PATCHNET: INTERPRETABLE NEURAL NETWORKS
FOR IMAGE CLASSIFICATION

ADITYANARAYANAN RADHAKRISHNAN, CHARLES DURHAM, ALI SOYLEMEZOGLU,
AND CAROLINE UHLER

Abstract. The ability to visually understand and interpret learned features from complex predictive models is crucial for their acceptance in sensitive areas such as health care. To move closer to this goal of truly interpretable complex models, we present PatchNet, a network that restricts global context for image classification tasks in order to easily provide visual representations of learned texture features on a predetermined local scale. We demonstrate how PatchNet provides visual heatmap representations of the learned features, and we mathematically analyze the behavior of the network during convergence. We also present a version of PatchNet that is particularly well suited for lowering false positive rates in image classification tasks. We apply PatchNet to the classification of textures from the Describable Textures Dataset and to the ISBI-ISIC 2016 melanoma classification challenge. Via these examples, we show that in addition to providing interpretable features, some of the filters used in PatchNet also directly perform image segmentation.

1. Introduction

There have been significant improvements to feed-forward convolutional networks culminating in the recent success of ResNet [11] in the ImageNet [17] challenge. These (now standard) networks are powerful due to their ability to exploit global context to make a more informed decision. By incorporating non-linear layers, these models use complex combinations of features to derive an accurate label estimate given all the pixels in the input image. The complex interactions between features make it difficult to inspect the learned features visually. Without an easy means of interpreting learned features, the applicability of these models to prediction sensitive areas such as health care is limited. For example, Caruana et al. [1] analyzed models trained to predict the risk of death by pneumonia for patients. They found that these models had learned that having asthma was indicative of a low risk of dying by pneumonia. However, the reason for this learned feature was that patients with asthma were immediately admitted to the ICU when they had pneumonia, and so they would be treated immediately. Thus, asthmatic patients appeared in the training data as examples of patients with low-risk of death by pneumonia. Although often highly accurate, due to the uninterpretability of the learned features, neural networks are often deemed too risky for applications in health care [5].

The goal of our work is to provide a convolutional network for binary classification problems on images that can provide interpretable visual representations of the learned features as heatmaps, simply by inspecting the outputs of the feed-forward layers. To accomplish this, our network PatchNet first provides classification decisions on small patches of an image to determine whether a given patch contains features of either class, and then averages the decisions made on all the patches across an image to make a global classification decision. Hence, PatchNet allows for a trade-off between generalization error and feature interpretability: by restricting the size of patches, generalization error increases since the model is limited in global context for classification, but feature interpretability increases as we can visualize the learned features for each patch of an image.

PatchNet is motivated by mean-field approximation techniques from variational inference as well as ensemble methods. As is done in mean-field approximations, instead of learning conditional probability distributions for predicting the label for an image given the entire image, we instead
learn a simpler conditional probability distribution for predicting the label given a small patch of the original image. Now unlike a true mean-field approximation, we do not multiply the predictions for each patch to get a global prediction, but rather treat each of these patch predictions as an ensemble of smaller classifiers and average their classifications to generate a global classification decision.

The remainder of this paper is organized as follows. Section 2 describes previous methods for interpreting or rationalizing features learned by neural networks. Section 3 describes the PatchNet architecture, its convergence behavior, how to extract heatmaps for interpreting the learned features, and Precision-PatchNet, an extension of PatchNet for classification tasks where obtaining a high precision is important, as is common when working with medical images. We present the results of PatchNet for classifying textures from the Describable Textures Dataset (DTD) [2] and the results of Precision-PatchNet on the ISIC-ISBI Melanoma Classification Challenge [9] in Section 4. We end with concluding remarks and an outline of future work in Section 5.

2. Related Work

A prominent approach used to understand the inner workings of a complex neural network is the “deconv” technique. Variations of this technique such as DeconvNet [23] or Guided Backpropagation [20] pick a neuron in a convolutional neural network (CNN), do a forward pass on a sample image, and set the chosen neuron’s gradient to 1 and all other gradients in the same layer to 0. Doing backpropagation on this signal and blocking the negative gradients, results in an alteration to the image that represents the neuron’s positive contributions to the network as a whole. A similar method was described in [18], where the authors propose an optimization technique that feeds an image forward through a CNN and then backpropagates from the last layer with a specific class target.

Although these techniques allow for visual introspection of the learned features and classes, there are a few drawbacks: (1) All techniques require applying alterations to the true gradient by blocking signals or using regularization parameters. (2) The backpropagation of the gradient through multiple layers provides noisy estimates of the true learned features. (3) The learned features are based on global context, making it inherently difficult to understand how different features work together to create a classification decision and substantially reducing feature interpretability.

