Brain Ageing Heat Maps via Deep Taylor Decomposition

Applied Mathematics MAM4001W: Year Project

Daniel Taylor
TYLDAN008
Supervised by Dr Jonathan Shock

Department of Pure and Applied Mathematics
University Of Cape Town
16-09-2019

Plagiarism Declaration:

1. I know that plagiarism is a serious form of academic dishonesty.
2. I have read the UCT document Avoiding Plagiarism: A guide for students, am familiar with its contents and have avoided all forms of plagiarism mentioned there.
3. Where I have used the words of others, I have indicated this by the use of quotation marks.
4. I have referenced all quotations and properly acknowledged other ideas borrowed from others.
5. I have not and shall not allow others to plagiarise my work.
6. I declare that this is my own work.

Signature:

Abstract

Deep Learning has been used with great success in medical image analysis. In particular, the use of Convolutional Neural Networks (CNNs) has led to state-of-the-art diagnostic capabilities in clinical applications. In the case of any Neural Network, there is the looming issue of the 'black box' problem – the fact that we do not necessarily understand why decisions are made when they are made. The use of heatmapping with Deep Taylor Decomposition allows us to visualise which components of an image allow for certain decisions to be made by a network. We look at heatmaps generated from a Neural Network based on the decision of whether an MRI image displays an 'aged' brain or not. We compare the salient regions of the heatmap to predicted areas of brain ageing to determine how the network characterises brain ageing features.

Contents

1 Introduction

1.1 Brain Ageing

While most neurodegenerative diseases are presented with increased risk with age, it is well-known that ageing itself affects the structure and function of the brain regardless of the presence of disease. Mattson and Arumugam [1] give a summary of the most pertinent aspects of brain ageing. While the ageing of the brain roughly parallels that of other organs and systems in the body, it is arguably the most notable and exaggerated ageing pathway, with its effects largely manifesting themselves both cognitively and physically.

We are most interested in the physical (and particularly the more clearly visible) traits of the ageing brain. Aggregation of proteins such as amyloid plaques and tau tangles are common results of lifetime accumulations of reactive oxygen species in the body, manifesting themselves in the brain in a particularly aggressive manner. Such aggregations are seemingly causal to diseases such as Huntington's and Alzheimer's. DNA repair decreases with age, and it is thought that telomere shortening and other DNA damage is not only a biomarker of ageing, but a cause thereof. Neurogenesis (the formation of new neurons) and neural plasticity markedly decrease with age, leading to decreased cognitive ability. Inflammation generally increases in the brain with age, and is aggravated by cellular senescence and reactive oxygen species.

Other major physical changes in the aged brain include the thinning of the cortex and the dilation of the cerebral ventricles. In particular, the volumetric decrease in grey-matter in the brain which comes with age seems to be a hallmark of cognitive decline. These large-scale physical changes are those for which we will be looking in particular in the analysis of MRI images.

We see that there are many tell-tale signs of brain ageing, which help us to characterise exactly what we are looking for in the analysis of brain images. It is our goal to see if these are physical changes that our machine learning techniques will recognise.

1.2 Machine Learning in Medical Images Analysis

A major tool of modern medicine is the use of machine learning in medical image analysis. Litjens et al. [2] give a thorough breakdown of such techniques and summarises 300 major contributions to the field as of the date of publication. Major implementations of object recognition networks include those designed to find cancerous tumours from mammograms, MRI and fMRI scans, PET scans, and ultrasound imaging. Other abnormalities such as tissue fibrosis, aneurysms and cysts can also be detected.

Hallmarks of ageing are made visible to radiologists most prominently through the use of imaging techniques like MRI, PET and ultrasound. Ageing leaves its traces throughout the body, leaving evidence such as decreased bone density, degeneration of elastin and type I collagen [3, 4], and ageing of the brain as discussed in Section (1.1).

The most common architecture for neural networks in medical imaging is that of Convolutional Neural Networks (CNNs). These are able to break images down into characteristic areas, first by local properties in groups of pixels, then by larger and larger attributes of images. In doing so, CNNs allow accurate assessment of the presence of particular attributes or objects in images, and are widely regarded as the best Neural Net architecture for the job. Most state-of-the-art classifier networks, such as GoogLeNet use some convolutional structures.

1.3 Convolutional Neural Networks

In the case of colour images (for which they are most commonly used), Convolutional Neural Networks instead of manipulating vectors are used to manipulate tensors of rank 3. An input image of height h and width w has dimensions $h \times w \times 3$ because of the two spatial dimensions and three colour parameters corresponding to each pixel (RGB activations in the pixel).

The architecture of CNNs is comprised of the initial input layer, followed by some number of sequences of convolution layers and pooling layers with interspersed nonlinearity layers (for example, a sigmoid or ReLU function), followed then by at least one fully-connected layer – the last of which corresponds to the output layer. An example structure is shown below in Figure (1).

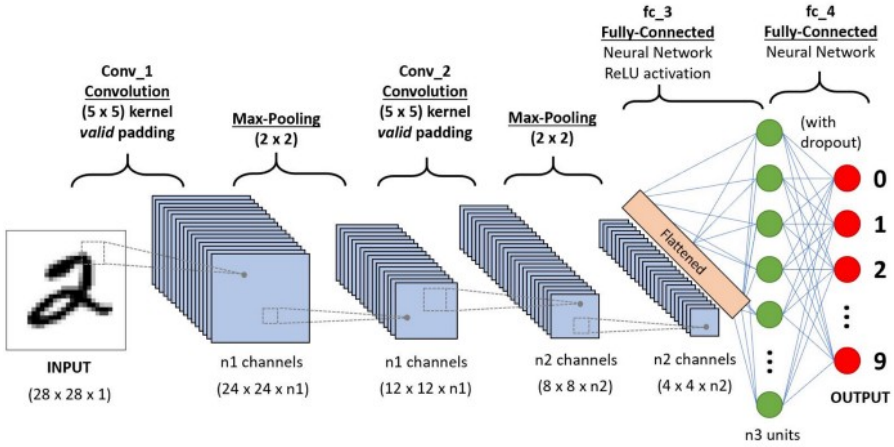


Figure 1: An example of the structure of a CNN used to classify handwritten digits from the MNSIT dataset. Image courtesy of `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`

Convolution layers have the effect of scanning for specific attributes in small sections of previous layers. In the case of facial recognition, the first convolution layer of a CNN might look for outlines of small features such as eyelashes or nostrils; later convolution layers in the network might look for larger attributes, such as compiling detection of the presence of an eye from the presence of eyelashes, brows, a pupil etc.

In the first convolution layer, the scan is executed by passing a filter tensor of weights over the input image. The filter tensor is of dimensions $h_1 \times w_1 \times 3$, typically with $h_1 = w_1 = 3$ or $h_1 = w_1 = 5$. The third dimension of the tensor is 3 in this case in order to correspond with the third dimension of the input tensor image (all colours of the image must be scanned through). The filter is passed over the input image from left to right, and top to bottom (as one reads in English) with stride size s, typically with $s = 1$ or $s = 2$. In this passing over, the filter is 'dotted'[1] with the corresponding region of the input image to produce a real number output for each position. For one filter, it is easy to see that the number of such outputs is

$$\left(\frac{h - h_1}{s} + 1\right) \times \left(\frac{w - w_1}{s} + 1\right) \equiv H_1 \times W_1$$

which we can arrange in a grid of such dimensions. For each filter in the convolution layer, there is a corresponding such grid, and we arrange these to form another tensor of rank 3. Thus the output

[1]The inner product is a rank-reducing operation on tensors by one. Here what we mean by the dot product is the sum of products of corresponding tensor elements, in the spirit of the dot product of vectors.

of the first convolution layer of the network is a tensor of dimensions $H_1 \times W_1 \times F$, where F is the number of desired filters in the layer. The intuition behind having multiple filters per convolutional layer is that each filter should learn to detect a specific feature at a given scale.

Subsequent convolution layers work in the same vain as the first, although the first does away with the colour separations in the third axis of the tensor, and for all subsequent layers, the third axis corresponds to different attribute searches from the most recent convolution layer.

For most CNNs, we are able to assume that the weights of a filter in a convolution layer can remain fixed with respect to the previous layer – that is to say, how we search for an attribute should not depend on where in the image we are looking. This allows us to reduce drastically the number of parameters in the convolution layers. This is not always exactly the case though; for example, in the use of facial recognition where the subject's face is symmetric in the input space. Then, for example, we will have to look for their ears differently one one side than on the other (because they are oriented differently on either side – a filter that is very good at detecting left ears is likely not very good at detecting right ears).

Pooling layers have the effect of reducing the dimensions of the tensor passing through, while conserving the rank, in order to reduce the number of parameters in the network. These are implemented by performing some operation on square grids in the matrices corresponding to the first two axes of the incoming tensor. Several methods of pooling have been proposed, the most commonly implemented of which are *sum pooling*, *average pooling* and *max pooling*. Max pooling is implemented by looking at $n \times n$ arrays in each matrix corresponding to a section of constant position on the third axis in the tensor. Commonly, we use $n = 2$, so we take the maximum value of the four elements in the subarray. The dimension of the third axis therefore remains the same after a pooling layer, and we reduce the dimensions of the first two axes by a factor of $\frac{1}{2}$. Sum pooling works by the same principle, but taking the sum of elements in the $n \times n$ array instead of the max element. Similarly, average pooling takes an average of such elements.

After the final pooling layer, the network has one or more fully connected layers, the last of which outputs the decision of the network in a vector.

2 Relevance Decomposition

The issue of understanding why Neural Networks make decisions has been around since the inception of this method of machine learning, and several methods have been developed to combat it. Heatmapping is a very useful method of determining the reasons for image recognition decisions. A heatmap is a measure of the relevance of pixels of an image to the corresponding decision by the network into which the image is fed.

