Explaining Failure: Investigation of Surprise and Expectation in CNNs

Thomas Hartley
Cardiff University
hartleytw@cardiff.ac.uk

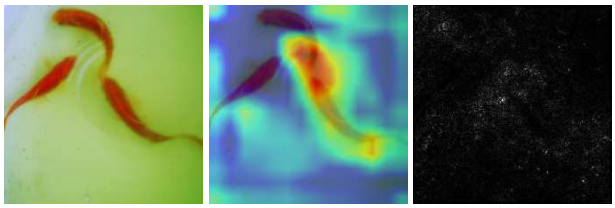
Kirill Sidorov
Cardiff University
sidorovk@cardiff.ac.uk

Christopher Willis
BAE Systems Applied Intelligence
chris.willis@baesystems.com

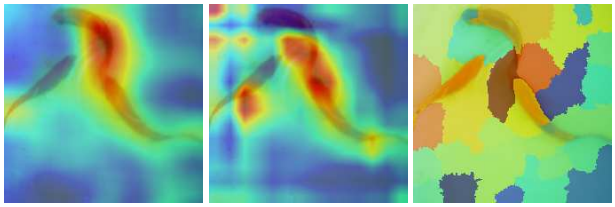
David Marshall
Cardiff University
marshallad@cardiff.ac.uk

Abstract

*As Convolutional Neural Networks (CNNs) have expanded into every day use, more rigorous methods of explaining their inner workings are required. Current popular techniques, such as saliency maps, show how a network interprets an input image at a simple level by scoring pixels according to their importance. In this paper, we introduce the concept of **surprise** and **expectation** as means for exploring and visualising how a network learns to model the training data through the understanding of filter activations. We show that this is a powerful technique for understanding how the network reacts to an unseen image compared to the training data. We also show that the insights provided by our technique allows us to "fix" misclassifications. Our technique can be used with nearly all types of CNN. We evaluate our method both qualitatively and quantitatively using ImageNet.*



Input Grad-CAM [26] Backprop [27]



(c) RISE [22] Grad-CAM++ [3] LIME [23]

Figure 1. An image from the 'Goldfish' class that is misclassified as a 'Roundworm'. Existing visualisation techniques fail to give a clear explanation of the reason for failure. A plausible explanation for the failure is shown in Figure 7.

1. Introduction

Convolutional Neural Networks (CNNs) have revolutionised image classification and detection tasks. However, they are often considered as being "black boxes", with their inner workings difficult to interpret. There is, therefore, a need for techniques that can explain and visualise how a network is interpreting an image. These explanations can take many forms, from high-level saliency maps to visualising the properties of individual filters.

A number of challenges arise when designing a method for visualising a network's explanations. The first is to accurately determine which areas of an image the network is basing its prediction on. The second is to convey how or why the network made its prediction based on the input image. Previous work can be classified into image- or network-centric methods. In image-centric methods, one attempts to assign a weight to each pixel in the input space to produce a saliency map for a specific input. Examples of this are occlusion maps [30], backprop [27], guided back propagation [29], class activated mappings (CAM) [33], Grad-CAM [33], LIME [23], etc. Network-centric methods, on the other hand, attempt to explain a filter's role within the network, often without regard to a specific input. Examples of this are filter visualisations or neuron labelling [2, 16, 29].

These techniques often require the network to predict an image correctly to offer the most valuable insights. An example of this is shown in Figure 1 where the VGG16 network [28] trained on ImageNet [24] fails to classify an image from the validation set correctly. However, the explanations generated for it show the network is focusing on the object. This begs the question: "How do we visually ex-

plain why the classification has failed when existing methods show that the network is using the seemingly relevant features in the object to inform its prediction?". Using our proposed methods we are able to give a plausible explanation as to misclassification of the goldfish (see Figure 7), and then provide a validation of this (see Figure 10).

In this paper, we propose a method that allows us to understand why an image has failed to be classified correctly by bridging the space between image-centric and network-centric explanation techniques. We do this by studying the distributions of filter activations for each class in the training data. Inspired by the idea of surprise in the free-energy principle [8] we compare the filter response to unseen images against these known distributions. This allows us to determine which filters have activated in an unusual way for that class. We consider two types of unusual activation: features causing filters to over activate (***surprise***) and under activate (***expectation***). These surprise and expectation values enable us to both visualise filters that are over or under activated, but to also refer back to the images used for training and pinpoint which features in the training data the network has used to learn its understanding of a given class.

