1 Appendix

We provide more architecture and training details of the proposed HiT as well as additional experimental results to help better understand our paper.

1.1 Attention Module

The proposed multi-axis blocked self-attention module and cross-attention module follow the conventional design of attention modules presented in Transformers [4, 19], which consist of layer normalization (LN) [1] layers, feed-forward networks (we denoted as MLPs), and residual connections. To be specific, after we add trainable positional embedding vectors [17] \( p \) to the input \( x \), an attention module updates the input \( x \) following the rule:

\[
y = x + \text{Attention}(x', x', x'), \quad \text{where } x' = \text{LN}(x)
\]

\[
x = y + \text{MLP}((\text{LN}(y)), \quad (1)
\]

where MLP is a two-layer network: \( \max(0, xW_1 + b)W_2 + b \), and \( \text{Attention}(Q, K, V) \) denotes the query-key-value attention [2]. Replacing \( \text{Attention}(Q, K, V) \) in Equation (1) with the blocking operation and multi-axis self-attention leads to the proposed multi-axis blocked self-attention module, while using the latent embedding as \( K \) and \( V \) results in the proposed cross-attention module. It is also obvious that if we remove the attention operation in Equation (1), it can be rewritten as \( f(p;x) \), where \( p \) represents the coordinate of each element in the input and \( x \) encodes the condition. Hence, if we omit normalization operations, the function \( f(p;x) \) can be viewed as a form of conditional implicit neural functions [3, 12].

To further improve the model performance and efficiency, we make two modifications to the vanilla attention module in [4]. First, we replace the layer normalization (LN) [1] and batch normalization (BN) [7] for image generation. We find BN can not only achieve better IS and FID scores than LN, but also stabilize the training process. Second, we replace the multi-head attention (MHA) [19] with multi-query attention (MQA) [14]. MHA consists of multiple attention layers (heads) in parallel with different linear transformations on the queries, keys, values and outputs. MQA is identical except that the different heads share a single set of keys and values. We observe that incorporating MQA does not lead to degradation in performance but can improve the computational speed of the model. We report detailed results in Table 1 on ImageNet 128 × 128.

Table 1: Comparison with different architectural components of the proposed HiT on the ImageNet 128 × 128 dataset. **Left**: Layer normalization (LN) [1] and batch normalization (BN) [7]. **Right**: multi-head attention (MHA) [19] and multi-query attention (MQA) [14].

<table>
<thead>
<tr>
<th>Method</th>
<th>FID ↓</th>
<th>IS ↑</th>
<th>Method</th>
<th>Throughput ↑</th>
<th>FID ↓</th>
<th>IS ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>HiT w/ LN [1]</td>
<td>38.96</td>
<td>18.18</td>
<td>HiT w/ MHA [19]</td>
<td>74.73</td>
<td>32.06</td>
<td>20.92</td>
</tr>
</tbody>
</table>
Table 2: Detailed architecture specifications of HiT-\{S, B, L\} on CelebA-HQ \cite{9} and FFHQ \cite{10} for the output resolutions of 256 × 256 and 1024 × 1024. We denote cross-attention modules as \{\cdot\}, multi-axis attention modules as \{\cdot\}, and MPLs as \{\cdot\}. We also note that stages 1 to 4 are low-resolution stages and 5 to 8 are high-resolution stages.

