	18.29	21.15
	Overlap Comb. (OC)	**43.78**	26.52	18.44	21.26
	Skip Comb. (SC)	43.17	26.37	17.81	**21.47**
	Cross Comb. (CC)	42.57	**27.09**	**18.60**	21.13

Table 1: BLEU score comparisons of different KD models. *No-KD* is a model trained with no KD techniques using the the same architecture and dataset as students'.

and 1982 for training, development, and test sets. With this dataset selection our $\mathcal{T}$ models' results are comparable to publicly reported results.[2] On these datasets, our teachers outperform all other existing models so we can ensure that we distill from reliable sources.

For En→De, the dataset is the same as the original Transformer's (Vaswani et al., 2017), namely the training set includes 4.5M sentences, `newstest2013` is used as the validation set and `newstest2014` is our test set with 3000 and 3003 sentences, respectively. We selected this dataset to be comparable to a well-known baseline and make sure our training pipeline yields high-quality engines.

We preprocess datasets with Sentence-Piece (Kudo and Richardson, 2018). For Pt→En, we extracted a shared vocabulary set for both source and target sides with 32K subwords. Both $\mathcal{S}$ and $\mathcal{T}$ are trained using the same training set. Tr→En follows the same setting. For En→De, we conduct two experiments. Since our focus in this paper is to work with low-resource settings, in En→De$_1$, $\mathcal{S}$ and $\mathcal{T}$ are trained on a dataset of 200K sentences randomly sampled from the main dataset (4.5M).[3] For this experiment the vocabulary set size is 15K. In En→De$_2$, we slightly changed the setting where we use the entire set of 4.5M sentences to train $\mathcal{T}$ but $\mathcal{S}$ still uses the same 200K dataset. In this scenario, we assumed that there already exists a high-quality teacher trained on a large dataset but we only have a small in-house dataset to train the student. For this experiment the vocabulary size is 37K.

[2]http://cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/

[3]Our code and datasets: https://github.com/yimeng0701/CKD_pytorch

Table 1 summarizes our results for all experiments. Models are compared based on BLEU (Papineni et al., 2002) scores computed using sacreBLEU (Post, 2018). As the table shows, our students outperform all other students trained with different KD techniques. Moreover, students in Pt|Tr→En and En→De$_1$ settings are even comparable to accurately-trained, deep teachers. All teachers are 12-layer Transformers (6 for encoding and 6 for decoding), whereas students only have 4 layers (2 encoder layers and 2 decoder layers). All settings in our experiments are identical to those of Vaswani et al. (2017), which means hyper-parameters whose values are not clearly declared in this paper use the same values as the original Transformer model.

CKD makes it possible to reduce the number of parameters in our students by 50% and yet deliver the same high-quality translations. Accordingly, this enables us to run these translation engines on edge devices. Table 2 shows the exact number of parameters for each model.

	Pt→En	Tr→En	En→De$_1$	En→De$_2$
$\mathcal{T}$	61M	61M	52M	63M
$\mathcal{S}$	31M	31M	22M	34M

Table 2: The exact number of parameters for different models and settings.

For results reported in Table 1, cross-model layer mappings between teacher and student layers are as follow:

$$M_{SC}(1) = \{1, 2\} \qquad M_{SC}(2) = \{5, 6\}$$
$$M_{CC}(1) = \{1, 3\} \qquad M_{CC}(2) = \{4, 6\}$$
$$M_{RC}(1) = \{1, 2, 3\} \qquad M_{RC}(2) = \{4, 5, 6\}$$
$$M_{OC}(1) = \{1, 2, 3, 4\} \quad M_{OC}(2) = \{3, 4, 5, 6\}$$

We tried a simple (and somewhat arbitrary) configuration for layer connections and there is no systematic strategy behind it. However, better results can be achieved with better heuristics or through a search process. Moreover, as the mappings show there is no connection between student and teacher models' decoder layers. In our experiments, we noticed that *any* KD technique applied to the decoder *considerably* decreases performance, so we only use KD on the encoder side. More specifically, each student model has two decoder layers which only receive inputs from the same model's encoder layers and they are not connected to the teacher side.