2. Related Work

A number of techniques have been developed to understand and visually explain why or how a network produced a classification. In image-centric techniques this is normally done by producing a score for each pixel to show its importance to the network. Initial work with image-centric techniques included methods such as DeConvNet [30] and the work by Simonyan *et al*. [27] which allowed gradients to be mapped back to the input image. Following on from these, work was done to establish how these could be made more precise through the use of guided backpropagation [29, 26]. Class Activation Maps [33] (CAMs) were developed to allow networks with an average pooling layer to locate regions of interest in an image. These were further generalised with the Grad-CAM [26] technique to work with CNNs regardless of whether they had an average pooling layer or not. Grad-CAM works by using the mean of the gradient to produce a score for each filter in the layer. This is then used to weight and visualise the activation maps. These filter scoring techniques are also found in network pruning literature [12, 17, 19] to identify the importance of each filter. Techniques also exist that attempt to perturb the input image multiple times and observe how the network reacts. Examples of these are Occlusion Maps [30], LIME [23], and Interpretable Explanations [7]. Early examples of network-centric techniques are filter visualisation [4, 29] where patterns learnt by the filters are visualised. More recent work has begun looking at how to apply meaningful labels to each filter to better identify its use within the network [2, 6, 16, 25]. Or to learn how to construct a semantic hierarchy of features within the CNN [31].

There are a number of works that attempt to tie these two areas together. Kim *et al*. [15] aim to produce examples which are either "prototypes' (images which fit the model well) or "criticisms" (images that *do not* fit the model well). This technique uses the Maximum Mean Discrepancy (MMD) to find the difference between two distributions and a witness function to find the largest discrepancies. Kim *et al*. [15] was followed up by Khanna *et al*. [14] who aimed to discover which training examples are most responsible for a set of predictions. This allows for the discovery of which training sample caused the misclassification of an unseen sample. Techniques for comparing distributions of image attributes have also been used to discover underlying biases in datasets [32]. However, this work makes use of dataset with attribution labels [18, 21] rather than object classification labels allowing a relationship between explicit attributes to be discovered. Unlike [14] and [15] who operate on training examples, we aim to identify the underlying filters that cause misclassification.

3. Measuring Surprise and Expectation

In this section we describe the methods required for our *surprise* and *expectation* framework. Firstly, a method of determining a measure of filter salience is presented, followed by a statistical understanding of how filters are used by the training data. We then formally define surprise and expectation.

3.1. Filter Importance Measure

In order to ascertain how much a filter is contributing to the final output of the network, a measure of importance is required. Numerous works have proposed methods of ranking filters for various purposes. For example, Grad-CAM [26] and Grad-CAM++ [3] effectively rank the final convolution layers filters to assign weights prior to visualising. Alternatively, work from the pruning literature have used methods ranging from a Taylor Expansion [5, 19] to simple measures such as the l_1-norm of the activation maps [17]. However, in this work we propose to use the *product* of the gradient and activation map from the final convolution layer of a network. We call this Grad-AMap. More concretely, let $A(n) \in \mathbb{R}^{H \times W}$ be the activation of the n^{th} filter. We can thus measure the importance α_n of the n^{th} filter as the mean of the product of the gradient for a given class from O (the output of network) and its activation map:

$$\alpha_n = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{\partial O(A_{ij}(n))}{\partial A_{ij}(n)} A_{ij}(n). \qquad (1)$$

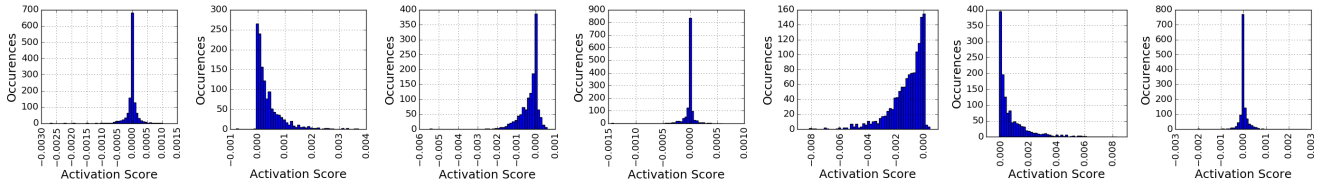


Figure 2. Seven typical distributions of filter importance scores (see Equation 1). Each distribution represents a class filter pair (i.e. class 30, filter 100).

