Study on a visual coder acceleration algorithm for image classification applying dynamic scaling training techniques

Junliang Huo*, Jiankun Ling*
College of Engineering, South China Agricultural University, Guangzhou, China

*Corresponding author email: 201821120211@stu.scau.edu.cn;
*a 201821120209@stu.scau.edu.cn

Jiankun Ling and Junliang Huo are co-authors.

Abstract. Nowadays, image classification techniques are used in the field of autonomous vehicles, and Convolutional Neural Network (CNN) is used extensively, and Vision Transformer (ViT) networks are used instead of deep convolutional networks in order to compress the network size and improve the model accuracy. The ViT network is used to replace the deep convolutional network. Since training ViT requires a large dataset to have sufficient accuracy, a variant of ViT, Data-Efficient Image Transformers (DEIT), is used in this paper. In addition, in order to greatly reduce the computing memory and shorten the computing time in practical use, the network is flexibly scaled in size and training speed by both adaptive width and adaptive depth. In this paper, we introduce DEIT, width adaptive techniques and depth adaptive techniques and combine them to be applied to image classification examples. Experiments are conducted on the Cifar100 dataset, and the experiments demonstrate the superiority of the algorithm on image classification scenarios.

Keywords: ViT; DEIT; width adaptive; depth adaptive.

1. Introduction
For a long time, CNN networks have been used for the task of processing images because only CNN can handle three-dimensional image data; however, the accuracy of CNN is still not high enough, and in the pursuit of higher accuracy, it has been proposed to apply transformer [1] to image classification [2]. It is necessary to transform three-dimensional data like images into serialized data in order to be available for transformer processing, and this transformer architecture for processing image data is called ViT; however, a very large dataset is required in the pre-training ViT phase, as well as consuming a lot of computational resources and time. So, it was proposed to apply DEIT [3], this network structure is based on transformer architecture, the difference lies in the training strategy, and knowledge distillation is used, DEIT only needs less pre-training data to achieve the effect of ViT. The general model compression methods are quantization, weight sharing, pruning, and distillation; quantization refers to the use of FP16 or INT8 instead of model parameters, but INT8 is not yet popular because it involves more accuracy loss. And weight sharing is that the neurons in each layer share the same parameters. Both methods are applied in ALBERT [4], but the speedup is not significant. Pruning is the
most direct method to remove the weights or neurons of the model to achieve the purpose of reducing the operations. Pruning is also divided into weight pruning and neuron pruning, structural pruning, weight pruning is to set the weight of a connection to 0 to obtain a sparse matrix; neuron pruning [5], is to remove a vector in the matrix directly; model level is structural pruning, which can be a direct removal of a layer. But the disadvantage of pruning is that we don't know which weights, which neurons, and which layers are unimportant, and often the accuracy may decrease after pruning, so a mechanism is needed to determine what exactly should be pruned. Distillation, on the other hand, allows the small model to learn the knowledge of the large model during training, and only uses the small model during prediction, and distillation has worked well so far.

In practice, reducing the number of network layers and distillation by pruning is a common and effective speedup practice. However, since different downstream tasks require different prediction speedups, it is very inconvenient to spend a long time to fine-tune and distill for different downstream tasks each time [6]. A large number of image classification techniques are applied to autonomous vehicle driving technology, but the control center of the vehicle itself has to process a very large amount of information and few computational resources are allocated to image classification; arranging a network with fewer parameters leads to insufficient accuracy of image classification, while arranging a large deep network with more parameters consumes a large amount of computational resources. Therefore, we choose to use ViT network instead of CNN, but pre-training ViT requires a huge dataset, so we use DEIT network instead of CNN, and we use width and depth adaptive techniques [8] to compress and distill multiple sub-networks simultaneously in order to better match other downstream tasks during pre-training.

Method

2. Vision Transformer

(1) Transformer Encoder, ViT is the model that stacks multiple layers of Transformer Encoder.

2.1. Mutil head self attention

2.1.1. self attention. First cut the image into multiple initial vectors (i=1,2,3 in the text ......) and assume a matrix called $W$.

\[ a_i = WX_i \]  

(1)

$a^i$ is the input to the self-attention layer, a step called mapping, and the matrix $W$ is also a learnable object.