A different approach is taken in [15], where a Natural Language Processing model is developed that provides rationales for why a text review is classified as positive or negative. Their model consists of two networks, one to provide tags for whether a word in the review is indicative of “sentiment”, and another to classify based on the tagged words. Unfortunately, this approach cannot directly be applied to image classification, since there is no clear analogy between tagging words in a sentence and tagging pixels in an image, as the space of words is much larger than the space of pixel values. Furthermore, naively dropping pixels by techniques such as zeroing them out is not equivalent to dropping irrelevant words in a sentence, since zero value pixels still provide context to the image.

3. PatchNet

In this section, we provide the mathematical motivation for our model architecture, present the architecture, and develop insights into its convergence behavior for special classes of data. We then present a slight modification that incentivizes our model to reduce the false positive rate for classification. We conclude the section with an explanation on how to easily extract global and filter heatmaps from our model to view the learned features.

3.1. Motivation and Notation. Suppose we are given a dataset \( D \) consisting of a list of images \( I^{(1)}, I^{(2)}, \ldots, I^{(k)} \) with each \( I^{(j)} \in \{0, 1, 2 \ldots N - 1\}^{m \times n \times c} \) along with a corresponding list of labels \( y_{I^{(1)}}, y_{I^{(2)}}, \ldots, y_{I^{(k)}} \) with each \( y_{I^{(j)}} \in \{0, 1\} \). Feed-forward CNNs such as VGG [19] or ResNet [11] directly estimate the distributions \( P_{y|I}(y|I) \), which are conditional distributions given all pixel values. Since we will concentrate solely on binary classification, it suffices to estimate \( P_{y|I}(1|I) \).
Then visual interpretability of the learned features suffers due to the scale of feature detection. By using binary cross entropy loss the global classification estimate Finally, a single value determined as follows. A CNN consisting of \( P \) each \( G \) local classification decision given a patch; see Figure 1.

**Model Architecture.** Our model consists of two components: (1) a global network \( G \) that outputs a global classification decision given an entire image; (2) a local network \( S \) that outputs a local classification decision given a patch; see Figure 1.

We now describe the feed-forward pass of our architecture. When an input image \( I \) is fed into \( G \), \( G \) chunks the image into a list of \( l \) patches of size \( m' \times n' \times c \), namely \( [P^{(1)}, P^{(2)}, \ldots, P^{(l)}] \) with each \( P^{(j)} \in \{0, 1, 2, \ldots, N - 1\}^{m' \times n' \times c} \) with \( 3 \leq m' \leq m \), \( 3 \leq n' \leq n \), then our model estimates \( P_{1|I}(1|I) \) as \( P_{1|I}(1|I) = \frac{1}{l} \sum_{j=1}^{l} Q_{1|P^{(j)}}(1|P^{(j)}) \), where \( Q \) is a single learned distribution applied to each patch.

There is an inherent tradeoff between patch size \( (m', n') \), generalization error, and visual interpretability of the learned features. For instance, with \( m' = m \) and \( n' = n \), our model can mimic any CNN by simply letting the estimate for \( Q \) be the estimate determined by the CNN. In this case, the generalization error achieved by the model is the same as that of the mimicked network, but the visual interpretability of the learned features suffers due to the scale of feature detection. By using small \( m', n' \), the distribution \( Q \) is estimated across a smaller input space allowing detection of local features. As we will show in Section 3.3 smaller patch sizes provide visual interpretability of the features used for classification at the expense of a potentially larger generalization error.

3.2. Model Architecture. Our model consists of two components: (1) a global network \( G \) that outputs a global classification decision given an entire image; (2) a local network \( S \) that outputs a local classification decision given a patch; see Figure 1.

We now describe the feed-forward pass of our architecture. When an input image \( I \) is fed into \( G \), \( G \) chunks the image into a list of \( l \) patches of size \( m' \times n' \times c \), namely \( [P^{(1)}, P^{(2)}, \ldots, P^{(l)}] \) with each \( P^{(j)} \in \{0, 1, 2, \ldots, 255\}^{m' \times n' \times c} \). Next, these patches are aggregated and sent to \( S \) as a mini-batch. Then \( S([P^{(1)}, P^{(2)}, \ldots, P^{(l)}]) = [Q_{1|P^{(1)}}(1|P^{(1)}), Q_{1|P^{(2)}}(1|P^{(2)}), \ldots, Q_{1|P^{(l)}}(1|P^{(l)})] \), where \( Q_{1|P^{(j)}} \) is determined as follows. A CNN consisting of 7 layers of convolutions with 64 filters with kernel size 3 and 1 pixel of padding followed by ReLU activations in each layer is applied to \( P^{(j)} \) to get 64 \( m' \times n' \) filter output images. Then 64 linear models are applied as a dot product to each of these filter output images to reduce the \( m' \times n' \) outputs to size 1 outputs, and a Tanh activation is applied to each output. Lastly, a linear layer is applied as a dot product to these 64 outputs resulting in a single value \( \tilde{Q}_{1|P^{(j)}} \) for each patch, which is converted to \( Q_{1|P^{(j)}} \) by a sigmoid transformation. Finally, \( G \) averages \( [Q_{1|P^{(1)}}(1|P^{(1)}), Q_{1|P^{(2)}}(1|P^{(2)}), \ldots, Q_{1|P^{(l)}}(1|P^{(l)})] \), the output from \( S \), to obtain the global classification estimate \( P_{1|I}(1|I) \). Given a label \( y^* \) and a global prediction \( P_{1|I}(1|I) \) we use the binary cross entropy loss to determine the loss of the model, namely

