Model agnostic interpretability

Jessica Rumbelow

A thesis submitted for the degree of PhD
at the
University of St Andrews

2024

Abstract

This thesis explores the development and application of model-agnostic interpretability methods for deep neural networks. I introduce novel techniques for interpreting trained models irrespective of their architecture, including Centroid Maximisation, an adaptation of feature visualisation for segmentation models; the Proxy Model Test, a new evaluation method for saliency mapping algorithms; and Hierarchical Perturbation (HiPe), a novel saliency mapping algorithm that achieves performance comparable to existing model-agnostic methods while reducing computational cost by a factor of 20. The utility of these interpretability methods is demonstrated through two case studies in digital pathology. The first study applies model-agnostic saliency mapping to generate pixel-level segmentations from weakly-supervised models, while the second study employs interpretability techniques to uncover potential relationships between DNA morphology and protein expression in CD3-expressing cells.

Acknowledgements

This work is supported by the Industrial Centre for AI Research in Digital Diagnostics (iCAIRD) which is funded by Innovate UK on behalf of UK Research and Innovation (UKRI) [project number: 104690].

Thank you Oggie, my constant champion, mentor, and friend. Thank you David, for your faith in me and for making this work possible. Most of all, thank you Jamie, my love, for everything.

Declaration

Candidate's Declaration

Supervisor’s Declaration

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree. I confirm that any appendices included in the thesis contain only material permitted by the ’Assessment of Postgraduate Research Students’ policy.

Date: 18th March 2024

Signature of supervisor:

Date: 18th March 2024

Signature of candidate:

Date: 18th March 2024

Signature of supervisor:

Underpinning Research Data or Digital Outputs

I, Jessica Rumbelow, hereby certify that no requirements to deposit original research data or digital outputs apply to this thesis and that, where appropriate, secondary data used have been referenced in the full text of my thesis.

Date: 18th March 2024

Signature of candidate:

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

As machine learning is applied to increasingly high-stakes domains, the ability to accurately explain model behaviour in a human-interpretable way is becoming evermore important [9, 109, 154]. If we are to enable the safe and widespread adoption of powerful and potentially life-saving AI applications in medicine and beyond, we must develop interpretability methods that can identify, for example, if a model classifies a biopsy image as malignant due to cell morphology or due to a smudge on the slide [39, 21]. Moreover, better understanding models during training and deployment is vital to improving them – interpretability methods can flag biases or spurious correlations learned from training data that standard evaluation metrics miss. Machine learning interpretability can also be used for *knowledge discovery*[38, 14]. If a model can be trained to perform some task that humans cannot, some pattern must exist in the data that humans do not yet understand – and with robust interpretability techniques, it becomes possible to parse what the model has learned. In this way deep learning can be re-framed, not just as a tool for automation, but as a lens through which complex patterns can be made visible to humans.

In Chapter 1 I introduce deep learning and provide some background information about deep learning, applications in digital pathology, and approaches to interpretability – with particular emphasis on *model-agnostic* methods (techniques that do not make assumptions about or require access to the model's underlying architecture, and so can be applied to any machine learning model). Chapter 2 explores model-agnostic *global* interpretability, specifically feature visualisation. I introduce the concept of learned *class prototypes* and use toy models to demonstrate how a model's architecture affects what it learns during training. I also propose an adaptation of the standard feature visualisation algorithm, enabling its application to segmentation models. In Chapter 3, I explore model-agnostic *local* interpretability in the form of saliency mapping, identify some current limitations of different saliency mapping techniques, and explain the difficulty of objectively assessing interpretability methods of this kind. Chapter 5 introduces a new evaluation metric to address this difficulty, and Chapter 4 presents Hierarchical Perturbation, a novel model-agnostic saliency mapping algorithm. This algorithm is evaluated against existing saliency mapping methods, and demonstrates comparable results on standard benchmarks with significantly faster performance.

The remainder of this thesis is given over to two case studies in which I apply these interpretability methods to machine learning tasks in digital histopathology. Chapter 6 demonstrates how saliency mapping can be employed to generate high-resolution pixel-level segmentation from weakly-supervised images. Chapter 7 shows how interpretability techniques can be used for knowledge discovery in both segmentation and classification tasks, identifying heretofore unknown morphological features with high discriminative power in Hoechst [119, 35] stained slides. Finally, I summarise the overall significance of this research and briefly discuss limitations, future work, and clinical translation potential in Chapter 8.

1.2 Deep Learning and Digital Pathology

Digital Pathology

Pathology is the study of disease. It involves examining bodily fluids, tissues, and organs, and plays a key role in the diagnosis, management and treatment of diseases [77]. The process begins with the collection of a sample from a patient. This could be a tissue biopsy, a swab, bodily fluids like blood or urine, or even an entire organ – the method of collection depends on the suspected disease and its location in the body. These samples then undergo a process of fixation to preserve cellular structures and prevent degradation. Formalin, a solution of formaldehyde in water, is the most commonly used fixative. Following fixation, the tissue is embedded in paraffin wax, and cut into very thin sections which are placed on to glass microscope slides. These sections are typically 4-5 micrometers thick, allowing for the visualisation of cellular structures under a microscope.

These slides are then stained, to highlight specific cellular components and structures. *Hematoxylin and Eosin* (H&E) is the most commonly used stain in histopathology. As shown in Figure 1.1 Hematoxylin stains cell nuclei blue, while eosin stains the cytoplasm and extracellular matrix pink.

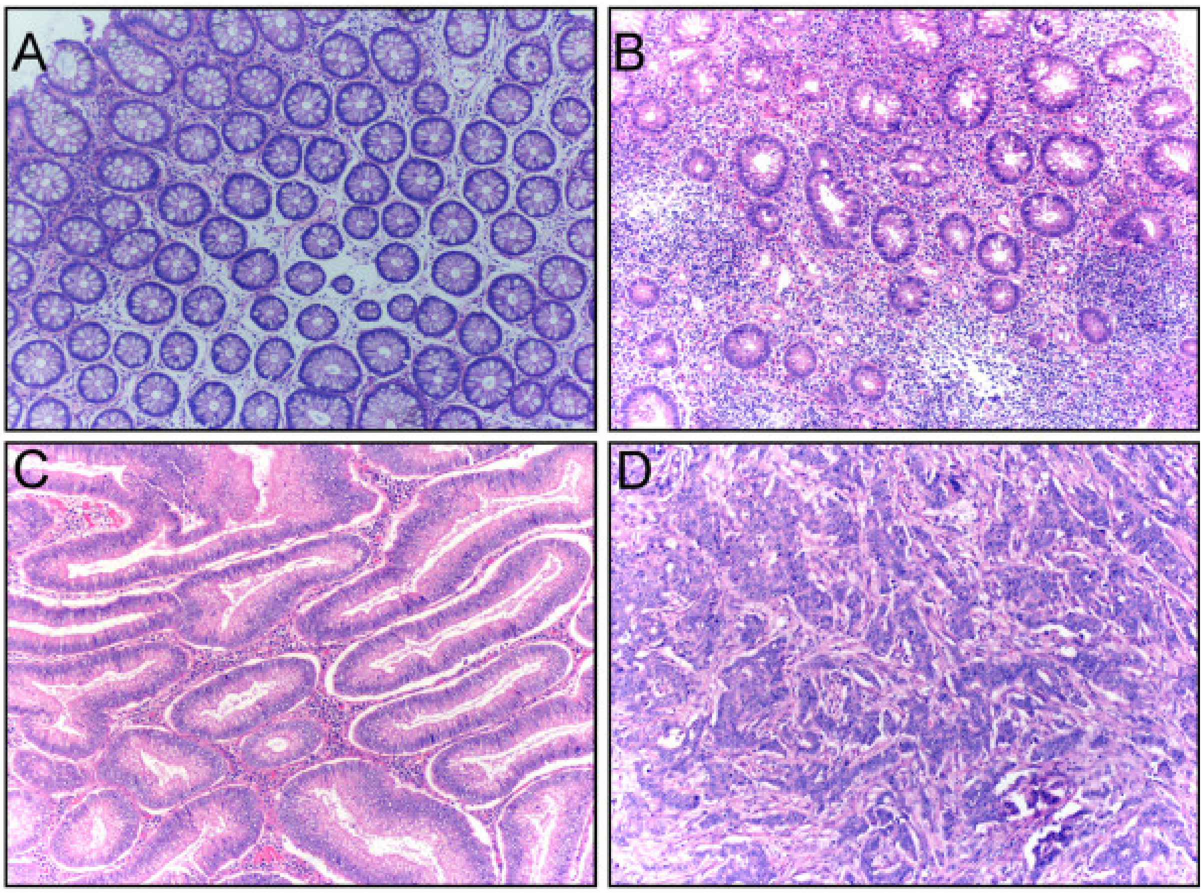


Figure 1.1: Examples of hematoxylin and eosin (H&E) staining of healthy tissues (A), ulcerative colitis (B), adenomas (C) and adenocarcinomas (D) (magnification ×100) from Cammarota et al. [31].

Pathologists then manually analyse these stained slides under a microscope. They look at cellular architecture, the morphology of individual cells, and the presence or absence of specific structures to arrive at a diagnosis. In some cases, further specialised techniques might be required, such as *immunohistochemistry* (IHC) to detect specific proteins, as described in more detail in Chapter 7.

Once the examination is complete, the pathologist compiles their findings into a pathology report describing the observed microscopic features and providing a diagnosis. It may also include recommendations for further tests or treatments, and in complex cases, slides might be reviewed by more than one pathologist or sent for consultation to experts in specialised areas of pathology.

The slides are then labelled and stored in a specialised physical cabinet for future reference or research projects. Slides must be kept in a controlled environment to prevent degradation of the tissue samples. Factors like temperature, humidity, and light exposure need to be regulated, and some facilities might even use climate-controlled rooms or cabinets for this purpose [110]. This method has clear limitations: physical slides are costly to store, they can degrade, and are difficult to share [13].

With the advent of digital pathology, this workflow has seen significant changes. Once the tissue is fixed, embedded, sectioned, and stained, it is then scanned using high-resolution slide scanners to produce digital images. These images can be viewed on a computer screen, eliminating the need for a physical microscope [11]. This digitisation offers several advantages, including the ability to easily and quickly share images, facilitating collaboration between researchers or consultations between pathologists; and to store images efficiently at scale on servers or cloud platforms for future reference [157, 13].

Furthermore, digital pathology has facilitated the introduction of advanced image analysis tools. These tools can not only make the visual examination of the tissue easier, quicker and more physically comfortable for pathologists [54], but even automatically identify and quantify specific structures or patterns in the tissue, aiding pathologists in their diagnostic process. For instance, algorithms can be designed to recognise cancerous cells, measure tumour margins, or quantify specific biomarkers. This not only enhances the accuracy of the diagnosis but can also speed up the diagnostic process [108, 49].

This transition has also led to the creation of large private and public repositories of digital slides. The United States National Cancer Institute's Cancer Imaging Archive (TCIA) is one such, holding 30.9 million radiology images from 37,568 patients [7]. Cancer Image Europe aims to do something similar in Europe [1]. The Image Data Resource (IDR) is another, hosting 364 TB consisting of over 13.5 million images from 121 published scientific studies to date, predominantly human tissue [5, 165]. Other, smaller but no less rich datasets abound: one example is The Pathology Atlas [6, 156], which hosts survival information for nearly 8000 cancer patients covering 17 major types of cancer, plus examples of protein expression patterns for 216 tumours representing the 20 most common forms of human cancer – but there are many more [57]. These are incredibly rich resources, and it is this volume of accessible digitised information that has made made the application of machine learning and deep learning possible in pathology [11].

Deep Learning

Machine learning (ML) is a subset of artificial intelligence that focuses on the development of algorithms and statistical models that enable computers to perform tasks without explicit instructions. Instead, these systems learn to make predictions based on patterns in their training data. The core idea behind machine learning is to train a model using a dataset. This model, once trained, can then make predictions or decisions without being explicitly programmed to do so. The "learning" in machine learning refers to the ability of the algorithm to improve its performance over time as it is exposed to more data [76], as described in Section 7.3.

There are various types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the algorithm is trained on a labeled dataset, meaning that each data sample is accompanied by the *correct* output – i.e., what the model *should* output in response to that sample. This thesis is concerned with supervised learning of this kind. In contrast, unsupervised learning deals with unlabeled data and seeks to find patterns or relationships within it. Reinforcement learning involves an agent that learns to make decisions by taking actions in an environment to maximise a reward [132].

Deep learning (DL) is a subset of machine learning that employs neural networks with many layers (hence 'deep'). The primary advantage of deep learning models is their ability to automatically identify relevant features from data, without manual feature extraction (where data scientists decide which parts of a dataset are relevant for a prediction task). This is particularly useful for tasks involving images, where manual feature extraction is challenging, and for very large, rich datasets, where it may not be clear ahead of time which parts of a raw data sample are most relevant for a given task.

Training Neural Networks

Given a neural network and a dataset, the training process happens through the iterative adjustment of the network's parameters to minimise the discrepancy between the network's output and the true label. This process involves several key steps: **Initialisation:** The training commences with the initialisation of the network's *parameters*, (also known as *weights*), which are typically set to small random values.

Forward Propagation: In each training iteration, a batch of input samples is fed into the network. The network processes these inputs layer by layer. Each layer consists of *nodes* (or *neurons*) that apply linear transformations followed by non-linear *activation* functions, resulting in an output.

Loss Calculation: The output of the network is then compared to the labels for that batch of data using a loss function, which quantifies the error between the model's predictions and the true labels.

Backpropagation: The gradient of the loss function with respect to each parameter in the network is calculated to determine how each parameter should be adjusted to minimise the loss.

Parameter Update: The network's parameters are then updated, using optimisation algorithms like stochastic gradient descent (SGD) or its variants, in a direction that minimises the loss.

Iteration and Convergence: The above steps are repeated for multiple iterations, often across all examples in the training dataset (an epoch), until the performance of the network stabilises or meets some predetermined criterion.

Regularisation: Techniques like dropout (when some parameters are randomly zeroed out) or L2 regularisation (when a penalty is added to the loss function to penalise very large parameters) are employed to prevent overfitting.

Validation: The network's performance is periodically evaluated on a validation dataset to monitor its ability to generalise during the training process.

One important objective in training a neural network is to minimise both overfitting and underfitting to the training data, in order to ensure the model's ability to generalise well to new, unseen data. Underfitting is when the model fails to learn meaningful patterns, and therefore has no predictive power even on the training set. This can occur when the model is too simple or not trained for long enough. For example, a linear model trying to fit a complex, nonlinear pattern in the data would likely underfit.

Overfitting is when the model's parameters are too highly optimised for the specific patterns in the training data, in such a way that they do not generalise to unseen samples. This often happens when the model is excessively complex relative to the training data (e.g. having too many layers or neurons in a neural network), or is trained for too many iterations. An overfit model may achieve very high accuracy on the training set, but perform poorly on the validation set or in deployment.

Therefore, the goal is to train a model such that it learns only relevant, generalisable patterns from the training data. Several techniques can help mitigate overfitting and underfitting:

Adjusting model complexity: Choosing an appropriate model architecture with the right level of complexity for the task and dataset size. This may involve experimenting with different numbers of layers, units, etc.

Early stopping: Monitoring the model's performance on a validation set during training and stopping the training process when the validation performance starts to degrade, even if training performance continues improving. This prevents the model from starting to memorise noise in the training data.

Regularisation: Adding penalty terms to the model's loss function that discourage large parameter values and thus limit the model's complexity. Common regularisation techniques include L1 and L2 regularisation, dropout, and weight decay.

Data augmentation: Either gathering more data, or artificially increasing the size and diversity of the training set by applying random transformations (e.g. rotations, flips, crops for images), making it harder for the model to overfit.

By carefully tuning model architecture, training procedure and regularisation, and leveraging techniques like early stopping and data augmentation, neural networks can be trained to achieve good generalisation performance.

Hyperparameter Selection

The training of a neural network involves several hyperparameters and architectural choices that can significantly influence the network's performance and generalisation ability. Key decisions include:

Learning Rate: The learning rate determines the size of the steps taken by the optimisation algorithm when adjusting the network's parameters in response to the calculated gradients. A higher learning rate means larger steps, which can lead to faster convergence but may also cause the optimiser to overshoot the optimal solution. A lower learning rate means smaller steps, which can lead to slower convergence but may allow the optimiser to find a better solution. The ideal learning rate strikes a balance between convergence speed and solution quality.

Batch Size: The batch size is the number of training examples used in one iteration of the training process. A larger batch size allows for more stable gradient estimates and can lead to faster computation due to parallelisation. However, it also requires more memory. A smaller batch size can introduce more noise into the gradient estimates but may allow the optimiser to escape from local minima. The choice of batch size often depends on the available computational resources and the specific problem.

Number of Epochs: An epoch is a complete pass through the entire training dataset. The number of epochs determines how many times the network will see the entire dataset during training. More epochs can lead to better performance but also increase the risk of overfitting. The optimal number of epochs depends on the complexity of the problem and the size of the dataset.

Network Architecture: The architecture of the network, including the number of layers, the number of nodes in each layer, and the type of layers (e.g., convolutional, recurrent, etc.). Deeper and wider networks can learn more complex features but are also more prone to overfitting. The choice of architecture often depends on the specific problem and the available computational resources – some widely used architectures are discussed later in this section.

Activation Functions: The choice of activation functions, which introduce non-linearity into the network, can also affect the network's performance. Common choices include sigmoid, tanh, ReLU, and their variants. ReLU has become popular due to its simplicity and ability to mitigate the vanishing gradient problem (in which the gradients become too small in later layers of deep networks).

Regularisation Parameters: Regularisation techniques, such as L1 and L2 regularisation, add penalty terms to the loss function to discourage large parameter values. The strength of these penalties is controlled by regularisation parameters. Higher values of these parameters lead to more regularisation and can help prevent overfitting, but if too high, they can lead to underfitting.

Optimiser: The choice of the optimisation algorithm, such as SGD, Adam, RMSprop, etc., and their associated hyperparameters (e.g., momentum for SGD), can affect the convergence speed and the quality of the found solution.

The optimal values for these hyperparameters are problem-specific and are often found through a process of hyperparameter tuning, which can involve techniques like grid search, random search, or more advanced methods like Bayesian optimisation. The interplay between these hyperparameters is complex, and it's often hard to know the optimal configuration ahead of time.

Neural Network Architectures

It is also possible to use many different *kinds* of neural networks, depending on the task at hand – and we have seen rapid evolution in terms of different model architectures tailored for specific types of tasks. Some key architectures include:

Recurrent Neural Networks (RNNs): RNNs are designed for sequential data tasks, making them suitable for time series analysis, natural language processing, and speech recognition.

Unlike traditional neural networks, RNNs have loops to allow information persistence, meaning they have a memory of previous inputs in their internal structure [123].

Autoencoders: these are unsupervised neural networks used for data compression and noise reduction. They work by compressing the input into a compact internal representation and then reconstructing the output from this representation [37].

Deep Belief Networks (DBNs): DBNs are generative models that consist of multiple layers of stochastic, latent variables. They stack multiple Restricted Boltzmann Machines (RBMs) to create a deep architecture, making them suitable for tasks like feature reduction and generative tasks [37].

Generative Adversarial Networks (GANs): This architecture involves *two* neural networks, namely the *generator* and the *discriminator*, which are trained together. The generator tries to produce data, while the discriminator tries to distinguish between real and generated data. This setup is primarily used for generative tasks, such as creating new images that resemble a set of training images [60].

Transformers: Transformer models have recently emerged as a dominant architecture for tasks related to natural language processing (NLP). The transformer architecture leverages self-attention mechanisms to weight different regions of the input, allowing for more flexible and context-aware representations of data. This architecture has led to state-of-the-art results in a variety of NLP tasks, including machine translation, text generation, and sentiment analysis, and there has been increasing interest in vision applications too. [144].

Residual Neural Networks (ResNets)

ResNets introduce the concept of "skip connections" or "shortcuts" that allow the activation from one layer to bypass one or more layers and be summed up with the activation of a later layer. This architecture was developed to address the vanishing gradient problem, allowing for the training of much deeper networks. This approach has shown significant improvements in training deep networks, leading to better performance in various tasks, especially in image classification [166].

Convolutional Neural Networks (CNNs)

CNNs are primarily used for tasks related to image processing, such as image classification, object detection, and facial recognition. The architecture is characterised by its convolutional layers that automatically and adaptively learn spatial hierarchies of features from input images [15].

This thesis is primarily concerned with image-based tasks, and so of the above architectures, convolutional and residual networks are the most relevant. Although there has been increasing interest of late in the application of transformers to vision tasks [40]), CNNs are still the most widely used for a number of reasons. They inherently possess inductive biases, such as translational equivariance and locality. These biases enable them to process spatial hierarchies in images efficiently [141]. Vision Transformers (ViTs) often necessitate substantial amounts of training data to achieve competitive outcomes, but CNNs can frequently yield satisfactory results with relatively smaller datasets [127]. From a computational standpoint, CNNs offer greater efficiency for image data, attributed to their weight-sharing mechanism and local connectivity – in contrast to ViTs, where the self-attention mechanism exhibits quadratic complexity concerning input length [61]. For this reason, CNNs remain widely used in digital pathology.

The Application of Deep Learning to Digital Pathology

Pathologists are experiencing an escalating workload, increasingly complex tasks (due to developments in testing), and a vacancy rate that has reached 10-12% in the United Kingdom, and continues to climb. Since 2007, there has been an average annual increase of approximately 4.5% in histopathology requests to labs [153]. Considering this alongside the advent of deep learning and the availability of digital data, the growing interest in applying AI to automating parts of digital pathology is unsurprising [159].

Current applications of deep learning to digital pathology include various combinations of image segmentation, tumour detection and classification (often by subtype), and cell counting and detection, among others [92, 118, 33, 67, 12, 56, 91, 169, 71, 66, 28, 130, 174, 25, 168]. As described in the previous section, convolutional neural networks (CNNs) are the most widely used architecture for these kinds of image tasks [86] – for example, Hameed et al. [62] trained a CNN to classify malignant and benign tissue from breast histology images, achieving an accuracy of 98%.

As work in this field progresses, we are beginning to see deep learning systems outperform human pathologists at predictive tasks where the ground truth is available after the fact [66, 28] – tasks like predicting patient mortality or response to treatment, the presence of genetic disease prior to genetic testing, the presence of malignancy prior to biopsy, et cetera. We are also beginning to see deep learning systems outperform the predictions of generalist pathologists, when the predictions of specialist pathologists in that particular area are available: for example, Nagpal et al. built a deep learning system for Gleason grading of prostate biopsies that agreed far more often with specialists in urologic pathology than it did with a cohort of general pathologists [107].

However, using these advances to benefit patients in practice is problematic due to concerns about the generalisation ability and interpretability of these models, along with ethical and legal issues surrounding the use of these systems in diagnostic decision-making. As such, robust interpretability is critical to making deep learning systems in digital pathology practically useful. Additionally, interpretability is important for ensuring that the model does not make predictions based on spurious or biased features, and that it will generalise to new patients and populations. If we are to use these powerful systems to inform life-and-death decisions, we must understand *what* they have learned and *why* any given prediction is made.

1.3 What Is Interpretability?

The field of machine learning interpretability research is relatively recent and, as such, is yet somewhat ill-defined: the terms 'interpretability' and 'explainability' are typically used interchangeably, and there is a lack of consensus regarding formal definitions of either [106, 84, 97], although there have been a few attempts to taxonomise the field [55, 84, 21].

It is also important to recognise that explanations can take many forms, and not all of them are equally meaningful or helpful to human users. For instance, a low-level, neuron-by-neuron tracing of a network's activations could be considered a form of explanation, but it may not provide the kind of insight that is actionable or understandable for most people. The usefulness of an explanation depends heavily on the specific context, the intended audience, and the purpose for which the explanation is being sought. Ultimately, what counts as a good explanation is largely determined by the needs and background knowledge of the user. An explanation that is illuminating for a machine learning researcher might be completely opaque to a non-technical stakeholder. The challenge is to provide explanations that are not only technically accurate but also *useful*.

The complexity of this issue is highlighted by the growing body of work examining the notion of interpretability and explainability from various angles. Some researchers critically examine the very concept of interpretability, arguing that it is often poorly defined and may not always be achievable or desirable [75]. Others provide overviews of different definitions, methods, and applications of interpretable machine learning, underscoring the diversity of approaches and the lack of standardization in the field [106]. The concept of interpretability is also sometimes invoked in vague and inconsistent ways [85], further complicating the discourse.

For the purpose of this work, we can think about interpretability methods as tools for answering questions about model behaviour – just as we might use the tools of biology to answer questions about the workings of the human body – and the project of interpretability research is then to design reliable, accurate tools for this task: the microscopes of machine learning. Selecting which tool to use – that is, which of the many contemporary interpretability methods are best suited to understand a given model – is primarily determined by the architecture of that model, the nature of the data it is trained on, and the task it is trained to perform.

Intrinsically Interpretable Models

Some simpler models are considered intrinsically interpretable [131]. One such model is the *decision tree*. Decision trees split the data based on feature values. Each node in the tree represents a feature in the dataset, and each branch represents a decision rule. However, they can easily overfit to the training data, especially when the tree is deep, necessitating pruning techniques [104]. Another model is *linear regression*. It models the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. The primary limitation of linear regression is its assumption of a linear relationship between variables, and it is also sensitive to outliers. *Logistic regression* is used for binary classification problems. It estimates the probability that a given instance belongs to a particular category. Like linear regression, it assumes a linear decision boundary, which might not be the case for all datasets [2]. Lastly, the *k-Nearest Neighbors (k-NN)* model classifies a data point based on how its neighbors are classified. It looks at the 'k' nearest data points and assigns a label based on the majority class among them. However, it is computationally expensive for large datasets and is sensitive to irrelevant features and the choice of distance metric [104].

All of these models share some overarching limitations: firstly, they often make strong assumptions about the underlying data distribution. For instance, both linear and logistic regression assume a linear relationship between the features and the target variable [27]. While this assumption simplifies the model and enhances interpretability, it can lead to sub-optimal performance when the true relationship is nonlinear. Secondly, these models can be sensitive to outliers and noise in the data. Anomalies in the dataset can disproportionately affect the performance of models like linear regression and k-NN. For instance, a few mislabeled examples can significantly alter the decision boundary in k-NN [104]. Another shared limitation is the potential for overfitting, especially when the models are not regularised or constrained. Decision trees, in particular, are notorious for creating overly complex trees that perfectly fit the training data but perform poorly on unseen data [104].

While intrinsic interpretability is a useful property, in real-world applications these models often lack the expressive power needed to capture intricate patterns in the data, leading to far inferior predictive performance compared to the deep learning models described in Section 1.2. For this reason, interpretability techniques that enable humans to understand powerful models that are *not* considered intrinsically interpretable are particularly useful [2].

1.4 Global and Local Interpretability

Interpretability methods can be either global (what has the model learned from the dataset?) or local (why has the model produced a specific output for a specific sample?) [102, 133]. *Global* methods aim to explain how a model makes predictions holistically, identifying which features are important across the whole dataset and how model outputs are distributed based on the data, learned parameters and model architecture. For complex models, this is hard to achieve in a way that is interpretable by humans, so in practice, most global methods focus only on some parts of the model – such as the learned weights in a convolutional layer – to explain which features the model has learned to identify during training.

Feature visualisation [43] is an example of this kind of global interpretability method, in which a single input sample is optimised to maximise the activation of a particular node, layer, or output. This optimised input can then be inspected to identify the kinds of features a particular node or layer in a network has learned to identify.

Local methods, by contrast, are concerned only with individual data samples and aim to explain why the model produces the output that it does *given* that particular input [101]. These methods are typically much easier for humans to parse as they answer questions which can be easily visually interpreted – 'Which pixels of this image influenced the prediction most?'. This kind of local explanation is also known as *attribution* [175] – that is, the identification of the portions of a given input to which the model's output can be attributed.

Both global and local interpretability methods fall into one of two broad categories – those which are *model-specific* and depend on access to the structure and internal state of the trained model, and those which are *model-agnostic*, and only require access to the input and output of the model [103]. For example, gradient-based attribution is a local, model-specific method, as it uses gradient information (which is model-specific) to explain a single (local) prediction.

Model-specific local methods are typically much more efficient, as they work by inspecting the internal state of the model and, as such, do not require iteration over different permutations of the input – but, they can only be used when the trained architecture of the model in question is both accessible, and of a suitable type, which limits their application [101]. They cannot be used at all for models in which the internal state and structure are unknown or inaccessible, such as when accessing a third-party model through an API.

In contrast, model-agnostic local methods work by iteratively perturbing different regions of the input and inspecting the change in the model output relative to each perturbation, irrespective of the internal state of the model – and so can be used for *any type of model at all*, including ensemble methods which could combine both white- and black-box models. However, existing techniques of this kind are slow, as they build up an empirical estimation of regional importance through many iterations – and require several hyperparameters to be specified – the optimal values of which are difficult to know ahead of time for new datasets, necessitating computationally expensive heuristic tuning [170].

Some interpretability methods, such as feature visualisation, can be applied in both model-specific and model-agnostic contexts depending on whether they are applied to output logits (un-normalised raw model outputs) or internal activations.

1.5 Model-Agnosticism

There are several applications of artificial intelligence in which interpretability is very important, but where the model architecture is inaccessible or unsuitable for the application of gradient- and activation-based model-specific explanatory methods, or those methods would result in saliency maps too coarse for the task at hand. For example, healthcare triage using ensemble architectures taking both clinical data and high-resolution medical imaging; very high-resolution image inputs in general; legal tasks using very large textual or tabular datasets, or autonomous vehicle decision-making combining video, sensor and time-series input, to name a few [95, 101]. By definition, model-agnostic interpretability methods can be applied to *any* model, and so research in this area is particularly valuable as the resulting algorithms are not limited to a single type of data or architecture.

More practically, in the course of designing a machine learning solution to an applied problem, testing a number of different types of model and model architecture is almost always necessary. Using model-specific interpretability methods can quickly become a burden, as each model or architecture will require different implementations and likely separate hyperparameter tuning. Adopting a model-agnostic approach to interpretability significantly reduces engineering overhead, speeding up development and drastically lowering the barrier to including interpretability techniques in day-to-day applied AI work. Moreover, using the *same* interpretability method for all different types of models tested allows for easy, objective, and often illuminating comparison of learnt features and salient regions between models. Being able to tell if two equally performing models have learned the same features or if they find the same input regions salient is valuable for model selection. If model A has learned a feature that humans find meaningful, and model B has learned some other, less obviously sensible feature, it is helpful to know that prior to deployment.

At the time of writing, state-of-the-art transformer models can have hundreds of billions of parameters [142]. Scaling laws predict that performance is likely to continue increasing as the number of parameters does [72]. Interestingly, model performance is far more strongly associated with scale than shape (i.e. depth or width), even as model architectures for all data types have evolved over time [152]. Nonetheless, state-of-the-art model architectures *do* keep evolving, and so another key benefit of model-agnostic approaches is their flexibility and robustness to this evolution – they remain valuable tools for objective and comparative interpretation even as the underlying architecture may vary wildly and continue increasing in size as AI research progresses. However good a model-specific interpretability method might be, it can never be future-proof in the same way.

Finally, unlike model-specific methods, model-agnostic interpretability methods can be directly and objectively evaluated and compared using the Proxy Model Test, which I propose in Chapter 5.

Chapter 2

Feature Visualisation

One intuitive way to understand what a model has learned is by using feature visualisation [100, 105, 43, 117, 112] (sometimes also called activation maximisation, input optimisation, or similar). This method generates an input to the model that maximises a selected output logit or internal activation. This input can be initiated as all zeros, as the mean of the training data, as some input sample from the dataset, as noise, or by some other method; this prior can have a substantial effect on the result [117]. Stochastic Gradient Descent (SGD) or a variant is then used to iteratively adjust this input, to maximise the output in question. In order to produce a range of diverse inputs that maximally activate the output in question, a diversity term can be used which penalises input examples similar to those already generated [117].

Regularisation of one form or another is often used to constrain the input to the dataset distribution and impose a more natural structure upon it, to avoid generating inputs which artificially maximise the output by exploiting out-of-distribution behaviour, and to limit high frequency patterns related to the network architecture, rather than to the learned features with which we are concerned. Limiting this kind of noise in order to more clearly visualise what the model responds to, whilst limiting the introduction of human biases is a key concern in feature visualisation research. This problem is similar in spirit to the accuracy-interpretability trade-off discussed in Chapter 3.4: we want to find a balance between *accurately* visualising what truly maximises some output for a given model, and doing so in a way that is *interpretable* by humans.

Three main kinds of regularisation have been used in this pursuit: *frequency penalisation*, *transformation robustness*, and *learned priors*. Frequency penalisation is achieved by either explicitly penalising variance between adjacent elements [93] or by simply blurring the input prior to each SGD step [112]. Transformation robustness is achieved via the application of various stochastic transformations to the input during the optimisation procedure, typically including scaling, small translational offsets and rotation. While simple, these approaches to regularisation have produced convincing and distinct visualisations [105].

Using learned priors is a different approach altogether, which works by attempting to learn a model of the real dataset and enforcing that model upon the input during optimisation. This can be enacted by training a generative model, such as a Generative Adversarial Network (GAN) or a Variational Auto-Encoder (VAE), to learn a mapping from the data to some latent space and then constraining the input to that latent space during optimisation [111]. Inputs generated in this way typically *look* very realistic and convincing to humans, but can no longer claim to represent what truly maximally activates a given output, as the optimised input is now reliant to an unknown degree on the latent space and the model that learned it.

Feature visualisation can be used in both model-specific and model-agnostic contexts. In the former, we designate an internal node, channel or layer output for which we wish to generate an input, while in the latter, we select an output logit. Note that the raw *logit* is maximised, rather than the softmax class probability – the most direct way to increase the softmax probability for a given class is to make the other classes less likely, which typically results in very noisy and indistinct feature visualisations [117].

2.1 Exploring Learned Prototypes

I propose that in the context of feature visualisation for image classification models, when optimising an input for a particular output class, we can understand the optimised input as the model's *prototypical concept* for the output class in question. We can see *what* the model has learned to respond to from its training on the dataset – what the model 'thinks' the most predictive features of some class *are*. This can be useful and diagnostic – we might find interesting (or spurious) correlations. Perhaps the optimised input for 'lawnmower' is mostly grass, or for 'risk-of-coronary-event', the presence of a pacemaker.

It is hard to objectively assess the trustworthiness of learned prototypes (that is, how well they reflect what a model has truly learned), as they are inherently dependent on heuristically chosen hyperparameters and regularisation efforts. Minor changes to the input initialisation, learned prior, regularisation protocol or gradient descent algorithm can significantly affect the optimised image. How 'accurate' a feature visualisation is, and how close the hyperparameters are to optimal – that is, how close the optimised input is to the *true* maximally activating input – is difficult to quantify in practice. However, one thing we *can* do is hold those hyperparameters fixed and use feature visualisation in a comparative manner across different outputs of a single model, or a set of different models. If multiple inputs are generated with a range of different feature visualisation algorithms, and they all contain similar features, this suggests that those visualised features are approaching the true learned prototype for that class. This is explored in practice in Chapter 7.

In this chapter, I will explore some of these ideas using a few simple neural networks trained on the MNIST dataset. The following class prototypes were generated using a feature visualisation algorithm proposed by Erhan et al. [43], to generate one prototype $\hat{\mathbf{X}}_c$ for a model h for each output class c, such that

$$\hat{\mathbf{X}}_c = \arg\max_{\mathbf{X}} h_c\left(\frac{\mathbf{X} - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X})}\right). \tag{2.1}$$

This is done by performing gradient ascent on $\mathbf{X}$ with respect to $h_c(\mathbf{X})$, i.e. computing the gradient of $h_c(\mathbf{X})$ and moving $\mathbf{X}$ in the direction of this gradient. In this case I initialised $\mathbf{X}$ as a $1 \times 28 \times 28$ zero matrix (the size of each MNIST image the model was trained on), and at each iteration normalised $\mathbf{X}$ to constrain it within the bounds of the MNIST dataset.

Architectural Artefacts

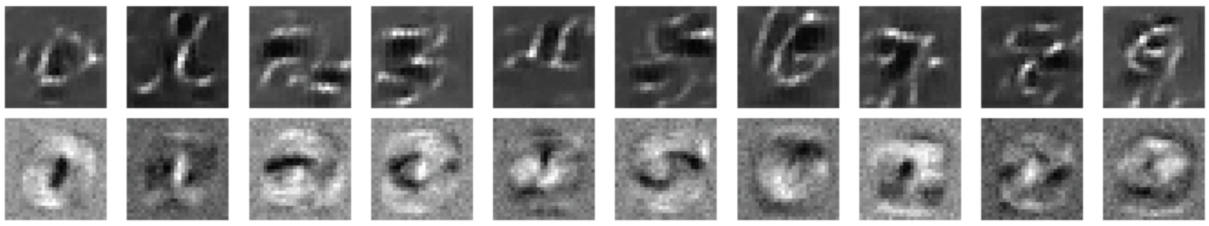


Figure 2.1: Feature visualisation for two classification models trained to 97% accuracy on MNIST data: a simple convolutional model (above) and a completely linear model (below).

Here I will briefly revisit the high frequency noise mentioned in Section 2. What does it mean, and where does it come from? Olah et al. [117] describe it as high frequency patterns that result from strided convolutions and pooling operations – what I shall call *architectural artefacts*.

I performed an experiment to see if this was the case, in which I trained two different models to classify images from the MNIST dataset. The results of this experiment are shown in Figure 2.1. Both models were simple neural networks with two layers. The first model's layers were convolutional and had 16 and 32 channels respectively (both with a kernel of size 3×3), with each followed by max-pooling, while the second's layers were entirely linear (each with 1024 nodes). Rectified Linear Units (ReLUs) were used after all layers except the last, as defined in Equation 2.2. Both were trained using SGD with a learning rate of 0.001 and momentum of 0.9, using cross entropy loss as the criterion, until 97% classification accuracy was reached.

$$\text{ReLU}(\mathbf{X}) = \max(0, \mathbf{X}) \tag{2.2}$$

I used stochastic gradient descent (SGD) for these experiments with a learning rate of 0.1. The only regularisation used was normalisation of the input prior to each optimisation step as shown in Equation 2.1, to constrain the input to $\mathbf{X} \in [0 \ldots 1]$ as per the MNIST training data.