(2) Suppose there are three matrices: $W_q, W_k, W_v$. These three matrices are trainable and are shared.

\[ q_i = a_i \cdot W_q \]  

(2)

In the latter case $q$ is used to match with each $k$.

\[ k_i = a_i \cdot W_k \]  

(3)

$k$ will be matched by every $q$.

\[ v_i = a_i \cdot W_v \]  

(4)

$q_i, k_i, v_i$ represents the information extracted from $a_i$.

Since the computation process of Transformer is parallel, it can be written in matrix form, Write the obtained $q_i$; $k_i$; $v_i$ in matrix form respectively: $Q$; $K$; $V$.
\[ \alpha = \frac{Q \times K^\top}{\sqrt{d}} \]  

(5)

\( d \) represents the dimension of \( K \).

\[ \hat{\alpha} = \text{soft max}(\alpha) \]

(6)

Here it means that each row vector of \( \alpha \) passes through the softmax layer.

\[ b = \hat{\alpha} \times V \]

(7)

This formula represents the final output matrix \( b \) combining all the input information, and the effect is similar to that of a convolutional network, so the CNN can be replaced by a transformer architecture.

2.1.2. Mutil head self attention.

The first few steps of the operation with MSA are the same as in equation (1)(2)(3) to obtain \( q_i \); \( k_i \); \( v_i \). Based on the number of heads set previously \( h_j \), divide the dimension \( d \) of \( q_i \); \( k_i \); \( v_i \) by \( h_j \), and it will get an integer, and then according to this integer respectively \( q_i \); \( k_i \); \( v_i \) to achieve a simple equal division, so \( q_i \) is divided into \( q_{i,1}, q_{i,2}, k_i \); \( v_i \) the same. So the last digit of the lower-right subscript is divided into the same group of heads, and each group of heads performs the same operation as self-attention. So the head of the \( j \)th group will get the matrix composed of \( b_{i,j} \). Then concat for the same \( i \). After concat, \( b \) is

\[
\begin{align*}
& b_{i,1} \\
& b_{i,2} \\
& b_{i,3} \\
& b_{i,j-1} \\
& b_{i,j}
\end{align*}
\]

arranged by column vectors as follows

\[
\begin{align*}
& b_{i,1} \\
& b_{i,2} \\
& b_{i,3} \\
& b_{i,j}
\end{align*}
\]

\[ b = W_0 \times (b_{i,3}) \]

(8)

\( W_0 \) is an updatable matrix so that the output of the MSA is obtained.

2.2. The special features of ViT

It uses position encoding because there is a relationship between each input vector. If we simply input each vector directly into the network, it will make the position of different image blocks inside any image change, but the corresponding output is still the same, which is equivalent to ignoring the position information of each vector. In order to use the position information of each input vector, the position information needs to be added. Calculate the position code \( pe_t \) according to the sine and cosine functions method. Add a corresponding \( pe_t \) to each \( a_t \) and then participate in the subsequent
operations. That is, before $a_i = W x_i$, add one corresponding to $p e_j$. The whole image cannot be directly input to the neural network, and the basis of image classification is one vector, which must be transformed into one vector first, but the dimension of one image is too large, so a large image has to be cut into multiple inputs of the same dimension and then transformed into vectors to input to the neural network. Firstly, two hyperparameters are defined, patch-size is how big the image is divided into, and the other hyperparameter is the step size, which has the same meaning as the step size in CNN. After dividing the image, each patch is then stretched into a one-dimensional vector by the expansion package of einops, because the transformer architecture accepts one-dimensional vector input directly. In ViT, an updatable cls-token is added at the location of the first input vector, and the input is in the form of a [CLS] symbol. The reason for using a symbol is that this symbol will become a vector after the final ViT network, and then this vector will be input to an MLP classifier to finally output the probability of an item. So the final output is related to this [CLS] symbol, and the mapping vector of a certain image should not be used as input to avoid the final output being biased towards this input. Finally, the output of ViT is multiple vectors, and there is only one vector that is useful to us, which is the output vector corresponding to the [CLS] symbol. This output is the final classification result of the network. This output is generally used for training to do cross-entropy with the truth label of the output image, and then to find the gradient of each parameter, and then use the chain rule of derivation and back propagation algorithm to update the parameters that need to be updated.

