Attribution of Predictive Uncertainties in Classification Models (Supplementary material)

Iker Perez[1] **Piotr Skalski**[1] **Alec Barns-Graham**[1] **Jason Wong**[1] **David Sutton**[1]

[1]Featurespace Research, Cambridge, United Kingdom

1 BAYESIAN PRESENTATION

In Bayesian settings, model uncertainties are often decomposed across *aleatoric* and *epistemic* components that help scrutinise different aspects in the functioning of a model, and can facilitate interpretability or fairness assessments in important machine learning applications [Awasthi et al., 2021]. Hence we may wish to offer attributions that are representative of isolated types of uncertainties.

On training a neural classifier $f : \mathbb{R}^n \times \mathcal{W} \to \Delta^{|\mathcal{C}|-1}$ within an (approximate) Bayesian setting, we commonly obtain a *posterior* over the hypothesis space of models, i.e. a distribution $\pi(\boldsymbol{w}|\mathcal{D})$ over model weights conditioned on the available train data $\mathcal{D} = \{\boldsymbol{x}_i, c_i\}_{i=1,2,\ldots}$. Popular approaches to procure such posterior often differ in their approach to incorporate *prior* knowledge and include *dropout* [Srivastava et al., 2014], *Bayes-by-Backprop* [Blundell et al., 2015] or SG-HMC [Springenberg et al., 2016]. Here, a model score for a new data point $\boldsymbol{x}^\star \in \mathbb{R}^n$ is derived from the *posterior predictive distribution* by marginalising over posterior weights, i.e.

$$\pi(\boldsymbol{x}^\star|\mathcal{D}) = \int_{\mathcal{W}} f(\boldsymbol{x}^\star, \boldsymbol{w})\pi(\boldsymbol{w}|\mathcal{D})d\boldsymbol{w} = \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[f(\boldsymbol{x}^\star, \boldsymbol{w})],$$

and is easily approximated as $\frac{1}{N}\sum_{i=1}^N f(\boldsymbol{x}^\star, \boldsymbol{w}_i)$, with weight samples $\boldsymbol{w_i} \sim \pi(\boldsymbol{w}|\mathcal{D})$, $i = 1, \ldots, N$. This setting is analogue to the presentation in Section 2, however, the point estimate score must now be averaged over the posterior, i.e. $f(\boldsymbol{x}) = \pi(\boldsymbol{x}^\star|\mathcal{D}) = \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[f(\boldsymbol{x}^\star, \boldsymbol{w})]$

The *entropy* is thus given by

$$H(\boldsymbol{x}|\mathcal{D}) = -\sum_{c\in\mathcal{C}} \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[f_c(\boldsymbol{x}, \boldsymbol{w})] \cdot \log \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[f_c(\boldsymbol{x}, \boldsymbol{w})],$$

and may be decomposed through the law of iterated variances [Kendall and Gal, 2017] so as to yield an *aleatoric* term

$$\begin{aligned} H_a(\boldsymbol{x}|\mathcal{D}) &= \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[H(\boldsymbol{x}, \boldsymbol{w})] \\ &= -\sum_{c\in\mathcal{C}} \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}\big[f_c(\boldsymbol{x}, \boldsymbol{w}) \cdot \log f_c(\boldsymbol{x}, \boldsymbol{w})\big], \end{aligned}$$

which measures the mean predictive entropy across models in the posterior hypothesis space, as well as the *mutual information* or *epistemic* term, $H_e(x|\mathcal{D}) = H(\boldsymbol{x}|\mathcal{D}) - H_a(\boldsymbol{x}|\mathcal{D})$ that represents model uncertainty projected into the latent membership vector $\pi(\boldsymbol{x}|\mathcal{D})$. Intuitively, aleatoric uncertainty represents natural stochastic variation in the observations over repeated experiments; on the other hand, epistemic uncertainty is descriptive of model unknowns due to inadequate data or inappropriate modelling choices.

