Dynamic Transformer Networks

Amanuel Mersha [1]

Abstract

The recent deep learning breakthroughs in language and vision tasks can be mainly attributed to large-scale transformers. Unfortunately, their massive size and high compute requirement have limited their use in resource-constrained environments. Dynamic neural networks could potentially reduce the amount of compute requirement by dynamically adjusting the computational path based on the input. Similar to soft attention, this work presents a simple way of constructing an oracle function that enables a transformer network to determine the dependency between its layers. It can then be used as a strategy to skip layers without a reinforcement learning agent. We show that such a model learns to skip, on average, half of its layers for each sample in a batch input.

1. Introduction

Transformers have been a major force behind the recent success of deep learning. These models, at their core, employ a multi-head self-attention mechanism that enables them to explicitly define relationships between inputs in terms of attention score. This mechanism has been shown to leverage very large datasets and performs significantly better than previous methods in NLP tasks (Devlin et al., 2018; Brown et al., 2020; Liu et al., 2019). Furthermore, the self-attention building block increases the model's explainability, helping researchers to understand neural networks by probing the attention heads. Later, (Dosovitskiy et al., 2021) showed that transformers could also be applied to vision tasks with little to no modification and perform on-par with CNN-based models. This unique ability of transformers, generalizing to different kinds of tasks (vision, language, multi-modal learning), made them more ubiquitous in the literature.

The past few years have seen significant advances in dynamic neural network research as well. These networks modify their inner workings (parameters, layers, connectivity) based on an input sample. Hence, a dynamic neural network can be perceived as an ensemble model in which a *sub-model* in a model can act upon a certain type of input. Thus, they promise efficiency, better representation power, generality and interpretability (Han et al., 2021).

(Dehghani et al., 2019) introduced the Universal Transformer (UT) architecture which applied the concept of recurrence, not over the input symbols, but over the depth-wise revision of the symbols. The technique only uses a single layer, which is repeatedly executed over the inputs. This enables the model to determine the depth length, and thus incorporate dynamic halting that is applied per-symbol level. Hence, some of the symbols in a given input may not halt with other symbols, in which case, the successive depth is just a copy of the halted symbols.

(Elbayad et al., 2020) improved the Universal Transformer by increasing the model capacity through incorporating additional layers. Their model, the Adaptive Depth Transformer, introduced several ways to estimate the depth and exit strategies. The token-specific depth determines the depth for every token in a block. For the sequence-specific depth estimation, an oracle function, given the average of the tokens in a block, predicts the depth.

(Wang et al., 2018) proposed SkipNet which is partly an inspiration to our work. SkipNet employed gating modules that decide whether to skip residual blocks or not. The gating functions are implemented in three ways: FFGate-I and FFGate-II are standard CNN based networks where FFGate-II has higher capacity. RNNGate on the other hand uses Recurrent Neural Network (RNN) in order to share parameters across multiple blocks. The work also introduced a reinforcement learning agent (Hybrid-RL) that mitigates the non-differentiable function of executing the gates.

In this work in progress, we propose a simple technique that makes transformers dynamic through soft-attention and layer skipping mechanisms. The layer skipping transformer is important because, as the self-attention mechanism incurs quadratic memory and compute cost, skipping a layer reduces the resource need. However, this paper presents the dynamic transformers evaluated on a limited experiment set due to resource constraints. Hence, instead of the literature

[1]School of Information Technology and Engineering, Addis Ababa Institute of Technology, Addis Ababa, Ethiopia. Correspondence to: Amanuel Mersha <amanuel.negash@aau.edu.et>.

DyNN workshop at the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, 2022.

standard datasets such as ImageNet, a relatively smaller ViT model is tested on CIFAR100 to showcase the feasibility of the concept. The models are discussed in section 2 and findings from the experiments are presented in section 3. Finally, although the technique presented here is relatively simple and is not novel by itself, appendix A discusses the reasoning and the hypothesis behind it.