2.3. Commonly used modules for deep neural networks: MLP, add & Norm layers, Layer Norm

Here MLP [9] is simply Neuron Full Connection + GELU Activation Function [10] + Dropout [11]. The effect is to enhance the fitting ability of the model. General deep neural networks add MLP layers. add & Norm layers This part of the structure is borrowed from the residual connection of CNN's famous ResNet [12], which sums the original input vectors and the vectors after MSA, preserving the original information and facilitating model convergence. In order to prevent the degradation of deep neural networks, i.e., we do not know the optimal number of layers, but we set more layers than the optimal number of layers, then we have to do a constant mapping of the extra layers, which is equivalent to the input equal to the output, but due to the design difficulties, the output can be set to 0 directly to achieve the same effect. The general deep neural network will use the residual connection. The general deep neural networks add Layer Norm [13] layer or Batch Norm layer, but because the transformer architecture is similar to RNN [14], Layer Norm is used instead of Batch Norm. Using Layer Norm can improve the training speed and accuracy of the model and make the model more robust. Layer Norm layers are added to deep neural networks in general.

![Figure 1. ViT inference flow](image)
3. DEIT

Because ViT requires huge amount of pre-training data, and the JFT dataset is privately owned by google, we cannot use this dataset for pre-training, and even if we can borrow it, the computational resources consumed are too large. Therefore, we are currently using ViT’s open source model and then tuning it for downstream tasks. We used the DEIT model (untrained) and were able to pre-train with a much smaller dataset and achieve the same accuracy.

The structure of the Deit model is the same as ViT, the difference lies in the subtle differences in the training methods. The only difference between the output of Deit and ViT is that the last input vector of Deit is also [CLS], called distillation-token. This vector can be input to an MLP classifier together with the class token output vector, and the output of the MLP classifier is the final classification result.

This total loss function is mainly used for the BP algorithm to update the parameters.

\[ L_{global} = (1 - \lambda)L_{CE}(\psi(Z_s), y) + \lambda \tau^2 KL(\psi(Z_s / \tau), \psi(Z_t / \tau)) \]  

(9)

This total loss function is mainly used for the BP algorithm to update the parameters.

- \( \psi(Z_s) \) is the class probability distribution of the STUDENT network, which is the result of the class token going through all TRANSFORMER encoder layers.
- \( y \) is ground truth.
- \( L_{CE} \) :Using the cross-entropy loss function for both.
- \( \psi(Z_s / \tau) \) :is the category probability distribution of the STUDENT network, which is the result of the distillation token going through all TRANSFORMER encoder layers.
- \( \psi(Z_t / \tau) \) :is the class probability distribution of the TEACHER network.
- \( KL \) :Use KL dispersion for both.
- \( \lambda \) :weight value, is a hyperparameter.

![Figure 2. Structure of DEIT](image-url)
Knowledge distillation [7]

Knowledge transfer is achieved by introducing soft-targets associated with the teacher network (teacher model: complex, but with superior inference performance) as part of the total loss to induce training of the student network (student model: streamlined, low complexity).

\[
\exp(\frac{z_j}{T}) = \sum_{i=1}^{N} \exp(z_{ij})
\]

(10)

Figure 3. Distillation of DEIT by CNN

This is the improved softmax formula, and \( T=1 \) is the most conventional softmax.