\[
L(I, y^*) = -y^* \log(P_{1|I}(1|I)) - (1 - y^*) \log(P_{0|I}(0|I)).
\]
To minimize this loss, we first analyze the derivative with respect to $p_1$, which is always negative (since $p_0, p_1, T > 0$) and hence closest to 0 when $p_1 = 1$. Taking derivatives with respect to $p_0$ yields

$$\frac{\partial L}{\partial p_0} = -T \frac{3p_0 + p_1 + 3p_0 - 3}{p_0(3p_0 + p_1)}.$$
which is 0 when \( p_0 = \frac{1}{3} \). Substituting these values back into the global predictions, we obtain that \( \mathbb{P}_{1|I_1}(1|I_1) = \frac{1}{2} \) for class 1 images \( I_1 \) and \( \mathbb{P}_{1|I_0}(1|I_0) = \frac{1}{3} \) for class 0 images \( I_0 \). Hence, with a rounding threshold for class 1 of \( .5 + \epsilon \), at convergence, our model would predict that all images belong to class 0. However, even though the resulting accuracy would only be 50\%, the feature heatmap (constructed as described in Section 3.3) would still indicate that there is a feature representative of class 1 in the upper left corner of the image, since \( \mathbb{Q}_{1|P(j)}(1|P(j)) = 1 \) for all patches \( P(j) \) containing only 1-valued pixels.

This limitation explains mathematically why there is a trade-off between generalization error and interpretability: if there are a few patches with features indicative of class 1 in class 1 images, then these patches must all output values \( \mathbb{Q}_P \) close to 1 in order for \( \mathbb{P}_{1|I_1}(1|I_1) \) to be closer to 1, and so the heatmap visualization will identify the regions of the image that are indicative of class 1. However, these patches can be outvoted by patches with features indicative of class 0 leading to a global misclassification of the image. See Appendix A for a generalized analysis of the convergence behavior and examples of the architecture displaying this behavior at convergence.

3.4. Extracting Visualizations. We now present how we can introspect layers of our model to easily provide visualizations of the features learned by the model. We refer to the gray-scale visualizations we generate as "heatmap" visualizations, since brighter pixels in the visualizations indicate that the model found a feature relevant to class 1, while darker pixels indicate a feature relevant to class 0. We provide two sets of heatmap visualizations and their corresponding pixel intensity histograms as follows.

The \textit{global heatmap visualization} for an image is constructed by first computing \( \mathbb{Q}_{1|P(i,j)}(1|P(i,j)) \) for all patches \( P(i,j) \), where \( P(i,j) \) is the patch centered at location \((i,j)\) in the first two dimensions of the original image (zero-padding is used for border locations), and then viewing an \( m \times n \) image where each pixel at location \((i',j')\) of the image is the value of \( \mathbb{Q}_{1|P(i',j')} (1|P(i',j')) \).

The \textit{filter heatmap visualizations} are similar to global heatmap visualizations, except that a heatmap image is constructed for each filter in the last convolutional layer by using the output of the Tanh layer across all patches. We show the construction for filter 1 and apply the same construction for the other 63 filters. For each patch \( P(i,j) \) centered at location \((i,j)\) in the first two dimensions of the original image (again with zero-padding for border locations), we compute the output of the Tanh layer and label the corresponding pixel \( F_1 (i,j) \) as \( F_1 (i,j) \in [-1,1] \). Now we simply view the \( m \times n \) image where each pixel at location \((i',j')\) of the image is the min-max rescaled value of \( F_1 (i',j') \) to get the filter heatmap for filter 1.

In this way, we generate and view 65 heatmap visualizations for each image fed to the model: 1 for the global heatmap and 64 for the filter heatmaps. As an aside, another approach to constructing filter or global heatmap visualizations is to average the predicted heatmap pixel values across all patches containing the given pixel. We found that this approach provided empirically inferior results in heatmap smoothness than the above described visualizations.

3.5. Precision-PatchNet. In various applications, it is crucial to obtain high precision results, i.e., with few false positives. For example, a common problem with mammography for breast cancer detection is its high false positive rate: recent studies revealed that the chance of having a false positive result after 10 yearly mammograms is about 50-60\% \[8, 12, 25\]. Such results can cause unnecessary worries and surgeries. We thus present a variation of PatchNet that is optimized for image classification tasks where having a low false positive rate matters more than overall accuracy.