2. Models

2.1. Weighted Stochastic Dense Connection (WSDC)

DenseNet (Huang et al., 2017) demonstrated that dense connectivity of current and previous layer activations is advantageous because it allows for feature reuse. The input passes through several layers in a given block, and at each layer, activations from previous layers are concatenated and fed to the next layer. The authors demonstrated that a layer may often attend to an output from a much lower layer rather than an immediately preceding layer. This increased the network's expressivity, resulting in feature reuse. Inspired by the findings, we set out to create a similar mechanism for transformers in the hope that the upper layers would have direct access to lower layer activations. In a transformer network, however, unlike DenseNet, concatenation of the layer activations results in a much higher computation and is thus replaced by the weighted average of the activations. The activation weighting mechanism is described further below.

Conventions: For the equations below, x_j represents inputs to the jth transformer layer T_j (MSA + FFN) (Vaswani et al., 2017). a_j represent activations of T_j. In a typical transformer model, $a_j = x_{j+1}$ as the activations pass without any modification. However, the formulations presented here apply some kind of function on the a_j resulting in different x_{j+1}. x_0 is the image patch, $x_1 = a_0$ is the encoding of the patches into the transformers *hidden* size. a_1 is the output of the first transformer layer T_1 and $x_2 = a_1$.

As a baseline, the standard ViT architecture (Dosovitskiy et al., 2021) with minimized capacity (due to computational constraints) is trained from scratch. In appendix B, table 2 shows the hyperparamaters the base model was trained on.

(Simoulin & Crabbé, 2021) hypothesized that the *CLS* token aggregates the representation of the entire input sequence, which is also supported by the fact that this token is used for classification purposes in the final layer of ViTs (Dosovitskiy et al., 2021). Hence, one can safely assume that an oracle function q^* could predict what the input to a layer should be given this token from an immediately preceding activation. For brevity, c_j represents the *CLS* token in x_j. An oracle function q_j^* is a single layer feed-forward network that is fed c_j and predicts weights for each activation $a_k \in \{a_0, a_1, ...a_{j-1}\}$ using $softmax$. This oracle

Figure 1. Layer dependency visualization for WDC model. $T_1, T_2, .., T_9$ are transformer layers. It should be read vertically: how much T_x depends on its preceding layers. Ex: T_2 depends 95% on layer 1 and 5% on layer 0. T_3 depends 66% on layer 2, 11% on layer 1 and 23% on layer 0.

function q_j^* is implemented for each layer and will also be utilized during inference. The predicted weights are then used to aggregate all the previous activations into a single activation as shown in eq (1) to (4).

$$w_j = softmax(q_j^*(c_j)) \tag{1}$$

$$^*w_j = dropout(w_j) \tag{2}$$

$$x_j = \sum_{k=0}^{j-1} {}^*w_j^k a_k \tag{3}$$

$$a_j = T_j(x_j) \tag{4}$$

The dropout is applied to the weights so that complete dependence on the previous individual layers is minimized. It increases quadratically from 0.0 to 0.25 as the number of layers increases. Passing the activation weights w_j without applying dropout in eq (2) results in weighted dense connection (WDC). While this model (WDC) didn't improve performance upon the base model, it brought significant insight on how the oracle mechanism operates. The dependency relationship can be visualized using the weights in eq (1). Fig 1 and 2 shows the dependency graph after collecting weights for 256 images and averaging them across the images. Figure 1 shows that like DenseNet, in WDC, some of the upper layers directly utilize lower layer activations, ignoring the ones preceding them. For example

T_9 depends 26% on layer-0 (the patch encoding layer) and 43% on layer-7. Layer T_8, on the other hand, relies 81% on layer-4.

2.2. Dynamic Transformer (DT)

WDC revealed an important fact: layers do not always rely on their immediately preceding activations, and thus they can utilize direct access to lower layer activations. Similar to SkipNet (Wang et al., 2018), this can be thought of as an oracle function guiding lower layer activations to pass to upper layers without routing through middle layers. Hence, the oracle function in WDC can be modified to predict whether to skip the next layer or not, given the *CLS* token and a skip history.