The optimised input images for each model are quite different. While both clearly show relevant features which are present in the training data, *what* each model has learned is distinct due to the difference in architecture. Notably, the digits visible in the optimised inputs for the model with only linear layers are centered are thickly outlined and centered in the input

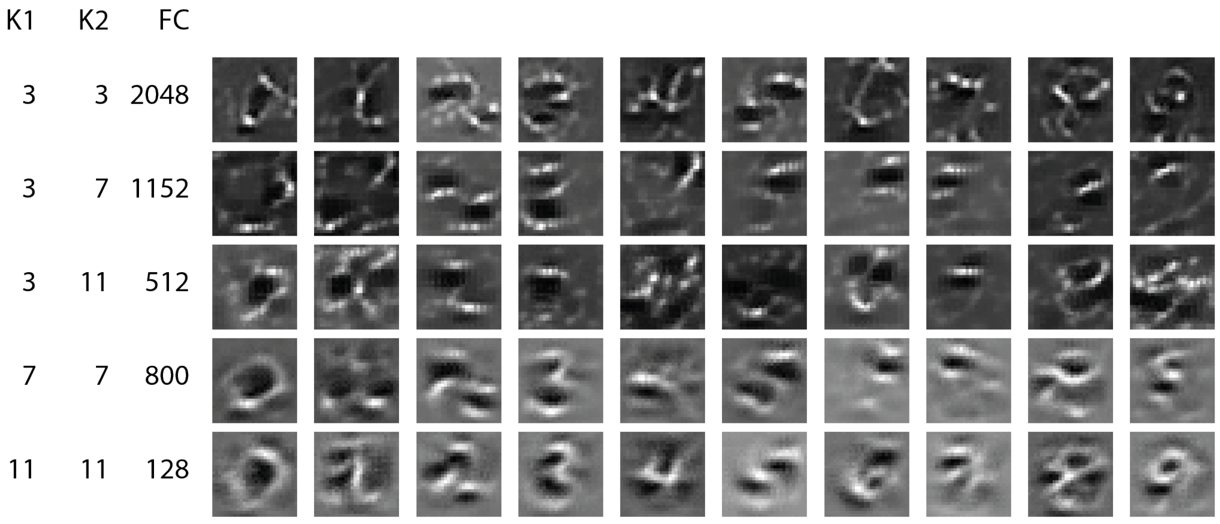


Figure 2.2: Feature visualisation for five classification models with slightly different architectures trained to 97% accuracy on MNIST data. K1: Kernel dimension ($n \times n$) of first convolutional layer, K2: Kernel dimension ($n \times n$) of first convolutional layer, FC: Number of nodes in the fully connected linear layer.

region, while the convolutional model's are thinner, and are frequently doubled and offset. The linear model pays little attention to the outer pixels – we can see that they are noisy, and close to the mean – while the convolutional model has learned high contrast features and expects a dark background. This is as expected: convolutional layers in neural networks identify features location-independently, while linear layers cannot. What can we take away from this experiment? It seems like chequerboard artefacts are indeed architectural – they are visible in the convolutional model's optimised inputs, but not in the linear – but that is not the most interesting thing about this little experiment. Firstly, it shows that it is possible for us to access information about a model's architecture, given access only to its inputs and outputs. Secondly, it shows that different models trained on the same data, performing equally well, may have learnt quite different features. This underlines the importance of feature visualisation specifically and the use of interpretability methods in general. In Chapter 3.4, I argue that we cannot assume that a deep learning model has learned to perform well by using the same features that a human would. Here I show that we cannot assume that two models, alike in performance, have learnt the same features.

Other information about the model is accessible, too: in Figure 2.2, I show the optimised inputs for five more convolutional models with slightly different architectures. The only difference

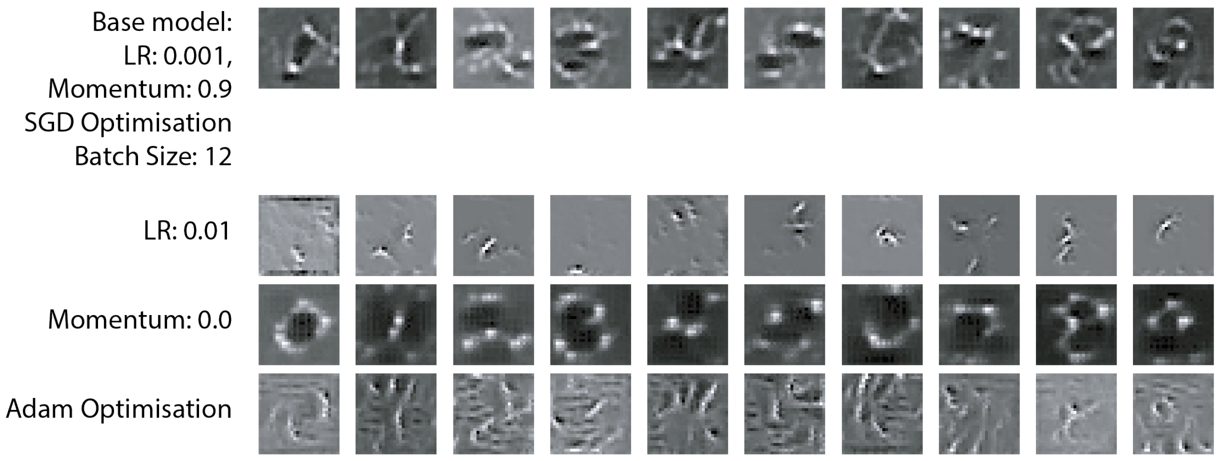


Figure 2.3: Feature visualisation for five classification models trained with different protocols, all trained to 97% accuracy on MNIST data.

between them and the first convolutional network is the kernel size used at each convolutional layer, and the number of nodes in the final layer (necessarily adjusted to accommodate the new kernel sizes). Small kernel sizes lead to more fragmented visualisations, and larger kernel sizes to smoother ones (this is particularly apparent due to the shallow nature of this toy model), which makes sense as larger kernels will learn to identify larger features than small ones during training.

The optimised inputs for the models with mixed filter sizes ((K1=3, K2 = 7) and (K1 = 3, K2 = 11)) are more difficult to interpret. They appear more diffuse, disjointed and noisy: it seems that following 3×3 kernel with one of a larger size is not ideal for this dataset. Looking inside the black box to visualise the trained model's learned kernels supports this, with increased noise and lower inter-filter variation evident in ((K1=3, K2=7) and (K1=3, K2=11)), as shown in Figure 2.4.

It is clear that the choice of kernel size has a significant effect on the learned prototype, as would be expected. However, it is not just architectural choices that change what the model learns: *how* the model learns also plays a large part. For example, in Figure 2.3, I compare three further models, all with *exactly* the same architecture, but each trained in a slightly different way. The first was trained with a higher learning rate, the second without momentum, and the third using Adam optimisation instead of vanilla SGD. The differences here are striking.

We can conclude that feature visualisation can be a powerful tool for use during model

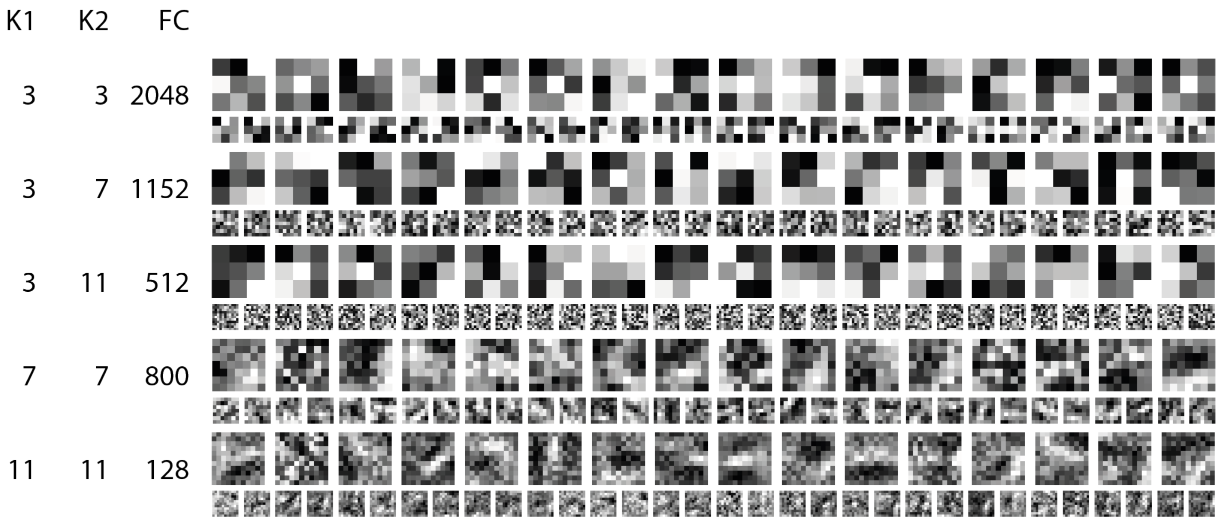


Figure 2.4: Learned convolutional filter kernels for five classification models trained with slightly different architectures trained to 97% accuracy on MNIST data. K1: Kernel dimension ($n \times n$) of first convolutional layer, K2: Kernel dimension ($n \times n$) of first convolutional layer, FC: Number of nodes in the fully connected linear layer.

design, not just for post-hoc interpretability. Optimising for clean, distinct class prototypes that are *aligned with human knowledge* during training could go a long way towards producing robust and interpretable AI systems.

2.2 Centroid Maximisation

Research in feature visualisation to date has focused almost exclusively on image classifiers [137]. However, in Chapter 7, I adapt feature visualisation for use with a semantic *segmentation* model. Segmentation models are trained to classify each *pixel* of an input image according to which class (or object, et cetera) it belongs to – thereby partitioning an image into multiple segments, where each segment corresponds to a specific class or object [65], as shown in Figure 2.5. This means that for each input image, in contrast to image classifiers, segmentation models do not just produce a probability for each class – but *for each pixel of that image*, for each class. They therefore typically output a matrix of size $d \times d \times C$, where d is the input image dimension (in this case assuming it's the same for width and height) and C is the number of classes. This presents a new challenge if we wish to understand what a segmentation model has learned in

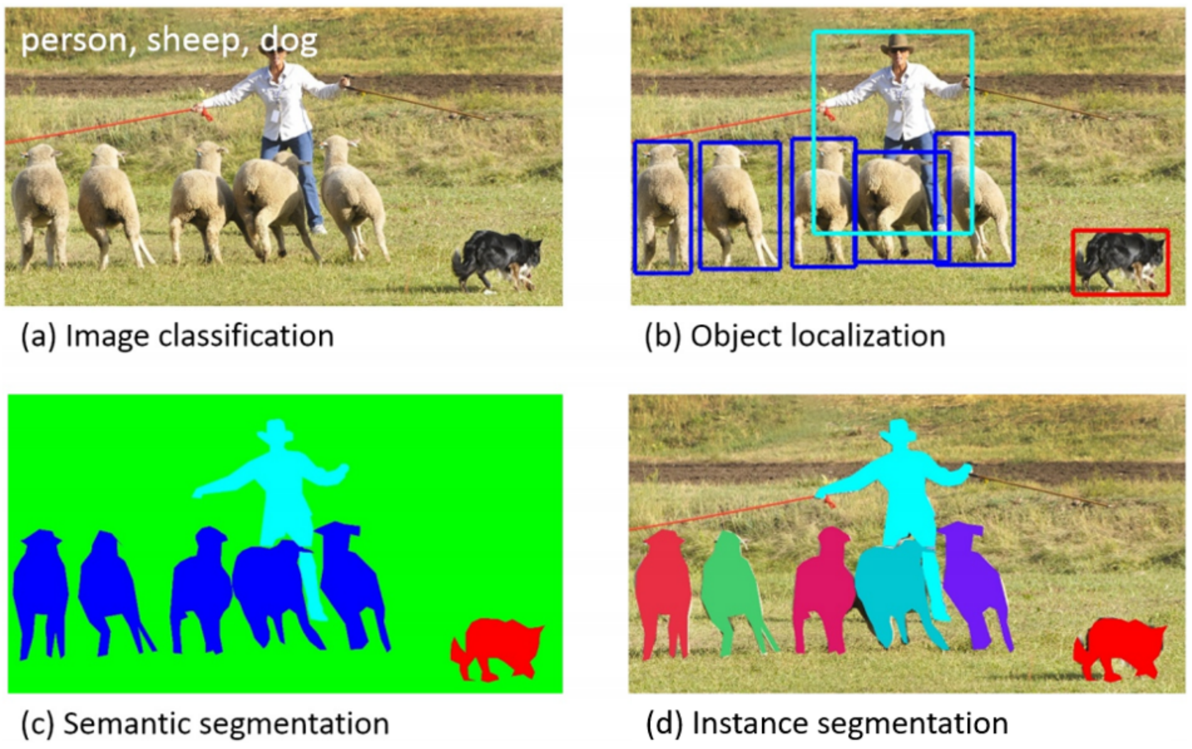


Figure 2.5: Example of different tasks in computer vision from Lin et al. [83]. Consider that the output format of the model varies significantly between the two tasks we are concerned with: in image classification (a), the model predicts the probability of each class being present in the image *as a whole*; whereas for semantic segmentation (c), the model predicts the probability that *each pixel* belongs to each class.

terms of *class prototypes*.

A naive approach would be to use the method defined in Section 2.1, and simply optimise the input image **X** to maximise the model's output for class c (denoted h_c) for *every pixel* (denoted $h_{i,j,c}$ for the model's output for class c for the pixel at location i, j) such:

$$\hat{\mathbf{X}}_c = \arg\max_{\mathbf{X}} \sum_{i=0}^{d-1} \sum_{j=0}^{d-1} h_{i,j,c}\left(\frac{\mathbf{X} - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X})}\right). \tag{2.3}$$

In practice (at least on MNIST), this does yield patterns which are identifiable as being relevant to the class in question (see Figure 2.6). However, these patterns are quite dissimilar to the distinct prototypical concepts we are able to uncover in classification models.

Here I will present a more sophisticated prototype generation method for visualising features learned by segmentation models – that of *centroid maximisation*. Instead of optimising an input

Precision	Recall	F1	Accuracy
0.6618	0.6563	0.6556	0.9941

Table 2.1: Segmentation model performance on MNIST test set. Metric formulas can be found in Equation 7.5.

X to maximise the models prediction for class c for all output pixels as above, I select just the centre pixel of the output (denoted $h_{\frac{d}{2},\frac{d}{2},c}(\mathbf{X})$) and optimise the input to maximise *only it*, leaving the surrounding pixels to change through gradient ascent as necessary without penalisation – essentially allowing the class prototype $\hat{\mathbf{X}}_c$ to 'grow' around this centroid:

$$\hat{\mathbf{X}}_c = \arg\max_{\mathbf{X}} h_{\frac{d}{2},\frac{d}{2},c}\left(\frac{\mathbf{X} - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X})}\right). \tag{2.4}$$

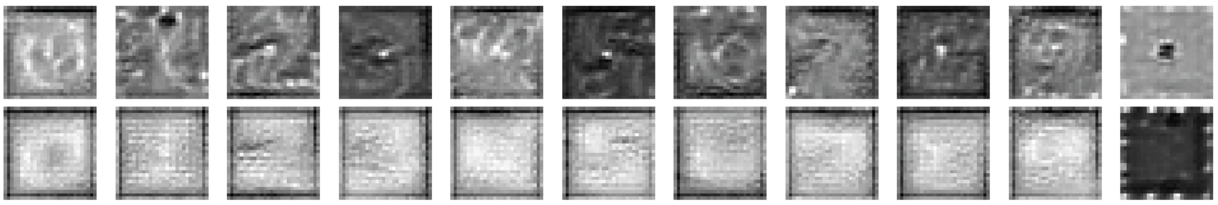


Figure 2.6: Feature visualisation for two segmentation models trained on MNIST data. The first (above) using centroid-only optimisation, and the second (below) optimising for high class predictions on all outputs.

To test this idea, I built another convolutional model and adapted it for the segmentation task by changing the number of nodes in the final linear layer to equal $(C+1) \times d \times d$. The output of this linear layer is then reshaped. As per the base convolutional model in the previous experiments, both convolutional layers used a 3×3 kernel and had 16 and 32 channels respectively. I trained this model for ten epochs using cross entropy loss, which resulted in test set performance as shown in Table 2.1:

This practice produces far more distinct feature visualisations, as shown on MNIST in Figure 2.6. Greater high-frequency noise is evident when all pixel outputs are maximised per class – this is inevitable as SGD attempts to maximise all outputs concurrently. In Chapter 7.5, I use this kind of feature visualisation via centroid maximisation to identify immune cell prototypes learned from a real-world histopathology dataset.

Chapter 3

Saliency Mapping and The Ground Truth Problem

A popular approach to interpretability, particularly for computer vision tasks, is saliency mapping (sometimes also called ‘attribution’) – that is, the generation of a heat map assigning colour or brightness to regions or elements of the input according to how much each region or element contributed to the model’s output. Saliency mapping of this kind is widely used in machine learning, particularly for image classification tasks, but also for language modelling [44], prediction from tabular data [164], and other tasks including time series prediction [122] that require the use of large and complex neural networks, which are troublesome to interpret otherwise. Saliency maps are intuitive to interpret, and are typically used to validate that models are learning to use sensible features to make predictions, to debug mistakes, and to identify biases and spurious correlations, which is not only crucial for safety as AI is increasingly applied to high stakes domains such as medicine and autonomous vehicles, but also a valuable tool for increasing trust in and adoption of these powerful technologies.

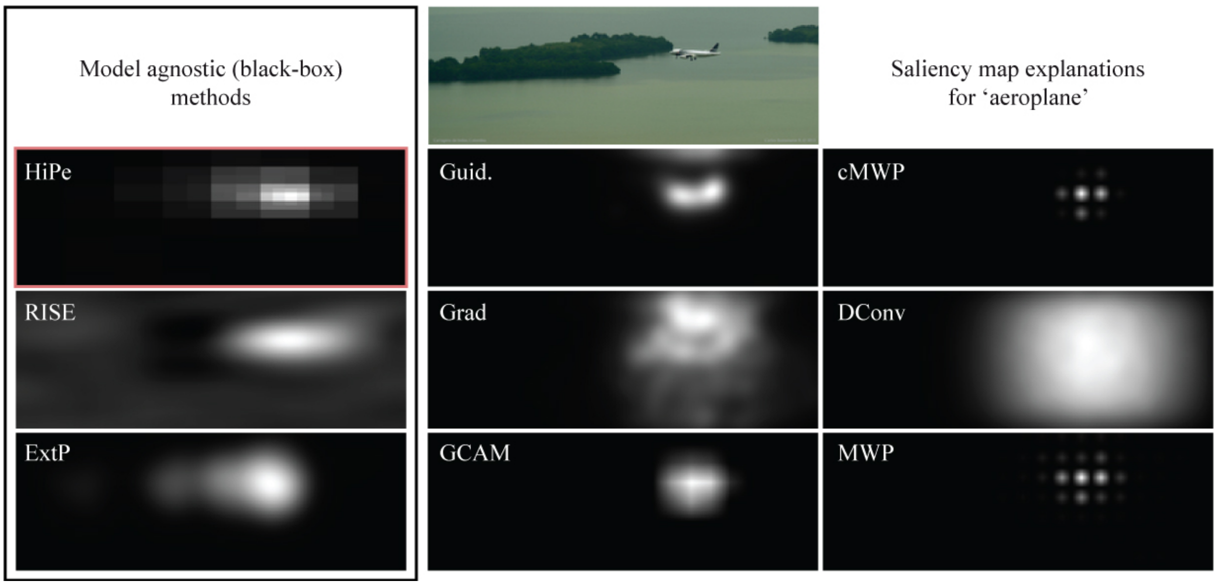

Model agnostic (black-box) methods
Saliency map explanations for 'aeroplane'
HiPe
RISE
ExtP
Guid.
Grad
GCAM
cMWP
DConv
MWP

Figure 3.1: Examples of saliency maps generated by: Hierarchical Perturbation (HiPe, proposed herein); Random Input Sampling for Explanation (RISE) [124]; Extremal Perturbation (ExtP) [46]; Guided Backpropagation (Guid.) [147]; Gradient (Grad) [143]; Grad-CAM (GCAM) [138]; Contrastive Excitation Backpropagation (cMWP) [172]; Deconvnet (DConv) [171]; and Excitation Backpropagation (MWP) [172].

3.1 Model-Specific Saliency Mapping Methods

Model-specific methods typically leverage network architecture to visualise explanations by using gradients, activations, or some combination of the two. They are efficient, and often produce convincing results, but have a few important limitations:

Gradient (Grad)

The *Gradient* method, sometimes referred to as *vanilla gradient* or simply *backpropagation*, is one of the most straightforward approaches for saliency map generation. By calculating the gradient of the model's output for a particular class with respect to the input image, it determines how small changes in input pixels would affect the prediction for that class. A high gradient value for a pixel indicates its importance in the model's decision. While this method provides a direct measure of pixel importance, it often results in extremely noisy saliency maps – this is because all possible pathways in the model that can influence the output are captured, leading to non-localised and scattered maps [143].

Guided Backpropagation (Guid.)

Guided Backpropagation is an enhancement of the standard backpropagation algorithm. In this method, only positive gradients are propagated back to the input, effectively filtering out the pixels that would decrease the model's output if they were increased. This results in saliency maps that are generally clearer and more focused than those produced by the Gradient method. However, it is prone to producing high-frequency patterns, which can make the saliency maps it produces hard to interpret and understand [147].

Grad-CAM (GCAM)

Grad-CAM stands for Gradient-weighted Class Activation Mapping. Grad-CAM uses the activations from the last convolutional layer of a CNN to determine which regions in the image were most influential for a specific output. By computing the gradients of the output class with respect to these activations, Grad-CAM produces a coarse heatmap that highlights the important regions of the image for the model's decision. This method offers spatially coherent visualisations, making it easier to identify the regions of interest in the image, but is incapable of capturing details. [138].

Excitation Backpropagation (MWP, Marginal Winning Probability)

Excitation Backpropagation propagates the 'excitation' (or, activation) of a target output backward through the network, and uses a probabilistic Winner-Take-All process to determine which neurons propagate their excitation to the preceding layer. By emphasizing only the most influential paths in the network, it produces maps that highlight distinct and isolated regions of importance – but this can also produce saliency maps that are overly sparse, with their granularity determined by the model's architecture [172].

Contrastive Excitation Backpropagation (cMWP, contrastive Marginal Winning Probability)

Contrastive Excitation Backpropagation builds upon the idea of Excitation Backpropagation, and has the may of the same properties. In this case, the *difference* in excitation between the target class and other classes is used to inform the saliency map. This results in even sparser saliency maps, which highlight only features unique to the target class [172].

Deconvnet (DConv)

Deconvnet is a visualisation technique that aims to reverse the operations of a CNN. By mapping the feature activations back to the input space, it reconstructs the patterns in the input that led to specific feature activations. This method is particularly useful for understanding what individual neurons in a CNN are looking for. However, the process of reconstruction often introduces artifacts - specifically, due to the unpooling and deconvolution steps, the resulting saliency maps often contain checkerboard patterns or other high-frequency noise, which can make them harder to interpret [171].

Sanity Checks

Adebayo et al. and others [10, 136, 113] found that visual inspection is not a reliable guide in determining whether an explanation is sensitive to the underlying model and data. The authors demonstrate that even when the parameters of a model are randomised (essentially turning the model into a non-informative random classifier), some gradient-based saliency methods (guided backprop and deconvnet, in particular) still produce visually appealing and seemingly meaningful explanations. This raises concerns about the validity of such methods, as they appear to be insensitive to the model's parameters, producing outputs that are strikingly similar to basic edge detectors. These saliency mapping methods appear to be highlighting generic features in the image, rather than features specifically relevant to the model's decision. This observation underscores the risk of confirmation bias, where human observers might mistakenly interpret these highlighted features as meaningful explanations for the model's prediction. This risk is decreased with model-agnostic methods due to their perturbation-based nature: there is an explicit mapping from a perturbed input to a change in output.

Aside from confirmation bias concerns, which are discussed further in Section 3.4, model-specific saliency-mapping methods share a number of limitations. The resolution of the saliency maps they generate is architecture dependent, irrespective of what the *actual* size of the most salient features in the input might be. Moreover, as noted by Fong et al. [47], these kinds of methods are fundamentally ungrounded in *what* makes some region of the input more or less salient – their explanatory power is assessed *a posteriori*. At present, even the most successful methods of this kind are only applicable to a limited subset of architectures and only when the trained model's internal state is accessible – see Section 1.5 for further discussion of the benefits of model-agnosticity and why we might care about the transferability of interpretability approaches to arbitrary models.

3.2 Model-Agnostic Saliency Mapping Methods

Model-agnostic saliency mapping methods work by iteratively perturbing regions of the input sample [124, 171] and using the sensitivity of the model output to these perturbations to generate a saliency map. These methods have the nice property of direct interpretability (i.e. a perturbation in the input can be directly mapped to a change in the output), but are computationally expensive due to their iterative nature – they must query the model many times to build up a saliency map for a single image. These methods also require heuristic parameter selection (for example, selecting the size of the perturbation kernel or the number of masks to generate) to produce informative visualisations, which may necessitate many trials, and thereby also prove costly and fall prey to biased tuning to generate attractive, rather than informative maps. Zeiler et al. [171] outline an early form of this approach, in which a perturbation kernel of fixed size is iteratively applied to the input, and the difference in output at each kernel location is collated to form a saliency map. This is intuitive but very time consuming, as it relies on running potentially many trials with different kernel dimensions to generate informative visualisations, since it is impossible to know the scale of the most salient features learned by the model ahead of time.

Other techniques include training a second model using a saliency criterion to generate attribution maps directly from the input sample [128, 36]. This approach is very fast once the saliency model has been trained. However, applying this approach to a new dataset and model would necessitate not only training a predictive model to succeed at the task at hand, but also training a separate second model for saliency, which may not be trivial. This effective doubling of the hyperparameter and architecture tuning burden and computational cost of training an additional interpretability model for every predictive model deployed is likely to discourage the widespread adoption of this technique. More importantly, explanations generated in this way are by their nature fundamentally divorced from the model in question, and invite biased tuning to generate saliency maps that *look* sensible to humans rather than optimising for explanatory power.

Extremal Perturbation (ExtP)

Extremal Perturbation uses gradient descent to localise the smallest region in the image, that when perturbed, causes the most significant change in the model's output. While it provides a clear visualisation of the most critical regions in the image for the model's decision, its iterative nature makes it computationally intensive [46]. More specifically, ExtP uses gradient descent to learn a perturbation mask which minimises (or conversely, maximises) the model's prediction for the target class when applied to the input. This method produces binary segmentations which look appealing but obscure any difference in feature salience *within the broadly salient region* of pre-specified size. Furthermore, because it uses gradient descent, Extremal Perturbation requires the selection of many hyperparameters (learning rate, momentum, number of iterations, mask upsampling factor, mask areas, et cetera), which are chosen empirically and may not generalise to novel models or datasets – once more raising the lengthy and uncertain prospect of manual tuning to generate informative, interpretable saliency maps. (Unlike Random Input Sampling and Hierarchical Perturbation, which do not use gradient descent and so do not require the same extent of hyperparameter tuning.) The published settings for PASCAL VOC and COCO result in excellent performance but take an extremely long time compared to other methods (over fifty seconds using an NVIDIA GeForce RTX 2060, compared to sub-second performance for most other methods).

Random Input Sampling for Explanation (RISE)

Random Input Sampling for Explanation [124] is also perturbation based, and works by generating a number of low resolution random binary masks, upsampling them using bilinear interpolation, using them to mask the input, and weighting each mask by the model's output for the correspondingly masked input (the perturbation, in this case, being the zeroing-out of random regions of the input). The weighted masks are then aggregated and normalised, producing a saliency map. The dimensions of the low resolution masks, and the number of masks used, are chosen empirically (8000 7×7 masks were used for ResNet50). The fact that the masks are randomly generated means that RISE must always use a large enough number of masks relative to the size of the input and the salient feature size in order to avoid biasing the saliency map with unevenly distributed perturbations, especially when there are several salient regions of varying sizes contained in the input. The larger the input dimension, the larger the number of masks must be; crucially, although we might be able to make reasonable guesses based on domain knowledge of the model's training data, we cannot know fur sure the number of masks and the resolution of those masks required to generate an accurate saliency visualisation. (This is because we do not know the dimensions of the features that the model has learned – hence the need for interpretability methods in the first place.) Decreasing the resolution of the initial binary mask before interpolation decreases the number of masks necessary – however, the lower resolution this mask is, the coarser the final saliency map will be, making RISE prohibitively expensive for high resolution data which demands high resolution saliency. These limitations are explicitly mentioned in the original publication [124], which calls for future work to address this by intelligently selecting a smaller number of masks.

3.3 Perturbation Substrates

A key consideration in perturbation-based saliency mapping methods and their evaluation is the fact that the choice of *perturbation substrate* (what is used to replace any perturbed regions in the model input) has a significant effect on output [30], and is very hard to generalise across different datasets for benchmarking purposes. For example, if a dataset consists of various features on a black field, as per the Hoechst stained slide data introduced in Chapter 7, perturbing input regions by replacing them with black pixels (or whatever the minimum element value of the input is after standardisation) makes sense – it essentially removes features by replacing them with the 'background' colour. However, performing the same type of perturbation on a dataset containing more natural images can be problematic, as replacing some input region with zeros in an RGB image essentially *introduces* a strange black artefact to the image, when our hope was to seamlessly *remove* information from it.

A number of different substrates have been tried, with blurred, noisy, or zero substrates used most commonly. Other methods such as *inpainting* offer the potential remove information from images while minimising out-of-distribution artefacts. During the course of my experimentation with Hierarchical Perturbation (HiPe), a novel saliency mapping algorithm which I introduce in Chapter 4, I found that using the local mean results in marginally superior performance to a zero substrate or to a Gaussian blurred substrate. This is likely because the zero substrate introduces confounding artefacts which unpredictably affect the model's output, and the Gaussian substrate may not remove sufficient information to get an accurate estimate of the blurred region saliency.

Blurred Substrate

By replacing the perturbed region with its blurred version, some original information is retained, albeit in a less distinct manner. The degree of blurring can be influenced by parameters such as the kernel size and the Gaussian function's standard deviation. A larger kernel size or a higher standard deviation results in more pronounced blurring, potentially erasing more information from the perturbed region. However, excessive blurring might introduce patterns unfamiliar to the model, leading to unpredictable outcomes.

Noisy Substrate

Introducing noise to perturbed regions is another strategy – this can be achieved using various noise distributions, such as Gaussian, uniform, or salt-and-pepper noise. The noise type and intensity can be controlled by parameters like mean, variance, and noise type. While noise can effectively obscure certain image features, high noise levels can introduce out-of-distribution patterns, causing the model to base decisions on these artificial patterns rather than the image's inherent features.

Inpainting Substrate

Inpainting involves replacing perturbed regions with content that appears natural given the surrounding context. Deep learning models trained for inpainting tasks typically achieve this. Inpainting can generate content coherent with the surrounding regions, minimizing out-of-distribution artifact introduction. However, inpainting quality can vary, and if not executed correctly, it can introduce confounding artifacts. The results may also vary based on the inpainting model used.

Local Mean Substrate

Using the local mean of the perturbed region as the substrate involves calculating that region's average pixel value per channel and replacing all its pixels with this mean value – I introduce this in Chapter 4. My aim with this method was to erase specific local details while maintaining the region's general color or intensity, and thereby minimise confounding or out-of-distribution information. In practice I found that this substrate only marginally increased the accuracy of the *Hierarchical Perturbation* technique I introduce in Chapter 4, by around 1% on the pointing game benchmark.

For perturbation based methods in general, model sensitivity to perturbation substrate can lead to suboptimal saliency maps, particularly when the perturbation method introduces confounding artifacts. More work is needed to identify perturbation methods that are robust across a wide range of datasets and models, and to quantify the affect of different substrates in different domains. For now, choice of perturbation substrate should be informed by domain knowledge.

3.4 The Ground Truth Problem

Consider the example of classifying general categories like cats versus trees, contrasted with the more nuanced task of distinguishing between two similar cat breeds, such as a British Shorthair and an American Shorthair. For a model trained to perform the first classification task, the features it learns are likely to be broad – like the presence or absence of leaves. In contrast, a second model trained to perform the breed-specific task would need to learn more subtle features, such as fur texture or ear shape. When asked to classify a photograph of a cat, the features that each model would use – the features that each model would find *salient* – would likely be quite different.

The distinction between differential and absolute classification is also relevant here. While many machine learning models, especially in domains like medical diagnosis, might operate on a differential basis (identifying a condition by excluding others), our focus is on the absolute identification of salient features as determined by the model, regardless of the classification task's differential (or otherwise) nature. In fact, accurate saliency mapping can be a powerful tool in determining whether a model has learned to determine the presence of one class by the absence of another others.

The concept of 'true' or 'actual' *saliency* as I use it here, is then model-dependent – it denotes the regions or features of an input that *a specific model* finds most influential for its prediction. This is distinct from a human-centric or universally ideal perspective of an object's defining features in a platonic sense. We are interested in understanding what ML models have learned, which may be quite different from what humans would learn, given the same task. In order to do that, we need saliency mapping methods that effectively capture what *the model* finds salient, irrespective of what the human does.

It is essential to recognise that while human vision is unparalleled in many aspects, machine learning models operate differently. They might reach correct classifications and excellent performance using criteria distinct from human reasoning, and so find very different regions of a given input salient, when compared to a human. Thus, while human judgment provides valuable insights, it isn't an appropriate benchmark for determining how accurate a *saliency mapping method* is. In order to do that, we need to know what *the model* actually finds salient, so we can see if a given saliency mapping method accurately identifies it. Unfortunately, this *ground truth* saliency does not exist – and because of this, the evaluation of saliency mapping algorithms (and indeed, all interpretability methods) is inherently problematic. We have no ground truth to which we can compare our explanations.

A perfect explanation of a model's prediction is provided by that model's architecture, parameters and hyperparameters, along with the input. But we cannot understand it: it is *accurate*, in that it precisely describes the model's behaviour, but it is not *interpretable*, in that we cannot use it to make sense of what a model has learned. Saliency maps and other ways of visualising model behaviour are far more interpretable – but are they accurate? Moreover, how could we tell?

Imagine two saliency maps, each generated by a different saliency mapping method, with respect to the same input (a photograph of a cat) and the same model (a cat classifier). Saliency map A indicates that the most salient regions are the ears and whiskers. Saliency map B indicates the eyes and tail. Both are equally easy to interpret, both look sensible to humans, but which algorithm produces saliency maps that most accurately show what the model actually finds most salient? Which is closest to the *true* saliency of the input, according to that model? Which saliency mapping method is more accurate?

In this chapter, I describe two widely used metrics (the Pointing Game, and Insertion and Deletion) for saliency mapping evaluation and explain how they fail to capture this crucial idea of accuracy adequately – of *faithfulness* to the true saliency of the input with regard to the model. I then propose a new test, which circumvents these issues and allows for a direct, objective comparison of saliency mapping methods through the use of a proxy model.

3.5 The Pointing Game

The pointing game measures the accuracy of a given saliency map by examining the correlation between the most salient point on that map and the location of the object in question. This is done by generating a saliency map for some class prediction for a given image, and comparing the location of the actual class object in the image with the maximum (therefore, most salient point) on that map. If the maximum point falls within the boundary of the object annotation, one point is gained, and the overall accuracy is calculated as the number of hits divided by the total number of saliency maps generated.

As observed by Zeiler et al. [171], the pointing game is something of a flawed metric: it relies on the assumption that models classify inputs based on the same features or objects that a human would, and so a good saliency map would highlight those same features or objects. It uses the *human annotation* as a proxy for the ground truth of what the model actually finds most salient in a given input.

This means that it rewards saliency maps that show what a human being would consider to be salient, rather than those which accurately capture what is salient *to the model in question* – i.e. what we would want a good saliency mapping algorithm to highlight. In practice, these two things often overlap, but when a model has learned some other feature – for example, if it uses the presence of green fields to identify cows – and a saliency mapping method *correctly identifies this* – the pointing game would count this as a failure of the saliency mapping method in question, because the human annotator would expect the cow, not the field, to be salient.

Similarly, an important distinction must be made between saliency and segmentation. Saliency mapping identifies only the input regions to which the model's output is sensitive – on a local level, there is no guarantee that these regions will contain the object in question. Indeed, the absence of salient features for some class can be just as relevant to a model's prediction as their presence. This is quite different to the task of localisation, where the goal is to identify the location of objects of each class within the input, or segmentation, where the aim is to classify all input regions according to the class they belong to, whether the prediction is sensitive to their perturbation or not. As such, it is unwise to assess saliency maps as if they were segmentations – although this is not to say that saliency mapping techniques cannot be useful for segmentation, as shown in Chapter 6.

Note also that this method requires at least bounding-box annotations and ideally pixel-level annotations of objects or features in the test dataset. These can be prohibitively expensive and time-consuming for humans to create, and as described above, are at best a proxy for the ground truth of model-specific saliency.

3.6 Insertion and Deletion Metrics

Proposed by Petsiuk et al. [124], the motivation behind the *insertion and deletion metrics* is grounded in the intuitive notion that a model's confidence in its prediction should be directly influenced by the presence or absence of salient features in the input. Unlike like the pointing game, these metrics solely focus on the relationship between the saliency values assigned to pixels and the corresponding change in model output. There is no attempt to find a ground truth

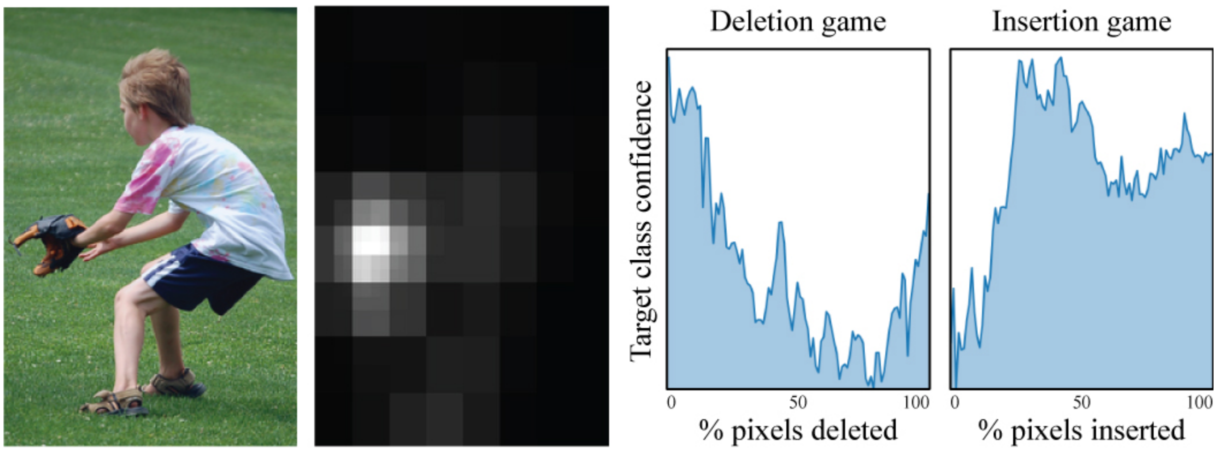


Figure 3.2: Example of the insertion and deletion metrics using HiPe for the class 'baseball glove'. The area under the curve (AUC) is used to benchmark the accuracy of the saliency maps: lower is better for deletion, and higher is better for insertion.

saliency to compare with the generated saliency map, rather, this is a self-contained evaluation metric which attempts to provide a measure of a saliency method's *internal consistency*.

