DiT: Efficient Vision Transformers with Dynamic Token Routing

Yuchen Ma, Zhengcong Fei, Junshi Huang
Meituan
{mayuchen, feizhengcong, huangjunshi}@meituan.com

Abstract

Recently, the tokens of images share the same static data flow in many dense networks. However, challenges arise from the variance among the objects in images, such as large variations in the spatial scale and difficulties of recognition for visual entities. In this paper, we propose a data-dependent token routing strategy to elaborate the routing paths of image tokens for Dynamic Vision Transformer, dubbed DiT. The proposed framework generates a data-dependent path per token, adapting to the object scales and visual discrimination of tokens. In feed-forward, the differentiable routing gates are designed to select the scaling paths and feature transformation paths for image tokens, leading to multi-path feature propagation. In this way, the impact of object scales and visual discrimination of image representation can be carefully tuned. Moreover, the computational cost can be further reduced by giving budget constraints to the routing gate and early-stopping of feature extraction. In experiments, our DiT achieves superior performance and favorable complexity/accuracy trade-offs than many SoTA methods on ImageNet classification, object detection, instance segmentation, and semantic segmentation. Particularly, the DiT-B5 obtains 84.8% top-1 Acc on ImageNet with 10.3 GFLOPs, which is 1.0% higher than that of the SoTA method with similar computational complexity. These extensive results demonstrate that DiT can serve as versatile backbones for various vision tasks. The code is available at https://github.com/Maycbj/DiT.

1 Introduction

Over the past few years, transformer-based networks have achieved promising results in various computer vision tasks, including image classification [1; 2], object detection [3; 4], and semantic segmentation [5]. However, some problems in those networks are still unresolved, one of which concentrates on the variance of object representations mainly stemming from two aspects: (1) The variance of object scales in the image. An example is presented in Fig. 1(a), where objects of different scales indicate the diverse number of tokens with different appearances of details. (2) The variance of feature discrimination for coarse-grained and fine-grained object recognition. For example, the recognition of "human" object may require higher-level semantic features than "football" object and background.

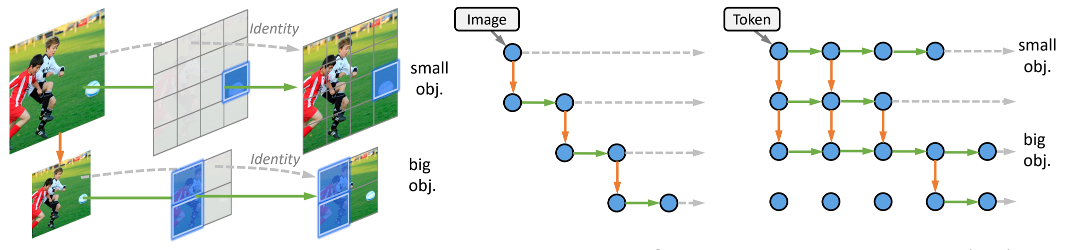
imagine you're looking at a big picture with both an elephant and an ant in it. Now, these networks we're talking about have a tough time figuring out things in pictures, especially when objects are of different sizes, like our huge elephant and tiny ant.

So, two problems they face are:

1. Object Size: Some objects in pictures are big, and some are tiny. Just like in our picture, the elephant takes up a lot more space than the ant. This means the computer sees more "pieces" or "tokens" of the elephant than the ant. These pieces have different details based on the size of the object.
2. Detail Needed: Think of it like this - recognizing a person in a picture might need more information than recognizing a football. It's like trying to tell apart two nearly identical Lego sets. One might need you to look more closely at tiny details, while the other is easier to identify with just a quick glance.

The large variance in token appearance and feature discrimination brings difficulties to feature representation learning, traditional methods try to solve this problem by well-designed network architectures. Conventional CNNs [6; 7], as well as recently developed transformer-based networks [8; 2; 9] usually follow a sequential methodology in their architecture design, which involves a gradual reduction in the spatial size of feature maps. This paradigm leads to a progressive shrinking pyramid to highlight the representation of large and discriminative objects while neglecting the unimportant regions. For dense prediction tasks like object detection and instance segmentation, multiple-scale features are indispensable for handling objects of varying scales. The canonical approach for multi-scale features involves FPN from RetinaNet [10] and Mask R-CNN [11], where the assignment of

Elephant vs ant



(a) Token routing of dynamic scale and depth (b) Architectures of PVT, Swin, etc (c) Architectures of DiT (ours)

Figure 1: In all figures, dotted grey lines represent the identical transformation of token features, solid green lines represent the feature extraction by transform block, and solid orange lines represent the down-scaling of tokens. **(a) Illustration of our main idea.** Given input image with different spatial scales, the proposed *dynamic token routing* adaptively selects the ***token-specific*** forward paths, including dynamic-scaling and dynamic-depth paths. In this way, scale-variant objects (*e.g.* football and children) may be activated on feature maps with appropriate resolutions. Meanwhile, the fine-grained tokens (*e.g.* children) require more elaborated procedures of feature extraction. **(b)(c)** present the difference between existing methods and our proposed DiT, where the node represents the image/token feature. Please note that the dynamic path in DiT is selected in a token-specific manner, comparing with image-specific configuration in PVT, Swin, and stochastic depth networks.

imagine you're trying to find and label things in a big picture. Now, in this picture, some things are really big like trees and some are really tiny like ants.

1. Why multiple-scale features are needed: It's like using different magnifying glasses for different items in the picture. A big magnifying glass helps you see small things (like ants) clearly, while a smaller magnifying glass might be best for the bigger things (like trees).
2. Traditional methods: Think of it as using a set of fixed magnifying glasses (each one has its own zoom level). Some methods, like RetinaNet and Mask R-CNN, already do this. They decide which magnifying glass to use based on the size of the thing you're trying to find in the picture.
3. The new idea (dynamic-scaling networks): Instead of always using the same set of magnifying glasses for every picture, what if you could pick and choose the best ones for each picture? That's the idea here. For each part of an image, the network dynamically selects the best "magnifying glass" or scale to use.

ground-truth labels in feature pyramid greatly depends on the scales of objects (*e.g.* the prediction of small objects occurs at the high resolution of feature map). However, those handcrafted architectures fully utilize the multi-scale features in a static way, we propose to select some appropriate scaling paths for image tokens as dynamic-scaling networks.