It is important to note that although we can measure input layer relevances according to decisions, we as humans are still the means of interpretation for what these relevances signify with respect to what we are looking for. This makes the interpretation of the heatmaps inherently subjective. This is not a fault of the heatmap methods; in fact, the question of 'what makes this object what it is?' is more a question of 'what makes this object what I call it?' This means the task of image recognition in and of itself is based on the subjective classifications by humans of said objects. It is remarkable therefore that some image recognition software is even comparable to human-level recognition abilities with regards to images created by humans (such as the MNIST handwritten digit dataset); however, it makes sense that for something less subjective such as cancer screening (the *truth* of which is not subjective), some well-designed Neural Nets perform very well. The issue

remains though, that the interpretation of the relevance mappings is necessarily subjective, and necessitates the background knowledge of the interpreter on the subject at hand.

Montanvon et al. [5] explain the methods of Taylor Decomposition and Deep Taylor Decomposition on the layers of classifier Neural Networks. These are performed neuron-by-neuron in a layer-dependent manner so as to propagate relevance backwards through the network in a manner similar to backpropagation training. Figure (2) shows a basic schematic of how the backwards relevance propagation methods are executed.

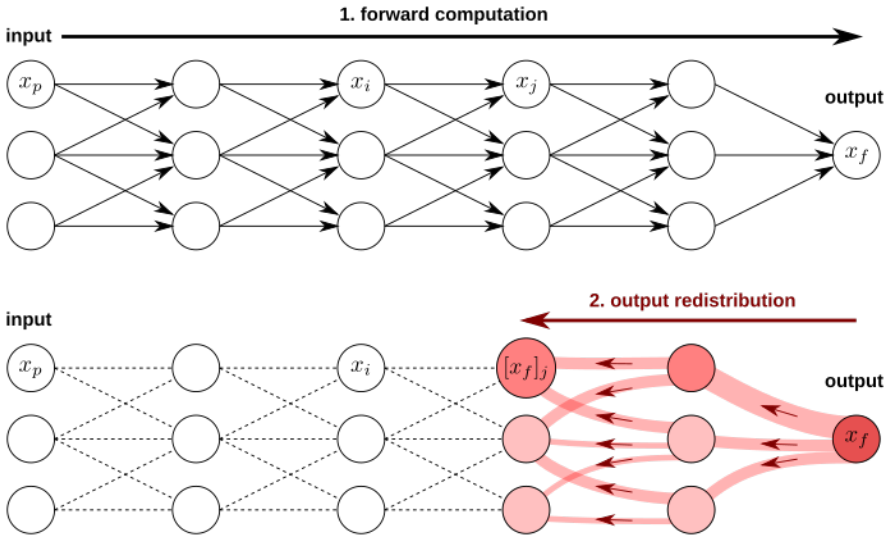


Figure 2: Redistribution of relevances across network layers and neurons. Courtesy of `http://www.heatmapping.org/deeptaylor/`

Generally, the method of relevance redistribution is a decomposition of the output value of the network onto the input variables. Since in our case the input variables correspond to the pixels of an image, we will refer to these input variables as the pixels. As is the convention of [5], the pixels are indexed by the variable p, such that the set of input pixels is $\{x_p\}$.

Consider a smooth positive-valued function $f : \mathbb{R}^d \to \mathbb{R}^+$. For our purposes, the input $\mathbf{x} = (x_p) \in \mathbb{R}^d$ of f is an image consisting of d pixels $|\{x_p\}| = d$. $f(\mathbf{x})$ indicates the presence of objects in the input image (in other words, the network f has been trained to desirable performance standards) with $f(\mathbf{x}) = 0$ indicating a lack of the object, and $f(\mathbf{x}) > 0$ indicating with proportional certainty the presence of the object in question.

The goal is to associate with each pixel x_p a relevance $R_p(\mathbf{x})$ which conveys the relevance of the pixel x_p to the output $f(\mathbf{x})$. Then, we define the heatmap of the image as $\mathbf{R}(\mathbf{x}) = (R_p(\mathbf{x}))$ having the same dimensions as $\mathbf{x}$, $(R_p(\mathbf{x})) \in \mathbb{R}^d$.

We would like our heatmaps to satisfy certain properties going forward. For one, we would like pixel values of $\mathbf{R}(\mathbf{x})$ to be non-negative, so that no pixels can contradict the presence of the object in question. We would also like for the total relevance to be unchanged, so that the heatmap relevance reflects the relevance of the object in question as output by our function f. We therefore define the following properties:

Definition 1. A heatmapping $\mathbf{R}(\mathbf{x})$ is called **conservative** if:

$$\forall \mathbf{x} : f(\mathbf{x}) = \sum_p R_p(\mathbf{x})$$

Definition 2. A heatmapping $\mathbf{R}(\mathbf{x})$ is called **positive** if:

$$\forall \mathbf{x}, p : R_p(\mathbf{x}) \geqslant 0$$

Definition 3. A heatmapping $\mathbf{R}(\mathbf{x})$ is called **consistent** if it is both conservative and positive.

2.1 Taylor Decomposition

If we have that f is an arbitrary differentiable function, we can look at the Taylor expansion of f around some root point $\tilde{\mathbf{x}}$, which is a point close to $\mathbf{x}$ with $f(\tilde{\mathbf{x}}) = 0$. In terms of the classifier Network, $\tilde{\mathbf{x}}$ is an image as similar as possible to $\mathbf{x}$ which the classifier will deem having no presence of the object in question. In terms of the input variable space, the vector $\tilde{\mathbf{x}}$ satisfies $f(\tilde{\mathbf{x}}) = 0$ while being close to $\mathbf{x}$ with respect to some metric in the space – one might think of this as the nearest root of f to $\mathbf{x}$. To first order, such a Taylor expansion is given by

$$f(\mathbf{x}) = f(\tilde{\mathbf{x}}) + \left(\frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}=\tilde{\mathbf{x}}}\right)^\top \cdot (\mathbf{x} - \tilde{\mathbf{x}}) + \varepsilon \tag{2.1}$$

where ε denotes terms of second order or higher. Then we can find the sum of elements $R_p(\mathbf{x})$ from the fact that:

$$\left(\frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}=\tilde{\mathbf{x}}}\right)^\top \cdot (\mathbf{x} - \tilde{\mathbf{x}}) = \sum_p \frac{\partial f}{\partial x_p}|_{\mathbf{x}=\tilde{\mathbf{x}}} \cdot (x_p - \tilde{x}_p) = \sum_p R_p(\mathbf{x})$$

and so

$$\mathbf{R}(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}}|_{\mathbf{x}=\tilde{\mathbf{x}}} \odot (\mathbf{x} - \tilde{\mathbf{x}}) \tag{2.2}$$

where $\odot$ denotes the element-wise product.

It is well-known that the gradient $\nabla f(x)|_{\mathbf{x}=\tilde{\mathbf{x}}}$ at the nearest root point to a vector $\mathbf{x}$ points in the same direction as the vector $(\mathbf{x} - \tilde{\mathbf{x}})$ (given that f is smooth). Hence, their element-wise product is non-negative, and so this definition of relevance propagation satisfies positivity according to Definition (2). It does not necessarily satisfy conservativeness, however, due to the possibility of higher-order terms possibly lurking in ε.

In general, we would like to work in a network composed of multiple layers, and so this method is not immediately implementable – finding the factors $\dfrac{\partial f}{\partial x_p}$ is by no means trivial in this case. We also need to tackle the issue of finding an appropriate root point $\tilde{\mathbf{x}}$.

2.2 Deep Taylor Decomposition

We introduce now the concept of Deep Taylor Decomposition, which expands on the Taylor Decomposition method, and aims to alleviate the issue of non-conservativeness. It is also generally applicable to multi-layer networks. It is easy to see that the function f learned by the network is in fact comprised of simpler subfunctions from one layer in the network to the next. This is a property of Deep Networks that is exploited by the method of Deep Taylor Decomposition. In fact, such subfunctions simply comprise affine functions of inputs followed by nonlinear activation functions such as ReLU or Softmax.

We can assume that the function $f(\mathbf{x})$ encoded by the output neuron $\mathbf{x}_f$ can be decomposed onto the set of neurons at a given layer, and denote the neurons in this layer $\mathbf{x}_j$ with associated relevance $\mathbf{R}_j$. We then consider the problem of decomposing $\mathbf{R}_j$ onto the set of neurons one layer lower in the network, $\mathbf{x}_i$. For visual supplementation to this, refer back to Figure (2). *Assuming that there is a function relating these two quantities*, $R_j((x_i))$, we can perform the desired decomposition by Taylor expansion.

