

Monte-Carlo Convolutions on Foveated Images

George Killick^a, Gerardo Aragon-Camarasa^b and J. Paul Siebert^c

School of Computing Science, University of Glasgow, Glasgow, U.K.

Keywords: Foveated, Convolution, Retina, Implicit, Neural, Representations.

Abstract: Foveated vision captures a visual scene at space-variant resolution. This makes the application of parameterized convolutions to foveated images difficult as they do not have a dense-grid representation in cartesian space. Log-polar space is frequently used to create a dense grid representation of foveated images, however this image representation may not be appropriate for all applications. In this paper we rephrase the convolution operation as the Monte-Carlo estimation of the filter response of the foveated image and a continuous filter kernel, an idea that has seen frequent use for deep learning on point clouds. We subsume our convolution operation into a simple CNN architecture that processes foveated images in cartesian space. We evaluate our system in the context of image classification and show that our approach significantly outperforms an equivalent CNN processing a foveated image in log-polar space.


1 INTRODUCTION


We are concerned with the application of parameterized convolution filters to non-uniformly sampled images, particularly those produced by foveated sensors. Foveated sensors reduce image size by decreasing sampling resolution as a function of eccentricity from some location within the visual scene (the fixation location). In some computer vision applications, it may be desirable to extract high-frequency information and operate on a wide field of view. This may be computationally intractable if the required resolution and field of view is excessively high, as many computer vision algorithms have a computational complexity that scales with image size. Foveated sensors aim to emulate operating on the full resolution image by fixating on important high-frequency information while still sampling a large field of view. The output of a foveated sensor is significantly smaller than the full resolution image and is consequently computationally feasible to operate on with deep neural networks.


Due to the non-uniform sampling of a visual scene through a foveated sensor, applying convolutional neural networks to foveated images is difficult as convolution layers expect input data to have a uniform grid representation. To solve this, a log-polar trans-

form is usually applied to the image to create a compressed representation that is in a grid format (Ozimek et al., 2019), (Schwartz, 1980). While this format is compatible with convolution layers, it fundamentally changes the representation of the data. Log-polar image representations do not have translation equivariance like cartesian images, instead having scale and rotation equivariance (about the fixation location) (Weiman and Chaikin, 1979). While not an inherently poor representation of visual data, log-polar representations may not be desirable for all applications. To this end, we aim to remove the need to apply a log-polar transform to foveated images in order to use them with CNNs.

In this paper, we propose a convolution operation that can directly operate on foveated images in cartesian space without requiring a log-polar representation. Our operation is inspired by convolution operators designed for point clouds (Wu et al., 2019)(Hermosilla et al., 2018)(Wang et al., 2018), which frame the convolution operation as the Monte-Carlo estimation of the continuous convolution integral. Monte-Carlo convolution operations require the convolution kernel to be represented continuously in the spatial domain. Typically, this is achieved through a coordinate-based MLP, which maps spatial locations to a filter weight. The current convention is to use the ReLU activation function within the MLP. However, adjacent work in implicit neural representations suggests that the Sine activation function is more suitable

^a  <https://orcid.org/0000-0002-6881-5535>

^b  <https://orcid.org/0000-0003-3756-5569>

^c  <https://orcid.org/0000-0002-9405-4872>

for this task. To this end, we propose a novel formulation of the Monte Carlo Convolution operation that uses a Sine activated MLP to represent spatially continuous convolution kernels.

We use our proposed convolution operation to build a simple CNN architecture and evaluate its performance on the CUB-200-2011 Dataset (Wah et al., 2011), a popular fine-grained image classification dataset comprised of images of 200 species of birds. We simulate a foveated sensor in software using the method proposed in (Balasuriya, 2006). We compare our convolution operation against a standard CNN processing foveated images represented in a log-polar space and show that classification performance is improved when applying convolutions in the cartesian coordinate frame. Additionally, we compare different MLP designs for representing spatially continuous convolution filters. We show that classification performance is significantly improved when using an MLP with the Sine activation function compared to the more traditional ReLU MLP. We do not intend to show that foveated vision itself is beneficial to this particular classification task. Rather, we intend to show that if one wishes to use foveated vision in a computer vision pipeline, our proposed convolution operation can operate on this data in the cartesian coordinate frame effectively. Finally, we show that in the case of a single hidden layer MLP with Sine activations, it suffices to freeze first layer weights at initialization and only learn the bias terms with no significant performance decrease.