What everyone did?

What we propose?

For the comprehensive learning of image representation, many recent works [7; 12] directly use some deeper and more complicated backbones [13; 9; 14]. Particularly, some works [15; 16] propose dynamic token adaptation strategies to discard or merge redundant tokens in classification tasks, leading to promising accuracy and Flops trade-off. However, these pruning-like strategies are hardly applied to the dense prediction tasks, due to the scarcity of pruned image tokens and unstructured pruned output. To eliminate this problem, we propose to identically transform image tokens in some blocks via skip-path, rather than token pruning. Consequently, the discrimination of token features from variant objects or background can be comprehensively learned, where the network tends to skip the feature extraction of simple objects or background, and requires more procedures of feature extraction for complex objects. With the learnable routing strategy, **the tokens in single image** conduct different feature transformation paths, describing the main difference between our dynamic-depth network to stochastic depth networks [17].

Imagine you're making a robot to sort LEGO bricks by size and color. The robot has a special camera (called a "vision transformer") that looks at the LEGO pieces, but sometimes it gets confused when the LEGO pieces are very different in size or color.

1. What the paper is doing: They're trying to make this robot's camera smarter. They're adding two special features to help it sort better: one to adjust for size (dynamic-scaling) and one to adjust for color complexity (dynamic-depth).
2. Dynamic-Vision-Transformer (DiT): This is the fancy new name for their smarter robot camera. It can adjust how it looks at the LEGO bricks on-the-fly.
3. The grid-like network: Think of this like a big game board. Each square on the board helps decide how the robot camera should adjust its view. Each square is a decision-making spot, which they call a "token routing module."
4. Token routing module: This is the brainy part of each square on the game board. It decides how to adjust the view for each LEGO piece it's looking at. It predicts whether it should zoom in, zoom out, or change the focus based on the LEGO's size and color.
5. Two binary gates: This is like having two switches at each decision-making square. One switch (vertical path) decides if the robot should zoom in or out (for size), and the other switch (horizontal path) decides how to adjust for the LEGO color complexity.
6. End goal: By doing all this fancy adjusting, the robot camera (or vision transformer) gets better at recognizing and sorting different LEGO pieces by their size and color.

In this paper, we integrate a dynamic-scaling network and dynamic-depth network into vision transformer, composing the main structure of Dynamic-Vision-Transformer, dubbed DiT. As presented in Fig. 1(c), we construct the grid-like network structure, and each node represents the *token routing module*. The token routing module determines the calculated path of each token and predicts the probability distribution of whether to be calculated by the transformer module. We hope to explore the unstructured and data-dependent token routing strategy for vision transformers. In the routing module, two binary gates are learned for the path selection of each token, where the vertical and horizontal paths correspond to the **dynamic-scaling** and **dynamic-depth** path, respectively. In this manner, response maps from proper sizes and depths of layers are dynamically activated for variant scales and the complexity of token paths. To balance the trade-off between performance and Flops, the constraints of computational budgets (*e.g.* FLOPS) are considered in our learning task. As our designed differentiable routing gates and computational constraints, the DiT could be learned in end-to-end manner with dynamic path selection per token on the fly. The overall approach can be easily integrated into most transformer-based vision backbones.

To elaborate on its superiority in both performance and efficiency, we follow the standard training recipe as in DeiT [18], and consistently achieve superior performance compared to recent SOTA Vision Transformers [1; 2; 8; 19]. Furthermore, We integrate DiT as the backbone into dense prediction tasks, *e.g.* object detection and instance segmentation task, on the COCO dataset and semantic segmentation on the ADE20K dataset, achieving state-of-the-art performance on all benchmarks.

2 Related Works

Transformer backbones. ViT [1] is the pioneer work to demonstrate that a pure Transformer can achieve state-of-the-art performance in image classification. ViT employs a pure Transformer model for image classification by treating an image as a sequence of patches. DeiT [18] further explores a data-efficient training strategy and a distillation method for ViT. After that, more recent methods such as CPVT [20], TNT [21], and CrossViT [9] further improve the performance of ViT by modifying the network structure.

Beyond classification, PVT [8] is an important work to introduce a pyramid structure in Transformer, which proves the potential of a pure Transformer backbone compared to CNN counterparts in dense prediction tasks. Subsequently, methods such as Swin [2], CvT [22], CoaT [23], and LeViT [24] enhance the local continuity of features and show the flexibility to model at various scales with linear computational complexity with respect to image size.

Multi-scale feature for Dense prediction. For dense prediction tasks, One of the problems in dense prediction comes from the huge scale variance among inputs, e.g., the tiny object instances and the picture-filled background stuff. Therefore, most existing works combine an image classification network and a feature fusion module to extract and fuse multi-scale features. FPN [25] uses a top-down pathway to sequentially combine multi-level features. Subsequently, the label assignment of the multi-scale feature pyramid in RetinaNet [10] and Mask R-CNN [11] depends on the scale of the objects. PANet [26] further introduces a bottom-up pathway to shorten the pathway between features. DeeplabV3+ [27] fuses multi-scale features obtained by atrous spatial pyramid pooling. MPViT [19] exploits tokens of different scales independently fed into the Transformer encoders via multiple paths and the resulting features are aggregated. Overall, those handcrafted architectures aim at utilizing multi-scale features in a static network, rather than adapting to input dynamically.

Dynamic Networks. Dynamic networks, adaptively adjusting the network architecture to the corresponding input, have been recently studied in the current network design. For the traditional CNN backbones, previous methods mainly focus on image classification by dropping blocks [28; 29] or pruning channels [30; 31] for efficient inference. [32] propose a framework generating data-dependent routes, adapting to the scale distribution of each image on the CNN architecture. Recently, many works explore the effects of dynamic networks in transformer-based vision backbones. DynamicVit [15], a dynamic token sparsification framework to prune redundant tokens progressively and dynamically based on the input, which gradually drops the uninformative tokens as the computation proceeds on the image classification task. [16] introduce a differentiable parameter-free Adaptive Token Sampling module, which can be plugged into any existing vision transformer architecture. However, all these works focus on transformer-based token pruning, and few works explore the possibility of making use of the characteristic of token routing to accelerate the dense prediction tasks. To utilize the dynamic property, an end-to-end dynamic routing framework is proposed in this paper to learn multi-scale token routing and improve the performance of the downstream tasks.