We define a root point of the function, $(\tilde{x}_i)^{(j)}$, using an index (j) to denote the fact that each of the neurons x_j in the current layer requires its own root point. Then the Taylor decomposition is given by

$$R_j = \left(\frac{\partial R_j}{\partial (x_i)}|_{(\tilde{x}_i)^{(j)}}\right)^\top \cdot \left((x_i) - (\tilde{x}_i)^{(j)}\right) + \varepsilon_j = \sum_i \frac{\partial R_j}{\partial x_i}|_{(\tilde{x}_i)^{(j)}} \left(x_i - \tilde{x}_i^{(j)}\right) + \varepsilon_j \tag{2.3}$$

where again, ε_j denotes terms of order 2 or higher. Now, we realise that

$$\sum_i \frac{\partial R_j}{\partial x_i}|_{(\tilde{x}_i)^{(j)}} \left(x_i - \tilde{x}_i^{(j)}\right) = \sum_i R_{ij}$$

where R_{ij} is the relevance propagated from a neuron x_j to one x_i a layer below. So we can recover the relevance for the neuron x_i from

$$R_i = \sum_j R_{ij}$$

to get that

$$R_i = \sum_j \frac{\partial R_j}{\partial x_i}|_{(\tilde{x}_i)^{(j)}} \left(x_i - \tilde{x}_i^{(j)}\right) \tag{2.4}$$

This method of layer-to-layer decomposition is conservative, by Definition (1), as long as $\sum_i R_{ij} = R_j$; that is, as long as the terms ε_j vanish. Thus, if each layer-to-layer Taylor decomposition is conservative as such, the full Deep Taylor Decomposition is conservative. Similarly, since we saw before that the Taylor Decomposition satisfies the positivity of Definition (2), the full Deep Taylor Decomposition satisfies positivity. Thus, as long as the terms ε_j vanish in each layer-to-layer propagation, the entire decomposition is consistent, according to Definition (3). Indeed, the terms ε_j do vanish, since we saw that the subfunctions between layers are in fact affine.

2.2.1 Behaviour in a Single-Layer Network

In Figure (3), we see the schematic of a neural net with one hidden layer, which is a detection-pooling layer with definition

$$x_j = \max\left\{0, \sum_i x_i w_{ij} + b_j\right\}; \quad x_k = \sum_j x_j \tag{2.5}$$

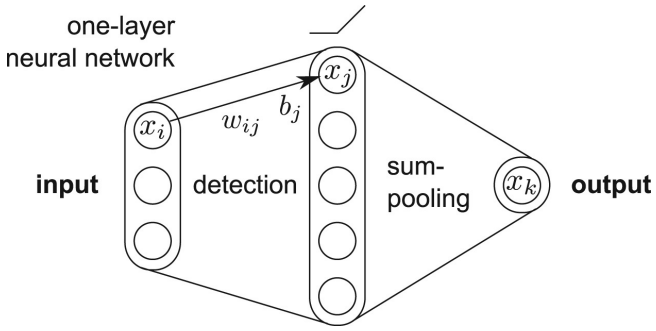


Figure 3: Image courtesy of the original Deep Taylor Decomposition paper, [5]

Redistribution of the relevance $R_k = x_k$ onto the hidden layer neurons begins with the expression

$$R_k = \sum_j x_j$$

which follows from above. Thusly, we can recover the relevances R_j from the original Taylor decomposition formula, giving us

$$R_j = \frac{\partial R_k}{\partial x_j}|_{(\tilde{x}_j)} (x_j - \tilde{x}_j) \tag{2.6}$$

Now, in the case of the sum-pooling layer, we can simply find a root point $(\tilde{x}_j)$ from the necessities that

1. $\sum_j \tilde{x}_j = 0$,
2. All components of the root point must be non-negative, due to the expectation of the use of the $\max(0, \cdot)$ function in the hidden layer.

The only such point is $(\tilde{x}_j) = \mathbf{0}$. With this choice, and using the fact that the sum-pooling method gives $\frac{\partial R_k}{\partial x_j} = 1$, we find the redistribution rule

$$R_j = x_j \tag{2.7}$$

so in the pooling layer the relevance is redistributed on the detection layer neurons in proportion to their activation values. We now look to decompose relevances onto the neurons of the input layer, $\{x_i\}$.

From the relation in Equation (2.7) and the definition of the activations of the hidden layer, we can describe the relevances R_j in terms of the input neuron activations by

$$R_j = \max\left\{0, \sum_i x_i w_{ij} + b_j\right\} \tag{2.8}$$

To redistribute the relevances R_i, we can now Taylor expand this relation according to Equation (2.4) to get

$$R_i = \sum_j \frac{\partial R_j}{\partial x_i}|_{(\tilde{x}_i)^{(j)}} \left(x_i - \tilde{x}_i^{(j)}\right) \tag{2.9}$$

We are posed again with the task of finding the root point $(\tilde{x}_i)^{(j)}$.

The w^2-Rule

The simplest case is an unconstrained space from which to choose $(\tilde{x}_i)^{(j)}$. We can in this case choose the closest root of f to (x_i) by the Euclidean metric or any equivalent. In the case that R_j is non-zero, the nearest root $(\tilde{x}_i)^{(j)}$ of $\mathbf{R_j}$ is given by the intersection of the plane equation $\sum_i \tilde{x}_i^{(j)} w_{ij} + b_j$ and the line of steepest descent $(\tilde{x}_i)^{(j)} = (x_i) + t\mathbf{w_j}$, for $t \in \mathbb{R}$, and where $\mathbf{w_j}$ is the vector of weights between the neurons (x_i) and the single neuron x_j. This intersection occurs when:

$$\begin{aligned}
&\sum_i \left[x_i w_{ij} + t(w_{ij})^2\right] + b_j = 0 \\
&\Rightarrow t = \frac{-1}{\sum_i (w_{ij})} \left[\sum_i x_i w_{ij} + b_j\right] \\
&\Rightarrow (\tilde{x}_i)^{(j)} = \left(x_i - \frac{w_{ij}}{\sum_i (w_{ij})} \left[\sum_i x_i w_{ij} + b_j\right]\right).
\end{aligned}$$

So the rule for relevance redistribution onto the x_i neurons becomes:

$$R_i = \sum_j \frac{w_{ij}^2}{\sum_{i'} w_{i'j}^2} R_j \tag{2.10}$$

This is true of course even when $R_j = 0$, such that there is no relevance contribution to x_i from an x_j.

The z^+-Rule

In the first case of domain restriction, we examine the case where the input domain is restricted to $\mathbb{R}^+$. This occurs, for example, in the spaces following ReLU activations. If we define the vector $(x_i)^-$ as the negatively weighted components of (x_i) with zeros elsewhere, this has the property that $f_i((x_i^-)) = \sum_i x_i^- w_{ij} + b_j$ is zero or negative, since these are the negatively weighted components of (x_i) (here, f_i is the sub-function of f acting on the (x_i) vector in the feedforward section of the network)[2]. Then the line segment $\left((x_i), (x_i^-)\right)$ has at least one root point of f_i.

Now, we are looking for the intersection of the plane equation $\sum_i \tilde{x}_i^{(j)} w_{ij} + b_j$ and the line described by $(\tilde{x}_i)^{(j)} = (x_i) + t((x_i) - (x_i^-)) = (x_i) - t(x_i^+)$, where now (x_i^+) is the vector of positively weighted components of (x_i) and zeros everywhere else. This can also be described as $(x_i^+)(w_{ij}) = (x_i)(w_{ij}^+)$. In this notation:

[2]Note that we do require here that the biases b_j are non-positive – this is an easy fix and does not affect anything other than the manner in which the Network is optimised.

$$\sum_i \left[x_i w_{ij} - t x_i w_{ij}^+ w_{ij}\right] + b_j = 0$$
$$\Rightarrow t = \frac{1}{\sum_{i'} x_{i'} w_{i'j}^+ w_{i'j}} \left[\sum_i x_i w_{ij} + b_j\right]$$
$$\Rightarrow (\tilde{x}_i)^{(j)} = \left(x_i - \frac{x_i w_{ij}^+}{\sum_{i'} x_{i'} (w_{i'j})^2} \left[\sum_i x_i w_{ij} + b_j\right]\right)$$

since summing over terms with $w_{ij}^+ w_{ij}$ is the same as summing over the terms instead with $(w_{ij}^+)^2$. Then, the propagation rule becomes:

$$R_i = \sum_j \frac{z_{ij}^+}{\sum_{i'} z_{i'j}^+} \tag{2.11}$$

where we have defined $z_{ij}^+ \equiv x_i w_{ij}^+$.

The $z^{\mathcal{B}}$-Rule

The pixel spaces of image recognition networks are often box-constrained. This means that the pixels are in the domain $\mathcal{B} = \{(x_i) : l_i \leqslant x_i \leqslant h_i \forall i \in [1, ..., d]\}$, where the lower bounds $l_i \leqslant 0$ and the upper bounds $h_i \geqslant 0$ are respectively the lowest and highest values that a pixel x_i can take. Now we restrict the search for the root point to the line segment $((l_i 1_{w_{ij}>0} + h_i 1_{w_{ij}<0}), (x_i))$, where the vector $1_{w_{ij}>0}$ is the d-vector with ones where w_{ij}^+ is non-zero and zeros elsewhere, and $1_{w_{ij}<0}$ is the d-vector with ones where w_{ij}^- is non-zero and zero elsewhere. Similarly to before, the vector $(x_i^-) \equiv (l_i 1_{w_{ij}>0} + h_i 1_{w_{ij}<0})$ has $f_i((x_i^-)) \leqslant 0$, and so somewhere on the line segment is a root point of f_i. Now, $(\tilde{x}_i)^{(j)}$ can be found at the point of intersection between the plane equation $\sum_i \tilde{x}_i^{(j)} w_{ij} + b_j$ and the line $(\tilde{x}_i)^{(j)} = (x_i) + t((x_i) - (x_i^-))$. Then:

$$\sum_i \left[x_i w_{ij} - t x_i w_{ij} + t(l_i w_{ij}^+ + h_i w_{ij}^-)\right] + b_j = 0$$
$$\Rightarrow t = \frac{1}{\sum_{i'} \left[x_{i'} w_{i'j} - l_{i'} w_{i'j}^+ - h_{i'} w_{i'j}^-\right]} \left[\sum_i x_i w_{ij} + b_j\right]$$
$$\Rightarrow (\tilde{x}_i)^{(j)} = \left(x_i - \frac{x_i - l_i 1_{w_{ij}>0} - h_i 1_{w_{ij}<0}}{\sum_{i'} \left[x_{i'} w_{i'j} - l_{i'} w_{i'j}^+ - h_{i'} w_{i'j}^-\right]} \left[\sum_i x_i w_{ij} + b_j\right]\right)$$

Since summing over terms with factors of $1_{w_{ij}<0} w_{ij}^-$ is the same as summing simply over terms instead with w_{ij}^- (and similarly for terms with $1_{w_{ij}>0} w_{ij}^+$, we sum instead over w_{ij}^-). Then our relevance propagation rule becomes:

$$R_i = \sum_j \frac{z_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_{i'} \left[z_{i'j} - l_{i'} w_{i'j}^+ - h_{i'} w_{i'j}^-\right]} R_j \tag{2.12}$$

where $z_{ij} \equiv x_i w_{ij}$.

