Ensemble Mechanisms in the Transformer

Johannes Eschbach-Dymanus
University of Heidelberg / Student Nr. 3363713
eschbach@cl.uni-heidelberg.de

1 Introduction

The introduction of the purely attention based transformer model by Vaswani et al. (2017a) proved to be a milestone in language modeling and replaced its recurrent or convolutional predecessors in nearly all fields of Natural Language Processing. While the self-attention mechanism surely is the core piece of the transformer, three implicit ensembling mechanisms contained in the architecture further boost the model's performance: Multi-head attention, residual connections and dropout. In this paper, I isolate the contributions of the three mechanisms to the model's performance and discuss noticeable interactions between them.

2 Transformer Ensemble Mechanisms

For any convex loss function, averaging the outputs of multiple models will always result in a lower loss than the average loss of the individual models (Rougier, 2016). However, training multiple models to average their predictions is prohibitively computationally expensive, especially when we consider the number of parameters required in a language model. Instead of actually averaging multiple models, the transformer architecture contains multiple mechanisms that approximate this effect with either no or very limited increased computational costs.

2.1 Multi-Head Attention

Self-attention is the core mechanism in the transformer architecture. Given some input sequence of n token-embeddings of size d_{model}, learnable matrices $W^Q \in \mathcal{R}^{d_{\text{model}} \times d_k}$, $W^K \in \mathcal{R}^{d_{\text{model}} \times d_k}$ and $W^V \in \mathcal{R}^{d_{\text{model}} \times d_v}$ project the embeddings into their corresponding **Q**uery, **K**ey and **V**alue representations. The projections are learned in a way so that the softmax of the scaled product of query and key

$$\text{AttentionScore}(Q, K) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$$

is high at row i and column j if token j is relevant for the contextualized meaning of token i.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

The result is then multiplied with the tokens' value representations so that the final contextualized representation of each token i is the sum of all tokens' value representations, weighted by the attention scores.

While this allows the model to attend at every token all other token-embedding, it comes with the limitation that the learnable matrices W^Q,W^K,W^V need to ensure that the dot-product attention captures all relevant types of relations between the tokens. To alleviate this problem Vaswani et al. (2017a) instead introduce h attention heads per attention sub-layer each, where each head has its own set of projection matrices. The attention outputs of each head are then concatenated and passed through a linear transformation W^O. In order to not increase the overall amount of model parameters, each projection matrix reduces the dimensionality of the representation from d_{model} to d_{model}/h.

If we consider an attention head a model that must fit some target mapping $\mathcal{F}$, we can interpret multi-head attention as an ensembling mechanism. Instead of employing one model with n parameteres, we choose to ensemble k smaller models with $\frac{n}{k}$ parameters and combine their predictions through a learnable linear transformation W^O. If $\mathcal{F}$ can be expressed as a linear combination of k less complex functions f_1 to f_k, the ensemble can dedicate a model to each of those functions. From

a linguistic point of view, it is reasonable to assume that the overall attention function $\mathcal{F}$ is in fact a combination of different token-relations such as verb dependencies or anaphoras. Vaswani et al. (2017a) demonstrate that the individual attention heads indeed capture such linguistic relations.

2.2 Residual Connections

Neural Networks with many layers suffer from the so called degradation problem (He et al., 2015). Assume we increase the size of a neural network by adding layer by layer to it. Initially, training performance will improve with increased number of parameters. However, this improvement comes with diminishing returns and eventually model accuracy is saturated. If we now have a model of n layers and add another additional m layers, it should nonetheless perform at least as well as the one with only n layers. After all, it can simply use the same n layers and have the remaining m layers model the identity function. The experiments by He et al. (2015) show, however, that model performance instead deteriorates, as the more complex model is harder to optimize and the additional layers fail to approximate the identity function.

Let $\mathcal{H}$ be the target function a layer needs to map. We now introduce a skip at each layer in our network by adding the layer's input to the layer's output. With hidden representation of the k'th layer h_k, weights w well as an activation function σ, we can formulate a layer with residual connection as follows

$$h_k = h_{k-1} + \sigma(W_k h_{k-1})$$

While normally the layers would struggle to map the identify function, they can instead now just drive the weights to 0 to achieve the desired effects.

In a realistic scenario, the layer will likely never have to fit the identity function. However, $\mathcal{H}$ is likely more similar to the identity mapping than it is to the zero-mapping (He et al., 2015). Instead of modeling $\mathcal{H}$ directly, the layer instead only needs to map the residual function $\mathcal{F}(x) = \mathcal{H}(x) - x$, where x is the input to the layer. If only minor alterations to x are necessary, the function $\mathcal{F}$ is easier to fit. More importantly, an 'unnecessary' layer cannot harm the overall network anymore. The back-propagation of the errors would just shut down the entire layer by making the weights approach zero.