The deletion metric gauges the impact on the model's confidence as salient pixels, as determined by the saliency map under assessment, are progressively removed from the image. The expectation is that as these crucial pixels are eliminated, the model's confidence in its prediction should diminish. A rapid decline in confidence, resulting in a small area under the probability curve, suggests that the saliency map has accurately identified the influential regions. To implement this metric, there are various strategies for pixel removal – see Section 3.3 for further discussion of this. Petsiuk et al. argue that setting regions to constant values is more effective for the deletion metric, as blurring small regions might not sufficiently fool a well-trained classifier, which can often infer missing details from surrounding regions, and so a zero substrate is used for the deletion metric.

Conversely, the insertion metric measures the increase in the model's confidence as increasingly more salient pixels are introduced into an initially obscured image. The rationale is that as salient regions are revealed, the model's confidence in its prediction should rise. A steep increase in confidence, leading to a large area under the curve, indicates a saliency map that aligns well with the model's decision-making process. Implementing the insertion metric presents its own challenges. Pixels can be introduced onto a constant canvas, but this can sometimes introduce misleading evidence that confounds the classifier. For instance, introducing an oval-shaped region onto a blank canvas might erroneously lead a classifier to predict 'balloon'. To mitigate such issues, Petsiuk et al. propose starting with a highly blurred image and gradually revealing regions, as this approach preserves some low-level details without introducing sharp, potentially misleading boundaries.

The exact number of pixels manipulated at each step isn't explicitly mentioned in the publication. Pixels could be added or removed one at a time, starting from the most salient pixel and proceeding in descending order of saliency. This would produce the most fine-grained AUC, but would be quite computationally expensive depending on the input size. Instead, we could set a certain percentage of the most salient pixels are added or removed in each iteration. For instance, 5% or 10% of the most salient pixels might be removed or added at each step. The public codebase from Petsiuk et al. [125] uses this approach, taking a 'steps' parameter. The total number of pixels in the input is divided by this to give the number of pixels to insert or delete at each step. The default number of steps is 100.

As neural networks are known to capture hierarchical and non-linear relationships in their training data, the relationship between individual input features or regions and a model's output might not be straightforward [59] – causing non-smooth and potentially confusing changes in the output when *any* region of the input is perturbed. Consider a task where a model is trained to identify emotions. An image of a face displaying a smile might be classified as 'happy'. Now, if a saliency mapping method identifies the corners of the mouth (where the smile forms) as highly salient, it seems intuitive. However, when using the deletion metric, as we progressively remove the pixels around the mouth, we might expect the model's confidence in the 'happy' classification to decrease. But what if it doesn't decrease as sharply as anticipated — or even increases?

It could be that our saliency mapping method is poor, and the pixels we are removing actually don't matter. Or, it could be that something else is going on: facial recognition models often learn to recognise emotions based on a combination of features, not just the mouth. The eyes, for instance, play a crucial role in emotion recognition. A genuine smile, often termed a Duchenne smile, involves not just the mouth but also the eyes [42] – so, even as the salient pixels of the mouth are removed, the model might still detect happiness based on the eyes – or even find the eye region *more* salient than before, as the mouth is obscured. This could cause a higher-than-expected AUC, even if the original saliency map happened to perfectly capture the saliency of the un-perturbed image – the act of perturbing the input *changes* its salient regions with respect to the model.

3.7 Summary

This chapter explored the use of saliency mapping in AI interpretability, particularly with respect to its applications in image classification, as an important tool for identifying biases, validating feature usage for predictions, and enhancing overall model transparency. I drew attention to the important distinction between model-specific and model-agnostic methods, and discussed their relative strengths and limitations: Model-specific methods, which include techniques like Gradient, Guided Backpropagation, Grad-CAM, and others, exploit network architecture, utilizing gradients and activations for visualisation. However, these methods are often hampered by their dependency on specific architectures and potential for noisy or coarse maps. Model-agnostic methods, such as Extremal Perturbation and Random Input Sampling for Explanation (RISE), focus on input perturbation to quantify model output sensitivity. These methods offer more direct interpretability and don't assume specific architectures, so can be used with any model – but they have high computational demands and require heuristic parameter selection.

I also discussed a key issue with *all* saliency mapping methods: 'The Ground Truth Problem'. This hampers assessment of different saliency mapping methods due to the difficulty in defining what constitutes 'true' saliency, as it varies from model to model and can diverge from human perceptions. I argue that we cannot assume that a given model has learned to use the same features in the input that a human would do, and therefore that saliency maps agreeing with human annotations is not guaranteed to be a reliable measure of their accuracy.

I then covered existing methods for the evaluation of saliency maps, such as the Pointing Game and Insertion and Deletion metrics. These metrics, while useful, have inherent limitations such as over-reliance on human judgment and potential inconsistencies in model responses to input perturbations.

Chapter 4

Hierarchical Perturbation

In this chapter I propose Hierarchical Perturbation (HiPe), a novel approach to perturbation-based saliency mapping, which identifies salient regions regardless of scale, largely removes the need for heuristic parameter selection, and dramatically reduces computational cost whilst maintaining accurate saliency identification. It provides these benefits by iteratively identifying salient sub-regions and disregarding relatively unimportant ones in increasing resolution. This approach is compared to other saliency mapping methods on the established pointing game benchmark [172] and the causal insertion/deletion metric [124], and evaluated on the commonly-used MSCOCO and VOC2007 validation and test datasets. HiPe is over 20$\times$ faster than existing model-agnostic methods while achieving comparable performance on these benchmarks.

Hierarchical Perturbation is a natural extension of iterative occlusion [171] and the random masking of RISE [124] described in Chapter 3, in which the same principles of empirical saliency estimation are adopted, but here applied in a more directed fashion to minimise computational cost and thereby make model-agnostic saliency mapping a realistic prospect for large samples or datasets. The critical insight is that a large amount of superfluous computation is performed by current methods when regions with little effect on the model output are iteratively perturbed, or when random perturbation region selection results in spatially similar or overlapping regions. By avoiding this unnecessary cost through saliency thresholding, it is possible to perform model-agnostic saliency mapping an order of magnitude faster than existing methods.

Algorithm

As shown in Figure 4.1, HiPe works by focusing on perturbing the most salient regions with increasing resolution whilst ignoring regions which do not change the model's output. Note that although the following explanation uses an image as the model input for clarity, HiPe does not require the input to be any specific shape.

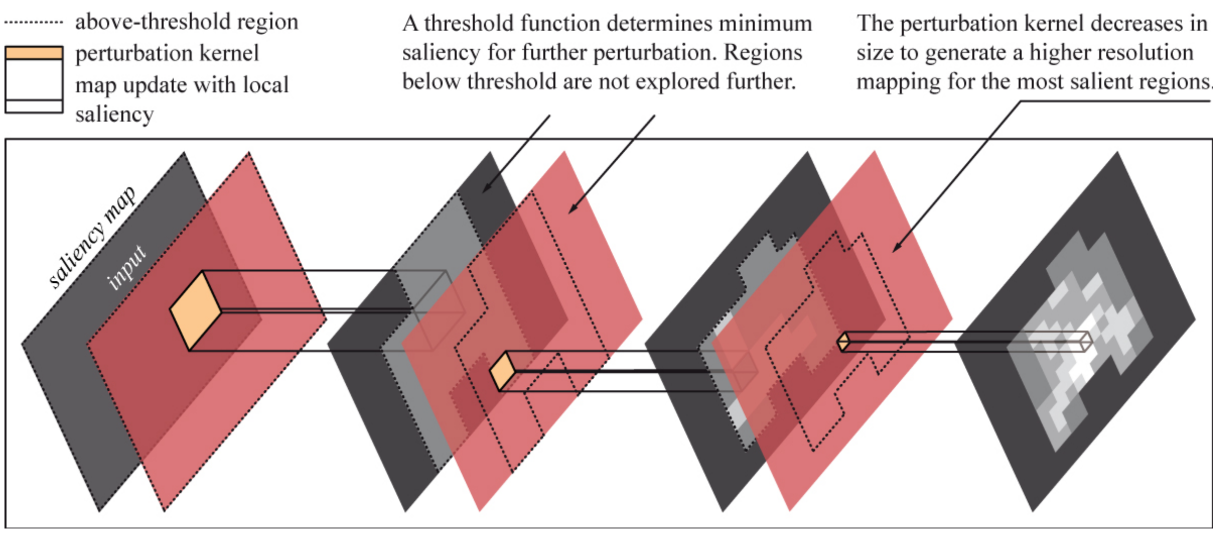

above-threshold region
perturbation kernel
map update with local saliency
A threshold function determines minimum saliency for further perturbation. Regions below threshold are not explored further.
The perturbation kernel decreases in size to generate a higher resolution mapping for the most salient regions.
saliency map
input

Figure 4.1: Saliency Mapping with Hierarchical Perturbation

Let f be the trained model, which takes $\mathbf{X}$, a matrix of size $3 \times h \times w$ (in this case, a three channel colour image with variable height and width), and returns a scalar output. $\mathbf{S}$ is instantiated as the saliency map, initially a zero matrix of size $h \times w$, and iteratively populated as follows. First, a set of masks is generated (denoted $P = \{\mathbf{M}_1, \mathbf{M}_2, \dots\}$), such that:

1. Each is matrix of size $d \times d$, where $d = \lceil \log_2(\min(h, w)) \rceil$. This results in a mask dimension that is always neatly divisible by two, ensuring that the perturbation regions will overlap evenly.

2. Each $\mathbf{M} \in P$ contains a unique 2×2 region of ones, with all other elements set to zero (this is to ensure an overlap of perturbation to capture features on the border of regions), and;

3. For all $\mathbf{M} \in P$, $t(\mathbf{S}, \mathbf{M}) = 1$, where t is a step function using the mid-range of the current saliency map as a threshold to identify regions of high salience for higher-resolution map-

ping: thus (where $\circ$ denotes the Hadamard (element-wise) product, and u an upsampling operation using proximal interpolation which resizes the $d \times d$ mask to $h \times w$):

$$t(\mathbf{S},\mathbf{M}) = \begin{cases} 1, & if \qquad \max(\mathbf{S} \circ u(\mathbf{M})) \geq \min\left(\mathbf{S} + \frac{\max(\mathbf{S}) - \min(\mathbf{S})}{2}\right) \\ 0, & otherwise \end{cases} \tag{4.1}$$

The mid-range of the current saliency map is used in preference to some arbitrary threshold as it allows us to handle varying saliency distributions across different samples. Selecting an optimal threshold for all samples would require either knowledge about the saliency distribution beforehand, or costly heuristic trials – whilst manually selecting one for each sample would optimise for achieving visually pleasing saliency maps, and render comparing the relative saliency of different samples across the dataset difficult. The mid-range is used for this rather than the mean or some other aggregate of the current map, as the mid-range is extremely sensitive to outliers. This sensitivity allows us to focus on the most salient regions quickly in cases where only a small region of the input is salient, significantly improving performance compared to the mean. An example of this is shown in Figure 4.2.

Note that the first time this thresholding operation is applied, $t(\mathbf{S},\mathbf{M}) = 1$ in every case, because both $\mathbf{M}$ and $\mathbf{S}$ are initialised as zero matrices, so the first set of masks contains every possible unique position of the 2×2 region of ones.

Next, a copy of the input $\mathbf{X}$ is made, in which the pixels that correspond to the non-zero region of each upsampled mask $u(\mathbf{M})$ are perturbed, by replacing all pixels therein with the mean pixel values of that region. Call this perturbation operation p. The saliency map $\mathbf{S}$ is then updated with the difference between the model's output given this perturbed input $f(p(\mathbf{X},u(\mathbf{M})))$, and its output given the original unperturbed input $f(\mathbf{X})$:

$$\mathbf{S}' = \mathbf{S} + \sum_{\mathbf{M} \in P} ReLU(f(\mathbf{X}) - f(p(\mathbf{X},u(\mathbf{M}))) \circ u(\mathbf{M})). \tag{4.2}$$

This means that the region of the saliency map which corresponds to the perturbed region in the input has the difference in the model output added to it.

ReLU (as defined in Equation 2.2) is used so that the saliency map is updated only in response to perturbations which *decrease* the confidence of the target class, and therefore only highlights regions that, when available to the model, *increase* the target class prediction. d is then doubled, halving the size of each perturbed region with respect to the input, and the above steps (generating a new set of masks, applying them to the input, updating the saliency mask with the change in the model's output, increasing d) repeated while $d \leq \frac{\min(h,w)}{4}$. This effectively means that HiPe can consider features as small as 2×2 pixels in size, as the minimum perturbation region size is 4×4, with a 50% overlap.

Usage of the local mean as a perturbation substrate is further discussed in Chapter 3.3, although it is quite possible to use any perturbation substrate with HiPe – this is an advantageous property, because *how* best to perturb the input for perturbation-based saliency mapping methods remains an open question [30] as discussed in Section 3.3.

To summarise intuitively, as described in algorithm 1, HiPe begins by perturbing large, overlapping regions and using the difference in the model output for each of these perturbations to update the saliency map. All regions of the saliency map that exceed the saliency threshold are split into smaller overlapping regions, which are then each perturbed, and the saliency map updated in turn – and so on – until either the minimum perturbation size is reached or no region remains above the saliency threshold. By discarding regions with little impact on the model's output and focusing only on the more salient areas, it is possible to generate saliency maps of comparable quality to the state-of-the-art for model-agnostic methods at a fraction of the computational cost.

4.1 Experiments

In this section, I compare HiPe to popular alternatives from the literature using two widely adopted saliency mapping benchmarks – the pointing game and insertion/deletion causal metrics, which are described in detail in sections 3.5 and 3.6 respectively. As discussed in Chapter 3.5, the pointing game is something of a flawed metric – it relies on the assumption that if a model

Algorithm 1 Hierarchical Perturbation Algorithm (HiPe)

```
Require: model $f$, input image $\mathbf{X}$ of size $3 \times h \times w$
  Initialise saliency map $\mathbf{S}$ as a zero matrix of size $h \times w$.
  Set $d \leftarrow \lceil \log_2(\min(h,w)) \rceil$.
  while $d \leq \frac{\min(h,w)}{4}$ do
      Initialise an empty list for set of perturbation masks $P$.
      Calculate the mid-range threshold $t$ of $\mathbf{S}$.
      $t \leftarrow \min(\mathbf{S}) + \frac{\max(\mathbf{S}) - \min(\mathbf{S})}{2}$
      for each unique $2 \times 2$ region within $d \times d$ do
          Initialise mask $\mathbf{M}$ as a zero matrix of size $d \times d$.
          Set the $2 \times 2$ region of $\mathbf{M}$ to ones.
          Upsample $\mathbf{M}$ to size $h \times w$ using proximal interpolation.
          if $\max(\mathbf{S} \circ \mathbf{M}) \geq t$ then
              Add $\mathbf{M}$ to the set $P$.
          end if
      end for
      for each mask $\mathbf{M} \in P$ do
          Create perturbed input $p(\mathbf{X}, \mathbf{M})$ by perturbing the region in $\mathbf{X}$ corresponding to the
  non-zero region in $\mathbf{M}$.
          Compute the model output difference $\mathrm{ReLU}(f(\mathbf{X}) - f(p(\mathbf{X}, u(\mathbf{M}))))$.
          Update saliency map: $\mathbf{S} \leftarrow \mathbf{S} + \mathrm{ReLU}(f(\mathbf{X}) - f(p(\mathbf{X}, \mathbf{M})) \circ \mathbf{M})$.
      end for
      Double $d$.
  end while
  return Saliency map $\mathbf{S}$
```

performs well, it has learned the same features that a human would, and so an accurate saliency map would highlight those same features. This means that it rewards saliency maps that highlight what a *human* would expect to be salient, rather than what is in fact salient *to the model*, which might well be different. Nonetheless, given a well trained model it can be considered a reasonable proxy, and since it is a standard comparator I include it here. I used a publicly available pre-trained ResNet50 model [64] for these experiments, along with the MSCOCO 2014 validation set and the VOC 2007 test set, to allow for comparison with previous works. Further evaluation in Chapter 5 follows using the Proxy Model Test.

		COCO14 Val			VOC07 Test		
	Method	All	Diff	Time (s)	All	Diff	Time (s)
	cMWP [172]	58.5	53.6	0.08	90.6	82.2	0.09
	GCAM [138]	57.3	52.3	0.03	90.4	82.3	0.03
MA	ExtP [46]	**55.7**	46.9	53.4	86.3	73.4	53.5
MA	RISE [124]	55.6	–	25.87	**88.9**	–	23.61
MA	**HiPe**	54.6	**49.6**	**0.94**	85.6	**75.1**	**0.95**
	MWP [172]	49.6	43.9	0.06	84.4	70.8	0.06
	Guid. [147]	42.1	35.3	0.1	77.2	59.5	0.1
	Grad [143]	35.0	29.4	0.06	72.3	56.8	0.06
	DConv [171]	30.0	21.9	0.06	68.6	44.7	0.06

Table 4.1: **Pointing Game**: Mean accuracy and time on *full* and *difficult* validation/test sets of VOC07 and COCO14 defined by by Zhang et al. [173], with the *difficult* subset containing only "images that meet two criteria: 1) the total area of bounding boxes (or segments in COCO) of the testing category is smaller than 1/4 the size of the image and 2) there is at least one other distracter (sic) category in the image."). Best results are highlighted within the model-agnostic (MA) subset. As the computational cost of rerunning saliency map generation on the entire COCO and VOC datasets was prohibitive, accuracy results for methods other than HiPe are taken from Fong et al. [47], and RISE results taken from Petsiuk et al. [124] (which excluded the difficult subset). Average saliency map generation times are calculated using Torchray implementations with default settings for 1000 random samples (one random class per sample) using an NVIDIA GeForce RTX 2060.

The Pointing Game

Table 4.1 shows that HiPe is competitive with existing model-agnostic saliency mapping methods Extremal Perturbation (ExtP) and RISE, whilst averaging under a second to produce a saliency map. In contrast, RISE requires over 23 seconds, and Extremal Perturbation takes nearly a minute per image. Both of these methods could be made faster by decreasing the number of random masks, or the number of iterations respectively – but at the cost of increasing noise in the saliency map generated.

Insertion and Deletion Metrics

In Table 4.2, I show that HiPe is competitive with all methods across both metrics and datasets, and marginally outperforms all methods on the insertion metric. This is expected, because unlike the model-specific methods, the saliency maps generated by HiPe are products of directly

		COCO14 Val		VOC07 Test	
	Method	Insertion	Deletion	Insertion	Deletion
MA	**HiPe**	**0.68**	0.43	**0.67**	0.42
	GCAM	0.67	0.41	0.67	0.39
	Guid.	0.66	0.39	0.65	0.38
	cMWP	0.66	0.39	0.65	0.38
MA	RISE*	0.65	**0.40**	0.65	**0.39**
	ExtP	0.63	0.45	0.62	0.45
	MWP	0.62	0.38	0.62	0.34
	DConv	0.62	0.39	0.62	0.44
	Grad	0.62	0.46	0.61	0.44

Table 4.2: **Insertion and Deletion AUC:** A comparison of mean Area Under Curve for insertion (higher is better) and deletion (lower is better) causal metrics described by Petsiuk et al. [124] on 1000 randomly selected samples, for: HiPe (proposed herein); Random Input Sampling for Explanation (RISE) [124]; Extremal Perturbation (ExtP) [46]; Guided Backpropagation (Guid.) [147]; Gradient (Grad) [143]; Grad-CAM (GCAM) [138]; Contrastive Excitation Backpropagation (cMWP) [172]; Deconvnet (DConv) [171]; and Excitation Backpropagation (MWP) [172]. As in Table 4.1, superiority within the subset of model-agnostic (MA) methods is highlighted. The percentage of pixels removed or added at each step is set to 1%, the blurred substrate is used for insertion, and the zero substrate for deletion as per the literature.

perturbing the input in a comparable way to the causal metrics used here. This is also the case for RISE – the small decrease in comparative performance is due to the stochasticity inherent in this method.

Figure 4.3 shows examples of saliency maps generated with HiPe, and Figure 4.4 compares saliency maps across all methods for a single image for the class ‘fork’. HiPe, Grad-CAM and cMWP identify the fork in the image as salient, but the other methods do not. It has been shown that deconvnet and guided backpropagation are in some cases invariant to reparameterisation in later layers [113], and essentially act as image reconstructors – this is particularly evident in the guided backpropagation (Guid.) map here. The failure of RISE and Extremal Perturbation to localise the fork in the image is due to the small size of the features in question – both RISE and Extremal Perturbation rely on heuristic hyperparameters which dictate the size of salient regions to be localised. HiPe, by contrast, performs well on inputs containing salient regions of all sizes.

Figure 4.5 shows an example of HiPe applied to a segmentation task – that of immune cell segmentation from Hoechst-stained whole slide images (WSIs) using a deep residual U-net, which will be discussed in more detail later in this work. This example showcases the applicability of this model-agnostic method to arbitrary architectures, and leverages the speed and robustness of HiPe in a use case where other perturbation-based techniques would have been prohibitively slow for multiple large WSI images, and gradient-based methods too indistinct for the relatively small and sparse salient features (as shown in the benchmark experiments – see Figure 4.4).

As shown by Adebayo et al. [10] and Kindermans et al. [73], robust saliency mapping algorithms must be sensitive to input with respect to their target output class – for example, given an image containing both a cat and a dog, and a trained neural network that classifies cats and dogs, one would expect that a saliency map generated with respect to the 'cat' output would look very different to that of 'dog' – given of course, that the neural network has successfully learned 'cat' and 'dog' specific features.

In order to investigate this property, *class-specific* HiPe saliency maps are generated using a simple model trained on the MNIST dataset. MNIST is used in this case partly because it is a common sanity-checking dataset in the literature and therefore allows for easy comparison with other saliency-mapping experiments [73], but primarily because due to its simplicity, it is free from potentially confounding spurious correlations in the input. Therefore one can confidently expect a well-trained model to classify digits based only on the digits themselves.

A simple three layer convolutional network was trained using cross entropy loss and SGD with a learning rate of 0.001 and momentum of 0.9. The network had two convolutional layers (the first having 16 channels and the second 32), each with a kernel size of 3, a stride of 1, and zero padding. Each was followed with a ReLU and maximum pooling with kernel size 2. The final was a standard linear layer. This network was trained to 97% accuracy, and HiPe applied to each class output in turn. Figure 4.6 shows the results of this experiment on a randomly chosen sample for each digit. The saliency maps for each digit are distinct and sensible. This image confirms that HiPe is sensitive to the target class for each map, and can be interpreted intuitively – the zero input has high saliency for the zero class around the entirety of the digit. For other class saliency maps on the zero input, HiPe finds portions of the input salient – for example, the upper curve for class 9, the lower curve for classes 3 and 5, and the leftmost curve for class 6.

4.2 Discussion

Unlike RISE [124], which generates masks randomly, and methods which learn secondary models through gradient descent such as those of Fong et al., Ribeiro et al. and Dabwoksi et al. [47, 128, 36], HiPe contains no random elements. However, HiPe is not able to capture instances where the salience of two spatially distinct features in combination is greater than the sum of each feature individually.

An example of this is shown in Figure 4.7. Because HiPe only perturbs one location at a time, it is unable to take into account how the model output changes if two or more separate relevant areas are perturbed. As RISE is able to capture this (*if* one of more of its the random masks happens to cover both areas), in this case it seemingly highlights not only the area around the fridge, but also the rest of the kitchen. It is hard to say this definitively, as we do not have ground truth saliency beyond human annotation, which, as discussed in Section 3.4, may not accurately reflect what the *model* finds salient. In any case, the saliency map generated by HiPe does not show the rest of the kitchen area to be as salient as that generated by RISE, and fails to assign the highest saliency to the fridge. In fact, the most salient point produced by HiPe is far from the fridge. This is likely due to the permutation method used – in this case the local mean – which replaces the perturbed region with the mean pixel values of that region (per channel). Looking at the bottom right corner of the image, where the highest saliency marker is located for HiPe, we can see that replacing a square region with the local mean there would result in a pale square – something that, combined with the other features HiPe has highlighted that appear in the image, perhaps appears to the model as a fridge, leading to the erroneous saliency assignation.

The perturbation method chosen for any perturbation-based saliency mapping method is important, as discussed in Section 3.3 – and this example suggests that HiPe is vulnerable to failure in edge-cases like these when the perturbation method introduces confounding artefacts.

The lack of ground-truth saliency information for any complex model makes these edge cases hard to predict ahead of time, and so selecting a perturbation method is difficult – the model *might* mistake a pale square for a fridge and so make attribution mistakes like this, or it may have learned other features entirely. Feature visualisation, discussed in Section 2 offers one route to understand which features a model is sensitive to, and may provide some evidence for or against a particular choice of perturbation method — for example, if a learned prototype for a fridge contains a pale square region, we might be able to predict that perturbation methods that can result in pale square artefacts will confound saliency mapping methods generated using them, making the perturbed region more likely to be salient, if the target class has similar features to those introduced by the perturbation method. Note that for this particular image RISE will not be confounded in the same way, as but default it a) blurs the masks it generates to remove hard edges, and b) zeros out the masked regions, rather than using the local mean perturbation method.

Another potential failure mode of HiPe, shown in Figure 4.8, is if the model in question finds many separate regions equally salient. Due to HiPe's thresholding process, any region less salient than the mid-range of the total saliency is not perturbed at greater depth – but this will result in ignoring regions that are only marginally less salient. This can also be the case with Extremal Perturbation, as it necessitates the selection of a number of mask area sizes – if these are set too small, the optimiser will focus on a subset of the salient regions [46]. RISE should not be limited by this, given a sufficient number of random masks, although it is affected by the probability of a cell within the mask being set to zero, which is also a hyperparameter [124]. HiPe's threshold could be adjusted to allow it to capture salient areas that are less salient than the current mid-range – indeed, the threshold *could* be set to perturb *all* regions that have a positive effect on the target class prediction, but at great computational cost.

As discussed, HiPe is model-agnostic and, as such, can be applied to any model which maps an input to an output, irrespective of the architecture. HiPe does not apply any smoothing, either to the perturbed region (typically done to make the perturbation appear more natural) or to the resulting saliency map (which some methods do purely for the sake of visual appeal rather than for a fundamental theoretical reason). Using a similar method to RISE in which a low resolution mask is upsampled using bilinear interpolation in order to generate smooth perturbations actually resulted in a slight decrease in performance when using HiPe – this is in contrast to RISE and Extremal Perturbation, which explicitly apply smoothing. A possible hypothesis for the appeal of smooth perturbations, and resulting smooth saliency maps, is simply human visual preference (sharp artefacts are more visually disturbing than smooth ones [47]) rather than explanatory power – it may be assumed that smooth perturbations are less confounding, but this is not borne out by these results.

Most state of the art work in saliency mapping focusses on model-specificity, assuming a certain subset of architectures and requiring access to the model's internal state. The few model-agnostic methods that do exist perform well on the benchmarks herein examined, but can be very slow – prohibitively so for large samples. HiPe is able to create model-agnostic saliency maps so quickly because it is saliency-aware in a way that existing perturbation-based saliency mapping algorithms are not, iteratively focussing perturbation only on regions where saliency has been established. Petsiuk et al., Fong et al., and Zeiler et al.[124, 48, 171] require a pre-specified number of iterations – whether that is epochs, number of random masks generated, or occlusion kernel size and step – which fix the amount of computation required for an input of given size irrespective of the proportion of the input that is actually salient. It is also impossible to know ahead of time what the optimal value for these parameters might be in order to trade off accuracy and efficiency, and finding the optimal parameters (for one input sample or across an entire dataset) requires many trials. Additionally, these parameters can also limit the size of the salient region that can be detected, which can lead to omissions, as in Figure 4.4. The proposed method, by contrast, continually disregards regions which have little impact on the model output and by doing so inherently limits the amount of computation required – without imposing restrictions on the size of the salient region it is possible to detect.

In summary, this chapter introduced hierarchical perturbation (HiPe), a novel model-agnostic saliency mapping method designed to identify salient regions at any scale, minimise heuristic parameter selection, and reduce computational costs – while making no assumptions about the model's underlying architecture. HiPe works by perturbing large regions initially, and progressively focussing on smaller, more salient regions, disregarding regions which have little impact on the model's prediction. It uses the local mean as a perturbation substrate, which is adjustable to suit different domains. HiPe's performance was validated on the pointing game and insertion/deletion benchmarks using standard datasets, demonstrating comparable accuracy to existing methods but with significantly ($20\times$) reduced computational cost.

HiPe's limitations include its inability to capture the combined salience of separate features, and potential susceptibility to perturbation method artifacts. It is also sensitive to the choice of perturbation method, as demonstrated in examples where local mean perturbation introduced misleading saliency.

Overall, HiPe offers a robust, efficient alternative for model-agnostic saliency mapping, applicable to various architectures without requiring access to the model's internal state. It addresses the computational efficiency concerns of existing methods while maintaining comparable accuracy.

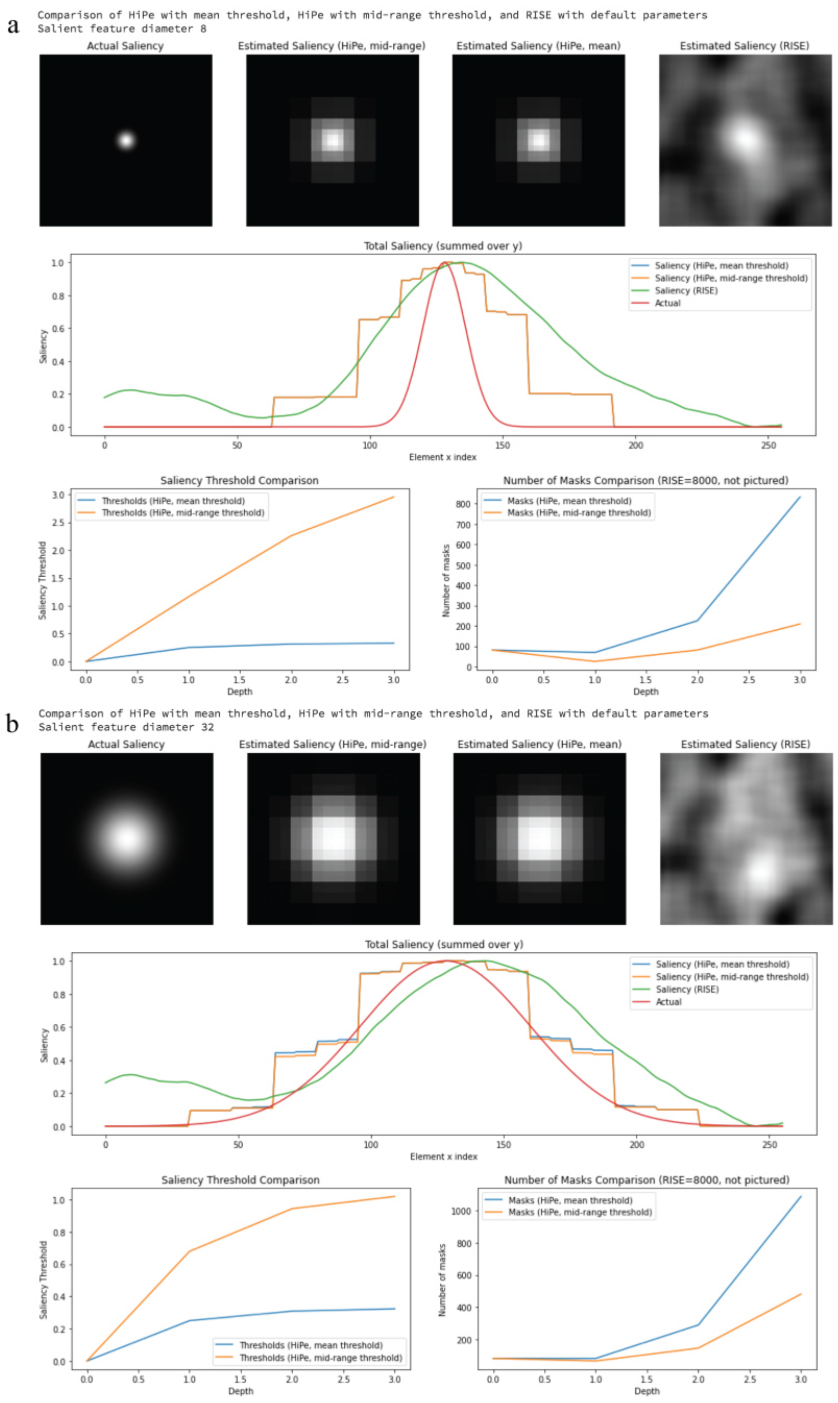


Figure 4.2: Here 256×256 images are generated with a central blurred 'salient' region of two sizes (diameter 4 above and diameter 32 below) to demonstrate the saliency maps produced by HiPe. I use a summation operation as a proxy model (see Chapter 5) and compare RISE with default parameters against HiPe with two different saliency threshold methods.

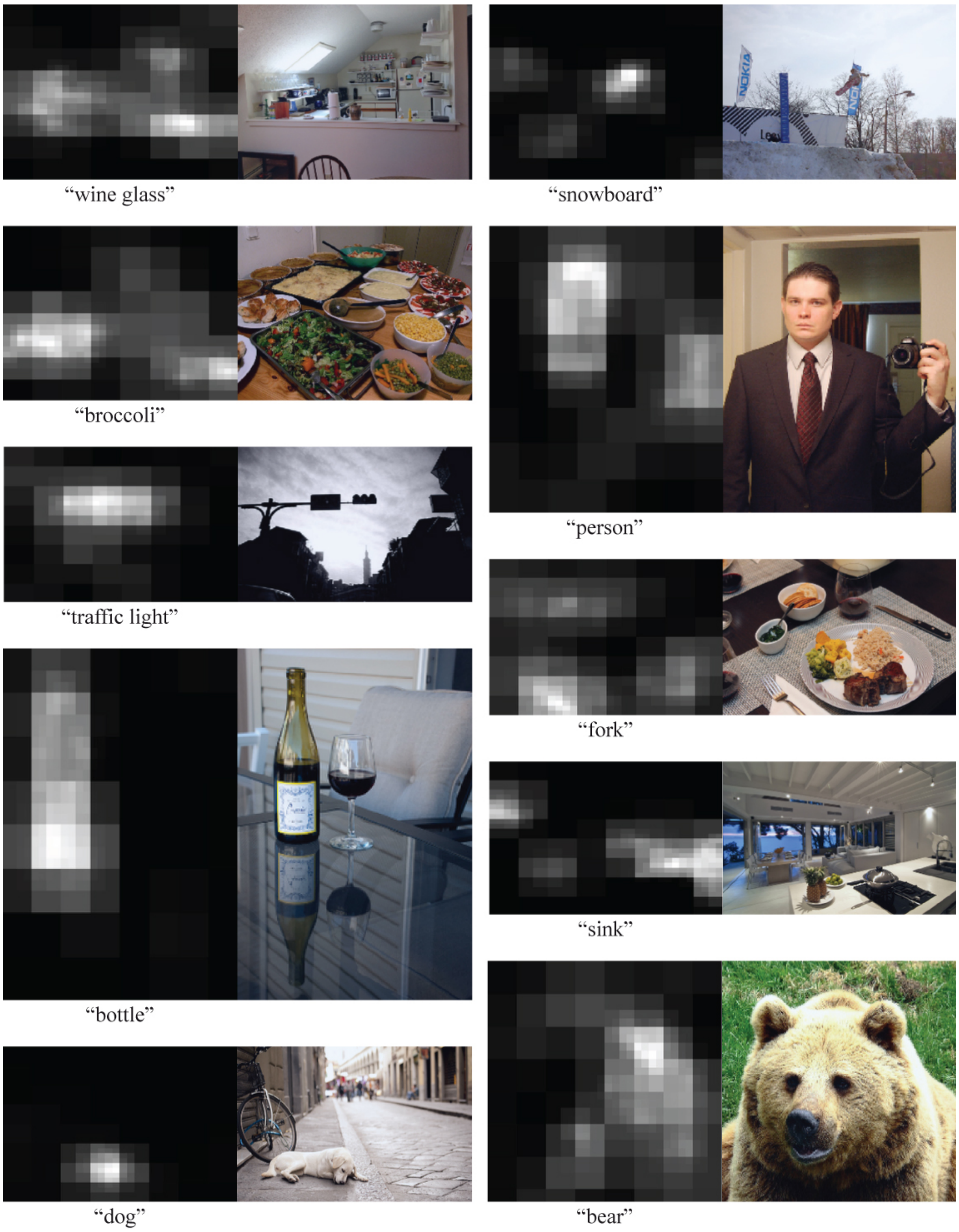


Figure 4.3: Examples of saliency maps generated with Hierarchical Perturbation. Note that more salient regions are of higher resolution - for example, for the "sink" class, the most salient region on the right has more finely-grained saliency than the less salient regions on the left.