3 Dynamic Token Routing

Compared with previous static network architectures, *dynamic token routing* has the superiority in better performance with the budgeted resource consumption. In this section, we first introduce the architecture of the DiT. Secondly, the details of the routing gate module will be clarified. We illustrate the constrained budgets and training details at the end of this section.

3.1 Overview Architecture

Our main idea is to integrate the dynamic token routing strategy into the transformer-based network to dynamically select the token-specific forward path through multi-scale feature maps and various network depths. Fig. 2 presents the overview of DiT, where we follow the similar principle of network design as PVT [1]. Specifically, we construct our network as a grid-like structure, and each node in the grid represents the response map used as the input of the following layers. For example, the response map at (i, j) stage of the grid is denoted as $F_{i,j}$, which is then used as input of patch embedding layer $\mathcal{P}_{i,j}$, transformer blocks $\mathcal{B}_{i,j}$, and identity mapping layer. Note that the token routing strategy is also dependent on this feature map, more details will be introduced in Sec. 3.2.

1. Grid-Like Structure: Imagine you have a chessboard, and on every square, there's a tiny lab station (node). Each lab station has a unique response map—sort of like a mini blueprint or a reference guide.
2. Response Map (Fi,j): This is a representation at a specific square (i,j) on your chessboard. You use this map as a starting point to decide what steps to take next.
3. Tools at each Square: At every square (or stage), you have three main tools:
• Patch Embedding Layer (Pi,j): Think of this as a microscope that looks closely at certain parts of your response map.
• Transformer Blocks (Bi,j): These are like tiny robots that process and modify the parts of the response map you're focusing on.
• Identity Mapping Layer: This is like a reference guide, ensuring you remember and use the original information from your response map.
4. Using the Tools: When working on a square, you combine or mix information from the current square and also possibly from the neighboring squares. You might look left (i,j-1) or up (i-1,j) on the chessboard to borrow information from there. The process of deciding where to get information from and how much to mix is based on a strategy called 'routing'.
5. Skipping Some Steps: Now, sometimes, you don't need to use all the tools. Depending on your strategy, you might skip some tools on certain squares. When you skip, you put a mask on the response map—sort of like putting a 'do not touch' sign on certain parts.

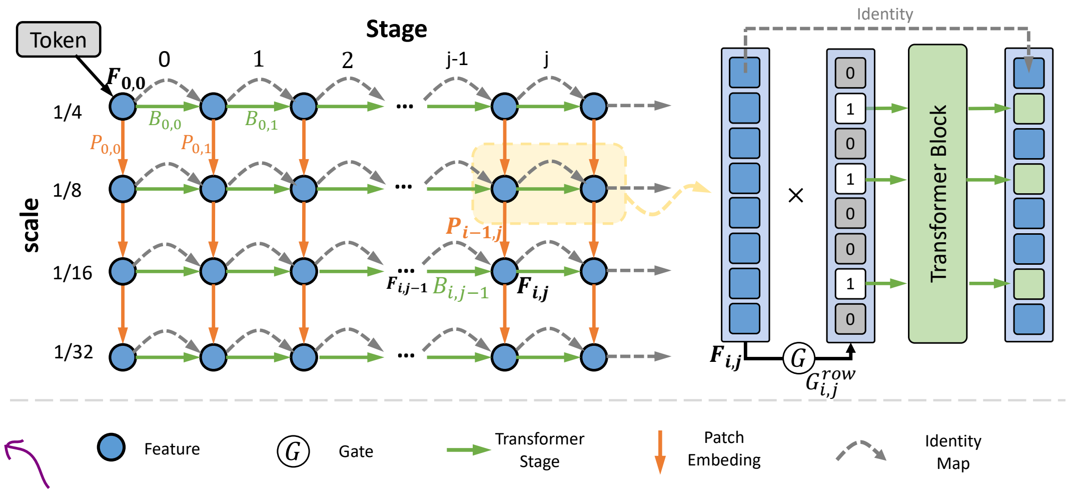


Figure 2: The proposed *dynamic token routing* framework of DiT. Left: The routing space with layer L and max downsampling rate 1/32. Lines in green, orange, and gray are alternative paths for dynamic routing, which represent Transformer Block, Patch Embedding Block, and identity mapping layer, respectively. Given the feature map from the former layer, we first generate a mask using the *routing gate*. The discrete value of the mask indicates whether the token is involved in the calculation of this layer. More details about the network are elaborated in Sec. 3.2 Best viewed in color.

Basically, the token features of response map $F_{i,j}$ are calculated by element-wise sum with routing strategy on 3 types of inputs at most, *i.e.*, the output of transformer blocks $\mathcal{B}_{i,j-1}$, the output of patch embedding layer $\mathcal{P}_{i-1,j}$, and the identity mapping of $F_{i,j-1}$. Note that the calculations of some token features in the transformer blocks and patch embedding layer may be skipped due to routing strategy, we simply mask the corresponding features of response maps in such cases.

In our network, the input image with dimensions $H \times W \times 3$ is divided into $HW/4^2$ patches in the first stage, which reduces the resolution to 1/4 of the raw size. After that, the space composed of patch embedding layers, transformer stages, and identity mapping layers are designed for dynamic routing, called *token routing space*. Generally, the entire space has N levels of patch embedding layers, each of which downsamples the scale of input maps into 1/2. Usually, we set N as 4 in our experiments, leading to 1/32 downsampling of the original image as presented in Fig. 2. In the same level of patch embedding, the number of transformer stages is set as M. For a fair comparison, the stage number M and the number of transformer blocks in each stage are similar to that of PVT [33].

