

A DOT PRODUCT ATTENTION FREE TRANSFORMER

Anonymous authors

Paper under double-blind review

ABSTRACT

We introduce Dot Product Attention Free Transformer (DAFT), an efficient variant of Transformers (Vaswani et al., 2017) that eliminates the query-key dot product in self attention. The core idea is to construct a decomposable attention map for each dimension of the query, key and value. This compositionality enables an implementation where the attention tensor does not to be computed or stored explicitly. A DAFT layer has a memory complexity linear w.r.t. both the context size and the dimension of features, making it compatible with both large input and model sizes. We also introduce DAFT-conv, a model variant that takes advantage of locality and spatial weight sharing while maintaining global connectivity. We conduct experiments on ImageNet-1K classification, as well as CIFAR10 and Enwik8, two autoregressive modeling tasks. We show that DAFT demonstrates competitive performance on all the benchmarks, while providing excellent efficiency at the same time.

1 INTRODUCTION

Self attention mechanisms, represented by Transformers (Vaswani et al., 2017), have driven the advancement of various machine learning problems, including language understanding (Devlin et al., 2018; Radford et al.) and computer vision applications (Chen et al.; Dosovitskiy et al., 2020; Touvron et al., 2020). Different from classic model architectures such as Convolutional Neural Nets (CNNs) or Recurrent Neural Nets (RNNs), Transformers enable direct interaction between every pair of elements within a sequence, which makes them especially powerful at capturing long term dependencies.

Transformers build on top of Multi-head Attention (MHA), which performs dot product attention on an input h times and concatenates the outputs along the channel dimension. This introduces a space complexity of $O(hT^2)$, where T is the input sequence length. It is thus challenging for Transformers to scale to long sequences and large model size at the same time. A number of recent works have been dedicated to addressing the scalability issue of Transformers (Child et al., 2019; Kitaev et al., 2020; Rae et al., 2020; Wang et al., 2020b; Katharopoulos et al., 2020; Tay et al., 2020a; Choromanski et al., 2020). The common idea here is to approximate the full attention operation, with the techniques ranging from sparsity (Child et al., 2019; Zaheer et al., 2020), locality sensitive hashing (Kitaev et al., 2020), low rank decomposition (Wang et al., 2020b), kernel approximation (Katharopoulos et al., 2020; Choromanski et al., 2020), etc..

In this paper, we propose a computational module that does not use or approximate the standard dot product attention. We hence name our model **Dot product Attention Free Transformer (DAFT)**. DAFT explores two simple ideas **1)** decrease the computation of an attention head by making it additive and decomposable **2)** increase the number of heads to be the same as the input dimension with minimal cost. In our implementation, DAFT rearranges the computational graph such that the key&value are multiplied element-wise, reduced along the time dimension (with a set of learned pairwise position biases), and finally multiplied with the query element-wise. This removes the explicit computation of the attention matrices, which results in a space complexity of $O(Td)$.

A closely related line of work to DAFT is the “linear attention” Transformers (LAT) (Katharopoulos et al., 2020; Choromanski et al., 2020; Peng et al., 2021; Bello, 2021). LAT can be interpreted as a special case of dot product attention, where the nonlinearity is changed from exp to identity. This modification allows for a similar rearrangement of the computation, where the context reduction of query&key happens before interacting with the query. There are two important differences

Table 1: Comparison for DAFT with standard Transformer and linear attention Transformers: Linear Transformer (Katharopoulos et al., 2020), Performer (Choromanski et al., 2020). Here T, d denote the sequence length and feature dimension, respectively. DAFT has linear space complexity wrt both T and d , while not performing query&key dot product, keeping the exp nonlinearity, and not computing the explicit attention. Both Linear Transformer and Performer have at least quadratic complexity in d , perform query&key dot product but without the exp nonlinearity, and do not need to compute explicit attention.

Model	Time	Space	Dot Product?	exp Nonlinearity?	Explicit Attention?
Transformer	$O(T^2 d + T d^2)$	$O(T^2 + T d)$	✓	✓	✓
Linear Transformer	$O(T d^2)$	$O(T d + d^2)$	✓	✗	✗
Performer	$O(T d^2 \log d)$	$O(T d \log d + d^2 \log d)$	✓	✗	✗
DAFT	$O(T^2 d + T d^2)$	$O(T d)$	✗	✓	✗

between DAFT and LAT: 1) DAFT maintains the same exp nonlinearity as standard MHA, which LAT gets rid of. Empirically, the removal of the exp nonlinearity often causes performance degradation. 2) LAT introduces a space complexity of $O(T d d')$, where d' is the dimension of query, key projections. This makes LAT expensive for large model sizes. DAFT on the other hand enjoys a space complexity of $O(T d)$, friendly to both large inputs and large models. See Table 1 for the summarized comparison.

In addition to the basic DAFT formulation, we also propose DAFT-conv, a variant that exploits locality and weight sharing in the position biases. DAFT-conv inherits the benefit of both Transformers and ConvNets. It is wired similar to a Transformer, maintaining the global connectivity and multiplicative interactions among the query, key and value; it also enjoys the parameter efficiency, sparse computation and translational equivariance property of ConvNets. We show that DAFT-conv provides excellent performance and efficiency gain over the basic version on all tasks we tested.

We perform experiments with DAFT on image classification, image auto-regressive modeling and character level language modeling tasks. We show that DAFT provides competitive performance, often matching or beating standard Transformers and other variants, while providing excellent efficiency. We also provide ablation studies to several design choices of DAFT, and discuss its unique properties such as compatibility with Transformers, robustness to the choice of kernel size and variable sized inputs.

2 BACKGROUND

2.1 MULTI-HEAD ATTENTION

We now introduce the Multi-Head Attention (MHA) operation in the mode of self attention. We denote an input sequence as $X \in R^{T \times d}$, with T, d, h being the sequence length, the feature dimension and the number of heads h , respectively. We use the superscript i to denote the i th attention head, and subscript t to denote the t th location. MHA performs a scaled dot product attention for each head i , defined as:

$$Y_t^i = \frac{\sum_{t'=1}^T \exp(Q_t^{i\top} K_{t'}^i) V_{t'}^i}{\sum_{t'=1}^T \exp(Q_t^{i\top} K_{t'}^i)}, \text{ s.t. } Q^i = X W_Q^i, K_i = X W_K^i, V^i = X W_V^i. \quad (1)$$

Here $W_Q^i \in R^{d \times d_k}$, $W_K^i \in R^{d \times d_k}$, $W_V^i \in R^{d \times d_v}$ are linear transformations for head i ; $Y_t^i \in R^{d_v}$ is output of the i th attention head for the t th query location. d_k, d_v are dimensions for key and value, respectively. MHA concatenates the output of h attention heads along the channel dimension, resulting in feature dimension $h d_v$. Unless otherwise mentioned, we assume $d_k = d_v$ and $h = \frac{d}{d_k}$. This means the query, key and value are the same dimension within each head, and the output dimension matches that of the input.

2.2 LINEAR ATTENTION

One can remove the \exp nonlinearity in Eq. 1 to give rise to the linear attention used in (Katharopoulos et al., 2020; Choromanski et al., 2020). This special form of dot product attention enables one to rearrange the order of computation as:

$$Y_t^i = \frac{\sum_{t'=1}^T (Q_t^{i\top} K_{t'}^i) V_{t'}^i}{\underbrace{\sum_{t'=1}^T Q_t^{i\top} K_{t'}^i}_{\text{explicit linear dot product attention}}} = \frac{Q_t^{i\top} \sum_{t'=1}^T K_{t'}^i V_{t'}^i}{\underbrace{Q_t^{i\top} \sum_{t'=1}^T K_{t'}^i}_{\text{implicit attention}}}. \quad (2)$$

Here we allow Q and K to have dimension $R^{T \times d'}$, where d' can be selected independently of d (e.g., in Performer (Choromanski et al., 2020), it is recommended to let $d' = O(d \log d)$); we also assume that Q, K contain additional linear projections and nonlinearities when required.