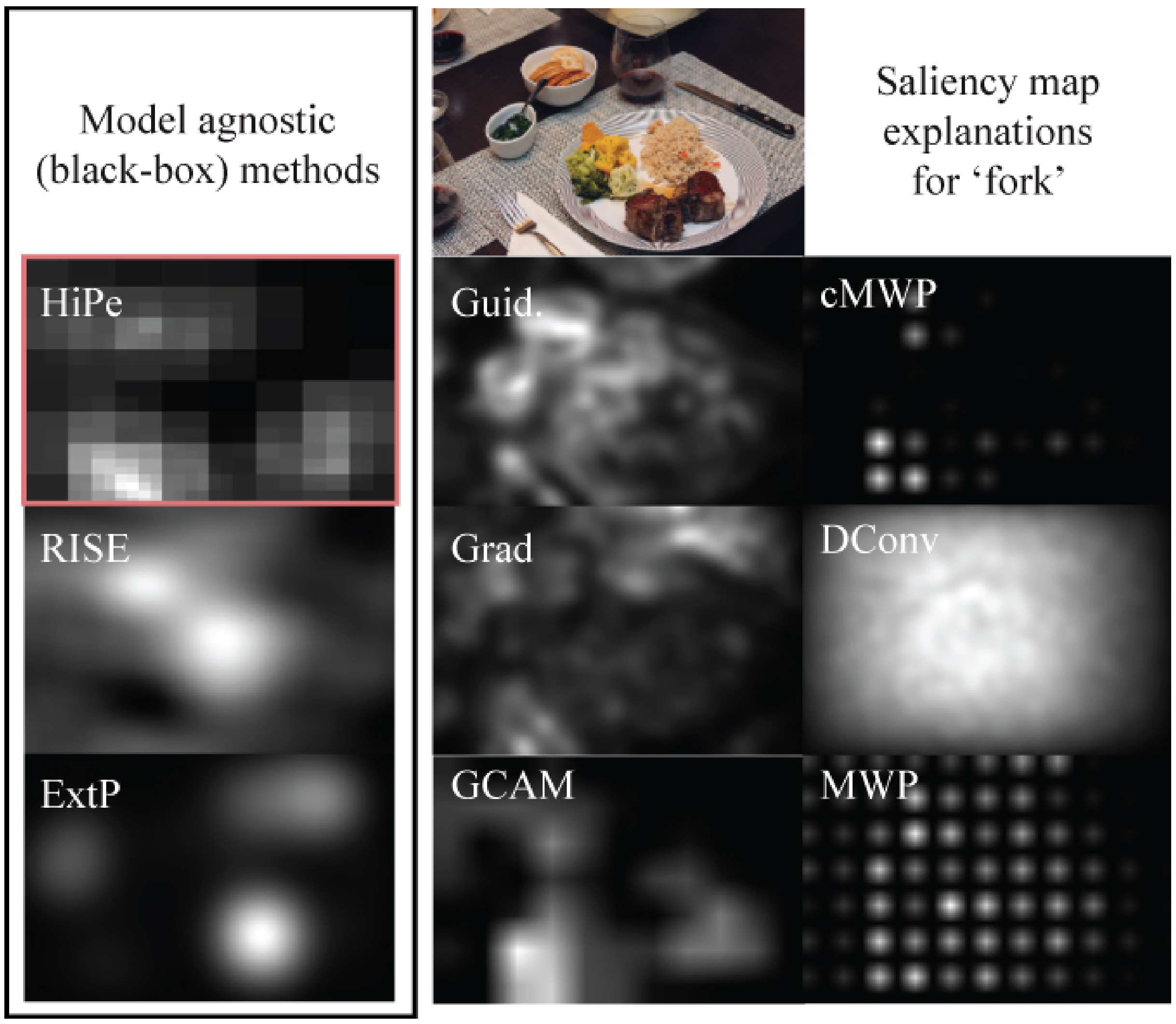

Model agnostic (black-box) methods
Saliency map explanations for 'fork'
HiPe
Guid.
cMWP
RISE
Grad
DConv
ExtP
GCAM
MWP

Figure 4.4: Examples of saliency maps for the class 'fork'. This example was chosen to highlight the surprising variation in maps generated with different methods.

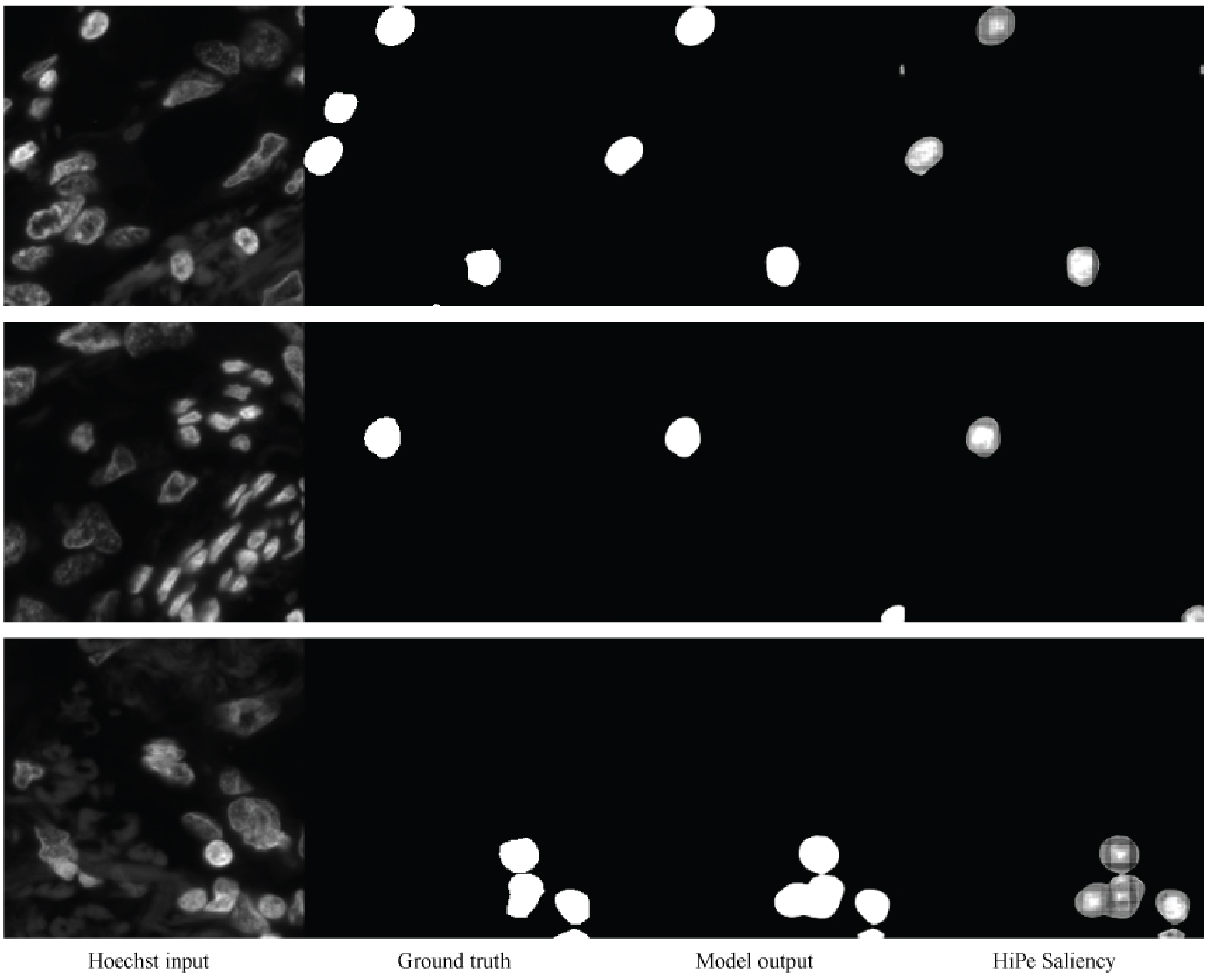


Figure 4.5: Use of HiPe for an immune cell segmentation task using a deep residual U-net on Hoechst-stained biopsy slides. Here HiPe enables fast and detailed saliency map generation for high resolution images with small, sparse features.

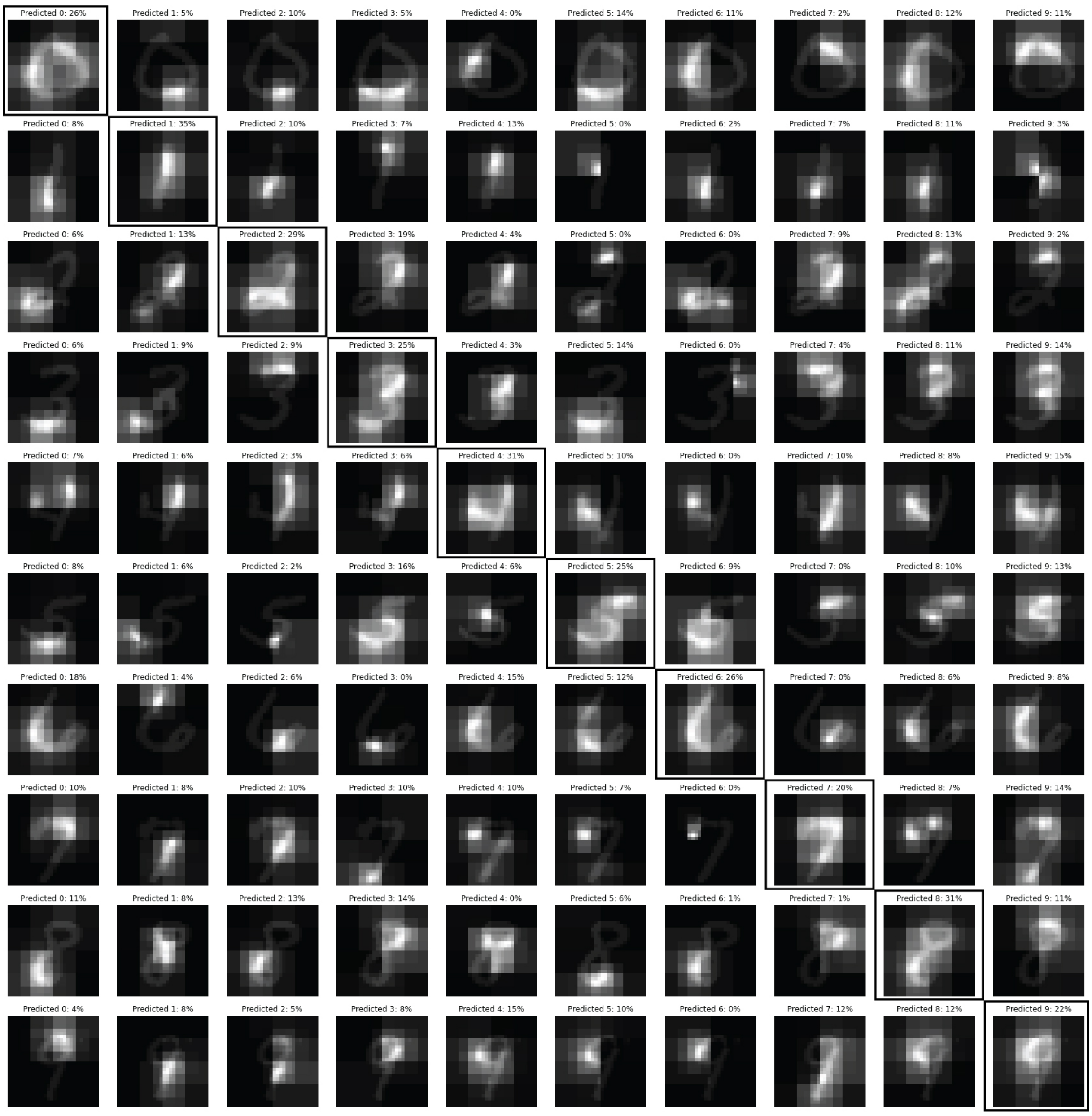
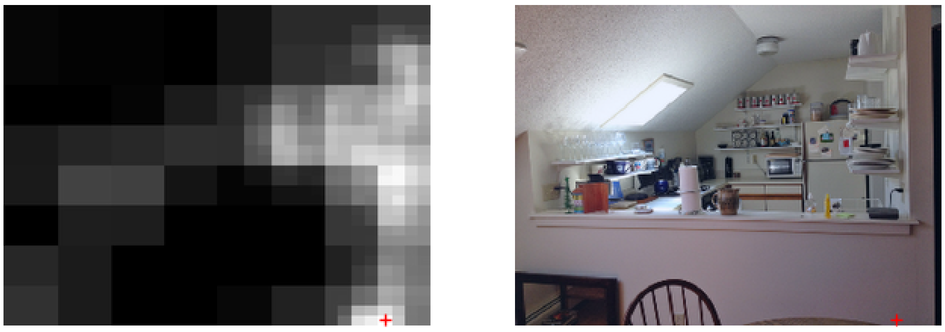
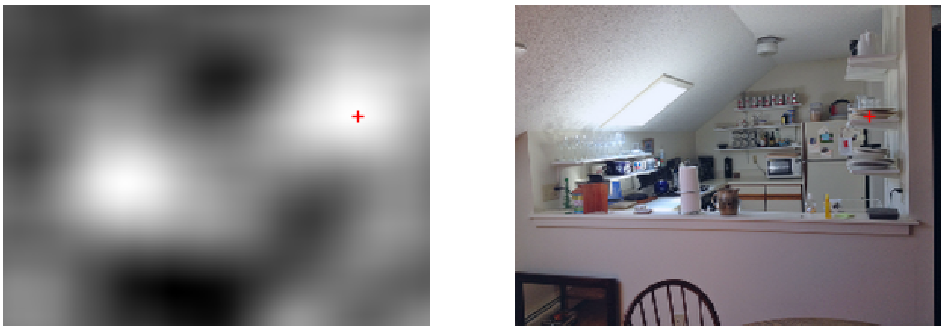


Figure 4.6: HiPe saliency on MNIST. Each row comprises saliency maps for each class on a single randomly chosen sample image. The top left to bottom right diagonal contains the saliency of the correct class for each sample. The probabilities shown here are the model output logits normalised to sum to one and rounded to the nearest percentage point.



(a) HiPe saliency map (left) and input image (right).



(b) RISE Saliency map (left) and input image (right).

Figure 4.7: Saliency maps for RISE and HiPe with respect to the ‘fridge’ class, for an image from the MSCOCO dataset. The location of the most salient point in the saliency map is marked with a red cross.

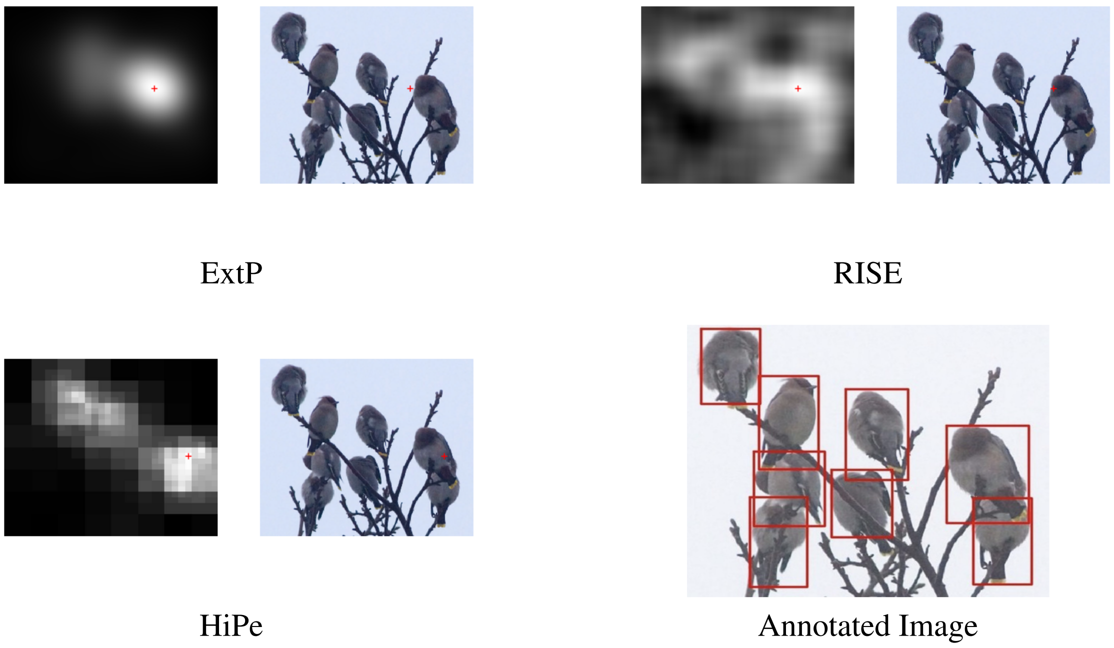


Figure 4.8: Saliency maps with respect to the ‘bird’ class, for an image from the MSCOCO dataset. The location of the most salient point in the saliency map is marked with a red cross. The lower right image shows the human-annotated bounding boxes for that class. Note that while we do not have ground-truth saliency for this model, since the model correctly classified the image and the image itself contains little else except birds, we can cautiously assume that the most salient regions are truly the birds themselves. Both HiPe and ExtP, and to a certain extent RISE (although more diffuse), show the birds on the left to be the most salient – which is surprising, since there is little visual difference between them and the birds on the right. Is this a failure of the model or the saliency mapping methods?

Chapter 5

The Proxy Model Test

In this chapter, I describe a new method for objectively evaluating model-agnostic saliency mapping methods. This method is motivated by the limitations of existing saliency mapping assessment methods discussed in the previous chapters – and by the observation that, in order to accurately assess the accuracy of a saliency map with respect to some input and some model, ideally we need access to the ground truth – to which regions of that input are *actually* salient *to that model* – so that we can compare it with the map generated by some proposed saliency mapping method.

However, such a ground truth is not typically available, because the models that we are interested in generating saliency maps for are too complex – we cannot just inspect the model architecture and parameters in order to verify that some region identified as salient *by the saliency mapping method* is actually salient to that model – we don't know what the model has learned to look for, and this is why we need saliency maps in the first place.

Crucially, if a given model is so complex that saliency mapping methods are required to understand which input regions are salient, then the 'ground truth' saliency is by definition made opaque by the complexity of the model – which means we don't have a ground truth saliency available to check if those saliency mapping methods are accurate!

But, what if we replace this model with one which **is** transparent, so that we know the ground truth saliency map?

This is the key insight of the proxy model test: for the purpose of assessing the accuracy of *model-agnostic* saliency mapping methods, *the model could be anything* – and so we may as well select a transparent proxy model, for which the ground truth is available.

Consider a very simple model that simply outputs the sum of its input. This model is simple enough that it is completely interpretable. Crucially, given some input, we can easily calculate what this model's output will be. Moreover, we *know* the exact saliency of each element in that input – how much each input element contributes to the output.

When using the summation operation as the proxy model, the ground truth saliency becomes immediately apparent: it is the input itself. The saliency of each input element, with respect to the proxy model, is precisely equal to the value of that element. Furthermore, the cumulative saliency of a specific *region* is the sum of all elements within that region. Other explicit proxy models could also be used – for example, a *product* proxy model would capture the ability of saliency mapping methods to identify multiple features that are salient individually, but much more so in combination. Other interpretable proxies could be designed to mirror behaviours of real-world models, and thereby evaluate the ability of different saliency mapping methods to accurately capture them.

We *could* do this with any existing dataset. Figure 5.1 shows the results on some randomly selected MSCOCO images. By comparing the generated saliency maps with the ground-truth obtained from a summation proxy model, we can use any distance metric to measure how well different saliency mapping methods recreate the input. Results on 1000 randomly selected images from MSCOCO (COCO14 validation set) using Hierarchical Perturbation (HiPe, which I propose in Chapter 4), Random Input Sampling for Explanation (RISE) and Extremal Perturbation (ExtP) are shown in Table 5.1 We can see that HiPe narrowly outperforms the other two methods across all metrics.

However, results on these kinds of natural image datasets are difficult to draw more specific conclusions from, because we haven't captured anything specific about the properties of the saliency mapping methods. For this reason, the proxy model test is best used to evaluate saliency

	MAE	MSE	CS
HiPe	**0.19**	**0.06**	**0.92**
ExtP	0.23	0.08	0.81
RISE	0.24	0.09	0.67

Table 5.1: Mean Absolute Error (MAE), Mean Squared Error (MSE) and Cosine Similarity (CS) for 1000 randomly selected images from MSCOCO (COCO14 validation set) and their corresponding saliency maps generated by Hierarchical Perturbation (HiPe), Extremal Perturbation (ExtP), and Random Input Sampling for Explanation (RISE) using the Proxy Model Test.

mapping methods with synthetic data, which we can generate to capture various properties of saliency maps that we care about.

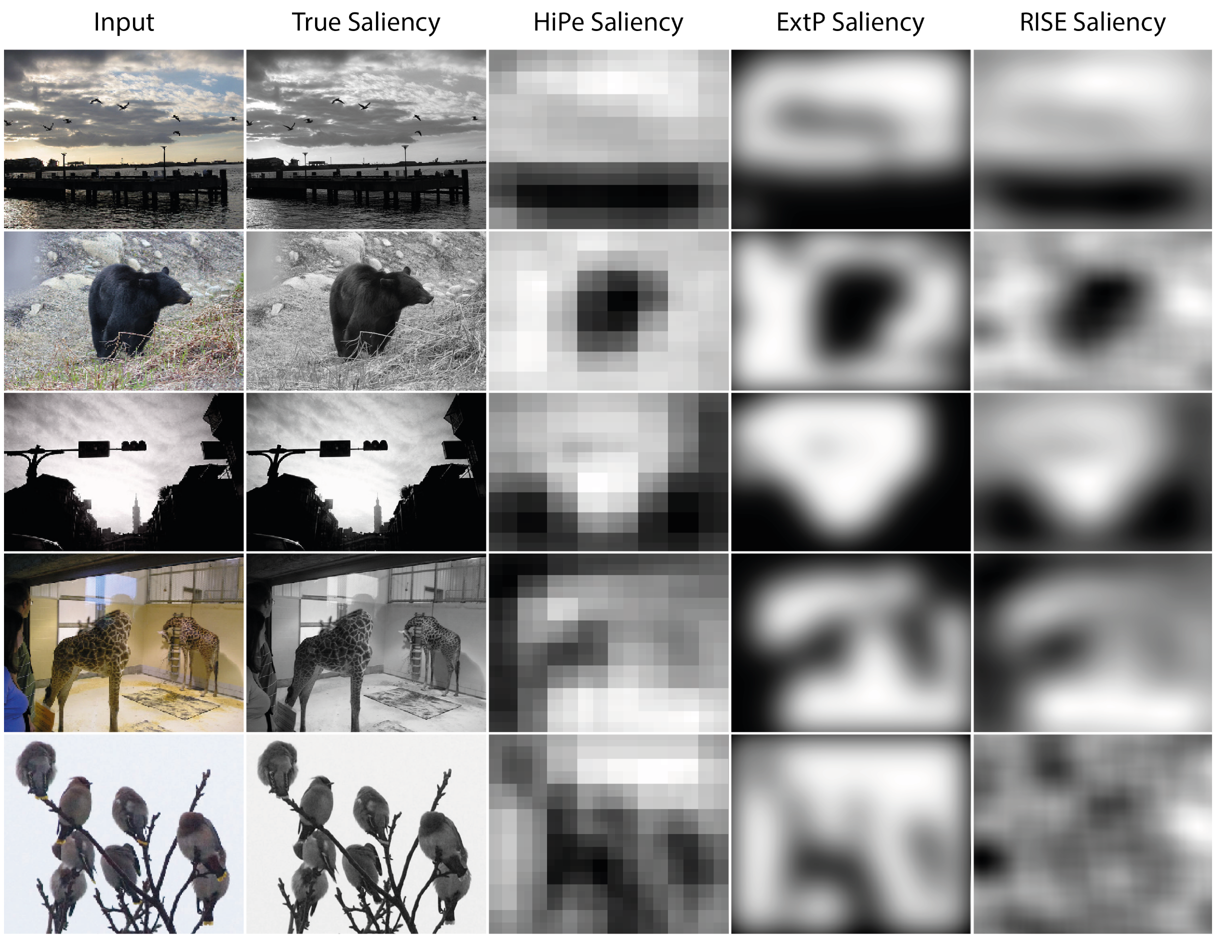


Figure 5.1: Examples of the Proxy Model Test performed on Hierarchical Perturbation (HiPe), Extremal Perturbation (ExtP) and Random Input Sampling for Explanation (RISE) using random images from the MSCOCO (COCO14 validation set) dataset.

5.1 Synthetic Data for the Proxy Model Test

Existing natural datasets are of limited utility when it comes to the proxy model test, and are only really useful as a sanity-check, as shown above. However, we can also generate synthetic datasets, to test and compare specific capabilities and sensitivities of different saliency mapping methods. When developing a saliency mapping method, or selecting one for a particular use-case, we typically want the saliency maps it generates to have a number of properties. For example, perhaps we want particularly fine-grained saliency *within broadly* salient regions, or high sensitivity to very small regions of saliency, or maybe coarse saliency maps are sufficient, as long as they are fast to generate. Using synthetic data with the proxy model test allows us to make these properties and trade-offs explicit, allowing us to choose the best saliency mapping method for the task at hand.

For example, if we have a model that classifies medical scans and wish to provide saliency maps for each prediction it makes (either for use by clinicians in deployment, or during training of the model to check that it is learning to look at sensible features), we might prefer one saliency mapping technique over another depending on our domain knowledge. That is, if the morphology of the disease that the model identifies is concentrated in one region only, we could perhaps select a faster, but coarser saliency mapping method to quickly identify the most salient region. If, however, the presentation of the disease takes the form of multiple lesions all over the input scan, we might select a slower saliency mapping method that is able to identify multiple small salient regions accurately. The proxy model test allows us to objectively compare different saliency mapping methods on data that captures our use-case specific concerns, for example:

Multiple Salient Regions

Different saliency mapping methods can be more or less effective at identifying multiple salient regions in an input. Tests like the pointing game (see Section 3.5) do not capture this property, but if we wish to explain predictions made by a model that is likely to use a number of spatially separate features in the input, we probably want our saliency mapping method to capture those separate regions reliably. We can evaluate this by generating a synthetic dataset with samples that contain a range of distinct salient regions.

Precision and Recall

To test for high precision in saliency maps, synthetic data can be generated with clearly defined, small salient regions against a uniform background. This would challenge the saliency mapping method to pinpoint the exact location of saliency without spilling over into non-salient regions. For high recall, the synthetic dataset might include inputs with subtle salient features that are widely distributed, or many different salient regions. This setup would test the method's ability to detect all relevant features without missing any, even if some regions are less pronounced than others.

Granularity

Some applications may require fine-grained saliency maps where the level of detail is paramount, to validate that the model has learned to use sensible features for prediction in high-stakes domains such as cancer detection. Others may be satisfied with coarse representations that provide a general indication of salient regions, if other constraints such as time are more important. For scenarios where a broad understanding of saliency is sufficient, such as in initial stages of content filtering, synthetic images with large, distinct salient areas may be generated. This would evaluate the saliency mapping method's efficiency in quickly providing an overall picture of saliency without the need for detail.

Salient Region Size

Some methods may perform well on large salient regions but fail to detect smaller ones. By creating a synthetic dataset with images that contain very small but critical salient regions, the saliency mapping methods can be evaluated on their sensitivity. The dataset can be varied with a range of sizes to see at which point the methods begin to fail.

Saliency Variation

In complex visual tasks, not all salient features contribute equally to the final decision. Some areas might be crucial while others less so, but still relevant (and so possibly still of interest). This hierarchical importance is significant in domains like scene recognition, where certain objects may be more critical than others, but where many contribute the model's prediction – for example, beach umbrellas, sea, and sand are likely all salient to a scene classifier predicting 'beach', but may contribute different amounts to the ultimate prediction. If we wish to identify these different contributions, we must use a saliency mapping method that is capable of capturing this. To evaluate this, synthetic datasets can be designed with salient features that have varying levels of intensity or contrast. The saliency mapping method's ability to differentiate between these levels and appropriately rank the salient areas can then be assessed. For instance, in an image, a few small, highly salient objects could be surrounded by larger, less salient ones to mimic the varying importance of visual features in a scene.

Shape

Real-world data often contains salient features of various shapes and sizes, from elongated roadways in navigation tasks to irregularly shaped tumours in medical imaging. Due to the use of differing masks, kernels etc in saliency mapping methods, they may vary in their ability to identify differently shaped salient regions. Synthetic datasets with a variety of geometric shapes or irregular patterns can be created to test how well a saliency mapping method can adapt to the diversity of shapes. These datasets would contain inputs with elliptical, linear, and free-form salient regions to challenge the method's flexibility in shape recognition.

Texture

Sometimes, saliency is not just about color or shape but about texture. Certain applications, such as material sorting in recycling, require identifying textures to determine saliency. A synthetic dataset could include images with varying textures where saliency is defined by a particular texture pattern. The ability of the saliency mapping method to detect salient regions based on texture rather than color or shape can thus be evaluated.

By creating synthetic data that specific real-world requirements and employing the proxy model test in this way, we can evaluate and select saliency mapping methods that are best suited for a particular task, with a level of granularity that isn't possible with existing saliency mapping evaluation methods, such as the pointing game and insertion/deletion metrics (described in Sections 3.5 and 3.6 respectively).

In the next section, I demonstrate the use of the proxy model test with synthetic data to characterise some properties of model-agnostic saliency mapping methods.

I created a synthetic dataset consisting of 1000 inputs each of size 224×224, each containing between 1 and 10 non-overlapping salient regions. These regions are each square, and *in total* for each sample occupy between 10 and 80 percent of the total input – e.g., for a sample where the total salient region size is 50%, this could consist of five separate salient areas each covering approximately 10% of the input, or two each covering 25%, or a single salient region covering 50%. The spatial distribution of these salient regions within each image is randomised. The salient regions are randomly placed such that they do not overlap, with 1000 placement attempts. This results in fewer samples with multiple large salient regions, as we run out of space in the input image. The generation algorithm is described in Algorithm 2 and the distribution of the resulting total salient region coverage (as % of total input) and number of samples per image over the dataset is described in Figure 5.2.

Salient regions are set to 1 (with the rest of the input set to zero), as shown in Figure 5.3.

This dataset is designed to measure the following properties of saliency mapping methods: sensitivity to the amount of the images that is salient, and the number of salient regions (so, both the proportion of the input that is salient, and how many distinct salient regions it is split between). I chose the input dimension of 224×224 to match the input dimensions used in previously published evaluations of saliency mapping methods (see Section 4.1 for further details), as the default settings for these methods are optimised for input of this dimension.

Algorithm 2 Synthetic Data Generation

```
Require: num_samples, dim, min_coverage, max_coverage, max_salient_regions
  Initialise empty list for samples, salient region sizes, and number of salient regions.
  num_covg ← num_samples / max_salient_regions
  Generate coverage values from min_coverage to max_coverage evenly spaced for num_covg values.
  Generate salient region counts from 1 to max_salient_regions.
  for each coverage in coverage values do
      for each count in salient region counts do
          Initialise a base matrix of zeros with dimensions dim.
          Calculate region size based on coverage and count.
          for each region up to count do
              Initialise start positions and attempts counter.
              while start positions are invalid or region overlaps do
                  Randomly choose start positions within bounds.
                  Increment attempts counter.
                  if attempts exceed maximum allowed then
                      Break the loop.
                  end if
              end while
              Set the region in the base matrix to 1.
          end for
          Append the base matrix to samples.
          Calculate and append the mean of the base matrix to salient region sizes.
          Append the count to the number of salient regions.
      end for
  end for
  return samples, salient region sizes, num_salient_regions
```

5.2 Experiments

Using the synthetic dataset described in the previous section, I compare Hierarchical Perturbation (HiPe, proposed in Chapter 4), Extremal Perturbation (ExtP) and Random Input Sampling for Explanation (RISE) (both introduced in Section 3.2), plus a random baseline. For ExtP and RISE, default parameters are used as in the experiments described in Section 4.1. This experiment was performed using Google Colab hardware consisting of an NVIDIA A100 with 80GB RAM.

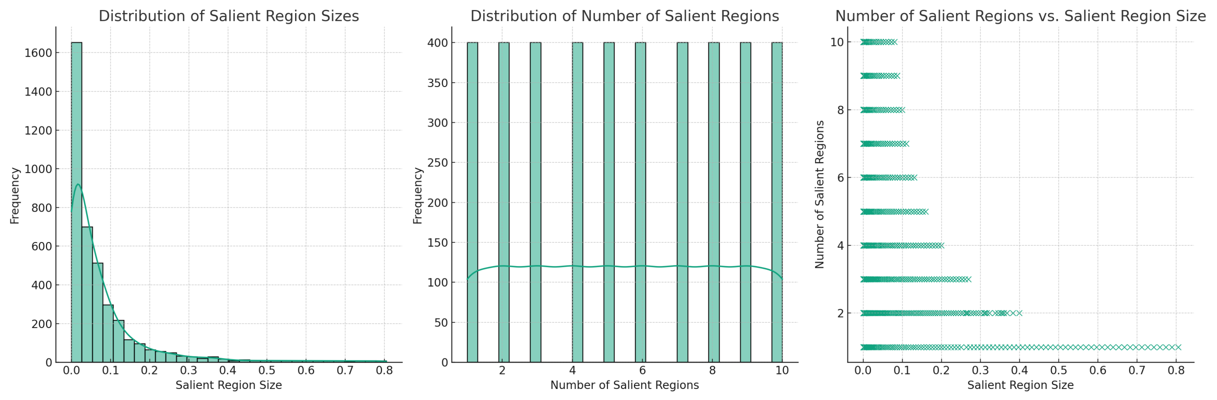


Figure 5.2: Plots describing the distribution of the two key properties of the synthetic dataset, salient region size (as % of total input) and number of salient regions.

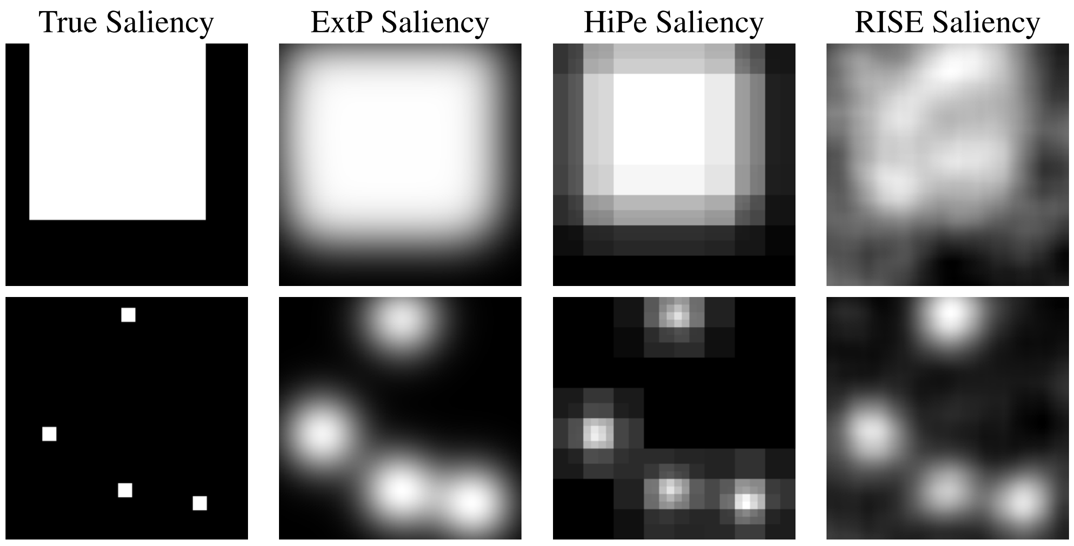


Figure 5.3: Randomly selected examples of synthetic data used for the Proxy Model Test, and the saliency maps generated for them using Hierarchical Perturbation (HiPe), Extremal Perturbation (ExtP) and Random Input Sampling for Explanation (RISE).

Evaluation Metrics

To quantify the performance of saliency mapping methods using the proxy model test on synthetic data, the pointing game framework as described in Section 3.5 could be used, if identification of the *most* salient region is of more importance than overall accuracy – but because the proxy model test provides pixel-level ground truth, more fine-grained metrics can be employed that more accurately capture different properties of the method. I use a number of metrics to capture different aspects of accuracy in evaluating saliency maps: the Mean Absolute Error (MAE); Mean

Squared Error (MSE); Cosine Similarity (CS); Precision (P), Recall (R) and F1, as described below. For saliency maps P and ground truths G of $n \times m$ dimensions:

$$\text{MAE} = \frac{1}{n \times m} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} |P_{ij} - G_{ij}|$$

$$\text{MSE} = \frac{1}{n \times m} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (P_{ij} - G_{ij})^2$$

For calculation of the cosine similarity, the saliency maps are first flattened into vectors $\mathbf{p}$ and $\mathbf{g}$ of length $n \times m$:

$$\text{CS} = \frac{\mathbf{p} \cdot \mathbf{g}}{\|\mathbf{p}\| \|\mathbf{g}\|} = \frac{\sum_{i=0}^{n \times m-1} \mathbf{p}_i \cdot \mathbf{g}_i}{\sqrt{\sum_{i=0}^{n \times m-1} \mathbf{p}_i^2} \cdot \sqrt{\sum_{i=0}^{n \times m-1} \mathbf{g}_i^2}}$$

MAE provides an understanding of the absolute differences between the predicted saliency map and the ground truth, without giving higher weight to larger errors, thus offering a straightforward measure of overall difference. MSE squares the errors before averaging, penalizing larger discrepancies more heavily than smaller ones. CS measures the cosine of the angle between the predicted and ground truth vectors of flattened saliency maps, focusing on the orientation rather than the magnitude – so assessing how closely the predicted saliency map follows the pattern or structure of the actual saliency. Unlike metrics that focus on the magnitude of errors, CS emphasizes the similarity in the spatial distribution and relative importance of salient features.

Precision and Recall are metrics often used to evaluate the performance of classification systems, and in the context of saliency maps, they can be adapted to measure the accuracy of the salient regions detected by the saliency mapping method against a ground truth, which is not usually available, but which the proxy model provides.

Ground Truth refers to the actual salient regions in a synthetic dataset, while Predicted Map (PM) is the output from the saliency mapping method, highlighting the areas it deems salient. To calculate the Precision and Recall of a saliency map, we need:

- True Positives (TP): these are the pixels that are correctly identified as salient by the saliency mapping method (i.e., they are salient in both the GT and the PM);
- False Positives (FP): the pixels that the saliency mapping method incorrectly identifies as salient (i.e., they are not salient in the GT but are salient in the PM);
- False Negatives (FN): the pixels that are salient according to the GT but were missed by the saliency mapping method (i.e., they are salient in the GT but not in the PM).

Precision is then calculated as the fraction of predicted salient pixels that are truly salient:

$$\text{Precision} = \frac{TP}{TP + FP}$$

This measures the accuracy of the saliency map in identifying salient regions. High precision means that the saliency map has a high rate of true salient predictions out of all salient predictions it made.

Recall is the fraction of actual salient pixels that were identified correctly:

$$\text{Recall} = \frac{TP}{TP + FN}$$

This measures the completeness of the saliency map in identifying salient regions. High recall means that the saliency map is able to identify most of the true salient regions in the ground truth.

Both precision and recall are important: precision of 1 shows that everything the saliency mapping method identifies as salient is actually so, while recall of 1 shows that the method captures *all* salient regions that are present. Precison and recall are often combined into a single metric called the F1 score, which is the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.3 Discussion

Method	Time	MAE	MSE	CS	Precision	Recall	F1
ExtP	3.4670	0.2408	0.1235	0.7028	0.1659	**0.7786**	0.2454
HiPe	0.1978	**0.1420**	**0.0536**	**0.8906**	**0.2429**	0.7440	**0.3182**
Random	**0.0005**	0.5000	0.3334	-0.0001	0.0696	0.4992	0.1040
RISE	0.4700	0.2756	0.1174	0.7328	0.1582	0.7606	0.2256

Table 5.2: Mean Absolute Error (MAE), Mean Squared Error (MSE), Cosine Similarity (CS), precision, recall and F1 score between predicted saliency and ground truth saliency of synthetic dataset. For time, MAE and MSE lower is better, for other metrics higher is better. Note poor precision from all methods.

Table 5.2 shows these metrics over the three different saliency mapping methods, plus a random baseline.

