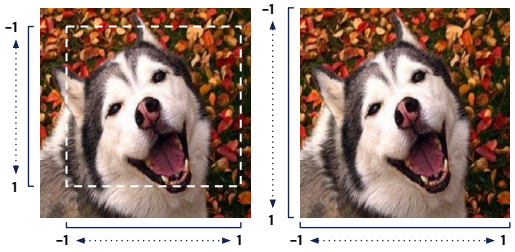

(a) Crop-relative coordinates (b) Image-relative coordinates

Figure 4. For ImageNet experiments, we generate position encodings using [-1, 1]-normalized (x, y)-coordinates drawn from (a) crops rather than from (b) the raw images, as we find the latter leads to overfitting.

B. Ablations

To illustrate the effect of various network hyperparameters, we considered a small Perceiver model and swept a number of options around it. Unlike ConvNets, each module in a Perceiver-based architecture takes as input the full input byte array. This makes it possible to sweep processing hyperparameters (e.g. depth, capacity, etc.), without reducing the effective receptive field size of the network as a whole. The base model did not share either self-attention or cross-attention parameters, used 8 heads per self-attention module, 4 self-attention modules per block, performed 2 cross-attends per image, and had a latent with an index dimension of 512 and a channel dimension 512. We used a small batch size of 64 across 32 TPUs to make sure all models fit comfortably in memory no matter how extreme the parameters. We trained all models for 5 million steps using a similar optimization procedure as in the main paper.

The results from a hyperparameter sweep centered on this base architecture are shown in Fig. 5. All results show top-1 accuracy on ImageNet. Consistent with our other experiments, these results suggest that increasing the size of the model tends to produce better results. The exception in this experiment was the number of latent dimensions, as the largest model showed signs of overfitting.

Similarly, we evaluated the effect of the latent array's initialization scale and the parameters of the Fourier frequency position encoding on ImageNet performance. The results of this experiment are shown in Fig. 6. These experiments use the full-sized ImageNet architecture, but were trained with a smaller batch size (256) and fewer TPUs (16) (for reasons of compute availability). These experiments suggest that standard and relatively small values for the initialization scale are best (values ≥ 1 may lead to instability), and generally suggest that a higher number of Fourier frequency bands and a higher maximum resolution (up to Nyquist) improve performance. We found that a scale of 1.0 worked best for initializing the position encoding: this value is used for the model reported in Tab. 2.

# cross-attends	Acc.	FLOPs	Params
4	39.4	173.1B	12.7M
8	45.3	346.1B	23.8M
12	OOM	519.2B	34.9M

Table 5. Performance of models built from a stack of cross-attention layers with no latent transformers. We do not share weights between cross-attention modules in this experiment. Models with 12 cross-attends run out of memory on the largest device configuration we use (64 TPUs). Results are top-1 validation accuracy (in %) on ImageNet (higher is better).

# cross-attends	Acc.	FLOPs	Params
1 (at start)	76.7	404.3B	41.1M
1 (interleaved)	76.7	404.3B	42.1M
2 (at start)	76.7	447.6B	44.9M
2 (interleaved)	76.5	447.6B	44.9M
4 (at start)	75.9	534.1B	44.9M
4 (interleaved)	76.5	534.1B	44.9M
8 (at start)	73.7	707.2B	44.9M
8 (interleaved)	**78.0**	707.2B	44.9M

Table 6. Performance as a function of # of cross-attends and their arrangement. In "interleaved," cross-attention layers are spaced throughout the network (for re-entrant processing), while in "at start" all cross-attends are placed at the start of the network followed by all latent self-attend layers. All cross-attention layers except the initial one are shared, and self-attends are shared as usual (using 8 blocks of 6 self-attention modules). Results are top-1 validation accuracy (in %) on ImageNet (higher is better).

For all FLOPs numbers reported here, we report unfused multiply-adds

All FLOPS reported here give theoretical FLOPS with multiplies and accumulates counted as separate operations. This is the strategy used in (Kaplan et al., 2020) and elsewhere in the literature. Note that some other papers in the literature report FLOPS using fused multiply-accumulates: using this strategy will approximately cut our reported figures in half.

C. Architectural details

The Perceiver consists of two modules: a cross-attention module and a Transformer. In the cross-attention module, inputs are first processed with layer norm (Ba et al., 2016) before being passed through linear layers to produce each of the query, key, and value inputs to the QKV cross-attention operation. The queries, keys, and values have the same number of channels as the minimum of the input channels, which is typically the key/value input (i.e. 261 for ImageNet) The output of attention is passed through an additional linear layer to project it to the same number of channels in the query inputs (so it can be added residually).