While the motivation behind residual connections is problem oriented, we can also interpret them as a design choice to simulate ensembling of multiple models of various complexities. At each layer, we essentially average[1] the outputs of two models; one that contains the layer and one that does not. We can exemplify this effect of residual connections for three layers h_1 to h_3 where $\hat{h}_k(x)$ is the output of the respective layer and x is the input to the first layer.

$$\begin{aligned} h_1 &= x + \hat{h}_1(x) \\ h_2 &= x + \hat{h}_1(x) + \hat{h}_2(h_1) \\ h_3 &= x + \hat{h}_1(x) + \hat{h}_2(h_1) + \hat{h}_3(h_2) \end{aligned}$$

Essentially, we add an additional model to the ensemble at each layer. Naturally, these models share weights as each new model contains the weights of the previous models plus the additional weights of the new layer. Although the ensemble implicitly constructed through residual connections is likely not as powerful as an ensemble of models with independent weights, we can expect beneficial effects.

The transformer architecture contains residual connections around each attention and feed-forward sub-layer. After adding the sub-layer's input x to the attention output $\textit{MultiHeadAttention}(x)$ or the feed forward layer output $\sigma((Wx) + b)$ respectively, layer normalization is applied (Ba et al., 2016).

2.3 Dropout

Dropout is a regularization technique that aims to approximate the Bayesian gold standard prediction. This prediction is the average output of all possible models, where each model's output is weighted by the model's parameters' posterior probability given the training data (Srivastava et al., 2014; Gal and Ghahramani, 2015). In other words, dropout approximates the expected output of a model given the training data.

Dropout works by dropping out neural network units with a probability of $1 - p$ at each training step. Given some neural network hidden layer h with weights W, biases b, and activation function σ, the feed-forward operation with nodes dropping out with probability of $1 - p$ can be described as

[1] Technically we sum the outputs of the two models rather than average them. However, since usually layer normalization is applied afterwards, the result is identical.

follows:

$$r \sim \text{Bernoulli}(p)$$
$$y = \sigma\left(W\left(r * h\right) + b\right)$$

, where $*$ is the element-wise product. If we apply this to all layers of neural network we end up sampling a different 'thinned' neural network at each training step. A neural network of n nodes thereby becomes an ensemble of up to 2^n networks with shared weights. Given a large enough n, however, most thinned networks will never be sampled throughout training and even less likely sampled multiple times. The practical consequence of this is that no thinned net has a chance to perfectly fit its weights to observation noise, especially since it also needs to share control over those weights with $2^n - 1$ other nets.

At test time we are interested in the average prediction of the 2^n possible nets to approximate the bayesian gold standard prediction. However, it is computationally not feasible to explicitly average the predictions. Instead, Srivastava et al. (2014) propose an approximate averaging method. If network units are retained with probability p at train time, we retain all of them at test time and multiply their outgoing weights by p to acquire the expected output of the node. In their experiments, Srivastava et al. (2014) observe significantly improved generalization error when dropout is applied to neural networks.

The transformer architecture as proposed by Vaswani et al. (2017a) differentiates between hidden dropout and attention dropout. **Hidden dropout** is applied to the sub-layer output right before adding it to the sub-layer's input (the residual connection). The residual connection therefore circumvents the dropout. While I am uncertain of the motivation behind this design choice, I suppose this is because the residual connections alleviate the degradation problem and thinning them out through dropout could prevent them from doing so effectively. **Attention dropout** is applied in the attention mechanism right after the softmax of the query-key matrix product (Klein et al., 2017). Consequently, some features of the attention scores are zeroed out. By doing so, we prevent the model from overfitting the projection matrices transforming the inputs into their query and key representations.

3 Experimental Setup

In the experiments, I use the RoBERTa model for a masked language model task as implemented in the 'transformers' library (Wolf et al., 2020). Training a masked language model is a resource intensive task and due to GPU-memory and time limitation, I was forced to scale down the overall size of the model. In particular, the number of layers is reduced to 4 (or 8 sub-layers respectively) and the maximum sequence length of the input to 256.

If multi-head attention is employed, the model features 8 attention heads. If Attention dropout or hidden dropout is set, nodes are dropped out with $p = 0.1$ probability. To the best of my knowledge, the RoBERTa implementation features only two kinds of residual connections. One circumventing the attention sub-layer and one the hidden sub-layer. If residual connections are removed, the input to the sub-layer is simply not added to the output before layer normalization.

