Supplementary Material: Foveation in the Era of Deep Learning

George Killick
2182951k@student.gla.ac.uk
Paul Henderson
paul.henderson@glasgow.ac.uk
Paul Siebert
paul.siebert@glasgow.ac.uk
Gerardo Aragon-Camarasa
gerardo.aragoncamarasa@glasgow.ac.uk

School of Computing Science
University of Glasgow
Glasgow, UK

1 Gaussian Derivative Basis Filters

The basis filters can be computed through the multiplication of a Gaussian windowing function G and the Hermite polynomials H where $H_m(x)$ computes the m^{th} order partial derivative along the x axis. Accordingly, a 2-D Gaussian Derivative basis filter can be computed as follows:

$$B(x,y,\sigma,m) = (-1)^{m_x m_y} H_{m_x}\left(\frac{x}{\sigma\sqrt{2}}\right) H_{m_y}\left(\frac{y}{\sigma\sqrt{2}}\right) G(x,y,\sigma) \tag{1}$$

Where x and y are the coordinates at which the filter is evaluated, σ controls the size of the filter, and $m = (m_x, m_y)$ defines the order of the partial derivatives in the x and y directions respectively. The Hermite polynomials are defined as:

$$\begin{aligned} H_0(x) &= 1 \\ H_1(x) &= 2x \\ H_2(x) &= 4x^2 - 2 \\ H_3(x) &= 8x^3 - 12x \\ H_4(x) &= 16x^4 - 48x^2 + 12 \\ H_5(x) &= 32x^5 - 160x^3 + 120x \end{aligned} \tag{2}$$

2 Gaussian Derivative Graph Convolution

Equation 3 defines our convolution operation for single channel input and single channel output layer.

$$\hat{f}_j = \sum_{m=0}^{|S|} w_m \sum_{i \in K_j} f_i B(\delta_{xij}, \delta_{yij}, \sigma_j, S_m) \tag{3}$$

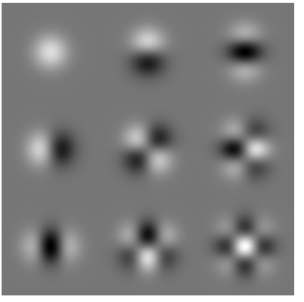

Figure 1: Visualization of Gaussian Derivative Basis Functions. The order of the partial derivative in the y direction increases from left to right, and the x direction increases from top to bottom. Our choice of basis functions for a given layer is determined by the hyperparameter M. The basis includes all Gaussian derivatives where the maximum order of their partial derivatives is $\leq M$

f_i is the feature associated with the vertex $u_i \in U$ in the input features, and w is a set of learned weights that weigh the responses to individual basis filters. B refers to equation 1 that computes basis filter weights. δ_{xij} and δ_{yij} are the delta offset labels associated with the edge $e_{ij} \in E$. σ_j is the size of the filter when centred on the output node $v_j \in V$. $S = \{(0,0),(0,1)...(M,M)\}$ where S_m defines the order of the partial derivatives in the x and y directions, for the m^{th} basis filter. M is a hyperparameter that defines the maximum order of a partial derivative that is permitted for the basis filters, i.e. any filter with a partial derivative greater than M is not included in the basis. K_j refers to the neighbourhood of the output vertex $v_j \in V$, i.e. the set of vertices in U that have an edge with v_j. Additionally, we normalise the filter values for each basis filter by subtracting its mean and dividing by its ℓ_2 norm for a given neighbourhood K. One exception is for the 0^{th} order gaussian derivative, i.e. a normal gaussian filter, which we normalize by dividing by its ℓ_1 norm.

3 Border vs. Zero Padding

In early pilot testing, we observed that zero padding would frequently lead to the collapse of the attention module (i.e. it would predict the same fixation regardless of the input image). We conduct an experiment comparing how this affects performance using a 2 fixation sequential network on the Imagewoof. We observe a decrease from 79.2% accuracy to 77.8%. This is approximately in line with the performance improvement from a 1 fixation network to a 2 fixation network (Figure. **??**). We could not ascertain the exact cause of this peculiarity. Fortunately, border padding is readily provided in most frameworks meaning this is not problematic.

4 Training Details (ImageNet100)

Networks are trained with a batch size of 64, and the AdamW [2] optimizer. We perform a linear warmup on the learning rate for 5 epochs, followed by cosine annealing for 85 epochs [1]. During training, we use trivial data augmentation [3], followed by resizing the shortest side to 256px and a random resized crop of 224×224. At test time, we resize the shortest side to 256px and perform a 224×224 centre crop. Images are normalized using Imagenet mean and standard deviation. We use a learning rate of 0.004 and a weight decay of 0.005

Foveated Graph ConvNeXt atto config	Imagenet 100k Settings
Fovea Radius	40%
Stem - kernel size	16
Stem - sigma	1.0
Stem - max order	4
Blocks - kernel size	49
Blocks - sigma	0.8
Blocks - max order	4
Downsampling - kernel size	4
Downsampling - sigma	0.6
Downsampling - max order	0

Table 1: Stem refers to the initial convolution layer in the ConvNeXt architecture, Blocks refers to the configuration of the depthwise convolution layers in the NeXt Blocks. Downsampling refers to the configuration of the downsampling layers that reduce spatial dimensionality between stages in the ConvNeXt architecture. Kernel size is analogous to kernel size in ordinary convolution layers. Sigma determines the size of Gaussian derivative basis filters; max order refers to the maximum order of partial derivatives used in the basis.

for the feature extractors. For the localization network, we use a learning rate of 0.0004 for the final 1×1 convolution (or MLP in case of the full affine spatial transformer) and a learning rate of 0.00004 for the convolution stages as done in [4]. We found many methods that use a localisation network collapse during training; therefore, we use weights pre-trained on Imagenet-1K for the localisation networks. We evaluate the model on the test set using the best-performing model checkpoint on the validation set. We include the implementation details that are specific to our graph convolutional ConvNeXt atto in table 4.

4.1 Implementation details (MNIST)

We utilise a 4-layer feature extractor for all methods, with 64 3×3 filters in each layer, followed by batch normalisation and ReLU activation. We found that larger networks did not increase performance. For our method, we use our graph convolution in place of the standard 2D convolution operator and a filter size of 9 so that the filters are the same size as in the standard grid convolution case. The convolutional features from the final layer are global average pooled and passed to a linear classifier. We restrict the number of input pixels to the feature extractor to be 28^2, the size of a normal MNIST image. We use the same architecture as the feature extractor, without average pooling and the linear classifier, for methods that use a separate localisation network to adjust the sampling grid. The localisation networks receive a 28×28 uniformly down-sampled image as input.

We independently perform a random hyperparameter search across learning rate and weight decay for all methods. We use the AdamW optimiser [2] and train with a batch size of 64 for 20 epochs, which we found was sufficient for networks to converge. We use a 5-epoch linear warm-up schedule followed by 15-epoch cosine decay [1]. We report accuracy on the test set using the best-performing model checkpoint on the validation set.

References

[1] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

[2] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[3] Samuel G Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 774–782, 2021.

[4] Adria Recasens, Petr Kellnhofer, Simon Stent, Wojciech Matusik, and Antonio Torralba. Learning to zoom: a saliency-based sampling layer for neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 51–66, 2018.