The query inputs for the first cross-attention layer (e.g. the

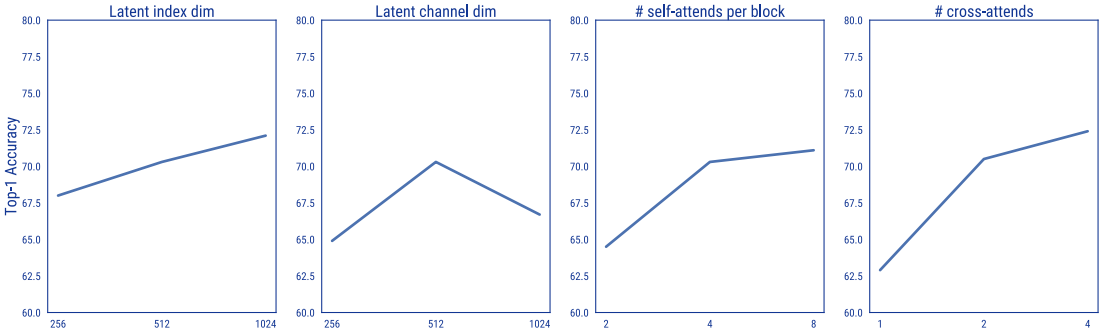

Figure 5. The effect of model hyperparameters, using a scaled-down Perceiver architecture on ImageNet. All plots show top-1 accuracy (higher is better). Increasing the size of the latent index dimension, the number of self-attends per block, and the number of cross-attends generally produced better results. Increasing the size of the latent channel dimension helps up to a point, but often leads to overfitting.

left-most latent array in Fig. 1) are learned, per-element weights with the same shape as the latent array (e.g. for ImageNet, they are a 512×1024 array). These function like learned position encodings in the Transformer literature or like a learned initial state in the recurrent neural network (RNN) literature. The latent array is randomly initialized using a truncated normal distribution with mean 0, standard deviation 0.02, and truncation bounds [-2, 2]. Network performance is fairly robust to the scale of this initialization (see Fig. 6, left facet).

In the self-attention block, inputs are processed with layer norm and passed through query, key, and value layers before being used to compute QKV self-attention. The output is passed through another linear layer.

Each cross-attention and self-attention block is followed by a dense (multi-layer Perceptron) block. In the dense block, inputs are processed with layer norm, passed through a linear layer, activated with a GELU nonlinearity (Hendrycks & Gimpel, 2016), and passed through a final linear layer. We used dropout throughout the network in earlier experiments, but we found this led to degraded performance, so no dropout is used.

All linear layers (including query, key, and value layers and dense block layers) preserve the dimensionality of their inputs and are tiled over input index dimensions (i.e. applied as a 1×1 convolution).

To produce output logits, we average the output of the final latent self-attention module over the index dimension. This produces a single global summary vector, which we then project to the number of target classes using a single linear layer. This is the process used by e.g. ResNets to convert a convolutional feature map to logits (He et al., 2016).

As with other Transformer architectures, the Perceiver's Transformer has a fully residual design, and its input is always added to its output for further processing. This applies to cross-attention modules as well: the latent component of the input is added to its output. We give details on the hyperparameters used on different datasets in the main paper.

	Valid	Train	Params	FLOPs
No weight sharing	72.9	87.7	326.2M	707.2B
W/ weight sharing	**78.0**	79.5	44.9M	707.2B

Table 7. Weight sharing mitigates overfitting and leads to better validation performance on ImageNet. We show results (top-1 accuracy) for the best-performing ImageNet architecture (reported in Tables 1-2 of the main paper) on train and validation sets. This architecture uses 8 cross-attends and 6 blocks per latent Transformer. The model labeled "W/ weight sharing" shares weights between cross-attention modules 2-8 and between the corresponding blocks of all latent Transformers. The first cross-attention module uses its own, unshared weights.

D. Position encodings and Fourier features

Crop-relative coordinates. As described in the main paper, we found that generating position coordinates using cropped data rather than on the raw data was important to prevent excessive overfitting. We illustrate the cropping procedure on ImageNet in Fig. 4.