Imagine you're analyzing a photograph using a microscope.
1. Dividing the Image: Initially, you take the image and divide it into smaller sections (think of these as zoomed-in areas under your microscope). Each section captures part of the image.
2. Resolution Reduction: Instead of viewing the whole photograph at once, you reduce the clarity by a factor. The "H×W×3" simply means the image's height, width, and the 3 color channels. "HW/4^2 patches" means you are segmenting the photo into smaller chunks, and after this step, each segment is 1/4 its original size.
3. Token Routing Space: Think of this as a multi-level laboratory where each level allows you to further analyze and process these segments. At each level, you can reduce the segment size further, making it smaller and simpler to analyze.
4. Progressive Downsampling: As you go deeper into these levels, the segments are downsized. By the time you reach the 4th level, your segments are tiny, being only 1/32 of their original size.
5. Transformers & Stages: Within each level of this lab, you use various tools and techniques (referred to as 'transformers') to process and understand these segments. The term "stage" here is like a phase or step in the process, and "M" denotes how many of these phases you have in a level.

3.2 Dynamic Token Routing

An important characteristic of our DiT is that the token routing is performed in a self-dependent manner, *i.e.*, the token features in response map $F_{i,j}$ are adopted to control the routing paths of themselves. Generally, both the transformer stage and patch embedding module have a binary gate to select the token routing paths, respectively.

For the routing gate in each transformer stage, a binary decision mask $G \in \{0,1\}^l$ is used to decide whether to calculate the token features via transformer blocks, where $l = h \times w$ is the number of tokens at the corresponding stage. Initially, we set all elements of the decision mask as 1. To learn the data-dependent mask at stage (i,j) of grid-like network , we use a linear projection $w^{row}_{i,j} \in \mathbb{R}^{d\times 2}$ to predict the probability map $P^{row}_{i,j}$ according to the feature map $F_{i,j} \in \mathbb{R}^{l\times d}$, where d is the feature dimension:

$$P^{row}_{i,j} = Softmax(F_{i,j} * w^{row}_{i,j}) \in \mathbb{R}^{l\times 2}. \tag{1}$$

As our target is to perform token-specific routing, we need to discrete the probability map of tokens. Therefore, the Gumbel-Softmax [34] is used to sample the binary decision mask from the probability map $P^{row}_{i,j}$ as follows:

$$G^{row}_{i,j} = Gumbel\text{-}Softmax(P^{row}_{i,j}) \in \{0,1\}^l \tag{2}$$

1. Self-Dependent Token Routing:
Imagine you have a bunch of little robots, called tokens, traveling on paths on a map (your chessboard network). Each robot looks at its own features (sort of like its own specifications or tools it's carrying) and decides which path it should take.

2. Decision Gates:
Now, there are special checkpoints (binary gates) in the network that help these robots choose their path. These checkpoints are present at every transformer stage and patch embedding module.

3. Making the Decision with a Mask:
At each checkpoint (transformer stage), there's a decision mask, which is kind of like a traffic light system. It can either show a green light (1) or a red light (0) for each robot, indicating whether it should proceed or stop.

4. Default Setting:
At the beginning, every traffic light is set to green (meaning every element in the decision mask is 1), allowing all robots to move freely.

5. Adapting to the Data:
However, we don't always want all robots to go in the same direction or use the same path. So, we use a specific mechanism (a linear projection) that looks at the robot's features and the entire map's scenario to decide which traffic lights should change from green to red or vice versa.

In order to obtain a more stable gating module in inference, we maintain a moving averaged version of linear projection weight $\hat{w}^{row}_{i,j}$ by momentum update:

$$\hat{w}^{row}_{i,j} \leftarrow m * \hat{w}^{row}_{i,j} + (1-m) * w^{row}_{i,j}, \tag{3}$$

where $m \in [0,1)$ is a momentum coefficient, and $w^{row}_{i,j}$ are updated by back-propagation.

The routing gate in patch embedding layer decides whether to downsample the response map or not. Similar to the routing gate in the transformer stage, we generate a binary scalar $G^{col}_{i,j}$ for the routing decision of the entire feature map:

$$\begin{aligned} P^{col}_{i,j} &= Softmax(\bar{F}_{i,j} * w^{col}_{i,j}) \in \mathbb{R}^{1\times 2} \\ G^{col}_{i,j} &= Gumbel\text{-}Softmax(P^{col}_{i,j}) \in \{0,1\} \end{aligned}$$

where $\bar{F}_{i,j} \in \mathbb{R}^d$ is obtained by average pooling on the response map $F_{i,j}$. Similarly, the moving averaged weight $\hat{w}^{col}_{i,j}$ is maintained in the training stage for stable inference.

With the proposed gating module as aforementioned, the feature map $F_{i,j}$ can be formulated as

$$F_{i,j} = \underbrace{\mathcal{B}_{i,j-1}(G^{row}_{i,j-1} \odot F_{i,j-1}) + (1-G^{row}_{i,j}) \odot F_{i,j-1}}_{row} + \underbrace{G^{col}_{i-1,j} \odot \mathcal{P}_{i-1,j}(F_{i-1,j})}_{col} \in \mathbb{R}^{l\times d}, \tag{4}$$

where $\mathcal{B}_{i,j-1}$ and $\mathcal{P}_{i-1,j}$ are the transformer blocks and patch embedding layer, respectively. It should be noted that we skip the calculation of token features in transformer blocks $\mathcal{B}_{i,j-1}$ if the corresponding elements of mask map $G^{row}_{i,j-1}$ are 0 in our implementation.

3.3 Complexity Constraint

To balance the trade-off between efficiency and effectiveness, we add the complexity constraint for dynamic token routing. The gate modules are inclined to choose a more efficient routing path instead of calculating some uninformative tokens. We denote constant value C as the computational costs associated with the predefined network complexity, *e.g.*, FLOPs. We formulate the cost of each stage as follows.