<table>
<thead>
<tr>
<th>Output resolution</th>
<th>256 × 256</th>
<th>1024 × 1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input size</td>
<td>Hit-S</td>
<td>Hit-B</td>
</tr>
<tr>
<td>Stage 1 8</td>
<td>\begin{align*} \text{dim} &amp; 512, \text{head} 16 \times 1 \ \text{block sz.} &amp; 4 \times 4 \ \text{pixel shuffle,} &amp; 256-d \end{align*} &amp; \begin{align*} \text{dim} &amp; 512, \text{head} 16 \times 1 \ \text{block sz.} &amp; 4 \times 4 \ \text{pixel shuffle,} &amp; 256-d \end{align*}</td>
<td>\begin{align*} \text{dim} &amp; 512, \text{head} 16 \times 1 \ \text{block sz.} &amp; 4 \times 4 \ \text{pixel shuffle,} &amp; 256-d \end{align*}</td>
</tr>
<tr>
<td>Stage 2 16</td>
<td>\begin{align*} \text{dim} &amp; 128, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 64-d \end{align*} &amp; \begin{align*} \text{dim} &amp; 128, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 64-d \end{align*}</td>
<td>\begin{align*} \text{dim} &amp; 128, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 64-d \end{align*}</td>
</tr>
<tr>
<td>Stage 3 32</td>
<td>\begin{align*} \text{dim} &amp; 64, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 32-d \end{align*}</td>
<td>\begin{align*} \text{dim} &amp; 64, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 32-d \end{align*}</td>
</tr>
<tr>
<td>Stage 4 64</td>
<td>\begin{align*} \text{dim} &amp; 32, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 32-d \end{align*}</td>
<td>\begin{align*} \text{dim} &amp; 32, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 32-d \end{align*}</td>
</tr>
<tr>
<td>Stage 5 128</td>
<td>\begin{align*} \text{dim} &amp; 16, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 32-d \end{align*}</td>
<td>\begin{align*} \text{dim} &amp; 16, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 32-d \end{align*}</td>
</tr>
<tr>
<td>Stage 6 256</td>
<td>\begin{align*} \text{dim} &amp; 8, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 32-d \end{align*}</td>
<td>\begin{align*} \text{dim} &amp; 8, \text{head} 4 \times 1 \ \text{block sz.} &amp; 8 \times 8 \ \text{pixel shuffle,} &amp; 32-d \end{align*}</td>
</tr>
<tr>
<td>Stage 7 512</td>
<td>\begin{align*} \text{dimension} &amp; 1 \times 1 \ \text{linear,} &amp; 3-d \end{align*}</td>
<td>\begin{align*} \text{dimension} &amp; 1 \times 1 \ \text{linear,} &amp; 3-d \end{align*}</td>
</tr>
<tr>
<td>Stage 8 1024</td>
<td>\begin{align*} \text{dimension} &amp; 1 \times 1 \ \text{linear,} &amp; 3-d \end{align*}</td>
<td>\begin{align*} \text{dimension} &amp; 1 \times 1 \ \text{linear,} &amp; 3-d \end{align*}</td>
</tr>
</tbody>
</table>

1.2 Detailed Architectures

The detailed architecture specifications of the proposed HiT on CelebA-HQ \cite{9} and FFHQ \cite{10} are shown in Table 2 where an input latent code of 512 dimensions is assumed for all architectures. “pixel shuffle” indicates the pixel shuffle operation \cite{15} which results in an upsampling of the feature map by a rate of two. “256-d” denotes a linear layer with an output dimension of 256. “block sz. 4 × 4” indicates the blocking operation producing non-overlapping feature blocks, each of which has the resolution of 4 × 4. On the ImageNet 128 × 128 dataset, we use the same architecture of HiT-L as shown in Table 2 except that its last stage for the resolution of 256 × 256 is removed. We also reduce the dimension of the input latent code to 256 on this dataset.

1.3 Implementation Details

We use Tensorflow for implementation. We provide the detailed description about the generative process of the proposed HiT in Algorithm 1. The Tensorflow code for computing the proposed multi-axis attention is shown in Algorithm 2. Our code samples use einsum notation, as defined in Tensorflow, for generalized contractions between tensors of arbitrary dimension. The pixel shuffle \cite{15} operation is implemented by \texttt{tf.nn.depth_to_space} with a block size of two. Feature blocking and unblocking can be implemented by \texttt{tf.nn.space_to_depth} and \texttt{tf.nn.depth_to_space} with reshape operations, respectively. See Algorithm 3 for more details about blocking and unblocking.

1.4 Objectives of GANs

As stated in the main paper, our model is trained with a standard non-saturating logistic GAN loss with $R_1$ gradient penalty \cite{13}. In the original GAN formulation \cite{6}, the output of the discriminator $D$ is a probability and the cost function for the discriminator is given by the negative log-likelihood.
Algorithm 1 Generation process of HiT.

Define: \( X_l \) denotes the feature map in the \( l \)-th stage. \( P_X \) is the positional encoding of \( X \). Linear(\( \cdot \)) denotes a linear projection function. ReShape\(_{8 \times 8}(\cdot)\) denotes the operation to reshape the input into the output with the spatial resolution of \( 8 \times 8 \). UpSampleNN\(_{2 \times 2}(\cdot)\) denotes nearest neighbour upsampling with the factor of \( 2 \times 2 \). PixelShuffle\(_{2 \times 2}(\cdot)\) means pixel shuffle upsampling with the block size of \( 2 \times 2 \).

Input: the latent code \( z \), \# of low-resolution stages \( M \), and \# of high-resolution stages \( N \).