For such \textit{precision-based tasks}, the CNN \( S \) should additionally minimize the number of patches \( P \) outputting high values of \( \mathbb{Q}_P \) for class 0 images, i.e., \( S \) should reduce the false positive rate for the patch-based classification in addition to being accurate. This can be achieved by a minor adjustment to the network, namely by introducing separate loss metrics for images in class 0 and class 1. For class 1 images, we keep the originally described binary cross entropy loss metric and hence the
resulting loss function is as described in equation (1). However, for class 0 images, we introduce a binary cross entropy loss on $Q_{1|P}(1|P)$ for each patch $P$ and average it over all patches, i.e.,

$$
\tilde{L}(I_0, 0) = \frac{1}{l} \sum_{i=1}^{l} \log(1 - Q_{1|P^{(i)}}(1|P^{(i)}))
$$

for an image $I_0$ with $l$ patches. This loss function incentivizes the model to assign low probabilities $Q_{1|P}(1|P)$ to each patch $P$ in class 0, enforcing a low false positive rate among patches in class 0.

4. Experimental Results

We now show the performance of PatchNet in providing interpretable features and analyze the trade-off between visual feature interpretability and generalization error for different patch sizes in applications to the classification of textures and melanoma. For each experiment we provide the patch size, and training, validation, and test sets used to train our model. All of our models use the Adam optimizer with a learning rate of $5 \cdot 10^{-5}$, with batch sizes of either 8 or 16. We used Kaiming normal initialization [10] for all convolution layers and used min-max scaling to normalize the input images to have pixel values in the range $[0, 1]$. For all experiments, we augmented the training set using random horizontal and vertical reflections of the input images. Finally, to select the best model, we used a patience strategy [6], where we declared convergence when the model had not seen any improvement in validation loss for 2000 epochs. The software and hardware used for these applications is described in Appendix D and E.

4.1. Describable Textures Dataset (DTD). DTD [2] is a collection of real-world texture images annotated with “human-centric” attributes. The dataset consists of 47 classes of textures with 120 images per class. The image sizes range from $300 \times 300 \times 3$ to $640 \times 640 \times 3$. Although other texture datasets exist, such as CUReT [7], UMD [21] or UIUC [14], this dataset is the most extensive in terms of number of images per class.

We trained our model on two classes of images: (1) class 0 images are drawn from the "cracked" textures; (2) class 1 images are drawn from the "perforated" textures. We chose these classes as they

| Model          | Train  | Validation | Test  |
|----------------|--------|------------|-------|
| Neural Model   | 88.8%  | 76.3%      | 67.5% |
| Linear Model   | 99.3%  | 80.7%      | 73.3% |

| Reference/Prediction | Cracked | Perforated |
|----------------------|---------|------------|
| Cracked              | 34      | 6          |
| Perforated           | 20      | 20         |

Figure 4. The accuracy of our models and the confusion matrix for the DTD test set.
have textures that are present on a local scale and these local textures generally repeat throughout the image. Samples from each of these classes are shown in Figure 3.

Since the images are of different sizes, we sampled random $128 \times 128 \times 3$ crops from the training set for each batch and used centered $128 \times 128 \times 3$ crops from the validation and test sets to validate and test. We used patches of size $15 \times 15 \times 3$ and a stride of 3 in each of the first two dimensions. As provided by DTD, we split the data into 40 images per class for each of the training, validation, and test sets. The classification performance is summarized in Figure 4 along with a comparison to the performance of SVM trained with the widely used Parameter Free Threshold Adjacency Statistics (PFTAS) features [4]. As shown in Appendix B, adding any other standard texture features caused the model to greatly overfit. PatchNet slightly under-performs the SVM model on test accuracy, even though both models over-fit the data.

The confusion matrix in Figure 4 shows that misclassification predominantly occurred for the perforated images. The global heatmaps for these images in Figure 5 show that the model correctly classified every perforation, but because in many images there is a larger amount of background than perforation, the model misclassified some perforated images. These results are thus in accordance with the mathematical derivation in Section 3.3.

As is also apparent from Figure 5, the model learned to identify dark contiguous regions in an image as perforations, while dark lines in an image were identified as cracks. Since the global heatmaps can be explained without any technical background, we would argue that the features learned by PatchNet are easier to interpret than the standardly used PFTAS texture features.

4.2. ISBI-ISIC 2016 Melanoma Challenge. Melanoma is a skin cancer that develops from pigmented lesions on the skin. The International Skin Imaging Collaboration (ISIC) Archive provided a collection of 1250 melanoma images (900 for training, 350 for testing) for the 2016 Melanoma Classification Challenge [9]. We used the 2016 melanoma detection challenge instead of the 2017 challenge, since testing data is currently only available for the 2016 challenge.

The data is split into benign (class 0) and malignant (class 1) skin lesions. Due to the high class imbalance with around 5 times more benign than malignant examples, we up-sampled the malignant samples from each class.