For each sample in a batch, for $j \geq 2$, the oracle function q_j^* predicts a sigmoid-based skip score s_j given the *CLS* token c_j and the skip history h_j. Then, based on this score, the next layer will be skipped or not. The skip history is simply a concatenation of previous skip scores $s_1, s_2, .., s_{j-1}$. Initially, $s_1 = \mathbf{1}$ as all samples must pass through at least in one transformer layer. Those samples x_j^T with $s_j >= 0.5$ are extracted from the batch and pass through the next layer. The remaining, x_j^I, will be cloned to become a_j^I. After the first group passes through the transformer layer T_j, the activations a_j^T will be merged with the a_j^I. The merging is simply placing each activation in its original index. Equations (5) - (12) show how the skipping mechanism is implemented.

$$h_j = concat(s_1, s_2, .., s_{j-1}) \tag{5}$$
$$skip\_score = q_j^*(c_j, h_j) \tag{6}$$
$$s_j = sigmoid(skip\_score) \tag{7}$$
$$x_j^T = x_j(s_j >= 0.5) \tag{8}$$
$$x_j^I = x_j(s_j < 0.5) \tag{9}$$
$$a_j^T = T_j(x_j^T) \tag{10}$$
$$a_j^I = x_j^I \tag{11}$$
$$a_j = merge(a_j^T, a_j^I) \tag{12}$$

The final activation a_j passes through one more transformer layer followed by a classifier layer. (Wang et al., 2018) used a Hybrid-RL technique where an RL agent handles the un-differentiable function that decides whether to skip a layer or not. In our implementation, a skip score of each sample s_j, is stored and fed to the oracle on each successive layer, which allows the network to propagate these decisions up to the final layer. However, a naive implementation of this oracle function results in a model that maximizes layer connections, leading to a full wired network. To mitigate this issue, a skip loss $\mathcal{L}_\int$ that is dependent on the skip score prediction can be constructed, forcing the model for a higher amount of skips. Combining it with cross entropy loss $\mathcal{L}_\rfloor$, the final loss $\mathcal{L}$ is given by:

$$\mathcal{L}_\int = \frac{1}{L}\sum_{l=1}^{L}(skip\_score)^2 \tag{13}$$
$$\mathcal{L} = \mathcal{L}_\rfloor + \mathcal{L}_\int \tag{14}$$

L is the number of layers that operate based on the oracle. This simple technique allows the optimizer to update the oracle function based on its decision. However, such a mechanism comes with a downside of its own and is discussed in subsection 3.2. The memory cost to implement the oracle function is 0.6%, a 0.1M parameter increase on a 14.4M parameter model. In general, if a transformer model has L number of layers and a hidden size of h, the amount of parameters that is required to construct the oracle functions for all the layers is:

$$\sum_{l=2}^{L}(h + l - 1) \tag{15}$$

This is because, at each layer, a single layer FFN predicts whether to skip the next layer or not, given a *CLS* token size of h and previous skip history of length $l - 1$.

3. Experiment and Result

Each of the models was run for 200 epochs with the hyper-parameter specified in table 2. As the batch size increases from 256 to 1568, all models perform better. However, in the DT model case (2.2), we found that it is consistently and significantly sub-optimal when compared to the base model for the same batch size. Precisely, to achieve comparable level of performance, it needed twice the batch size of the base model. Investigating this, we discovered that, due to the skipping mechanism, the model processes only about half of the given batch in a layer, which explains why it requires twice the batch size to compete with the baseline. Hence, to make a fair comparison, the batch size was doubled to 1568 and the epoch increased to 400, as higher batch size needs more epochs in all the models. Finally, the experiments were run on 10 and 16-layer models to evaluate how scale affects the models.

3.1. Performance

As table 1 shows, in the 10-layer case, WSDC has shown better performance than the base model, while DT and other models were sub-optimal. In 16-layer case, however, WSDC came second to the base model and was lower in performance when compared to the layer 10 setting. It's probable that a better dropout rate fixes the performance drop as it is the only factor that is affected by the scale increase.

Model	val-acc-10	val-acc-16
Base	68.11	68.29
WDC	67.46	67.55
WSDC	**68.98**	67.76
DT	67.18	66.62

Table 1. Performance of the models on CIFAR100. val-acc-10 and val-acc-16 are the validation accuracy of the 10 and 16 layer models respectively.