Proposition 1. For all learnable functions f of a single-layer network, the w^2-rule is consistent in the sense of Definition (3).

Proof. We will first show that the w^2-rule is conservative, in the sense of Definition (1). To this end, note that

$$\begin{aligned}
\sum_i R_i &= \sum_i \left[\sum_j \frac{w_{ij}^2}{\sum_{i'} w_{i'j}^2} R_j \right] \\
&= \sum_j \frac{\sum_i w_{ij}^2}{\sum_{i'} w_{i'j}^2} R_j \\
&= \sum_j R_j \\
&= \sum_j x_j \\
&= f((x_i))
\end{aligned}$$

where we have assumed that the weights' squares w_{ij}^2 do not sum to 0 for any j (i.e. they are not all zero).

Now we show that the w^2-rule is positive in the sense of Definition (2). Observe that

$$\begin{aligned}
R_i &= \sum_j \frac{w_{ij}^2}{\sum_{i'} w_{i'j}^2} R_j \\
&= \sum_j \underbrace{w_{ij}^2}_{>0} \underbrace{\frac{1}{\sum_{i'} w_{i'j}^2}}_{>0} \underbrace{R_j}_{\geqslant 0} \\
&\geqslant 0
\end{aligned}$$

In the case that for some j', $\sum_i w_{ij'}^2 = 0$, then all weights $w_{ij'}$ must be zero, and so since the biases $b_{j'} \leqslant 0$, we will have $R_{j'} = 0$. Then there is no relevance to redistribute to the lower layer from this neuron. In either case, we see that the w^2-rule is both conservative and positive, and so is consistent. □

Proposition 2. For all learnable functions f of a single-layer network and input activations $x_i \in \mathbb{R}^+$, the z^+-rule is consistent in the sense of Definition (3).

Proof. The proof for this rule is identical to that for the w^2-rule. In the case that $\sum_i z_{ij}^+ > 0$, we simply replace w_{ij}^+ in the previous proof by z_{ij}^+. In the case where $\sum_i z_{ij}^+ = 0$, then similarly to before, we must have that all $z_{ij} \leqslant 0$, so that due to the negative biases b_j the relevance $R_j = 0$ and there is no relevance to propagate to the lower layer. □

Proposition 3. For all learnable functions f of a single-layer network and input activations $x_i \in \mathcal{B}$, the $z^{\mathcal{B}}$-rule is consistent in the sense of Definition (3).

Proof. For the slightly more complex $z^{\mathcal{B}}$-rule, $R_i = \sum_j \frac{z_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_{i'} \left[z_{i'j} - l_{i'} w_{i'j}^+ - h_{i'} w_{i'j}^- \right]} R_j$, we first show that the numerator is non-negative for all $(x_i) \in \mathcal{B}$:

$$\begin{aligned}
z_{ij} - l_i w_{ij}^+ - h_i w_{ij}^- &= x_i(w_{ij}^+ + w_{ij}^-) - l_i w_{ij}^+ - h_i w_{ij}^- \\
&= \underbrace{w_{ij}^+}_{\geqslant 0} \underbrace{(x_i - l_i)}_{\geqslant 0} + \underbrace{w_{ij}^-}_{\leqslant 0} \underbrace{(x_i - h_i)}_{\leqslant 0} \\
&\geqslant 0
\end{aligned}$$

From here, the proof is the same as those for the w^2- and z^+-rules. Simply replacing w_{ij}^2 with $z_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-$ gives that for the sum over such i components being greater than 0, the $z^{\mathcal{B}}$-rule is both positive and conservative. Then in the case that for some j', $\sum_i z_{ij'} - l_i w_{ij'}^+ - h_i w_{ij'}^- = 0$, we must have $z_{ij'} - l_i w_{ij'}^+ - h_i w_{ij'}^- = 0$ for each i. We then have three cases:

1. $x_i = h_i$ and $w_{ij'}^+ = 0$. Then the contribution of the input to the neuron $x_{j'}$ is $z_{ij'} = h_i w_{ij'} \leqslant 0$.
2. $x_i = l_i$ and $w_{ij'}^- = 0$. Then the contribution of the input to the neuron $x_{j'}$ is $z_{ij'} = l_i w_{ij'} \leqslant 0$.
3. $w_{ij} = 0$. Then there is no contribution to the input neuron $x_{j'}$ and $z_{ij'} = 0$.

Thus, the total contribution to the neuron $x_{j'}$ is $z_{j'} = \sum_i z_{ij'} \leqslant 0$, and since the biases $b_{j'} \leqslant 0$, we must have $R_j = x_j = 0$, and so there is no relevance to propagate backwards.

In either case, we see that the $z^{\mathcal{B}}$-rule is consistent. □

2.2.2 Behaviour in Deep Networks

We have assumed up to this point that we are able to find functions relating activations in one layer to the relevances in the higher layer (as emphasised at the beginning of Section (2.2)). It is not always the case that such functions are known a priori; for example, the convolutional layers of a CNN are not always understood before empirical analysis of the network. Here – to the end of making our relevance mappings explicit – we explore the use of relevance models. These are functions mapping sets of neuron activations in a given layer to the relevance of a neuron in a higher layer (usually just the layer directly higher), and whose outputs can be redistributed onto the input variables in order to allow backward relevance propagation.

The first of the two relevance models explained in [5] is the Min-max Relevance Model. Consider the feature extractor of a deep network, feeding forward from a layer of neurons (x_i) to (x_j), then to x_k and subsequently (x_l). We would like to try to predict the relevance R_k from the activations (x_i) so as to relate relevances between higher and lower layers. This model aims to approximate the relevance R_k from (x_i) in a given layer by the output $\hat{R}_k$ of a function defined by

$$y_j = \max\left\{0, \sum_i x_i v_{ij} + a_j\right\},$$
$$\hat{R}_k = \sum_j y_j$$

where $a_j = \min\{0, \sum_l R_l v_{lj} + d_j\}$ is a non-positive bias parameter dependent on weights connected to the higher layer (x_l), and the relevances of that layer. The negative bias prevents the activation of (y_j) in the case that none of the upper-level abstractions of (R_l) correspond to detected features in (x_i). In other words, if the relevances (R_l) are independent of (x_i), we do not propagate relevance back to these neurons. The introduced parameters $\{v_{ij}, v_{lj}, d_j\}$ are learned by the relevance model in the minimisation of the optimisation function

$$\min\langle(R_k - \hat{R}_k)^2\rangle.$$

Clearly, this requires further training on a network which itself has likely already had to be trained for a significant amount of time. Instead of training such a model for each neuron of a network, we would like to exploit global properties of the network to build a relevance model.

The Training-Free Relevance Model

Consider now a feature extractor in a deep network defined by:

$$x_j = \max\left\{0, \sum_i x_i w_{ij} + b_j\right\}, x_k = ||(x_j)||_q$$

and the layer (x_k) feeds forward into a layer (x_l). Here, the norm $||\cdot||_q$ can represent any sort of pooling function, such as average, max or sum pooling. Assume that relevance has been redistributed onto the uppermost of these layers by the z^+-rule, such that we know (R_l). Then we can write

$$R_k = \sum_l \frac{x_k w_{kl}^+}{\sum_{k'} x_{k'} w_{k'l}} R_l.$$

Now, letting $\sum_j x_j = ||(x_j)||_1$, we have

$$\begin{aligned} R_k &= ||(x_j)||_q \frac{\sum_j x_j}{\sum_{j'} x_{j'}} \sum_l \frac{R_l w_{kl}^+}{\sum_{k'} x_{k'} w_{k'l}^+} \\ &\equiv \left(\sum_j x_j\right) c_k d_k \end{aligned}$$

where $c_k \equiv \dfrac{||(x_j)||_q}{||(x_j)||_1}$ is an L^q/L^1 norm ratio and $d_k \equiv \sum_l \dfrac{R_l w_{kl}^+}{\sum_{k'} x_{k'} w_{k'l}^+}$. The term d_k describes the ratio between relevances and activations in a layer in terms of the same activations and their outgoing weights, and the relevances in higher layers. Montavon et al. describe d_k as a 'top-down contextualisation' term.

We argue that the terms c_k and d_k can be modelled as constant under perturbations to in the lower activations (x_j). Thusly, we are able to formulate a training-free relevance model. In this derivation, it is important to note why we have made use of the z^+-rule for relevance propagation. Not only is this feasible in propagations to lower hidden layers, but it in particular allows us to write the relevance R_k in terms of the sum of the elements x_j, which could not be done for the w^2- and $z^\mathcal{B}$-rule.

We will now give arguments as to why the variables c_k and d_k can be modelled as constants.

- $c_k = \dfrac{||(x_j)||_q}{||(x_j)||_1}$. We note first that simply rearranging the components of a vector leaves its norm unchanged. We see that for a general perturbation of the components x_j in a dimensionally large vector (x_j), we can approximate the change in component magnitudes simply as a swapping of components in the vector. If the dimension of the vector is large enough and the perturbation approximately random, then this swapping of components model of the vector perturbation leaves the L^q/L^1 ratio approximately constant.