3.2. Building Filter Score Distributions

To build an understanding of how training data is represented by the model, we build a distribution over every filter in the final convolution layer for each class by applying the network to the images. For example, for a 10 class dataset with 512 filters we would gather 5120 distributions. We observe that the distributions created are typically skewed unimodal. Figure 2 shows examples of such distributions generated using VGG16 with ImageNet. From these distributions we can discover the mean activation for every filter in each class: μ_{cn}, where c is the class and n is the filter index. This allows us to encapsulate the range of activations a filter may produce for each class. For example, we would expect a filter that has learnt to activate on dog faces to activate highly when a dog is present and weakly if not. We gather the above statistics using a pre-trained model and all of the training data.

3.3. Surprise and Expectation

With every distribution precomputed for each class and filter combination, we can pass unseen data to the network and observe how α_n compares to μ_{cn}. If the input image is modeled well by the learnt distribution, we should expect α_n to fall somewhere close to μ_{cn}. However, if α_n deviates either side of μ_{cn} we can say that the input data is deviating from the model's expectations. If α_n has a negative deviation we can say that the model was expecting to see the n^{th} filter activate higher. Conversely, if α_n has a positive deviation we can say the model is surprised to see the n^{th} filter activate so highly. As our distributions are not normal, we define surprise and expectation in terms of percentiles. For example we consider a filter surprising if $\alpha_n > P_{cn}(99.9)$ and the filter expecting if $\alpha_n < P_{cn}(0.1)$. Where $P_{cn}(x)$ is the x^{th} percentile, c is the ground truth class and n is the filter index. Values that fall within these percentiles are considered to be within the expected values for that class and filter combination. When visualising filters that are expected ("expectation filters") we ensure that we are only working with filters that have a μ_{cn} value greater than the mean of the precomputed class values: $\frac{1}{N}\sum_{n=1}^{N}\mu_{cn}$. This prevents filters with a very low μ_{cn} from being labelled as expected if an even lower α_n is encountered.

Once a set of surprise or expectation filters have been identified we can rank them using their deviation from the training mean $(\alpha_n - \mu_{cn})$. The more the value deviates from 0, the more the filter is under- or over-performing. We also propose doing this over the entire set of filters for a given image to compute a value, β, that measures how a filter's importance scores, α_n, for an unseen image, deviate from the mean μ_{cn} of the target class training data. For every unseen image in the ImageNet validation set, we subtract the mean of the training data from their importance scores and take the mean of those values:

$$\beta = \frac{1}{N}\sum_{n=1}^{N}(\alpha_n - \mu_{cn}). \tag{2}$$

Large values of β signify that an image is a strongly typical representative of the training set.

4. Experiments

In this section we begin by justifying the use of GradAMap (our filter measurement technique), before investigating our surprise and expectation measures. We then draw these together using our β value to investigate the reasons for failure. Finally we try to understand how we can 'fix' images that fail using the insights we have gained. We apply the proposed method across all images present in ImageNet [24] selecting typical examples for further discussion. We use pretrained models from the PyTorch model zoo [20]. We experiment using two types of network architecture, VGG16 [28], and ResNet [10]. This allows us to compare our proposed methods networks that use BatchNorm [13] and those that do not. In particular for the ResNet architectures, we being by using ResNet18 to test the filter scoring methods, then scale up to show our proposed surprise and exception method works for ResNet50. We use ResNet18 for this experiment as the final convolution layer only contains 512 filters, as opposed to 2,048 in ResNet50 allowing us to explore multiple methods in a timely manner. We take our filter activation score from the final convolution layer of each network.

4.1. Filter Ranking Method

In this experiment, we rank the filters in the selected convolution layer and incrementally zero out the channels of the activation map produced by a networks final convolution layer based on the filter rankings. Zeroing out a single channel of an activation map is the equivalent of pruning the corresponding convolution filter. We increment in steps of 16 filters, for a total of 32 increments for a layer consisting of 512 filters. This is similar to a number of metrics used to evaluate the accuracy of heatmap for CNN explanations, for example, Remove and Retrain (ROAR) [11] or the deletion game [22]. In these methods, pixels are iteratively removed (pixels are set to 0) and the accuracy or softmax is recorded and averaged over all validation images in a dataset. Typically regions are removed from most important to least. As the regions are removed, the accuracy or softmax drops accordingly. We follow this pattern for our experiment, recording the accuracy of the network (which we normalise to be between 0% – 100%) as we remove the most important filters first. We do this for every image in the ImageNet validation set. The better a technique is at producing accurate filter scores, the fewer filters we should be able to remove before the model can no longer accurately predict the correct class and the overall model accuracy drops. We measure this using the Area under the Curve (AUC) value as per the metric used in RISE [22]. An AUC score approaching 0 suggests that a method is able to better rank the importance of filters, as we are able to rapidly reduce the accuracy of the network. For this experiment we use VGG16 and ResNet50 and test against a number of techniques found in both pruning and interpretability literature. From the pruning literature we test both the Taylor expansion and the l_1-norm of the activation maps. The Taylor expansion is similar to Grad-AMap except for two crucial differences. Firstly, we backpropagate from the desired class's softmax score as we only want gradients pertinent to that class, with the Taylor expansion, we back propagate from the loss value. Secondly, to make the Taylor expansion valid, the absolute value of the product of the activation maps and gradients is taken. In the pruning literature this was found to be beneficial as it effectively ranks the best and worst filters similarly, which helps to keep the network stable when retraining [19]. We also test a variant of the Taylor expansion using the gradients backpropagated from the class's softmax which we refer to as 'Taylor (class)' in the results.