To summarise, our contributions in this work are three-fold:

- We demonstrate that Monte-Carlo convolutions, frequently used for point clouds, can be applied effectively to foveated images in the cartesian coordinate frame.
- We empirically show that the expressiveness of the continuous filter kernel derived from a coordinated based MLP is significantly improved when using the Sine activation instead of ReLUs.
- We propose a parameter efficient coordinate based MLP that can achieve comparable performance to its inefficient counterpart and provide intuition on why it works through a comparison to Fourier series synthesis.

While the results from this work are promising, it is in its infancy, with many issues still present. We address these in the limitations section. Nonetheless, we believe the work presented in this paper will be helpful as a strong baseline for future work on processing foveated images with deep neural networks in the cartesian coordinate frame.

2 RELATED WORK

2.1 Deep Learning on Foveated Images

Foveated vision has been incorporated into deep learning vision systems in a variety of forms. (Karpathy et al., 2014) adopt a simple approach to foveated vision using two images, a high-resolution crop and a low-resolution full field of view. Each image is processed by its own CNN before being concatenated and fed to further fully connected layers. (Li et al., 2017) approach differs slightly through sharing weights between the Convolution Neural Networks (CNNs) that process the high-resolution crop and the low-resolution full field of view. This approach does not require a log-polar representation and is easily integrated into CNN architectures. (Balasuriya, 2006) comments on the potential pitfalls of this approach, particularly in the extraction of features that span different fields of view. To our knowledge, no one has reported on whether this discontinuity has significant effects on a deep learning system in practice. While a comparison could be made between this approach and log-polar foveated images, it is not a like for like comparison as the log-polar space fundamentally changes the way filters are applied to visual scene. In this paper, we provide a framework for investigating this claim in the future.

(Nakada et al., 2018) use a random distribution of points in log-polar space to sample a simulated visual scene. These points are passed to an MLP to process the visual scene. While this approach is simple in design and does not require a log-polar representation, it is likely impractical for real images which have much richer visual information than simulated visual scenes. (Balasuriya and Siebert, 2003) apply difference-of-gaussian and Gabor filters to a Software Retina (Balasuriya, 2006). Support regions are constructed using distance on the Delaunay triangulated graph of the sampling locations. Spatial offsets of points in the support relative to the centre of the support are used to compute filter weights with the analytic expressions for difference-of-gaussian and Gabor filters. This method is similar to the Monte-Carlo convolutions used in point clouds (Section 2.2)

Log-polar representations are the prevalent approach to integrating foveated images into CNNs. (Esteves et al., 2017) extend spatial transformer networks (Jaderberg et al., 2015) to use the log-polar transform allowing them to fixate the high-resolution region on objects of interest. (Amorim et al., 2018) evaluate the rotation invariance of log-polar image representations in conjunction with CNNs. (Kim et al., 2020) similarly evaluate scale and rotation in-