- $d_k = \sum_l \dfrac{R_l w_{kl}^+}{\sum_{k'} x_{k'} w_{k'l}}$. The only ways in which the components x_j affect this term is through the quantities R_l and $x_{k'}$. It is plain to see that small perturbations in (x_j) will cause very little change in R_l, and so we say $\dfrac{\partial R_l}{\partial x_j} \simeq 0$. Since we are only concerned with the effect on

$x_{k'} = x_k$, and assuming that there is a large number of neuron activations being summed over in this denominator, we can argue that the effect of perturbations on the sum over k' terms is negligible.

With this training-free relevance model, we have the same structure of layer-to-layer relevance propagation (up to the constant factors c_k and d_k) for feature extractors as we had in our simple single-layer networks. Specifically, since the pooling function from (x_j) to (x_k) has no weights:

$$R_j = \frac{x_j}{\sum_{j'} x_{j'}} R_k$$

so that the relevance R_k is redistributed onto the R_j by proportion of neuron activations x_j. Then:

$$R_i = \sum_j \frac{q_{ij}}{\sum_{i'} q_{i'j}} R_j$$

where we have introduced q_{ij} as

$$q_{ij} = \begin{cases} w_{ij}^2 & w^2\text{-rule} \\ x_i w_{ij}^+ & z^+\text{-rule} \\ x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^- & z^{\mathcal{B}}\text{-rule} \end{cases}$$

If we choose to decompose R_i via the z^+-rule again onto the layer below, we would be able to use the same training-free model all the way through the network up to the input layer. Thus, we can use the training-free relevance model to redistribute relevances through all hidden layers of a network, then at the input layer use the w^2- or $z^{\mathcal{B}}$-rule to redistribute the lowest hidden-layer relevances onto $\mathbf{R}_p$.

2.2.3 Application to the MNIST Dataset

Application of the Deep Taylor method of relevance decomposition to the MNIST dataset of handwritten digits is very straightforward to implement. This is in no small part due to the ease with which one can create a Neural Network that very accurately classifies the test images of the dataset.

One Fully-connected Neural Net was created in this vein with one input layer of $28 \times 28 = 784$ input neurons, a decision layer of 10 output neurons, and two hidden dense layers – the first having 300 neurons and the second, 100 neurons. This gives a total of 266610 parameters (both weights and biases), all of which are trainable. The model was trained over five epochs to an accuracy of 98.89%.

The other network trained on this dataset was a Convolutional Neural Net with an input layer as a $1 \times 28 \times 28 \times 1$ tensor. This is followed by a 2-dimensional convolution window of dimensions 3×3, producing a $1 \times 26 \times 26 \times 32$ tensor; the next layer is an average-pooling layer with output a $1 \times 13 \times 13 \times 32$ tensor; then there is another 3×3 convolution window with output a $1 \times 11 \times 11 \times 64$ tensor, followed by another average-pooling layer which outputs a $1 \times 5 \times 5 \times 64$ tensor; then there is a final 3×3 convolution window which outputs a $1 \times 3 \times 3 \times 64$ tensor; this is then flattened to a layer of 576 neurons, which is connected to a fully-connected layer of 64 neurons, itself connected to the final (output) fully-connected layer of 10 neurons. This gives a total of 93322 trainable parameters. The model was trained over five epochs to an accuracy of 99.03%.

The summaries of these networks are given in the tables below. Both networks were subjected to Deep Taylor Decomposition on the *correct* predictions of several randomly chosen test images. The resulting heatmaps are shown alongside their corresponding MNIST images in Figures (4) and