3 METHODOLOGY

3.1 DOT PRODUCT ATTENTION FREE TRANSFORMER

We now define Dot product Attention free transformer (DAFT), which is a plugin replacement of MHA without the need of changing other architectural aspects of Transformers. The basic idea is to simplify the dot product attention to a form of additive attention (Bahdanau et al., 2014) that amounts to an efficient implementation. We start by considering an attention head defined in the following way:

$$att_{t,t'}^i \propto \exp(Q_t^i + K_{t'}^i + w_{t,t'}), \quad i = 1, 2, \dots, d, \quad (3)$$

where $w \in R^{T \times T}$ is a set of learned pairwise position biases. Here the attention score (before \exp) becomes an additive composition of the query, the key and a “static attention” score (w). By doing so, we are also automatically making the number of attention heads equal to the feature dimension. One can then interpret this new formulation as trading the expressivity of a single attention head for more heads. However, because of the normalizing mechanism of attention, the query term cancels out from the numerator and denominator. We thus make a slight modification and make it:

$$att_{t,t'}^i = \frac{\sigma_q(Q_t^i) \exp(K_{t'}^i + w_{t,t'})}{\sum_{t'=1}^T \exp(K_{t'}^i + w_{t,t'})}, \quad i = 1, 2, \dots, d, \quad (4)$$

where σ_q is a nonlinearity defaulting to *sigmoid*. By doing so, we take Q_t^i out of the denominator, which makes it similar to the role of an output gate in LSTMs Hochreiter & Schmidhuber (1997). We can now define the full computation of DAFT as:

$$Y_t = \sigma_q(Q_t) \odot \frac{\sum_{t'=1}^T \exp(K_{t'} + w_{t,t'}) \odot V_{t'}}{\sum_{t'=1}^T \exp(K_{t'} + w_{t,t'})}, \quad (5)$$

where \odot is the element-wise product, $Q_t, K_{t'}, V_{t'} \in R^d$.

Efficient Implementation. The question now remains to be answered is how to implement DAFT in an efficient way. It is obvious that explicitly computing the attention in Eq. 4 as done in Transformers is infeasible, as it requires storing a tensor of size $T^2 d$. Luckily, the compositionality of DAFT allows for a rearrangement of computation similar to that in linear attention. In essence, we take advantage of the simple equality of $\exp(x + y) = \exp(x) \exp(y)$, and decompose the computation into a series of element-wise multiplication/division and matrix-tensor multiplication.

In addition, a naive parameterization of w is parameter intensive. We thus adopt a factorized form of w as:

$$w = pe_0 pe_1^\top, \quad s.t. \quad pe_0 \in R^{T \times n}, pe_1 \in R^{T \times n}, \quad (6)$$

where n is a small embedding dimension (e.g., 128). This simple factorization not only greatly reduces the parameter counts ($2Tn$ vs T^2), but also empirically improves model’s performance in both training and testing.

We denote this version as **DAFT-full**, and show the PyTorch-style pseudo code in Algorithm 1.

Algorithm 1 Pseudo code of DAFT-full attention layer in a PyTorch-like style.

```

def daft_attn(q, k, v, pe0, pe1):
    # B: batch size; T: sequence length; d feature dimension.
    # q, k, v: query, key and value each of shape (B, T, d)
    # pe0, pe1: position embeddings of shape (T, n)
    # y: output of shape (B, T, d)
    q = q.sigmoid()
    k = k.exp()
    kv = k * v
    w = pe0.mm(pe1.T) # compute pairwise position bias (T, T)
    w = w.exp()
    numerator = einsum('lt,btd->bld', w, kv) # context aggregation
    denominator = einsum('lt,btd->bld', w, k)
    y = q * numerator / denominator # normalization and output gating
return y

```

Algorithm 2 Pseudo code of DAFT-conv attention layer in a PyTorch-like style.

```

def daft_conv_attn(q, k, v, r):
    # B: batch size; T: 1d sequence length; (H, W): 2d input size
    # s: half of window size; d feature dimension.
    # q, k, v: query, key and value each of shape (B, d, T) or (B, d, H, W)
    # r: depth-wise separable convolutional filters of shape (d, 2s+1) or (d, 2s+1, 2s+1)
    # y: output of shape (B, d, T) or (B, d, H, W)
    q = q.sigmoid()
    k = k.exp()
    kv = k * v
    r = r.exp() - 1 # assuming r is already reparameterized as Eq. (9)
    numerator = conv1d(kv, r, groups=d, padding='same') + kv.sum(dim=2, keepdim=True)
    # for 2d inputs
    numerator = conv2d(kv, r, groups=d, padding='same') + kv.sum(dim=[2,3], keepdim=True)
    denominator = conv1d(k, r, groups=d, padding='same') + k.sum(dim=2, keepdim=True)
    # for 2d inputs
    denominator = conv2d(k, r, groups=d, padding='same') + k.sum(dim=[2,3], keepdim=True)
    y = q * numerator / denominator # normalization and output gating
return y

```

3.2 DAFT-CONV

In DAFT-full, w encodes to the static bias between two positions. It is thus straightforward to conceive a “multi-head” version where one learns h sets of position biases $\{w_i\}_{i=1\dots h}$, each interacting with a subset of feature dimensions in the same way as in MHA. However, as w is learned as free parameters, increasing h to large values poses immediate challenge to the parameter efficiency.

We propose a simple solution to this dilemma, where we simultaneously make w_i local and share parameters across positions. To be more exact, we exploit the native structure of the input (1d or 2d), and assign a non-zero value to a position pair (t, t') based on their relative position, if they are within a specified window size $2s + 1$ (assuming the window size is an odd number). In the 1d case (the 2d case can be derived similarly), this becomes

$$w_{t,t'}^i = \begin{cases} r_{t'-t+s}^i, & \text{if } |t - t'| \leq s \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

Here $r^i \in R^{2s+1}$, $i = 1, 2, \dots, h$ are the relative position vectors¹.

We can now introduce **DAFT-conv**, which is a multi-head, convolutional version of DAFT. DAFT-conv can be implemented with the **convolution** primitive as (again using 1d convolution as an example):

$$Y_t^i = \sigma_q(Q_t^i) \odot \frac{\text{dw-conv}(\exp(K^i) \odot V^i, \exp(r^i) - 1) + \sum_{t'=1}^T \exp(K_{t'}^i) \odot V_{t'}^i}{\text{dw-conv}(\exp(K^i), \exp(r^i) - 1) + \sum_{t'=1}^T \exp(K_{t'}^i)}, i = 1, 2, \dots, h. \quad (8)$$

here h is the number of heads; $\text{dw-conv}(x, r)$ denotes the depth-wise separable 1d convolution with $x \in R^{T \times d}$ being the input signal and $r \in R^{(2s+1)}$ being the convolutional filter, which are convolved along the first dimension with “same” padding.

¹One can construct the pairwise matrix $w^i \in R^{T \times T}$ by tiling r^i along the spatial dimension.

Compared with DAFT-full, DAFT-conv provides both parameter and computational efficiency. Moreover, DAFT-conv also enjoys the same locality and translational equivariance inductive bias of ConvNets, which have been proven useful in many domains. There is also a distinctive property of DAFT-conv, which is that **global connectivity** between any two positions is maintained regardless of the selection of the window size. We verify experimentally that this enables DAFT-conv to achieve strong performance even with very small window sizes.

Implementation. During training, we adopt a re-parameterization of the convolutional filters r , where for each head i , we let

$$r^i = \gamma^i \frac{r^i - \text{mean}(r^i)}{\text{std}(r^i)} + \beta^i, \quad (9)$$

where $\gamma \in R^d, \beta \in R^d$ are learnable gain and bias parameters, both initialized as 0. We found that this accelerates learning and gives consistent boost to the performance. We show the PyTorch-style pseudo code in Algorithm 2, where we assume $h = d$ for simplicity.

4 RELATED WORK

Since the Transformer was introduced, there have been numerous attempts to address the major source of inefficiency in the architecture, the quadratic cost of the attention operation. Improving this operation can enable larger context sizes and more efficient implementations. For a comprehensive, recent survey of efficient transformers, see (Tay et al., 2020c).

Approximating the dot product. Katharopoulos et al. (2020); Choromanski et al. (2020); Peng et al. (2021) propose to approximate the exponential kernel with inner product of projections, which leads to a linearized attention operation of complexity $O(Td^2)$. The d^2 term of these models however makes it difficult to scale with model size, which is not a problem for DAFT. Reformers (Kitaev et al., 2020) apply LSH as an approximation to the dot product, where DAFT completely gets rid of it.

Sparse, local attention. Sparse Transformers (Child et al., 2019) and Image Transformer (Parmar et al., 2018) proposes to use fixed sparse or local context patterns. Attention models in vision tasks (often combined with convolutions) use image structure to help handcraft relevant spatial patterns to attend (Wang et al., 2020a; Huang et al., 2019b; Zhu et al., 2019; Huang et al., 2019a; Ramachandran et al., 2019). DAFT-conv also borrows the locality idea, but we put it as a bias rather than hard constraint. This allows DAFT-conv to take advantage of the full context, rather than relying only on a subset.

Context compression. Other approaches try to learn context patterns. Adaptive-Span Transformers (Sukhbaatar et al., 2019) learn a range for each attention head within which to attend. Routing transformers (Roy et al., 2020) use clustering to compute dot-product attention only over a subset of elements within the same cluster. The Linformer (Wang et al., 2020b) reduces the length of the context by compressing the keys and values with a linear layer. Compressive Transformers (Rae et al., 2020) compute and update reduced representations of the input that are far enough back in the input sequence, and attend to those compressed representations. DAFT is largely complementary to these approaches, as our focus is to improve the complexity of any given sequence from the operation level.