![Figure 5](image.png)  
**Figure 5.** The global heatmaps for images sampled from the DTD test set.

![Figure 6](image.png)  
**Figure 6.** From left to right: train, validation, and test accuracies for the PatchNet model trained using $15 \times 15 \times 3$ patches and $31 \times 31 \times 3$ patches; confusion matrix for the PatchNet model trained using $15 \times 15 \times 3$ patches; confusion matrix for the PatchNet model trained using $31 \times 31 \times 3$ patches.
class during training. The size of the original images is around $700 \times 1000 \times 3$. Due to memory limitations, we rescaled these images to $128 \times 170 \times 3$ prior to training, validating, and testing.

We now present the results of training two precision-based models. The first uses patches of size $15 \times 15 \times 3$ with a stride of 5 in each of the first two dimensions. The second uses patches of size $31 \times 31 \times 3$ with a stride of 10 in each of the first two dimensions. The resulting accuracies and confusion matrices are shown in Figure 6. Although both models over-fit, they both succeeded in minimizing the number of false positives during training and validation. In addition, as predicted by the mathematical derivation in Section 3.3, the larger patch size achieved a higher accuracy overall.

In Figure 7 we visualize the global heatmaps for sample test data using both models. As predicted in Section 3.3, there is a trade-off between generalization and visual interpretability for different patch sizes: the heatmaps for the model with larger patch size are more noisy as a consequence of training on patches with little overlap, but this model generalizes better. The heatmaps show that the model has learned to identify dark regions or spotted regions (known as globules) and irregular surroundings of regions (known as streaks). Interestingly, these are some of the features used by domain experts to classify melanoma. The 2016 ISBI-ISIC Melanoma Challenge also gives examples of images where these features were annotated. A comparison to the features found by PatchNet is shown in Figure 8.

As a side remark, the 2016 ISBI-ISIC Melanoma Challenge also illustrates the risks of using CNNs for medical applications. Since various malignant training examples had a lens around the melanoma, the model learned this as a feature. The second malignant example in Figure 7 shows that the model learned to classify the lens as part of a malignant lesion, when clearly the background is not part of the region of interest.
The ISIC also provided ground truth segmentations for the skin lesions. Interestingly, while most models for segmentation need to be trained on labeled segmentation data, by viewing the filter heatmaps from our models trained solely for classification, we were able to identify filters that provide lesion segmentations directly; see Figure 8.

Inspired by our segmentation results for melanoma, we trained PatchNet to also classify between images of human cell nuclei, an important preprocessing task in the classification of cancerous cells. In Figure 9, we present sample segmentation results obtained by PatchNet; for a careful analysis see Appendix C. Remarkably, although PatchNet was not trained on any labeled data for nucleus segmentation, it achieves precise segmentations regardless of the proximity of multiple nuclei and halo (i.e., intensity drop-off) effects.

5. Conclusion and Future Work

By forcing independent local classification decisions on patches, PatchNet is able to re-construct interesting global and filter heatmaps that provide easy introspection into learned features. In addition to interpretability, an interesting side effect of PatchNet is that select filters can directly be used for image segmentation. In future work, it will be interesting to further investigate the performance of our model for image segmentation. Since the described implementation is limited by existing hardware, an interesting line of future work is to investigate patch-level data parallelism techniques to increase our model’s scalability.

Appendix A. Generalized Convergence Behavior of PatchNet

Generalizing from the results in Section 3.3, we analyze how the value of $Q_P$ for each patch $P$ is related to the fraction of features indicative of class 1 present in class 1 images, the fraction of features indicative of class 0 present in class 0 images, and the fraction of features shared among both classes. Intuitively, the model will assign low $Q_P$ outputs for each patch $P$ with a feature indicative solely of class 0, high $Q_P$ outputs for each patch $P$ with a feature indicative solely of class 1, and roughly $Q_P = .5$ for patches with features shared between class 0 and 1. In this section, we perform a case analysis on simplified images that contain only features that are indicative of class 1 or features that are shared between class 0 and class 1. We mathematically determine the exact values of $Q_P$ for each patch after convergence.

Due to the nature of our loss function, we can mathematically analyze the behavior of the model to understand the relationship between the visual heatmap and presence of class 1 features in the dataset under some simplifying assumptions. Namely, suppose that the images in the dataset contain only features, $f_c$, common to both classes of the image or features, $f_1$, indicative of class 1. Now suppose that on average there are $a$ patches in class 1 that are indicative of $f_c$ and $b$ patches in class 1 that are indicative of $f_1$, and that all $a + b$ patches of class 0 are indicative of $f_c$ with $a > b$. Let us further assume that each patch contains only one of $f_c$ or $f_1$, and so $Q_{1|P^{(j)}}(1|P^{(j)}) = p_0$ for all $j$.