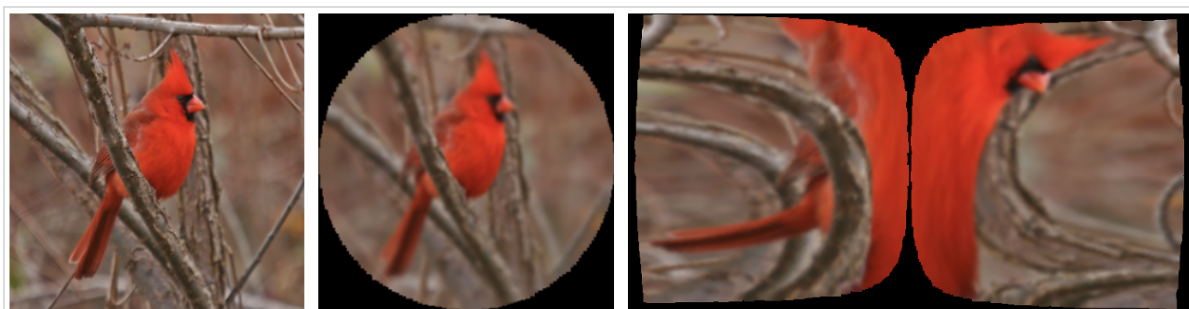


Figure 1: Left to Right: An image from the CUB-200-2011 Dataset of a cardinal bird, represented as a standard uniform image grid. The image sampled by a 8192 receptive field Software Retina (Balasuriya, 2006), resampled to a uniform grid. The Software Retina sampled image represented in cortical space ((Schwartz, 1980)), a biologically plausible variant of the log-polar transform.

variance of log-polar and polar images with wrap around padding. In both cases, network performance suffered under the log-polar and polar image representations, only benefiting in the specific case of training on non-rotated images and testing on rotated images. Interestingly, networks that had been trained on rotated images and tested on rotated images outperform log-polar images as well. (Ozimek et al., 2019) evaluate a more biologically plausible variant of the log-polar transform, dubbed the cortical transform (Schwartz, 1980), applied to Balasuriya’s Software Retina (Balasuriya, 2006). Ozimek et al. again reported decreased performance for cortical images when processed by a CNN for image classification. Ozimek et al. also evaluated the foveated sensor resampled to a uniform grid in cartesian space and showed an increase in performance more comparable to the full resolution image. This suggests that much of the performance decrease is from the log-polar space and not reduced information through foveated sampling.

2.2 Deep Learning on Point Clouds

2D images can be considered a specific case of 2D point clouds where diagnostic information is only carried in the RGB values attached to each point and not in the spatial relationships between points. While in the conventional uniform sampling setting of 2D images, the application of convolutions is possible using the standard discrete convolution. However, this is not the case when the image is sampled non-uniformly. For this reason, methods developed for performing deep learning on point clouds are of particular relevance to performing deep learning on foveated images. In this subsection, we give a general overview of deep learning architectures designed for processing point clouds.

PointNet (Qi et al., 2017a) stands as one of the first successful attempts in applying deep learning architectures directly on point clouds. Shared MLPs are applied in a point-wise fashion to each point in the set, projecting them into a higher dimensional space. These features are aggregated spatially with a global max-pooling operation before being passed to further layers for classification. Due to the global spatial aggregation, PointNet cannot capture local features in the input signal, only global features. PointNet++ (Qi et al., 2017b) addresses this by applying a PointNet to local neighbourhoods on the point cloud to capture local information hierarchically.

(Wang et al., 2018) use Monte-Carlo integration to estimate the convolution integral where points from the point cloud serve as random samples of the underlying continuous signal. Support regions are constructed on a point cloud using nearest neighbours. The spatial offset of each point in the support region, relative to the centre of the support region, is passed to a coordinate-based MLP to produce the filter weights for the corresponding spatial location. A weighted sum between point features and filter weights is then computed to estimate the convolution response at that location. (Hermosilla et al., 2018) adopt a similar approach, but inversely weight samples with their probability density function (PDF) given the support region, obtained through kernel density estimation (KDE). This method is a more accurate approximation of Monte-Carlo integration and allows the estimation of the convolution response to be more robust to non-uniform sampling. (Wu et al., 2019) similarly employ inverse weighting through a point’s PDF but first pass the PDF to a parameterised MLP to allow the network to decide how to use PDFs to inversely weight samples. In our use case, robustness to non-uniform sampling is not as pertinent as in point clouds as a local support region on our foveated sensor is approximately uniform.

