VisionLLaMA: A Unified LLaMA Interface for Vision Tasks

Xiangxiang Chu[1], Jianlin Su[3], Bo Zhang[1], Chunhua Shen[2]
[1] Meituan Inc. [2] Zhejiang University, China [3] Moonshot AI, China

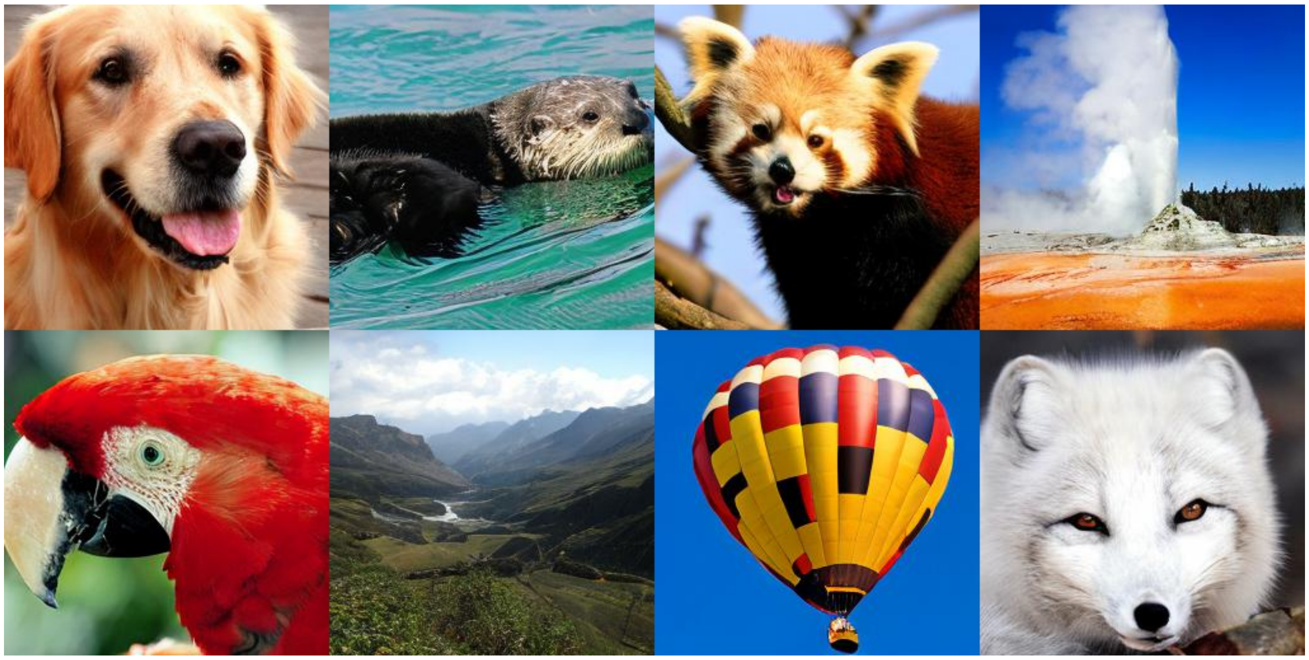
Figure 1. **Generated image samples by our DiT-LLaMA-XL** of resolution (256, 256) with a CFG ratio of 4.0. Best viewed on screen.

Abstract

Large language models are built on top of a transformer-based architecture to process textual inputs. For example, the LLaMA family of models stands out among many open-source implementations. Can the same transformer be used to process 2D images? In this paper, we answer this question by unveiling a LLaMA-like vision transformer in plain and pyramid forms, termed **VisionLLaMA**, *which is tailored for this purpose. VisionLLaMA is a unified and generic modeling framework for solving most vision tasks. We extensively evaluate its effectiveness using typical pre-training paradigms in a good portion of downstream tasks of image perception and especially image generation. In many cases, VisionLLaMA have exhibited substantial gains over the previous state-of-the-art vision transformers. We believe that VisionLLaMA can serve as a strong new baseline model for vision generation and understanding. Our code will be released at* `https://github.com/Meituan-AutoML/VisionLLaMA`.

1. Introduction

Large language models have aroused great interest in the research community. One of the most influential and representative work is LLaMA [66, 67]. Many recent works have converged to this architecture and solutions for various applications are built upon the open-sourced models. Besides, we have witnessed the blooming of multimodal models, where many methods also heavily rely on LLaMA for text processing and CLIP-fashioned [51] vision transformers [22] for visual perception. Meanwhile, many endeavors [23, 38, 73] have been devoted to accelerating the inference speed and/or the memory cost of LLaMA. In a word, LLaMA is now the *de facto* architecture.

Observing its success, a straightforward and interesting question is whether the LLaMA architecture can be another victory in the vision modality. If the answer is affirmative, then both vision and language models can use the same unified architecture and enjoy various deployment techniques designed for LLaMA on the fly. Unfortunately, it is nontrivial to answer this question because there are some distinct differences between these two modalities. Firstly, it is common sense that text sequences are organized into one dimension, while vision requires two or more. Secondly, numerous vision tasks rely on pyramid backbones to perform better, while the LLaMA is a plain encoder. Thirdly, it is necessary to handle input images and videos with different resolutions. Our paper aims to resolve these difficulties and bridge the architectural gap between different modalities. Our main contributions are summarized as follows:

1. We propose VisionLLaMA, a vision transformer architecture similar to LLaMA to reduce the architectural differences between language and vision.

2. We investigate means to adapt VisionLLaMA to tackle common vision tasks, including image comprehension and creation (Figure 1). We examine two well-known vision architecture schemes (plain and pyramid) and assess their performance under supervised and self-supervised learning scenarios. Additionally, we introduce AS2DRoPE (*i.e.* auto-scaled 2D RoPE), which expands rotated positional encoding from 1D to 2D and utilizes interpolation scaling to accommodate arbitrary resolutions.

3. Without bells and whistles, VisionLLaMA significantly outperforms the widespread and carefully fine-tuned vision transformer by clear margins across many representative tasks such as image generation, classification, semantic segmentation, and object detection. Extensive experiments indicate that VisionLLaMA demonstrates faster convergence speed and better performance than existing vision transformers.

2. Related Work

Vision Transformer. ViT [22] successfully applied Transformer [68] from natural language processing to the vision world and many more efficient and powerful follow-up works are induced, like DeiT [65], Swin [43], PVT [70], and Twins [12]. The pre-training paradigm has been shifted from supervised learning on large-scale categorically labeled datasets like ImageNet [19] to unsupervised learning [25], and to contrastive learning on huge amounts of image-text pairs as in CLIP [51]. DiT [50] adopts a transformer that operates on latent patches for diffusion models [28, 60], outperforming the commonly used U-Net backbone [54].

Large Language/Multi-modal Models Proprietary models like GPT4 [48] have been taking the lead in the LLM competition, though their technical details are hidden from the public. In contrast, the community has blossomed to release a myriad of open-source counterparts. For instance, BLOOM [57] and LLaMA [66] catch up with the performance of the closed model GPT-3 [6]. Later in copious detail, LLaMA-2 [67] describes a pack of architectural tweakings including pre-normalization called RMSNorm [80], the activation function SwiGLU [59], rotary positional embeddings RoPE [62], as well as a dedicated training pipeline, which comprises self-supervised pre-training and supervised fine-tuning enhanced by Reinforcement Learning with Human Feedback (RLHF). Many vision language models [36, 40, 41, 72, 83] are built on LLaMA and show impressive results on the visual dialog, reasoning, perception, and so on. The LLaMA architecture has also been applied in resource-limited multimodal scenarios such as mobile phones [10, 11] recently and shows potential applications.

Diffusion Models. Diffusion models, represented by Denoising Diffusion Probabilistic Models (DDPMs) [28, 60], score-based generative models (SGMs) [32, 61] and classifier-free diffusion guidance [29], are the new de facto paradigm for image generation, surpassing the previous methodology GAN [24]. The mechanism of diffusion models is based on the idea of gradually adding noise to data and then learning to denoise it. Challenges remain for the computationally expensive training and sampling process, the need for large amounts of data for training, and the difficulty in controlling the generation process. Most lately, OpenAI brings about transformer-based text-conditional diffusion models (the largest one called Sora) [5] jointly trained on videos and images of variable durations, resolutions, and aspect ratios to deliver high-fidelity videos simulating real-world scenes. The recent and concurrent work [45] explores how to deal with image generation with flexible target resolutions. Compared with [45], our target is to build a universal vision transformer for various vision tasks.