To train students with different KD techniques we use different loss functions. In $\mathcal{T}$ and *No-KD* we only have a single loss function ($\mathcal{L}$) as described in the original Transformer model (Vaswani et al., 2017). For models trained with *RKD*, an additional loss is involved to match teacher and student predictions ($\mathcal{L}_{KD}$). The final loss in this case is an interpolation of the aforementioned losses: $\big((\beta \times \mathcal{L}) + (\eta \times \mathcal{L}_{KD})\big)$. In our experiments, $\beta = (1 - \eta)$ where $\eta = 0.1$ is obtained through a search process over the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

For students trained using *PKD* and *CKD*, a third loss is also used in addition to $\mathcal{L}$ and $\mathcal{L}_{KD}$. Similar to other losses, the third one is also multiplied by a weight value (λ) to incorporate its impact into the training process. In this new setting, $\beta = (1 - \eta - \lambda)$, $\eta = 0.1$, and $\lambda = 0.7$. The high value of λ compared to other weights shows the importance of intermediate KD for deep models. All these values are learned through an empirical study in order to minimize the final loss of translation engines.

4.1 How Powerful is *CKD*?

In order to study the behaviour of *CKD*, we designed multiple, small experiments in addition to those reported in Table 1. *PKD* proposes a solution to define a loss between internal components of teacher and student models. The original model implemented this idea for intermediate layers. In one of our experiments we extended *PKD* by adding an extra loss function for *self-attention* components. Therefore, this new extension compares final outputs of student and teacher models as well as their intermediate layers and self-attention parameters. In this experiment, BLEU for Pt→En increased from 42.27 to 43.28, but our model is still superior with the BLEU score **43.78**. For this setting, *CKD* outperforms even a very complicated variation of *PKD* that could be an indication of our model's capacity in training high-quality students. For Tr→En and En→De$_1$ we also observed slight improvements by matching teacher and student self-attention components but results were not statistically significant and *CKD* was still better.

We also studied how *CKD* behaves in large experimental settings, for which we used En→De and En→French (Fr) datastes with 4.5M and 36M training samples, respectively, and trained 12-layer teachers and 4-layer students. For this experiment, we used the same settings, and test and development sets suggested in Vaswani et al. (2017). Table 3 summarizes our results.[4]

	$\mathcal{T}$	*No-KD*	*PKD*	*RC*	*OC*
En→Fr	38.41	35.45	34.97	36.10	35.85
En→De	27.03	24.31	23.38	24.14	23.97

Table 3: BLEU scores of different KD models for large datasets.

As the table shows, *CKD* is better than *PKD* in large experimental settings too. However, in order to have a better understanding of the large-dataset scenario we need to explore more configurations. We emphasize that for this paper our focus was to work with small students and datasets.

5 Conclusion and Future Work

In this paper, we proposed a novel model to distill from intermediate layers as well as final predictions. Moreover, we addressed the *skip* problem of *PKD*. We applied our technique in NMT and showed its potential in training high-quality and compact models. In our future work, *i*) we are interested in distilling from deep NMT models into extremely small students with *CKD*, in the hope of achieving the same results of large models with much smaller counterparts. *ii*) We also try to improve the combination module and find a better alternative than concatenation. *iii*) Finally, we plan to evaluate *CKD* in other tasks such as language modeling.

Acknowledgement

We would like to thank our anonymous reviewers for their valuable feedback.

[4]Performing a comprehensive study for large datasets was not our priority but we recently started related investigations and will report more inclusive results soon.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In 3rd International Conference on Learning Representations, ICLR 2015.

Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 535–541.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. In Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 276–286.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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), pages 4171–4186.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. arXiv preprint arXiv:1702.01802.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. arXiv preprint arXiv:1909.10351.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In Proceedings of ACL 2018, System Demonstrations, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1317–1327.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In Proc. ACL.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 66–71.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In Proceedings of NAACL-HLT 2019: Demonstrations.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics, pages 311–318. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In Proceedings of the Third Conference on Machine Translation: Research Papers, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4314–4323.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. arXiv preprint arXiv:2004.02984.

Xu Tan, Yi Ren, Di He, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In International Conference on Learning Representations.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. CoRR, abs/1803.07416.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008.

Hao-Ran Wei, Shujian Huang, Ran Wang, Xinyu Dai, and Jiajun Chen. 2019. Online distilling from checkpoints for neural machine translation. 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), pages 1932–1941.