(5). These heatmaps are normalised to pixel relevances in the range $[0, 1]$.

```
Fully-Connected Network Summary:
Layer (type)                 Output Shape              Param #
=================================================================
flatten1 (Flatten)           (None, 784)               0

dense3 (Dense)               (None, 300)               235500

dense4 (Dense)               (None, 100)               30100

dense5 (Dense)               (None, 10)                1010
=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
```

```
Convolutional Network Summary:
Layer (type)                 Output Shape              Param #
=================================================================
conv2d3 (Conv2D)             (None, 26, 26, 32)        320

averagepooling2d2 (Average (None, 13, 13, 32)        0

conv2d4 (Conv2D)             (None, 11, 11, 64)        18496

averagepooling2d3 (Average (None, 5, 5, 64)          0

conv2d5 (Conv2D)             (None, 3, 3, 64)          36928

flatten1 (Flatten)           (None, 576)               0

dense2 (Dense)               (None, 64)                36928

dense3 (Dense)               (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

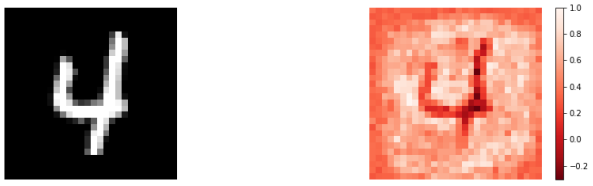
In the heatmap images of Figures (4) and (5), the darker red colours are associated with lower relevance, while the lighter colours are associated with higher relevance.

We see that in the Fully-connected Network, based on the hypothesis of the Taylor decomposition, the decision is made largely by considering the area around the figure itself – in a squared area which seems to characterise the space in which white pixels are likely to appear in the input space. The square area in question is smaller than the full image, and the boundaries of the image are always completely disregarded in terms of relevance. It is interesting to note the disregard in

these heatmaps of the digit curve itself.

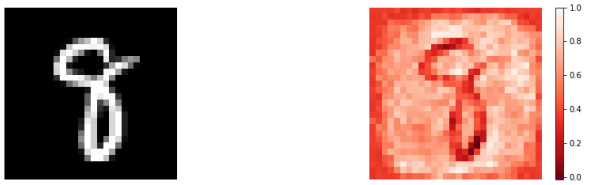
We see in the CNN that the decision is made largely considering the space very close to the lines comprising the digits themselves. In several of these example images, we see that not only is relevance limited to the curves themselves, but to 'characteristic' parts of the curves. For example, in 5q), we see the heatmap of the number 8 character, in which the most relevant parts are considered to be the topmost arc and the left curve of the upper 'circle' of the character – by contrast, the areas considered least relevant are the joints of characteristic lines of the curve, such as the upper left corner of the character, and the crossing of the lines in the middle of the character. Relevance in these images is clearly much more localised to the characters themselves and very little is considered in the surrounding space. This is positive evidence for the theory that CNNs tend to focus on particular regions (and then subregions, and so on) and their relevance to the decision, in relation to one-another.

In Figure (5), we also see the heatmaps from the output neuron corresponding to what was considered to be the least likely decision for the digit (for example, in 5c), the image was decided to be least likely a 6). These 'Anti-heatmaps', with pixel relevances normalised to $[-1, 0]$, display negative relevances in the input layer. It is interesting to note that in the *normalised* versions, the heatmaps and their anti-heatmaps are exact negatives. The sum of corresponding (normalised) pixel relevances gives 0 for every pixel. Not only is this true for the least likely decisions, but also for every other possible decision which was not the one made. Perhaps a useful way of thinking about this is that we ask the Network, "Why is this not a six?" To which the network replies, "Because it is a four."



a) 4 b) Heatmap

c) 7 d) Heatmap

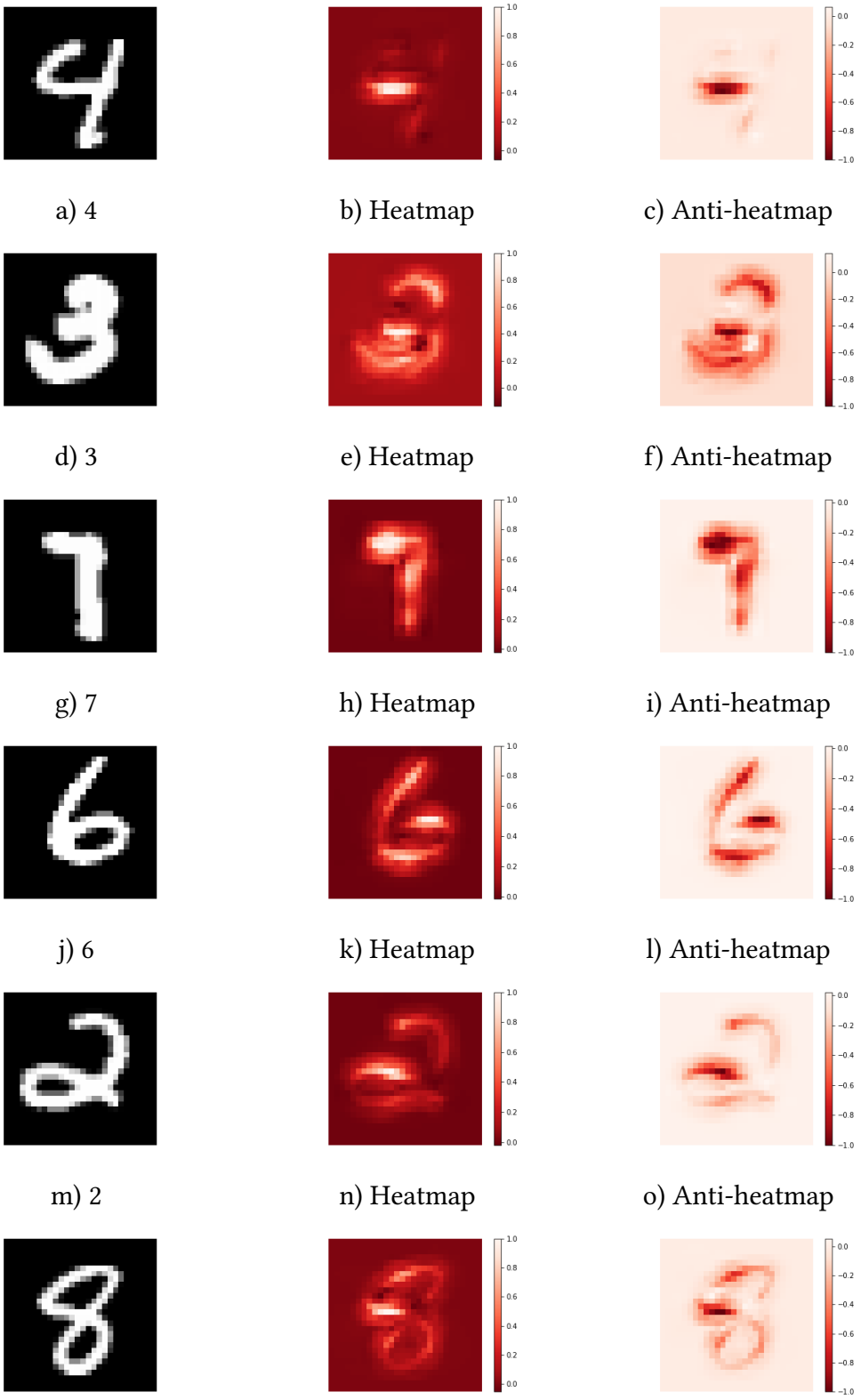


e) 8 f) Heatmap

Figure 4: Normalised Heatmaps for three positive predictions of MNIST samples via the Fully-connected Network

We see by comparison of the heatmaps from the two networks, that the Fully-connected Network seems to focus on the shape of the space *not* occupied by the character, which potentially *could be*

occupied by it, while the CNN seems to focus on the characteristics of the curves themselves which make up the digit, with relevance in the latter case being much more localised.



a) 4 b) Heatmap c) Anti-heatmap

d) 3 e) Heatmap f) Anti-heatmap

g) 7 h) Heatmap i) Anti-heatmap

j) 6 k) Heatmap l) Anti-heatmap

m) 2 n) Heatmap o) Anti-heatmap

p) 8 q) Heatmap r) Anti-heatmap

Figure 5: Normalised Heatmaps and 'Anti-heatmaps' for several positive MNIST samples via the CNN

In Figure (6), we see one of the test set images of the MNIST dataset which was incorrectly

classified by the CNN, alongside the corresponding heatmap. The image was of a handwritten 3, and was misclassified as a 5. We now delve deeply into the realm of subjectivity to examine why this misclassification took place. This is of course anecdotal, but the image is seemingly ambiguous – particularly insofar as it could be construed either as a 5 or a 3. The point though is not to defend the integrity of the CNN's decision-making. We see that in the heatmap, apart from the middle horizontal segment of the character, the most salient area of pixels would seem to be the top right corner of the character. Here, it would appear that the network is interpreting the *lack* of downward continuation of the curve at this point (as in the upper arc of the character '3') indicates that the character is less likely a 3. Due to the geometry of the character further down, then, the most likely choices for the network to make are 3 and 5 (incidentally, these were the two leading probabilities, with 5 leading by a small margin). The final, erroneous, decision was thus that the character must be a 5.

(a) MNIST sample 3 incorrectly classified as a 5

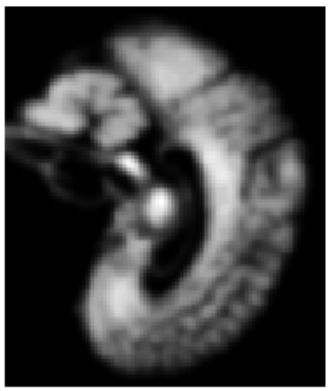
(b) Heatmap

Figure 6

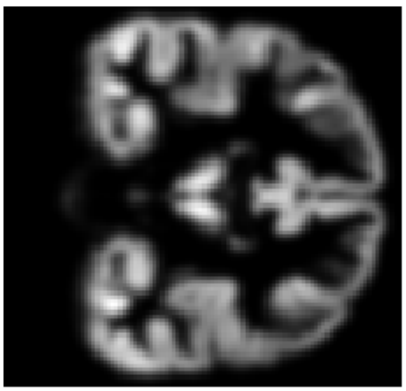
3 Heatmaps For Brain Ageing

We now look to the task of examining salient areas of brain ageing. To this end, we will implement the heatmaps relevance propagation on the decisions of a neural net which acts as a classifier (according to age) of MRI image data. The datasets used were supplied by PHOTON AI via the 2019 PAC competition. The task of the competition was to create a CNN classifier of the MRI data by age with the objective of minimising the network's Mean Absolute Error (MAE).

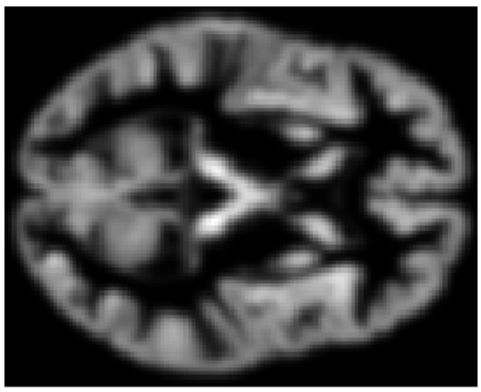
The MRI image data used in training the networks was the grey-matter MRI image set from the PAC competition. These were stored as a single numpy array of dimensions $342 \times 100 \times 124 \times 104$ – that is, 342 scans of dimensions $100 \times 124 \times 104$. In Figure (7) are shown some cross-sections of a randomly chosen subject's gray-matter image.



(a) Example GM image: sagittal section



(b) Example GM image: frontal section



(c) Example GM image: transverse section

Figure 7

3.1 CNN Structure and Background

In order to extract details about subject ages from the MRI data, a CNN was constructed according to the same general structure used in [6]. This paper in fact precipitated the PAC competitions. The structure of the corresponding CNN was of repeated blocks, each consisting of a 3D convolutional layer, followed by a ReLU activation, then another 3D convolutional layer, followed by a Batch-norm layer, then another ReLU activation, and finally a 3D average pooling function. In the CNN concerned in this report, four such blocks were implemented. This resulted in a network of 253291 parameters, of which 253051 were trainable. The 240 non-trainable parameters are from the Batch-normalisation layers, which are used only in training to determine the mean and variance of moving batch variables. There are 240 corresponding learned variables from the Batch-normalisation layers – the *learned* means and variances. The structure of the CNN is given below:

```
Layer (type)                 Output Shape              Param #
=================================================================
Conv3d1a (Conv3D)            (None, 98, 122, 102, 8)   224
```