3.2. Skip-Connection Loss

The dynamic transformer (DT) learns faster in both layer-10 and layer-16 models. However, validation accuracy stops increasing even though it did not overfit. We believe that this is because of the loss that punishes the connections of layers in equation 13. If roughly half of the connections are skipped, the skip loss becomes zero. This is a strong bias and thus not optimized. Thus, an RL agent seems more appropriate to make better decisions while skipping layers as rewards can be driven from both the train losses and accuracy.

3.3. Observation

- As figure 6 shows, the base model overfits on the train dataset while the proposed models achieve similar validation accuracy without overfitting as much as the base model. It is possible that proper hypermarameter tuning might unlock their learning capacity as the base model's hyperparamaters may not work well due to the extra manipulations applied on the layer activations. The performance of the WSDC model in the layer-10-model is an indicator of this assumption.

- In WDC case (figure 1), layer 9 receives 26% of its information from layer 1, which is the patch encoding layer. Consequently, it almost doesn't require its preceding layer's activation. There is a consistent reliance of layers on their preceding outputs up until layers 3, similar to a standard transformer network. This pattern breaks after the fifth layer, implying that the construction of higher level representations still necessitates low level features such as raw patch encoding outputs. Similar dynamics is visible in the WSDC model case (figure 2).

- In the WSDC case (figure 2), because of the dropout on the weights, the model produced almost evenly distributed weights over the previous activations instead of extreme values. It's similar to stochastic depth (Huang et al., 2016) but in this case, the model can chose which lower layer to connect and how strongly. Given the success of Stochastic Depth as a regularizer, such wiring can be expected to benefit deeper ViT's through improved gradient flow.

- Furthermore, we can see how hyperparameter affects a model. Figure 3 depicts the weight visualization of WDC with dropout of 0.0 while figure 4 shows visualization of a model with drop 0.2. In both cases, the left-side figures show the weight predictions of correctly predicted images while right-side figures show weight prediction for incorrectly predicted images. In the model that was trained using $dropout = 0.0$, layer 9 gets about 50% of its information from the patch encoding layer. This is very different from the model trained with $dropout = 0.2$ which shifts majority of this dependency to layer 7, although layer 9 still relies 26% on the encoding layer. Similar behaviour is visible in WSDC (figure 2) case. Due to both the dense connection and normal dropouts, upper layers are more dependent on layers $j >= 3$. It seems that, the dropout hyperparamater forces the model to rely on higher level features.

4. Conclusion

This paper presents soft-attention and layer skipping dynamic transformer networks. The methods heavily rely on the *CLS* token to determine how inputs are routed as they pass through layers. In the case of weighted stochastic depth connection (WSDC), an oracle function, similar to attention, determines the weights of previous activation. In the layer skipping dynamic transformer (DT) case, the oracle function determines whether a sample should pass through a layer based on the skip history and the *CLS* token. While WDC allows for better transformer explainability, DT provides memory and compute efficiency, potentially allowing edge devices to enjoy the benefits of transformers.

Being an unfinished work, there are several analyses that need to be done. The DT model can be much more efficient in both resource and performance by utilizing an RL agent and we predict that this RL agent will cost less than the current oracle function. The linear cost in equation 15 can be reduced to a constant cost that is independent of the number of layers. It's also worth combining DT and WSDC to take advantage of layer skipping and the sum of weighted activations. The models also provide ample opportunity from explainablity standpoint, as probing the weights and layers is now much easier. Finally, hyperparameter tuning and testing the models on state of the art experiment settings are critical.

Acknowledgements

The author would like to thank Tewodros Wondifraw Ayalew for his support on computing power, Selameab Setargew Demilew and Sammy Assefa (PhD) for their support on

various aspects.

References

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T. J., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

Dai, Z., Liu, H., Le, Q. V., and Tan, M. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977, 2021.

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

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2021.

Elbayad, M., Gu, J., Grave, E., and Auli, M. Depth-adaptive transformer. *ArXiv*, abs/1910.10073, 2020.

Guo, J., Han, K., Wu, H., Xu, C., Tang, Y., Xu, C., and Wang, Y. Cmt: Convolutional neural networks meet vision transformers. *arXiv preprint arXiv:2107.06263*, 2021.