Positional Encoding for Transformers. Transformer [68] originally comes with 2D absolute position embeddings in sinusoidal forms. In contrast, the relative ones as in [58] pay attention to the relations of input tokens and can handle variable lengths of sequences. Rotary positional embeddings [62] are introduced to encode both absolute and relative positional information, which is proven to be effective in large language models [66]. Conditional positional embeddings [13] are proposed to add positional information for vision transformers according to the input image, with the benefit of boosted performance and generalizability to arbitrary input resolutions. As for LLMs, the models are usually pre-trained with a given fixed context length [66, 67, 77] and then fine-tuned to a larger context length to support long context inference. [8] extends the context length of LLaMA by simple positional interpolations. Base frequency adjustment of RoPE is also studied by [76] to enable long-context continued training. NTK-Aware scaled RoPE allows LLaMA to have an extended context size without fine-tuning and minimal perplexity degradation [55].

Masked Image Modeling. Masked image modeling is a powerful pre-training scheme that learns strong representations. BEiT [3] extends BERT [20] to computer vision by pre-training a Transformer model with masked embeddings to predict discrete visual tokens. Masked Autoencoder (MAE) [25] is a self-supervised learning approach that masks random patches of input images and trains an autoencoder to reconstruct the original images. SiMMIM [75] is a simplified version of the MAE approach that uses a lightweight one-layer head to predict raw pixel values. MaskFeat [71] is an extension of the MAE approach that involves predicting not only the raw pixel values of the masked patches but also additional features such as handcrafted HOG descriptor [17] and deep features, which can improve the performance of the model on downstream tasks.

3. Method

3.1. Plain Transformer

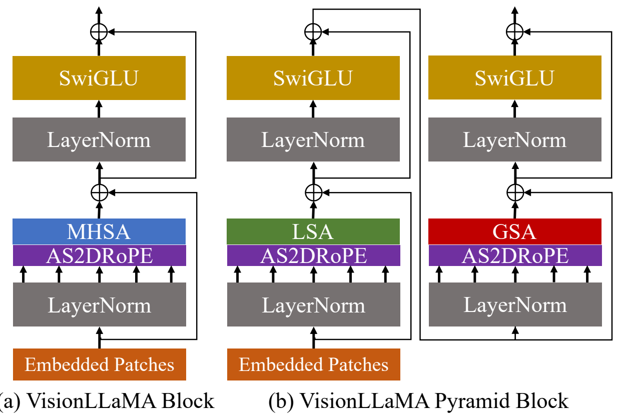

Figure 2. Our VisionLLaMA block (a) in plain Transformer and its variant block (b) in pyramid Transformer.

Our plain VisionLLaMA follows the pipeline of ViT [22] and we retain the architecture design of LLaMA as closely as possible. For an image of $H \times W$, it's firstly transformed and flattened into $N = \frac{H \times W}{P^2}$ non-overlapped patches $X \in \mathcal{R}^{N \times C}$. Then a class token is prepended at the beginning of the sequence and the whole sequence is processed by L VisionLLaMA blocks. Unlike [22], we do not add positional encodings to the input sequence since our basic block readily contains positional encoding. Specifically, the basic block differs from the standard ViT block by two components: self-attention with positional encoding (RoPE) [62] and SwiGLU activation [59]. We still utilize LayerNorm [2] instead of RMSNorm [80] since we find the former behave better through the classification experiment (see Table 11g). The basic block is illustrated in Figure 2 (a). It should be noted that directly applying 1D RoPE in vision tasks cannot well generalize to other resolutions, which is different from the training resolution. Therefore, we extend it to the 2D form. It can be formally written as,

$$\begin{aligned}
\mathbf{z}_{ij}^{l} &= \text{MHSA}\left(\text{AS2DRoPE}\left(\text{LayerNorm}\left(\mathbf{z}_{ij}^{l-1}\right)\right)\right) + \mathbf{z}_{ij}^{l-1}, \\
\mathbf{z}_{ij}^{l} &= \text{SwiGLU}\left(\text{LayerNorm}\left(\mathbf{z}^{l}{}_{ij}\right)\right) + \mathbf{z}_{ij}^{l}, \\
& i \in \{1, 2, ..., m\}, j \in \{1, 2, ..., n\}.
\end{aligned} \tag{1}$$

where z_{ij}^{l} means the output of the l block at position (i, j).

3.2. Pyramid Transformer

It's straightforward to apply VisionLLaMA to window-based transformers that utilize additive relative position encoding, such as Swin [43]. In this paper, we choose a stronger baseline Twins [12] to explore how to build a powerful pyramid transformer under strictly controlled settings. The original architecture of Twins exploits a conditional position encoding and interleaved local-global information exchange in the form of local and global attention. These components can be found in various transformers, which means it is not difficult to apply VisionLLaMA in other pyramid transformer variants by following our method. Note that our target is not to invent a novel pyramid vision transformer, but to show how we adapt the basic design of VisionLLaMA based on the existing ones. Therefore, we simply conform to the smallest modifications to the architecture and hyper-parameters. Following the name convention of [12], the two consecutive blocks can be written as,

$$\begin{aligned}
\hat{\mathbf{z}}_{ij}^{l} &= \text{LSA}\left(\text{AS2DRoPE}\left(\text{LayerNorm}\left(\mathbf{z}_{ij}^{l-1}\right)\right)\right) + \mathbf{z}_{ij}^{l-1}, \\
\mathbf{z}_{ij}^{l} &= \text{SwiGLU}\left(\text{LayerNorm}\left(\hat{\mathbf{z}}_{ij}^{l}\right)\right) + \hat{\mathbf{z}}_{ij}^{l}, \\
\hat{\mathbf{z}}^{l+1} &= \text{GSA}\left(\text{AS2DRoPE}\left(\text{LayerNorm}\left(\mathbf{z}^{l}\right)\right)\right) + \mathbf{z}^{l}, \\
\mathbf{z}^{l+1} &= \text{SwiGLU}\left(\text{LayerNorm}\left(\hat{\mathbf{z}}^{l+1}\right)\right) + \hat{\mathbf{z}}^{l+1}, \\
& i \in \{1, 2, ..., m\}, j \in \{1, 2, ..., n\}.
\end{aligned} \tag{2}$$

where LSA is the local self-attention operation within a group and GSA is the global sub-sampled attention by interacting with the representative keys from each sub-window $\hat{\mathbf{z}}_{ij} \in \mathcal{R}^{k_1 \times k_2 \times C}$ and $m \times n$ is the sub-window shape.

We remove the conditional position encoding in our pyramid VisionLLaMA since AS2DRoPE already contains positional information. Besides, we also remove the class tokens and use GAP (global average pooling) before the classification head as [12, 13]. The basic block in this setting is illustrated in Figure 2(b).

3.3. Training or Inference Beyond the Sequence Length

From 1D RoPE to 2D. Handling different input resolutions is a common requirement in vision tasks. Convolutional neural networks use the sliding window mechanism to deal with the variable length. In contrast, most vision transformers apply local window operations or interpolations. For instance, DeiT [65] adopts bicubic interpolations when trained on different resolutions. CPVT [13] uses convolution-based position encoding. Here we evaluate the performance of 1D RoPE [62]. Specifically, our pyramid VisionLLaMA based on Twins-SVT-S with 1D RoPE achieves 81.5% top-1 accuracy on an input of 224×224. However, the performance severely degrades to zero when evaluated on 448×448. Therefore, we extend the 1D RoPE to 2D. As for the multi-head self-attention, the 2D RoPE is shared across different heads. Specifically, given a token $x_{i,j} \in \mathcal{R}^d$, we obtain its position-encoded token $x^{\mathrm{PE}}_{i,j} = \mathbf{R}_{i,j} x_{i,j}$, and the diagonal matrix $\mathbf{R}_{i,j} \in \mathcal{R}^{d \times d}$ can be written as,