Fourier feature parameterizations. We choose the Fourier feature parameterization described in section 3.2 of the paper to allow us to intuitively set the maximum band when the sample rate of the input signal is regular and known. By setting the number of bands independently, we allow it be easily controlled in line with a computational budget: we generally found that more bands helped for a given architecture (assuming it fits in memory). For signals with irregular or very fine sampling, such as ModelNet40

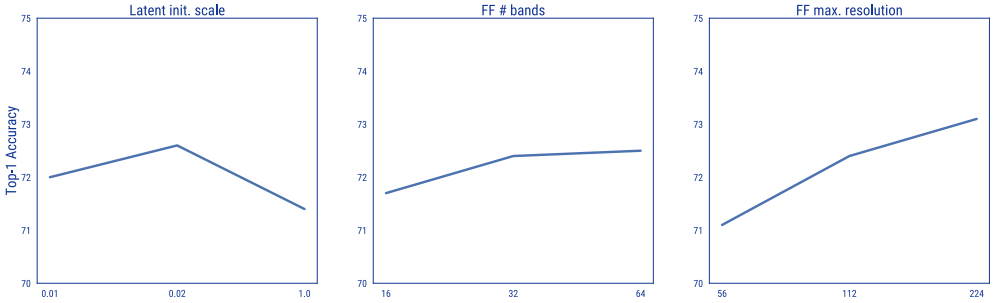

Figure 6. The effect of latent initialization scale and Fourier feature (FF) position encoding parameters on performance. All plots show top-1 accuracy (higher is better). The model with initialization scale of 0.1 diverged during training. Generally, increasing the number of bands and max resolution (up to Nyquist) increased performance. We observed the same effects whether using linearly or logarithmically spaced position encoding bands.

point clouds, the maximum band can also be treated as a hyperparameter. This is in contrast to the parameterization used in NeRF (Mildenhall et al., 2020), which produces very high frequencies if a moderate number of bands are used (e.g. the 64^{th} band would have a frequency of $2^{64} = 1.8e19$). Rather than tying the maximum frequency to the number of bands, our parameterization samples the spectrum more densely as more bands are added. Our parameterization is identical to the parameterization described in the original Transformer paper, except we express each band in terms of its frequency rather than its wavelength (we find this more natural in the context of signals like images) and we assume that input positions are in $[-1, 1]$ rather than $[0, s)$ for a sequence of length s.

E. Audiovisual attention maps

We visualize video and audio attention maps (respectively) for the first and second cross-attention module of a multi-modal Perceiver model trained on AudioSet using 2x4x4 video patches and 61,440 raw audio samples

We visualize video attention maps similarly to static image attention maps (Fig. 3), but with the addition of a time dimension: each column shows the attention to the full image at a time step of the video. Because this AudioSet Perceiver takes space-time patches of shape time 2 $\times$ height 4 $\times$ width 4, the same attention is applied to pairs of subsequent frames. For visualization purposes, we show every other frame of the input video and attention maps: each attention map is applied to two video frames.

All attention maps of this network appear to be sensitive to both static and dynamic features of the input video. Some attention maps exhibit spatiotemporal structure reminiscent of the filters seen in spatiotemporal image processing (Adelson & Bergen, 1985; Simoncelli & Heeger, 1998). Because the Perceiver uses learned attention rather than a fixed bank of spatiotemporal filters, it can adapt its attention to the input content.

We visualize audio attention maps by displaying the mel-spectrogram and attention maps as images. Mel-spectrograms are plotted with time plotted on the x- and frequency on the y-axis. Although they are harder to interpret visually than the image attention maps, they appear to share a common structure of Fourier-frequency positional banding and content-related modulation.

F. Notes on changes from the original version

Our Audioset mAP results in the first arXiv version were flawed (and unfortunately higher) so we repeated and expanded those experiments and now provide correct numbers. The issue was that when computing AP using the sklearn package, we passed the matrix of class scores transposed to what the function expects – hence the number of classes and number of examples were switched.

We have slightly improved the results reported on ImageNet since the first version by (i) removing dropout, (ii) removing a linear layer that was originally (unintentionally) included following the initial latent array, and (iii) averaging before rather than after projecting when computing the output logits.