(Wang et al., 2018) (Hermosilla et al., 2018) (Wu et al., 2019) use ReLU MLPs to map spatial offsets to filter values. (Xu et al., 2021) instead map spatial offsets to a vector applied with the Softmax activation function using an MLP. This vector is used to softly combine a set of weights stored in a weight bank into a single weight matrix for that specific spatial location. (Thomas et al., 2019) adopt a similar approach but use a predetermined interpolation function to combine the weights of a weight bank into a position-specific weight matrix.

While the methods presented by (Xu et al., 2021), (Thomas et al., 2019) outperform coordinate based MLP approaches, we believe the performance discrepancy can be partially explained by adjacent work in implicit neural representations. That is, (Tancik et al., 2020) show that coordinate based MLPs with ReLU activations have a spectral bias to low-frequency functions. In Monte-Carlo convolutions, the coordinate-based MLP is an implicit neural representation of a filter bank. (Mildenhall et al., 2020) overcome this bias by first applying a positional encoding to the coordinates, significantly improving the networks ability to model high-frequency functions. SIRENs (Sitzmann et al., 2020) similarly show the benefits of periodic activations when modelling continuous signals by using an MLP with Sine activations.

3 METHOD

3.1 Foveated Sensor

To produce foveated images from uniform images, we simulate a foveated sensor in software using the method proposed by (Balasuriya, 2006). A self-similar neural network (Clippingdale and Wilson, 1996) is used to produce a foveated sampling pattern that serves as the centres for overlapping Gaussian receptive fields. These receptive fields are used to sample a uniform image and output a list of RGB values along with their corresponding receptive fields 2D spatial location in cartesian space. We use this approach to foveated sensing as the Gaussian receptive fields remove aliasing in the output image; however, our method for performing convolutions is aimed to be largely agnostic to the foveated sampling strategy. Our foveated sensor has 8192 receptive fields with a fovea radius of 0.1 to sample a 256x256 image.

To compare our method with previous methods, we use the cortical transform proposed by (Schwartz, 1980) to produce a compact, dense representation of Balasuriya’s Software Retina.

3.2 Non-uniform Convolution

Our approach to convolution is the same in concept to that of (Hermosilla et al., 2018), (Wu et al., 2019), (Wang et al., 2018). We frame convolution as the Monte-Carlo estimation of convolution integral and represent convolution kernels continuously through a parameterized MLP. Our novel contribution comes from how we design the MLP (section 3.4), not the overall operation. The integral definition of the 2D continuous convolution operation is given by equation. 1, and the Monte-Carlo estimation of the convolution operation is given by equation. 2

$$f * g(x, y) = \int_{n=-\infty}^{\infty} \int_{m=-\infty}^{\infty} f(n, m) \cdot g(x - n, y - m) \partial n \partial m \quad (1)$$

$$f * G(x, y) \approx \sum_{\delta(x, y) \in S} f\left(\frac{\delta(x, y)}{r}\right) \cdot G(x + \delta x, y + \delta y) \quad (2)$$

Where f is the continuous convolution kernel represented by a coordinate-based MLP, G is the discrete function of the foveated image, S is the set of spatial offsets of points in a given support region relative to the support centre, and r is the radius of the support region.

(Hermosilla et al., 2018) and (Wu et al., 2019) inversely weight each sample by its probability density function. While this is important in point clouds due to the potential for the sampling to be highly non-uniform, this is less important in our use case as we can guarantee an approximately uniform sampling for any small local neighbourhood of the foveated sensor. Consequently, we avoid this inverse weighting as it adds unnecessary computation to the process. For foveated sensors, such as that used by (Nakada et al., 2018), which have a less uniform sampling scheme, the inclusion of inverse weighting by PDF may prove beneficial.