$$\begin{bmatrix}
\cos(i\theta_0) & -\sin(i\theta_0) & 0 & 0 & \dots & 0 & 0 & 0 \\
\sin(i\theta_0) & \cos(i\theta_0) & 0 & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & \cos(j\theta_0) & -\sin(j\theta_0) & \dots & 0 & 0 & 0 \\
0 & 0 & \sin(j\theta_0) & -\cos(j\theta_0) & \dots & 0 & 0 & 0 \\
 & & & & & & & \\
0 & 0 & 0 & \dots & \cos(i\theta_{d-4}) & -\sin(i\theta_{d-4}) & 0 & 0 \\
 & & & & \sin(i\theta_{d-4}) & \cos(i\theta_{d-4}) & 0 & 0 \\
0 & 0 & 0 & \dots & 0 & 0 & \cos(j\theta_{d-4}) & -\sin(j\theta_{d-4}) \\
0 & 0 & 0 & \dots & 0 & 0 & \cos(j\theta_{d-4}) & -\sin(j\theta_{d-4})
\end{bmatrix}$$

where $\theta_m = 10000^{-m/d}$ and $m \in \{0, 4, 8, ..., d-4\}$. Note that $\mathbf{R}$ is an orthogonal matrix. We make minor modifications to the frequency selection [62] and make two axes share the same frequency. It is easy to verify that

$$R^T_{i_1,j_1} R_{i_2,j_2} = R_{i_1-i_2,j_1-j_2}. \tag{3}$$

Positional interpolation helps 2D RoPE to better generalize. Inspired by [8], which uses interpolation to extend the context window of LLaMA, involving higher resolution is analogous to extending the 2D context window of VisionLLaMA. Unlike the language task [8] with an enlarged fixed context length, vision tasks like object detection usually deal with different sampled resolutions at different iterations. We train our small model using an input resolution of 224×224 and evaluate the performance on the larger resolutions without re-training, which guides us to apply good strategies of interpolation or extrapolation. Consequently, we apply *auto-scaled interpolation* (so-called AS2DRoPE) based on an 'anchor resolution'. Without loss of generality, we assume handling the square image of $H \times H$ and an anchor resolution $B \times B$ during the training, we calculate

$$\mathbf{R}'_{i,j} x_{i,j} = \mathbf{R}_{i \cdot B/H, j \cdot B/H}, \tag{4}$$

which can be efficiently implemented and does not introduce an extra cost. Note if the training resolution is kept unchanged, AS2DRoPE degenerates as a 2D RoPE.

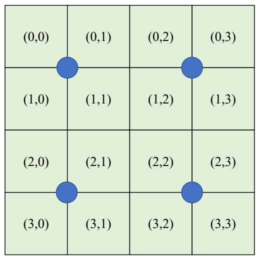

Figure 3. Position calibration for GSA's keys using a simple case of 4 × 4 resolution and a kernel size of 2 × 2. The positions of the four points (abstraction keys) are (0.5, 0.5), (1, 2.5), (2.5, 0.5), (2.5, 2.5).

As for the GSA under the pyramid setting, we require special treatments since we need to add positional information to the summarized keys. These sub-sampled keys are generated by abstraction on the feature maps. Without loss of generality, we use a convolution with a kernel size of $k \times k$ and stride of k. The coordinate of the generated key can be formulated as the average of the sampled features. We show a simple example in Figure 3.

4. Experiments

We evaluate the effectiveness of VisionLLaMA on image generation, classification, segmentation, and detection. Unless otherwise specified, all models are trained on 8 NVIDIA Tesla A100 GPUs.

4.1. Image Generation

Image generation based on the DiT framework. We apply VisionLLaMA under the DiT framework [50], which is a representative work of image generation using vision transformers and DDPM [28]. Specifically, we replace the original vision transformer of DiT with VisionLLaMA while keeping other components unchanged. This controlled experiment manifests the generality of VisionLLaMA on the image generation task. Moreover, we do not change the original hyper-parameters, although it may be sub-optimal to achieve the best performance. We also use the pre-trained VAE [34] (the ft-EMA VAE model) from SD [53], which has a down-sample factor of 8. For classifier-free guidance, we use a coefficient of 1.5. The training resolution of the image is 256 × 256. As suggested by [50], we choose the strongest adaLN-Zero version as our implementation. We also use flash attention [18] and mixed precisions to speed up the training. Note that FID is known to be sensitive to small implementation details [49]. To make accurate calculations and fair comparisons, we use the TensorFlow tool from [21] as [50].

We choose 250 sample steps of DDPM as [50] and show the result in Table 1. As a common practice, FID is regarded as a primary metric. We also report other secondary metrics such as sFID [47], Precision/Recall [35], and Inception Score [56]. Most experiments are controlled on 400k training steps. VisionLLaMA significantly outperforms DiT across various model sizes. We also extend the training steps of XL models to 2352k steps to evaluate whether our models have the faster convergence advantage or still behave better under the setting of longer training epochs. DiT-LLaMA-XL/2 has 0.83 lower FID [27] than DiT-XL/2, indicating that VisionLLaMA not only has better computing efficiency but higher performance than DiT. We show some generated samples in Figure 1 using our XL model.

Image generation based on the SiT framework. SiT [46] has a flexible choice of drift and diffusion coefficients, which is supported by the recently proposed interpolant framework [1]. It improves the performance of image generation using vision transformers by clear margins. Orthogonally, we replace the vision transformer in SiT with VisionLLaMA to evaluate the benefits of better model architecture, which we call SiT-LLaMA. Our implementation is based on the released code of [46] with carefully controlled experiments. Specifically, we do not change the hyperparameters, although its default setting may be sub-optimal. All the models are trained using the same number of steps. We use *linear interpolant* and the velocity model for all experiments. To make fair comparisons, we also rerun the released code and sample 50k 256×256 images using the 250 steps SDE sampler (Euler) and report the result in Table 2. SiT-LLaMA uniformly outperforms SiT across models with various levels of capacities by clear margins. Compared with SiT-L/2, SiT-LLaMA-L/2 decreases by 5.0 FID, whose magnitude is larger than the boost from the invention of a new framework (4.0 FID). We also report the more efficient ODE sampler (dopri5) in Table 13, our performance gap remains. Similar to the observation of [46], SDE has better performance than its ODE counterpart.

4.2. Classification on ImageNet

4.2.1 Supervised Training

In this section, we focus on supervised training on the ImageNet-1K dataset [19] to make fair comparisons. We exclude other datasets or distillation tricks. All the models are trained using the ImageNet-1K training set, and we report the accuracy of the validation set in Table 3.

Plain Vision Transformer Comparison. DeiT3 [65] is the state-of-the-art plain vision transformer, which proposes special data augmentations and performs extensive hyperparameter search to boost the performance of DeiT [64]. During the reproduction of DeiT3, we observe that it is sensitive to hyperparameters and prone to overfitting. Replacing the class token with GAP (global average pooling) [13] leads to a 0.7% top-1 accuracy drop for the DeiT3-Large model after 800 epochs of training. Therefore, we use the class token instead of GAP in the plain transformer and report the result in Table 3, where VisionLLaMA achieves a top-1 accuracy comparable to DeiT3. The detailed hyperparameter is listed in the appendix. Note that the accuracy on a single resolution does not provide comprehensive comparisons, we also evaluate the performance across different image resolutions as [13] and report the result in Table 4. As for DeiT3, we use the bicubic interpolation for the learnable positional encoding. Although these two models have comparable performance at the resolution of 224×224, the gap is enlarged when the resolution is increased, which means our method generalizes better across different resolutions, which is a vital function for many downstream tasks such as object detection.

Pyramid Vision Transformer. We use the same architecture as Twins-SVT [12] and the detailed configuration is listed in Table 17. We remove the conditional position encoding since VisionLLaMA already contains one kind of rotary position encoding. Therefore, VisionLLaMA is a convolution-free architecture. We do not tune the hyperparameters and directly follow the setting provided in [12]. Although it's suboptimal, it can still achieve competitive performance. As [12, 13], we do not use the class token and apply GAP. In particular, all the models are trained for 300 epochs with a batch size of 1024. The learning rate is initialized to be 0.001 and decayed to zero within 300 epochs following the cosine strategy. The result is shown in Table 3 and our method achieves comparable performance as Twins across various levels of models and outperforms Swin [43] consistently. We further compare the pyramid transformers using popular downstream tasks, which are shown in the later sections.