Han, Y., Huang, G., Song, S., Yang, L., Wang, H., and Wang, Y. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Huang, G., Liu, Z., and Weinberger, K. Q. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., and Dosovitskiy, A. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34, 2021.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Simoulin, A. and Crabbé, B. How many layers and why? an analysis of the model depth in transformers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, pp. 221–228, 2021.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34, 2021.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, X., Yu, F., Dou, Z.-Y., and Gonzalez, J. Skipnet: Learning dynamic routing in convolutional networks. *ArXiv*, abs/1711.09485, 2018.

Wu, H., Xiao, B., Codella, N. C. F., Liu, M., Dai, X., Yuan, L., and Zhang, L. Cvt: Introducing convolutions to vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22–31, 2021.

Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., and Feris, R. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8817–8826, 2018.

A. Motivation

In the original Vision Transformer paper (Dosovitskiy et al., 2021), the authors showed that the ViT models overfit on the imagenet-1k (Russakovsky et al., 2015) dataset and thus need to be pretrained on JFT-300 dataset (Sun et al., 2017) in order to surpass CNNs in performance. Later works (Wu et al., 2021; Guo et al., 2021; Dai et al., 2021) tried to incorporate CNNs into transformer. This sparked the question, why do Transformer as well as MLPs (Tolstikhin et al., 2021) overfit on the standard dataset, and what can be done to curtail the issue? The obvious way is pretraining. However, architectural treatments such as merging with CNN, as well as initialization techniques, also play a significant role to mitigate the issue. Ultimately, however, the fact that ViT model was trained using the distillation technique without a large dataset(Touvron et al., 2021) and performed just like CNNs shows that there is still room for improvement on vision transformers without the need for the larger dataset.

While surveying the representation power of architectures, the following observations were made:

- (Raghu et al., 2021) showed that ViTs have uniform similarity between lower and higher layers while in ResNet, such similarity doesn't exist. (Kornblith et al., 2019) also observed the fact that ResNets don't have uniform representation across lower and higher layers. (Raghu et al., 2021) ViTs rely highly on the skip connection for their uniform layer representation and turning these connections off results in a performance drop. This meant that a layer in ViT has a very high dependency on the information that flows from lower layers.
- (Raghu et al., 2021) showed that ViTs and ResNets learn similar representation in the lower layers, despite being different architectures. Such phenomenon was also uncovered by (Kornblith et al., 2019) (a) between ResNets that are trained with CIFAR10 and CIFAR100 and (b) between ResNets that are trained with different weight initialization. This meant that lower-level representation is general to different architectures and datasets when compared to higher-level representation.
- (Raghu et al., 2021) also showed that as the level of the layers increases from the lower the higher, ViT's attention focus on global features rather than local ones. They showed that ViT-H and ViT-L don't learn to attend locally when trained on imagenet-1k, in which case they overfit. For the ViT-B/32, however, lower layers learn to focus on local features. Combining the above thought with this one, we hypothesize that as the number of layers of the model increase, the depth forces the attention span to get wider. But, if there is a mechanism by which such depth can be systemically lowered while still maintaining the original depth physically, models such as ViT-B and ViT-L can learn to attend locally.

Based on our observation and hypothesis, implementing the following tweaks will lead to lesser overfitting transformers:

- Minimize the dependency of layers on the layer that immediately precedes them. This is different from stochastic depth (Huang et al., 2016) or BlockDrop (Wu et al., 2018) in that, in this case, layers can choose from which layer to extract information.
- Systemically minimize the distance between higher and lower while keeping the original depth intact as an option (for very deep networks such as ViT-L/ViT-H).

B. Base model Hyperparams

Hyperparam	Value
Embedding size	384
MLP Hidden	384
Num of layers	10, 16
batch size	768
dropout	0.2
lr	0.001
label smoothing	yes
autoaugment	yes
warm-up epoch	5
precision	16 bit
num of head	12
weight-decay	5e-5