Eliminating dot product attention. Instead of limiting the number of comparisons, other methods change the operation used to compute attention. The Synthesizer (Tay et al., 2020a) uses attention weights predicted from inputs, rather than derived from dot-product interactions. The LightConv module introduced in (Wu et al., 2019) proposes to replace the dot product self-attention with dynamic lightweight depthwise convolution, where the weights are normalized across temporal dimension. The Sinkhorn Transformer (Tay et al., 2020b) uses a differentiable sorting operation to identify relevant comparisons that may not be local in the original sequence order. DAFT offers a different approach along this line, while highlighting strong empirical performance and efficiency.

MLPs for vision. Recent works (Tolstikhin et al., 2021; Liu et al., 2021) explore the use of MLP in place of the attention operation for vision tasks. While DAFT can be viewed in a similar way, it is also equipped with a more sophisticated gating mechanism. In particular, the weighting of values are composed of both the key and position biases, which are normalized to non-negative values

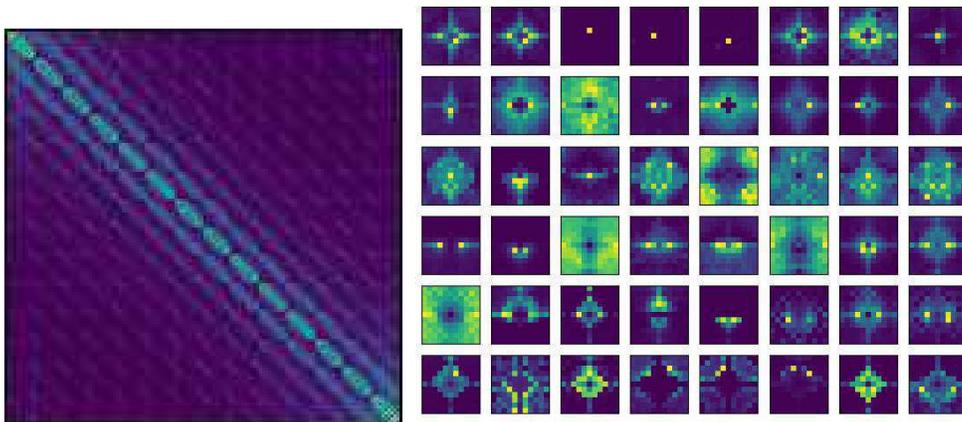


Figure 1: **Left:** The 197×197 position bias from the 3rd layer of DAFT-full. **Right:** relative position biases learned by a DAFT-conv. Each row represents a layer (with layer index ranging from $\{0, 2, 4, 6, 8, 10\}$); Each column represents a head.

Table 2: Imagenet 1K classification results with the Transformer architecture from DeiT (Touvron et al., 2020), crop size is 224. Speed and memory consumption are measured in inference mode on a V100 GPU, batch size is 256.

Model	Kernel	Heads	Epochs	Top1 Acc	#Params (MB)	Images/Sec	Mem (GB)
ResNet50	3	-	90	76.9	25.6	1257	6.5
DeiT-S	-	6	300	79.9	22.1	1010	2.9
MLP Mixer	-	-	300	75.0	18.6	1033	2.4
Linear Trans.	-	6	300	74.9	22.1	1089	2.6
Performer	-	6	300	79.0	22.1	1024	2.9
DAFT-full	-	1	300	79.8	22.6	1011	2.6
DAFT-conv	11	16	300	80.2	20.3	989	2.5
DAFT-conv	11	384	300	80.8	23.0	936	3.2
DAFT-conv	11	384	200	80.1	23.0	936	3.2

(similar to attention). This allows DAFT to be a plugin module to existing Transformers without any architectural changes and extra tuning. Besides, DAFT-conv inherits the valuable properties of CNNs, allowing it to achieve excellent parameter efficiency, strong performance as well as ability to handle variable sized inputs.

5 EXPERIMENTS

We conduct experiments on three tasks: image classification (Sec. 5.1), image autoregressive modeling (Sec. 5.2) and character level language modeling (Sec. 5.3). All the experiments are designed in the plug and play fashion, where we obtain a baseline Transformer architecture for the specific task and replace the attention module with an DAFT module. Hyperparameters such as initialization, learning rate scheduling are also directly inherited from the Transformer counterparts. Unless otherwise mentioned, all experiments are conducted on $8 \times V100$ GPU machines.

5.1 IMAGE CLASSIFICATION

We first test the encoder version of DAFT, focusing on an image classification task. We adopt the Vision Transformer architecture (Dosovitskiy et al., 2020), and perform experiments on the Imagenet 1K classification dataset. We adopt training setting and hyper parameters (batch size, data augmentation, regularization and learning rate scheduling) from DeiT (Touvron et al., 2020).

In a nutshell, A ViT splits an image into 16×16 non-overlapping patches, then linearly projects each patch with shared weights to the equivalence of token embeddings. A learned class token is appended to the resulting representation, resulting in a sequence of length $T = 1 + \frac{H/16}{W/16}$. A

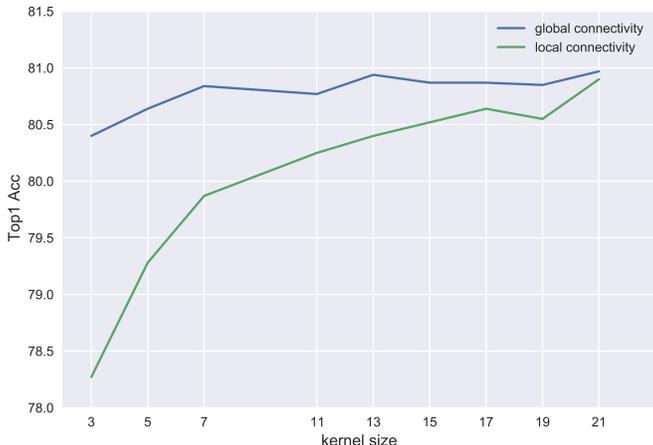


Figure 2: Effectiveness of the global connectivity of DAFT-conv, with varied kernel size on ImageNet. DAFT-conv’s performance is robust to the kernel size. When removing the global connectivity, small kernel sizes lead to severe performance degradation.

linear classification head is attached to the final layer’s class token to obtain the final outputs. See (Dosovitskiy et al., 2020) for more details of the model configuration. All the experiments are conducted on the ImageNet-1K dataset, without using extra data.

Since the sequence size is relatively small in this task ($T = 197$ for input sizes of 224×224), we first experiment with DAFT-full. The hidden dimension of factorized position bias is set as $n = 128$. We also experiment with DAFT-conv. In this setting, we remove the use of position embedding and class token, and apply global average pooling after the final layer’s output, which is then fed into the classification linear layer. This modification not only simplifies the model design, but also makes DAFT-conv **fully convolutional**.

We adopt the DeiT “small” (L=12, d=384, h=6, denoted as DeiT-S) configuration as the baseline. We also compare with MLP-Mixer (Tolstikhin et al., 2021), where we replace the attention layer with an MLP of hidden layer size $D_S = 4T$. This roughly matches the number of parameters in the corresponding MHA layer. We also compare with two linear attention models, Linear Transformer (Katharopoulos et al., 2020) and Performer (Choromanski et al., 2020). We keep the Q, K dimension the same as that in DeiT-S in both cases, and adopt the $1+elu$ and $relu$ nonlinearity, as recommended in the respective papers.

Our result is shown in Table 2. We first see that DAFT-full achieves comparable performance with the baseline Transformer DeiT-S, while with better memory footprint and similar speed. DAFT-conv significantly improves the top-1 accuracy of both configurations, with similar or smaller parameter counts. DAFT also outperforms Linear Transformer and Performer wrt the top 1 accuracy. Empirically, we also observe that DAFT has faster convergence than Transformers. In fact, we are able to match the performance of DeiT-S by training a DAFT-conv model for only $\frac{2}{3}$ of the total epochs.

Visualization. We also tried to visualize the position biases ($\exp(w) - 1$ to be precise) learned by DAFT-full and DAFT-conv, as shown in Figure 1. Note that interesting local, symmetric sparse patterns emerge. We show in the Appendix that we can regularize the position biases to achieve more sparsity. We also show an extreme version of DAFT-conv, where each head is assigned one non-zero context points, while still keep good accuracy. This effectively transforms convolution into indexing.

Variable size inputs. DAFT-conv is fully convolutional, which means that it can handle an input size different from that in training. We tested a DAFT-conv model (second to last row of Table 2, trained with crop size 224) on a larger crop size of 384. This results in an improved accuracy of 81.6, compared with the original 80.8. This makes DAFT-conv well suited for the pretraining finetuning workflows, as often seen in Vision tasks.