The output of the teacher network (left side of Figure 3) can be obtained as a smooth probability distribution (often called soft target or soft label) with values between 0 and 1, taking a more moderate distribution than the conventional softmax output. The larger the value of Temperature, the smoother the distribution; and the decreasing value of Temperature, the closer to the unique heat vector one-hot, less information, and easy to misclassify and introduce unnecessary noise. The hard label is then the true labeling of the sample, which can be represented by the one-hot vector.

\[
L = \alpha L_{\text{soft}} + \beta L_{\text{hard}}
\]

(11)

The BP algorithm is run according to the total loss function to update the parameters to be updated.

\[
L_{\text{soft}} = -\sum_{j}^{N} p_j^T \log(q_j^T)
\]

(12)

\[
p_j^T = \frac{\exp(v_j / T)}{\sum_{k}^{N} \exp(v_k / T)}
\]

(13)

\[
q_j^T = \frac{\exp(z_j / T)}{\sum_{k}^{N} \exp(z_k / T)}
\]

(14)

\[
L_{\text{hard}} = -\sum_{j}^{N} c_j \log(q_j^T)
\]

(15)

\[
q_j^T = \frac{\exp(z_j)}{\sum_{j}^{N} \exp(z_j)}
\]

(16)
\( v_i \): Teacher models of logits.

\( z_i \): Student models of logits.

\( p_i^T \): The softmax output of the teacher model at temperature = T on class i.

\( q_i^T \): The softmax output of the student model at temperature = T for the value of class i.

\( c_i \): The value of the ground truth on the i-th class of, \( i \in \{0,1\} \). Positive labels are taken as 1 and negative labels are taken as 0.

\( N \): Total number of labels.

\( \alpha, \beta \): The weights of the corresponding loss functions.

The total loss is designed as a weighted average of the cross-entropy corresponding to the soft and hard targets (where a larger weighting factor \( \alpha \) for the soft target cross-entropy indicates that the more knowledge migration relies on the contribution of the teacher network, which is necessary for the early stage of training because the output of the teacher network is smoother and contains more information than the truth-value labels, such as dark knowledge in addition to the maximum probability value corresponding to the classification [15]: the probability of a certain category is the smallest, etc., which helps the student network to identify simple samples more easily, but the weighting factor \( \beta \) of the hard target needs to be increased appropriately in the later stage of training, so that the truth-value labels can help to identify difficult samples, which refer to some easily confused pictures. In addition, the inference performance of the teacher network is usually better than that of the student network, while there is no specific limitation on the model capacity, and usually the more parameters the teacher network has, the better the fitting performance is, and the more beneficial to the student network learning[16].

4. Adaptive Width

The overall training of DynaDEIT is divided into two stages, first training the width-adapted DynaDEITw and then the width+depth-adaptive training DynaDEIT. the width-adaptive training process starts by sorting the neurons in the heads and MLPs of each layer of the MSA according to Network Rewiring [17], and once the sorting is completed, we obtain An initial and fixed teacher network is obtained [18]. This is done by calculating a change in the relationship between the loss before the head or neuron is removed and the loss without them, the larger the change, the more important the head or neuron is. Before pruning, a is the number of attention heads and b is the number of neurons in the MLP layer. After pruning the head and neurons separately, the number of heads becomes \( w_a \) and the number of neurons becomes \( w_b \). After pruning, the multiple student models are called DynaDeitw, so all the DynaDeitw models are trained using the distillation algorithm, where the distillation The total loss function used here consists of three aspects, which are (1) the input vector after image slicing and linear transformation, (2) the output vector of the last layer of Transformer Encoder, and (3) the output vector of class token and distillation token together after the MLP classifier. Finally, all the patches are used to train all the DynaDeitw, which will be used as the teacher model in the next step.

The final loss equation.

\[
\mathcal{L} = \lambda_1 \ell_{\text{pred}}(y^{(m_y)}, y) + \lambda_2 (\ell_{\text{emd}}(E^{(m_y)}, E) + \ell_{\text{hidn}}(H^{(m_y)}, H)),
\]

\[
\ell_{\text{pred}}(y^{(m_y)}, y) = -\text{softmax}(y) \cdot \log \text{softmax}(y^{(m_y)})
\]

\[
\ell_{\text{emd}}(E^{(m_y)}, E) = \text{MSE}(E^{(m_y)}, E)
\]

\[
\ell_{\text{hidn}}(H^{(m_y)}, H) = \sum_{j=1}^{L} \text{MSE}(H^{(m_y)}_j, H_j)
\]
\( \lambda_1 \) and \( \lambda_2 \) represents the weights of the corresponding loss functions.

\[ \ell_{\text{pred}} ; \ell_{\text{emb}} ; \ell_{\text{hid}} \] all three are loss functions.