$$C_{i,j} = \bar{G}^{row}_{i,j} \cdot C^{row}_{i,j} + G^{col}_{i,j} \cdot C^{col}_{i,j} + C^{gates}_{i,j} \tag{5}$$

where $C^{row}_{i,j}$, $C^{col}_{i,j}$, and $C^{gates}_{i,j}$ are the FLOPs of modules $\mathcal{B}_{i,j}$, $\mathcal{P}_{i,j}$, and routing gates, respectively. $\bar{G}^{row}_{i,j}$ is the average of binary decision map. Generally, the complexity constraint of each stage contains the cost of transformer blocks, patch embeddings, and routing gates. $\mathcal{C}^{space}$ and $\mathcal{C}^{base}$ denote the cost of our routing space and the cost of the baseline(PVTv2). Finally, the complexity constraint of the overall network is designed as:

$$\begin{aligned} \mathcal{L}_C &= (C^{space}/C^{base} - \mu)^2 \\ &= (\sum_{i,j\in\mathcal{S}} C_{i,j})/C^{base} - \mu)^2 \end{aligned} \tag{6}$$

where μ is the ratio of limiting the computational complexity for the network. With different μ, the selected routes in each propagation would be adaptively restricted to corresponding calculated budgets. Overall, the network weights, as well as the routing gates, can be optimized with a joint loss function $\mathcal{L}$ in a unified framework. $\mathcal{L}_N$ and $\mathcal{L}_C$ denote the loss function of the whole network and complexity constraint:

$$\mathcal{L} = \mathcal{L}_N + \lambda_C \mathcal{L}_C \tag{7}$$

where λ_C is utilized to balance the optimization of network prediction and complexity cost.

4 Experiment

In this section, we illustrate the effectiveness and the efficiency of DiT as a vision backbone on image classification (ImageNet-1K), dense predictions such as object detection and instance segmentation (COCO), and semantic segmentation (ADE20K).

Table 1: Image classification performance on the ImageNet validation set. "#Param" denotes the quantity of valid parameters utilized.

Model	#Params.	GFLOPs	Top-1(%)
ResNet18 [7]	11.7	1.8	69.8
DeiT-Tiny/16 [18]	5.7	1.3	72.2
PVTv1-Tiny [8]	13.2	1.9	75.1
PVTv2-B1 [33]	13.1	2.1	78.7
DiT-B1(ours)	30.3	2.0	**79.9**
DeiT-Small [18]	22.1	4.6	79.9
TNT-S [21]	23.8	5.2	81.3
Swin-T [2]	29.0	4.5	81.3
CvT-13 [22]	20.0	4.5	81.6
PVTv2-B2 [33]	25.4	4.0	82.0
DiT-B2(ours)	55.9	3.8	**83.1**
ResNet101 [7]	44.7	7.9	77.4
CvT-21 [22]	32.0	7.1	82.5
PVTv2-B3 [33]	45.2	6.9	83.2
DiT-B3(ours)	86.3	6.8	**84.2**
PVTv1-Large [8]	61.4	9.8	81.7
TNT-B [21]	66.0	14.1	82.8
Swin-S [2]	50.0	8.7	83.0
PVTv2-B4 [33]	62.6	10.1	83.6
DiT-B4(ours)	126.0	9.9	**84.5**
DeiT-Base/16 [18]	86.6	17.6	81.8
Swin-B [2]	88.0	15.4	83.3
PVTv2-B5 [33]	82.0	11.8	83.8
DiT-B5(ours)	158.0	10.3	**84.8**

Table 2: Comparing with ResNet, Swin Transformer, PVTv2, Focal [41] across different object detection methods, "AP^b" signifies the average precision of bounding box AP. The calculation of "GFLOPs" is based on the scale of 1280 $\times$ 800.

Backbone	Method	GLOPs	AP^b	AP^b_{50}	AP^b_{75}
ResNet50[7]	Cascade Mask R-CNN [42]	739	46.3	64.3	50.5
ConvNeXt-S[43]		827	51.9	70.8	56.5
Swin-T[2]		745	50.5	69.3	54.9
PVTv2-B2[33]		788	51.1	69.8	55.3
DiT-B2(ours)		745	**52.3**	**71.0**	**56.9**
ResNet50[7]	ATSS [44]	205	43.5	61.9	47.0
Swin-T[2]		215	47.2	66.5	51.3
Focal-T[41]		239	49.5	68.8	53.9
PVTv2-B2[33]		258	49.9	69.1	54.1
DiT-B2(ours)		247	**51.2**	**70.5**	**55.7**
ResNet50[7]	GLF [45]	208	44.5	63.0	48.3
Swin-T[2]		215	47.6	66.8	51.7
PVTv2-B2-Li[33]		197	49.2	68.2	53.7
PVTv2-B2[33]		261	50.2	69.4	54.7
DiT-B2(ours)		235	**51.3**	**70.6**	**55.9**
ResNet50[7]	Sparse R-CNN [42]	166	44.5	63.4	48.2
Swin-T[2]		172	47.9	67.3	52.3
Focal-T[41]		196	49.0	69.1	53.2
PVTv2-B2[33]		215	50.1	69.5	54.9
DiT-B2(ours)		202	**51.3**	**70.5**	**56.0**

4.1 Image classification on ImageNet-1K

The experimental settings involve performing image classification experiments on the ImageNet 2012 dataset [35], which contains 1.28 million training images and 50K validation images distributed across 1,000 categories. To make a fair comparison, all models are trained on the training set and evaluated with the top-1 error on the validation set. We strictly follow the setting of DeiT [18], which employs random cropping, random horizontal flipping, label-smoothing regularization [36], mixup [37], CutMix [38] as data augmentations during training. Following the network structure and hyperparameters of different sizes of the PVTv2 range from B1 to B5, the size of the DiT is denoted as DiT-B1 to DiT-B5.

We utilize AdamW [39] with a momentum of 0.9, a mini-batch size of 128, and a weight decay of 5×10^{-2} to optimize models. The initial learning rate is set to 1×10^{-3} and decreased following the cosine schedule [40]. All models are trained from scratch for 300 epochs on 8 V100 GPUs. For benchmark testing, we use a center crop of the validation set, where a 224 $\times$ 224 patch is cropped to evaluate the classification accuracy.

In Tab. 1, we summarize the results for baseline models and the current SoTA models on the image classification task. We can find our DiT consistently outperforms other methods with similar computational complexity (GFLOPs). For the GFLOPs of DiT scaling from 2.0 to 10.3, our model outperforms the state-of-the-art models consistently with a gap of 1.0 points. It should be noted that the FLOPS of DiT is the average value of multiple experiments. With the model size increases, our Focal-Base model achieves 84.8%, surpassing all other models using comparable FLOPs.