Table 3: Finetuning DAFT-conv for 100 epochs from a pretrained “DeiT base” on 384×384 crops. “ft” and “rand” stand for finetuning and random initialization, respectively. See the Appendix for more details.

Model	Kernel	Heads	Top1 Acc	#Params (MB)	Images/Sec	Mem (GB)
DeiT base	-	12	82.9	86.9	89.6	13.6
DAFT-conv ft	25	32	83.4	79.7	98.5	8.9
DAFT-conv rand	25	32	81.6	79.7	98.5	8.9

Table 4: NLL results on CIFAR10, evaluated by bits/dim, the lower the better. Speed and memory are measured during training time, with a batch size of 32 across 8 V100 GPUs.

Method	L	d	h	Train loss	Test loss	Iters/Sec	GB/GPU
PixelCNN	-	-	-	3.08	3.14	-	-
PixelSNAIL	-	-	-	-	2.85	-	-
Sparse Transformer strided	128	256	2	-	2.80	-	-
Image Transformer local2d	12	512	4	-	2.90	1.61	22.3
Transformer	12	512	4	2.90	2.88	1.35	30.6
Synthesizer	12	512	4	2.79	2.91	1.45	23.0
Reformer	12	512	4	2.88	2.93	1.41	14.3
DAFT-full	12	512	1	2.78	2.89	1.68	10.2
DAFT-conv-256	12	512	512	2.73	2.81	1.63	9.0

Compatibility with Transformers. Although DAFT is not designed to directly approximate MHA, they do share considerable similarity in that the value vectors are aggregated with learned non-negative weighting in both models. We hypothesize that representations learned by one model can be transferred to another. To test this, we obtain a pretrained “DeiT base” model with crop size 384. We then train an DAFT-conv by initializing its weights with that of the DeiT model, excluding the position embeddings, the class token, key and query projections. We use a batch size of 64 and train the model for 100 epochs. As a control, we also train a randomly initialized DAFT-conv for the same number of epochs. The results are shown in Table 3. Interestingly, we see that the finetuned version of DAFT-conv achieves significantly higher accuracy than that randomly initialized version. The resulting model is also more accurate, faster and memory efficient than the original DeiT model.

Global connectivity. DAFT-conv maintains global connectivity regardless of the local kernel size, which is distinctive from sparse and local attention works. To see the benefit of this design, we trained a degenerate variant of DAFT-conv, where we modify Equation 7 to assign $-\infty$ values to $w_{t,t'}$ outside the local window (zero weights after exponentiation). We then train a set of DAFT-conv models by varying the kernel size, the results are shown in Fig. 2. We see that DAFT-conv’s performance is extremely robust to the choice of kernel-size. Without the global connectivity, small kernel sizes lead to severe performance degradation.

5.2 IMAGE AUTOREGRESSIVE MODELING

In the next set of experiments, we consider the problem of image autoregressive modeling by minimizing the negative log likelihood (NLL). Similar to (Parmar et al., 2018), we represent an RGB image as a sequence of length $H \times W \times 3$, with H, W being the height and width, respectively. Each sub-pixel is represented as a 256-way discrete variable. We use CIFAR10 as the benchmarking dataset.

Our reference Transformer design largely follows that of (Chen et al.), which follows a 12 layer, 512 dimensional and 4 heads design. We use AdamW (Loshchilov & Hutter, 2019), and follow a standard warmup learning rate schedule as in (Vaswani et al., 2017). We use an initial learning rate of 1×10^{-3} a weight decay of 0.1 applied to all linear transformations weights, and a dropout rate of 0.1. We adopt simple data augmentation. During training, we first randomly flip each image horizontally, then add or subtract a value in the range $[-10, 10]$ from all its subpixels, and clip resulting pixel values to $[0, 255]$. We use cross entropy loss, and a default batch size of 32 for 100 training epochs.

We train two DAFT models: DAFT-full with factorized position bias (hidden dimension = 256); and DAFT-conv with 1d kernel size of 256. Note that in this case, the 1d DAFT-conv does not take

Table 5: Enwik8 results, measured in bits per character (bpc), the lower the better. Baselines compared are Reformer (Kitaev et al., 2020), Synthesizer (Tay et al., 2020a) (its best performing dense version), Linear Transformer (Katharopoulos et al., 2020) and Performer (Choromanski et al., 2020). L, d, h, T denote number of blocks (depth), dimension of features, number of heads, and sequence length, respectively. Speed and memory are measured during training time, with a batch size of 128 on a 8 V100 GPU node. Both Linear Transformer and Performer are implemented with customized CUDA kernels (github.com/idiap/fast-transformers), and all other models are implemented in native Pytorch.

Method	L	d	h	T	Train bpc	Test bpc	Iters/Sec	GB/GPU
Transformer	12	512	8	1024	0.977	1.137	1.42	29.4
Reformer	12	512	8	1024	1.04	1.195	1.05	20.9
Synthesizer	12	512	8	1024	0.994	1.298	1.49	29.9
Linear Transformer	12	512	8	1024	0.981	1.207	1.46	10.6
Performer	12	512	8	1024	1.002	1.199	1.44	10.1
DAFT-full	12	512	1	1024	0.88	1.201	1.85	11.3
DAFT-conv-64	12	512	512	1024	0.854	1.180	1.77	11.1

advantage of the true structure of pixels (which is 2d). It’s locality and parameter sharing mainly serve as a way of saving computation.

Comparing with the state of the art. CIFAR10 is a crowded benchmark for image autoregressive modeling, and we compare with a few competitive baselines, as shown in Table 4. Note that CIFAR10 has an unrolled sequence length of 3072, which is already prohibitive to train a full Transformer with reasonable size. Another baseline is Image Transformer (Parmar et al., 2018), which restricts attention to local2d windows of size of 256. We also compare to Synthesizer (Tay et al., 2020a) and Reformer (Kitaev et al., 2020).

From Table 4, we see that DAFT-full yields similar performance to the baseline Transformer, and outperforming Image Transformer, Synthesizer and Reformer. Interestingly, DAFT-conv provides even stronger performance than DAFT-full, although not utilizing the 2d local structure of pixels. Efficiency wise, both DAFT variants are faster than standard Transformer and other baselines, while consuming only half of the memory ².

5.3 LANGUAGE MODELING

We apply DAFT to character level language modeling on Enwik8 (Mahoney, 2011), which is another popular benchmark for auto-regressive modeling. We follow the standard preprocessing procedures and training/validation/test splits as in (Dai et al., 2019). Our base Transformer reference is a 12 layer 512 dimensional 8 head architecture with 2048 feed forward dimensions. For the first set of experiments, we use sequence length of 1024. Our training protocol is largely the same as the previous experiment, except that we increase the weight decay to 0.5 and train for 100 epochs with batch size 128. We evaluate DAFT-full and DAFT-conv with a kernel size of 64. We also compare to several efficient Transformer baselines, namely Reformer (Kitaev et al., 2020), Synthesizer (Tay et al., 2020a), Linear Transformer (Katharopoulos et al., 2020) and Performer (Choromanski et al., 2020). From Table 5, we see that, both DAFT models achieve the lowest training bits per character (bpc), which is an indicator for high model capacity. Their test performance is slightly worse than that of the basic Transformer, but outperforms all other Transformer variants.

6 CONCLUSIONS

We have introduced the Dot product Attention Free Transformer that replaces dot product attention with an efficient new operation. We have demonstrated strong results on a set of standard benchmarks with excellent efficiency. We believe that our model opens a new design space for Transformer-like models, and will see impact in various areas where self attention is needed.

²Fair speed/memory comparison against Sparse Transformer is infeasible, as it relies on a set of advanced implementation tricks such as mixed precision and gradient checkpointing, whereas DAFT is implemented with standard Pytorch utilities run in full precision.