The results for filter removal versus model accuracy can be seen in Figure 3 and Table 1. These results confirm that Grad-AMap, our proposed method of measuring filter importance works well compared to other common methods. In particular, note that Grad-AMap is robust to the change in model, whilst Grad-CAM performs poorly with VGG16. An interesting property displayed in the ResNet18 results

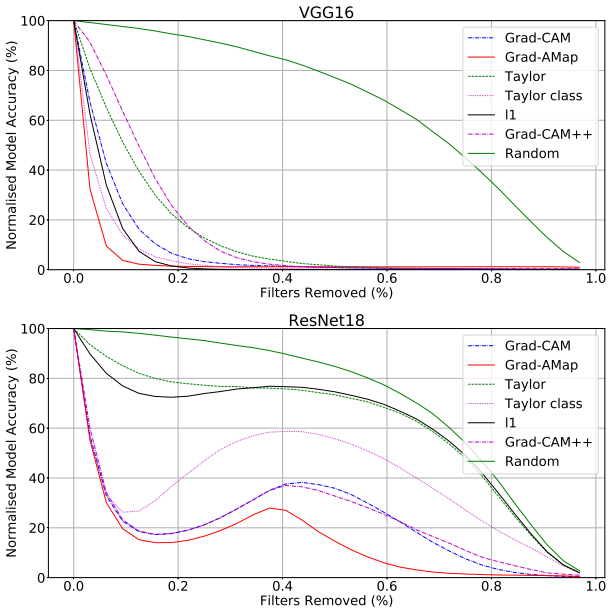


Figure 3. Showing the percentage of filters than can be removed vs the normalised model accuracy. Our proposed method is able to quickly decrease the accuracy be removing important filters.

	VGG16	ResNet18
Random	0.685	0.733
l_1-norm [17]	0.057	0.618
Grad-CAM++ [3]	0.147	0.230
Grad-CAM [26]	0.079	0.225
Taylor [5, 19]	0.132	0.629
Taylor (class)	0.052	0.392
Grad-AMap (Ours)	**0.042**	**0.146**

Table 1. Filter Removal Results. A lower AUC score suggests we can identify and remove the most important filters first, resulting in the network accuracy to drop.

is the small bump in accuracy as the filters are removed. This is likely happening because we are pruning a single convolution layers activation, which are then added back to the activations stored from the previous block. The effect of taking the absolute value in the Taylor method results in both the best and worst filters being removed early, which accounts for the earlier bump in accuracy. Comparing the Taylor and Taylor (class) results highlights the importance of back propagating gradients from the target class in the softmax rather than the networks loss.

4.2. Understanding the Reasons for Failure

Understanding why an input image fails to classify is an important problem. Does the network fail to predict a class correctly because there was a feature present that "distracted" the network? Or is it simply due to lack of relevant features? By passing unseen validation images into the net-