3.3 Support Regions

The support region of the standard discrete 2D convolutional operation is defined by the height and width of the kernel used for convolution. To define a support region on a non-uniformly sampled image, we use a nearest-neighbours search to find the k nearest neighbours to the centre of our receptive field where k is a hyperparameter analogous to filter size in standard convolutions. Unlike point clouds, the spatial locations of points are consistent across all input images

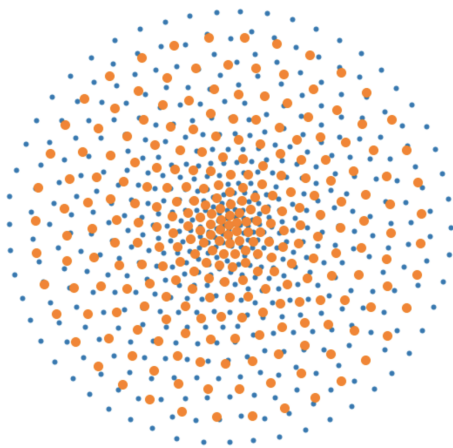


Figure 2: The pixel locations of the foveated image (blue) and the support region centres established over the foveated image (orange). Both are generated by a Self-Similar Neural Network ((Clippingdale and Wilson, 1996). The ratio between the number of pixels and the number of support regions determines the downsampling factor the convolution operation provides.

meaning we can compute nearest neighbours once at initialization of the network, removing the need for a nearest neighbours search in the forward pass of the network. To maintain foveated sampling in the intermediate feature maps, support centre locations use the sampling locations of the input feature map or the sampling locations of a lower resolution Software Retina to perform downsampling (Figure 2).

For every support region, a shared MLP is applied to the spatial offsets of the support points relative to the support centre, normalized by the maximum spatial offset (equivalent to the radius of the support region). Utilizing the normalized spatial offsets produces translation-invariant filter weights and scale the filter appropriately for the sampling resolution. Evaluating the MLP for all sampling locations in the support produces a $k \times I \times O$ weight tensor representing the filter, where k is filter size, I is the depth of the input feature map, and O is the number of filters in the convolutional layer. The filter response is computed as the weighted sum of the filter and the feature values of the support region. In accordance with the integral definition of the convolution operation, this method is equivalent to the Monte-Carlo estimation of the convolution integral.

3.4 Continuous Filter Representations

We use a coordinate-based MLP to map the spatial offsets of pixels relative to the receptive field centre to filter coefficients. (Hermosilla et al., 2018), (Wu et al., 2019), (Wang et al., 2018) use the ReLU ac-

tivation function for their MLPs, however (Sitzmann et al., 2020), (Tancik et al., 2020) show that coordinate based MLPs struggle to effectively learn high-frequency functions when using the ReLU activation function. This may impede the CNNs ability to learn expressive filters for higher frequency signals. (Tancik et al., 2020), (Mildenhall et al., 2020) demonstrate that the mapping of coordinates first through multiple sinusoids of different frequencies allows the MLP to learn higher frequency functions. (Tancik et al., 2020), (Sitzmann et al., 2020) scale the input data by some scaling factor to increase the range of frequencies the MLP can easily learn.

Our MLP design uses one hidden layer of size 64 with the Sine activation function and outputs $I \times O$ where I is the depth of the input featuremap and O is the depth of the output featuremap. Like (Sitzmann et al., 2020) (Tancik et al., 2020), we apply a scaling factor to the first layer weights. This scaling factor is a hyperparameter with 6 giving the best network performance (Figure 4). We draw our first layer weights from a uniform distribution of -1 to 1.

In the case of a Sine activated MLP with one hidden layer, the functionality of this network is similar to the synthesis of a function through a Fourier series. In this case, input layer weights, bias terms, and output weights correspond to the frequency, phase and amplitude of the Fourier basis functions. We find it suffices to freeze first layer weights at initialization and only learn the bias terms. Provided that the first layer weights cover a sufficient frequency range uniformly, this should provide the network with a series of sine basis functions suitable for approximating continuous functions without the need to learn their frequencies. Freezing the bias terms proved detrimental to network performance, suggesting that learning appropriate phase offsets is crucial to the modelling of continuous functions for the size of the MLP we use.