4.2.2 Self-Supervised Training

There are two common approaches to evaluating the performance of the self-supervised vision transformers [25] using the ImageNet dataset. In this section, we make comparisons based on these two ways. To make fair comparisons, we limit the training data to ImageNet-1K. We also exclude any component that utilizes CLIP [51], DALLE [52], or distillation, which can be orthogonally combined to further boost the performance. Our implementation is based on the MMPretrain framework [15]. We utilize the MAE framework and replace the encoder using VisionLLaMA while keeping other components unchanged. This minor modified setting forms a controlled experiment to evaluate the role of our approaches. Moreover, we use the same hyperparameter as [25], which is suboptimal to our method. Fortunately, this simple setting still achieves a significant performance boost over the strong baseline.

Model	CFG	Flops (G)	Params (M)	Training Steps (K)	Learning Rate	FID↓	sFID↓	Precision↑	Recall↑	IS↑
DiT-B/4	N	5.56	130	400	0.0001	68.38	12.66	36.07	54.71	20.27
DiT-LLaMA-B/4	N	5.56	130	400	0.0001	63.17	12.63	38.27	56.75	22.47
DiT-B/4	Y	5.56	130	400	0.0001	45.38	9.97	46.89	53.66	34.27
DiT-LLaMA-B/4	Y	5.56	130	400	0.0001	39.51	9.82	50.46	54.75	40.17
DiT-L/4	N	19.70	458	400	0.0001	44.37	8.97	48.16	61.53	32.25
DiT-LLaMA-L/4	N	19.70	458	400	0.0001	40.32	9.04	49.87	61.61	36.56
DiT-L/4	Y	19.70	458	400	0.0001	22.51	7.08	62.67	55.27	66.58
DiT-LLaMA-L/4	Y	19.70	458	400	0.0001	18.64	7.01	65.40	54.35	78.52
DiT-XL/4	N	29.05	675	400	0.0001	43.01	-	-	-	-
DiT-LLaMA-XL/4	N	29.05	675	400	0.0001	35.99	8.48	52.31	61.65	41.18
DiT-XL/4	Y	29.05	675	400	0.0001	22.52	7.09	62.68	55.27	66.58
DiT-LLaMA-XL/4	Y	29.05	675	400	0.0001	18.69	7.02	65.67	55.57	78.32
DiT-XL/2	N	118.64	675	2352	0.0001	10.67	-	-	-	-
DiT-LLaMA-XL/2	N	118.64	675	2352	0.0001	9.84	6.47	67.45	66.71	117.72
DiT-LLaMA-XL/2	Y	118.64	675	2352	0.0001	**2.42**	4.51	83.03	56.82	265.39

Table 1. Image generation comparisons using the DiT framework [50]. All the models are trained using an image resolution of 256×256 with a batch size of 256. Metrics are calculated using the sampled 50k images. IS: inception score [56].

Model	Flops (G)	Params (M)	Training Steps (K)	Learning Rate	FID↓	sFID↓	Precision↑	Recall↑	IS↑
SiT-S/2 †	6.06	33	400	0.0001	58.15	9.12	41.01	60.23	24.72
SiT-LLaMA-S/2	6.06	33	400	0.0001	53.90	8.78	42.98	60.36	26.74
SiT-B/2 †	23.01	130	400	0.0001	35.54	6.57	52.68	64.38	42.33
SiT-LLaMA-B/2	23.01	130	400	0.0001	29.53	6.32	56.07	64.07	50.13
DiT-L/2	80.71	458	400	0.0001	23.3	-	-	-	-
SiT-L/2 †	80.71	458	400	0.0001	19.34	5.28	63.00	63.60	70.47
SiT-LLaMA-L/2	80.71	458	400	0.0001	14.32	5.17	66.39	63.64	86.85
SiT-XL/2 †	118.64	675	400	0.0001	16.98	5.07	65.12	64.10	77.06
SiT-LLaMA-XL/2	118.64	675	400	0.0001	**12.20**	5.03	67.86	63.08	95.28

Table 2. Image generation comparisons using the SiT framework [46]. All the models are trained using an image resolution of 256×256 with a global batch size of 256. Metrics are calculated using the sampled 50k images without classifier-free guidance. IS: inception score. The FID is calculated by 250 steps SDE Euler sampler. †: reproduced result using the released code.

Full fine-tuning. In such a setting, the model is first initialized using the pre-trained weights and then trained for extra epochs with totally trainable parameters. Trained by 800 epochs on the ImageNet, VisionLLaMA-Base achieves 84.0% top-1 accuracy, which exceeds ViT-Base by 0.8%. Note that our method uses a mask ratio of 0.75 as [25], whose training speed is about 3 times faster than SimMIM [75]. We also increased the training epochs to 1600 to verify whether VisionLLaMA keeps the advantage given sufficient training resources. VisionLLaMA-Base achieves new state-of-art result among MAE variants, 84.3% top-1 accuracy, which outperforms ViT-Base by 0.9%. This result is even higher than MaskFeat [71] where new training objectives are proposed. Regarding full fine-tuning having a risk of performance saturation [42, 69], our boost is significant. Next we resort to the linear probing metric to provide extra evaluations, which is considered a more reliable evaluation for representative learning by a recent work [9].

Linear probing. In this setting, the model is initialized by the pre-trained weights from the SSL stage. Then, the whole backbone is frozen except for the classifier head during the training. The result is shown in Table 5. With a training cost of 800 epochs, VisionLLaMA-Base outperforms ViT-Base-MAE by 4.6%. It also exceeds ViT-Base-MAE, which is trained for 1600 epochs. When VisionLLaMA is trained for 1600 epochs, VisionLLaMA-Base achieves 71.7% top-1 accuracy. We also scale up to have VisionLLaMA-Large, where our method exceeds ViT-Large by 3.6%.

Model	Param (M)	Setting	Top-1 (%)
DeiT-Small [64]	22	224I 300E	79.9
CPVT-Small-GAP [13]	23	224I 300E	81.5
DeiT3-Small [65]	22	224I 800E	81.4
VisionLLaMA-S [65]	22	224I 800E	81.6
Swin-T [43]	29	224I 300E	81.3
Twins-SVT-S [12]	24	224I 300E	81.7
Pyramid VisionLLaMA-S	24	224I 300E	81.6
Swin-S [43]	50	224I 300E	83.0
Twins-SVT-B [12]	56	224I 300E	83.2
Pyramid VisionLLaMA-B	56	224I 300E	83.2
DeiT3-Base [65]	86	192I 800E + 224I 20E	83.8
VisionLLaMA-B	86	192I 800E + 224I 20E	83.6
Swin-B [43]	88	224I 300E	83.3
Twins-SVT-L [13]	99	224I 300E	83.7
Pyramid VisionLLaMA-L	99	224I 300E	83.6
DeiT3-Large†	310	160I 800E+224I 20E	84.5
VisionLLaMA-L	310	160I 800E+224I 20E	**84.6**

Table 3. Comparisons on ImageNet-1K supervised classification. All the models are trained using the ImageNet-1K dataset. †: re-trained using the official code. 160I 800E+224I 20E means two-stage training, the model is firstly trained for 800 epochs using 160×160, then trained for 20 epochs with higher image resolution 224×224.

Model	160	224	256	288	512	768
DeiT3-Large [65]	83.1	84.5	84.7	84.6	82.1	76.5
VisionLLaMA-L	83.1	**84.6**	84.7	**84.8**	**83.5**	**79.1**

Table 4. Top-1 accuracy comparison on different resolutions. The models are trained on 224 and directly evaluated on other resolutions.

4.3. Semantic Segmentation on ADE20K

4.3.1 Supervised Training

Following [12, 43], we evaluate our method using semantic segmentation on the ADE20K [82] dataset. To make fair comparisons, we limit the baselines to only using ImageNet-1K in the pre-training stage. Specifically, we make use of the UperNet [74] framework and replace the backbone with pyramid VisionLLaMA. Our implementation is based on the MMSegmentation framework [14]. Our models are trained for 160k steps with a global batch size of 16. The detailed setting of the hyperparameter is shown in Section B.7. We report the result in Table 6. Under similar FLOPs, our method outperforms both Swin and Twins by more than 1.2% mIoU.

Models	Pretrain Epochs	SFT Acc (%)	LP Acc (%)
ViT-Base-MAE† [25]	800	83.2	65.1
SemMAE [37]	800	83.4	65.0
SimMIM [75]	800	83.8	56.7
MFF-MAE [42]	800	83.6	67.0
VisionLLaMA-Base-MAE	800	84.0	69.7
ViT-Base-MAE [25]	1600	83.4	67.0
MaskFeat [71]	1600	84.0	62.3
VisionLLaMA-Base-MAE	1600	84.3	71.7
ViT-Large-MAE† [25]	800	85.4	73.7
VisionLLaMA-Large-MAE	800	**85.5**	**77.3**

Table 5. Comparison with masked image modeling SSL methods on the ImageNet validation set. †: reproduced in MMPretrain.