---

1We would like to thank G. V. Shivashankar (National University of Singapore) for providing the images.
such that patch \( P^{(j)} \) contains \( f_c \) and \( Q_{1|P^{(j)}}(1|P^{(j)}) = p_1 \) for all \( j \) such that \( P^{(j)} \) contains \( f_1 \). Then, as in Section 3.3, at convergence the model minimizes the loss

\[
L = -\log \left( 1 - \frac{(a + b)p_0}{(a + b)} \right) - \log \left( \frac{ap_0 + bp_1}{a + b} \right)
\]

\[
= -\log(1 - p_0) - \log \left( \frac{ap_0 + bp_1}{a + b} \right)
\]

In order to minimize the loss, we set the derivatives with respect to \( p_0 \) and \( p_1 \) equal to 0. The derivative with respect to \( p_1 \) is:

\[
\frac{\partial L}{\partial p_1} = -\frac{a + b}{ap_0 + bp_1} \frac{b}{a + b}
\]

\[
= -\frac{b}{ap_0 + bp_1}
\]

This derivative is always less than 0 as \( a, b, p_1, p_0 \geq 0 \). Hence, the model will simply try to maximize the value of \( p_1 \) to bring the derivative closer towards 0. Thus, at convergence, \( p_1 = 1 \).

Now examining the derivative with respect to \( p_0 \):

\[
\frac{\partial L}{\partial p_0} = 0
\]

\[
\Rightarrow -\frac{1}{p_0 - 1} - \frac{a + b}{ap_0 + bp_1} \frac{a}{a + b} = 0
\]

\[
\Rightarrow ap_0 + bp_1 + ap_0 - a = 0
\]

\[
\Rightarrow ap_0 + b + ap_0 - a = 0
\]

\[
\Rightarrow p_0 = \frac{a - b}{2a}
\]

Hence the largest possible value for \( p_0 \) is \( \frac{1}{2} \), which occurs when \( b = 0 \). Note that we performed this analysis for \( a > b \). If \( a \leq b \), as \( p_0 \) is constrained to be in the range \([0, 1]\), the value of \( p_0 \) will simply be 0 at convergence.

As a concrete example of this convergence behavior in practice, we can examine the behavior of the model on a sample binary data set where class 0 consists of all black \( 128 \times 128 \) images (each pixel has value 0), and class 1 consists of \( 128 \times 128 \) images with a white square of size \( 64 \times 64 \) in the upper left hand corner (each pixel has value 1) (a larger version of the sample image in Section 3.3).

We train our model using \( 17 \times 17 \) patches and a stride of 17 in either direction. In this case, there are some patches that contain both \( f_c \) and \( f_1 \). Yet as there are only a few such patches, the actual values of \( p_0 \) and \( p_1 \) should only be slightly noisy. We use zero padding and a stride size of 1 to visualize the outputs of \( Q \) on each of the \( 128 \times 128 = 16384 \) patches centered at each pixel value in the original image. Since the \( f_1 \) features are contained in a \( 64 \times 64 \) square in the upper left corner of the image, we claim that there are approximately \( b = 64 \times 64 = 4096 \) patches that contain \( f_1 \).

Now, by our calculations, we expect that \( p_0 = \frac{16384 - 2 \times 4096}{2 \times (16384 - 4096)} = \frac{1}{3} \) and that \( p_1 = 1 \). That is, we expect our model to output a value of \( \frac{1}{3} \) for patches containing only \( f_c \) and 1 for patches containing only \( f_1 \). Indeed, after our model had converged to a local minimum, the model output a value of 0.333 for patches consisting of only \( f_c \) and output a value of 0.982 for a patch consisting of only \( f_1 \).

**Appendix B. Linear Models for DTD**

In this section, we describe all linear models used on the DTD data for cracked and perforated images [2]. The training, validation, and test accuracies for the SVM model and logistic regression
| Feature                          | Train | Validation | Test  |
|---------------------------------|-------|------------|-------|
| PFTAS + Haralick                | 99.9% | 83.1%      | 45.7% |
| PFTAS + Zernike + Haralick      | 100.0%| 82.6%      | 44.7% |
| PFTAS                           | 99.3% | 80.7%      | 73.3% |
| PFTAS + Zernike Moments         | 99.8% | 80.0%      | 46.8% |
| Zernike + Haralick              | 92.4% | 78.8%      | 46.3% |
| Haralick                        | 91.9% | 78.4%      | 49.5% |
| Zernike                         | 65.7% | 64.3%      | 45.6% |

Table 1. Training, Validation, and Test results for SVM across DTD [2] data splits.

| Feature                          | Train | Validation | Test  |
|---------------------------------|-------|------------|-------|
| PFTAS + Zernike + Haralick      | 98.9% | 82.9%      | 46.2% |
| PFTAS + Haralick                | 98.9% | 82.8%      | 44.5% |
| PFTAS + Zernike Moments         | 98.3% | 81.6%      | 48.5% |
| PFTAS                           | 98.0% | 81.4%      | 65.0% |
| Zernike + Haralick              | 90.3% | 79.2%      | 45.9% |
| Haralick                        | 89.1% | 77.6%      | 45.5% |
| Zernike                         | 65.7% | 64.6%      | 45.6% |