3.5 CNN Architecture

The overall CNN architecture for all experiments uses 5 convolutional blocks where a block has the form of Conv-BatchNorm-ReLU. Each convolution layer uses a receptive field size of 9 nearest neighbours. At each block we double the number of filters, giving an overall signature of (32, 64, 128, 256, 512). The final feature maps are global average pooled in the spatial dimension before being flattened and passed to a fully connected layer that outputs class predictions under the softmax activation. Each convolution block reduces the spatial dimensionality by a factor of 4. For experiments that use standard convolutions,

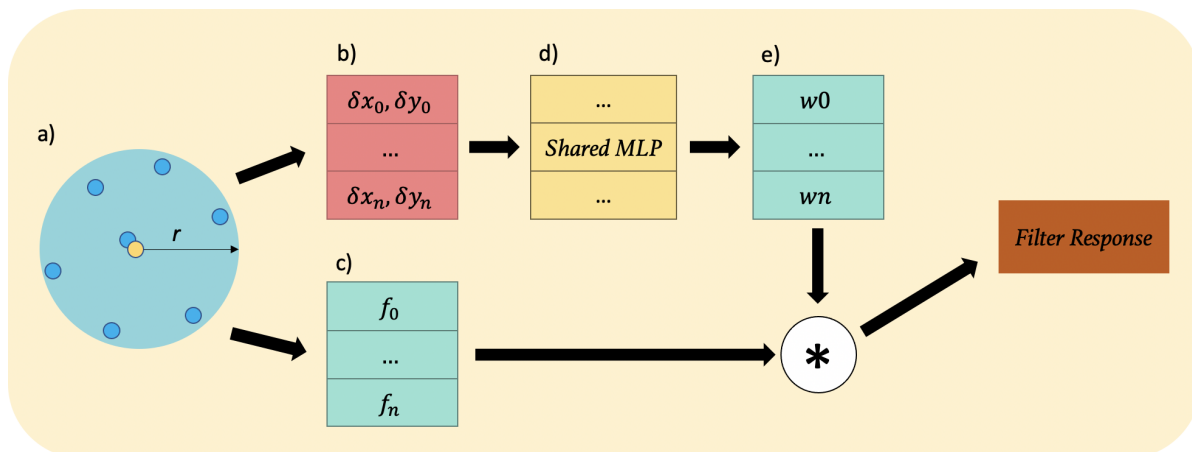


Figure 3: A demonstration of the Monte-Carlo convolution process. **a)** The local support region established on the foveated image using 7 nearest neighbours from the support centre (yellow point). **b)** the spatial offsets of all points in the support region from the support region centre, normalized by radius r . **c)** the features associated with each point in the support. **d)** normalized spatial offsets are passed to a shared MLP to produce the filter weights. **e)** these weights are used to perform a weighted sum of the point features and in turn produce the filter responses for that support region.

we use a kernel size of 3×3 and a stride of 2 to make a fair comparison between the discrete convolution and Monte-Carlo Convolution.

3.6 Training and Dataset

We evaluate our method on the CUB-200-2011 Dataset (Wah et al., 2011), a fine-grained classification dataset comprised of 11,788 images of 200 species of birds. The dataset has a train-test split provided by the dataset creators (an approximately 50-50 split). We further split the test set into a test and validation set with a 4:1 ratio respectively. We choose the CUB-200-2011 dataset as it is challenging due to its fine-grained nature and will require expressive convolution filters to disambiguate the classes.

All images are normalized to the range of 0-1 and resized to 256×256 . At training time we apply random rotations of ± 20 degrees, randomly flip horizontally, and randomly crop 80% of the image and resize to 256×256 . This data augmentation scheme produces translation, rotation and scale transformations of input data. Depending on the image representation used, the subsequent CNN will have a degree of invariance to these transformations. We assume that the augmentation scheme is equally useful for all image representations as they are either invariant to translation or rotation but never both. We train for 100 epochs using the Adam optimizer (Kingma and Ba, 2014). We take the best performing model across all epochs as the final model and report its performance on the held-out test set as the final performance.

Table 1: Classification performance on the CUB-200-2011 Dataset using Monte-Carlo Convolutions. Bracketed numbers refer to the width of each layer of the network. SIREN refers to the networks proposed by (Sitzmann et al., 2020), while ReLU refers to a standard MLP with ReLU activation.