Table 2. Base model hyperparamaters

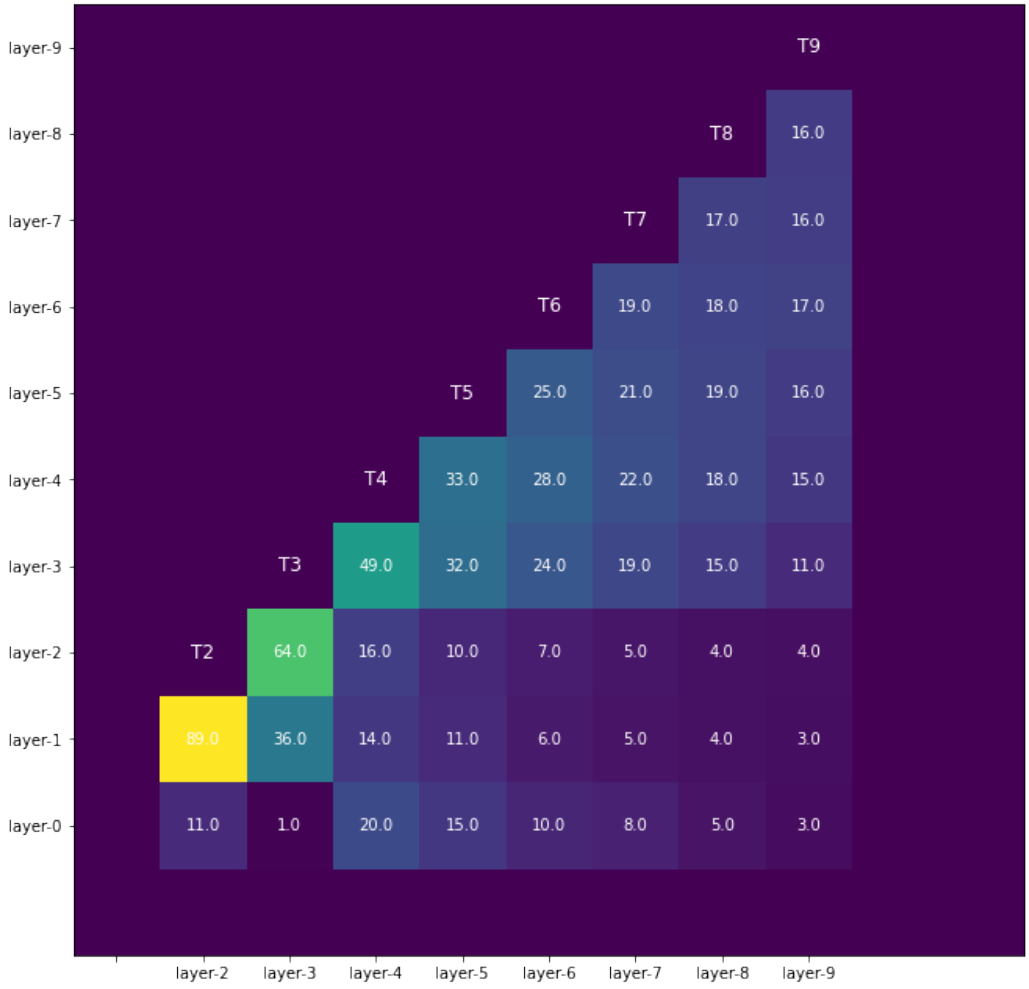

Figure 2. Layer dependency visualization for WSDC model. $T1, T2, .., T9$ are transformer layers. It should be read vertically: how much T_x depends on its preceding layers. Ex: T2 depends 89% on layer 1 and 11% on layer 0. T3 depends 64% on layer 2, 36% on layer 1 and 1% on layer 0.