The masked language model is fit on the *Books* data set by Zhu et al. (2015). I limit the data to only the initial 10% to allow the training of multiple epochs within reasonable time. The last 10.000 data points of the data set are used as test set. Learning rate is slightly higher than usually at 0.0002 to allow faster convergence at the cost of less fitness.

4 Results

We evaluate the contributions of the ensemble mechanisms in the transformer architecture in multiple iterations. First, we start with a 'baseline', where all mechanisms have been removed. At each iteration, we add each remaining mechanism to the model, measure the performance and then keep only the best performing one to enter the next iteration. After four iterations we will consequently have constructed a full transformer.

4.1 Iteration 1 - Baseline

The first iteration demonstrates the technical necessity of the residual connections. As figure 1 shows, models without them cannot fit the training data. This is surprising, as the transformer used for the analysis features only 4 layers or 8 sub-layers respectively and should therefore not suffer from the degradation problem discussed in 2.2. Instead, the problem likely lays with the activation functions. For inputs smaller than -2, the GELU function approaches 0 and so does its derivative. Although far less extreme than with the RELU activation

Model	Perplexity
baseline	708.29
+ residual connections	**10.46**
+ attention dropout	655.21
+ hidden dropout	703.24
+ multihead	680.18
residual connections	10.46
+ attention dropout	10.91
+ hidden dropout	10.78
+ multihead	**7.88**
residual connections + multihead	7.88
+ attention dropout	**8.06**
+ hidden dropout	8.62
full transformer	8.75

Table 1: Model Perplexity on Test Set

function, the model can make use of this and learn weights that purposefully zero out the input to the layer. However, this also will make the respective partial derivative approach 0. If this occurs in each layer to sufficiently many features the entire gradient will vanish eventually. Note that this problem is not exclusive to the GELU activation function. In fact, GELU is designed to improve on RELU by alleviating this problem to some degree (Nguyen et al., 2021). The residual connections help prevent

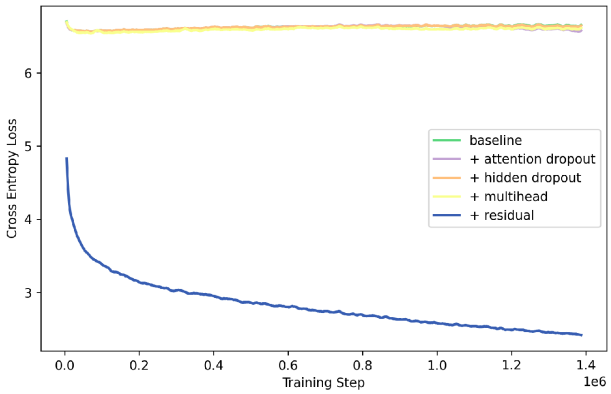

Figure 1: Base Model - Baseline

the gradient from vanishing. Consider an input x and a layer h with output $h(x) = f(x) + x$, where $f(x)$ is the mapping of the layer's forward pass and the input x is added to the mapping as residual connection. With loss ℓ, the partial derivative would then be respectively (Adaloglou, 2020):

$$\frac{\partial \ell}{\partial x} = \frac{\partial \ell}{\partial h}\frac{\partial h}{\partial x} = \frac{\partial \ell}{\partial h}\left(\frac{\partial f}{\partial x} + 1\right) = \frac{\partial \ell}{\partial h}\frac{\partial f}{\partial x} + \frac{\partial \ell}{\partial h}$$

As we can see, we receive an addition of two terms of which the latter is spared from the chain rule. Even if the former term vanishes, the second will not and consequently neither the entire gradient.

The experiment shows that residual connections are an integral part of transformers. However, this is much less so due to the implicit ensembling that occurs with them, but much rather due to technical limitations they help overcome.

4.2 Iteration 2 - Residual Connections

Once the model is stabilized through the use of residual connections we can effectively compare the benefits of the two dropout types and the multihead attention. As Vaswani et al. (2017b) also record in their experiments, using multiple attention heads rather than just one substantially improves model performance.