Output: The final image \( I \) with the target resolution.

\[
I \leftarrow 0 \quad \# \text{initialize the output image to zeros}
\]
\[
Z \leftarrow \text{ReShape}_{8 \times 8}(\text{Linear}(z)) + P_Z \quad \# \text{create the latent embedding with positional encoding}
\]

for \( 0 \leq l < M + N \) do

\[
X_l \leftarrow X_l + P_{X_l} \quad \# \text{add positional encoding to the feature map of the } l\text{-th stage}
\]

\[
X_l \leftarrow \text{MultiQueryAttention}(X_l, Z, Z) \quad \# \text{perform cross-attention – } X_l \text{ as } Q \text{ and } Z \text{ as } K, V
\]

if \( l < M \) then

\[
X_l \leftarrow \text{Block}(X_l) \quad \# \text{block the feature map into patches in low-resolution stages}
\]

\[
X_l \leftarrow \text{MultiAxisAttention}(X_l, X_l, X_l) \quad \# \text{perform multi-axis self-attention – } X_l \text{ as } Q, K, \text{ and } V
\]

\[
X_l \leftarrow \text{UnBlock}(X_l) \quad \# \text{unblock non-overlapping patches to the feature map}
\]

else

\[
X_l \leftarrow \text{MLP}(X_l) \quad \# \text{use only MLP in high-resolution stages}
\]

end if

\[
I \leftarrow \text{UpSampleNN}_{2 \times 2}(I) + RGB(X_l) \quad \# \text{upsample and sum the results to create the image}
\]

\[
X_{l+1} \leftarrow \text{Linear}(\text{PixelShuffle}_{2 \times 2}(X_l)) \quad \# \text{upsample and produce the input for the next stage}
\]
end for

Algorithm 2 Tensorflow code implementing Multi-Axis Attention.

```
def MultiAxisAttention(X, Y, W_q, W_k, W_v, W_o):
    # Multi-Axis Attention.
    X and Y are blocked feature maps where m is # of patches and n is patch sequence length.
    b is batch size; d is channel dimension; h is number of heads; k is key dimension; v is value dimension.
    Args:
    X: a tensor used as query with shape [b, m, n, d]
    Y: a tensor used as key and value with shape [b, m, n, d]
    W_q: a tensor projecting query with shape [h, d, k]
    W_k: a tensor projecting key with shape [d, k]
    W_v: a tensor projecting value with shape [d, v]
    W_o: a tensor projecting output with shape [h, d, v]
    Returns:
    Z: a tensor with shape [b, m, n, d]
    Q = tf.einsum("bmdk->bhmk", X, W_q)
    Q1, Q2 = tf.split(Q, num_or_size_splits=2, axis=1)
    K = tf.einsum("bdnk->bmnk", Y, W_k)
    V = tf.einsum("bdnv->bmnv", Y, W_v)
    logits = tf.einsum("bhxyk,bzyk->bhyxz", Q1, K)
    scores = tf.nn.softmax(logits)
    Q1 = tf.einsum("bhzyb->bhxv", scores, V)
    logits = tf.einsum("bhxk,bzkk->bhzx", Q2, K)
    scores = tf.nn.softmax(logits)
    Q2 = tf.einsum("bhzyb->bhzv", scores, V)
    scores = tf.nn.softmax(logits)
    Q1 = tf.einsum("bhzyb->bhxv", scores, V)
    0 = tf.concat([Q1, Q2], axis=1)
    Z = tf.einsum("bhmv,hdv->bhmd", 0, W_o)
    return Z
```
of the binary discrimination task of classifying samples as real or fake. Concurrently, the generator $G$ optimizes a cost function that ensures that generated samples have high probability of being real. The corresponding loss functions are defined as:

$$\mathcal{L}_D = -\mathbb{E}_{x \sim P_x} [\log(D(x))] - \mathbb{E}_{z \sim P_z} [\log(1 - D(G(z)))] + \gamma \cdot \mathbb{E}_{x \sim P_x} [\|\nabla_x D(x)\|_2^2],$$

$$\mathcal{L}_G = -\mathbb{E}_{z \sim P_z} [\log(D(G(z)))]$$

where $\gamma$ is the weight of $R_1$ gradient penalty \cite{Gretton2012} and we set it as 10 in the experiments. $R_1$ gradient penalty penalizes the discriminator from deviating from the Nash Equilibrium via penalizing the gradient on real data alone. We use these adversarial losses throughout our experiments.