Path integrals. The *posterior predictive classifier* $\pi(\boldsymbol{x}|\mathcal{D})$ accepts a path importance for an arbitrary scalar output $F(\boldsymbol{x}, \boldsymbol{w})$ at index i, given by

$$\text{attr}_i^\delta(\boldsymbol{x}) = \int_0^1 \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}\left[\frac{\partial F(\delta(\alpha), \boldsymbol{w})}{\partial \delta_i(\alpha)}\right]\frac{\partial \delta_i(\alpha)}{\partial \alpha}d\alpha.$$

This represents a *mean-average* trajectory over a curve δ and follows from *dominated convergence*. This easily amends to the attribution of uncertainties, i.e.

$$\text{attr}_i^\delta(\boldsymbol{x}) = -\sum_{c\in\mathcal{C}} \int_0^1 \Delta_i(\alpha)\frac{\partial \delta_i(\alpha)}{\partial \alpha}d\alpha$$

which is defined s.t.

$$\Delta_i(\alpha) = \big(1 + \log \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}[f_c(\delta(\alpha), \boldsymbol{w})]\big) \cdot \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}\left[\frac{\partial f_c(\delta(\alpha), \boldsymbol{w})}{\partial \delta_i(\alpha)}\right]$$

If we wish to only attribute aleatoric uncertainties, we may replace the above for

$$\Delta_i(\alpha) = \mathbb{E}_{\boldsymbol{w}|\mathcal{D}}\left[\big(1 + \log f_c(\delta(\alpha), \boldsymbol{w})\big) \cdot \frac{\partial f_c(\delta(\alpha), \boldsymbol{w})}{\partial \delta_i(\alpha)}\right].$$

Finally, attributions for any variation in epistemic uncertainty is readily shown to be *explained* as the difference in attributions between full and aleatoric uncertainties. We showed an example of this in Figure 1 within Section 2.

2 ROBUSTNESS TO CHANGES IN THE AUTOENCODER

In Table 1 we show evaluations of performance metrics for the attribution method proposed in this paper, over resampled Mnist and Fashion validation images. We train multiple variational autoencoders and use them as the generative process to define integration paths in our method. These differ in the dimensionality of the latent space used to encode reduced representations of images. This is the most impactful layer for the functioning of the attribution method we have presented, since straight integration lines are defined in this space and later projected into pixel space. Too small or large a space could lead to out of distribution images and integration paths. Additionally, we also experiment with altering the data augmentation mechanism used for modifying images prior to training the autoencoder (results are reported at latent space dimension of 32). No significant changes in performance where noticed as training regimes and learning rates were modified.

In the table, we notice consistent performance which plateaus after a certain threshold, which is equivalent in these two data sets. Consistency in performance is a consequence of the regularisation term in latent space observed in (2). This tunes fiducial points and integration paths strictly in distribution, even if large latent spaces overparametrise the encoding space.

Table 1: Performance metrics for generative attribution method, for architecture variations in the autoencoder.

Setting	Area over EIC		Uncertainty Reduction Curve			
	Mnist	Fashion	Mnist		Fashion	
			1%	5%	1%	5%
4	0.999	0.918	0.474	0.738	0.165	0.350
8	0.999	0.916	0.661	0.845	0.192	0.374
16	0.999	0.919	0.704	0.846	0.196	0.393
32	0.999	0.925	0.743	0.868	0.204	0.395
- Aug	0.999	0.930	0.687	0.876	0.184	0.403
64	0.999	0.922	0.752	0.877	0.203	0.392
128	0.999	0.925	0.756	0.879	0.206	0.400
259	0.999	0.927	0.762	0.884	0.204	0.405

3 EXAMPLES

In Figure 1 we find examples of attributions of aleatoric and epistemic uncertainty types, applied to dog versus cats images. Attributions are produced by vanilla integrated gradients as described in Section 2. Saliency masks are combined with a Gaussian kernel in order to draw attention to regions in images associated with different uncertainty types. Similarly, Figure 2 shows attributions of uncertainty types across selected Mnist images, produced by the generative method presented in this paper.