REFERENCES

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=xTJEN-gg11b>.
- Mark Chen, Alec Radford, Rewon Child, Jeff Wu, and Heewoo Jun. Generative pretraining from pixels.
- Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509, 2019. URL <http://arxiv.org/abs/1904.10509>.
- Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020.
- Zihang Dai, Z. Yang, Yiming Yang, J. Carbonell, Quoc V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *ArXiv*, abs/1901.02860, 2019.
- 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.
- 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.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.
- Lang Huang, Y. Yuan, Jianyuan Guo, Chao Zhang, X. Chen, and Jingdong Wang. Interlaced sparse self-attention for semantic segmentation. *ArXiv*, abs/1907.12273, 2019a.
- Zilong Huang, Xinggang Wang, Lichao Huang, C. Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 603–612, 2019b.
- Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2017.
- A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. URL <https://fleuret.org/papers/katharopoulos-et-al-icml2020.pdf>.
- Nikita Kitaev, L. Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451, 2020.
- Hanxiao Liu, Zihang Dai, David R. So, and Quoc V. Le. Pay attention to mlps, 2021.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.
- Matt Mahoney. Large text compression benchmark, 2011.
- Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.
- Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=QtTKTdVrFBB>.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

- Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and T. Lillicrap. Compressive transformers for long-range sequence modelling. *ArXiv*, abs/1911.05507, 2020.
- Prajit Ramachandran, Niki Parmar, Ashish Vaswani, I. Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *ArXiv*, abs/1906.05909, 2019.
- Aurko Roy, M. Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *ArXiv*, abs/2003.05997, 2020.
- Sainbayar Sukhbaatar, E. Grave, P. Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *ACL*, 2019.
- Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models, 2020a.
- Yi Tay, Dara Bahri, L. Yang, Donald Metzler, and D. Juan. Sparse sinkhorn attention. *ArXiv*, abs/2002.11296, 2020b.
- Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey, 2020c.
- Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021.
- Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- Huiyu Wang, Y. Zhu, B. Green, H. Adam, A. Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. *ArXiv*, abs/2003.07853, 2020a.
- Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *ArXiv*, abs/2006.04768, 2020b.
- Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and M. Auli. Pay less attention with lightweight and dynamic convolutions. *ArXiv*, abs/1901.10430, 2019.
- Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, C. Alberti, S. Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and A. Ahmed. Big bird: Transformers for longer sequences. *ArXiv*, abs/2007.14062, 2020.
- Zhen Zhu, Mengdu Xu, Song Bai, Tengting Huang, and X. Bai. Asymmetric non-local neural networks for semantic segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 593–602, 2019.

A APPENDIX

B ADDITIONAL ABLATIONS

We conducted more experiments on the ImageNet-1K classification settings.

Factorization of w . We first verify the importance of the factorized parameterization of DAFT-full. As shown in Tab 6, the non factorized parameterization of DAFT-full achieves worse training and test performance than the factorized version.

Reparameterization of r . For DAFT-conv, we by default apply the reparameterization as described in Sec. 3.2. We verify that this design effectively improves the model’s performance, as shown in Table 7.

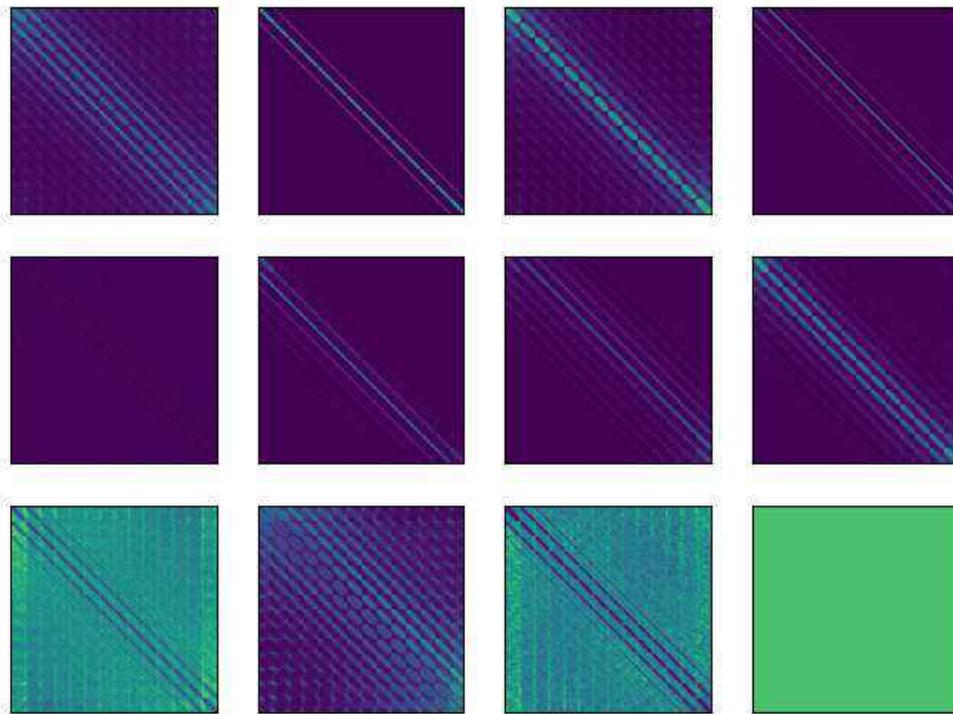


Figure 3: Exponentiated position biases learned by DAFT-full, trained on ImageNet-1K, shown from layer 1, 2, ..., 12, arranged from top left to bottom right. Each image is of size 197×197 , where the first element corresponds to the class token, and the remaining 196 correspond to the 14×14 positions. We see that local, sparse patterns are learned without explicit supervision.

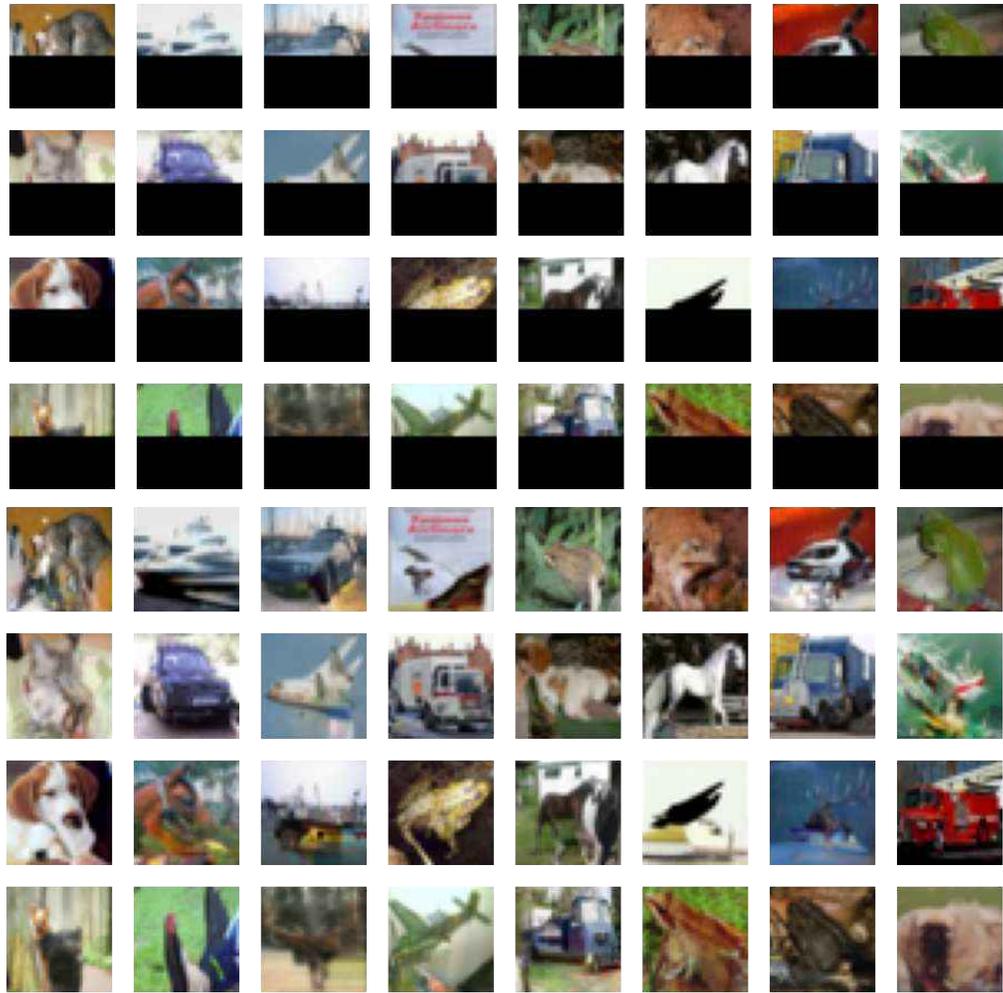


Figure 4: Image completion with the AFT-local trained on CIFAR10 autoregressive modeling task. **Top:** masked images from the test set. **Bottom:** completed images.