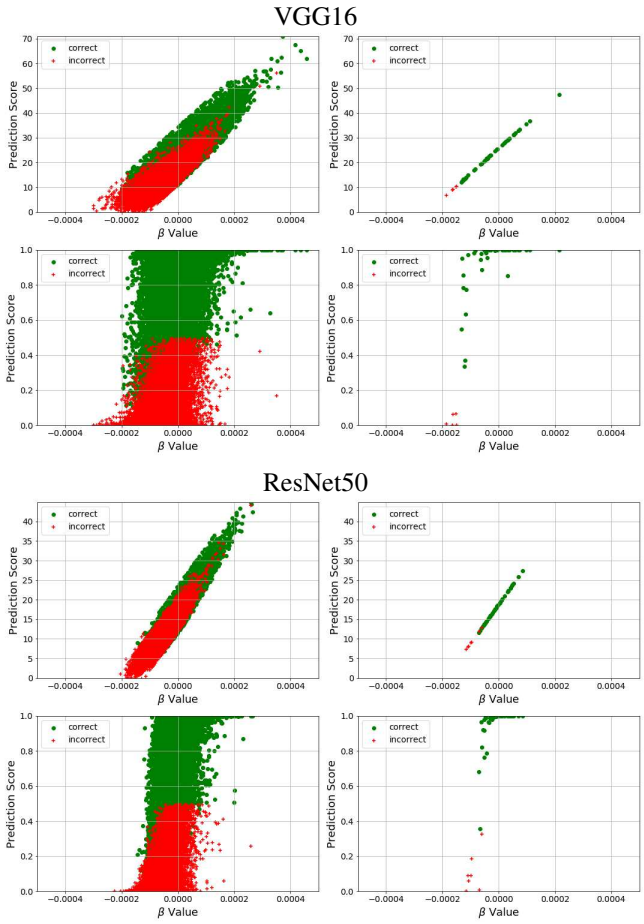


Figure 4. All β values for the validation images within ImageNet. For each model the top row are prediction scores prior to softmax being applied, while the bottom row are scores post softmax. Left columns are images from all classes whilst the right columns are images from a single class.

work and taking their filter score we are able to compute the β value from Equation 2. Using this value we perform Spearman rank correlation between all the β values within a class, and the prediction scores both pre- and post-softmax for each image. For VGG16, the mean Spearman score over all the classes between β values and the pre-softmax predictions is 0.998 whilst the score for post-softmax predictions is 0.782. For ResNet50, the mean Spearman score over all the classes between β values and the pre-softmax predictions is 0.999 whilst the score for post-softmax predictions is 0.801. These results can be seen in Figure 4. For the images classified incorrectly, we observe that for VGG16 86.60% of them had a negative deviation from the mean, whilst only 45.88% of correct classifications have a negative deviation. For ResNet50 we found these values to be 90.20% and 47.67% respectively. This suggests that the majority of images fail to be classified correctly because they lack the features the network is expecting, rather than that something in the image is surprising the network and overpowering other features.

In order to understand the reasons for failure we examine the unseen images that generate extreme β values. We find that they fall into two groups, those with a high β value (a positive deviation from the mean) fail due to being mistaken for a class that is similar to its real class. Those with a low β (a negative deviation from the mean) fail due to being noticeably different to the distribution of the training data.

Classifications with High β Values. When an image fails to be classified correctly but maintains a high β value it suggests that the filters activated by the target class have activated as expected or at a higher value. It also suggests that the unseen image, whilst posessing the relevant features to activate those filters, has also activated another set of filters that has tipped the network's decision. Indeed, we find that for all unseen images with high β values, the failure is due to the network predicting a class that is visually similar to the target class. Figure 5 illustrates this: shown are example images with high β values alongside their predicted and ground truth labels. From this it is easy to understand how the network may have made the wrong prediction due to the similarity of the predicted and ground truth classes.

Classifications with Low β Values. Unseen images that have a low β value tend to be those that deviate from the distribution of the class's training images. This deviation could occur due to the actual object itself being different from those in the training data, or the environment the object is in. We show examples of this in Figure 6. Here we display five misclassified images for each network, alongside a correctly classified image with a high β value. From these we can see that misclassified images with low β values tend to display a number of common characteristics. For example, scale seems to play a role in misclassification, with small objects being difficult to classify compared to those that fill the image. Another characteristic is that the misclassified image may be of a different medium, for example see the panda, tick and oxcart images, which whilst well-sized, fail to classify because they are atypical of the rest of the images in the class.

4.3. Visualising Surprise

Understanding which features in an image the network is using to form a class prediction is an important problem [22, 23, 33]. An interesting extension to this task is to identify regions of an image that are unexpectedly activating highly given the class of the image. This can be measured by our surprise score developed in Section 3.3. To visualise the features accurately we use weighted superpixels [9], a method of scoring image regions by backpropagating gradients and pooling them into superpixels. We use SLIC [1] to generate superpixels and 'vanilla' backpropa-

Figure 5. Examples of unseen images with high β values. The caption under each image is the predicted class with the corresponding ground truth in parentheses.