Table 2. Training, Validation, and Test results for Logistic Regression across DTD [2] data splits.

| Model               | Train | Validation |
|---------------------|-------|------------|
| Neural Model        | 94.9% | 90.9%      |

Table 3. Training and validation results for PatchNet across the nucleus dataset.

| Reference\Prediction | BJ       | NIH3T3    |
|----------------------|----------|-----------|
| BJ                   | 428/68   | 47/14     |
| NIH3T3               | 1/1      | 474/81    |

Table 4. Confusion matrix for training\validation data from the nucleus dataset.

model are shown in Table 1 and Table 2 respectively. We ran the linear models using popular texture features extracted using the Mahotas library [3]. From these tables it is apparent that all linear models overfit with these texture features. The best test accuracy was achieved by using the SVM model with only PFTAS features (as reported in Section 4.1). However, if we were to select our models based on validation accuracy, as is typically done in practice, the best model is SVM using both PFTAS and the Haralick features, which achieves a poor test accuracy of 45.7%, significantly worse than the test accuracy achieved by PatchNet.
Appendix C. Cell Nucleus Data

In this section, we discuss the results of running our model for the problem of extracting cell nuclei from images obtained by our collaborator G. V. Shivashankar (National University of Singapore). This task is an important preprocessing step for identifying cancerous cells in tissue images. The dataset of cell nuclei consists of connective tissue cells from humans (BJ cells) and mice (NIH3T3 cells). Sample images of NIH3T3 nuclei and BJ nuclei are shown in Figure 10.

C.1. Nucleus Crops. We first run PatchNet on pre-segmented nuclei crops of BJ (class 0) and NIH3T3 (class 1). The images are of size $128 \times 128 \times 1$. We used 557 images of each class for training and 82 images of each class for validation. We trained using $17 \times 17 \times 1$ patches with a stride of 5. The resulting accuracies and confusion matrix given in Table 3 and 4 show that the model was able to achieve a high validation accuracy for this dataset.

We now present the global heatmaps in Figure 11. From the heatmaps, we see that the model very accurately learns the regions occupied by NIH3T3 and BJ nuclei. We also see that the model assigns a grey value to the background indicating that the background is common to both images. All 64 filter heatmaps are presented in Figure 12 and 13.

As indicated by the large proportion of grey filter pairs, we see that not all filters are required to achieve high accuracy on this dataset, indicating that our model has more capacity than required for
Figure 12. Four columns of 8 pairs of filter heatmaps with the class 0 filter heatmap appearing in the left of the column pairs and the class 1 filter heatmap appearing in the right of the column pairs.

Figure 13. The remaining 4 columns of 8 pairs of filter heatmaps with the class 0 filter heatmap appearing in the left of the column pairs and the class 1 filter heatmap appearing in the right of the column pairs.
Figure 14. Samples of class 0 (BJ) images and class 1 (NIH3T3) images are depicted on the left and right respectively.

Figure 15. Heatmaps for filter 55 of PatchNet for the nuclei in Figure 14.

Figure 16. Heat maps for filter 57 of PatchNet for the nuclei in Figure 14.
this problem. In addition, we see that some filters that are contributing to classification are able to pick up the bright dots on the NIH3T3 nuclei as features. These are known features indicative of mouse cells. We also note the presence of filters that seem similar to edge detectors on the internal regions of the nuclei.

C.2. Nucleus Segmentation. Inspired by our ability to find filters that segmented melanoma from the ISBI-ISIC challenge as presented in Section 4.2, we ran PatchNet on the original un-segmented 512 \times 512 \times 1 images of BJ and NIH3T3 cells with a patch size of 17 \times 17 \times 1 in order to determine if there were filters able to segment the nuclei without any preprocessing.

Representative examples of unsegmented nucleus images are shown in Figure 14. Note the following two interesting properties of these unsegmented images: (1) NIH3T3 images tend to have more nuclei per image than BJ images; (2) there are noticable “halo” effects, creating a region of higher pixel values around the nuclei, which makes the segmentation task difficult using standard segmentation methods.

Figure 15 and 16 show the results of two particular filters of PatchNet when trained on classifying between the unsegmented images. These filters accurately segment out the nuclei as dark ovals regardless of halo effects and regardless of the fact that 2 BJ nuclei appear together in an image. Figure 16 shows that filter 57 is able to segment out the cell nuclei and at the same time identifies the texture features that identify NIH3T3 cells, namely the bright spots in the cell nuclei. These segmentation results are quite remarkable, taking into account that the model was not trained on labeled data for nucleus segmentation.

APPENDIX D. SOFTWARE

We ran our models on PyTorch [26] 0.1.12 on Python 3.6 packaged under Anaconda [24] 4.3.17 with Nvidia driver version 375.51, Cuda version 8.0.61 and CuDNN version 5.1.0.