HiPe emerges as the most proficient overall method, with MAE, MSE, cosine similarity scores of 0.14, 0.05, and 0.89. These metrics collectively suggest that HiPe is quite accurate in approximating the ground truth, with minimal errors and a strong alignment in the spatial distribution of salient features. RISE and Extremal Perturbation (ExtP) also show good performance on most metrics, and both exceed HiPe performance on recall. However, all methods show notably poor precision, with the highest score (from HiPe) being just 0.24. This indicates numerous false positives (where pixels that are **not** salient are falsely highlighted).

Figure 5.4 shows the same metrics in box plot format, allowing us to understand the spread of performance on each metric. All methods (except the random baseline) show wide spreads with many outliers, suggesting that the performance of these saliency mapping methods is actually quite dependent on the size and number of salient regions in the input. It seems there is further work to be done, either in terms of tuning the hyperparameters of these methods to better suit the task, or in developing more *reliable* saliency mapping methods overall. For cases where fine-grained, precise saliency maps are of particular importance (such as AI for medical diagnostics), none of these methods are reliable in their current form.

Further investigation, shown in Figure 5.5 shows that precision is correlated with the size of the salient regions, with far worse performance on small salient regions across all methods.

This is also evident upon visual examination of samples from the synthetic dataset, as shown in Figure 5.3, where we can observe that the predicted salient regions from all methods extend outside of the true salient regions. Figure 5.6 shows that precision also appears to correlate with the number of salient regions present in an input. However, because of the way the synthetic data is generated (described in Section 5.1) we know that these two properties are not independent, as inputs where the salient regions are small have room for more of them (see Figure 5.2).

Figure 5.7 therefore shows precision by salient region size for inputs with a single salient region only, demonstrating that as the proportion of the input that is salient to the model increases, so does the precision of the saliency mapping methods.

These results provide evidence that in cases where we a single region is salient, and takes up more than 20% of the input, we can expect HiPe to perform with 75% precision or higher. RISE and ExtP are not far behind, but take approximately $2\times$ and $10\times$ longer to run respectively (as shown in Table 5.2).

Unlike RISE and ExtP, the time taken to generate a saliency map with HiPe varies depending on the distribution of salient pixels in the image. Because, by using a proxy model, we have access to information about the true saliency distribution, we can test this empirically. Note that this is at best an approximation of HiPe's performance on real-world data and models, as the true saliency distributions are likely to be much more complex. Figure 5.8 shows how the time taken to generate saliency maps on synthetic data varies with the size and number of salient regions. We can see that, as expected, time increases with salient region size (as there are more salient regions that require exploring).

In summary, this chapter demonstrates how the proxy model test can be used in conjunction with constructed synthetic datasets to identify strengths and weaknesses of arbitrary saliency mapping methods, with respect to specific use cases. The results underline the utility of the proxy model test for more comprehensive evaluation of saliency mapping methods – if, for a specific domain, high-precision saliency maps that capture multiple salient regions are necessary, none of the methods here are well-suited. This might be improved with post-processing of the saliency maps to remove lower-saliency pixels, or by tuning their hyperparameters for this specific task. For other use cases, synthetic data can be constructed to capture desirable properties of saliency mapping methods *for that particular use case*. More generally, the Proxy Model Test provides a flexible quantitative evaluation metric for arbitrary model-agnostic saliency mapping methods, without requiring human data annotation and the assumptions inherent therein (as discussed in Section 3.4).

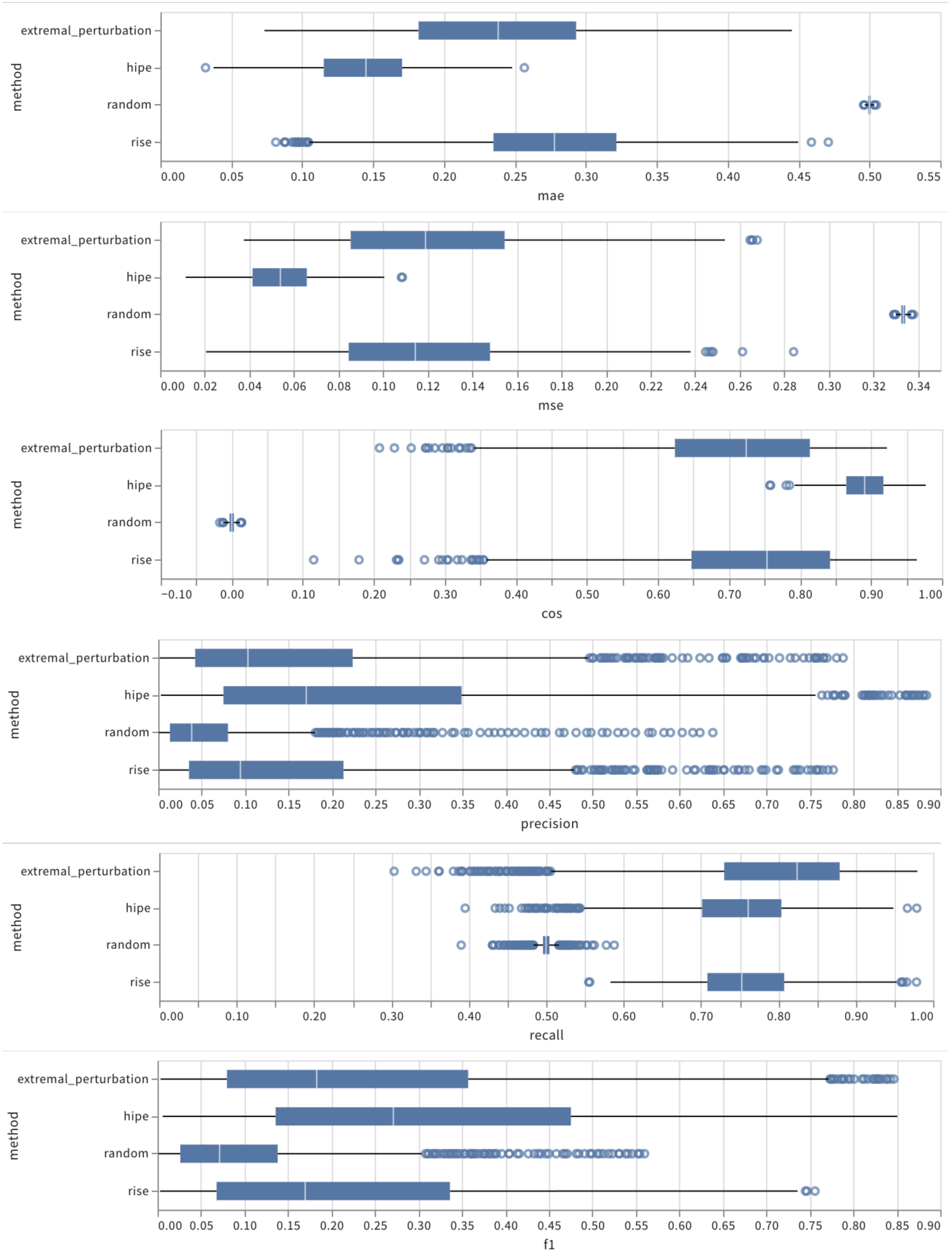


Figure 5.4: Results on synthetic dataset. For mean absolute error (mae) and mean squared error (mse) lower is better, for cosine similarity (cos), precision, recall and F1 score higher is better. Note high variance of performance across all methods, and poor precision over all.

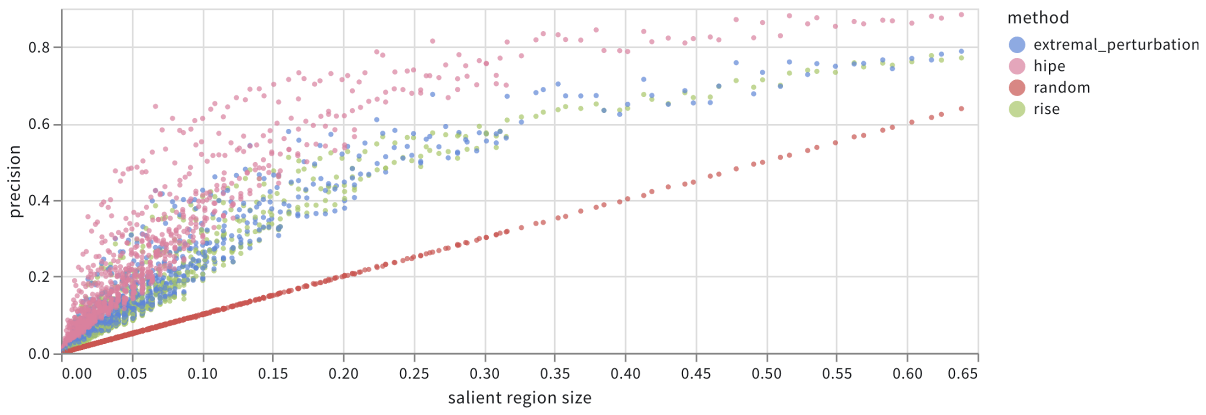


Figure 5.5: Saliency map precision by salient region size, across all salient region counts.

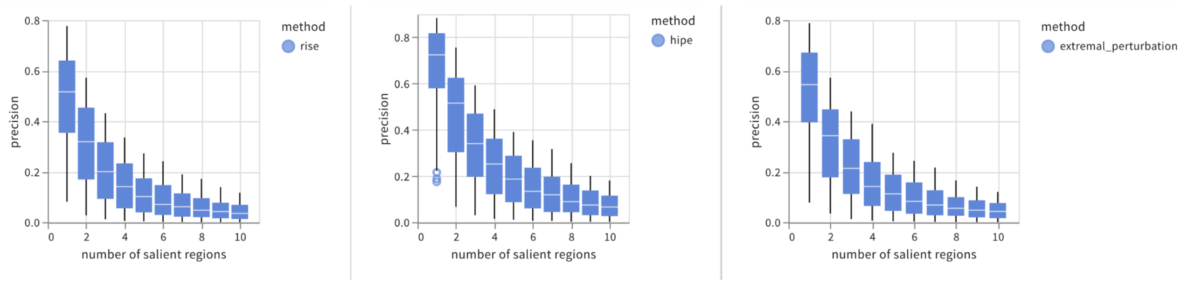


Figure 5.6: Saliency map precision by number of salient regions, across all salient region sizes.

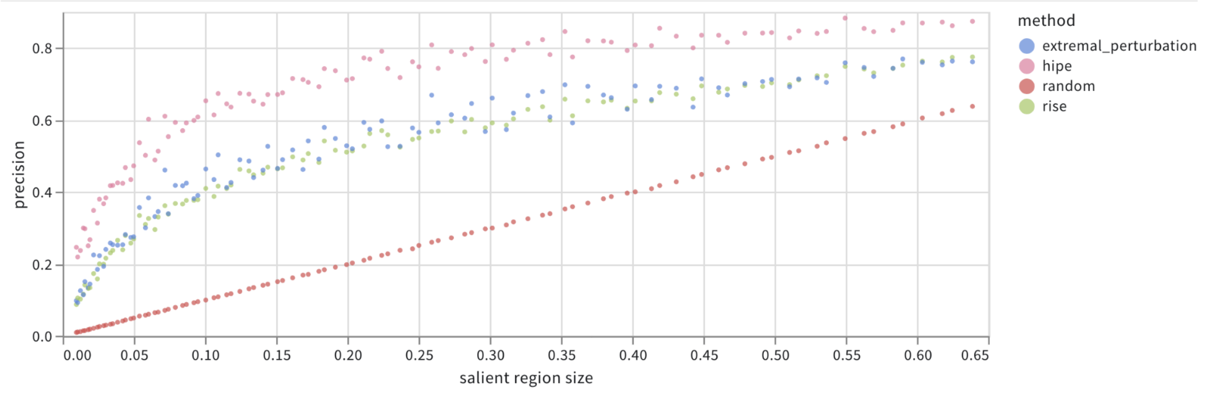


Figure 5.7: Precision by salient region size for inputs with a single salient region only.

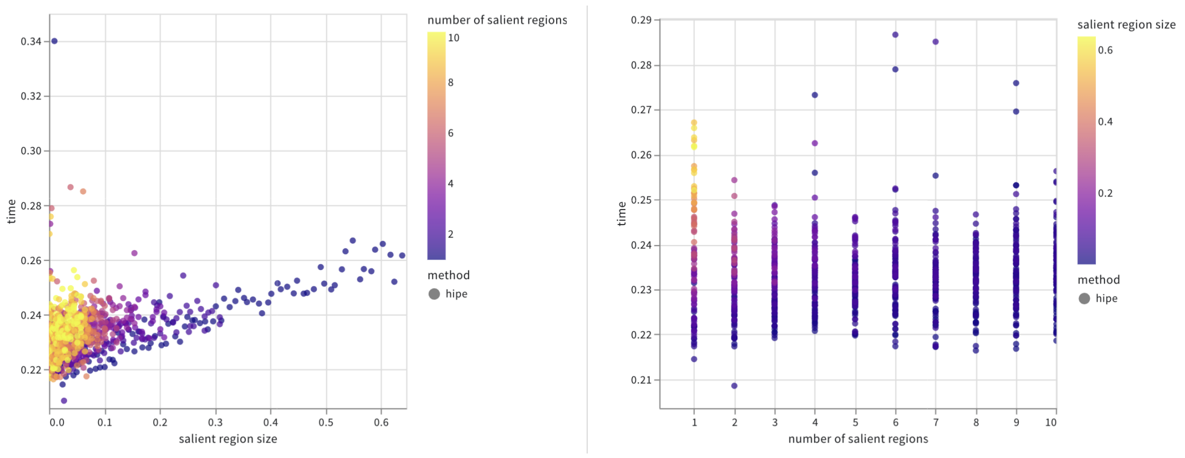


Figure 5.8: Time taken by HiPe to generate saliency maps, over different salient region coverage % and distinct salient region counts. This experiment was performed using Google Colab hardware consisting of an NVIDIA A100 with 80GB RAM.

Chapter 6

Weakly Supervised Saliency Segmentation

As introduced in Chapter 1, digital pathology has emerged as a promising alternative to traditional microscopy for disease diagnosis, leveraging whole slide imaging and computerised analysis methods [51]. Deep learning, especially convolutional neural networks (CNNs), have been applied to applications ranging from nuclei segmentation, to tissue classification, disease detection, and survival prediction [87, 161]. However, state-of-the-art deep learning algorithms often require large-scale datasets with detailed per-pixel annotations for training, which are expensive and time-consuming to collect in the medical domain [19] – and so much recent work focusses on developing weakly supervised approaches that do not require detailed labels.

In this chapter and the following one I consider two case studies in applied deep learning for digital pathology. I also show how the interpretability methods proposed in earlier chapters can be applied to improve human understanding of model behaviour. I discuss interpretability in the context of weakly supervised WSI classification, taking an existing model trained using weakly supervised data and applying feature visualisation to explain the predictions made and identify distinct morphologies learned by the model to differentiate between tissue classes. I also demonstrate how saliency mapping can be used to generate very detailed pixel-level saliency segmentations. Then, in Chapter 7, I train a residual neural network to identify a specific immune cell subtype from Hoechst-stained slides only – a task that was previously not thought to be possible. I do this using both classification and segmentation paradigms, and then demonstrate how interpretability methods can be used for knowledge discovery by identifying the trained model's learned class prototypes in both cases.

Semantic Segmentation

One application of deep learning in digital histopathology (and in computer vision in general) is semantic segmentation – that is, given some input image, producing class labels for each pixel in that image. For example, given a set of biopsy slides which have been divided into cancerous and non-cancerous regions by a human annotator, one could train a neural network model to label each pixel in the image as cancerous or non cancerous [155, 78, 80, 33]. This is distinct from classification tasks, in which a single label is provided for an entire slide or region, and pixel-level labels are not required (see Figure 4.5).

Convolutional and Fully-Convolutional Neural Networks (CNNs & FCNNs)

Traditional CNNs, used in segmentation tasks, typically consist of convolutional layers followed by fully connected layers. Each convolutional layer applies a series of filters to extract features from the input image, and the fully connected layers perform classification based on these features (see section 1.2). In segmentation tasks, the output layer is modified to produce a pixel-wise classification. In digital pathology, CNNs are used to differentiate between various tissue types, identify disease markers, and segment regions of interest within histopathological images, and can be adapted to a wide range of segmentation tasks in digital pathology. However, the pooling layers in CNNs can lead to a loss of resolution. The U-Net architecture is a notable example [129] of this approach. It consists of a contracting path to capture context and a symmetric expanding path for precise localisation, resulting in a 'U' shaped architecture – more on this in the next chapter.

Fully Convolutional Networks (FCNs) are a modification of CNNs where fully connected layers are replaced by convolutional layers [89]. This architecture enables the network to directly output spatial maps instead of classification scores, allowing for native pixel-wise segmentation.

FCNs are particularly suited for segmenting images in domains where fine-grained segmentation is important – by retaining spatial information throughout the network, they can achieve more precise segmentation compared to traditional CNNs.

Generative Adversarial Networks (GANs)

GANs consist of two competing neural network models: a generator and a discriminator (see also section 1.2). The generator creates images (or segmentations) that mimic the real data, while the discriminator evaluates them against actual data. For semantic segmentation, the generator learns to produce segmented images, and the discriminator assesses whether the segmentation is real (ground truth) or fake (generated). This adversarial process continues until the generator produces segmentations indistinguishable from real annotations[145, 146].

GANs can generate highly detailed and accurate segmentations, and can also be used for generating synthetic training data. Unfortunately, GANs are notoriously difficult to train due to the delicate balance required between the generator and discriminator, and often suffer from mode collapse (where the generator produces limited varieties of output, reducing the diversity of the segmentation) [134]. Despite their ability to generate data, the effectiveness of GANs still heavily relies on the availability of high-quality ground truth for initial training.

The above approaches all require pixel-level semantic segmentations for training, which are costly and time-consuming to obtain in digital histopathology – hence the widespread interest in weakly-supervised methods.

Weakly Supervised Learning in Digital Pathology

Fully supervised learning (i.e., at the pixel or region level) is often not possible due to the considerable cost and time investment necessary for experts to manually annotate each slide. Weakly supervised learning approaches relieve a large part of this burden by either reducing the number of slides required to effectively train a predictive model, or by removing the necessity of pixel or region level labels. However, predicting only slide level labels decreases the amount of explicit information available to clinicians when considering the model's prediction, which is undesirable in this medical context where understanding *what* in the input influenced a classification is crucial – hence the need for interpretability. Weak supervision takes various forms, from patch or bounding-box labels to whole slide or image labels, and different approaches have been proposed to enable learning predictive models from this data.

Transfer Learning

Transfer learning involves leveraging a pre-trained model on a large, annotated dataset from a different domain where data acquisition is less costly, and fine-tuning it for the target domain on limited labeled data. Sharma et al. demonstrated this approach by adapting a models pre-trained on natural images for histopathology tasks[140]. The goal is for the pre-trained network to adapt to the specifics of histopathological imagery, significantly reducing the need for large annotated datasets in the target domain, by using the generic feature-detection capabilities of models trained on more extensive datasets. However, the success of this approach relies on the similarity between the source and target domains. When features in the target domain differ substantially from the features present in the pre-training dataset, the effectiveness of transfer learning may be limited.

Semi-Supervised Learning

Semi-supervised learning is a hybrid approach that uses a combination of labeled and unlabeled data for training. In pseudo-labeling, a model's own predictions are used to augment the training set [96]. In digital pathology, this involves initially training a model on a small set of annotated slides and then using it to label a larger, unlabeled set. Pseudo-labeling effectively utilises large volumes of unlabeled data, which is particularly useful in digital pathology due to the abundance of such data – however, the quality of pseudo-labels is crucial. Inaccurate labels can reinforce errors, leading to model drift.

Consistency regularisation is another semi-supervised technique that works by encouraging the model to produce consistent predictions for perturbed versions of the same input[148, 22]. In pathology, this could involve applying transformations (like rotations, shifts, or changes in saturation) to slide images, ensuring that the model's predictions are invariant to such changes. This approach forces the model to learn robust features that are invariant to minor variations, a desirable trait in histopathological analysis – but if the perturbations are not carefully chosen, they might obscure important features or create confounding artefacts.

Multiple-Instance Learning (MIL)

MIL is a form of weakly supervised learning where labels are provided for bags (collections) of instances (e.g., patches) rather than individual instances. In digital pathology, a whole slide image (WSI) is treated as a bag, with patches as instances.

MIL approaches typically involve aggregating instance-level features to predict the bag-level label[32]. Attention-based MIL further refines this by weighing instances differently, highlighting areas most indicative of the pathology [69]. Multiple instance learning (MIL) is a weakly supervised approach that assigns a single label to a group of instances, e.g. patches from a whole slide image. The original MIL algorithm was restricted to binary classification, assuming that if at least one patch is positive, the whole slide is positive, while if all patches are negative, the slide is negative. This assumption leads to using the maximum patch probability for the slide-level prediction. However, this rigid aggregation function limits MIL for multiclass and more complex binary tasks [69]. Recent works have developed more flexible aggregation methods for better accuracy, such as attention-based MIL [69], which uses a weighted average of patch predictions.

Other MIL aggregation methods include ranking patches by probability [32] and passing top patches sequentially to an RNN for the slide prediction. Weakly supervised approaches have also been proposed using patch-based networks and context-aware feature selection [162]. The clustering-constrained attention MIL method resolves limitations of earlier approaches for multiclass subtype prediction [91]. The model used in this chapter employs a clustering-constrained attention MIL approach, which extends MIL for multiclass subtyping[91]. While state-of-the-art performance in this domain still requires full supervision, weakly supervised methods have shown competitive performance for realistic triage settings.

Attention Mechanisms

Attention mechanisms are typically used to enhance model performance by enabling higher weighting of relevant parts of the input data, rather than treating all data uniformly. However, including attention mechanisms typically increases the complexity of the network, which can lead to longer training times and higher computational requirements. Attention can be integrated into models in various ways, to selectively emphasize important features and suppress irrelevant ones [167], achieved through mechanisms like spatial attention (focusing on specific regions) and channel attention (focusing on relevant feature channels) [115]. A notable example in the medical domain is the Attention U-Net proposed by Oktay et al., which incorporates attention gates into the U-Net architecture [116].

Although attention is primarily a mechanism to improve model performance, it is often considered useful for interpretability – but attention maps differ from saliency maps in a few key ways. Attention is an explicit mechanism in certain neural architectures, to enable models to weight specific parts of a larger input more highly. In contrast, saliency maps are generated using external interpretability techniques that assign importance values to input features post-hoc. The aim is to *explain* model predictions by highlighting relevant parts of the input, rather than modeling attention explicitly. Recent work (in language [70, 18], where attention mechanisms are predominantly employed, and in image classification [120, 88]) has shown that attention maps are often unreliable as interpretability methods, as imposing random different attention scores on a model can lead to the same prediction. It is possible that this is due to 'combinatorial shortcuts' [18], in which the attention weights are used by the model to express extra information to be used by downstream layers – it could be that attention mechanisms don't do what we assume they do. A key work exploring this concern with image models by Liu et al demonstrates that attention often incorrectly identifies the direction (positive or negative) of a feature's impact on predictions: higher attention weights do not reliably mean features are contributing to, rather than suppressing, model outputs [88]. This calls into question the faithfulness of attention-based explanation techniques in representing true influences – contrary to common assumptions, higher attention on features does not necessarily imply higher positive attribution regarding their role in predictions – and has lead to calls for renewed focus on saliency mapping methods as more reliable alternatives to attention for the purposes of model interpretability [23, 26].

6.1 Model and Data

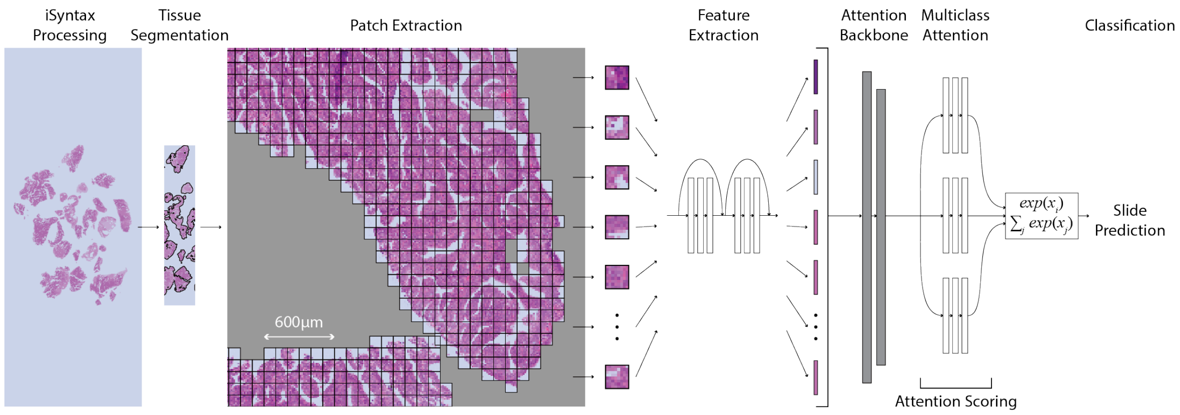


Figure 6.1: Weakly supervised Whole Slide Image classification using Clustering-constrained Attention Multiple instance learning (CLAM) [91].

Dr Mahnaz Mohammadi of the School of Medicine at St Andrews kindly provided the model architecture and learned parameters that I used for this experiment. Full details of the data collection and training protocol for this model are available in the published work [99]. The model is based on the Clustering-Constrained Attention Multiple Instance Learning (CLAM) method proposed by Lu et al. [91], discussed above and illustrated in Figure 6.1. This model was trained to classify WSIs as either 'Malignant', 'Other/Benign', or 'Insufficient' with over 87% test accuracy on a dataset consisting of around 3000 endometrial WSIs, obtained from four hospitals in Glasgow, Scotland.

The images vary in size, resolution, tissue thickness and staining laboratory, and also vary in colour, as H&E staining intensity and hue varies across different labs. No colour normalisation was applied to ensure that the trained model would generalise to unseen hue and intensity in the test set, which was from a further lab. Additionally, each slide is from a different patient to ensure robustness to inter-patient application of the trained model. Due to all of these factors, the dataset exhibits a wide variance. The slides were labelled by one of four participating pathologists, who assigned a slide-level classification to each:

1. Malignant: Slides containing any endometrial carcinoma, carcinosarcoma, uterine sarcoma, or endometrial hyperplasia with atypia).
2. Insufficient: Slides which do not contain sufficient tissue to make a diagnosis, often containing blood, mucus, and very small amounts of tissue.
3. Other/Benign: All other slides, typically containing menstrual/shedding endometrium, inactive/atrophic endometrium, evidence of hormonal change, proliferative endometrium or secretory endometrium.

6.2 Saliency Segmentation

As discussed above, a considerable problem in machine learning from WSIs is the lack of pixel-level annotations available for training. Classifying slide regions requires many hours of expert annotation, which is often prohibitively costly and time-consuming – hence the widespread interest in weakly supervised approaches.

One way of obtaining class-specific segmentation in this context, without the need for human annotators, would be by inspecting the attention apportioned to each patch by the attention backbone component of the classifier model. This approach may be helpful for validating model predictions, as it enables easy inspection of smaller regions and insight into which areas were more or less important in producing the final slide prediction. However, it can tell us little about smaller learned features present within patches, other than whether they are present or not, and can produce only patch-level segmentations. Additionally, questions have been raised about the trustworthiness of using attention map visualisation for interpretability, as discussed in section 6. In order to create more detailed and reliable saliency maps to aid in understanding and validating the model's behaviour, I here demonstrate the use of a saliency mapping algorithm to generate pixel-level segmentations, without the need for training a segmentation model on pixel-level annotations (which are typically unavailable).

For this, I use Hierarchical Perturbation (HiPe) as described in Chapter 4, a saliency mapping method which is both model-agnostic and highly computationally efficient. This is necessary in order to mitigate the substantial computational cost of pixel-level attribution on gigapixel input images. Furthermore, given that the classification model is two-stage (consisting of a feature extraction step, followed by a classification network), HiPe's model-agnosticism was key in this choice of saliency algorithm.

As the goal is to generate saliency *segmentation* maps for interpretability purposes, in this chapter I adapt HiPe to focus on perturbing regions not with *higher* saliency as before, but with *lower saliency variance* between classes. This focusses the most fine-grained saliency mapping on the *edges* between regions that are salient to *different* classes.

Thus, I optimise for more detailed mapping in regions where the model is less certain which class is predominant by adapting HiPe to operate over all classes simultaneously, and replacing the standard threshold function:

$$t(\mathbf{S},\mathbf{M}) = \begin{cases} 1, & if \quad \max(\mathbf{S} \circ u(\mathbf{M})) \geq \min\left(\mathbf{S} + \frac{\max(\mathbf{S}) - \min(\mathbf{S})}{2}\right) \\ 0, & otherwise \end{cases} \tag{6.1}$$

with the following:

$$t(\mathbf{S},\mathbf{M}) = \begin{cases} 1, & \text{if} \quad \max(\mathrm{Var}(\mathbf{S}_c \circ u(\mathbf{M}_c)) \text{ for } c = 0,\dots,C-1) \\ & \quad \leq \mathrm{Var}\left(\min\left(\mathbf{S}_c + \frac{(\max(\mathbf{S}_c) - \min(\mathbf{S}_c)}{2}\right) \text{ for } c = 0,\dots,C-1\right) \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

where **S** and **M** have an additional dimension of size C (the number of classes), and Var is the saliency *variance* for that class. This means that the resulting saliency maps are more fine-grained where there is high variance between classes – such as the borders of segmented areas, where a single perturbed region contains more than one class. This results in fine-grained segmentation maps (see Figure 6.2) which identify similar regions to those labelled by experts, as shown in Figure 6.3 (although with some interesting divergences, as I will discuss later).

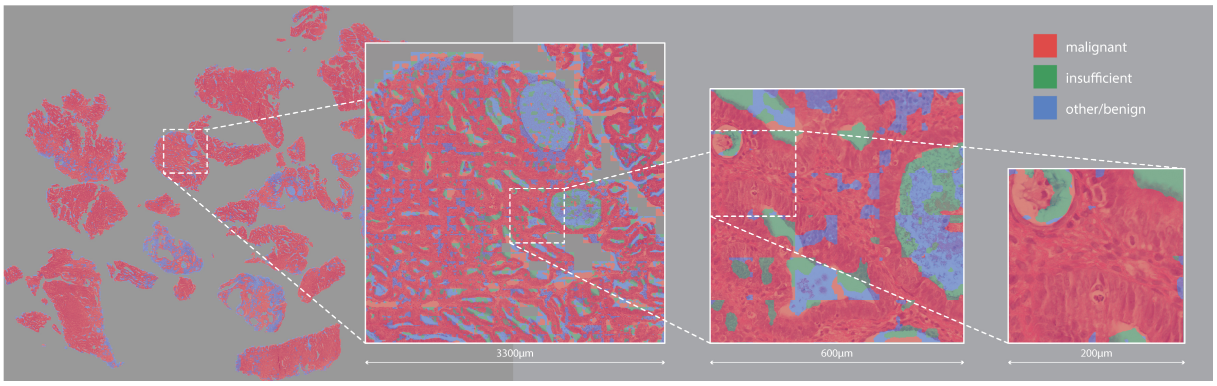


Figure 6.2: High-resolution saliency segmentation of a malignant slide.

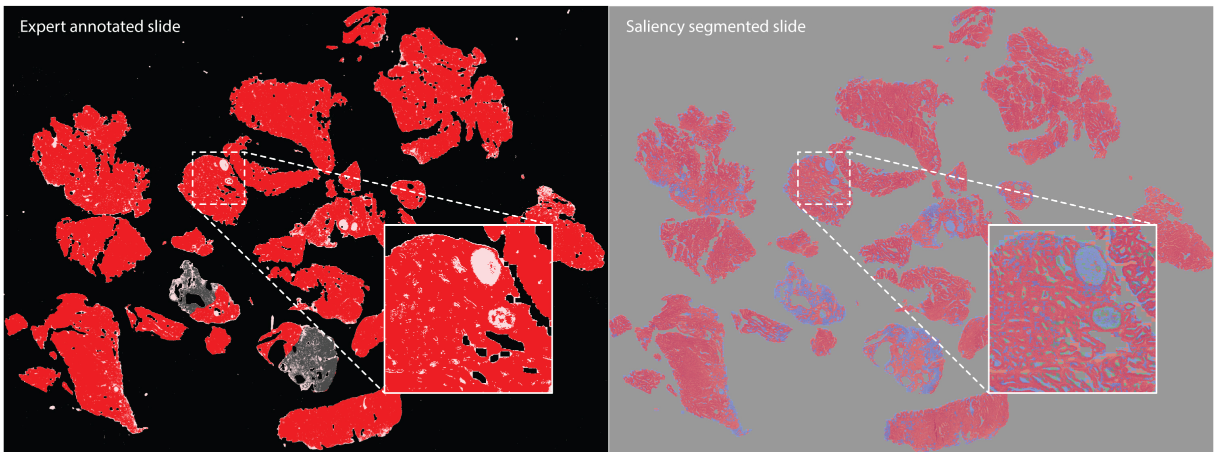


Figure 6.3: Comparison of areas identified as malignant (in red) by a human expert (left) and by the HiPe-based segmentation method (right).

As discussed in Chapter 3.4, evaluating saliency mapping methods on complex models is difficult due to the lack of ground truth – we don't know what the model *actually* finds salient. However, if we hope to use saliency mapping for segmentation, some measure of accuracy must be obtained. Evaluating the accuracy of saliency maps on this kind of data is difficult, as at high resolution, the human expert annotations are actually far coarser than the saliency-based segmentation, so we do not have access to a 1:1 comparator. Similarly to the weakly-supervised slides, any region of tissue may contain cells that are actually benign, yet still labelled malignant by humans, as visible in Figure 6.3.

Conceptually, segmentation is *not* the same as saliency, as there is no guarantee that the model has learned *all* the features that are part of a particular class. For example, if a model was trained to classify images of cats and fish, the only feature it would really *need* to learn is 'fur' – if there's fur in the image, output 'cat', otherwise, output 'fish' (assuming all cats have fur, and all fish don't). If we generated saliency maps for each class with respect to this model, it seems unlikely that those maps would accurately *segment* the cat or the fish. This is borne out by the figures in Chapter 4 (see Figures 4.8 and 4.4 in particular) – what the model finds most salient for a given class prediction hopefully bears some relation to what humans would consider relevant for that class, but the resulting saliency maps don't look much like segmentation. Or at least, this seems to be the case in natural images like the ones used in previous chapters. However, the features present in histopathology images are quite different from those in natural images – possibly more akin to textures, and with potentially less variation within a single class (although this is very domain dependent). I thought perhaps the features learned by models trained on histopathology data would be quite different to those learned by models trained on natural images, which lead me to wonder if saliency maps generated from histopathology models might approximate semantic segmentation far better than one would expect.

Unlike segmentation maps produced by semantic segmentation models, *saliency* maps for models that predict more than one class are typically not mutually exclusive. In saliency mapping, it can be sensible for the same region to contribute significantly to the saliency maps for both class A and class B – for example, if a model trained to classify different scenes was presented with a photograph of a street with a car on it, might assign high saliency to the car with respect to both the 'car park' and 'road' classes – and this is quite reasonable. However, *segmentation* models (and human expert annotators) typically do not do this, and conceptually treat each pixel as being part of one class *or* another. So how should we adapt saliency maps for segmentation purposes?

Post-Processing

Softmax

For classification models, a softmax function is typically used as the activation function of the final layer, in order to convert the raw model outputs to probabilities:

$$\sigma(\mathbf{z})_c = \frac{e^{z_c}}{\sum_{c=0}^{C-1} e^{z_c}} \tag{6.3}$$

where $\sigma(\mathbf{z})_c$ is the softmax function applied to the vector of logits $\mathbf{z}$ for class c, and C is the total number of classes.

We could treat a set of saliency maps (one for each class) as raw model outputs for a given image, and apply softmax over each pixel location to obtain a probability distribution over the classes for each pixel. However, in regions where two (or more) classes have high saliency, this would artificially decrease the prediction for both classes - for example, if pixel **a** has saliency [10,10,0] over three possible classes, and pixel **b** has saliency [0,0,1] for the same three classes, using softmax would result in pixel **a** having scores [0.5, 0.5, 0.0], and pixel **b** approximately [[0.2, 0.2, 0.6]].

Using softmax artificially increases the importance of the first two classes for pixel **b**, which weren't actually salient at all. It also assigns the third class for pixel **b** a value of 0.6, which is lower than that assigned to the first two classes of pixel **a** – even though their saliency values were 10× higher.

Normalisation

Another route would be normalisation, where we subtract the minimum saliency value and then divide the saliency maps by their range (over all classes). For a matrix **X** of size (C,d,d) where

C is the number of classes, and d is the saliency map dimension (assuming a square saliency map), the function normalise($\mathbf{X}$) subtracts the minimum value of $\mathbf{X}$, denoted as $\min(\mathbf{X})$, from each element in $\mathbf{X}$. The resulting values are then divided by the maximum of two values: the difference between the maximum and minimum values of $\mathbf{X}$, denoted as $\max(\mathbf{X}) - \min(\mathbf{X})$, and a small constant ε to avoid division by zero errors. The formula for normalisation is given by:

$$\text{normalise}(\mathbf{X}) = \frac{\mathbf{X} - \min(\mathbf{X})}{\max(\mathbf{X}) - \min(\mathbf{X}) + \varepsilon}. \tag{6.4}$$

Importantly, the normalisation is applied over the entire matrix including all saliency maps, *not* per-map, to ensure relative differences in saliency between classes are retained. This results in the saliency maps for all classes being in the range [0, 1], and proportionately scaled across all classes. However, expert semantic segmentation annotations are binary – a region is typically part of one class or another, without overlap or uncertainty – and the purpose of semantic segmentation is to segment an image into distinct regions.

Max

To get a binary mask for each class, as a human expert annotator would provide, there are a number of possible strategies. One approach is, for each pixel, to set the class with the highest saliency value to 1 and the others to zero. This ensures distinct binary masks with no overlap – but disregards a lot of information. For instance, if two classes have very close saliency values for a given pixel, the one with the larger value will appear artificially inflated, and the other diminished.

Rounding

Another way to enforce distinct segmentation would be to apply a step function to the saliency maps, such as rounding the saliency maps after normalisation. Rounding has similar properties to taking the max, such as providing distinct binary saliency maps for each class – although they may overlap, if more than one normalised class saliency value is greater than 0.5.

Rounding Up

Rounding *up* the normalised saliency map also provides distinct segmentation – for each class, a pixel is either part of that class or is not. However, maps will overlap if more than one class saliency value is greater than 0. Recall, these saliency maps were produced by Hierarchical Perturbation, and so only capture regions that *positively* contribute to a class prediction – so *all* saliency map values are zero or greater in the first place.