\( y^{(m_\text{w})} \) is the output of the last MLP classification header of the width cropped Deit.

\( E^{(m_\text{w})} \) is the output of the embedding layer of the width-cropped Deit.

\( H^{(m_\text{w})} \) is the output of the width-cropped Deit after all transformer encoder layers.

\( y \) is the output of the last MLP classification header of the initialized teacher model Deit.

\( E \) is the output of the initialized teacher model Deit that goes through the embedding layer.

\( H \) is the output of the initialized teacher model Deit that goes through all transformer encoder layers.

5. Adaptive Depth

\[ \text{mod}(d + 1, 1/m_d) = 0 \]  \hspace{1cm} (21)

Using the trained DynaDeitw as the teacher model, distill the smaller DynaDeit after cropping the depth. Let the depth factor be \( m_d \). Let the depth of the layer be \([1, D]\). According to equation (21) remove the number of layers with depth \( d \). The \( d + 1 \) layer is taken because the last layer should be the most important. For Deit, containing \( D \) transformer layers \( 1, 2, ..., D \). Assume that \( D = 12, \ m_d = 0.75 \). The layers that are dropped are those with depths of 3, 7, and 11. The basis for this is still Network Rewiring. Meanwhile, each sub-network that performs Adaptive Depth will again perform Adaptive Width. distillation training is performed once the compression is completed [19].

Overall loss:

\[ \mathcal{L} = \lambda_1 \ell_{\text{pred}} (y^{(m_\text{w}, m_\text{d})}, y^{(m_\text{w})}) + \lambda_2 \ell_{\text{emb}} (E^{(m_\text{w}, m_\text{d})}, E^{(m_\text{w})}) + \ell_{\text{hid}} (H^{(m_\text{w}, m_\text{d})}, H^{(m_\text{w})})) \]  \hspace{1cm} (22)

\( y^{(m_\text{w}, m_\text{d})} \) is the output of the last MLP classification header of the width- and depth-cropped Deit.

\( y^{(m_\text{w})} \) is the output of the last MLP classification header of the width cropped Deit.

\( \lambda_1 ; \lambda_2 \) represents the weights of the corresponding loss functions.

\( E^{(m_\text{w}, m_\text{d})} \) is the output of the embedding layer of the width- and depth-cropped Deit.

\( E^{(m_\text{w})} \) is the output of the embedding layer of the width cropped Deit.

\( H^{(m_\text{w}, m_\text{d})} \) is the output of Deit that has been cropped by width and depth through all transformer encoder layers.

\( H^{(m_\text{w})} \) is the output of all transformer encoder layers of the width- and depth-cropped Deit.

\[ \ell_{\text{pred}} ; \ell_{\text{emb}} ; \ell_{\text{hid}} \] are loss functions.

6. Experiments

This section presents some experiments and results. We use knowledge distillation to enable the Transformer-based model to learn some inductive bias of the CNN-based model, which improves the processing of image-type data. We also use width adaptive and depth adaptive training. During training, the width and the depth are cropped to train different sub-networks, and during inference, they are directly cropped according to the speed needs and the cropped sub-networks are used for prediction. The overall training is divided into two phases, with width adaptive training followed by width and depth adaptive training. We then compared and analyzed the efficiency and accuracy of the convolutional neural network and Deit. The dataset was only cifar100 and did not include any convolution. We built the environment on Pytorch and used three RTX3060Ti graphics cards for the experiments.
Table 1. Indicates the final results of DeiT on Cifar100, including acc and loss of Deit.

| Model | Top accuracy/% | Total training time/h |
|-------|----------------|-----------------------|
| Cnn   | 89.9           | 30                    |
| Vit   | 86.1           | 32                    |
| Deit  | 90.7           | 34                    |
| Deit with width and depth adaptive training | 90.5 | 24 |

According to our experimental setup, what we can easily find is that despite less data and less computational resources, DeiT still generates higher performance image classification models, and the overall results are still good.