Models	Param (M)	mIoU (%)
Swin-S [43]	81.3	47.6
Twins-SVT-B [12]	88.5	47.7
Pyramid VisionLLaMA-B	88.5	**49.1**
Swin-B [43]	121	48.1
Twins-SVT-L [12]	133	48.8
Pyramid VisionLLaMA-L	133	**50.0**

Table 6. Performance comparisons with different backbones on ADE20K validation dataset. All backbones are pre-trained on ImageNet-1K with labels. mIoU is evaluated by the single scale setting.

4.3.2 Self-Supervised Training

We use the UperNet [74] framework to perform semantic segmentation on the ADE20K dataset, which is a popular benchmark for backbones. We carefully control the experiment and replace the ViT backbone with VisionLLaMA while keeping other components and hyperparameters unchanged. Our implementation is based on MMSegmentation [14] and the detailed hyperparameters are provided in Section B.6. The result is given in Table 7. As for the 800 epoch pre-training groups, VisionLLaMA-B significantly boosts ViT-Base by 2.8% mIoU. It also outperforms some other modifications such as introducing extra training objectives or features [42, 71] by clear margins. Moreover, those approaches introduce extra overhead for the training process and slow down the training speed. We emphasize that the training speed of a method is becoming more and more important in the age of large models. In contrast, VisionLLaMA only involves the replacement of the base model and has the same fast training speed as [25]. In principle, our method can be seamlessly combined with these modifications. We further evaluate the performance of longer pre-training epochs of 1600, VisionLLaMA-B achieves 50.2% mIoU on the ADE20K validation set, which boosts ViT-B by 2.1% mIoU.

Models	Pretrain Epochs	mIoU (%)
ViT-B†	800	46.2
SemMAE [37]	800	46.3
MFF-MAE [42]	800	47.9
VisionLLaMA-B	800	**49.0**
ViT-B	1600	48.1
MaskFeat [71]	1600	48.3
VisionLLaMA-B	1600	**50.2**

Table 7. Performance comparisons with different SSL trained backbones on ADE20K validation dataset. All backbones are pre-trained on ImageNet-1K **without labels**. mIoU is evaluated by the single scale setting. †: reproduce result using [14].

4.4. Object Detection on COCO

4.4.1 Supervised Training

We evaluate the performance of pyramid VisionLLaMA on the COCO objection detection task. Specifically, we use the Mask RCNN framework [26] and replace the backbone with pyramid VisionLLaMA, which is pre-trained for 300 epochs on the ImageNet-1K dataset as [12, 43]. Therefore, our model has the same number of parameters and FLOPs as Twins. Since our target is not to achieve a new state-of-the-art detector, this carefully controlled experiment is used to verify the validity of our method without loss of generality. Our implementation is based on the MMDetection framework [7] and the hyperparameter setting is provided in Section B.8. We report the result on standard 36 epochs (3×) in Table 8. Under this carefully controlled setting, our model outperforms both Swin and Twins. Specifically, VisionLLaMA-B exceeds Swin-S by 1.5% box mAP and 1.0 mask mAP. Compared with the stronger baseline Twins-B, our method also has an advantage of 1.1% higher box mAP and 0.8% higher mask mAP.

4.4.2 Self-Supervised Training

We apply VisionLLaMA based on the ViTDet framework [39], which utilizes plain vision transformers to achieve comparable performance as the pyramid counterpart. Specifically, we use the Mask RCNN detector and replace the vit-Base backbone (trained for 1600 epochs using MAE) with our VisionLLaMA-Base model, which is pre-trained for 800 epochs using MAE. The original ViTDet converges slowly and requires dedicated training strategies like longer training epochs (*e.g.* 100) to achieve optimal performance. During the training process, we find VisionLLaMA achieves similar performance after 30 epochs. Therefore, we directly utilize the standard 3x training strategy. We use AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We also use a layer-wise learning rate of 0.7 as [39]. The initial learning rate is 0.0001 and decayed by 0.1 at epochs 27 and 33. We use a weight decay of 0.1 and a global batch size of 64. The input image resolution is 1024×1024. Therefore, our training cost is only **36%** of the baseline. Unlike [39], we do not search for the optimal hyperparameter. The result is shown in Table 9 and VisionLLaMA outperforms ViT-B by 0.6% Box mAP and 0.8 % mask mAP.

Backbone	FLOPs (G)	Mask R-CNN 3× + MS					
		AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Swin-S [43]	222	47.6	69.4	52.5	42.8	66.5	46.4
Twins-SVT-B [12]	224	48.0	69.5	52.7	43.0	66.8	46.6
Pyramid VisionLLaMA-B	224	**49.1**	**70.5**	**54.0**	**43.8**	**67.4**	**47.0**

Table 8. Object detection and instance segmentation performance on the COCO `val2017` dataset using the Mask R-CNN framework. FLOPs are evaluated on an 800×600 image. All the backbones are trained for 300 epochs on the ImageNet-1K dataset.

Model	Pretrained	mAP^{Box}	mAP^{Mask}	Epochs
Swin-S [43]	ImageNet sup 300e	47.6	42.8	36
Twins-SVT-B [12]	ImageNet sup 300e	48.0	43.0	36
ViT-B [39]	MAE 1600e	51.6	45.7	100
VisionLLaMA-B	MAE 800e	**52.2**	**46.3**	36

Table 9. Object detection result on COCO 2017 dataset based on ViTDet [39]. sup: supervised training on ImageNet-1K

Method	100k	200k	300k	400k
SiT-S/2	89.9	71.9	64.5	59.6
SiT-LLaMA-S/2	82.88	67.1	59.3	54.6
SiT-B/2	65.76	48.37	41.05	36.90
SiT-LLaMA-B/2	56.60	40.62	34.09	30.22
SiT-L/2	45.07	29.11	23.40	20.14
SiT-LLaMA-L/2	35.39	21.82	17.23	14.91
SiT-XL/2	42.25	26.49	20.89	17.83
SiT-LLaMA-XL/2	40.46	19.00	14.84	12.79

Table 10. FID calculated with the 250-step ODE sampler in view of efficiency based on the SiT framework.

5. Ablation Study and Discussion

5.1. Ablation Studies

Unless otherwise specified, we choose the ViT-Large model (160I 800E+224I 20E) to perform ablations because we observe that it generates small variance across multiple runs, where a performance gap of more than 0.2 suffices as a guide to choosing appropriate components.

Ablation of FFN and SwiGLU. We replace FFN with SwiGLU and report the result in Table 11a. We do not observe performance gaps, therefore, we utilize SwiGLU and avoid introducing extra modifications to the LLaMA architecture. This also motivates us to focus on the ablation of the self-attention block. As we apply multi-head self-attention, the remaining two differences become the normalization and positional encoding.

Ablation of the normalization strategy. We compare the two widely used normalization methods in transformers: RMSNorm [80] and LayerNorm [2] and report the result in Table 11g. The latter has a better final performance, which indicates that *re-centering invariance* is also important in the vision tasks. We also report the training speed by the average time spent per iteration, where LayerNorm is only 2% slower than RMSNorm. Therefore, we choose LayerNorm instead of RMSNorm for better tradeoff. Note that the training speed might differ across different hardware devices and might also be affected by the overall architecture.

Next, we evaluate the role of positional encoding in two aspects, a static case using a fixed resolution and a dynamic case using variable resolutions. The former is common in the classification task while the latter is vital in downstream tasks such as segmentation and object detection.

Partial PE. We adjust the ratio of overall channels using RoPE to report the result in Table 11b, which shows good performance can be achieved if the ratio is set above a small threshold value. We do not observe significant differences across these settings. Therefore, we keep the default setting of [66] and do not follow [4, 30].

Frequency base. We change the base frequency and report the result in Table 11c, which means the performance is robust to a large range of frequencies. As a result, we keep the default value of [66] to avoid extra special treatments for deployment.

Shared PE for each head. We find that sharing the same PE across different heads (the frequency varies from 1 to 10000 in each head) is better than independent ones (the frequency varies from 1 to 10000 across all channels). The result is shown in Table 11d.