This method retains the most saliency information, in that, for a given class, *any* region that contributes to that class positively *at all* will be captured in the saliency map, even if that region *also* contributes to other classes. Despite this issue, I found that rounding up normalised saliency values to produce overlapping, binary maps for each class resulted in very similar maps to expert annotations, at least on this small test set. For visualising these saliency maps, where a pixel is associated with more than one class, I show the class with the highest saliency value.

A comparison of these different saliency map post-processing methods is shown in Figure 6.4. Rounding up produced the most accurate saliency maps when compared to human expert annotations, with the highest recall and F1 scores averaged across classes, at the cost of somewhat lower precision. This makes sense, as rounding up registers every pixel that results *any* increase in a class prediction as part of that class.

6.3 Results

A comparison of saliency-generated and expert-annotated segmentations, using standard segmentation metrics as defined in Equation 7.5, to 261,290 patches (described in Table 6.1) from six 'malignant' slides is shown in Table 6.2 and Figure 6.5. Note that only very tiny regions are labelled 'insufficient' by either saliency-generated or expert-annotated segmentations, so precision, recall and F1 are correspondingly low for that class. These results also show that the saliency-generated segmentations consistently overestimate the amount of benign coverage (as a proportion of the total image, 'saliency %') when compared to expert annotations ('expert %'),

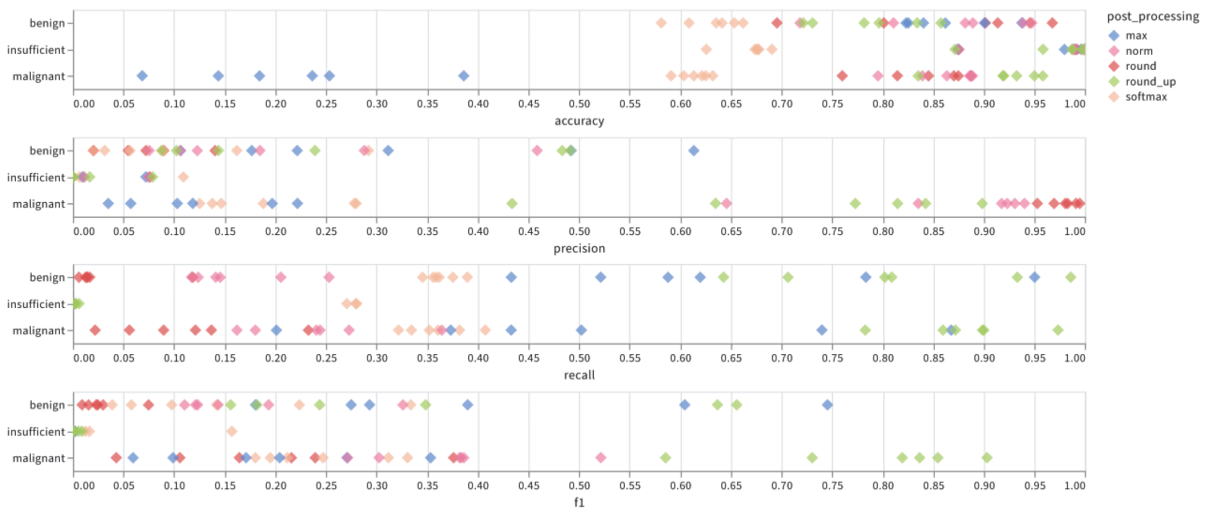


Figure 6.4: Comparing saliency-generated and human expert annotations of ‘benign’, ‘insufficient’ and ‘malignant’ regions in whole slide images, using different saliency map post-processing methods. Note that precision, recall and F1 for the ‘insufficient’ class are very low, due to minimal representation of that class in the images tested.

Slide	**Total Segmented**	**Insufficient**	**Benign**	**Malignant**	**Empty**
Slide 1	12054	0	1196	8820	54299
Slide 2	55329	751	15513	12748	70470
Slide 3	36324	1054	3302	26469	79299
Slide 4	29291	9470	4110	9827	52505
Slide 5	99390	0	27012	26402	46878
Slide 6	28902	0	6046	12564	52781
Total	261290	11275	57179	96830	356232

Table 6.1: Number of patches segmented per slide by (expert labelled) type. 261,290 patches were used in total. Empty patches (those containing no tissue whatsoever) were not used in training the model and were set to zero in the expert annotations. No saliency-based segmentation was performed on them.

but are very close for ‘insufficient’ and ‘malignant’ classes, with 97% and 92% average accuracy respectively.

These results suggest that HiPe-generated saliency maps can be effectively transformed into segmentation maps that closely resemble those drawn by expert pathologists, even without any dedicated training for segmentation. This is a significant finding, as it opens up the possibility of generating accurate segmentations without the need for expensive and time-consuming pixel-level annotations. I found that post-processing of saliency maps was important, with the accuracy of the final results varying widely between different methods. Among the various

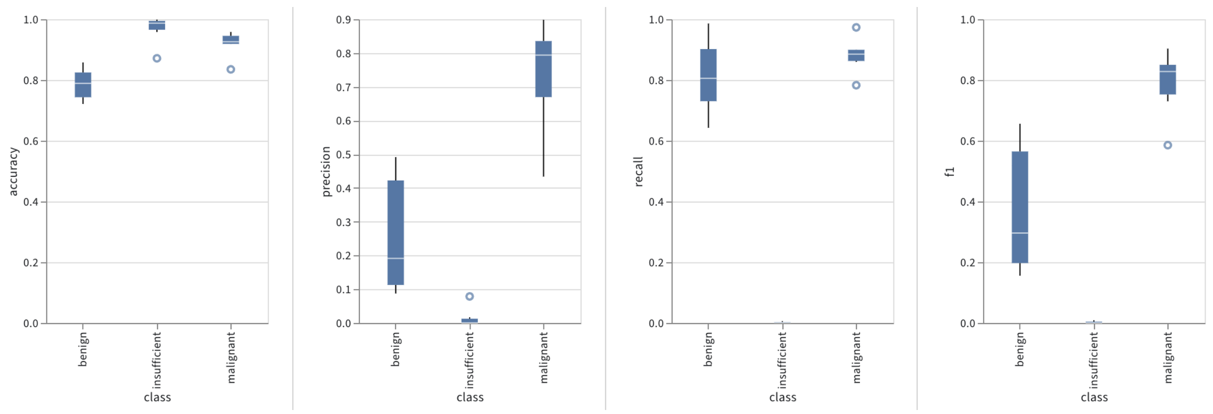


Figure 6.5: Accuracy, precision, recall and F1 scores comparing semantic segmentations obtained from expert human annotators and from HiPe-generated saliency maps, using ‘round-up’ post-processing as described in section 6.2.

class	expert %	saliency %	accuracy	precision	recall	f1
benign	0.10	0.29	0.79	0.26	0.81	0.37
insufficient	0.02	0.01	0.97	0.02	0.00	0.00
malignant	0.18	0.22	0.92	0.73	0.88	0.79

Table 6.2: Comparison of saliency-generated segmentation (with ‘round-up’ post-processing) and human expert annotations, averaged over 261,290 patches from six slides. ‘expert %’ and ‘saliency %’ refer to the portion of the slide image covered by the expert and saliency-generated segmentations respectively.

post-processing techniques tested, rounding up the normalised saliency maps yielded the most accurate segmentations, particularly for the "malignant" and "insufficient" classes. This suggests that capturing even weak positive contributions to class predictions is important for accurate segmentation in histopathology images. However, the saliency-based segmentation consistently overestimated the extent of "benign" regions compared to expert annotations. This discrepancy could be due to several factors, such as the model’s sensitivity to subtle features that human experts deem less relevant for classifying a region as "benign," potential variability in expert annotations, or thresholding effects in the post-processing step as discussed above. Despite this overestimation, the overall accuracy for the "malignant" and "insufficient" classes suggests that saliency-based segmentation maps are a promising approach when automated semantic segmentation is practically useful (for example, to support human experts performing diagnostic tasks, or to provide data for training segmentation models) but pixel-level labels are costly to obtain.

6.4 Discussion

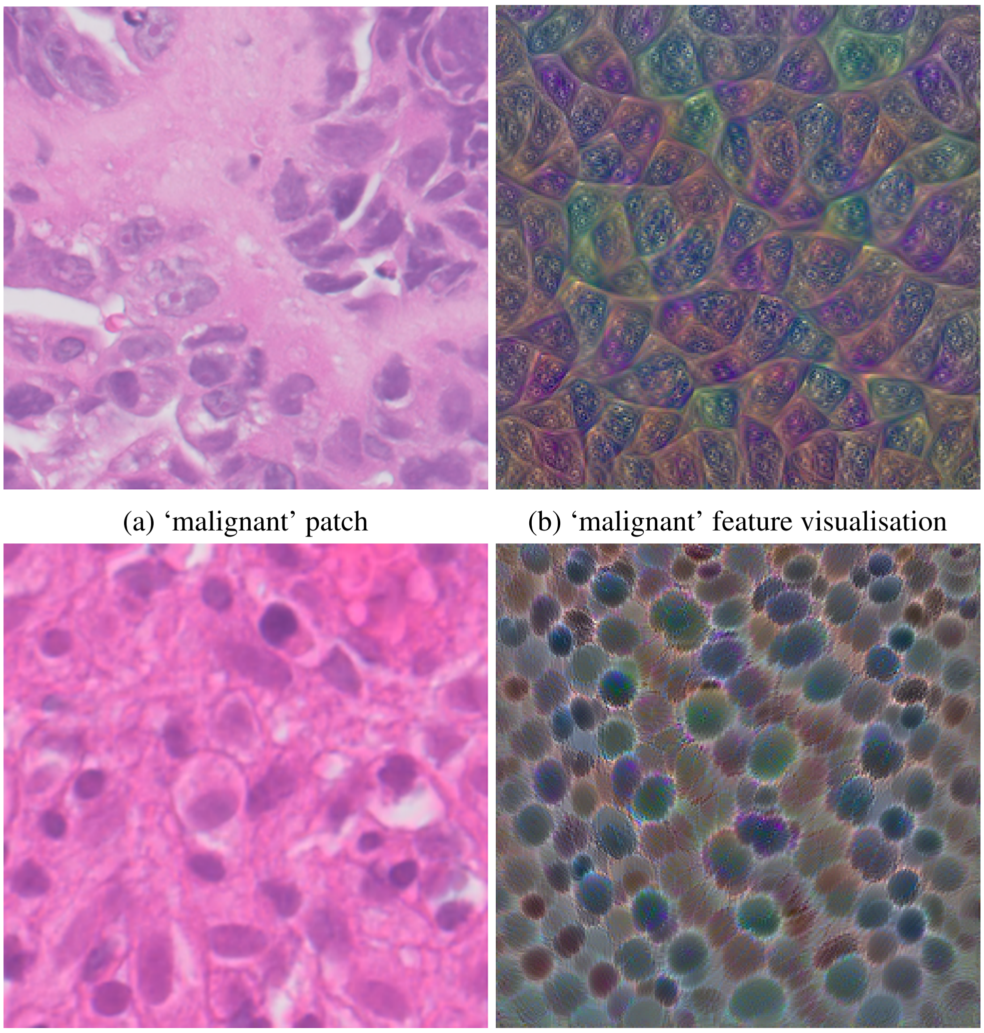
To better understand *what* a this model has learned to look for, as discussed in detail in Chapter 2, learned features for each class can be visualised via input optimisation. In this case, we begin with a zero input matrix of size $3 \times 256 \times 256$ (equal to the slide patch size), and optimise it using gradient ascent to maximise the output logit for each class in turn, using a learning rate of 0.01. This is done for 1000 steps for each class.

Figure 6.6 presents two example patches – one labeled "malignant" and one "other/benign" – alongside their corresponding feature visualisations. Examining these images reveals that, rather than focusing on specific objects or shapes, the model appears sensitive to *textural patterns* within the patches. This aligns well with the nature of histopathology, where diagnostic features often lie in the arrangement and texture of cells and tissues, rather than readily distinct objects.

The malignant patch (Figure 6.6a) exhibits irregular cell shapes, a common feature of cancer cells [45]. This is also evident in the feature visualisation (Figure 6.6b), where the pattern appears to capture irregularly shaped cells with granular and indistinct nuclei. This hints at nuclear atypia – a key feature of malignancy characterised by enlarged, misshapen nuclei with altered chromatin distribution [41].

The other/benign patch (Figure 6.6c) displays more homogeneous, circular cells, a feature captured by the feature visualisation (Figure 6.6d). While individual nuclei are not visibly distinct, the smoother, more organised pattern implies recognition of regular nuclear shapes and sizes, characteristic of healthy tissue [45].

Overall, this model appears to be identifying malignant regions primarily by through irregularly shaped calls with granular and indistinct nuclei. In contrast, benign tissue appears to be identified by more regular, circular cells. These are sensible features that are diagnostically relevant to human experts, suggesting that this model has learned sensible features. These feature visualisations may also help to explain why the saliency-based segmentation overestimates the



(a) ‘malignant’ patch (b) ‘malignant’ feature visualisation

(c) ‘other/benign’ patch (d) ‘other/benign’ feature visualisation

Figure 6.6: (a) and (c) show example ‘malignant’ and ‘other/benign’ labelled patches, for reference. (b) and (d) show input images optimised to maximise output logits for each class, as described in Chapter 2.

proportion of benign pixels. We can see that the model will assign salience for the benign class to regions containing regular circular features, and as shown in Figure 6.6a some cells within a ‘malignant’ patch do seem to have circular features of this kind.

Figure 6.8 highlights an interesting failure mode of saliency-based segmentation mapping with this model. In these benign slides, the saliency maps show that the model finds epithelial structure (both glandular and surface) highly salient for the malignant class – even though the tissue is *not* malignant, according to expert annotation. This is so pronounced that it could be mistaken for an epithelium segmentation model (see Figure 6.7 for context). Endometrial cancer

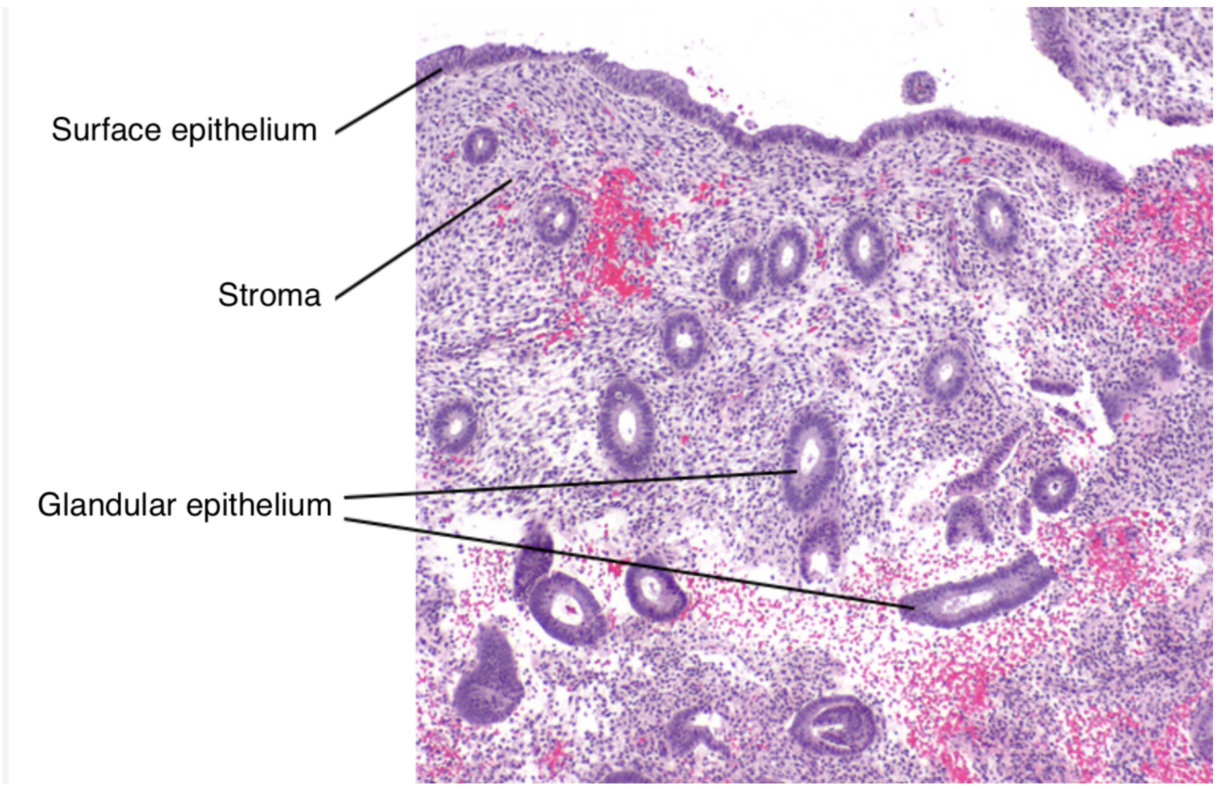
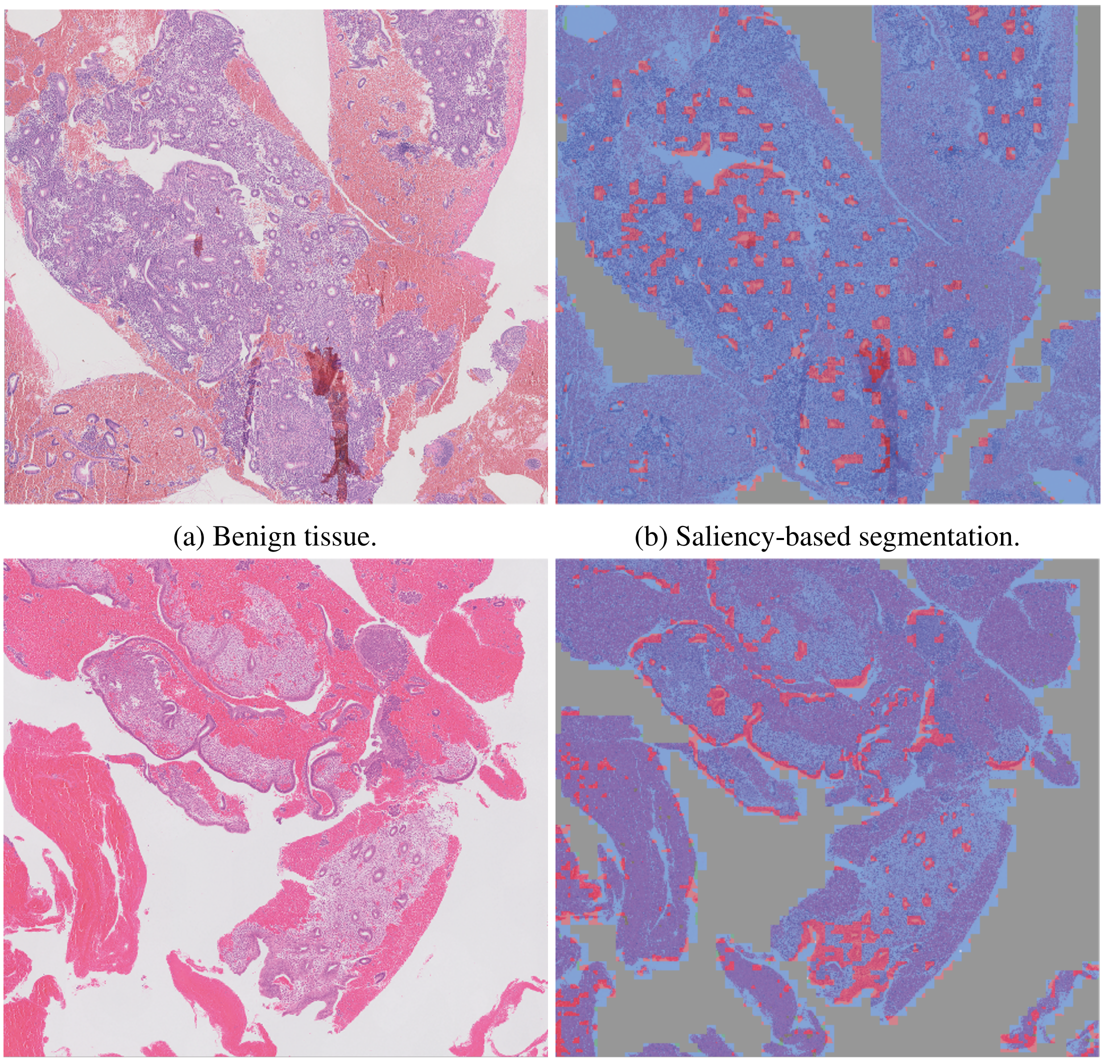


Figure 6.7: Example annotated endometrial tissue with epithelium and stroma labelled. Stroma is connective tissue. Epithelium is a thin layer of cells that cover body surfaces and line cavities, and is seen here forming the lining of glands that appear as tubes cut in cross section.

is an epithelial tumour, typified by glandular abundance and complexity [94], so it is reassuring that the model appears to have learned this and so finds epithelial cells salient. However, it does demonstrate a failure of the saliency-based segmentation method on benign slides.

Note that while the model sees both benign and malignant epithelium as salient for malignancy at the pixel level, which may seem counter-intuitive, it is also very accurate in its slide-level classifications, as shown in the published work [99]. This suggests that while the model sees *all* epithelial cells as relevant for a prediction of malignancy, it is also able to distinguish between a normal arrangement of epithelium and an abnormal one. Pathologists do this by looking at the proportion of epithelium to stroma (connective tissue) and the architecture and complexity of glands present in the tissue [4] – and based on the emphasis on epithelium and glands shown by the saliency maps, it seems likely that this model has learned to do something similar.

In summary, this chapter introduced deep learning in digital pathology, with a particular



(a) Benign tissue. (b) Saliency-based segmentation.

(c) Benign tissue. (d) Saliency-based segmentation.

Figure 6.8: Tissue ((a) and (c)) and saliency segmentations ((b) and (d)) cropped from two benign slides – malignant saliency shown in red, 'other/benign' in blue. Note the distinct epithelium segmentation associated with the malignant class: although the model correctly classified both slides as benign, it has appears to have learned, also correctly, that epithelium is highly salient for endometrial malignancy [94]. (Epithelium is a thin layer of cells that cover body surfaces and line cavities, visible in (a) and (b) as the lining of glands, that appear as cross-sectioned tubes.)

emphasis on weakly-supervised approaches which mitigate the need for costly expert annotations. I demonstrated a novel method to generate pixel-level segmentations from a weakly-supervised model trained on a diverse dataset of endometrial WSIs, in which a saliency-mapping algorithm (HiPe, introduced in Chapter 4) is adapted to focus on regions with lower saliency variance, enabling fine-grained segmentation at the pixel-level. Saliency maps generated by the modified

HiPe were transformed into segmentation maps using post-processing. These saliency-generated segmentations were compared to expert annotations, and showed a close resemblance for malignant and insufficient classes, but overestimation in benign areas. These maps also highlighted this model's emphasis on epithelial structures, a key feature in endometrial cancer, demonstrating the model's learning in line with expert knowledge. However, a limitation of this approach was also highlighted, as the model found epithelial cells salient for the malignant class even in benign slides, leading to inaccurate segmentation maps in that context.

Chapter 7

Hoechst Is All You Need

Supervised machine learning is typically used to automate tasks that humans already know how to do. The previous chapter presented one example of a task of this kind, where human experts provide labels for the data, and a neural network is trained to replicate that ability. In this chapter, I take a different approach – training a neural network to perform a task that humans *cannot* do.

Multiplex immunofluorescence and immunohistochemistry benefit patients by allowing cancer pathologists to identify proteins expressed on the surface of cells, enabling cell classification, better understanding of the tumour microenvironment, more accurate diagnoses, prognoses, and tailored immunotherapy based on the immune status of individual patients [79, 139, 160]. However, they are relatively expensive and time consuming processes which require complex staining and imaging techniques by expert technicians. Hoechst staining is much cheaper and easier to perform, but as it binds to DNA rather than to the proteins targeted by immunofluorescence techniques [35, 34], it has not been used for immune cell classification. In this chapter I show that through the use of deep learning it is in fact possible to identify immune cell subtypes *without* immunofluorescence, training a deep convolutional neural network to identify cells expressing the T lymphocyte marker CD3 from Hoechst 33342 stained tissue only. This model learns previously unknown morphological nuclear features associated with expression of this protein, which can be used to identify CD3 expressing cells for use in key prognostic metrics such as assessment of immune cell infiltration [17]. Immune cells and analysis of the immune contexture are described in more detail in Section 7.1.

For image segmentation tasks Convolutional Neural Networks (CNNs) are most widely used [149]. One type of CNN is the U-Net [129] – a type of residual neural network [63] – so named for its 'U' shaped architecture. Residual neural networks in general, and the U-Net in particular, are well suited for segmentation tasks as they allow spatial information from the input to propagate directly to the output. Recent work using U-Nets has shown great promise for digital pathology applications [118, 114, 78, 92, 74], but publications to date have focused on automating tasks that human experts can already do. In this work a different approach is taken, asking instead – can a deep neural network learn to do something that human experts cannot?

7.1 The Immune Contexture

Among patients with cancers of the same stage, clinical outcomes vary widely [29]. This is thought to be in large part due to the complex interaction between tumour cells and the immune response of individual patients, as the proportion, location, and sub-type of immune cells present in the tissue has been shown to have important implications for patient prognosis [29, 98].

The *immune contexture* refers to the this complex interplay and organisation of the immune cells present within the microenvironment of a tumour.

The immune system is composed of various cell types, each with specialised functions. These cells can be broadly classified into two categories: *innate* and *adaptive* immune cells [8]. Innate immune cells respond rapidly to a wide array of pathogens in a non-specific manner. They lack the ability to remember past infections, leading to a consistent response to repeated exposures to the same pathogen. Adaptive immune cells, comprising T-cells and B-cells, mount a slower but highly specific response against pathogens, recognising and targeting specific antigens. A key feature of adaptive immunity is its memory; upon re-encounter with the same pathogen, these cells can mount a more efficient and robust response [50].

Innate Immune Cells

Neutrophils

Neutrophils are the most abundant type of white blood cells in mammals, playing a key role in the early phase of fighting infections. They respond rapidly to infection and engulf pathogens to eliminate them from the body.

Macrophages

Macrophages are large cells that engulf and digest pathogens and cellular debris. They are also important in antigen presentation, a process critical for activating the adaptive immune response.

Dendritic Cells

Dendritic cells act as messengers between the innate and adaptive immune systems. They capture antigens from pathogens and present them on their surface, stimulating T-cells.

Natural Killer (NK) Cells

NK cells are a type of lymphocyte which identifies and destroys other cells that display signs of infection, stress, or abnormality, such as virus-infected cells or tumour cells. Upon recognition of a target cell, NK cells release cytotoxic (cell killing) granules containing enzymes that induce apoptosis (programmed cell death) in the target cell. NK cells also secrete cytokines (small proteins or signaling molecules) that can modulate the immune response and enhance the activity of other immune cells.

Mast Cells

Mast cells are involved in wound healing and defense against pathogens. They also have roles in allergy and anaphylaxis, releasing histamine and other mediators.

Eosinophils and Basophils

Eosinophils and basophils are involved in the body's response to parasites and allergens. They release enzymes and toxic proteins against parasites but also contribute to allergic reactions.

Adaptive Immune Cells

T-Cells

T-cells originate from bone marrow but mature in the thymus (a small, specialised organ of the immune system, located in the upper anterior part of the chest cavity, just behind the sternum), for which they are named. They recognise antigens through T-cell receptors and come in various types: Helper T-cells (CD4+) that activate other immune cells and aid in antibody production, Cytotoxic T-cells (CD8+) that directly destroy infected or cancerous cells, and Regulatory T-cells which help maintain immune tolerance and prevent autoimmune diseases.

B-Cells

B-cells are produced in bone marrow and mature in lymphoid follicles in lymph nodes throughout the body. They are responsible for antibody production, and can differentiate into plasma cells to secrete large amounts of antibodies. They also serve roles in antigen presentation and cytokine secretion. Memory B-cells, a subset of B-cells, provide a faster immune response upon re-exposure to an antigen.

Memory Cells

Memory cells are a subset of T and B cells that remain in the body after an infection has cleared, to provide a rapid response to subsequent infections by the same pathogen.

Key Factors

Characterisation of the immune contexture looks specifically at the *type* of immune cells infiltrating the tumour, the *density* of these cell populations, the *location and distribution* of the cells within the tumour, and the *functional orientation* of the cells, in terms of being activated or suppressed. Critical immune populations examined typically include T-cells (and their subsets like cytotoxic, helper, regulatory T cells), B-cells, natural killer cells, macrophages, dendritic cells and more. High densities of certain cell types like cytotoxic T lymphocytes and natural killer cells tend to correlate with more favorable clinical outcomes.

By understanding the immune landscape of an individual's tumour, the contexture provides insight into the dynamic immune response against that cancer, including whether the response is eliminating cancer cells or promoting tumour tolerance and growth. This can significantly impact tumour progression and prognosis. A "hot" immune contexture with abundant activated T cells is often associated with better prognosis, while a "cold" contexture with suppressed immune response signifies poor outcomes [163].

The immune contexture can also be predictive of patient response to cancer therapy: certain immunotherapies, rely on a pre-existing active immune response within the tumour microenvironment for their efficacy. Analyzing the immune contexture can help predict which patients are most likely to benefit from these therapies, and enables oncologists to select treatments better matched to the existing immune conditions for each patient [24].

Analysing the Immune Contexture

Researchers employ various techniques to assess the immune contexture. Staining tissue sections with antibodies against specific immune cell markers allows for visualisation and quantification of different immune cell populations within the tumour.

There exist proprietary methods to assess immune cell infiltration, which formally quantify CD3+ and CD8+ T cell lymphocytes both in the centre of tumour and in the invasive margin, as proposed by Galon et al. [52]. Combining their evaluation with T-and B score (CD8+ T cell and

CD20+ B cell) as per Mlecnik et al. had significant predictive power for colorectal cancer patient survival [158, 98], and compared to to the latest guidelines of the American Joint Committee on Cancer/ Union for International Cancer Control (AJCC/ UICC) tumour-node-metastasis (TNM) classification, immune cell infiltration evaluation alone has shown superior prognostic value in international studies of stage I-IV colon cancer patients, and has life-saving applications in clinical decision-making [150, 151, 126, 121, 29, 52, 53].

In order to identify the cells necessary to calculate these valuable metrics, either multiple immunohistochemistry (IHC) or multiplexed immunofluorescence (mIF) are required – both of which are time consuming and expensive protocols [158, 16].

These techniques are used to analyse the immune contexture through visualizing immune cells and proteins in tumour tissues. They work by using antibodies to detect and analyse proteins in tumour tissue in order to visualise and characterise the tumour immune microenvironment. In IHC, antibodies are tagged with enzymes that catalyze a colorimetric reaction when they bind to target proteins, allowing 1-3 proteins to be visualised as colored stains under a microscope in each tissue sample. In contrast, mIF uses antibodies tagged with fluorescent labels that emit signals of varying wavelengths. Using specialised microscopes and computational analysis, mIF allows simultaneous visualisation and quantification of 5-50+ different fluorescently labeled proteins within a single undivided tissue sample. While more technologically complex, mIF expands the ability to profile a diversity of immune cell types/states and spatial organisation by multiplexing more protein targets compared to traditional IHC assays.

Using contemporary equipment, three simultaneous rounds of immunohistochemistry takes around three hours and costs approximately \$20 in reagents, whilst multiplex immunofluorescence requires 9 hours and the associated reagents cost upward of \$70 for a single slide.

CD3 Expression

CD3 is a protein complex and T-cell co-receptor that plays an important role in the adaptive immune system. It is present on *all* T-cells, and found *only* on the surface of T-cells, and so is used as a target for these imaging techniques to identify T-cells within tissue sections. The number of CD3-positive cells in tissue samples can provide valuable insights into the state of the immune response in various diseases. For example, an increased number of CD3-positive T-cells in a tissue may suggest an ongoing immune response, while a lack of CD3-positive cells might be indicative of an immunodeficient state or specific types of lymphomas where T-cells are absent or abnormal.

7.2 Data

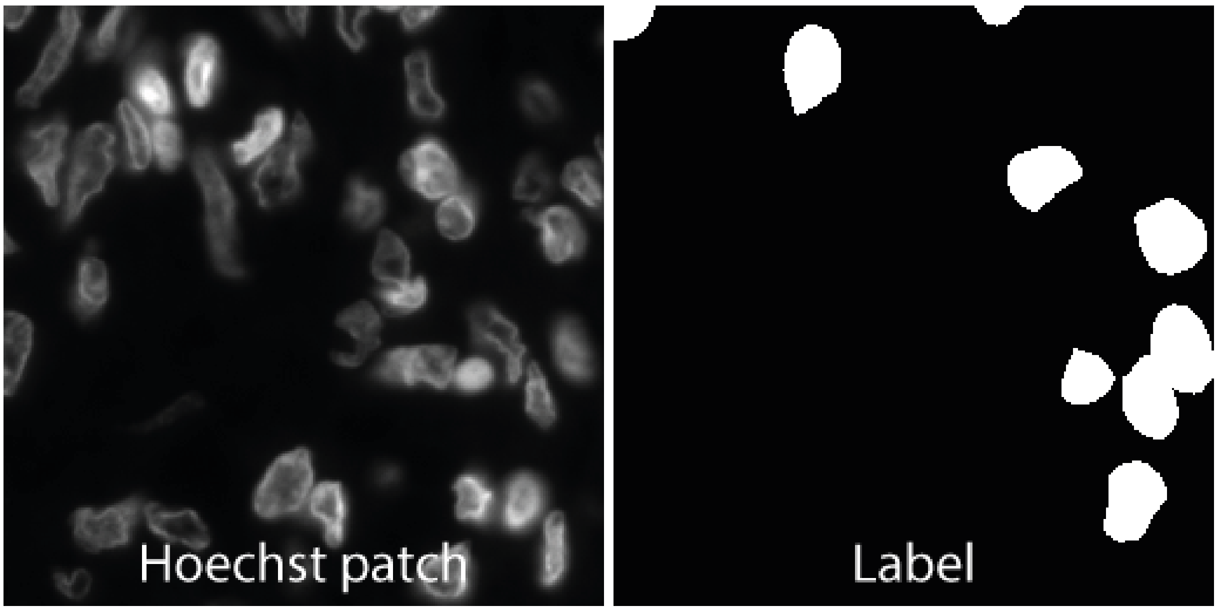


Figure 7.1: Labelled immune cell mask from immunofluorescence image (right) and corresponding Hoechst 33342 stained patch (left).

The data in this study comprised thirty WSIs taken from lung cancer biopsies. The slides were provided by NHS Lothian and were deidentified to preserve patients' anonymity. The thirty slides were randomly selected from three larger cohorts of consenting patients, and each slide was from a different patient. Ten slides were from lung cancer patients, ten from colon cancer patients, and ten from kidney cancer patients. These different tissue locations were used to ensure that the model is unlikely to overfit on any spuriously correlated features unique to individual organs, rather than learning to identify the actual features of CD3 expressing cells. These were imaged using Hoechst 33342, and also using immunofluorescence targeting CD3 expressing immune cells, with a Zeiss Zen Axioscan scanner. Individual cells were then labelled in the immunofluorescence images based on the stain intensity, the results of which were quality controlled by direct visual inspection to ensure label accuracy. These labels were used to create segmentation maps, which were then paired with the Hoechst images, as in Figure 7.1. Additionally, a cell classification dataset was generated by extracting individual images of each cell and pairing them with the immunofluorescence-generated labels.

Immunofluorescence (IF) Protocol and Image Acquisition

Thanks to my colleague Dr In Hwa Um for performing the tissue preparation, imaging and post-processing necessary to obtain the data for this study, and for providing the following two paragraphs, which outline the immunofluorescence protocol she used.

Leica BOND RX automated immunostainer (Leica Microsystems, Milton Keynes, UK) was utilised to perform mIF. The sections were dewaxed at 72°C using BOND dewax solution (Leica, AR9222) and rehydrated in absolute alcohol and deionised water, respectively. The sections were treated with BOND epitope retrieval 1 (ER1) buffer (Leica, AR9961) for 20 min at 100°C to unmask the epitopes. The endogenous peroxidase was blocked with peroxide block (Leica, DS9800), followed by serum free protein block (Agilent, x090930-2). Then the sections were incubated with the primary antibody (CD3, Agilent, A045229-2, 1:70 dilution), followed by biotinylated anti-rabbit secondary antibodies (Thermo Fisher, 65-6140), which was visualised by Alexa flour 750 conjugated streptavidin (Thermo Fisher, S21384). Cell nuclei were counterstained by Hoechst 33342 (Thermo Fisher, H3570, 1:100) and the sections were mounted with prolong gold antifade mountant (Thermo Fisher, P36930).

Zeiss Zen Axioscan was used to capture fluorescent images. Two different fluorescent channels, Hoechst3334 and AF750 were simultaneously used to capture individual channel images under 20× object magnification. The exposure time of the channels were 8 and 800 milliseconds, respectively. The images were saved in Carl Zeiss Image (CZI) format (a file format developed by Carl Zeiss Microscopy, used for storing microscopy images and metadata in a single file) [3] and then opened in QuPath v.0.2.3 [20] (a widely used open-source image analysis program for whole-slide digital pathology images).

Image Post-Processing

QuPath's implementation of the segmentation algorithm StarDist [135] was used to segment cell nuclei, with the probability threshold of cell detection, pixel size and cell expansion set to 0.6, 0.2270 and 1.0 respectively. StarDist is a deep learning method for 2D and 3D image segmentation, particularly designed for the segmentation of star-convex shapes in microscopy images. Star-convex shapes are shapes in which at least one point exists within the shape exists, from which any other point within the shape can be reached with a straight line that lies entirely within the shape's boundaries. StarDist uses convolutional neural networks (CNNs) to predict the shape of objects in the form of star-convex polygons for 2D images or polyhedra for 3D images. It segments objects by predicting distances from the object's centroid to its boundary, in a fixed set of directions resembling a star shape, and is particularly effective at segmenting densely packed or overlapping objects [135]. All segmented cells with an intensity of greater than 2200 in the AF750 channel were classified as CD3 expressing. QuPath automatically captures a number of different measurements from these segmented and classified cells, as described in the following section.

Data Analysis

The total number of labelled cells present across all slides was 8,160,203. These were unequally distributed across the slides, ranging from 16,991 to 723,458 labelled cells per slide. Of these cells, 1,167,715 expressed CD3, representing 14.3% of the total.