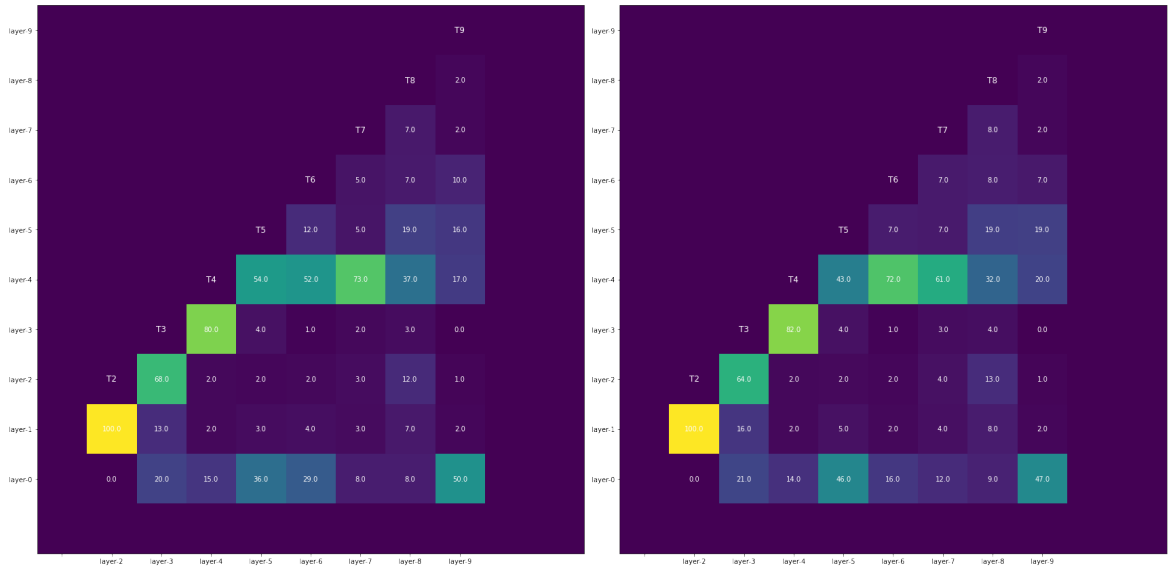

Figure 3. Layer weight visualization of WDC model with dropout 0.0. Left is visualization of correctly predicted images. Right is the weights of wrong prediction

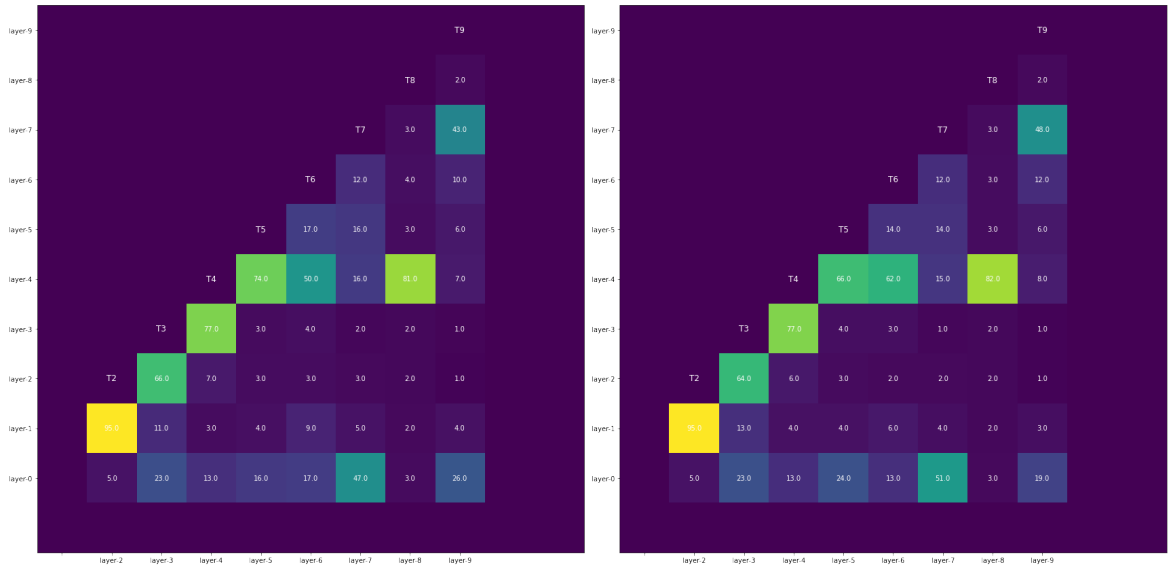

Figure 4. Layer weight visualization of WDC model with dropout 0.2. Left is visualization of correctly predicted images. Right is the weights of wrong prediction