Figure 6. Misclassified images with low β values on the left of each column alongside an image from the same class that both classifies correctly and has a high β value, suggesting it displays features the network desires. Captions are the ground truth.

gated gradients [27] to weight them. To ensure we only visualise the regions important to a specific filter, we zero out all gradients produced by filters in the same layer as the surprised one. This has the effect of only backpropagating the gradients back to the input pixels that are relevant to the surprised filter. We use 50 superpixels and take the highest scored superpixel as the one most important to the specific filter. Passing an unseen image to the network and visualising the images from the training data that most activate the surprised filter gives us a visual insight into how the network is understanding the picture. An example of this is shown in Figure 7. Here we visualise a misclassified image marked with the boundary of the region most important to the surprised filter. Next to these we display the images from the training data that most activates the surprising filter and the region of the training image that activates the filter. By visualising these sets of images we are are able to gain an insight into why an image has been misclassified. For example, the top image is an image from the goldfish class misclassified as being from the roundworm class. It seems that the bottom fish highly activates a surprising filter. When we view the training images that activate the filter, we see that they are wheels. This high activation of the 'wheel' filter is presumably what causes the image to be misclassified as a roundworm. Indeed we show in Section 4.5 that this image could be made to classify correctly by removing the bottom fish. For other images, such as the velvet misclassified as swimming trunks, it is easy to see how a filter that activates highly for jigsaw may have fired, but not clear why it contributes to the swimming trunks class.

4.4. Visualising Expectation

Visualising the filters that did not activate highly is problematic due to a lack of features in the image that activate the specific filters. By identifying misclassified images that have an expected filter we are able to find images from the training set's target class that maximally activate the expected filter. Displaying these next to the misclassified image gives us an indication as to why the image may be misclassified. The results can be seen in Figure 8. We find that these images take two forms. The first is where the misclassified image simply doesn't seem to contain the correct

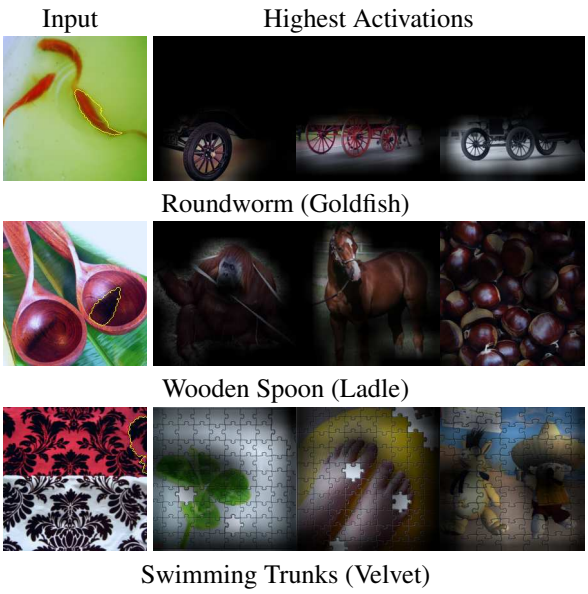


Figure 7. A misclassified image shown alongside the images from the training data that activates the surprising filter, masked to show the regions that activate the filter. The yellow boundary shows the region important to the filter. The caption is the predicted class with the ground truth in parentheses.

features. For example, with the gila monster, the expected filter is highly activated by white objects around the Gila Monster rather than the animal itself. This potentiality indicates a bias in the training data that the model has learnt to represent. A second example of this is the image of the partridge surrounded by barren branches. In contrast, the regions of the partridge training images that maximally activate the expected filter seem to be leaves or foliage. Again, this indicates a possible underlying bias within the partridge class. The second type of misclassified image seems to be where the target object is present, but small within the image and the expected filter simply wants to see more of the object. The image of the violin is a good example of this, although the violin and bow are present in the image, they are small. The filter that expected to be activated more is one that seems to activate highly on the bow region of the training data. The peacock image is similar to the violin, where the target object is present, but the expected filter seem to activate highly on a peacocks tail, as well as the crest on its head. Although these features are present in the misclassified image, they are not there in the scale required.

4.5. "Fixing" Incorrect Classifications

Using our method of generating surprise and expectation we can take specific misclassified images, and try to understand what caused their failure. This offers an understanding of how the network is representing the failed class. These corrections fall into two categories, suppressing features that are causing surprise to the network and diverting