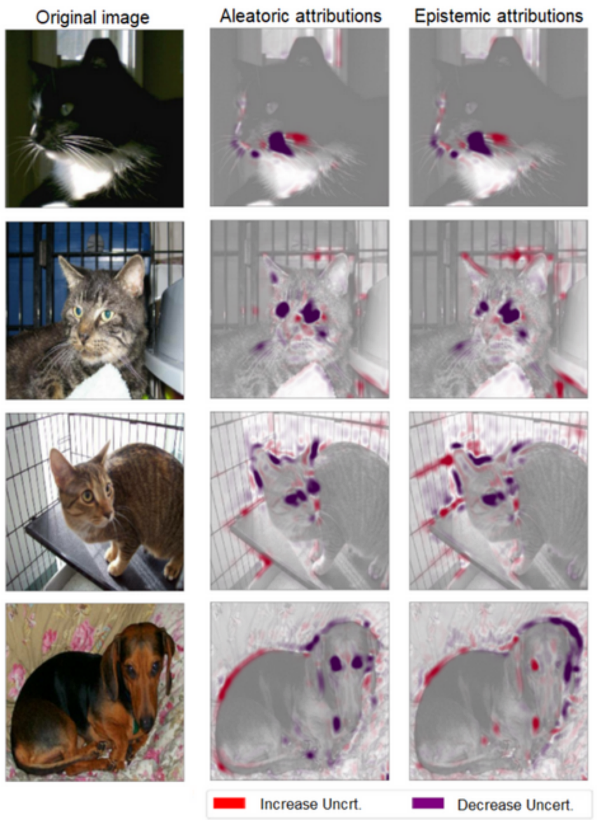

Figure 1: Aleatoric and epistemic contributions to uncertainty for a classification task in *dogs versus cats* data.

3.1 QUALITATIVE EVALUATIONS

Finally, in Figure 3 we show uncertainty attribution masks across a range of classification tasks, on all the data sets explored in this paper. In all cases, we note that attributions relying on counterfactual mechanisms are humanly interpretable. Further integration of counterfactual methods with path integrals ensures that attributions are isolated to few pixels. In application to human gestures, these are always restricted to facial features around the mouth, cheeks or eyebrows, depending on the classification task. On the contrary, vanilla attributions through integrated gradients (averaged over *black and white* fiducial baselines) are noticeably noisy.

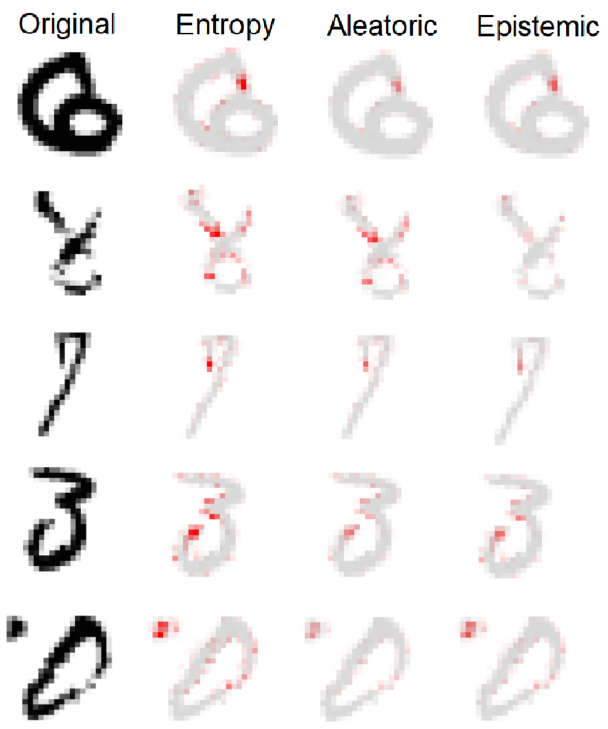

Figure 2: Aleatoric and epistemic contributions to uncertainty for a classification task with MNIST digits.