Feature abstraction strategy. We compare the two common feature extraction strategies: class token [22] and GAP [13] using the plain 'large' model and report the result in Table 11e. Using a class token is better than GAP, which is different from [13]. However, the training settings of the two cases are quite different. We also make an extra experiment using DeiT3-L to observe a similar performance gap of 0.3%. We further evaluate the performance of the 'small' and 'base' models. It's interesting to see the opposite conclusions for the small model. We suspect that the higher drop-path rate used in [65] makes it difficult for the parameter-free abstraction such as GAP to fit in the purpose.

Positional encoding strategy. We also add other absolute position encoding strategies such as a learnable PE [64] and PEG [13] on pyramid VisionLLaMA-S. We use the 'small' model due to the existence of a strong baseline and report the result in Table 11f. While the learnable PE does not boost performance, PEG slightly improves the baseline from 81.6% to 81.8%. However, we do not include PEG as a basic component regarding three aspects. Firstly, we try to keep the smallest modifications on LLaMA [66]. Secondly, our target is proposing a universal approach for various tasks like ViT [22]. For masked image frameworks like MAE [25], it is non-trivial to keep the reduced training cost of masked tokens if the backbone contains PEG. If we mask patches in the input like [75], it would greatly slow down the training speed. Moreover, containing masked patches in the encoder would incur a data distribution shift to the encoder, which severely hurts the performance of downstream tasks. In principle, we can apply sparse PEG under the MAE framework, but it will introduce the deployment-unfriendly operators. It remains an open problem whether sparse convolution contains enough positional information as its dense version [13, 33]. Thirdly, avoiding modality-bound designs paves the way for further studies that cover other modalities beyond text and vision.

Sensitivity to the input size. We further compare the performance on the enlarged and commonly used resolutions without training to report the result in Table 12. Here we use the pyramid transformer since it is more popular in downstream tasks than the plain counterpart. It is not surprising that 1D-RoPE severely suffers from the changed resolutions. NTK-Aware interpolation with $\alpha = 2$ achieves similar performance as the 2D-RoPE[1], which is indeed NTK-Aware ($\alpha = 1$). AS2DRoPE shows the best performance for larger resolution.

5.2. Discussion

We further investigate the underlying mechanisms behind our method's superior performance over ViT in various tasks. As the ablation studies have indicated, our positional encoding strategy makes a big difference. In this section, we discuss the boosted convergence speed and attempt to theoretically rationalize the underlying mechanism.

Convergence speed. For image generation, we study

[1]Although we can apply the dynamic NTK-Aware to keep the performance at 224, it does not bring in boosted performance on larger resolutions.

case	Acc
SwiGLU	84.6
FFN	84.6

(a) **FFN or SwiGLU**.

Ratio	Acc
25%	84.5
50%	84.5
100%	**84.6**

(b) **Partial ratio of RoPE**. A partial ratio does not bring improved performance.

Base	Acc
100	84.6
1000	84.6
10000	**84.6**
100000	84.4

(c) **Base frequency of RoPE**.

Shared PE	Acc
N	84.2
Y	**84.6**

(d) **Shared PE across different heads**. Shared PE for all heads is better.

Method	Class Head	Acc
VisionLLaMA-S	Class Token	81.6
VisionLLaMA-S	GAP	81.8
VisionLLaMA-B	Class Token	83.6
VisionLLaMA-B	GAP	83.6
VisionLLaMA-L	Class Token	84.6
VisionLLaMA-L	GAP	84.3
DeiT3-L [65]	Class Token	84.5
DeiT3-L†	GAP	84.2

(e) **Feature extraction strategy**. The class token is better than GAP in the training setting of DeiT3 [65].

case	Acc
Pyramid LLaMA-S	81.6
Pyramid LLaMA-S + learnable PE [64]	81.6
Pyramid LLaMA-S + PEG [13]	**81.8**

(f) **PE comparison**. Applying PEG [13] can further improve the performance.

case	Acc	Train Speed
LayerNorm [2]	84.6	0.4971s
RMSNorm [80]	84.4	0.4874s

(g) **Normalization choice**.

Table 11. **Ablation experiments** with plain transformer ViT-L/16 (DeiT3-L) on ImageNet-1K. We report the top-1 accuracy (%). If not specified, the default is: and the pre-training length is 800 epochs under an image resolution of 160×160 and 20 epochs using 224×224. Default settings are marked in gray. †: running the release code. All accuracies are top-1.

Model	224	448	512
1D-RoPE	81.5	0.01	0.01
2D-RoPE	81.6	79.5	78.4
NTK($\alpha = 2$)	81.6	79.6	78.5
NTK($\alpha = 5$)	81.3	79.6	78.6
NTK($\alpha = 10$)	81.1	79.6	78.6
AS2DRoPE	81.6	80.3	79.5

Table 12. Top-1 accuracy on different resolutions of the pyramid small model. The models are trained on 224x224 and directly evaluated on other resolutions.

the performance w.r.t the training steps. Specifically, we store the checkpoint at 100k, 200k, 300k, and 400k iterations to calculate the fidelity metrics. Since SDE is significantly slower than ODE, we opt to use the ODE sampler instead. The result of the strictly controlled experiment is listed in Table 10. It appears that VisionLLaMA converges much faster than ViT across all models. SiT-LLaMA with 300k training iterations even outperforms the baseline with 400k steps.

We also compare the convergence speed using the DeiT3-Large under the supervised training setting on ImageNet to show the top-1 validation accuracy during the 800 epochs in Figure 4. It also indicates that VisionLLaMA converges faster than DeiT3-L. We further compare the training loss across 800 epochs of the ViT-Base model under the MAE framework [25] and illustrate it in Figure 5. VisionLLaMA has lower training loss at the beginning and the trend is kept till the end.

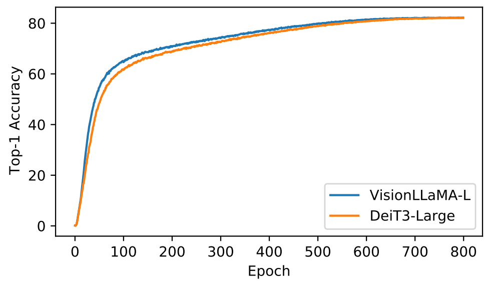

Figure 4. Faster convergence of VisionLLaMA using the setting of DeiT3.

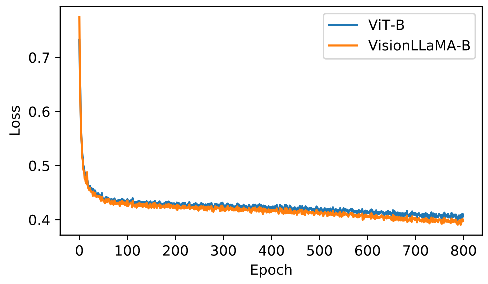

Figure 5. Loss curve of MAE pre-training on VisionLLaMA compared with ViT-B.

Theoretical Reasoning. We dive into the mechanism of our positional encodings from the theoretical viewpoint. Without loss of generality, given an input embedding of dimension $d = 4$, the query at location (i, j) can be written as $q_{i,j}$. We use $k_{i,j}$ to represent the key vector at (i, j) and $p_{i,j}$ to be the positional encoding using 2D sin-cos encoding [25,46]. The inner dot product between q_{i_1,j_1} and k_{i_2,j_2} using this additive encoding can be written as,

$$\begin{aligned} q^T_{i_1,j_1} k_{i_2,j_2} &= (q_{i_1,j_1} + p_{i_1,j_1})^T (k_{i_2,j_2} + p_{i_2,j_2}) \\ &= q^T_{i_1,j_1} k_{i_2,j_2} + p^T_{i_1,j_1} p_{i_2,j_2} + q^T_{i_1,j_1} p_{i_2,j_2} + p^T_{i_1,j_1} k_{i_2,j_2} \\ &= q^T_{i_1,j_1} k_{i_2,j_2} + f(i_1 - i_2, j_1 - j_2) + M. \end{aligned} \tag{5}$$

The first item is the inner dot product of contents. The second item reflects the positional effect in the form of $f(i_1 - i_2, j_1 - j_2)$, which plays a long-distance decaying effect. However, the third item $M = q^T_{i_1,j_1} p_{i_2,j_2} + p^T_{i_1,j_1} k_{i_2,j_2}$ means positions directly interacting with the content features, which slows down the learning process.