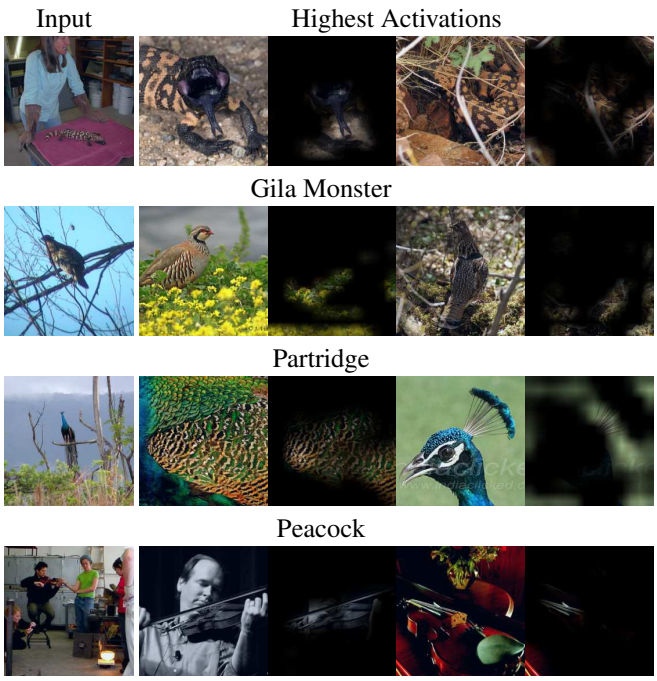


Figure 8. A misclassified input image is shown alongside the top two images and the regions that maximally activates the expected filter from the training data within the same class.

attention from the target class, and correcting expectation. The latter is a difficult task. It requires additional user input to interpret what the image is missing and whether it can be inserted back into the image. The following techniques are all supervised, they require knowledge of the ground truth class to "fix" the image.

Suppressing Surprise For all the images that are misclassified in the validation set, we identify the most surprised filter as in Section 3.3. Using the weighted superpixel technique discussed in Section 4.3, we obtain a number of superpixels with corresponding scores that tell us how important that region is to a specific filter. We are then able to suppress surprise by setting the superpixels that highly activate that filter to the mean of each RGB channel. As the images are normalised prior to inputting, this is 0. We perform this experiment over a range of superpixel sizes, as well as how many superpixels to blank out. We also perform the same technique using vanilla backpropagation (i.e. the same as our method without the surprise filtering) to ascertain that our method is superior at finding the regions causing misclassification. We show the results for this experiment in Figure 9. Here we see that the use of gradients backpropagated only through the surprised filter allows us to more precisely locate the detrimental regions of the image compared to backpropagating through all filters. This results in our method being able to 'fix' more misclassified images. Using VGG16 and sweeping through all the superpixel sizes and removal amounts, we are ul-

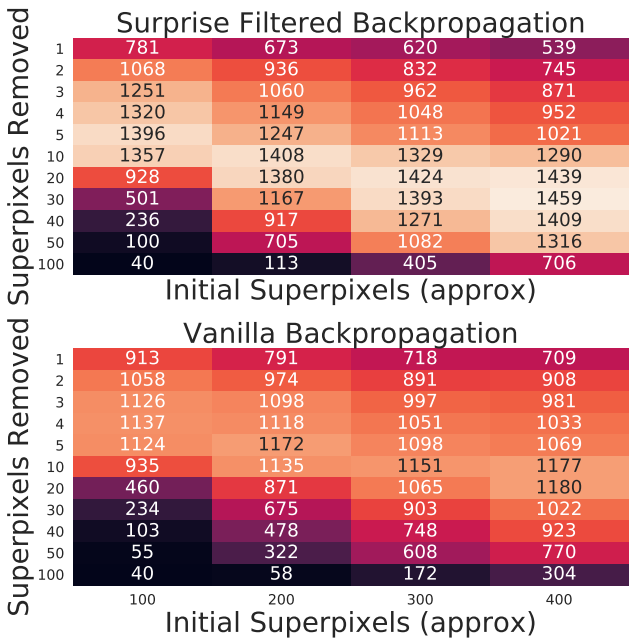


Figure 9. Results for suppressing the superpixels of both vanilla backpropagated gradients, and only the gradients backpropagated through the most surprising filter. These results are for VGG16.

timately able to 'fix' 5,116 misclassified images, using our technique versus 4,554 using vanilla backpropogation without surprise filtering. This is $\sim 36\%$ and $\sim 32\%$ of the misclassified images respectively. Examples of fixed images are in Figure 10, including the goldfish from Figure 7.

Figure 10. Surprise suppression examples from VGG16. All images initially misclassified, but 'fixed' using our method.