As shown in Table 7.1, the circularity, size, and nucleus density of the CD3 expressing cells also varied somewhat, both inter- and intra-slide. To test whether these simple morphological features alone had predictive power, I attempted to train a number of different binary classification

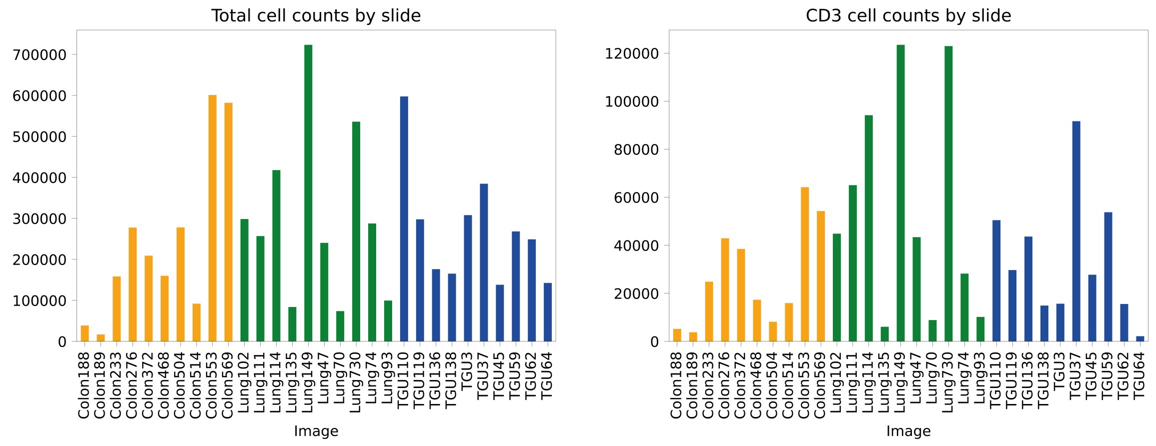


Figure 7.2: Cell counts per slide. Colon and lung slides (shown in yellow and green respectively) formed the training and validation set. Kidney cancer slides (shown in blue) exclusively formed the holdout test set. This is to ensure that the model is validated on its ability to identify CD3 expressing immune cells based on their own morphological features, guarding against overfitting to any correlated features unique to specific tissues.

	mean	std	min	25%	50%	75%	max
Nucleus: Area µm^2	29.70	21.23	6.00	16.20	24.33	35.83	1,084.42
Nucleus: Length µm	19.74	6.40	8.84	15.28	18.58	22.84	252.07
Nucleus: Circularity	0.88	0.10	0.15	0.84	0.91	0.95	0.99
Nucleus: Solidity	0.99	0.03	0.32	0.99	1.00	1.00	1.00
Nucleus: Max diameter µm	7.39	2.57	2.95	5.59	6.86	8.65	60.04
Nucleus: Min diameter µm	4.90	1.65	1.23	3.76	4.67	5.71	47.91
Cell: Area µm$\hat{2}$	48.21	26.95	6.28	30.36	41.97	57.66	1,200.90
Cell: Length µm	25.35	6.52	9.56	20.76	24.23	28.58	258.87
Cell: Circularity	0.89	0.08	0.16	0.85	0.91	0.95	0.99
Cell: Solidity	0.98	0.03	0.39	0.98	0.99	1.00	1.00
Cell: Max diameter µm	9.22	2.58	3.56	7.42	8.70	10.50	62.15
Cell: Min diameter µm	6.56	1.71	1.84	5.35	6.33	7.43	48.78
Nucleus/Cell area ratio	0.58	0.09	0.27	0.52	0.58	0.64	1.00

Table 7.1: Cell and nucleus statistics for all data.

models to discriminate between the labelled CD3 data and an equal number of randomly selected other cells, based on nucleus and cell area, length, circularity, maximum and minimum diameter, and solidity.

Models tested included logistic regression, support vector machines (SVM), k-nearest neighbours (kNN), naive bayes, a simple neural network, decision trees, and random forests, implemented with scikit-learn (an open-source statistical modelling library). I partitioned the data into

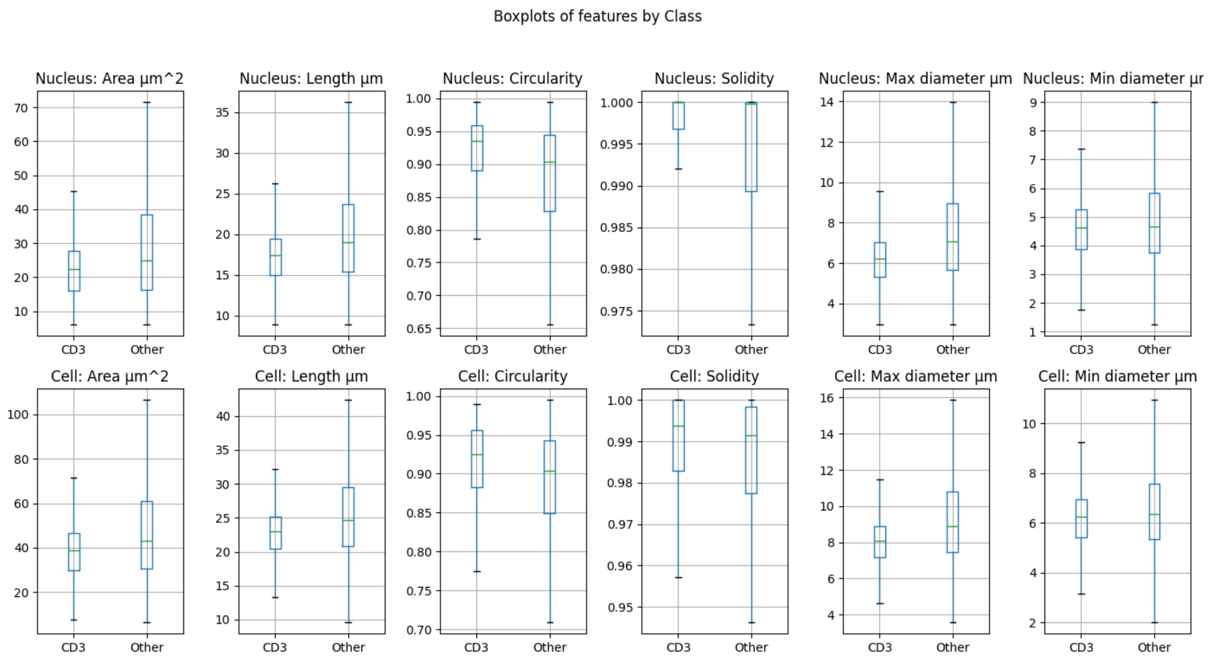


Figure 7.3: Cell feature box plots for CD3-expressing and 'Other' cells in the dataset.

a training set, constituting 70% of the data, a validation set, constituting 15% of the data, and a test set with the remaining 15% . Feature scaling via standardisation was applied to ensure similar feature magnitude to aid convergence. I also used gridsearch for hyperparameter tuning. However, none of these classifiers achieved more than chance accuracy, suggesting that the cell measurements lack the necessary discriminative power for this classification task.

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
LogisticRegression	0.5003	0.5003	0.5003	0.5003
SVM	0.4985	0.4980	0.4988	0.4422
kNN	0.5003	0.5003	0.5003	0.5003
NaiveBayes	0.5003	0.4999	0.4999	0.4485
MLP	0.5005	0.2503	0.5000	0.3336
DecisionTree	0.4998	0.5009	0.5002	0.3848
RandomForest	0.4997	0.4996	0.4996	0.4944

Table 7.2: Tabular model evaluation results on holdout test set data. The cell measurements alone do not appear sufficient to discriminate between CD3-expressing cells and others.

The results of these experiments are shown in Table 7.2.

As it was not possible to achieve above-chance performance on these features alone, I turned to more complex convolutional neural networks to enable direct representation learning from images. From the data, I created two different tasks: cell classification, and patch segmentation.

For the segmentation task, patches of dimension 256 × 256 were extracted from each of the Hoechst 33342 stained slides at full resolution and paired with the per pixel labels from the immunofluorescence intensity classifier as shown in Figure 7.1.

For the cell classification task, in order to create a balanced dataset, from the Hoechst stained slides all CD3 expressing cells and an equal number of randomly selected non-CD3 expressing cells were exported at full resolution. Individual cells were isolated by masking out the background such that each sample contained one cell only. Each of these single-cell images was of dimension 64×64. Each Hoechst image (from both the segmentation and classification dataset) was normalised by applying min-max scaling individually prior to training. Normalisation was used instead of standardisation to counteract variability in pixel value range between slides, and to ensure that all input features (pixel values) were on a similar scale, thereby facilitating more stable and faster convergence during the training process. From the thirty slides, all ten kidney cancer slides were held out as test set. Two slides were selected randomly from each of the remaining lung and colon cancer cohorts for use in validation, and the remaining eight from each were used for training.

Due to differing numbers of patches available per slide, this provided a total of 174,388 training patches from eight colon slides and eight lung slides, 66,259 validation patches from two colon slides and two lung slides, and 142,189 test patches from all ten kidney slides.

7.3 Model Architecture and Training

For the cell classification dataset, I used a standard resnet50 [63]. For the segmentation dataset, I used a U-Net backbone [129] with a resnet50 encoder [63].

All computation was performed using eight NVIDIA Tesla V100 GPUs (high-end computing GPUs designed for deep learning and high-performance computing), running on a GNU/Linux operating system. All model training code was written in Python, using Pytorch (an open-source machine learning library) for model training, plus pandas (an open-source library providing high-performance data structures and data analysis tools) for data logging and Matplotlib (another open-source library providing plotting utilities) for plot generation.

The classification model was trained for up to 100 epochs using Adam optimisation [90], with a batch size of 512 and a learning rate of 0.000001. Early stopping was performed to limit overfit, with training halted if no decrease in validation loss was observed for 10 epochs – this resulted in the model being trained for 34 epochs in total. In order to directly optimise for a balance of precision and recall, I used the F1 score as the loss function. In the equations below, τ_c is the true class label (encoded in binary form) and ρ_c is the softmaxed model prediction for that class. Precision and recall for a class c are defined as P_c and R_c respectively. ε is a small constant (0.0001) to avoid division by zero.

$$P_c = \frac{\tau_c \rho_c}{\tau_c \rho_c + (1 - \tau_c)\rho_c + \varepsilon} \tag{7.1}$$

$$R_c = \frac{\tau_c \rho_c}{\tau_c \rho_c + \tau_c(1 - \rho_c) + \varepsilon} \tag{7.2}$$

The F1 score for class c, denoted as $F1_c$, is calculated using P_c and R_c:

$$F1_c = 2 \cdot \frac{P_c \cdot R_c}{P_c + R_c} \tag{7.3}$$

The overall F1 loss (L) to be minimised is expressed as:

$$L = 1 - \frac{1}{C} \sum_{c=0}^{C-1} F1_c \tag{7.4}$$

I found that using this F1 loss instead of the more usual cross entropy loss resulted in an increase in accuracy of around 7%.

The segmentation model was also trained using the F1 loss, but with a batch size of 128. The same early stopping protocol was used, resulting in the segmentation model being trained for 29 epochs, also using Adam optimisation. An initial learning rate of 0.001 was used, and a learning rate decay protocol employed in which the initial learning rate was divided by the current number of epochs, at each epoch. This was found to improve performance, as it enabled the model to quickly learn the ‘background’ features before fine-tuning the more complex cellular features.

The batch sizes between the two models differ in order to take advantage of available memory — the samples for the classification task are 64×64 pixels in size, versus the 256×256 patches used for the segmentation model, allowing the classification batch size to be far larger.

This protocol was designed after significant experimentation, considering a range of architectures and hyperparameters, as shown in Figure 7.4. These deeper encoders, as implemented in the Segmentation Models Pytorch library [68] failed to improve performance over the resnet50.

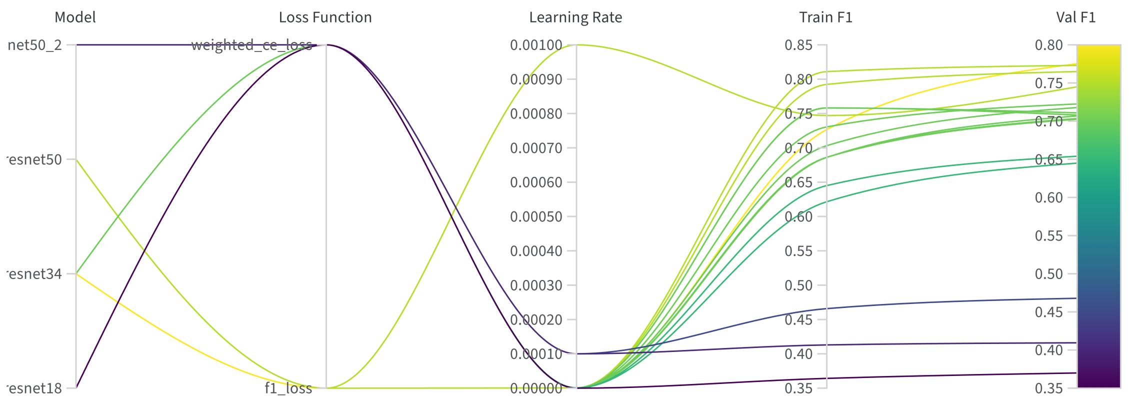
7.4 Evaluation

To assess model performance I use several metrics for both the classification and segmentation task: precision (P), recall (R), F1 score (F), accuracy (A), and intersection over union (IOU) as defined in Equation 7.5.

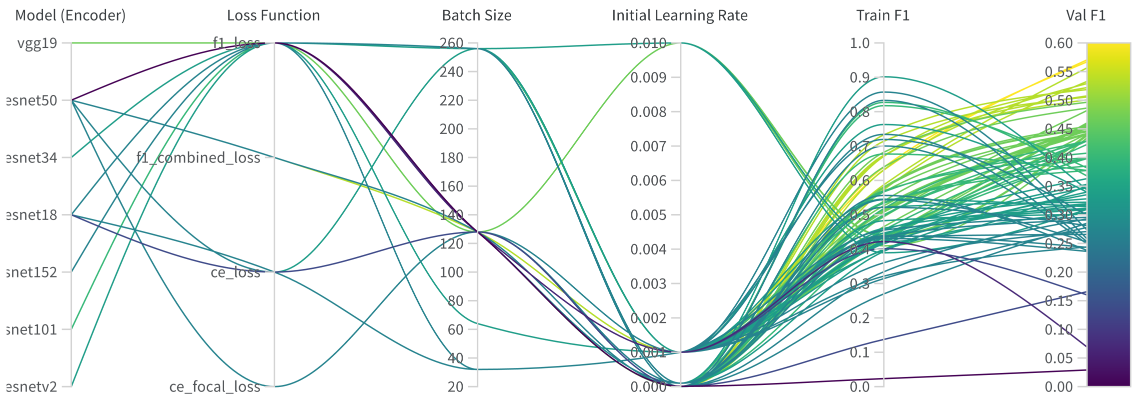
For the segmentation task, the ground truth (denoted as τ) is a matrix of the same dimensions as the input image, where each element is either 0 or 1 indicating the absence or presence of a CD3-expressing cell in each pixel in the input. The model output (ρ) is a matrix of the same size, containing probability values ranging from 0 to 1, representing the *model’s prediction* for each pixel. The precision, recall, F1 score, accuracy, and IOU are computed for all pixels, and the mean of all pixel scores across all images used as the metric.

For the classification task, the same metrics are used, but the mean is taken over the ground truth and class predictions for each image, (rather than for *each pixel* in each image.

$$\begin{aligned}
\mathrm{tp} &= \sum(\tau \cdot \rho) \\
\mathrm{tn} &= \sum[(1-\tau)\cdot(1-\rho)]
\end{aligned}$$



(a) Training configuration for the cell classification task. *weighted_ce_loss* is standard cross entropy loss, weighted proportionately with the number of CD3 expressing cells in the training dataset. Models tested for the classification task were, from top to bottom, wide-resnet50, resnet50, resnet34 and resnet18.



(b) Training configuration for the cell image segmentation task. *ce_loss* refers to cross entropy loss; *f1_combined_loss* is calculated as the mean of cross entropy loss and the F1 loss described in Section 7.3; and *ce_focal_loss* is the focal loss for imbalanced datasets, proposed by Lin et al. [82]. All models consisted of a U-Net backbone, with a number of different encoders. Encoders tested for the segmentation task were, from top to bottom, vgg19, resnet50, resnet34, resnet18, resnet152, resnet101 and inception-resnet2.

Figure 7.4: Training runs with different models, loss functions, batch sizes and learning rates for the segmentation and classification tasks.

$$
\begin{aligned}
\mathrm{fp} &= \sum[(1-\tau)\cdot\rho] \\
\mathrm{fn} &= \sum[\tau\cdot(1-\rho)] \\
P &= \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fp}} \\
R &= \frac{\mathrm{tp}}{\mathrm{tp}+\mathrm{fn}} \\
F1 &= \frac{2\cdot P\cdot R}{P+R} \\
A &= \frac{\mathrm{tp}+\mathrm{tn}}{\mathrm{tp}+\mathrm{tn}+\mathrm{fp}+\mathrm{fn}} \\
\mathrm{IOU} &= \frac{\mathrm{tp}}{\sum(\rho+\tau)-\mathrm{tp}}
\end{aligned}
\tag{7.5}
$$

Evaluation metrics: *tp* is true positive classifications, *tn* is true negative classifications, *fp* is false positive classifications, and *fn* is false negative classifications. These are aggregated over all pixels for semantic segmentation tasks.

P is precision, *R* is recall, *A* is accuracy, and *IOU* is the Intersection Over Union. τ is the true label, ρ is the model's prediction.

Table 7.3 shows the performance of the classification and segmentation models according to these metrics. The cell classification model achieved over 80% precision, recall and F1 score on

		F1	Precision	Recall	Accuracy	IOU
Segmentation (Pixel)	Training	0.630	0.579	0.693	0.971	0.460
	Validation	0.570	0.511	0.648	0.962	0.399
	Test	0.599	0.596	0.605	0.976	0.429
Segmentation (Centroid)	Training	0.826	1.0	0.704	0.968	0.704
	Validation	0.827	1.0	0.705	0.705	0.705
	Test	0.789	1.0	0.652	0.652	0.652
Classification	Training	0.802	0.806	0.803	0.802	
	Validation	0.773	0793	0.776	0.776	
	Test	0.805	0.807	0.805	0.805	

Table 7.3: Performance on training, validation and test slides for both segmentation and classification models. Pixel-based performance uses all pixels in the segmentation mask for evaluation, while centroid-based uses only the labelled cell centroids.

the test set, showing excellent generalisation to unseen slides. Moreover, since the test slides were from kidney cancer slides and the training and validation sets from only lung and colon cancer slides, this shows that the model has successfully learned to identify CD3 expressing cells based on their own features, rather than from features specific to a single cancer type or location. This suggests that the ability to identify CD3 expressing cells from morphological features made visible by Hoechst staining is not limited to lung and colon cancer patients, and can be generalised from them to patients with other cancers.

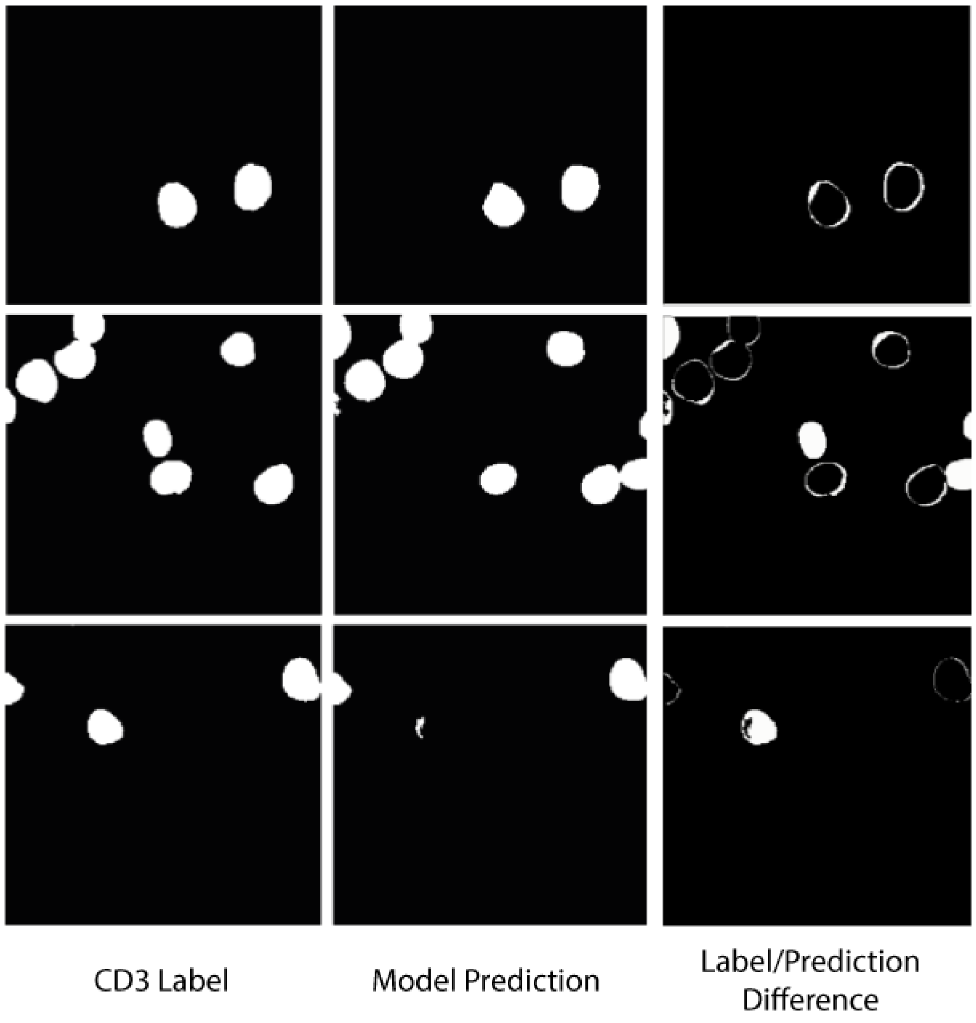


Figure 7.5: Visualisation of where the segmentation model prediction differs from the true CD3 pixel labels. Note that, while the model correctly identifies most of the CD3 expressing cells, it often misses pixels around the cell boundaries.

At first glance the segmentation model does not fare as well, with F1 scores around 60%. However, when the evaluation metrics are computed using the cell centroids – that is, comparing the label and model prediction at the centre point of each labelled cell only – the performance improves significantly, approaching that of the classification model with an F1 score of nearly 80% on the test dataset, and perfect precision (although the recall is slightly diminished). This is due to the fact that the per-pixel metrics are calculated against segmentation masks generated from immunofluorescence images using StarDist, which only estimates the cell boundaries (details of the entire image acquisition protocol can be found in Chapter 7.2). This means that there is likely some inaccuracy in the pixel labels around the cell boundaries. As such, we would expect the segmentation model to perform poorly around the edges of cells. This can be seen in Figure 7.5, and explains the performance increase when the model is evaluated based only on the labels and predictions for pixels corresponding to the cell centroids.

Figure 7.6 shows a number of example cells from the test set, along with their ground-truth classification and the model's prediction. Figure 7.7 shows the cell classification confusion matrices for training, validation and test sets, demonstrating robust and generalisable classification ability with little evidence of overfit.

Inspection of the dataset and statistics in the previous chapter (see Figure 7.3) shows that CD3 expressing cells are on average smaller, and exhibit a higher degree of nuclear solidity than other cells in Hoechst imaging. Since each cell image was individually normalised prior to training and inference, any *relative* difference in intensity between cells of different types would be mitigated to a large extent. However, most of these differences in distribution would remain even after normalisation, so to explore whether this higher solidity and difference in size is used by the classification model in preference to morphological features, training and validation were repeated using the same slides at 2x lower magnification level. This preserves shape, relative size and relative intensity but obscures fine-grained features at a cellular level. On this training data the model performance on validation was far lower, indicating that small features visible at the highest magnification level were necessary to achieve these results.

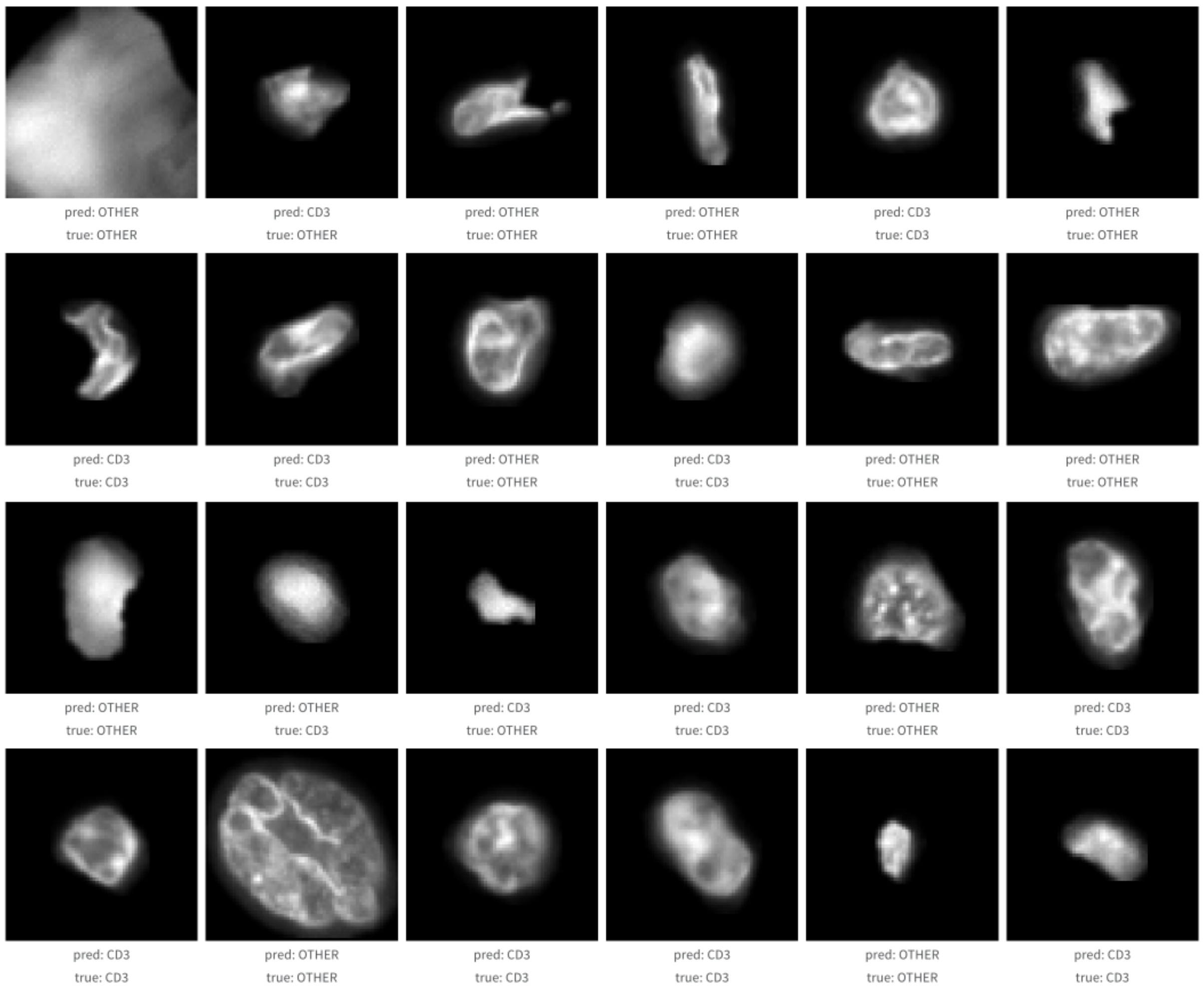


Figure 7.6: Example cell samples, model predictions and actual classifications from the test set.

7.5 Interpretability

To explore *how* each trained model is able to distinguish between different lymphocytes, I generated prototypical input images for the CD3 class, as described in Chapter 2, by optimising an input image to maximise the output logit for the CD3 class. To encourage the optimisation process to modify *only* those pixels necessary for a positive classification, I used a regularisation term consisting of the sum of the absolute value of the pixels in the input, multiplied by a hyperparameter r.

For the segmentation model, this technique was extended to isolate learned features for individual cells, by maximising the CD3 output logit of a single central pixel, as proposed in Chapter 2.2. This results in clearer visualisations than naively optimising the input to maximise

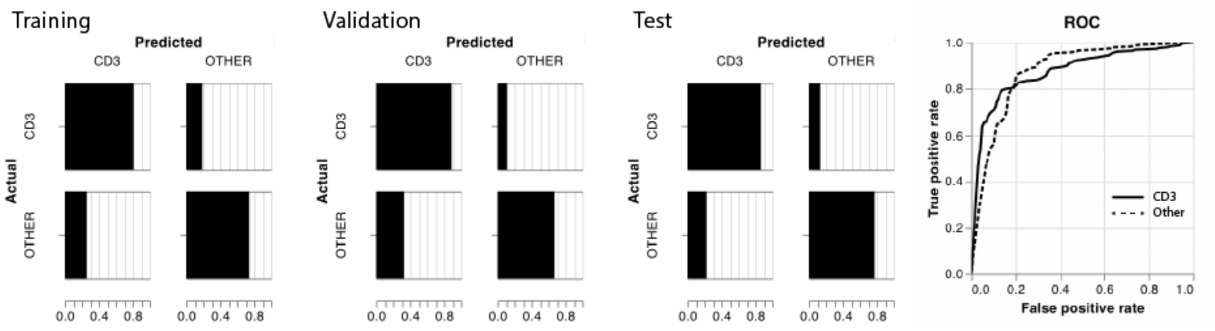


Figure 7.7: Confusion matrices for training, validation and test sets, along with Receiver Operator Characteristic (ROC) curve from test set only.

Transforms	
Gaussian Blur (applied to input, with a 3×3 kernel, and sigma of 1.0)	True/False
Rotation (randomly rotate, between 0 and 360 degrees)	True/False
Jitter (shift input in a random direction by one pixel at each step)	True/False
Sigmoid (applied to input)	True/False
Clamp (clamp input pixel values between 0 and 1)	True/False
ReLu (applied to input)	True/False
Loss Function	
Which output pixel logit to maximise (only applicable to segmentation model)	Centroid/All
Hyperparameter *r* for the regulariation term (which is multipled by the sum of the absolute value of the input pixels)	0/0.5/1
Optimiser	
Gradient Descent	SGD/Adam
Learning Rate	0.1/0.01/0.001
Optimisation Steps	100/200/500
Initialisation	
Initial input pixel values (‘random’ is random noise uniformly distributed between 0.0 and 1.0)	0/0.5/1.0/random

Table 7.4: The range of different transformations applied to inputs prior to inference at each step, regularisation strategies, and hyperparameters explored for prototype generation. Clamp and sigmoid are used to reduce the inherent tendency of this kind of feature visualisation to optimise for adversarial inputs, as the actual Hoechst patches used in training and inference are normalised between 0 and 1.

the CD3 logit for *every* pixel in the output, as shown in Figure 7.10.

I experimented with a range of different protocols for prototype generation, as described in Table 7.4 - however, I found that even given a broad range of hyperparameter exploration, the variation in the outputs was typically small. This suggests that the models have learned fairly robust features.

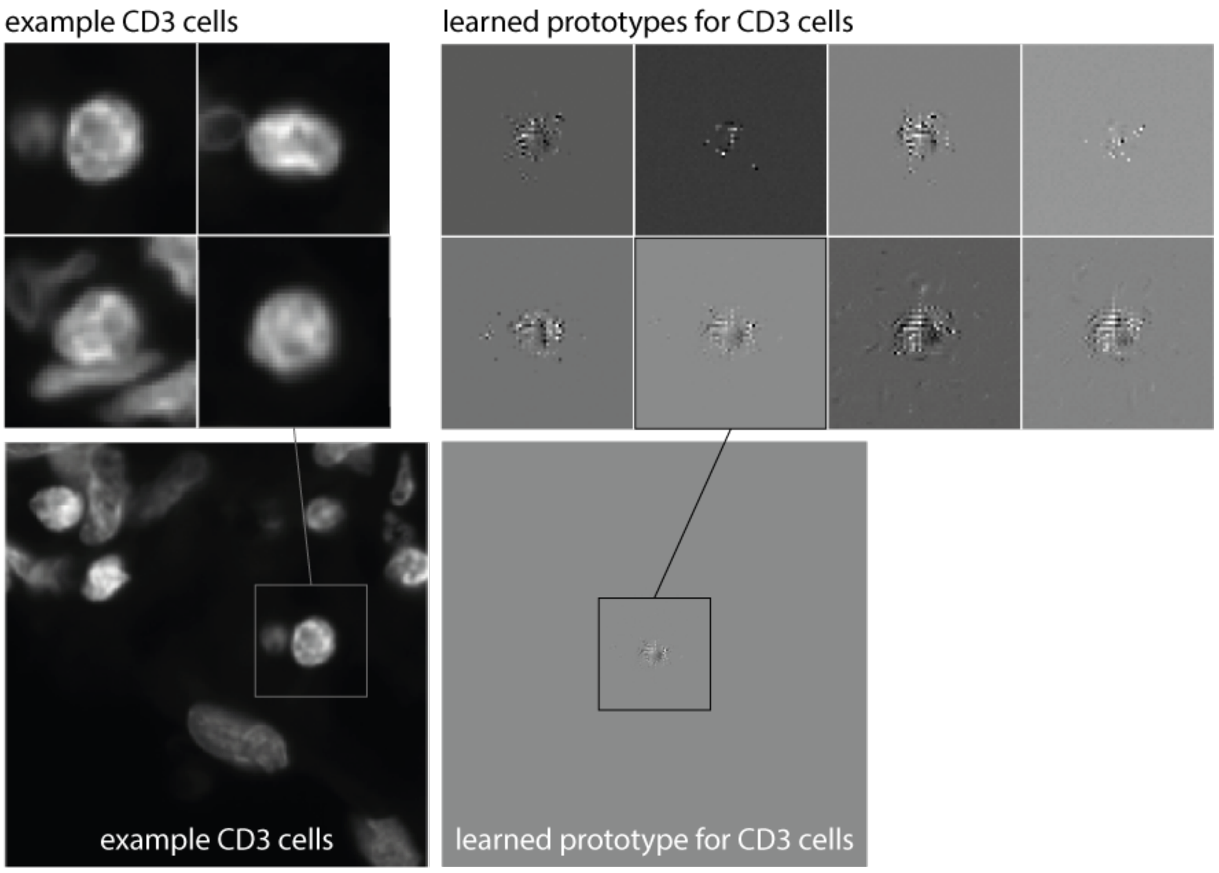

example CD3 cells
learned prototypes for CD3 cells
example CD3 cells
learned prototype for CD3 cells

Figure 7.8: Comparing actual CD3 expressing cells from the training data with a selection of the segmentation model's learned CD3 prototypes, generated with a range of hyperparameters, as outlined in Table 7.4. These prototypes were generated using centroid logit maximisation (proposed in Chapter 2.2) and result in >99% probability of CD3 classification for the model's central output pixel.

Figure 7.10 shows CD3 prototypes for both the segmentation and classification models. These were generated for 100 steps, using SGD or Adam optimisation; a learning rate of 0.01; and the initial input set uniformly to zero. These optimised inputs show that using centroid maximisation effectively generates the model's learned prototypical CD3 cell around the optimised pixel in the output. These generated cells have distinct textural quality which is somewhat also observable in the CD3 labelled cells themselves, and are of a slightly smaller radius than the average CD3 expressing cell, as shown in Figure 7.8. In all generated cells, a small darker region is evident, with two lighter regions of unequal size on either side. These lighter regions have a marked striped quality, with alternating lines of dark and light pixels. I initially thought that

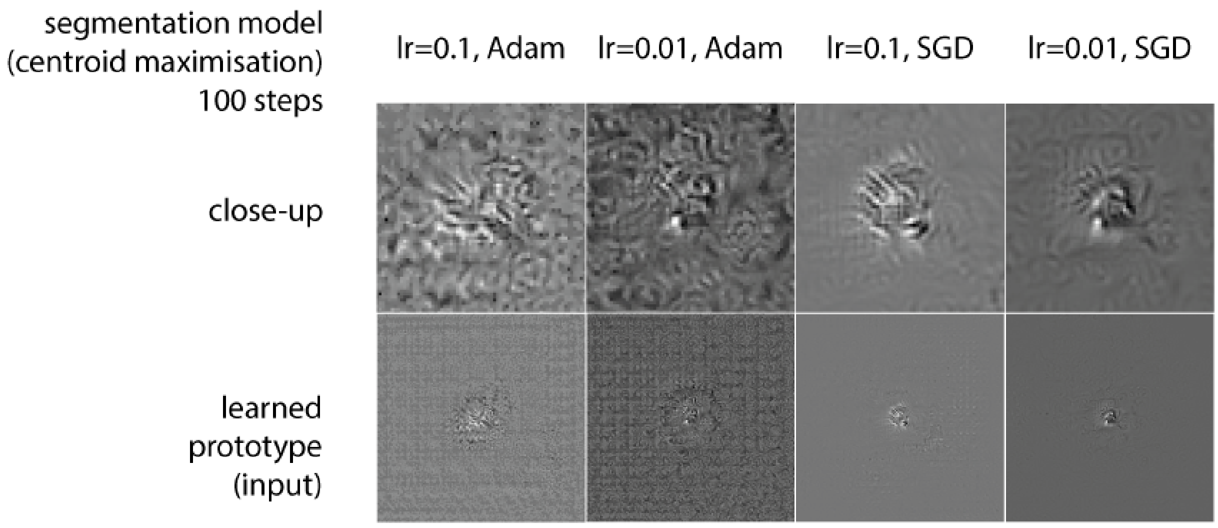


Figure 7.9: CD3 prototypes with Gaussian blur used during the optimisation process, with a kernel size of 3×3 and sigma of 1.0.

this was an architectural artefact due to the convolutional layers in the model, as discussed in Chapter 2, but after further experimentation I determined this is unlikely to be the case – these high contrast regions persist even when Gaussian blur and jitter are applied during generation, which effectively remove chequerboard artefacts associated with convolutional networks, as shown in Figure 7.9. Additionally, these high contrast stripes also appear in the learned CD3 prototype of the classification model, as shown in Figure 7.10.