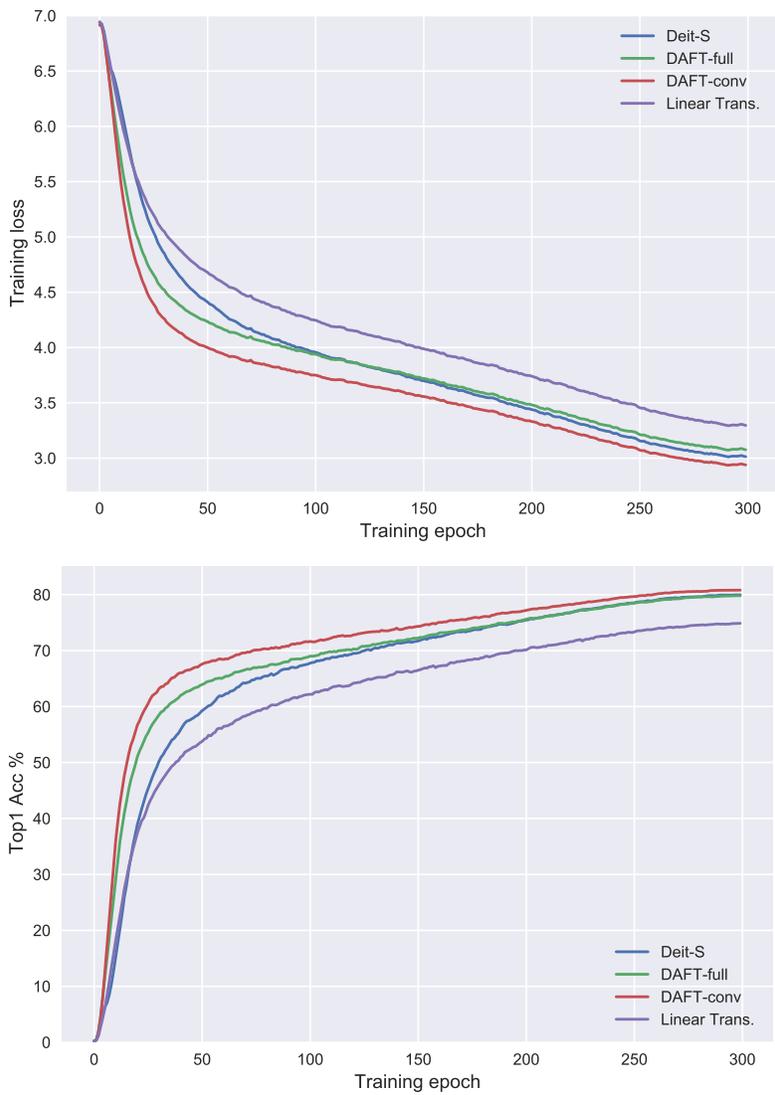


Figure 5: Training dynamics of DAFT, compared with baseline models on ImageNet. DAFT-conv enjoys faster convergence than others.

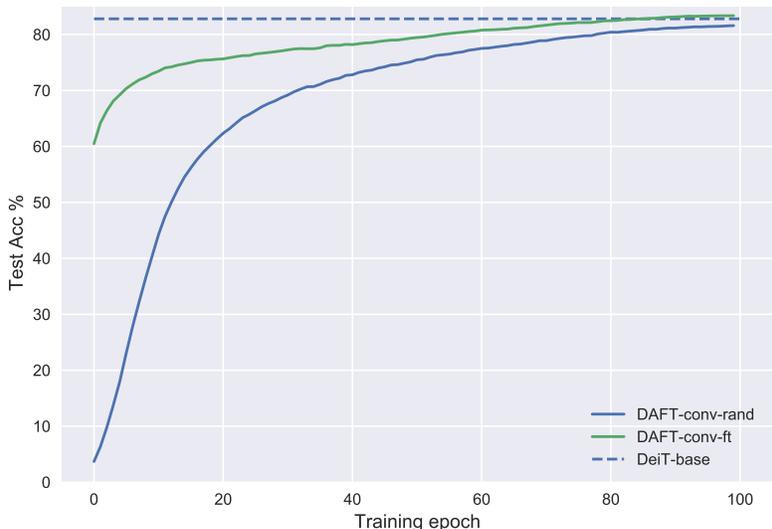


Figure 6: The top 1 test accuracy of finetuning a pretrained DeiT model to DAFT-conv (DAFT-conv-ft), compared that initialized from random (DAFT-conv-rand). DAFT-conv benefits greatly from a pretrained DeiT model, indicating the compatibility between the two family of models.

Table 6: The effect of factorized parameterization of DAFT-full.

	Train loss	Top 1 Acc
Non Factorized	3.17	78.2
Factorized (default)	3.08	79.8

Table 7: The effect of reparameterization of DAFT-conv (kernel size 7×7).

	Train loss	Top 1 Acc
Naive param	3.11	79.4
Reparameterized (default)	2.94	80.8

Convergence. We found that DAFT variants enjoy quick convergence, compared to Transformer like models. We show the training curves in Fig 5. In addition, we also show the training curve for the finetuning experiment corresponding to Tab. 3, in Fig 6.

Model scaling. Next we investigate the performance of DeiT vs. DAFT-conv as we scale the model size. We train models with different feature dimensions, while keeping the dimension of each attention head fixed for DeiT, and the kernel size fixed for DAFT-conv. The result is shown in Fig. 7. We see that throughout the size of models considered, DAFT-conv consistently outperforms DeiT model of similar size.

Contribution of the query. The query term contributes a small fraction to the computation of DAFT, but it contributes significantly to DAFT’s performance. We conducted an additional experiment with DAFT-conv (384 heads, kernel size in 11×11 and 15×15), where we remove the query term. The result is shown in Tab 8.

Visualizing the key. The keys play a central role in DAFT, as they provide content dependent reweighting for effective context reduction. In order to understand their behavior, we visualized the feature maps for a DAFT-conv model on randomly sampled images from the validation set of ImageNet-1K, as shown in Fig. 11, 12, 13, 14. Interestingly, we see that the keys gradually evolve to “object detectors” as the layer level goes up.

Table 8: Top 1 accuracy of DAFT-conv without the query term (w/o q). This results in significant performance drops.

Kernel	11	15
with q (default)	80.8	80.9
w/o q	79.3	79.5

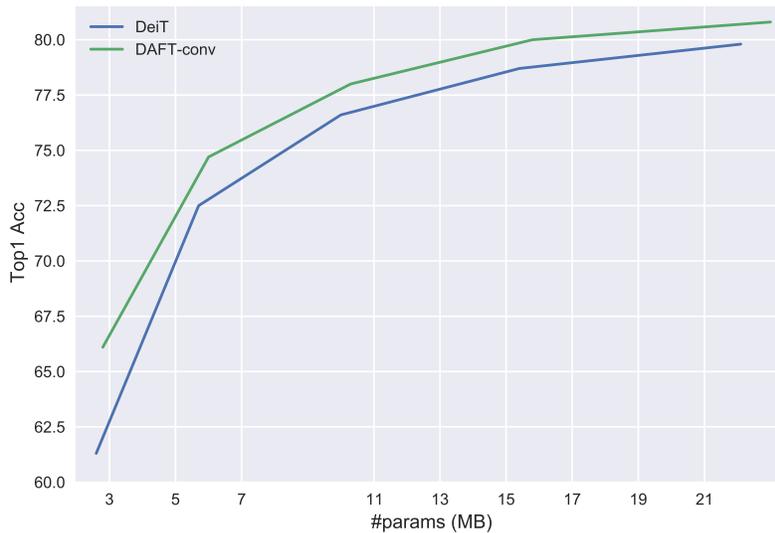


Figure 7: Performance of DeiT vs DAFT-conv, as model size scales.

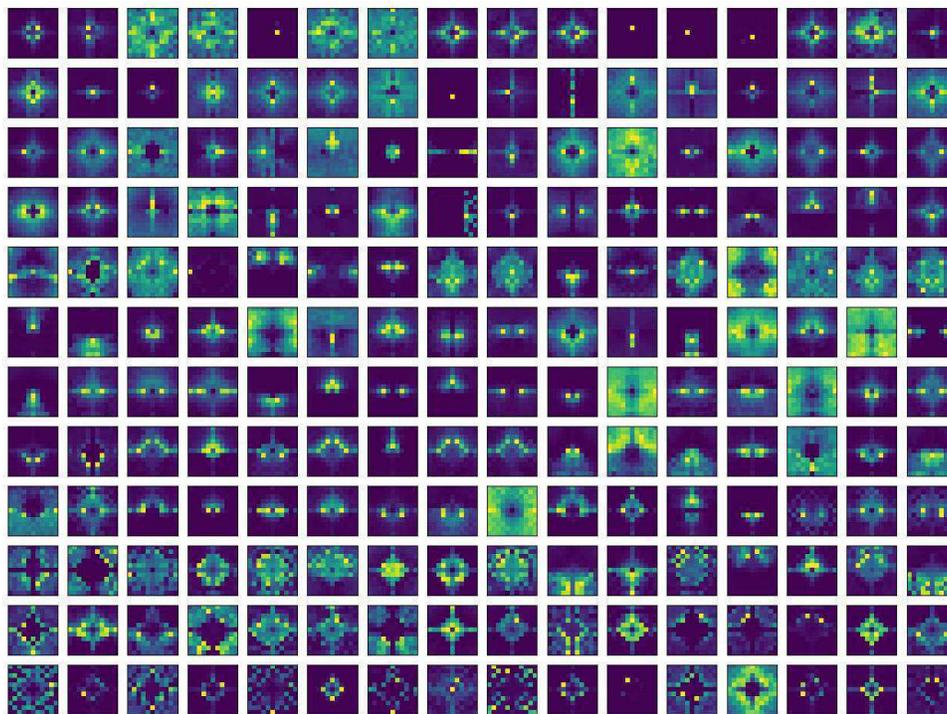


Figure 8: Exponentiated position biases learned by DAFT-conv, trained on ImageNet-1K. Each row corresponds to a layer, each column corresponds to a head (the first 16 are shown). This model has top 1 accuracy of 80.8%.