MLP Design	Classification Accuracy (%)
ReLU (32-32)	19.4
SIREN (64-64-64)	29.5
Ours (64)	28.7

4 EXPERIMENTS

4.1 Effect of MLP Design

We evaluate different MLP designs for use within our proposed convolution operation and report how they impact classification accuracy for a CNN trained on the CUB-200-2011 Dataset (Table 4). The role of the MLP within the Monte-Carlo convolution operation is to act as an implicit neural representation of convolution filters. The ReLU MLP, as used in (Wang et al., 2018), (Wu et al., 2019), (Hermosilla et al., 2018), performs significantly worse than SIRENs (Sitzmann et al., 2020) which can be considered state-of-the-art for implicit neural representations. Our proposed MLP design, which can be seen as a special case of a SIREN, uses only one hidden layer and has first layer weights frozen at initialization (not including biases). We show that despite operating under far fewer learnable parameters, there is a negligible performance decrease over the best performing SIREN. These results

suggest that it suffices to represent convolution kernels continuously through a linear combination of different sinusoids and that a many hidden layer SIREN is not strictly necessary to achieve good performance.

4.2 Scaling Factor Hyperparameter

An important hyperparameter in the MLP is the scaling factor applied to the input coordinates (equivalent to the omega hyperparameter in SIRENs (Sitzmann et al., 2020)). Figure 4 shows how classification accuracy changes under different scaling factors. The CNN’s classification accuracy is highly dependent on this hyperparameter with 6 providing the best performance. The scaling factor is closely tied to the frequency range the MLP can represent. The sampling rate of the continuous convolution filter increases with kernel size. Therefore, it is likely that the scaling parameter should be tuned in accordance to the convolutional layer’s kernel size. We do not explore this further in this paper, however the importance of this hyperparameter makes this an important avenue to explore in future work.

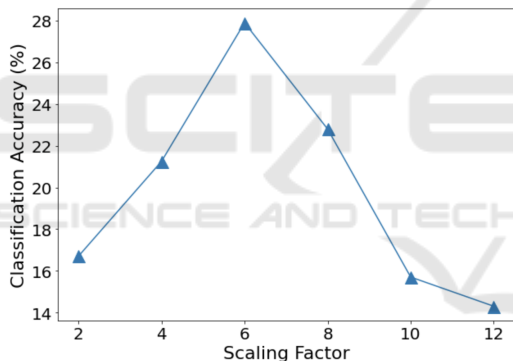


Figure 4: The effect of changing the input scaling factor of the MLP in our proposed convolution operation on classification accuracy on the CUB-200-2011 Dataset.

4.3 Image Representations

We compare classification accuracy when operating on the foveated image in cartesian space against a foveated image represented in log-polar space (Table 4.3). We provide additional context for performance by also reporting classification accuracy on the uniform image before foveated sampling. For the foveated image in cartesian space we use our proposed Monte-Carlo Convolution operation. For all other experiments standard 2D discrete convolutions are applied. We show that classification accuracy on the cartesian foveated image when using Monte-Carlo convolutions is significantly higher than standard convolutions applied to the log-polar foveated

Table 2: Classification performance on the CUB-200-2011 Dataset. Unless specified to use MC-Conv, all results are obtained with a CNN using standard convolutions. Results are averaged over 5 training runs.

Image type	Classification Accuracy (%)
Uniform (256x256)	33.1
Foveated - Cortical Transform	22.0
Foveated w/ MC-Conv (Ours)	28.7

image. Both foveated images carry the same amount of information but differ in their representation, suggesting that the performance discrepancy is simply due to the log-polar space being a poor image representation for this task. Operating on the cartesian foveated image also produced results comparable to that of standard convolutions operating on the uniform image suggesting that our proposed convolution operation can achieve similar representational power to standard convolutions.