We used SciKit-Learn [16], Numpy [22] and Mahotas [3] as tools while developing our models. We used Facebook's Visdom [27] as a tool to visualize filters and display them for this paper.

APPENDIX E. HARDWARE

For training, we used one server running Ubuntu 16.04.2 with an Intel i7-4930K at 3.40GHz with 56GB DDR3 RAM and two Nvidia Titan x (Pascal) with 12GB of GDDRX RAM.

REFERENCES

[1] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In Knowledge Discovery and Data Mining (KDD), 2015.
[2] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In the Conference on Computer Vision and Pattern Recognition (CVPR), 2014.
[3] L.P. Coelho. Mahotas: Open source software for scriptable computer vision. Journal of Open Research Software. 1(1), p.e3. DOI: http://doi.org/10.5334/jors.ac
[4] Coelho L.P. et al. (2010) Structured Literature Image Finder: Extracting Information from Text and Images in Biomedical Literature. In: Blaschke C., Shatkay H. (eds) Linking Literature, Information, and Knowledge for Biology. Lecture Notes in Computer Science, vol 6004. Springer, Berlin, Heidelberg.
[5] G. Cooper, C. Aliferis, R. Ambrosino, J. Aronis, B. Buchanan, R. Caruana, M. Fine, C. Glymour, G. Gordon, B. Hanusa, J. Janosky, C. Meek, T. Mitchell, T. Richardson, and P. Spirtes. An evaluation of machine-learning methods for predicting pneumonia mortality. Artificial Intelligence in Medicine, 9(2):107–138, 1997.
[6] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016.
[7] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real world surfaces. ACM Transactions on Graphics, 18(1):1–34, 1999.
[8] J. G. Elmore, M. B. Barton, V. M. Moceri, S. Polk, P. J. Arena, and S. W. Fletcher. Ten-year risk of false positive screening mammograms and clinical breast examinations. The New England Journal of Medicine, 338: 1089-96, 1998.
[9] D. Gutman, N. C. F. Codella, E. Celebi, B. Helba, M. Marchetti, N. Mishra, and A. Halpern. Skin lesion analysis toward melanoma detection: A challenge at the International Symposium on Biomedical Imaging (ISBI) 2016, hosted by the International Skin Imaging Collaboration (ISIC), 2016; arXiv:1605.01397.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In the International Conference on Computer Vision (ICCV), 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In the Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[12] R. A. Hubbard, K. Kerlikowske, C. I. Flowers, B. C. Yankaskas, W. Zhu, D. L. Miglioretti. Cumulative probability of false-positive recall or biopsy recommendation after 10 years of screening mammography: a cohort study. Annals of Internal Medicine, 155(8):819-92, 2011.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization; The International Conference on Learning Representations (ICLR), 2015.

[14] S. Lazebnik, C. Schmid, and J. Ponce. Sparse texture representation using local affine regions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(8):2169–2178, 2005.

[15] T. Lei, R. Barzilay, and T. S. Jaakkola. Rationalizing Neural Predictions. Conference on Empirical Methods in Natural Language Processing, 2016.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondep, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research (JMLR) 12, pp. 2825-2830, 2011.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. In the International Journal of Computer Vision (IJCV), 2015.

[18] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013; arXiv:1312.6034.

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

[20] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The All Convolutional Net, 2014; arXiv:1412.6806.

[21] Y. Xu, H. Ji, and C. Fermüller. Viewpoint invariant texture description using fractal analysis. In the International Journal of Computer Vision (IJCV), 83(1):85–100, jun 2009.

[22] S. van der Walt, S. Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

[23] M. D. Zeiler and R. Fergus. (2013) Visualizing and understanding convolutional networks. In the European Conference on Computer Vision (ECCV) (1) 2014: 818-833.

[24] Anaconda Software Distribution. Computer software. Vers. 2-2.4.0. Continuum Analytics, Nov. 2016. Web. https://continuum.io

[25] Independent UK Panel on Breast Cancer Screening. The benefits and harms of breast cancer screening: an independent review. Lancet. 380(9855):1778-86, 2012.

[26] PyTorch. http://pytorch.org

[27] Visdom. https://github.com/facebookresearch/visdom

MIT INSTITUTE FOR DATA, SYSTEMS, AND SOCIETY, LABORATORY FOR INFORMATION AND DECISION SYSTEMS, MASSACHUSES INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MA, USA

E-mail address: aradha@mit.edu

Ann Arbor, MI 48103,
E-mail address: cpdurham@gmail.com

MIT, CAMBRIDGE, MA, USA

E-mail address: alican@mit.edu

MIT INSTITUTE FOR DATA, SYSTEMS, AND SOCIETY, LABORATORY FOR INFORMATION AND DECISION SYSTEMS, MASSACHUSES INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MA, USA

E-mail address: cuhler@mit.edu