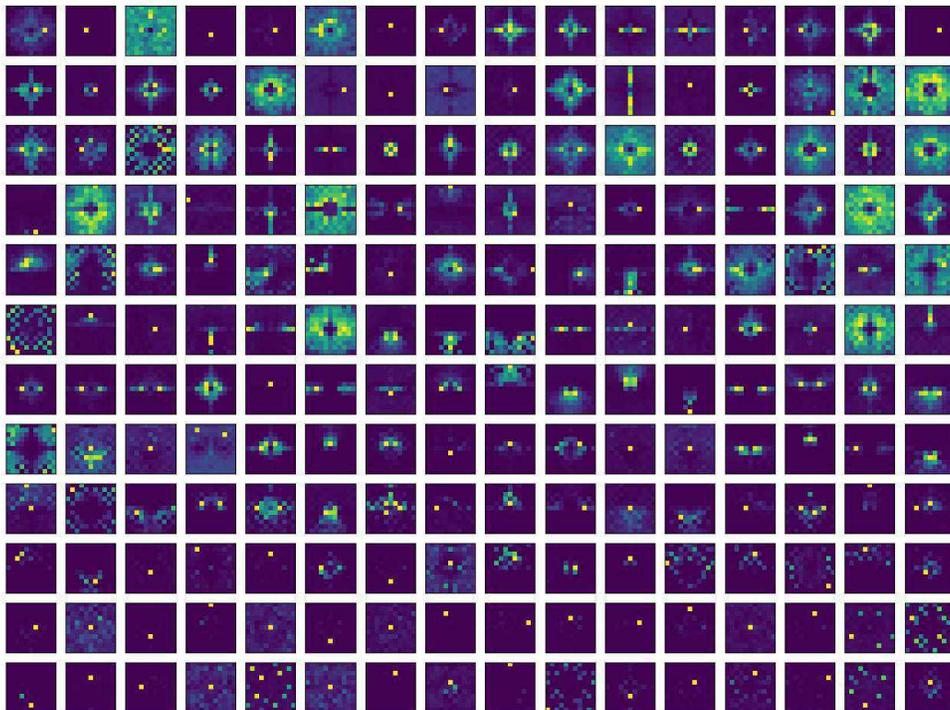


Figure 9: Exponentiated position biases learned by DAFT-conv (kernel size 11×11) with **sparsity regularization**, trained on ImageNet-1K. Each row corresponds to a layer, each column corresponds to a head (the first 16 are shown). This model has top 1 accuracy of 80.9%.

C SPARSITY

The position biases learned by DAFT-conv (kernel size 11×11) as shown in Figure 8 demonstrates interesting sparsity patterns, which suggests great potential for quantization and pruning. To this end, we experimented with a simple sparsity promoting regularization term:

$$reg(w) = \sum_{i=1}^h H(w^i), \quad H(w^i) = \text{entropy}(\text{softmax}(w^i)). \quad (10)$$

Where we simply minimize the entropy for each head, with the softmax distribution using w_i as the logits. We combining $reg(w)$ with the cross entropy loss with a small weighting (0.001) and train with the DAFT-conv with kernel size 11 and 384 heads. This results in a slight improvement in accuracy (due to its regularization effect) of 80.9 vs 80.8, as well as sparser looking position biases. The visualization is shown in Fig. 9. We see that the position biases are much more sparsely distributed as expected.

Encouraged by this, we continued to push the sparsity to an extreme form. Now for each head, we only assign a learned relative position bias for a **single position**. To do this, during training, we multiply the position biases w for each layer and each head with a sample from its corresponding Gumbel softmax distribution (Jang et al., 2017):

$$w^i = w^i * \text{gumbel}(w^i; \tau), \quad (11)$$

where τ is the temperature term for Gumbel softmax, and we set it as 0.5; $\text{gumbel}(w^i; \tau)$ produces a (sparse) sample with the same shape as w_i . During inference, the Gumbel softmax is replaced with hard max, i.e., a one hot vector is returned. This results in a model with top 1 accuracy 79.9, with less than 1 point drop compared with the unregularized model. The position biases are visualized in Fig. 10. This extreme model variant makes it possible to implement the context reduction of K, V with a combination of global average pooling and indexing operations.

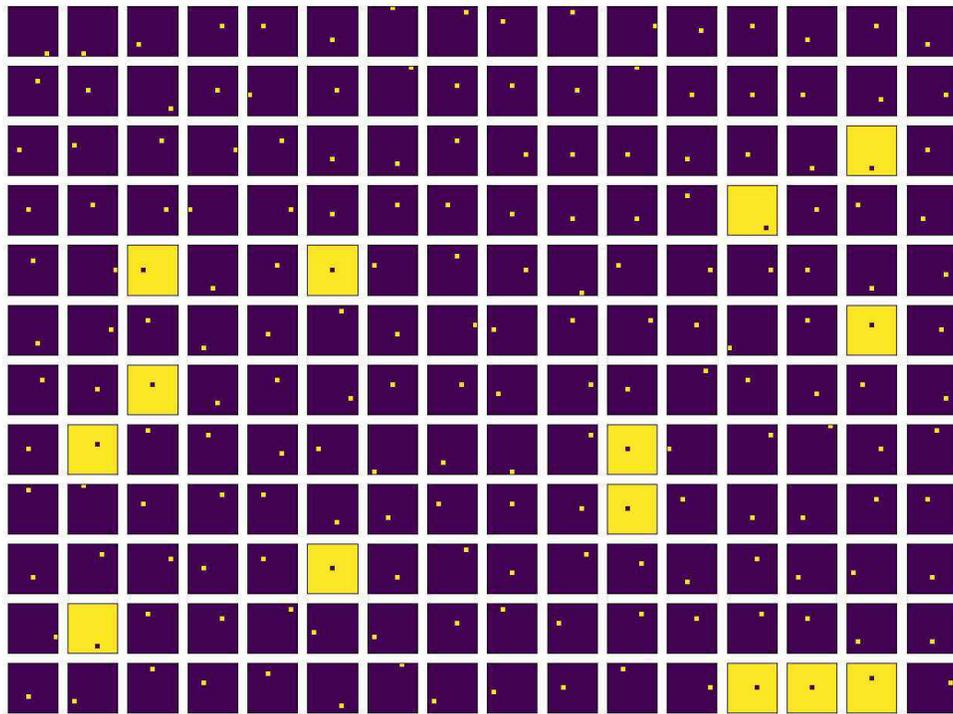


Figure 10: Exponentiated position biases learned DAFT-conv (kernel size 11×11) with **Gumbel softmax sampling**, trained on ImageNet-1K. Each row corresponds to a layer, each column corresponds to a head (the first 16 are shown). This model has top 1 accuracy of 79.9%.

Figure 11: **Top:** sample image from the validation set of ImageNet-1K. **Bottom:** visualization of the keys for DAFT-conv, with each row corresponding to a layer, each column corresponding to a head.