Correcting Expectation A more challenging application of our method is correcting images that failed due to an expected filter not activating. This means the image did not contain enough of a certain feature to activate the filter enough to ensure a correct classification. This becomes a difficult task as it is not a case of identifying and removing pixels as with suppressing surprise, rather we are required to identify and enhance or insert the required features. An example of this can be seen as the input in Figure 11. This shows an image from the ImageNet class Gila Monster that

Figure 11. Top Left: input. Top Right: 'fixed' image. Bottom: the images and masked regions that most activate the expected filter.

fails to be classified correctly. By visualising the training images that maximally activate the most expected filter, we gain an insight into which features the network is expecting to see. In Figure 11, we visualise the top 6 images for the expected filter. The first interesting aspect to notice is that none of the images seem to be activate the filter with the gila monster itself. Rather, the region surrounding the animal seems to activate the expected filter. The second is that all the images that activate the filter, appear to have a regions surrounding the animal that is lightly coloured. We hypothesise that altering the surface found in our misclassified image to a colour more consistent with the training data will boost this filters activation score and the overall score from the network. Using a photo editing tool, we select the red pixels from the surface and make them white. This image now classifies correctly suggesting that our proposed technique has correctly highlighted the pixels in the input image that were inconsistent with the training data. This suggests there is a bias towards gila monsters that are located on certain surfaces.

5. Conclusion

In this paper we have introduced the concept of surprise and expectation, a technique for understanding and visualising which filters are under- or over-performing for an unseen example. To do this we have proposed and quantitatively evaluated a filter ranking method, Grad-AMap. We have then qualitatively shown that this surprise and expectation allows us to gain insights into how and why images fail to classify correctly. We quantitatively show that our method is efficient with regards to locating features that are over activating and suppressing them to correct a classification.

6. Acknowledgements

This work is generously funded by BAE Systems and the EPSRC via iCASE award 1852482.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. 5

[2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3319–3327, July 2017. 1, 2

[3] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks. In *IEEE Winter Conference on Applications of Computer Vision*, pages 839–847, 2018. 1, 2, 4

[4] Dumitru Erhan, Y Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *Technical Report, Univerist de Montral*, 01 2009. 2

[5] Mikhail Figurnov, Aizhan Ibraimova, Dmitry P Vetrov, and Pushmeet Kohli. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *Advances in Neural Information Processing Systems*, pages 947–955, 2016. 2, 4

[6] Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2018. 2

[7] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 3449–3457, Oct 2017. 2

[8] Karl Friston. The free-energy principle: A rough guide to the brain? *Trends in Cognitive Sciences*, 13(7):293–301, 2009. 2

[9] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Gradient weighted superpixels for interpretability in CNNs. In *BMVC 2019: Workshop on Interpretable and Explainable Machine Vision*, September 2019. 5

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. 3

[11] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 9734–9745. Curran Associates, Inc., 2019. 4

[12] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016. 2

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. 3

[14] Rajiv Khanna, Been Kim, Joydeep Ghosh, and Oluwasanmi Koyejo. Interpreting black box predictions using fisher kernels. *CoRR*, abs/1810.10118, 2018. 2

[15] Been Kim, Rajiv Khanna, and Sanmi Koyejo. Examples are not enough, learn to criticize! Criticism for interpretability. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 2

[16] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677. PMLR, 10–15 Jul 2018. 1, 2

[17] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 2, 4

[18] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738, December 2015. 2

[19] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *CoRR*, abs/1611.06440, 2016. 2, 4

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019. 3

[21] Genevieve Patterson, Chen Xu, Hang Su, and James Hays. The sun attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision*, 108(1-2):59–81, 2014. 2

[22] Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: randomized input sampling for explanation of black-box models. In *British Machine Vision Conference 2018, BMVC*, 2018. 1, 4, 5

[23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016. 1, 2, 5

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1, 3

[25] Ramprasaath R. Selvaraju, Prithvijit Chattopadhyay, Mohamed Elhoseiny, Tilak Sharma, Dhruv Batra, Devi Parikh,

and Stefan Lee. Choose your neuron: Incorporating domain knowledge through neuron-importance. In *The European Conference on Computer Vision (ECCV)*, pages 540–556, September 2018. 2

[26] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, Oct 2017. 1, 2, 4

[27] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *2nd International Conference on Learning Representations, ICLR 2014, Workshop Track Proceedings*, 2014. 1, 2, 6

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 1, 3

[29] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. In *3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings*, 2015. 1, 2

[30] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014. 1, 2

[31] Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu. Interpreting cnn knowledge via an explanatory graph. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[32] Quanshi Zhang, Wenguan Wang, and Song-Chun Zhu. Examining CNN representations with respect to dataset bias. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[33] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, June 2016. 1, 2, 5