4.2 Object detection and instance segmentation

Our experiments on the challenging COCO benchmark [46] involved object detection settings. For training, we used the COCO train2017 dataset (118k images) and for evaluation, we used the val2017 dataset (5k images). We verify the effectiveness of DiT backbones on top of two standard detectors, namely Mask R-CNN [11] and RetinaNet [10]. We leveraged ImageNet pre-trained weights to initialize backbones alongside Xavier [47] to initialize newly added layers. To train our detection models, we utilized 8 V100 GPUs with a batch size of 16 and applied AdamW [39] for optimization with an initial learning rate of 1×10^{-4}. As per common practices [48; 11; 49], we employed either a 1$\times$ or 3$\times$ training schedule (i.e., 12 or 36 epochs) respectively. During training, we resized the

Table 3: Object detection and instance segmentation on COCO val2017. “#P” denotes the parameter number. AP^b, AP^m refers to the bounding box AP and mask AP, respectively.

Backbone	RetinaNet $3\times$ schedule + MS							Mask R-CNN $3\times$ schedule + MS						
	GFLOPs	AP^b	AP^b_{50}	AP^b_{75}	AP^b_S	AP^b_M	AP^b_L	GFLOPs	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet50[7]	239	39.0	58.4	41.8	22.4	42.8	51.6	260	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small[8]	226	42.2	62.7	45.0	26.2	45.2	57.2	245	43.0	65.3	46.9	39.9	62.5	42.8
Swin-Tiny[2]	245	45.0	65.9	48.4	29.7	48.9	58.1	264	46.0	68.1	50.3	41.6	65.1	44.9
ConvNeXt-T[43]	243	45.2	65.7	48.5	28.8	49.3	58.9	262	46.2	67.9	50.8	41.7	65.0	44.9
Focal-Tiny[41]	265	45.5	66.3	48.8	31.2	49.2	58.7	291	47.2	69.4	51.9	42.7	66.5	45.9
MViTv2-T[51]	-	-	-	-	-	-	-	279	48.2	70.9	53.3	43.8	67.9	47.2
PVTv2-B2 [33]	290	46.4	67.6	50.1	30.6	50.2	60.2	309	47.8	69.6	52.7	42.9	66.7	46.3
DiT-B2(ours)	278	**47.5**	**68.9**	**51.4**	**31.6**	**50.3**	**61.0**	290	**48.9**	**70.5**	**53.8**	**43.7**	**67.6**	**47.2**
ResNet101[7]	315	40.9	60.1	44.0	23.7	45.0	53.8	336	42.8	63.2	47.1	38.5	60.1	41.3
PVT-Medium[8]	283	43.2	63.8	46.1	27.3	46.3	58.9	302	44.2	66.0	48.2	40.5	63.1	43.5
Swin-Small[2]	335	46.4	67.0	50.1	31.0	50.1	60.3	354	48.5	70.2	53.5	43.3	67.3	46.6
Focal-Small[41]	367	47.3	67.8	51.0	31.6	50.9	61.1	401	48.8	70.5	53.6	43.8	67.7	47.2
PVTv2-B3 [33]	378	46.8	68.0	50.5	31.2	50.7	60.4	397	48.1	69.9	53.1	43.2	67.0	46.6
DiT-B3(ours)	355	**47.6**	**68.5**	**52.0**	**31.7**	**51.6**	**61.1**	369	**48.8**	**70.7**	**53.7**	**43.7**	**67.7**	**47.3**
ResNeXt101-64x4d[7]	473	41.8	61.5	44.4	25.2	45.4	54.6	493	44.4	64.9	48.8	39.7	61.9	42.6
PVT-Large[8]	345	43.4	63.6	46.1	26.1	46.0	59.5	364	44.5	66.0	48.3	40.7	63.4	43.7
Swin-Base[2]	477	45.8	66.4	49.1	29.9	49.4	60.3	496	48.5	69.8	53.2	43.4	66.8	46.9
Focal-Base	514	46.9	67.8	50.3	**31.9**	50.3	61.5	533	49.0	70.1	53.6	43.7	67.6	47.0
PVTv2-B4 [33]	482	47.1	67.8	50.7	30.2	51.0	62.0	500	48.5	69.2	52.9	43.1	66.6	46.8
DiT-B4(ours)	456	**48.0**	**68.7**	**51.7**	31.7	**51.9**	**63.0**	465	**49.4**	**70.2**	**53.8**	**44.0**	**67.8**	**47.9**

images to have a shorter side of 800 pixels, while ensuring that the longer side did not exceed 1,333 pixels. For models trained using the $3\times$ schedule, we randomly resized the shorter side of the input image within the range of [640, 800]. In the testing phase, however, the shorter side of the input image was fixed at 800 pixels.

In order to validate the effectiveness of our DiT, we trained four different object detectors including Cascade R-CNN [50], ATSS [44], RepPoints [45] and Sparse R-CNN [42] by following the approach of PVTv2 [33]. We use DiT as the backbone for all four models and employed a $1\times$ training schedule. The box mAPs on the COCO validation set are presented in Tab. 2. Our DiT-B2 achieved superior performance compared to Swin-Tiny, PVTv2-B2 by 1.2-4 points on all methods. These consistent and substantial improvements across various detection methods, as well as RetinaNet and Mask R-CNN, indicate that our DiT can serve as a versatile backbone for a wide range of object detection applications.

To further verify the effectiveness of our proposed DiT, the performance comparison between our DiT, CNN-based models and current Transformer-based state-of-the-art methods are presented in Tab. 3, with bbox mAP (AP^b) and mask mAP (AP^m) reported. we consistently outperform CNN-based models by a margin of 6.7-8.5 points. Furthermore, compared with other methods that utilize multi-scale Transformer architectures, we demonstrate significant improvements across all settings and metrics. Notably, in comparable settings, our DiT achieves a mAP gain of 0.8-1.1 points over the current best approach, PVTv2 [33] and Focal [41]. Unlike other multi-scale Transformer models, our approach enables both low-resolution fine-grain and high-resolution coarse-grain interactions for the visual token. We also provide more comprehensive comparisons by training our models with a 1× schedule and presenting the detailed numbers for RetinaNet and Mask R-CNN in Tab. 4, along with the number of associated computational costs for each model. Even in $1\times$ schedule, our models demonstrate a gain of 0.9-1.1 over the SoTA models in comparable settings.