Also, segmentation based mechanisms do not perform well in the data sets we have explored, which do not contain multiple objects that can be easily segregated.

4 IMPLEMENTATION DETAILS

All of our predictive models are implemented through Keras. The following is a summary of architectures, hyper-parameters, training regimes and further details.

4.1 MNIST HANDWRITTEN DIGITS

Our classifier is a convolutional neural network with *max-pooling* layers and dropout, structured as:

- Two convolutional layers of kernel size 3×3 and *relu* activation; *filter counts* are 32 and 64 for the first and second layers. We use *stride* length of 1 followed by *max-pooling* layers of *pool size* 2×2.
- The output is flattened and fed through a *dense* layer of 128 neurons with *relu* activation, followed by dropout with deactivation rate of 0.5, and a final *softmax* regression layer for categorical outputs.

We train to minimize the *categorical cross entropy* wrt the train labels, using the *Adam* optimizer, over 10 epochs, with a constant learning rate of $1e^{-3}$ and with *batch size* of 32.

The **variational autoencoder** relies on convolution and deconvolution layers. The encoder is structured as:

- Two convolutional layers of kernel size 3×3, stride 2 and *relu* activation; *filter counts* are 32 and 64 for the first and second layers.
- A *dense* layer of 128 neurons, with *relu* activation.
- Two *dense* layers mapping the 128 neurons to a distributional mean vector and a log-standard-deviation vector, for the latent space for an image. Dimension of the latent space varies in order to assess robustness, see Appendix 2 for details.
- A random *sampling* operation from a normal distribution, with the afore-defined distributional parameters.

In addition, the decoder is defined as:

- A dense layer with *relu* activation, mapping a latent element to a vector of dimensionality $7 \times 7 \times 64$.
- Two deconvolutional layers of kernel size 3×3, stride length 2 and *relu* activation; the *filter counts* are 64 and 32 for the first and second layers.
- An output deconvolutional layer of kernel size 3×3, *filter counts* 1, stride length 1 and *sigmoid* activation for pixel values.

The autoencoder is fitted to minimize a custom loss, with a reconstruction term (through a cross-entropy loss) and the Kullback-Leibler divergence among latent mappings and a normal distribution $\mathcal{N}(\mathbf{0}, I)$. We use the *Adam* optimizer, over 50 epochs, with a constant learning rate of $1e^{-3}$ and with *batch size* of 32.

4.2 FASHION-MNIST DATASET

The classifier and autoencoder are defined similarly to the above example. However, we add two additional *dropout* layers (with probability 0.5) after each *max-pooling* operation in the classifier. Training proceeds with the *Adam* optimizer, at a constant learning rate of $1e^{-3}$ with *batch size* 32. The classifier is trained for 10 epochs using the cross-entropy as the cost function. The autoencoder is trained for 50 epochs using a combination of binary cross-entropy and the Kullback-Leibler divergence as a regularisation term.

4.3 CELEBA DATASET

Images are centred around the face and cropped to size 128×128, further standardized to pixel values in the range $[0, 1]$. During training, we leverage data augmentation with random rotations; we use a *maximum angle* of ± 18 degrees, random translation by a maximum factor of 0.1 and random horizontal flip.