```
Conv3d1b (Conv3D)            (None, 96, 120, 100, 8)   1736

BatchNorm1 (BatchNormalizat (None, 96, 120, 100, 8)   32

Relu1 (ReLU)                 (None, 96, 120, 100, 8)   0

AvePool1 (AveragePooling3D) (None, 48, 60, 50, 8)     0

Conv3d2a (Conv3D)            (None, 46, 58, 48, 16)    3472

Conv3d2b (Conv3D)            (None, 44, 56, 46, 16)    6928

BatchNorm2 (BatchNormalizat (None, 44, 56, 46, 16)    64

Relu2 (ReLU)                 (None, 44, 56, 46, 16)    0

AvePool2 (AveragePooling3D) (None, 22, 28, 23, 16)    0

Conv3d3a (Conv3D)            (None, 20, 26, 21, 32)    13856

Conv3d3b (Conv3D)            (None, 18, 24, 19, 32)    27680

BatchNorm3 (BatchNormalizat (None, 18, 24, 19, 32)    128

Relu3 (ReLU)                 (None, 18, 24, 19, 32)    0

AvePool3 (AveragePooling3D) (None, 9, 12, 9, 32)      0

Conv3d4a (Conv3D)            (None, 7, 10, 7, 64)      55360

Conv3d4b (Conv3D)            (None, 5, 8, 5, 64)       110656

BatchNorm4 (BatchNormalizat (None, 5, 8, 5, 64)       256

Relu4 (ReLU)                 (None, 5, 8, 5, 64)       0

AvePool4 (AveragePooling3D) (None, 2, 4, 2, 64)       0

flatten (Flatten)             (None, 1024)              0

dense1 (Dense)               (None, 32)                32800

dense2 (Dense)               (None, 3)                 99
=================================================================
Total params: 253,291
Trainable params: 253,051
Non-trainable params: 240
```

The output of the network was three nodes (trained with softmax activation). These represented three bins, $(0, 1, 2)$, into which the subjects were categorised by age. These bins were respectively

ages 10-39, 40-69, and 70-100. These bins loosely represented demographics of 'young', 'middle-aged' and 'elderly'. The oldest participants were 90 years old, and the youngest was 17. The overall dataset was heavily skewed towards the first bin (with about $\frac{2}{3}$ of all subjects lying in the first bin). Thus, the training set was truncated so as to contain the same number of members of each bin. Anecdotally, when attempting to train the network with the full datasets, the network ended up learning to classify every subject into the first bin, achieving a 66.67% accuracy in a profound local minimum of the error function.

The provided data from the PAC competition was a file of raw data, one of purely grey-matter data, and another of white-matter data. It was noted in [6] that the greatest accuracy was achieved from training with the grey-matter data only.

This network was trained over 34 epochs to an accuracy of 72.22%, minimising Sparse Categorical Crossentropy (as opposed to the MAE, as in [6]). This is not a remarkable accuracy overall, but interestingly, on the set of elderly subjects, the network performed with 100% accuracy; on the set of young subjects, the network performed with 81.58% accuracy; and on the middle-aged subjects, it performed with a dismal 35.09% accuracy – not much better than guessing. This would suggest that the network's ability to detect some features corresponding to the age of the brain is significant, but the network still finds the middle-aged subjects' images ambiguous.

As a matter of fact, we are most interested in the comparison between the two extreme groups in any case, and so for these two, we have fairly good accuracies from our network.

3.2 Heatmaps

Two concerns regarding the relevance propagation of this network are:

1. The change from 2D to 3D convolutions and pooling functions;
2. How to propagate relevance backward through a Batch Normalisation layer.

Thankfully, the first concern is laid to rest by the fact that the 3D analogues for these layer types have very much the same layout as the 2D versions.

To address the second concern, we look at what exactly the BatchNorm layer does. In training, the BatchNorm layer is used to ensure that weights, biases and activations do not get too big. This helps avoid the famous 'exploding gradient' problem. To this end, the BatchNorm layer normalises and standardises the moving activations to have zero mean and unit variance. Thus, the layers *learn* two parameters γ and β in training, and *calculate* two parameters μ and σ^2 (the mean and variance) from the training data. These values are then used in the feedforward portion of the network on input variables x_i as follows:

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$
$$y_i = \gamma \hat{x}_i + \beta$$

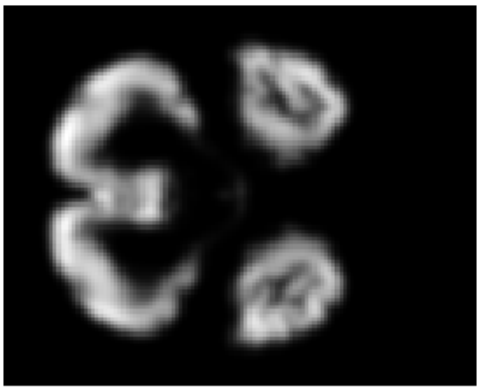
and the layer activations are y_i. Here, ε is a small parameter used to ensure that we do not divide by zero. In practive, ε was set to 10^{-9}.

We can see from the structure of the BatchNorm layer, that activations are simply affine functions of the input variables. Thus, since there is no mixing of activations from the inputs, the relation

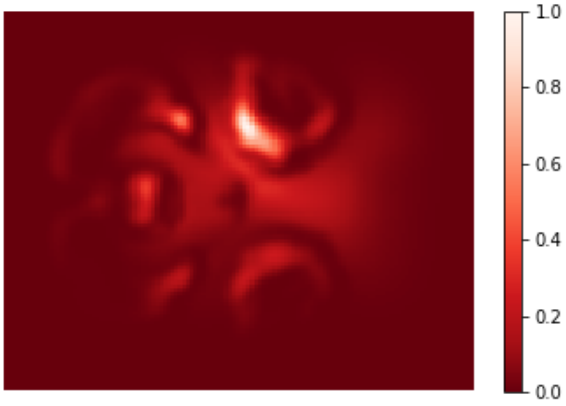
between the relevances of the layers must be linear. Furthermore, for lower-level activations x_i and BatchNorm activations x_j, the corresponding relevances must satisfy $\sum_i R_i = \sum_j R_j$. Therefore we have as a solution:[3]

$$R_i = R_j. \tag{3.1}$$

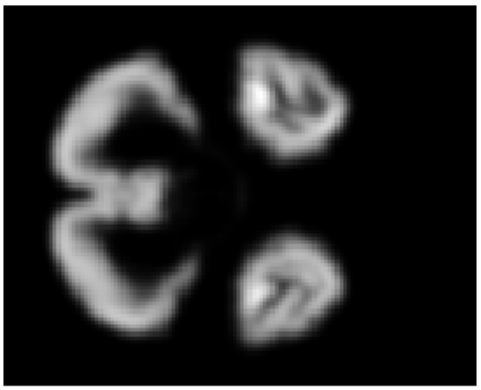
We now finally have all we need to form the heatmaps of the MRI age decisions. Figures (8), (9), (10) and (11) show comparisons of the MRI sections and corresponding heatmaps of an elderly individual versus a young individual, for four transverse grey-matter image sections. Again, we have normalised the relevance data such that the shown relevances fall in the range $[0, 1]$. The two subjects in question were *correctly* categorised by the network. This is important, since we are examining the relevances of what the network believes to show youth or age in the MRI images.



(a) Elderly individual MRI section



(b) Heatmap



(c) Young individual MRI section

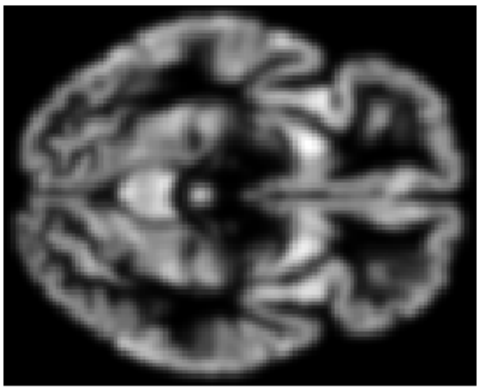
(d) Heatmap

Figure 8

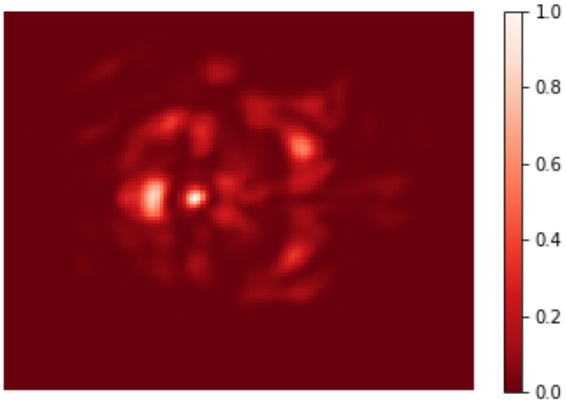
The relevance propagations are demonstrably consistent, in the sense of Definition (3). The maximum activation in the final layer of the CNN (without the softmax activation) for the elderly

[3]This is a slight oversimplification, but for the case of this network (which is relatively simple), the relation holds well, as we will see. [7] shows that in fact, for some more complex networks, this method of propagation through BatchNorm layers is not necessarily consistent, and shows a method to rectify this issue. This is beyond the scope of this report.

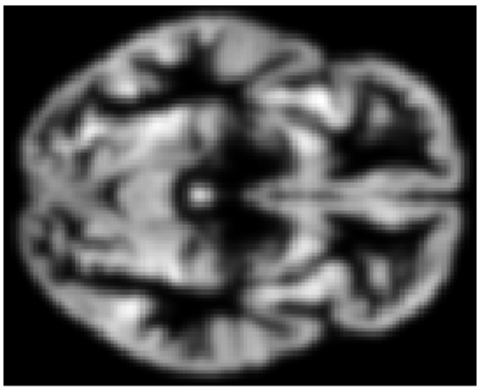
individual was ~ 9.3623, and the final propagated relevance also summed to ~ 9.3623; with corresponding to a total absolute error in relevance propagation of $\sim 0.0008\%$. For the young individual, the maximum activation in the final layer was ~ 6.6078, and the final propagated relevance summed to ~ 6.6079, corresponding to a total absolute error in relevance propagation of $\sim 0.0007\%$.



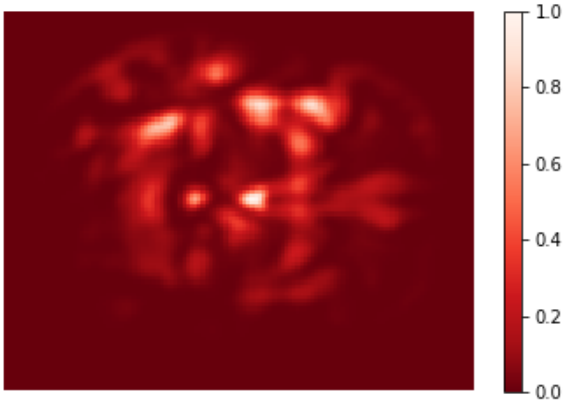
(a) Elderly individual MRI section



(b) Heatmap

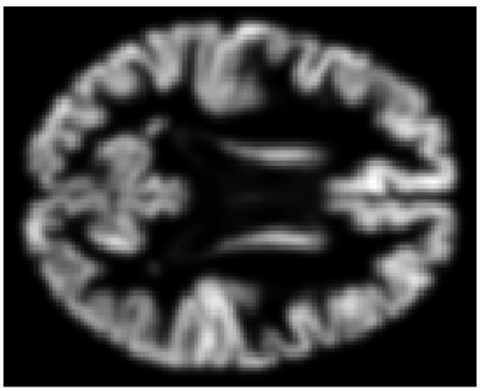


(c) Young individual MRI section

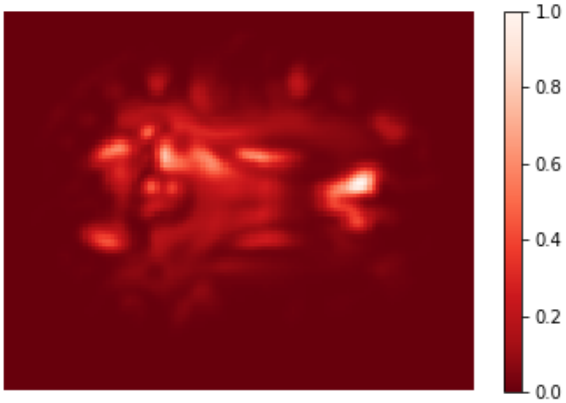


(d) Heatmap

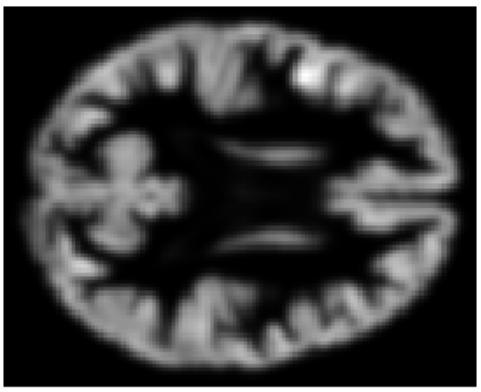
Figure 9



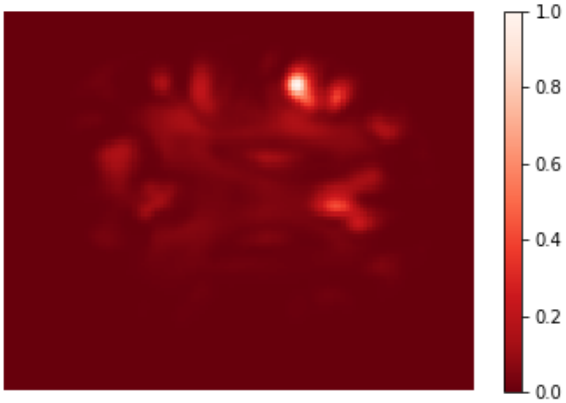
(a) Elderly individual MRI section



(b) Heatmap

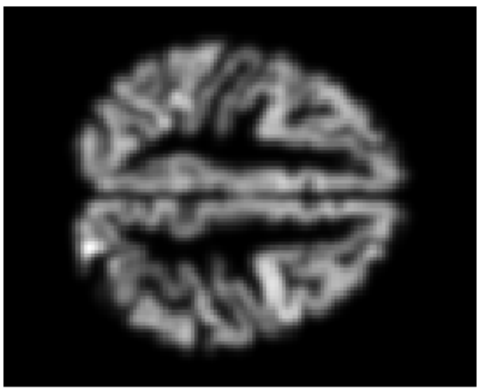


(c) Young individual MRI section

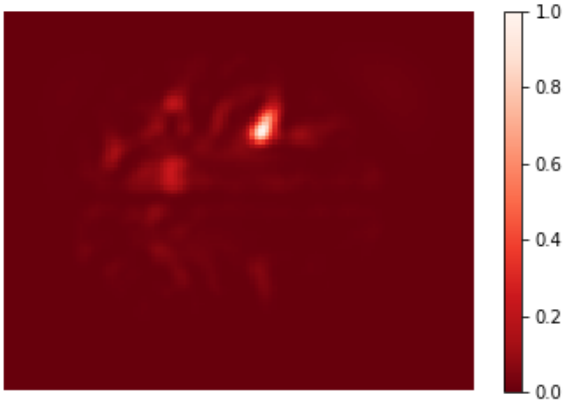


(d) Heatmap