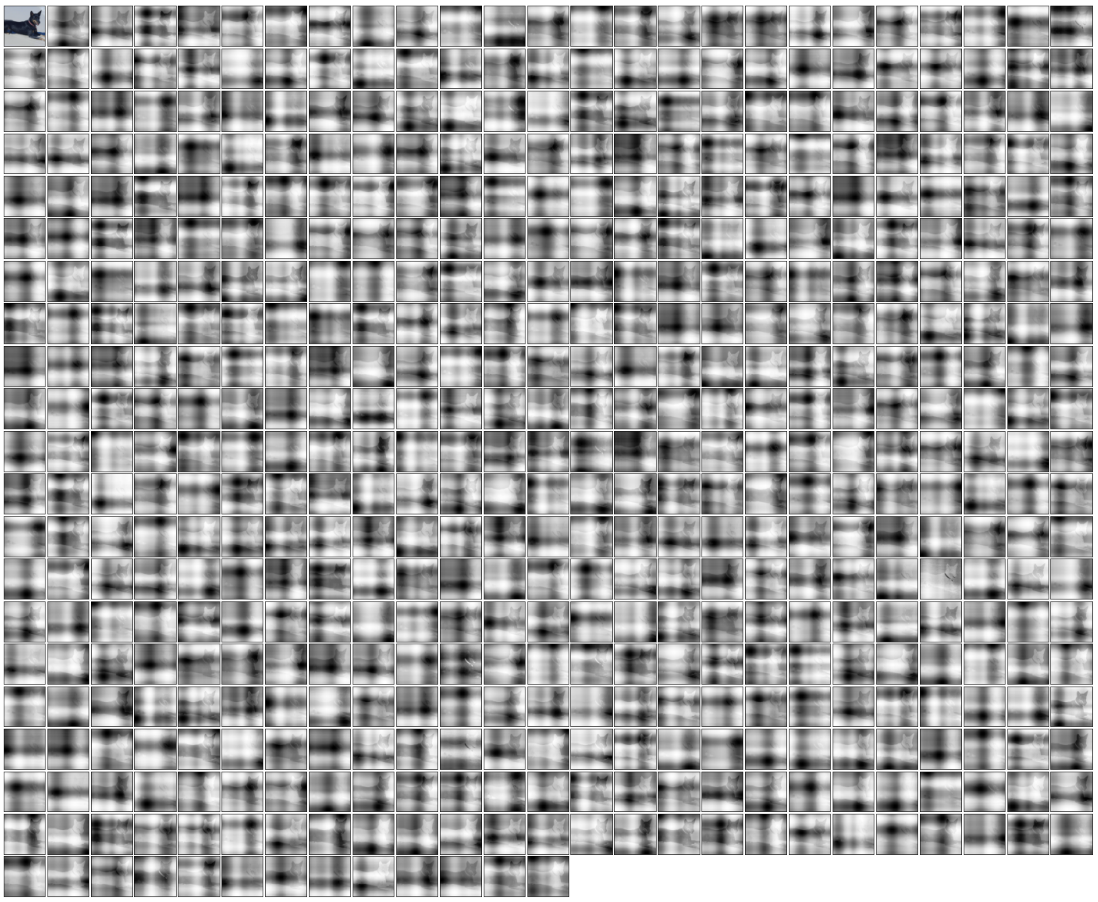

Figure 7. Example attention maps from the **first cross-attend** of an ImageNet network trained with **2D Fourier feature** position encodings.