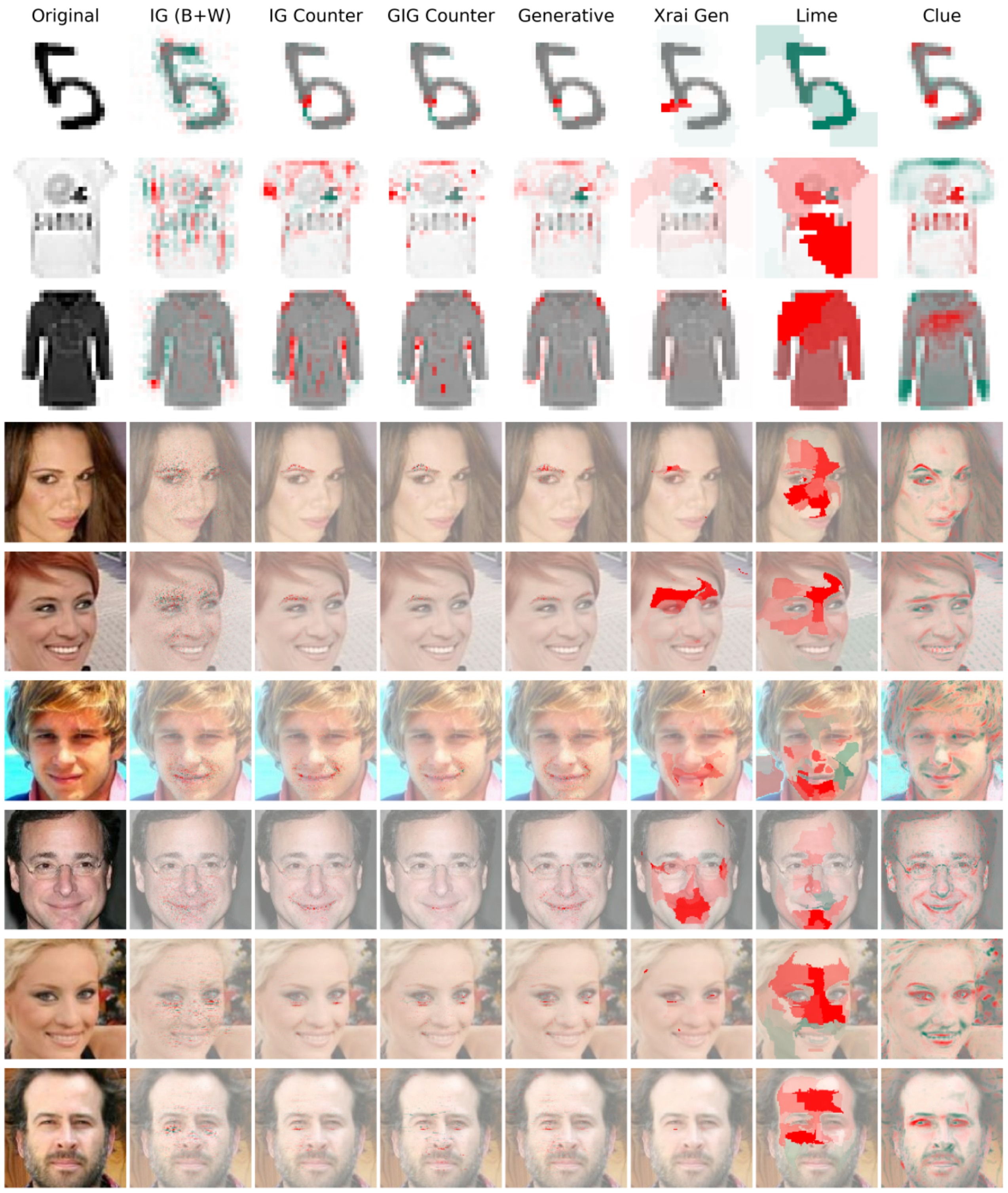

Figure 3: Uncertainty attribution masks across multiple classification tasks and data sets. We display best performing attribution methods with a counterfactual mechanism.

The **classifier** is composed of 6 convolutional blocks followed by a dense layer with *softmax* activation. Each convolutional block utilizes a *kernel* size of 3 and *stride* 1, along with *batch normalization*, *dropout* with deactivation probability of 0.2, *relu* activation and *max-pooling* (*pool size* 2 and *stride* 2). The number of channels in convolutional layers is, respectively, 32, 64, 128, 128, 256 and 256. The last block is followed by a flattening operation and a *dropout* layer with deactivation probability 0.4.