In contrast, the inner dot product using RoPE can be written as,

$$\begin{aligned} (R_{i_1,j_1} q_{i_1,j_1})^T (R_{i_2,j_2} k_{i_2,j_2}) &= q^T_{i_1,j_1} R^T_{i_1,j_1} R_{i_2,j_2} k_{i_2,j_2} \\ &= q^T_{i_1,j_1} R_{i_1-i_2,j_1-j_2} k_{i_2,j_2}. \end{aligned} \tag{6}$$

$R_{i_1-i_2,j_1-j_2}$ contributes a larger absolute value if the positions of q and k are close, and a smaller value if opposite. This introduces certain localities as a prior bias, which resembles the function of a convolution. Moreover, $R_{i_1-i_2,j_1-j_2}$ adjusts the dot product by the multiplication of a factor between 0 and 1, which is more flexible and faster than the addition of $f(i_1 - i_2, j_1 - j_2)$. We believe that this flexibility allows the transformer to leverage its model capacity effectively, learning a good representation without dedicating some of that capacity to introducing bias or separating position from content. In this way, VisionLLaMA not only converges faster but also has better final performance.

6. Conclusion

In a nutshell, we present VisionLLaMA to enjoy the benefits of the LLaMA architecture in the vision modality. It is trained either in supervised or self-supervised schemes to validate the power in a myriad of downstream vision tasks like image classification, detection, and segmentation. We particularly explore its image generation capacity under the diffusion framework DiT and SiT to confirm its potency. We conclude that VisionLLaMA has strong potential to serve as a new vision backbone to facilitate a large realm of downstream applications.

Acknowledgements: This work was in part supported by National Key R&D Program of China (No. 2022ZD0118-700).

References

[1] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023. 5

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3, 9, 10

[3] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 3, 16

[4] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023. 9

[5] Tim Brooks, Bill Peebles, Connor Homes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Wing Yin Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. 2

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2

[7] 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. 8

[8] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023. 3, 4

[9] Xinlei Chen, Zhuang Liu, Saining Xie, and Kaiming He. Deconstructing denoising diffusion models for self-supervised learning. *arXiv preprint arXiv:2401.14404*, 2024. 6

[10] Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, et al. Mobilevlm: A fast, reproducible and strong vision language assistant for mobile devices. *arXiv preprint arXiv:2312.16886*, 2023. 2

[11] Xiangxiang Chu, Limeng Qiao, Xinyu Zhang, Shuang Xu, Fei Wei, Yang Yang, Xiaofei Sun, Yiming Hu, Xinyang Lin, Bo Zhang, et al. Mobilevlm v2: Faster and stronger baseline for vision language model. *arXiv preprint arXiv:2402.03766*, 2024. 2

[12] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *Adv. Neural Inform. Process. Syst.*, 2021. 2, 3, 5, 7, 8, 15

[13] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, and Chunhua Shen. Conditional positional encodings for vision

transformers. In *The Eleventh International Conference on Learning Representations*, 2023. 2, 3, 4, 5, 7, 9, 10

[14] MMSegmentation Contributors. Mmsegmentation: Openmmlab semantic segmentation toolbox and benchmark, 2020. 7, 8

[15] MMPreTrain Contributors. Openmmlab's pre-training toolbox and benchmark. https://github.com/open-mmlab/mmpretrain, 2023. 5

[16] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 16

[17] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. 3

[18] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. 4

[19] 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. 2, 5

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 3

[21] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 4, 15

[22] 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. 1, 2, 3, 9

[23] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022. 1

[24] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 2

[25] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022. 2, 3, 5, 6, 7, 9, 10, 11, 15

[26] 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. 8

[27] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 5

[28] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 4

[29] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 2

[30] https://stability.ai/. Stable code 3b: Coding on the edge. 2024. 9

[31] 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. 15

[32] Aapo Hyv"arinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 2005. 2

[33] Md Amirul Islam*, Sen Jia*, and Neil D. B. Bruce. How much position information do convolutional neural networks encode? In *International Conference on Learning Representations*, 2020. 9

[34] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 4

[35] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019. 5

[36] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023. 2

[37] Gang Li, Heliang Zheng, Daqing Liu, Chaoyue Wang, Bing Su, and Changwen Zheng. Semmae: Semantic-guided masking for learning masked autoencoders. *Advances in Neural Information Processing Systems*, 35:14290–14302, 2022. 7, 8

[38] Liang Li, Qingyuan Li, Bo Zhang, and Xiangxiang Chu. Norm tweaking: High-performance low-bit quantization of large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, 2024. 1

[39] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European Conference on Computer Vision*, pages 280–296. Springer, 2022. 8, 15

[40] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023. 2

[41] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *NeurIPS*, 2023. 2

[42] Yuan Liu, Songyang Zhang, Jiacheng Chen, Zhaohui Yu, Kai Chen, and Dahua Lin. Improving pixel-based mim by reducing wasted modeling capability. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5361–5372, 2023. 6, 7, 8

[43] 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. 2, 3, 5, 7, 8

[44] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 15

[45] Zeyu Lu, Zidong Wang, Di Huang, Chengyue Wu, Xihui Liu, Wanli Ouyang, and Lei Bai. Fit: Flexible vision transformer for diffusion model. *arXiv preprint arXiv:2402.12376*, 2024. 2

[46] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024. 5, 6, 11, 16

[47] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021. 5

[48] OpenAI. Gpt-4 technical report. 2023. Technical Report. 2

[49] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11410–11420, 2022. 4

[50] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 2, 4, 6

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1, 2, 5

[52] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. arXiv, 2022. 5

[53] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 4, 15

[54] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 2

[55] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023. 3

[56] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016. 5, 6

[57] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022. 2

[58] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 2

[59] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. 2, 3

[60] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 2

[61] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 2

[62] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, page 127063, 2023. 2, 3, 4

[63] 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. 16

[64] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers and distillation through attention. In *International Conference on Machine Learning*, volume 139, pages 10347–10357, July 2021. 5, 7, 9, 10

[65] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *European Conference on Computer Vision*, pages 516–533. Springer, 2022. 2, 4, 5, 7, 9, 10, 15

[66] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, and Faisal Azhar. Llama: Open and efficient foundation language models. 2023. 1, 2, 9

[67] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 1, 2

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[69] Kirill Vishniakov, Zhiqiang Shen, and Zhuang Liu. Convnet vs transformer, supervised vs clip: Beyond imagenet accuracy. *arXiv preprint arXiv:2311.09215*, 2023. 6

[70] 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. 2

[71] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14668–14678, 2022. 3, 6, 7, 8

[72] Fei Wei, Xinyu Zhang, Ailing Zhang, Bo Zhang, and Xiangxiang Chu. Lenna: Language enhanced reasoning detection assistant. *arXiv preprint arXiv:2312.02433*, 2023. 2

[73] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023. 1

[74] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018. 7

[75] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9653–9663, 2022. 3, 6, 7, 9

[76] Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023. 3

[77] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*, 2023. 2

[78] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. 16

[79] 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. 16

[80] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019. 2, 3, 9, 10

[81] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 16

[82] 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. 7

[83] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. In *The Twelfth International Conference on Learning Representations*, 2024. 2

A. More Experiments

We evaluate the image generation using the 250 steps ODE sampler (dopri5) based on the SiT framework in Table 13.

B. Hyperparameters

B.1. Supervised Training of VisionLLaMA on ImageNet-1K

As for the plain transformer, we use the same hyperparameters as [65]. The detailed setting is provided in Table 14. VisionLLaMA-S is trained on ImageNet-1K for 800 epochs with a resolution of 224×224. VisionLLaMA-B is first trained for 800 epochs with an input size of 192× 192 and then fine-tuned for 20 epochs on 224 × 224. VisionLLaMA-L is first trained for 800 epochs with a resolution of 160 × 160 and then finetuned for 20 epochs on 224 × 224.

B.2. Supervised Training of Pyramid VisionLLaMA

We use the same setting as [12]. Specifically, all the models are trained on ImageNet-1K for 300 epochs with a global batch size of 1024 using the AdamW optimizer. The learning rate is increased to 0.001 within 5 warm-up epochs and decayed to zero following the cosine schedule. We use the same data augmentation as [12] and an image resolution of 224×224 for all models. To avoid overfitting, we use a weight decay of 0.05 and drop path [31] (0.2, 0.3, 0.5 for small base and large models respectively).