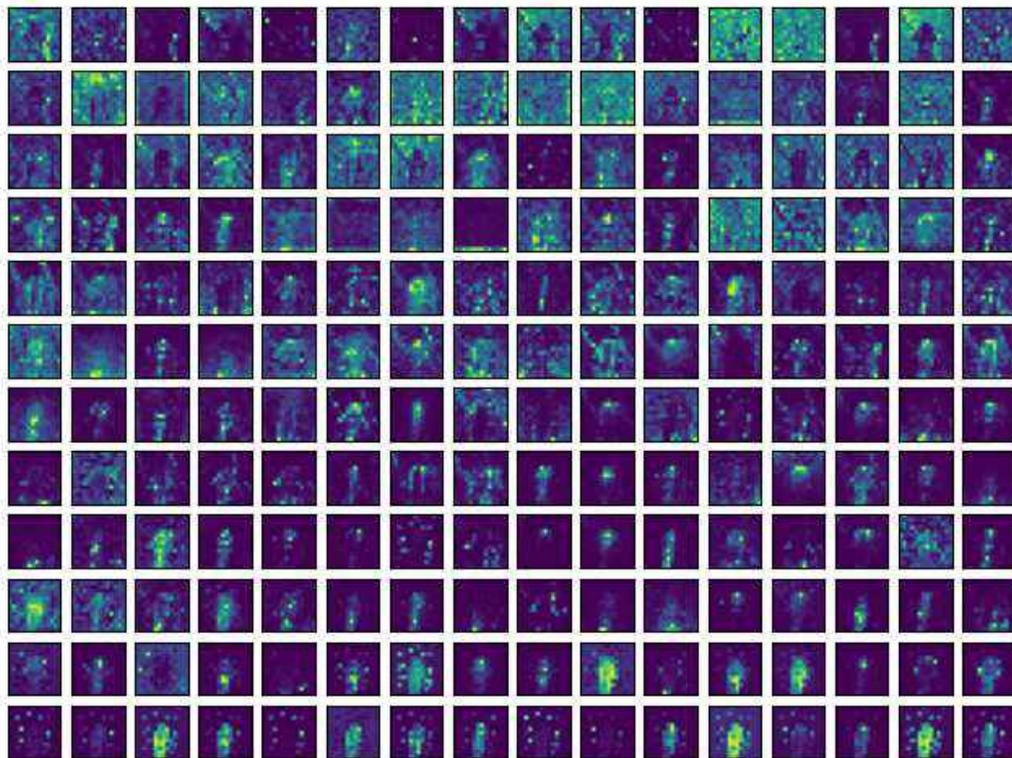


Figure 12: **Top:** sample image from the validation set of ImageNet-1K. **Bottom:** visualization of the keys for DAFT-conv, with each row corresponding to a layer, each column corresponding to a head.

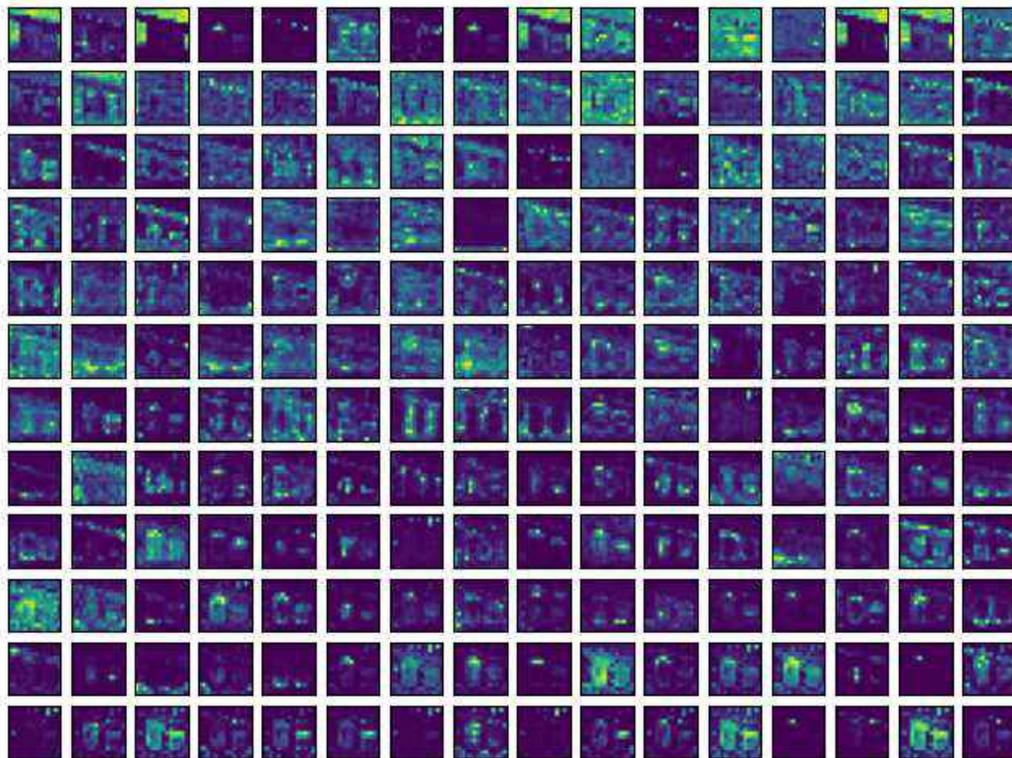


Figure 13: **Top:** sample image from the validation set of ImageNet-1K. **Bottom:** visualization of the keys for DAFT-conv, with each row corresponding to a layer, each column corresponding to a head.

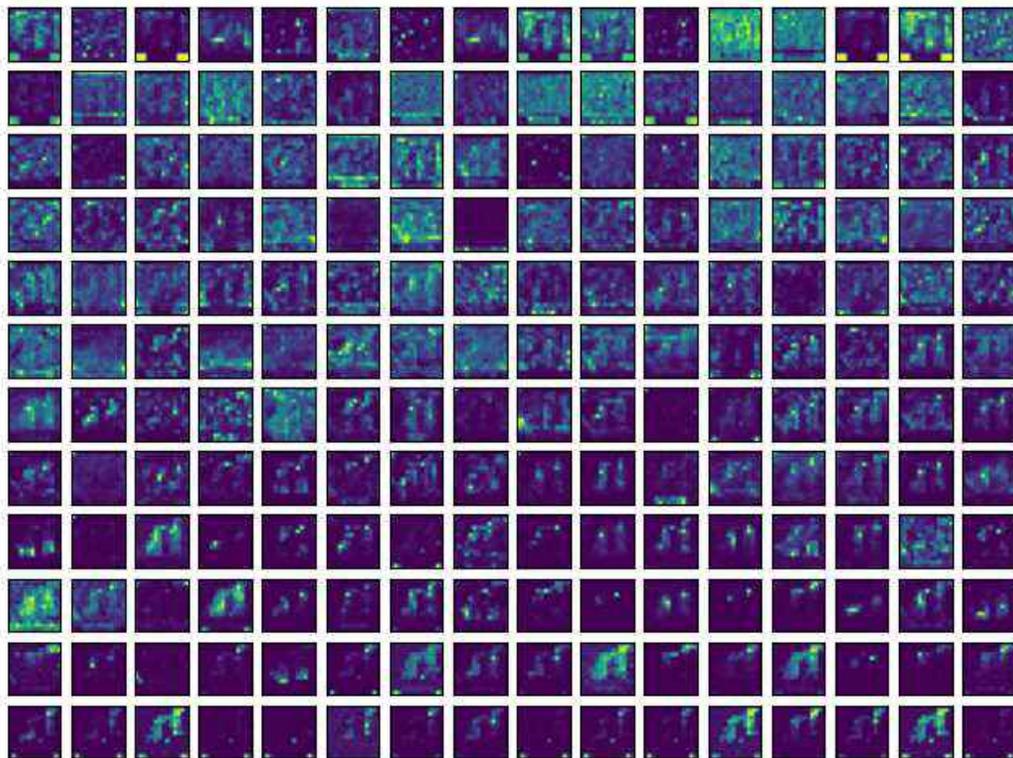
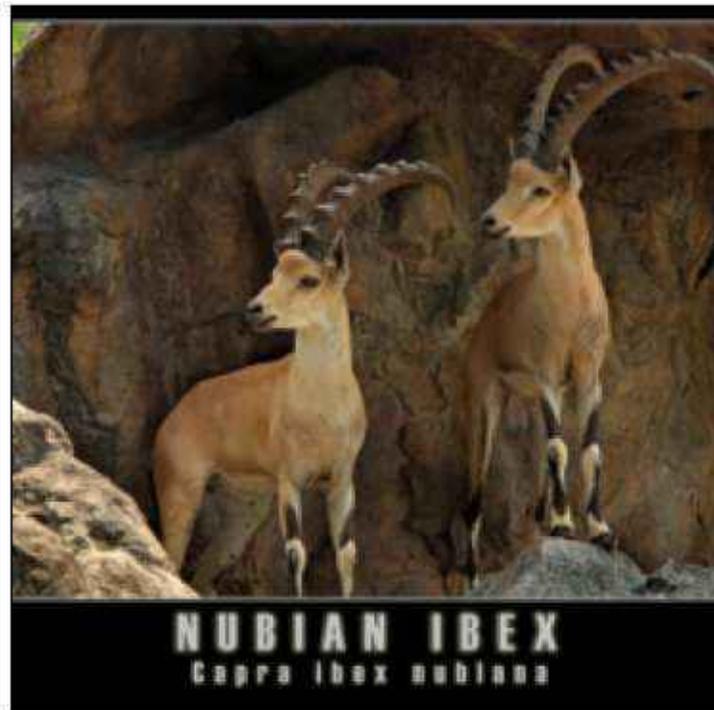


Figure 14: **Top:** sample image from the validation set of ImageNet-1K. **Bottom:** visualization of the keys for DAFT-conv, with each row corresponding to a layer, each column corresponding to a head.