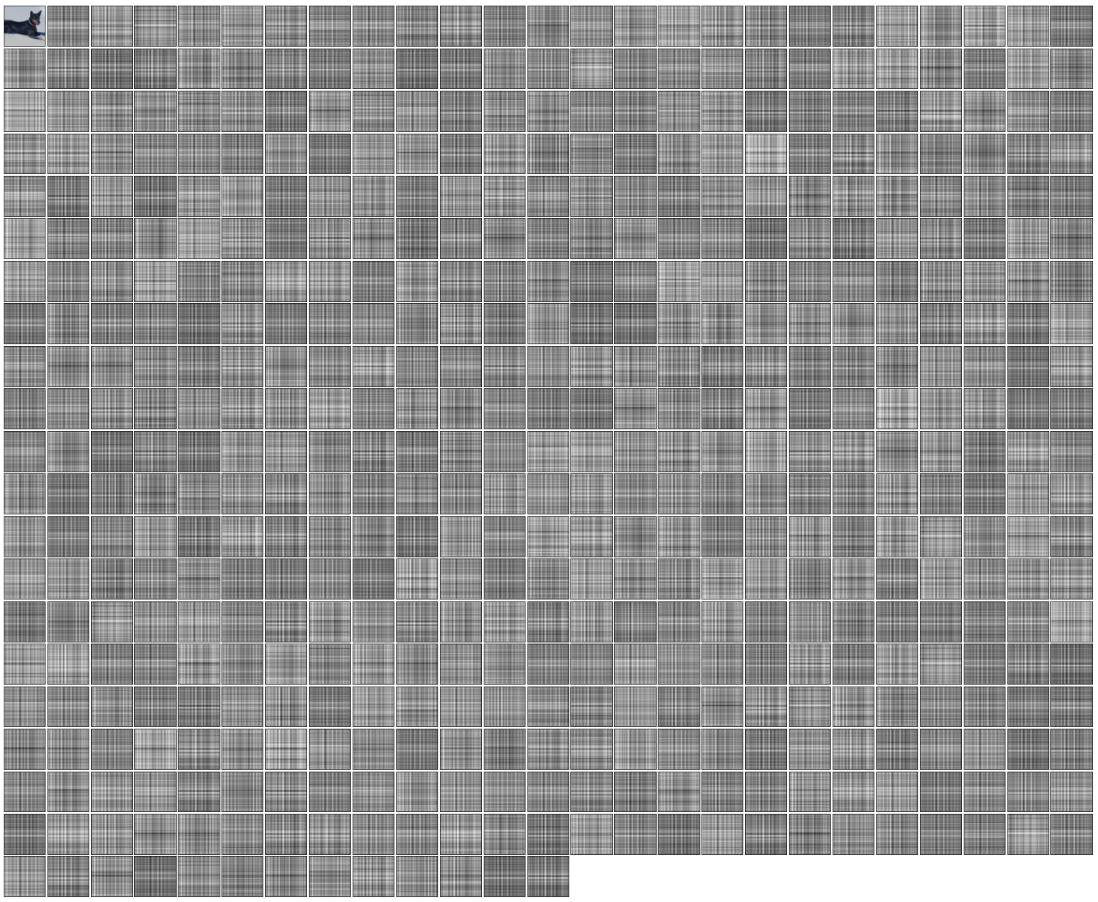

Figure 8. Example attention maps from the **eighth (final) cross-attend** of an ImageNet network trained with **2D Fourier feature** position encodings.