We train this classifier for 5 epochs using the *Adam* optimizer with batch size 64 and the *cross-entropy* as cost function. The learning rate is decreased after each epoch by a factor of 0.8; starting from $1e^{-4}$ for the *smiling* and *arched eyebrows* classifiers, and $3e^{-5}$ for the *bags under eyes* classifier.

The **encoder** in the variational autoencoder is a series of 5 convolutional blocks. Each block shares the same structure, with *kernel* size 3, *stride* 2, *batch normalization* and *leaky-relu* activation with negative slope coefficient of 0.3. The number of filters at the output of each block is 32, 64, 128, 256 and 512. After the last block we insert a flattening layer and two dense layers each with 256 output neurons for the distributional mapping to the latent space. The **decoder** is a fully connected dense layer with 80192 output neurons (reshaped into a $4 \times 4 \times 512$ activation map) followed by 5 up-sampling blocks. Each block up-samples the input by a factor 2 and feeds it into a convolutional layer with kernel size 3 and stride 1, followed by *batch normalisation* and *leaky-relu* activation with 0.3 negative slope coefficient. The number of channels at the output of each block are 256, 128, 64, 32 and 3 respectively. We apply an additional convolutional layer with kernel size 3, stride 1, 3 output channels and *sigmoid* activation for a final reconstructed RGB image with values restricted in the $[0, 1]$ interval.

The autoencoder is trained for 100 epochs using the *Adam* optimizer, with batch size 64 and a learning rate of $5e^{-4}$ which is decreased after each epoch by a factor of 0.98. We use a *perceptual loss* function together with the Kullback-Leibler divergence regularisation term, following details on [Hou et al., 2017] (VAE-123 model).

4.4 ATTRIBUTION METHODS

We use standard implementations of attribution methods with recommended parameters in corresponding publications or public repositories. In all cases, *black+white* and *counterfactual* variants of methods are implemented equivalently. For path methods requiring trapezoidal integration, we use 50 bins with grayscale images and 25 bins with high resolution images. The process to procure counterfactual fiducials is explained in Section 3.

Vanilla IG is implemented with a straight line as domain of integration.

Blur IG is specified with an integration path which decreases blurring from a masked image, using successive Gaussian filters. The maximum standard deviation is set to the minimum required to maximise the average predictive entropy across train data.

Guided IG is configured s.t. the subset of pixels traversing value in each step is the 10% with smallest partial derivatives of entropy wrt pixel values. We use 50 steps.

LIME is implemented through *quickshift* segmentation, with kernel 1, maximum distance 5 and ratio of 0.2. We use a binomial mask with deactivation probability 0.2, and *Lasso* regression to attribute importances.

SHAP proceeds through $2 * (\text{Pixel Count}) + 2^{11}$ index perturbations of varying size; masked index points are resampled from their corresponding marginal distributions. We use *Lasso* regression to attribute importances.

CLUE attributions are derived as the difference between an image and its decoded CLUE counterpart [cf. Antoran et al., 2021, Appendix F]. The cost function weighs reconstruction and uncertainty terms, and is tuned on a validation set.

Xrai is implemented with Felzenszwalb's segmentation algorithm in order to retrieve masks. We use multiple scale values of 50, 100, 150, 250, 500 and 1200, as well as a dilation radius of 5. This is applied to normalised images at range $[-1, 1]$ and size 224×224 pixels. Resizing is undertaken with anti-aliasing. Segments are accepted for appending into attributions with a required difference of 50 pixels.

References

Javier Antoran, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a CLUE: A method for explaining uncertainty estimates. In *International Conference on Learning Representations*, 2021.

Pranjal Awasthi, Alex Beutel, Matthäus Kleindessner, Jamie Morgenstern, and Xuezhi Wang. Evaluating fairness of machine learning models under uncertain and incomplete information. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 206–214, 2021.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622, 2015.

Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141, 2017.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in neural information processing systems*, 29:4134–4142, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15:1929–1958, 2014.