DeiT has better results than CNN probably because DeiT is based on a modified version of Vit, which has a Transformer architecture. DeiT can focus on those relatively important regions instead of treating all pixels equally as CNN does, regardless of the content of the pixel and whether it is important or not. Also, convolution is difficult to relate spatially distant concepts, and each convolution filter is restricted to work in a small region, but long-term interaction between semantic concepts is crucial. CNN performance is closely related to the size of the dataset. And DeiT also uses knowledge distillation, which makes full use of the dark knowledge of the teacher model, and the loss function is more reasonable.

DeiT has better results than Vit because the overall structure of DeiT is similar to Vit, but DeiT is based on knowledge distillation to learn image transformer, and the processing ability of image type data is really improved, so the operation related to knowledge distillation is worth learning and learning from.

DeiT with width and depth adaptive training compared to DeiT, the result is decreased but the difference is not big, and we use less time for training and lower computer hardware cost. So the application of width and depth adaptive training is necessary. Because it needs to crop the student model, resulting in fewer neurons, the model fitting ability is not as strong.
7. Conclusions
This paper presents a dynamic prediction-based visual coder acceleration technique for image classification on a knowledge distillation-based technique. This paper combines DeiT and width and depth adaptive training. Because pre-training Vit needs a huge dataset and a multi-GPU environment. To reduce the computational cost, we use DeiT model and to further compress DeiT we use width and depth adaptive training and get better results on the test set of Cifar100 and can greatly reduce the pre-training time and computational cost. After completing the experiments, it is expected that the pre-training time of DeiT is faster compared to traditional CNN and Vit because we added width and depth adaptive training techniques. Unexpectedly, the results of DeiT are slightly better than CNN because we use a smaller dataset and we apply knowledge distillation techniques on DeiT.

References
[1] Vaswani A, Shazeer N, Parmar N., et al. Attention Is All You Need [J]. arXiv, 2017.
[2] Dosovitskiy A, Beyer L, Kolesnikov A., et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale [J]. 2020.
[3] Touvron H, Cord M, Douze M., et al. Training data-efficient image transformers & distillation through attention [J]. 2020.
[4] Lan Z, Chen M, Goodman S, et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations [J]. 2019.
[5] Srinivas S, Babu R V. Data-free parameter pruning for Deep Neural Networks [J]. Computer Science, 2015:2830-2838.
[6] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network [J]. Computer Science, 2015, 14(7):38-39.
[7] Jou J, Yu B, Maybank S J, et al. Knowledge Distillation: A Survey [J]. 2020.
[8] Hou L, Shang L, X Jiang, et al. DynaBERT: Dynamic BERT with Adaptive Width and Depth [J]. arXiv, 2020.
[9] Tolstikhin I, Houlsby N, Kolesnikov A., et al. MLP-Mixer: An all-MLP Architecture for Vision. 2021.
[10] Hendrycks D, Gimpel K. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units [J]. 2016.
[11] Srivastava N, Hinton G, Krizhevsky A., et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting [J]. Journal of Machine Learning Research, 2014, 15(1):1929-1958.
[12] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition [J]. IEEE, 2016.
[13] Ba J L, Kiros J R, Hinton G E. Layer Normalization [J]. 2016.
[14] Zaremba W, Sutskever I, Vinyals O. Recurrent Neural Network Regularization [J]. Eprint Arxiv, 2014.
[15] Mishra A, Marr D. Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy. 2017.
[16] Alkhulaifi A, Alsahli F, Ahmad I. Knowledge Distillation in Deep Learning and its Applications [J]. 2020.
[17] Pizzati F, García. Enhanced free space detection in multiple lanes based on single CNN with scene identification [J]. IEEE, 2019.
[18] Liu W, Zhou P, Zhao Z., et al. FastBERT: a Self-distilling BERT with Adaptive Inference Time. 2020.
[19] Fukuda T, Suzuki M, Kurata G, et al. Efficient Knowledge Distillation from an Ensemble of Teachers [C]. Interspeech 2017. 2017.