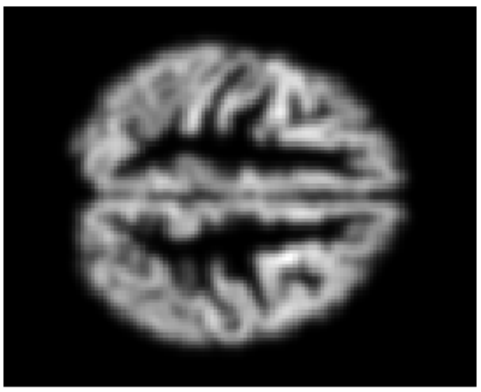
Figure 10



(a) Elderly individual MRI section



(b) Heatmap



(c) Young individual MRI section

(d) Heatmap

Figure 11

3.3 Analysis of Findings

Examination of Figures (8)-(11) shows that the network seems to base its decisions on the densities of grey-matter on the outside of distinguishing structures, such as the cerebrum in Figures (9)-(11), and the cerebellum in Figure (8). A particular focus of the network in the images is grey-matter density in the gyri and sulci (the peaks and troughs of the outer folds of the brain structures). In Figures (9) and (10), a main focus also seems to be the size and shape of grey-matter presence in the ventricles (the cavities in the brain containing cerebrospinal fluid).

The particular interest of the network in the grey-matter density in the sulci hearkens back to the discussion about the effect of brain ageing on grey-matter density. As we can see by comparison of the original MRI images for the elderly and younger subjects, the grey-matter density of the younger subject is much higher (as expected), but particularly so in the sulci. In fact, in Figure (11), this is the only area of relevance to the network for both subjects.

The focus on the neural ventricles is also suggestive of the discussion on brain ageing. The ventricles of the elderly subject are clearly much larger than those of the younger, and have less even distribution of grey-matter. In Figure (10) this is shown particularly clearly, where the network fo-

cuses very much on the size and shape of the ventricles of the elderly patient (among other areas).

In Figure (8), the network is clearly heavily focused on the grey-matter density in the arms of the left and right temporal lobes (the isolated globules on the right of the MRI images of this figure). The network seems to focus on the fact that the density is much higher in the younger subject than in the elder.

4 Discussion

We have derived the method of Deep Taylor Decomposition, which proves to be a consistent form of layer-wise relevance propagation for analysis of network decision-making. We showed the basic premise of the method on a CNN classifying handwritten digits from the MNIST dataset, and examined the implications of the resulting heatmaps. We also saw an example of how heatmaps can show us intuitively how and why networks can make incorrect decisions.

We ended with the application of the relevance decomposition to a CNN trained to classify brain MRI images by age bracket; and this showed us the focus of the network on specific areas of the brain images in this age determination. To this extent, the relevance heatmaps seemed to agree with the expected areas of brain ageing salience.

The code developed and used in the implementation of the networks and the associated heatmaps for this report are available at the following sites:

MNIST fully-connected network:
`https://github.com/DanielTaylor97/HonoursProject/blob/master/FCN_MNISTheatmaps.ipynb`
or
`https://colab.research.google.com/drive/1PjoaxZ5fj7Kut82GAOA6IibM-ecHeV1M`

MNIST CNN:
`https://github.com/DanielTaylor97/HonoursProject/blob/master/CNN_MNISTheatmaps.ipynb`
or
`https://colab.research.google.com/drive/1Gxyx1HkWEOqJJuiuE5DlNw8YvtiYsn9T`

MRI data CNN:
`https://github.com/DanielTaylor97/HonoursProject/blob/master/projectCNN.ipynb`
or
`https://colab.research.google.com/drive/1rg_39BdPrx67605AvXlE-Kq6i95yugmz`

5 Conclusion

To expand on this report, the first order of business would be to train a more complex CNN for the MRI data, to act as a better classifier for more narrow age bins. While it is infeasible to get a high-accuracy network to implement such predictions with the accuracy of a single year at this point, networks such as those submitted to the PAC competition perform far better than the one implemented in this report. The task then would be to implement the heatmaps on these far more complex architectures, some of which have layer types not dealt with in the scope of this report.

To this end, it would be of aid to go into detail on the findings and methods developed in [7] to propagate relevance backward through the network more precisely in complex network structures including BatchNorm layers.

The analysis of the heatmaps could also be improved with the input of a radiologist. The opinions of a highly trained individual would be invaluable to the relevance of the focuses of the networks. Comparing the opinion of a professional to the relevance redistribution of a competent CNN decision would help to see if there are improvements that must be made to the architectures of such networks in order to focus more directly on salient areas of brain ageing.

Deeper and more specific analyses of MRI images could also be implemented. As opposed to the broad spectrum of brain ageing phenotypes, one could look in specific at a diease of brain ageing, for example; such as Alzheimer's Disease. Then one could use methods of relevance propagation to determine salient areas of the effect of disease on the brain according to the classifier.

The incidence of diseases of ageing, and particularly cognitive decline, has been increasing in developed parts of the world. This is a terribly emotionally taxing experience for all involved in such cases. The development of new technologies and techniques for combating such diseases is of utmost import in the world of healthcare. As we extend the lifespan of humans with the use of new technological developments, we must remember too to increase the *healthspan* of humans – that is, the length of time of a person's life for which they are active and healthy, and able to enjoy the life they lead. As we progress in our development and use of machine learning and other forms of artificial intelligence, we ultimately are aiming to a brighter future in all aspects of our lives, through the aid provided by these tools.

Acknowledgements

Many thanks to Dr Matthias Treder for a) his sharing of the cropped numpy images for the grey- and white-matter training data; and b) access to his CNN submission for the PAC competition for 2019, off of which my simpler CNN was largely based.

References

[1] Mark P Mattson and Thiruma V Arumugam. Hallmarks of brain aging: adaptive and pathological modification by metabolic states. *Cell metabolism*, 27(6):1176–1199, 2018.

[2] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[3] Laëtitia Gorisse, Christine Pietrement, Vincent Vuiblet, Christian EH Schmelzer, Martin Köhler, Laurent Duca, Laurent Debelle, Paul Fornès, Stéphane Jaisson, and Philippe Gillery. Protein carbamylation is a hallmark of aging. *Proceedings of the National Academy of Sciences*, 113(5):1191–1196, 2016.

[4] Stéphanie Baud, Laurent Duca, Brigida Bochicchio, Bertrand Brassart, Nicolas Belloy, Antonietta Pepe, Manuel Dauchez, Laurent Martiny, and Laurent Debelle. Elastin peptides in aging and pathological conditions. *Biomolecular concepts*, 4(1):65–76, 2013.

[5] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[6] James H Cole, Rudra PK Poudel, Dimosthenis Tsagkrasoulis, Matthan WA Caan, Claire Steves, Tim D Spector, and Giovanni Montana. Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker. *NeuroImage*, 163:115–124, 2017.

[7] Lucas YW Hui and Alexander Binder. Batchnorm decomposition for deep neural network interpretation. In *International Work-Conference on Artificial Neural Networks*, pages 280–291. Springer, 2019.