B.3. Mask Image Modeling on ImageNet

We use AdamW optimizer with momentum $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The global batch size is 4096. The initial learning rate is 1.5×10^-4 and decayed to zero within 800 or 1600 epochs. We also use 40 epochs to warm up the learning rate. We only use simple data augmentation RRC(random-resize-crops)as [25]. Besides, we use a weight decay of 0.05.

B.4. Linear Probing on ImageNet

We follow the setting of [25] and show the details in Table 15.

B.5. SFT on ImageNet for SSL-pre-trained Models

We follow the same setting of [25] and show the details in Table 16. The only modification is the layer-wise learning rate decay because we find 0.75 of [25] is overfitting for our method, and we set it to 0.45.

B.6. Segmentation for SSL-pre-trained Models

We follow the default setting of [12]. We use AdamW [44] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The global batch size is 16. The initial learning rate is 6×10^{-5} and linearly decayed to zero. We also use 1500 iterations to warm up. We also utilize l_2 weight decay of 0.05 and a drop-path rate [31] of 0.1.

B.7. Segmentation for Pyramid Transformers

We follow the setting of [12], which is almost the same as B.6. We use a drop-path rate of 0.2 for the pyramid VisionLLaMA-B model.

B.8. Object Detection of Pyramid Transformers

We use the same setting as [12]. We use the AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All the models are trained for 36 epochs with a global batch size of 16. The initial learning rate is 1×10^{-4} with 1000 iterations warm up and decayed by 10.0 at epoch 27 and 33. To avoid overfitting, we apply a l_2 weight decay for all models.

B.9. Object Detection of Plain Transformers

We use the AdamW optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The training resolution is fixed as 1024 × 1024 as [39]. Our model is trained for 36 epochs with a global batch size of 64. The initial learning rate is 1×10^{-4} with 1000 iterations warm up and decayed by 10.0 at epoch 27 and 33. We use a L_2 weight decay of 0.1. We also apply layer-wise learning rate decay with 0.7 as [39].

B.10. Image Generation of DiT-LLaMA

We use the same VAE as [53]. We make use of the AdamW optimizer with momentum $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use a global batch size of 256 across all models. The learning rate is fixed as 1×10^{-4}. The training resolution is 256 × 256. As for inference, we use 250 steps DDPM. We keep the default setting of ADM [21] without tuning. Specifically, we use $t_{max} = 1000$ linear schedule with β from 0.0001 to 0.02 and learnable variance σ_θ.

B.11. Image Generation of SiT-LLaMA

We use the same VAE as SD [53]. As for the ODE sampler, we utilize dopri5 and set atol and rtol to 1e-6 and 1e-3 respectively.

C. Architecture Setting

C.1. Pyramid VisionLLaMA

The detailed setting of the pyramid architecture is shown in Table 17.

C.2. Plain Transformer for Vision Understanding

The detailed setting of the architecture is shown in Table 18.

Model	Flops (G)	Params (M)	Training Steps (K)	Learning Rate	FID↓	sFID↓	Precision↑	Recall↑	IS↑
SiT-S/2 †	6.06	33	400	0.0001	59.60	9.16	39.41	58.48	23.32
SiT-LLaMA-S/2	6.06	33	400	0.0001	54.62	8.81	41.69	61.16	26.07
SiT-B/2 †	23.01	130	400	0.0001	36.90	6.70	51.00	64.10	39.78
SiT-LLaMA-B/2	23.01	130	400	0.0001	30.23	6.36	54.99	64.90	48.34
DiT-L/2	80.71	458	400	0.0001	23.3	-	-	-	-
SiT-L/2 †	80.71	458	400	0.0001	20.14	5.34	61.53	64.53	67.08
SiT-LLaMA-L/2	80.71	458	400	0.0001	14.91	5.16	64.97	64.30	82.23
SiT-XL/2 †	118.64	675	400	0.0001	17.83	5.13	63.52	64.61	73.64
SiT-LLaMA-XL/2	118.64	675	400	0.0001	12.79	5.02	66.77	64.37	90.93

Table 13. Image generation comparisons using the SiT framework [46] All the models are trained using an image resolution of 256×256 with a global batch size of 256. Metrics are calculated using the sampled 50k images without classifier-free guidance. IS: inception score. The FID is calculated by 250 steps ODE sampler because of the efficiency, which is a bit different from [46]. †: reproduced result using the released code.

	Sec B.2	Sec B.1
Batch size	1024	2048
Optimizer	AdamW	LAMB
LR	1.10^{-3}	3.10^{-3}
LR decay	cosine	cosine
Weight decay	0.05	0.02
Warmup epochs	5	5
Label smoothing ε	0.1	✗
Dropout	✗	✗
Stoch. Depth	✓	✓
Repeated Aug	✓	✓
Gradient Clip.	✗	1.0
H. flip	✓	✓
RRC	✓	✓
Rand Augment	9/0.5	✗
3 Augment (ours)	✗	✓
LayerScale	✗	✓
Mixup alpha	0.8	0.8
Cutmix alpha	1.0	1.0
Erasing prob.	0.25	✗
ColorJitter	✗	0.3
Test crop ratio	0.875	1.0
Loss	CE	BCE

Table 14. Training procedures with ImageNet-1K.

config	value
optimizer	LARS [78]
base learning rate	0.1
weight decay	0
optimizer momentum	0.9
batch size	16384
learning rate schedule	cosine decay
warmup epochs	10
training epochs	90
augmentation	RandomResizedCrop

Table 15. **Linear probing setting.**

config	value
optimizer	AdamW
base learning rate	1e-3
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.999
layer-wise lr decay [3]	0.45
batch size	1024
learning rate schedule	cosine decay
warmup epochs	5
training epochs	100 (B), 50 (L)
augmentation	RandAug (9, 0.5) [16]
label smoothing [63]	0.1
mixup [81]	0.8
cutmix [79]	1.0
drop path	0.1

Table 16. **End-to-end fine-tuning setting for SSL.**

C.3. Plain Transformer for Generation

The detailed setting of the architecture is shown in Table 19.

	Output Size	Layer Name	S	B	L
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	Patch Embedding	$P_1 = 4; C_1 = 64$	$P_1 = 4; C_1 = 96$	$P_1 = 4; C_1 = 128$
			$\begin{bmatrix} LSA \\ GSA \end{bmatrix} \times 1$	$\begin{bmatrix} LSA \\ GSA \end{bmatrix} \times 1$	$\begin{bmatrix} LSA \\ GSA \end{bmatrix} \times 1$
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	Patch Embedding	$P_2 = 2; C_2 = 128$	$P_2 = 2; C_2 = 192$	$P_2 = 2; C_2 = 256$
			$\begin{bmatrix} LSA \\ GSA \end{bmatrix} \times 1$	$\begin{bmatrix} LSA \\ GSA \end{bmatrix} \times 1$	$\begin{bmatrix} LSA \\ GSA \end{bmatrix} \times 1$
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	Patch Embedding	$P_3 = 2; C_3 = 256$	$P_3 = 2; C_3 = 384$	$P_3 = 2; C_3 = 512$
			$\begin{bmatrix} LSA \\ GSA \end{bmatrix} \times 5$	$\begin{bmatrix} LSA \\ GSA \end{bmatrix} \times 9$	$\begin{bmatrix} LSA \\ GSA \end{bmatrix} \times 9$
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	Patch Embedding	$P_4 = 2; C_4 = 512$	$P_4 = 2; C_4 = 768$	$P_4 = 2; C_4 = 1024$
			$\begin{bmatrix} GSA \end{bmatrix} \times 4$	$\begin{bmatrix} GSA \end{bmatrix} \times 2$	$\begin{bmatrix} GSA \end{bmatrix} \times 2$

Table 17. Configuration details of Pyramid VisionLLaMA.

Model	Layers	Dims	Heads
VisionLLaMA-S	12	384	6
VisionLLaMA-B	12	768	12
VisionLLaMA-L	24	1024	16

Table 18. Architecture settings for VisionLLaMA on image understanding tasks.

Model	Layers	Dims	Heads
DiT-LLaMA-S / SiT-LLaMA-S	12	384	6
SiT-LLaMA-B / SiT-LLaMA-B	12	768	12
SiT-LLaMA-L / SiT-LLaMA-L	24	1024	16
SiT-LLaMA-XL / SiT-LLaMA-XL	28	1152	16

Table 19. Architecture settings for VisionLLaMA on image generation.