Attention and hidden dropout both appear to worsen test perplexity (see table 1). However, we likely should not draw conclusions from this observation for multiple reasons. First of all, training convergence could not be reached. It is intuitive to assume that dropout increases necessary training time. As 10% of all nodes are dropped out at each train step, we can naively assume the overall training to require at least 10% more time. Secondly, we use model perplexity as evaluation metric. As dropout prevents overfitting, the model learns to be less confident about its predictions which will in turn cause model perplexity to be higher, even if the model would land the same predictions as a model without dropout. Vaswani et al. (2017a) evaluate the transformer on a downstream task and record increased performance with dropout rates. Unfortunately, they do not distinguish between hidden layer dropout and attention dropout. Figure 2

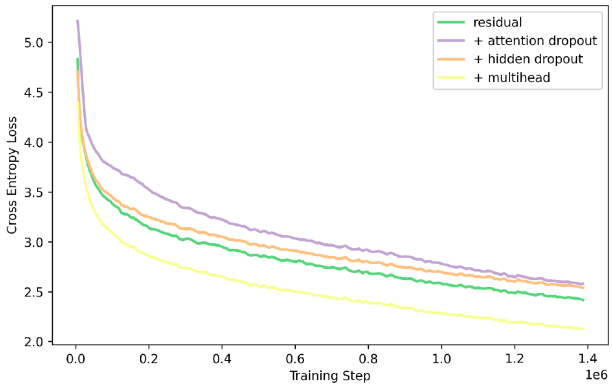

Figure 2: Base Model - Residual Connections

shows that the transformer with attention dropout

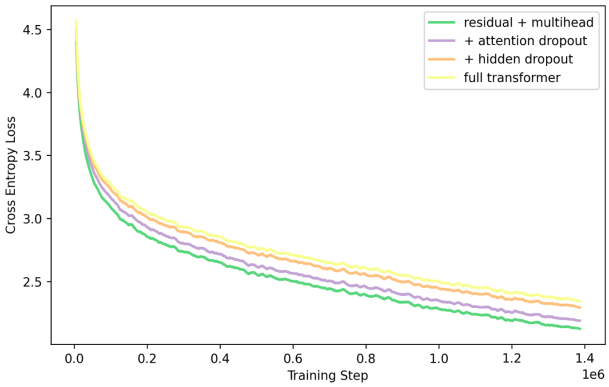

Figure 3: Base Model - Residual Connections and Multi-head Attention

initially converges much slower than the one with hidden layer dropout but then eventually catches up. However, this is entirely sensible as the attention sub-layers are the core of the transformer. If dropout is applied there, the query, key and value projection matrices are learned slower. Without a proper working attention mechanism, the feed-forward sub-layers have very few to work with and therefore the overall model performs worse. This observation suggests that it might be worthwhile to set attention dropout to 0 initially and then only slowly increase it to the desired value within the first quarter of training steps.

4.3 Iteration 3 - Residual Connections + Multi-Head Attention

The final iteration once again seems to show harmful effects of dropout. As discussed in section 4.2, we should not draw conclusions based on the test perplexities when comparing models with dropout to the ones without. We can nonetheless make the noteworthy observation that attention dropout no longer causes the model to struggle throughout the initial part of training. In iteration 2, attention dropout was applied to a single attention head. This head had to attend all possible relations between tokens and therefore already map a highly complex function. Applying dropout makes this an even more difficult task. With multiple heads, each head can focus on an simpler mapping. Additionally, each of the 8 heads has only $\frac{1}{8}$ the amount of parameters. Fitting less parameters is naturally a faster, less complex, task as well. Dropout in turn has a less drastic effect on model convergence speed.

5 Conclusion

The experiments allow following conclusions. Residual connections are an integral part of the transformer architecture and removing them renders the model useless. This however, is likely much more due to the preservation of the gradient rather than due to implicit ensembling. The usage of multiple smaller attention heads instead of one larger one provides a substantial boost in performance. Each head can model a different type of attention mapping and the transformer as a whole ensembles the different heads' outputs. Dropout proved to have negative effects on model performance, although this can be explained by shortcomings in the experimental setup. A final observation is that attention dropout can substantially slow model convergence if used with only one large attention head, but not when employed together with multi-head attention.

References

Nikolas Adaloglou. 2020. Intuitive explanation of skip connections in deep learning. *https://theaisummer.com/*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Yarin Gal and Zoubin Ghahramani. 2015. Dropout as a bayesian approximation: Representing model uncertainty in deep learning.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition.

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

Anh Nguyen, Khoa Pham, Dat Ngo, Thanh Ngo, and Lam Pham. 2021. An analysis of state-of-the-art activation functions for supervised deep neural network.

Jonathan Rougier. 2016. Ensemble averaging and mean squared error. *Journal of Climate*, 29(24):8865 – 8870.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017a. Attention is all you need.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017b. Attention is all you need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.