4.3 Semantic segmentation

To evaluate the performance of semantic segmentation, We utilized ADE20K [52] as our benchmark dataset for semantic segmentation, following the methodology in PVTv1 [8]. To ensure fair comparisons, we evaluated the performance of DiT using the Semantic FPN [53] framework. During the training phase, we initialize the backbone with pre-trained weights from ImageNet [35] and apply Xavier [47] initialization to the newly added layers. Our model is optimized with AdamW [39] using an initial learning rate of 1e-4. For consistency with common practices [53; 54], we train our models for 40k iterations with a batch size of 16 on 4 V100 GPUs. A polynomial decay schedule with a power of 0.9 is employed to decay the learning rate. We randomly resize and crop the image to 512×512 during training, while only the shorter side is rescaled to 512 pixels during testing.

Table 4: Comparisons with CNN and Transformer baselines and SoTA methods on COCO object detection. The box mAP (AP^b) and mask mAP (AP^m) are reported with 1× schedule.

Backbone	RetinaNet	Mask R-CNN	
	AP^b	AP^b	AP^m
ResNet18 [7]	31.8	34.0	31.2
PVTv1-Tiny [8]	36.7	36.7	35.1
PVTv2-B1 [33]	41.2	41.8	38.8
DiT-B1(ours)	**42.3**	**42.9**	**39.9**
ResNet50 [7]	36.3	38.0	34.4
PVTv1-Small [8]	40.4	40.4	37.8
Swin-Tiny [2]	42.0	43.7	39.8
Focal-Tiny [41]	43.7	44.8	41.0
PVTv2-B2 [33]	44.6	45.3	41.2
DiT-B2(ours)	**45.6**	**46.4**	**42.2**
ResNet101 [7]	38.5	40.4	36.4
PVTv1-Medium [8]	41.9	42.0	39.0
Swin-Small [2]	45.0	46.5	42.1
Focal-Small [41]	45.6	47.4	42.8
PVTv2-B3 [33]	45.9	47.0	42.5
DiT-B3(ours)	**46.8**	**48.0**	**41.7**
PVTv1-Large [8]	42.6	42.9	39.5
Swin-Base [2]	45.0	46.9	42.3
Focal-Base [41]	46.3	47.8	43.2
PVTv2-B4 [33]	46.1	47.5	42.7
DiT-B4(ours)	**46.9**	**48.2**	**43.3**

Table 5: Semantic segmentation performance of different backbones on the ADE20K validation set. "GFLOPs" is calculated under the input scale of 512×512. We verify the effectiveness of DiT backbones on the Semantic FPN framework which is similar to common practices like PVTv2.

Backbone	Semantic FPN	
	GFLOPs	mIoU(%)
ResNet18 [7]	**32.2**	32.9
PVTv1-Tiny [8]	33.2	35.7
PVTv2-B1 [33]	34.2	42.5
DiT-B1(ours)	32.7	**45.0**
ResNet50 [7]	45.6	36.7
PVTv1-Small [8]	44.5	39.8
PVTv2-B2-Li [33]	**41.0**	45.1
PVTv2-B2 [33]	45.8	45.2
DiT-B2(ours)	43.4	**47.4**
ResNet101 [7]	65.1	38.8
ResNeXt101-32x4d [7]	64.7	39.7
PVTv1-Medium [8]	61.0	41.6
PVTv2-B3 [33]	62.4	47.3
DiT-B3(ours)	**59.0**	**48.7**
ResNeXt101-64x4d[7]	103.9	40.2
PVTv1-Large [8]	79.6	42.1
PVTv2-B4 [33]	81.3	47.9
DiT-B4(ours)	**78.4**	**49.0**

According to Tab. 5, DiT outperforms PVTv2 and other models in semantic segmentation. The results show that DiT-B1/B2/B3/B4 achieve 1.1-2.5 higher performance than other method, with similar parameters and GFLOPs. Additionally, although the GFLOPs of DiT-v2 are 6% lower than those of SoTA PVTv2, the mIoU is still 2.2 points higher (47.4 vs 45.2). It demonstrates that DiT backbones extract powerful features for semantic segmentation, benefiting from our dynamic routing selection strategy.

Table 6: "Static" denotes that we initialize the parameters of the gating module and use gating masks to select the routing path. We fix all the gating parameters and set them to 1.0 which is marked as "fully" connected.

Model	GFLOPs	Acc.(%)
Static	3.8	80.7(-2.4)
Dynamic(ours)	3.8	83.1
Fully	15.6(×4.1)	83.5(+0.4)

Table 7: Different gating mask generation methods consist of *random* mask generation, selection by *attention* probabilities and selection by a *learnable* routing gating module.

Model	GFLOPs	Acc.(%)
Random	3.8	79.4(-3.7)
Attention	3.8	81.3(-1.8)
Learnable(ours)	3.8	83.1

Table 8: Ablation studies of each contribution, including dynamic scale and depth comparison on the ImageNet validation set. The baseline of them is a pure PVTv2[33].

Scale	Depth	GFLOPs	Acc.(%)
		4.0	82.0
✓		8.4	82.9
	✓	3.7	82.2
✓	✓	3.8	83.1

4.4 Ablation studies

The studies involve comparing various routing selection methods, including static, dynamic, and fully connected token routing as depicted in Tab. 6. We see our method achieves better trade-offs between Flops and accuracy than other routing selection methods. The fully-connected strategy significantly increases computational complexity and leads to minor improvement(+0.4). We also compare our method with other gating mask generation strategies including "random" mask generation, selection by tokens "attention" probabilities, and our "learnable" gating mask generation in Tab. 7. We find that our data-dependent method chooses better token routing paths and balances the complexity of the network. As illustrated in Tab. 8, the ablation studies of the dynamic scale and dynamic network depth denote our methods lead to better performance without introducing computational complexity.