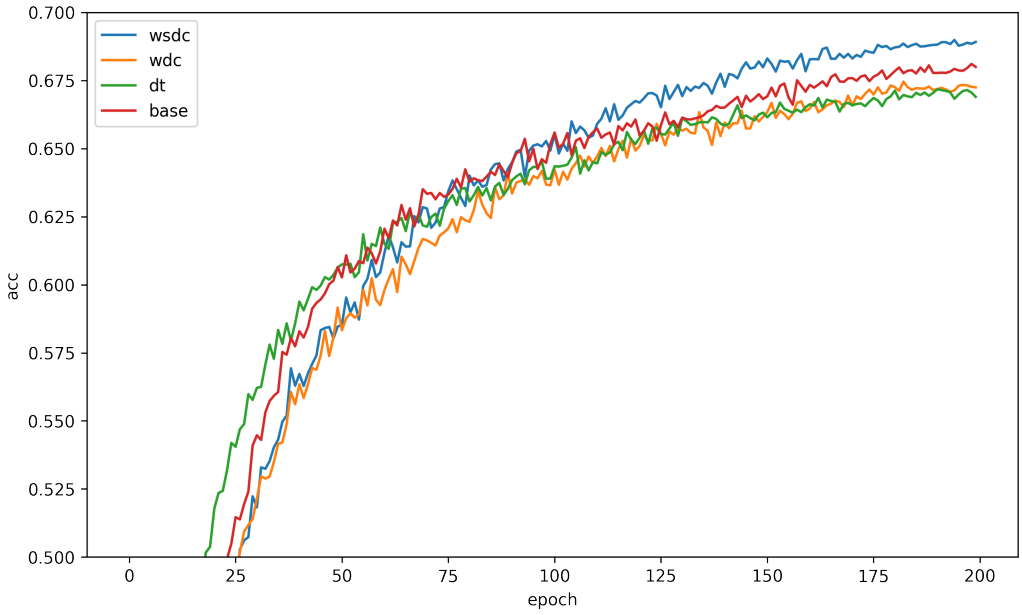

Figure 5. Validation accuracy of the proposed models with 10 layers on CIFAR100

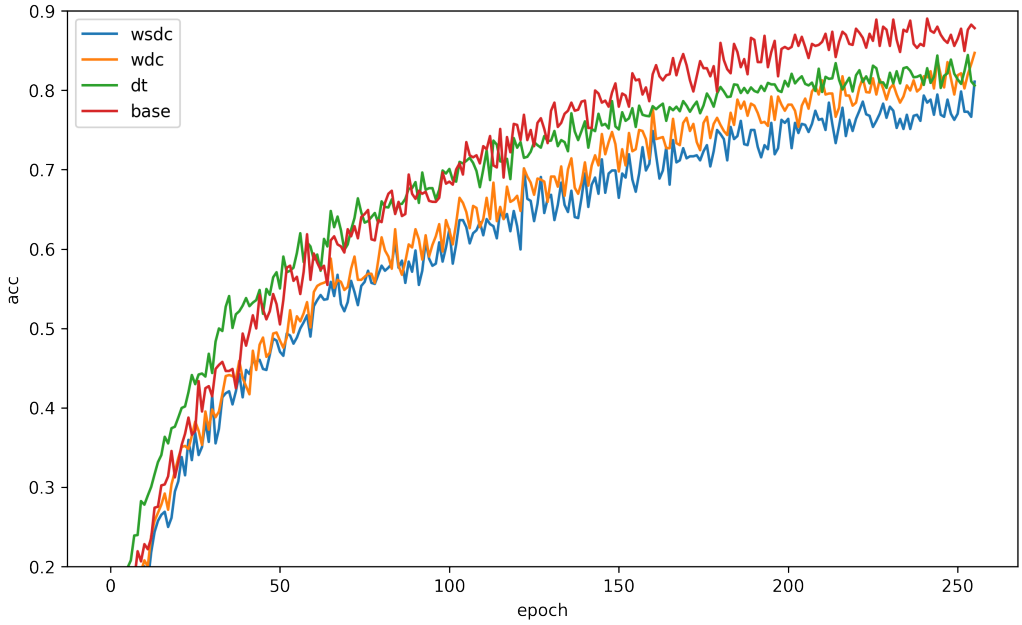

Figure 6. Train accuracy of the proposed models with 10 layers on CIFAR100