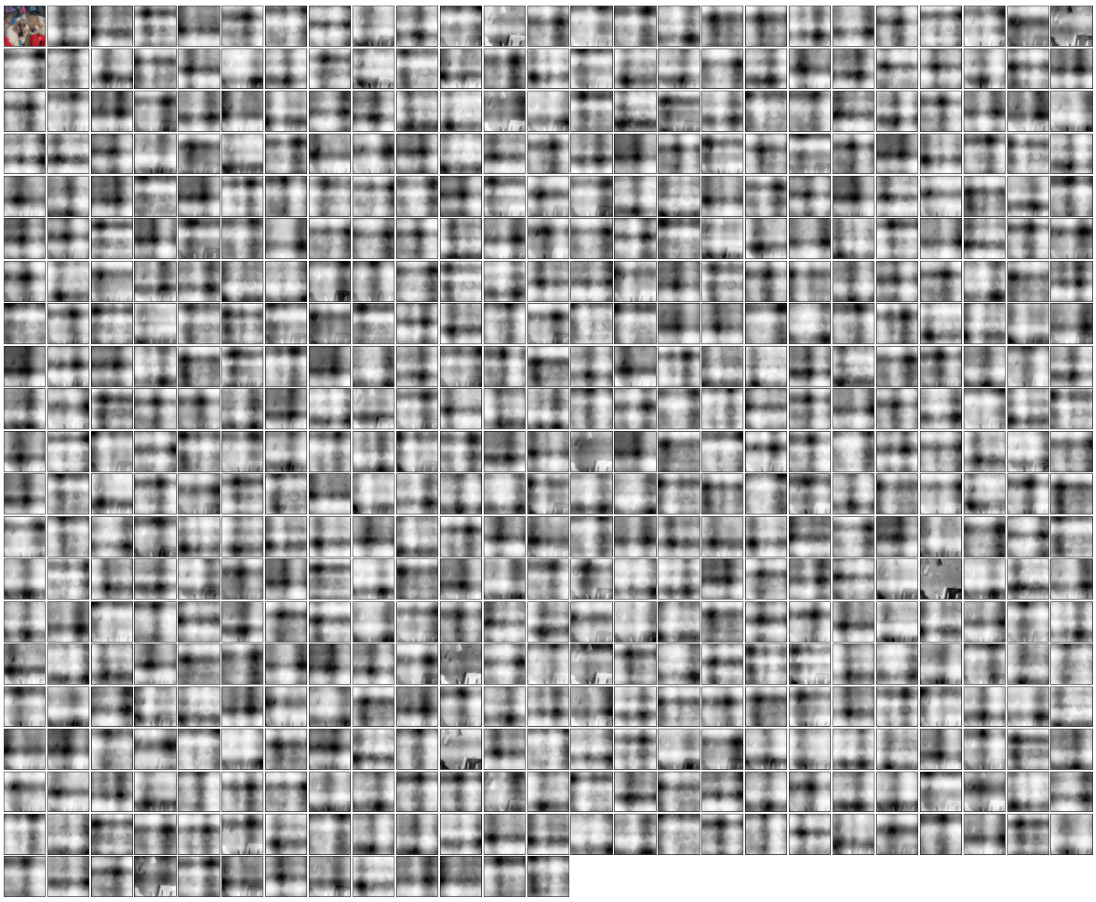

Figure 9. Example attention maps from the **first cross-attend** of an ImageNet network trained with **2D Fourier feature** position encodings.

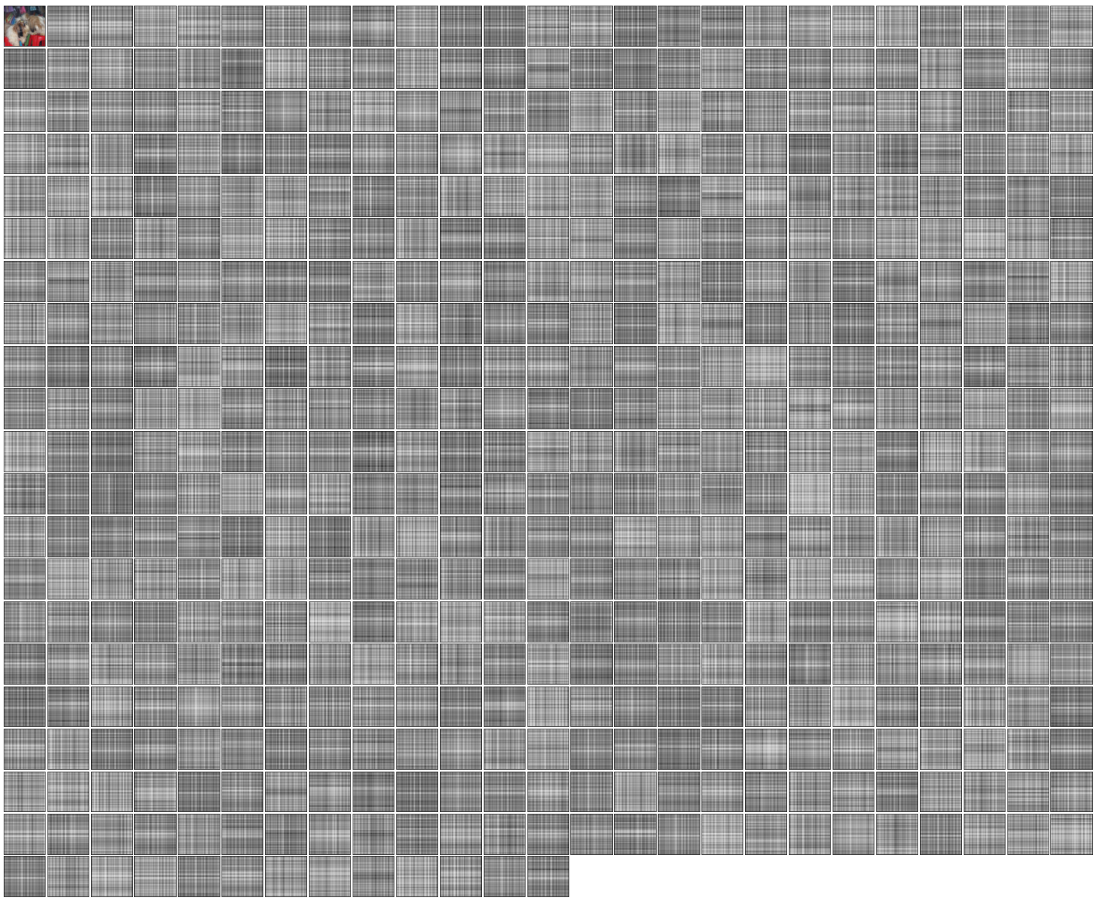

Figure 10. Example attention maps from the **eighth (final) cross-attend** of an ImageNet network trained with **2D Fourier feature** position encodings.