4.5 Analysis

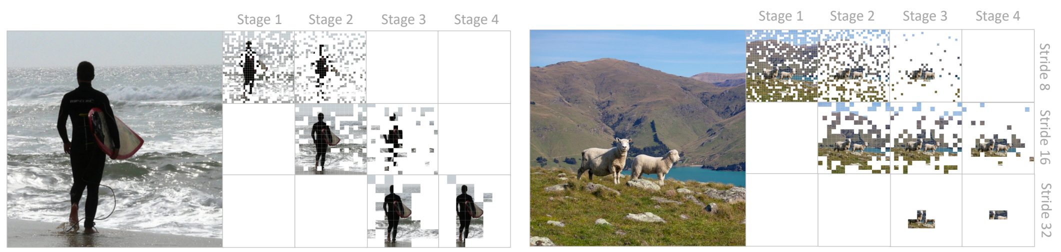


Figure 3: Visualization of the dynamic tokens routing. We show the original input image and the routing results of the four stages, where each row of the images denotes the multi-scale feature map scaling from stride of 8 to 32. The masks(in white) represent the corresponding tokens that select the paths of identify mapping. Our method can gradually focus on the most representative regions in the image and the feature map(stride=32) is more sensitive to large-scale objects. The observation implies that DynamicViT exhibits superior interpretability.

Visualization of the routing selection. To further investigate the behavior of DiT, we visualize the routing selection procedure in Fig. 3. We show the original input image and the token routing selection results after the four stages and multi-scale feature map scaling from stride of 8 to 32, where the masks in white represent the corresponding tokens select the paths of identify mapping.

We find that through the token paths selection based on the dimension of scale and depth, our DiT may neglect the uninformative tokens(e.g. background) and finally focus on the objects in the images. For the low-resolution feature(e.g. stride=32), DiT is more sensitive to large objects. This phenomenon also suggests that the DiT leads to better interpretability, i.e., it can locate the important parts in the image which contribute most to the final output. And it also illustrates that DiT may adaptively select the routing path according to the spatial scale of the objects and the difficulty of the recognition.

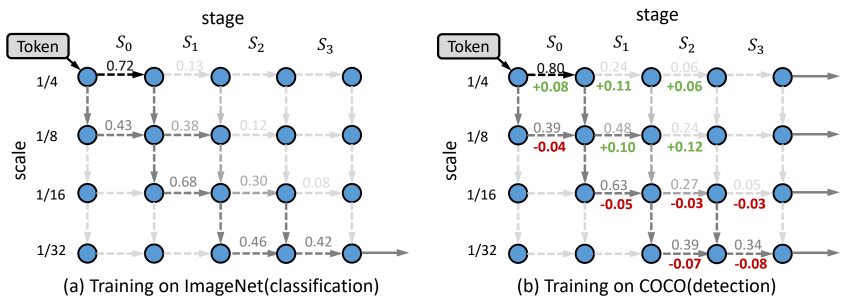


Figure 4: Statistical Analysis of the dynamic routing selection of ImageNet(Cls) and COCO(Det). The numbers over the line represent the ratio of tokens being skipped.

Statistical Analysis of the routing selection. Besides the sample-wise visualization we have shown above, we are also interested in the statistical characteristics of the routing selection, i.e., what kind of general patterns does the DiT learn from the different datasets? We then use the DiT to generate the gating mask for all the images in the ImageNet set and COCO detection set to compute the probability of mask in different scales and stages, as shown in Fig. 4. Compared to the classification task(ImageNet), the spatial scales of objects in the detection task(COCO) are smaller. Unsurprisingly, we find the token selection of the high-resolution features tends to be calculated.

5 Conclusion

In this work, we present the dynamic routing for dense prediction. The key difference from prior works lies in that we generate data-dependent dynamic routing paths according to the scale and difficulty distribution of each token. The *dynamic routing gate* selects the scale transformation and calculation routes in an end-to-end manner, which learns to skip some useless operations for the trade-off between accuracy and Flops. Gumbel-Softmax and *dynamic routing gate* techniques are also incorporated for the end-to-end training of the transformer model together with the prediction module. Through inserting the dynamic token routing into the backbone, each token will be calculated on an adaptive spatial scale and the dynamic number of transformer layers. DiT achieves state-of-the-art performance on COCO object detection and ADE20K semantic segmentation, surpassing previous best methods. However, we believe our dynamic token routing is also applicable to monolithic vision Transformers and Transformers in both vision and language domains. We leave this as a promising direction to further explore in the future.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[2] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.

[4] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris M Kitani. Rethinking transformer-based set prediction for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3611–3620, 2021.

[5] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34:12077–12090, 2021.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021.

[9] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366, 2021.

[10] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[12] Matus Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR, 2016.

[13] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

[14] Gang Zhang, Ziyi Li, Jianmin Li, and Xiaolin Hu. Cfnet: Cascade fusion network for dense prediction. *arXiv preprint arXiv:2302.06052*, 2023.

[15] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

[16] Mohsen Fayyaz, Soroush Abbasi Kouhpayegani, Farnoush Rezaei Jafari, Eric Sommerlade, Hamid Reza Vaezi Joze, Hamed Pirsiavash, and Juergen Gall. Adaptive token sampling for efficient vision transformers. *European Conference on Computer Vision (ECCV)*, 2022.

[17] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016.

[18] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[19] Youngwan Lee, Jonghee Kim, Jeffrey Willette, and Sung Ju Hwang. Mpvit: Multi-path vision transformer for dense prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7287–7296, 2022.

[20] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

[21] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021.

[22] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31, 2021.

[23] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9981–9990, 2021.

[24] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12259–12269, 2021.

[25] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[26] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[27] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[28] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8817–8826, 2018.

[29] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

[30] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *NIPS*, 2017.

[31] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

[32] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8553–8562, 2020.

[33] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, 2022.

[34] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[35] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[37] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[38] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[39] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[40] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[41] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*, 2021.

[42] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14454–14463, 2021.

[43] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

[44] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9759–9768, 2020.

[45] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *Advances in Neural Information Processing Systems*, 33:21002–21012, 2020.

[46] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[47] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[48] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[49] Yuchen Ma, Songtao Liu, Zeming Li, and Jian Sun. Iqdet: Instance-wise quality distribution sampling for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1717–1725, 2021.

[50] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[51] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *CVPR*, 2022.

[52] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.

[53] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6399–6408, 2019.

[54] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.