### 1.5 Training Details of VQ-HiT

We explore using HiT as a decoder in the vector quantised-variational auto-encoder (VQ-VAE) \cite{van2018neural}. We use the same encoder $E$ as the one of VQGAN \cite{esser2021taming} while replace its ConvNet-based decoder $G$ with the proposed HiT using the HiT-B configuration in Table \ref{tab:config} for producing $256 \times 256$ images.

Apart from the reconstruction loss and GAN loss, we also introduce perceptual loss \cite{johnson2016perceptual} by using the feature extracted by VGG \cite{simonyan2014very} following \cite{esser2021taming, rombach2017flying}. Hence, the training process of VQ-HiT can be formulated as:

$$\mathcal{L}_D = -\mathbb{E}_{x \sim P_x} [\log(D(x))] - \mathbb{E}_{z \sim P_z} [\log(1 - D(G(z)))] + \gamma \cdot \mathbb{E}_{x \sim P_x} [\|\nabla_x D(x)\|_2^2],$$

$$\mathcal{L}_{VAE} = \|x - G(E(x))\|_2^2 + \lambda_1 \cdot \|F(x) - F(G(E(x)))\|_2^2 - \lambda_2 \cdot \mathbb{E}_{z \sim P_z} [\log(D(G(E(x))))],$$

where $x$ is the input image to be reconstructed, $\lambda_1$ and $\lambda_2$ are the perceptual loss weight and discriminator loss weight, and $F(\cdot)$ denotes the VGG feature extraction model. We set $\lambda_1 = 5 \times 10^{-5}$ and $\lambda_2 = 0.1$ in the experiments. Adam \cite{kingma2014adam} is utilized for optimization with $\beta_1 = 0$ and $\beta_2 = 0.99$. The learning rate is 0.0001 for both the auto-encoder and discriminator. We set the mini-batch size to 256 and train the model for 500,000 iterations.
1.6 More Ablation Studies

We implement two additional variants of HiT where the model uses only blocked axial attention and Nested Transformer [20], respectively. The results on the ImageNet 128 × 128 dataset are shown in Table 3. We can observe that using only blocked multi-axis attention can still lead to significant performance improvement which demonstrates its effectiveness. We also find that incorporating cross-attention and multi-axis attention as in HiT yields much better performance than vanilla Nested Transformer.

Table 3: Comparison with different variants of the proposed HiT on the ImageNet 128 × 128 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>FID  ↓</th>
<th>IS ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (INR)</td>
<td>56.33</td>
<td>16.19</td>
</tr>
<tr>
<td>Nested Transformer [20]</td>
<td>42.52</td>
<td>18.68</td>
</tr>
<tr>
<td>HiT w/ only blocked multi-axis attention</td>
<td>35.43</td>
<td>19.75</td>
</tr>
<tr>
<td>HiT w/ only cross-attention</td>
<td>35.94</td>
<td>19.42</td>
</tr>
<tr>
<td>HiT (Full model)</td>
<td>31.87</td>
<td>21.32</td>
</tr>
</tbody>
</table>

1.7 More Qualitative Results

We include more visual results that illustrate various aspects related to generated image quality of HiT. Figure 1 shows uncurated results on the ImageNet 128 × 128 dataset. We randomly sample from the ConvNet baseline and the proposed HiT as qualitative examples for a side by side comparison. From the results, we can see that the samples generated by HiT exhibit much more diversities in object category, texture, color, and image background than the ConvNet baseline. Figure 2 shows additional hand-picked synthetic face images illustrating the quality and diversity achievable using our method on the CelebA-HQ 256 × 256 and 1024 × 1024 dataset.

Figure 1: Uncurated ImageNet 128 × 128 samples from ConvNet-R1 (left, FID: 37.18, IS: 19.55) and the proposed HiT (right, FID: 30.83, IS: 21.64). Results are randomly sampled from batches.

Figure 2: Additional synthetic face images by HiT-B on CelebA-HQ 1024 × 1024 and 256 × 256.
Interpolation. We conclude the visual results with the demonstration of the flexibility of HiT. As well as many other generators, the proposed HiT generators have the ability to interpolate between input latent codes with meaningful morphing. Figure 3 illustrates the synthetic face results on the CelebA-HQ $256 \times 256$ dataset. As expected, the change between the extreme images occurs smoothly and meaningfully with respect to different facial attributes including gender, expression, eye glass, and view angle.

Figure 3: Latent linear morphing on the CelebA-HQ $256 \times 256$ dataset between two synthetic face images – the left-most and right-most ones. HiT-B is able to produce meaningful interpolations of facial attributes in terms of gender, expression, eye glass, and view angle (from top to bottom).

References