5 CONCLUSION

In this paper, we propose a method for performing convolutions on non-uniform foveated images in the cartesian coordinate frame. Our approach uses the Monte-Carlo estimation of the convolution operation between the foveated image and a continuous filter kernel represented by a coordinate MLP. Unlike previous approaches, which use an MLP with ReLU activations, we adopt Sine activation functions which have been shown to significantly improve our model’s ability to learn filters that can extract diagnostic features from a foveated image. Our approach significantly outperforms standard convolutions applied to foveated images represented in a log-polar space and approaches performance comparable to applying standard convolutions to the full resolution uniform image.

6 LIMITATIONS

The computational overhead of the computer vision system operating on the foveated image should be less than operating on the full resolution image. A standard convolution layer requires a $k \times I \times O$ weight tensor to represent its convolution filters, where k is the size of the filter, I is the number of input channels, and O is the number of filters. Our convolution operation requires $k \times I \times O \times N$, where N is the number of support regions established on the featuremap. This

factor N increase significantly increases the memory overhead of our operation beyond the memory savings achieved through a reduced image size through foveated sensing. In future work, we would like to address this limitation through factorization methods to achieve memory savings that make operating on foveated images in cartesian space viable from a computational overhead point of view.

REFERENCES

- Amorim, M., Bortoloti, F., Ciarelli, P. M., de Oliveira, E., and de Souza, A. F. (2018). Analysing rotation-invariance of a log-polar transformation in convolutional neural networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE.
- Balasuriya, L. and Siebert, J. (2003). A low level vision hierarchy based on an irregularly sampled retina. In *Proceedings of the International Conference on Computational Intelligence, Robotics and Autonomous Systems, Singapore*.
- Balasuriya, S. (2006). *A computational model of space-variant vision based on a self-organised artificial retina tessellation*. PhD thesis, University of Glasgow.
- Clippingdale, S. and Wilson, R. (1996). Self-similar neural networks based on a kohonen learning rule. *Neural Networks*, 9(5):747–763.
- Esteves, C., Allen-Blanchette, C., Zhou, X., and Daniilidis, K. (2017). Polar transformer networks. *arXiv preprint arXiv:1709.01889*.
- Hermosilla, P., Ritschel, T., Vázquez, P.-P., Vinacua, À., and Ropinski, T. (2018). Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (TOG)*, 37(6):1–12.
- Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025.
- Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.
- Kim, J., Jung, W., Kim, H., and Lee, J. (2020). Cyclic: a rotation invariant cnn using polar mapping and cylindrical convolution layers. *arXiv preprint arXiv:2007.10588*.
- Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Li, X., Jie, Z., Wang, W., Liu, C., Yang, J., Shen, X., Lin, Z., Chen, Q., Yan, S., and Feng, J. (2017). Foveanet: Perspective-aware urban scene parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 784–792.
- Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer.
- Nakada, M., Chen, H., and Terzopoulos, D. (2018). Deep learning of biomimetic visual perception for virtual humans. In *Proceedings of the 15th ACM Symposium on Applied Perception*, pages 1–8.
- Ozimek, P., Hristozova, N., Balog, L., and Siebert, J. P. (2019). A space-variant visual pathway model for data efficient deep learning. *Frontiers in cellular neuroscience*, 13:36.
- Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.
- Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*.
- Schwartz, E. L. (1980). Computational anatomy and functional architecture of striate cortex: a spatial mapping approach to perceptual coding. *Vision research*, 20(8):645–669.
- Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33.
- Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*.
- Thomas, H., Qi, C. R., Deschard, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420.
- Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The caltech-ucsd birds-200-2011 dataset.
- Wang, S., Suo, S., Ma, W.-C., Pokrovsky, A., and Urtasun, R. (2018). Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597.
- Weiman, C. F. and Chaikin, G. (1979). Logarithmic spiral grids for image processing and display. *Computer Graphics and Image Processing*, 11(3):197–226.
- Wu, W., Qi, Z., and Fuxin, L. (2019). Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630.
- Xu, M., Ding, R., Zhao, H., and Qi, X. (2021). Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3173–3182.