To gain further insight into the learned features, HiPe was used to generate saliency maps for both the classification and segmentation tasks, as shown in Figures 7.11 and 7.12 respectively. The standard implementation of HiPe described in Chapter 4.2 was used on the softmaxed output of the model, with a zero perturbation substrate (where perturbed portions are replaced with zeros, as described in Chapter 3.3. Zero substrate is a natural choice in this case, as during training and inference inputs to the model are normalised so that all pixels are between 0 and 1, and the ‘background’ of Hoechst stained images are typically black. Input saliency based methods like HiPe are more transparently interpretable than input optimisation, as they explicitly show which areas of the input image were more or less important in determining the output for each class for a specific input. HiPe was used in preference to other input saliency based explanatory techniques as it is much quicker than similar perturbation-based saliency methods for

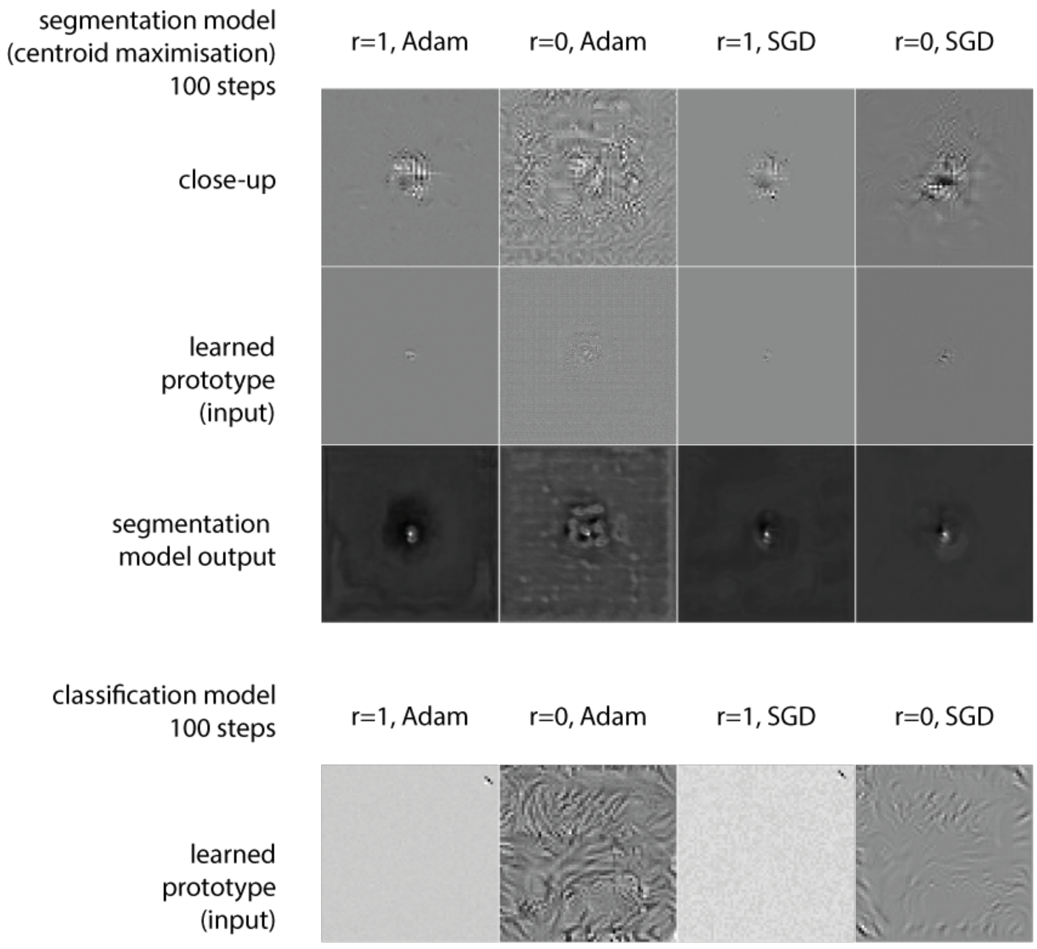


Figure 7.10: CD3 Prototypes for the segmentation model (above) and the classification model (below), comparing the effect of regularisation and different optimisers on the optimised input. All generated prototypes shown here result in >99% probability of CD3 classification (of the center pixel in the case of the segmentation model).

large images containing relatively small salient features, and is more precise than gradient-based methods which are often indistinct. This is particularly important in this case, as the generated prototypes suggest that key learned features are relatively small.

For the purposes of comparison, I also apply standard iterative occlusion [171] (ItP) with kernel sizes of 2×2 and 1×1. Saliency maps generated with both HiPe and ItP for a selection of CD3 cells are shown in Figures 7.11 and 7.12, for the classification and segmentation models respectively. The HiPe saliency maps for the maximum depth (i.e. at the smallest kernel size) were retained before aggregation and are displayed in addition, in order to visually isolate the smallest, most salient features. The resulting saliency maps show that larger salient regions comprised the cell nuclei themselves, as would be expected. In all cases, the *most* salient regions highlighted texture in the centre of the cell nucleus. This saliency pattern is similar in cases where the model made incorrect classifications. The outer edges of nuclei do not appear salient at all, indicating that the model did not learn to use the circumference, circularity or size of the nuclei to make predictions, as suspected based on my previous attempt to train a tabular classifier model from these measurements (details in Section 7.2). It is natural that the salient regions in the saliency maps are the cell nuclei, rather than the cell surface or cytoplasm, which are not visible in Hoechst-stained images.

7.6 Discussion

Hoechst 33342 binds to cell DNA. More specifically, it attaches to the *minor groove* [35] (DNA has two grooves – the major and the minor – which are formed due to the geometrical arrangement of the two strands of DNA as they spiral around the double helix.) Hoechst dyes bind most strongly to A-T (adenine-thymine) pairs in the minor groove, and less strongly to other pairs – meaning that A-T rich regions have more pronounced fluorescence.

Because Hoechst makes DNA visible, it is commonly used for cell quantification (locating and counting cells in tissue) [81] and DNA quantification (measuring the DNA content of cells, which can be predictive of cell health and cell cycle stage) [58]. It is never used in practice to identify which proteins a cell expresses – indeed, it was surprising to all of the pathologists that I spoke to that the models are able to accurately classify CD3 expressing cells based on the features made visible by Hoechst. So, what could the models be using to make predictions?

It is possible that the model is simply exploiting a spurious correlation in the data, which might be due to technical artefacts or biases in image acquisition, labelling or annotation. For example, the CD3-expressing cells might appear smaller and have different texture due to the spatial resolution or contrast of the images, or the way in which the tissue samples are prepared.

These technical factors might be confounded with CD3 expression, leading the model to exploit them as proxies for CD3 expression, even though they might not be biologically relevant. If this were the case, the model's performance might not generalise to unseen data, and the findings might not be of biological significance. However, I think this is unlikely – model accuracy was evaluated on a holdout test set from a different set of patients and a different tissue type; analysis of cell measurements did not show any correlations of this kind; and attempts to train statistical models based on cell measurements failed (as discussed in Section 7.2). In addition, both saliency mapping and prototype generation suggest that it is morphological features in the cells' DNA structure that are used for prediction.

In the absence of spurious correlation, the fact that these models make accurate predictions suggests they are capturing some underlying biological pattern or relationship between Hoechst staining and CD3 expression – a previously unknown feature of cell morphology or staining patterns. The models seem to infer CD3 presence without direct evidence of the protein itself, finding a signal within the nuclear patterns as highlighted by the saliency maps – perhaps related to the density or spatial arrangement of chromatin or other nuclear structures, or differences in the pattern or intensity of Hoechst binding to DNA sequences. This could be indicative of a deeper and currently unknown biological link between nuclear morphology and T-cell function.

The identification of such features has potential implications for our understanding of T-cell biology, and might shed light on the relationship between molecular and cellular phenotype in the immune system. This would require further experimentation and validation, e.g., by exploring the relationship between chromatin structure, gene regulation and CD3 expression using additional experimental techniques, such as comparing the nuclear features of T-cells with other cell types in a controlled setting.

In summary, in this chapter I trained two deep neural network models to identify CD3 expressing cells from Hoechst stained images – something that was not previously considered possible, as Hoechst does not make proteins visible, binding instead to DNA in the cell nucleus.

This is valuable in and of itself, because immune contexture analysis currently requires expensive and time-consuming imaging protocols (as discussed in Section 7.1), whereas Hoechst staining is far cheaper and quicker. Furthermore, the use of interpretability makes it possible to see *how* the model is able to do this task. Saliency mapping and prototype generation suggest that there is a relationship between protein expression and nuclear morphology that humans are currently unaware of, but that is robust enough to predict one from the other. Further experiments by domain experts will be necessary to fully understand what the model has learned in this case. Primarily, I see this chapter as a proof-of-concept of a new approach to *knowledge discovery* – in which models are trained to perform tasks that humans *can't do*, and then interpretability methods are leveraged to help us understand *how*.

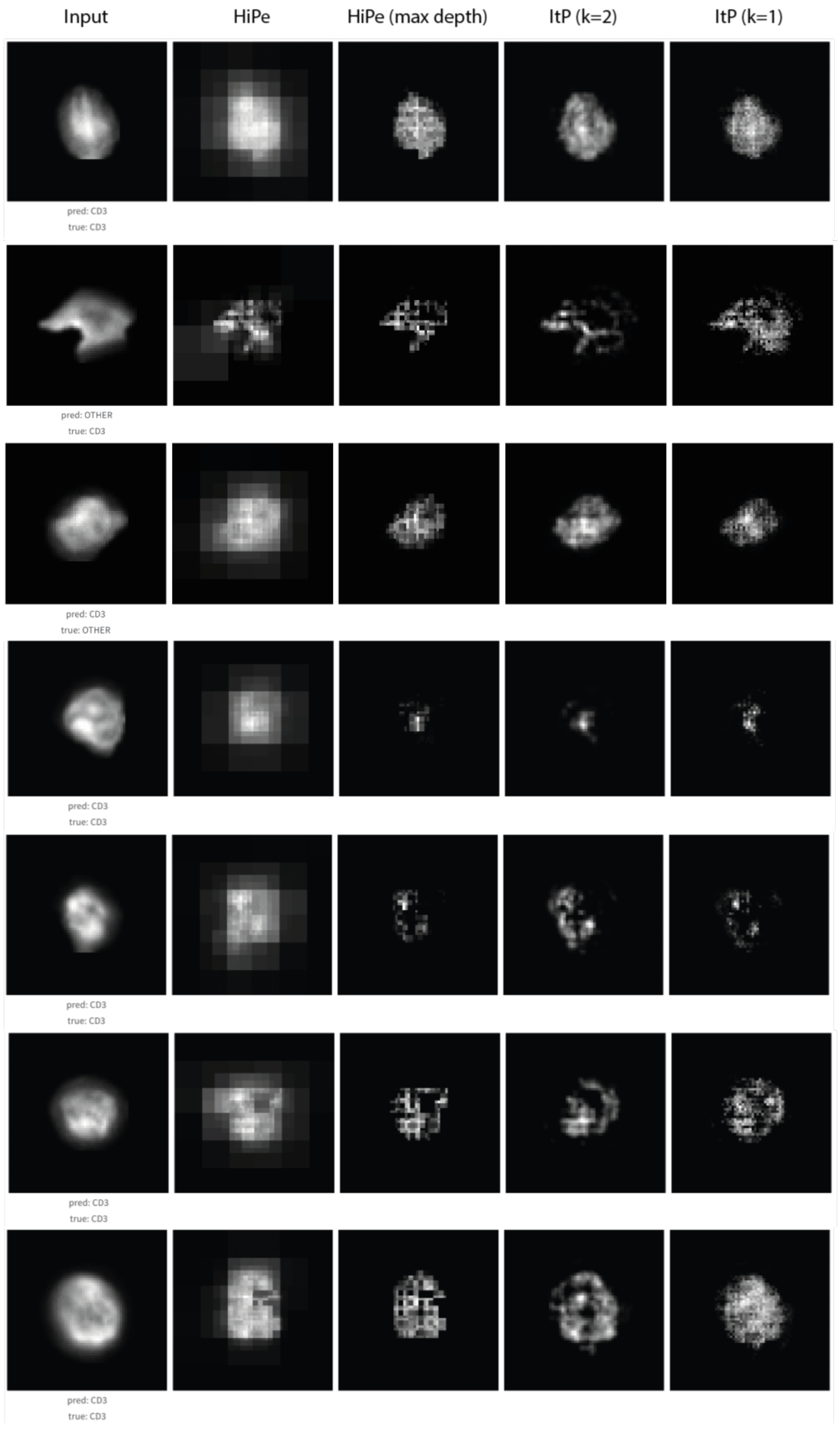


Figure 7.11: Saliency maps generated by Hierarchical Perturbation (HiPe) at both all depths and maximum depth only, plus saliency maps generated by standard Iterative Perturbation (ItP) with kernel sizes of 1 and 2.

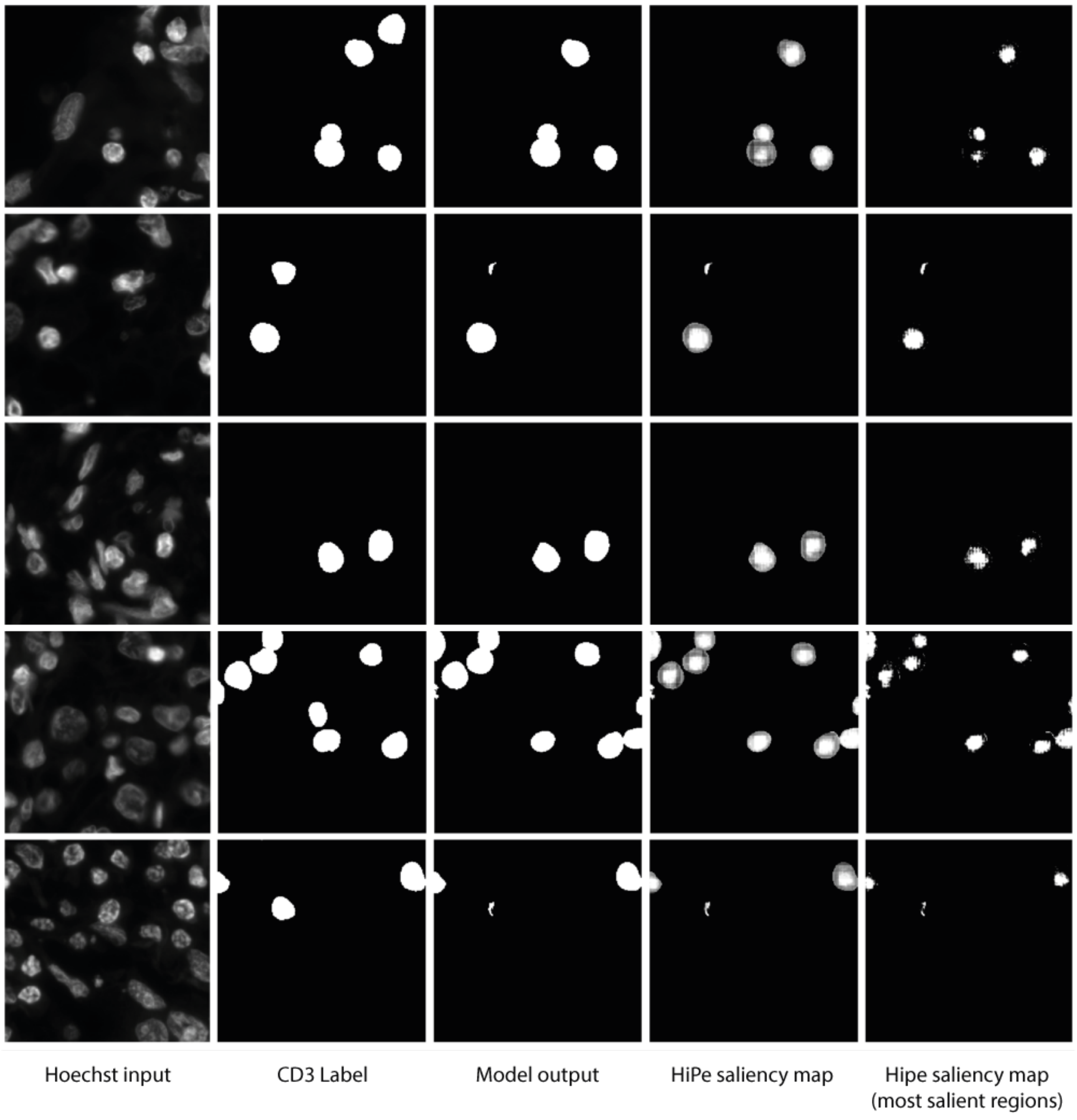


Figure 7.12: Examples of model predictions on five test set images – here I show the Hoechst 33342 stained input image, the ground truth segmentation mask, the predicted segmentation output, and HiPe saliency maps.

Chapter 8

Conclusion

8.1 Summary

In this thesis, I have explored the application of model-agnostic interpretability methods in deep learning, with a particular focus on their use in digital pathology. I began by introducing the concept of learned prototypes and demonstrating how feature visualisation can be used to understand what a model has learned from its training data. I showed that the choice of model architecture influences the features learned, and proposed a novel adaptation of feature visualisation to enable its application to segmentation models.

I then addressed the challenge of evaluating saliency mapping methods, proposing the Proxy Model Test as a way to objectively compare different approaches. This test uses a transparent proxy model for which the ground truth saliency is known, allowing for direct assessment of the accuracy of saliency maps. I introduced Hierarchical Perturbation (HiPe) as a novel saliency mapping technique that achieves state-of-the-art performance while being significantly (20$\times$) faster than existing methods.

I demonstrated the practical utility of these interpretability methods through two case studies in digital pathology. The first showed how saliency mapping can be used to generate pixel-level segmentations from weakly-supervised models, providing a cost-effective alternative to manual annotation. The second case study demonstrated how interpretability techniques can lead to new insights, by training a model to identify CD3-expressing cells from Hoechst stained images – a task previously thought to be impossible.

8.2 Contributions

The main contributions of this thesis are:

1. Demonstrating the influence of model architecture on learned features through feature visualisation, and proposing an adaptation of feature visualisation for segmentation models.
2. Introducing the Proxy Model Test as a way to objectively evaluate and compare saliency mapping methods.
3. Proposing Hierarchical Perturbation (HiPe), a novel model-agnostic saliency mapping method that achieves state-of-the-art performance while being significantly ($20\times$) faster than existing techniques.
4. Demonstrating the practical utility of interpretability methods in digital pathology, through case studies on weakly-supervised segmentation and the identification of CD3-expressing cells from Hoechst stained images.

Looking ahead, the potential for interpretability methods to enable discovery in digital pathology is particularly exciting. As I demonstrated in Chapter 7, interpretability techniques can uncover previously unknown relationships in data – in this case, between cellular morphology and protein expression – opening up new avenues for biomarker discovery and mechanistic understanding of disease. By providing a window into the learned representations of deep learning models, interpretability tools can help to bridge the gap between artificial and human intelligence in the analysis of complex medical data.

In conclusion, this work has advanced the state-of-the-art in model-agnostic interpretability for deep learning, with a particular focus on applications in digital pathology. The methods and case studies I have presented demonstrate the potential for these techniques to enhance the transparency, reliability, and discovery power of AI in healthcare. While further validation and translation work is needed, I believe the work I have presented here contributes to a more interpretable and impactful future for deep learning in medicine.

8.3 Limitations and Future Work

I see several promising directions for future work building on this thesis.

Firstly, while I demonstrated the influence of model architecture on learned features in Chapter 2, my experiments were limited to a small set of architectures and hyperparameters. A more comprehensive exploration of the design space, including different layer types, activation functions, and regularisation techniques, would provide a more complete understanding of how these factors impact learned representations. While somewhat outside the scope of this work, I think figuring out exactly how different architectures affect the *kinds* of features a model is able to learn from its training data is fascinating and important, as it has the potential to inform future model architecture design. Similarly, although I demonstrated the potential of interpretability methods for uncovering novel insights, I did not systematically explore how these insights could be used to improve model performance or robustness. Future work could investigate how both model-agnostic and model-specific interpretability can be integrated into the model development process, for example, by informing data augmentation strategies or guiding the selection of architectures and hyperparameters. This could lead to models that are inherently more interpretable, reducing the need for post-hoc explanations.

More broadly, while model-agnostic methods have a number of nice properties, there is scope for further work exploring the trade-offs between model-agnostic and model-specific interpretability techniques in different application domains. For knowledge discovery in particular, it would have been interesting to apply model-specific interpretability methods – i.e. visualising filters in the convolutional layers – to improve understanding of the model's learned features.

One limitation of this work is the lack of extensive clinical validation of the findings. While my results demonstrate the potential of interpretability methods in digital pathology, further work is needed to assess their robustness and generalisability across a wider range of datasets and clinical contexts, as well as collaborations with clinical partners to explore how interpretability techniques can be integrated into real-world diagnostic workflows. While my focus has been on digital pathology, the techniques I introduced could also potentially be applied to other types of computer vision in medicine, as well as to problems beyond healthcare where interpretability is important, such as autonomous vehicles or financial decision-making.

Additionally, while I demonstrated the potential of interpretability methods for generating insights and explanations, I did not conduct user studies to assess their practical utility for domain experts, such as pathologists. User studies could provide valuable feedback on the usability and effectiveness of the proposed methods in real-world settings and guide future refinements.

While these limitations are important to acknowledge, they also highlight the many opportunities for future work in this area. By addressing these challenges and continuing to develop more robust, efficient, and reliable interpretability methods, we can unlock the full potential of deep learning in digital pathology and beyond.

8.4 Clinical Assessment and Translation Potential

The methods and results I have presented in this thesis have significant potential for clinical translation in the field of digital pathology. The ability to generate pixel-level segmentations from weakly-supervised models, demonstrated in Chapter 6, could greatly reduce the time and cost associated with manual annotation of histopathology images. This could in turn accelerate the development and deployment of AI-based diagnostic tools, by enabling the creation of large-scale training datasets with reduced manual effort.

The discovery that CD3-expressing cells can be identified from Hoechst stained images, as I showed in Chapter 7, also has important clinical implications. This finding suggests that it may be possible to perform certain types of immune profiling using a much simpler and cheaper staining protocol than is currently standard. If validated on a larger scale, this could make immune profiling more widely accessible, potentially enabling more personalised treatment strategies for cancer patients.

However, it is important to note that further validation work is needed before the methods I have proposed can be translated into routine clinical use. The robustness and generalisability of the techniques need to be assessed across a wider range of datasets, representing different cancer types, staining protocols, and imaging platforms. The regulatory requirements for clinical deployment of AI-based tools would also need to be navigated, with rigorous evidence required to demonstrate safety and efficacy.

References

[1] Cancer image europe – the european federation for cancer images. https://cancerimage.eu/. Accessed: 2023-8-7.

[2] Communications of the ACM - january 2020 - 70. https://mags.acm.org/communications/january_2020/?folio=68&&pg=70. Accessed: 2023-8-8.

[3] CZI image file format. https://www.zeiss.com/microscopy/en/products/software/zeiss-zen/czi-image-file-format.html. Accessed: 2024-2-10.

[4] Endometrial carcinoma-general. https://www.pathologyoutlines.com/topic/uterusendometrialcarc.html. Accessed: 2024-1-21.

[5] IDR: Image data resource. https://idr.openmicroscopy.org/tissue/. Accessed: 2023-8-7.

[6] The pathology atlas. https://www.atlasantibodies.com/resources/human-protein-atlas/pathology-atlas/. Accessed: 2023-8-7.

[7] Welcome to the cancer imaging archive - the cancer imaging archive (TCIA). https://www.cancerimagingarchive.net/, September 2019. Accessed: 2023-8-7.

[8] *The innate and adaptive immune systems*. Institute for Quality and Efficiency in Health Care (IQWiG), July 2020.

[9] A Adadi and M Berrada. Peeking inside the Black-Box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 2018.

[10] J Adebayo, J Gilmer, Muelly M, I Goodfellow, Hardt M, and B Kim. Sanity checks for saliency maps. *NeurIPS*, 2018.

[11] Famke Aeffner, Mark D. Zarella, Nathan Buchbinder, Marilyn M. Bui, Matthew R. Goodman, Douglas J. Hartman, Giovanni M. Lujan, Mariam A. Molani, Anil V. Parwani, Kate Lillard, Oliver C. Turner, Venkata N.P. Vemuri, Ana G. Yuil-Valdes, and Douglas Bowman. Introduction to digital image analysis in whole-slide imaging: A white paper from the digital pathology association. *Journal of Pathology Informatics*, 10, 2022.

[12] Saad Ullah Akram, Juho Kannala, Lauri Eklund, and Janne Heikkilä. Cell segmentation proposal network for microscopy image analysis. In *Deep Learning and Data Labeling for Medical Applications*, Lecture notes in computer science, pages 21–29. Springer International Publishing, Cham, 2016.

[13] Shaimaa Al-Janabi, André Huisman, and Paul J Van Diest. Digital pathology: current status and future perspectives. *Histopathology*, 61(1):1–9, July 2012.

[14] Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. Explainable artificial intelligence (XAI): What we know and what is left to attain trustworthy artificial intelligence. *Inf. Fusion*, 99:101805, November 2023.

[15] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J Big Data*, 8(1):53, March 2021.

[16] M Angelova, B Mlecnik, A Vasaturo, G Bindea, T Fredriksen, L Lafontaine, B Buttard, E Morgand, D Bruni, A Jouret-Mourin, C Hubert, A Kartheuser, Y Humblet, M Ceccarelli, N Syed, F M Marincola, D Bedognetti, M Van den Eynde, and J Galon. Evolution of metastases in space and time under immune selection. *Cell*, 175(3):751–765, 2018.

[17] Maria-Gabriela Anitei, Guy Zeitoun, Bernhard Mlecnik, Florence Marliot, Nacilla Haicheur, Ana-Maria Todosi, Amos Kirilovsky, Christine Lagorce, Gabriela Bindea, Dan Ferariu, Mihai Danciu, Patrick Bruneval, Viorel Scripcariu, Jean-Marc Chevallier, Franck Zinzindohoué, Anne Berger, Jérôme Galon, and Franck Pagès. Prognostic and predictive values of the immunoscore in patients with rectal cancer. *Clin. Cancer Res.*, 20(7):1891–1899, April 2014.

[18] Bing Bai, Jian Liang, Guanhua Zhang, Hao Li, Kun Bai, and Fei Wang. Why attentions may not be interpretable? *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14, 2021, Virtual Event, Singapore*, 1(1), August 2021.

[19] Alyson J Balkwill, William T Barry, Franz Cap, Kevin N Christie, Babak Ehteshami Bejnordi, David J Foran, David A Gutman, Brian Helba, Linda Joseph, Anna Kalinovsky, et al. Deep learning for digital pathology image analysis: a survey. *Annual Review of Pathology: Mechanisms of Disease*, 17:77–102, 2022.

[20] P Bankhead, M B Loughrey, J A Fernández, Y Dombrowski, D G McArt, P D Dunne, S McQuaid, R T Gray, L J Murray, H G Coleman, J A James, M Salto-Tellez, and P W Hamilton. QuPath: Open source software for digital pathology image analysis. *Scientific Reports*, 7(1):16878, 2017.

[21] A Barredo Arrieta, N Diaz-Rodriguez, J Del Ser, A Bennetot, S Tabik, A Barbado, S Garcia, S Gil-Lopez, D Molina, R Benjamins, R Chatila, and F Herrera. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[22] Hritam Basak, Rajarshi Bhattacharya, Rukhshanda Hussain, and Agniv Chatterjee. An exceedingly simple consistency regularization method for Semi-Supervised medical image segmentation. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pages 1–4. IEEE, March 2022.

[23] Jasmijn Bastings and Katja Filippova. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, Stroudsburg, PA, USA, 2020. Association for Computational Linguistics.

[24] Etienne Becht, Nicolas A Giraldo, Marie-Caroline Dieu-Nosjean, Catherine Sautès-Fridman, and Wolf Herman Fridman. Cancer immune contexture and immunotherapy. *Curr. Opin. Immunol.*, 39:7–13, April 2016.

[25] Francesco Bianconi, Jakob N Kather, and Constantino Carlos Reyes-Aldasoro. Experimental assessment of color deconvolution and color normalization for automated classification of histology images stained with hematoxylin and eosin. *Cancers*, 12(11):3337, November 2020.

[26] Adrien Bibal, Rémi Cardon, David Alfter, Rodrigo Wilkens, Xiaoou Wang, Thomas François, and Patrick Watrin. Is attention explanation? an introduction to the debate. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3889–3900, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[28] T J Brinker, A Hekler, A H Enk, J Klode, A Hauschild, C Berking, B Schilling, S Haferkamp, D Schadendorf, T Holland-Letz, J S Utikal, C von Kalle, and Collaborators. Deep learning outperformed 136 of 157 dermatologists in a head-to-head dermoscopic melanoma image classification task. *European Journal of Cancer*, 113:47–54, 2019.

[29] D Bruni, H K Angell, and J Galon. The immune contexture and immunoscore in cancer prognosis and therapeutic efficacy. *Nature Reviews Cancer*, 20(11):662–680, 2020.

[30] L Brunke, p Agrawal, and N George. Evaluating input perturbation methods for interpreting cnns and saliency map comparison. In *European Conference on Computer Vision (ECCV)*, volume 12535 of *Lecture Notes in Computer Science*, pages 120–134. Springer International Publishing, 2020.

[31] Rosaria Cammarota, Valentina Bertolini, Giuseppina Pennesi, Eraldo O Bucci, Ornella Gottardi, Cecilia Garlanda, Luigi Laghi, Massimo C Barberis, Fausto Sessa, Douglas M Noonan, and Adriana Albini. The tumor microenvironment of colorectal cancer: stromal TLR-4 expression as a potential prognostic marker. *J. Transl. Med.*, 8:112, November 2010.

[32] Gabriele Campanella, Matthew G Hanna, Luke Geneslaw, Allen Miraflor, Vitor Werneck Krauss Silva, Klaus J Busam, Edi Brogi, Victor E Reuter, David S Klimstra, and Thomas J Fuchs. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. *Nat. Med.*, 25(8):1301–1309, August 2019.

[33] K H Cha, L M Hadjiiski, R K Samala, H Chan, R H Cohan, E M Caoili, C Paramagul, A Alva, and A Z Weizer. Bladder cancer segmentation in CT for treatment response assessment: Application of Deep-Learning convolution neural Network-A pilot study. *Tomography*, 2(4):421–429, 2016.

[34] B Chazotte. Labeling nuclear DNA using DAPI. *Cold Spring Harbor Protocols*, 2011(1):db.prot5556, 2011.

[35] B Chazotte. Labeling nuclear DNA with hoechst 33342. *Cold Spring Harbor Protocols*, 2011(1):db.prot5557, 2011.

[36] P Dabkowski and Y Gal. Real time image saliency for black box classifiers. *NeurIPS*, 2017.

[37] Li Deng. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, 3:e2, January 2014.

[38] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. February 2017.

[39] F K Došilović, M Brčić, and N Hlupić. Explainable artificial intelligence: A survey. In *International Convention on Information and Communication Technology, Electronics and Microelectronics*, pages 0210–0215. IEEE, 2018.

[40] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. October 2020.

[41] Bojan Drobic, Katherine L Dunn, Paula S Espino, and James R Davie. Abnormalities of chromatin in tumor cells. *EXS*, (96):25–47, 2006.

[42] Paul Ekman, Richard J. Davidson, and Wallace V. Friesen. The duchenne smile: Emotional expression and brain physiology ii. *Journal of Personality and Social Psychology*, 64(2):280–292, 1993.

[43] D Erhan, Y Bengio, A Courville, and P Vincent. Visualizing Higher-Layer features of a deep network. DIRO, Universite de Montreal, 2009.

[44] Nils Feldhus, Leonhard Hennig, Maximilian Dustin Nasert, Christopher Ebert, and Robert Schwarzenberg Sebastian Möller. SMV: Code and data for the ACL 2023 NLReasoning workshop paper "saliency map verbalization: Comparing feature importance representations from model-free and instruction-based methods" (feldhus et al., 2023).

[45] Edgar G Fischer. Nuclear morphology and the biology of cancer cells. *Acta Cytol.*, 64(6):511–519, June 2020.

[46] R Fong, M Patrick, and A Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019.

[47] R Fong and A Vedaldi. Explanations for attributing deep neural network predictions. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 149–167. Springer International Publishing, 2019.

[48] R C Fong and A Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437. IEEE, 2017.

[49] Filippo Fraggetta, Salvatore Garozzo, Gian Franco Zannoni, Liron Pantanowitz, and Esther Diana Rossi. Routine digital pathology workflow: The catania experience. *Journal of Pathology Informatics*, 8, 2017.

[50] Wolf-Herman Fridman, Marie-Caroline Dieu-Nosjean, Franck Pagès, Isabelle Cremer, Diane Damotte, Catherine Sautès-Fridman, and Jérôme Galon. The immune microenvironment of human tumors: general significance and clinical impact. *Cancer Microenviron.*, 6(2):117–122, August 2013.

[51] MY Gabril and GM Yousef. Informatics for practicing anatomical pathologists: marking a new era in pathology practice. *Modern pathology*, 23(1):349–358, 2010.

[52] J Galon and A Lanzi. Immunoscore and its introduction in clinical practice. *The Quarterly Journal of Nuclear Medicine and Molecular Imaging*, 64(2):152–161, 2020.

[53] J Galon, B Mlecnik, G Bindea, H K Angell, A Berger, C Lagorce, A Lugli, I Zlobec, A Hartmann, C Bifulco, I D Nagtegaal, R Palmqvist, G V Masucci, G Botti, F Tatangelo, P Delrio, M Maio, L Laghi, F Grizzi, M Asslaber, C D'Arrigo, F Vidal-Vanaclocha, E Zavadova, L Chouchane, P S Ohashi, S Hafezi-Bakhtiari, B G Wouters, M Roehrl, L Nguyen, Y Kawakami, S Hazama, K Okuno, S Ogino, P Gibbs, P Waring, N Sato, T Torigoe, K Itoh, P S Patel, S N Shukla, Y Wang, S Kopetz, F A Sinicrope, V Scripcariu, P A Ascierto, F M Marincola, B A Fox, and F Pagès. Towards the introduction of the

'immunoscore' in the classification of malignant tumours. *The Journal of Pathology*, 232(2):199–209, 2014.

[54] Evan George. Occupational hazard for pathologists: microscope use and musculoskeletal disorders. *Am. J. Clin. Pathol.*, 133(4):543–548, April 2010.

[55] L H Gilpin, D Bau, B Z Yuan, A Bajwa, M Specter, and L Kagal. Explaining explanations: An overview of interpretability of machine learning. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89. IEEE, 2018.

[56] Aditya Golatkar, Deepak Anand, and Amit Sethi. Classification of breast cancer histology using deep learning. February 2018.

[57] Johanna Goldberg. LibGuides: Artificial intelligence: Datasets and repositories. April 2019.

[58] Cecil J Gomes, Michael W Harman, Sara M Centuori, Charles W Wolgemuth, and Jesse D Martinez. Measuring DNA content in live cells by fluorescence microscopy. *Cell Div.*, 13(1):1–10, September 2018.

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[60] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv [stat.ML]*, June 2014.

[61] Jianyuan Guo, Kai Han, Han Wu, Yehui Tang, Xinghao Chen, Yunhe Wang, and Chang Xu. CMT: Convolutional neural networks meet vision transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022.

[62] Z Hameed, B Garcia-Zapirain, J J Aguirre, and M A Isaza-Ruget. Multiclass classification of breast cancer histopathology images using multilevel features of deep convolutional neural network. *Scientific Reports*, 12(1):15600, 2022.

[63] K He, X Zhang, S Ren, and J Sun. Deep residual learning for image recognition. *arXiv*, 2015.

[64] K He, X Zhang, S Ren, and J Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[65] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[66] Achim Hekler, Jochen S Utikal, Alexander H Enk, Wiebke Solass, Max Schmitt, Joachim Klode, Dirk Schadendorf, Wiebke Sondermann, Cindy Franklin, Felix Bestvater, Michael J Flaig, Dieter Krahl, Christof von Kalle, Stefan Fröhling, and Titus J Brinker. Deep learning outperformed 11 pathologists in the classification of histopathological melanoma images. *Eur. J. Cancer*, 118:91–96, September 2019.

[67] Yuki Hiramatsu, Kazuhiro Hotta, Ayako Imanishi, Michiyuki Matsuda, and Kenta Terai. Cell image segmentation by integrating multiple CNNs. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2286–22866. IEEE, June 2018.

[68] P Iakubovskii. segmentation_models.pytorch: Segmentation models with pretrained backbones. PyTorch.

[69] Maximilian Ilse, Jakub M Tomczak, and Max Welling. Attention-based deep multiple instance learning. February 2018.

[70] Sarthak Jain and Byron C Wallace. Attention is not explanation. February 2019.

[71] Gabriel Jiménez and Daniel Racoceanu. Deep learning for semantic segmentation vs. classification in computational pathology: Application to mitosis analysis in breast cancer grading. *Front Bioeng Biotechnol*, 7:145, June 2019.

[72] J Kaplan, S McCandlish, T Henighan, T B Brown, B Chess, R Child, S Gray, A Radford, J Wu, and D Amodei. Scaling laws for neural language models. *arXiv*, 2020.

[73] P Kindermans, S Hooker, J Adebayo, M Alber, K T Schütt, S Dähne, D Erhan, and B Kim. The (un)reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer International Publishing, 2019.

[74] R Koga, N Hashimoto, T Yokota, M Nakaguro, K Kohno, S Nakamura, I Takeuchi, and H Hontani. Detection of DLBCL regions in H&E stained whole slide pathology images using bayesian U-Net. In *International Forum on Medical Imaging in Asia*, volume 11792, page 1179203. International Society for Optics and Photonics, 2021.

[75] Maya Krishnan. Against interpretability: a critical examination of the interpretability problem in machine learning. *Philos. Technol.*, 33(3):487–502, September 2020.

[76] Miroslav Kubat. *An Introduction to Machine Learning*. Springer International Publishing, September 2017.

[77] V. Kumar, A.K. Abbas, and J.C. Aster. Robbins basic pathology. 2018.

[78] S Lal, D Das, K Alabhya, A Kanfade, A Kumar, and J Kini. NucleiSegNet: Robust deep learning architecture for the nuclei segmentation of liver cancer histopathology images. *Computers in Biology and Medicine*, 128:104075, 2021.

[79] Richard M Levenson, Alexander D Borowsky, and Michael Angelo. Immunohistochemistry and mass spectrometry for highly multiplexed cellular molecular imaging. *Lab. Invest.*, 95(4):397–405, April 2015.

[80] Z Li, J Zhang, T Tan, X Teng, X Sun, H Zhao, L Liu, Y Xiao, B Lee, Y Li, Q Zhang, S Sun, Y Zheng, J Yan, N Li, Y Hong, J Ko, H Jung, Y Liu, Y Chen, C Wang, V Yurovskiy, P Maevskikh, V Khanagha, Y Jiang, L Yu, Z Liu, D Li, P J Schuffler, Q Yu, H Chen,

Y Tang, and G Litjens. Deep learning methods for lung cancer segmentation in Whole-Slide histopathology Images-The ACDC@LungHP challenge 2019. *Journal of Biomedical and Health Informatics*, 25(2):429–440, 2021.

[81] Anna Ligasová and Karel Koberna. DNA Dyes-Highly sensitive reporters of cell quantification: Comparison with other cell quantification methods. *Molecules*, 26(18), September 2021.

[82] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):318–327, February 2020.

[83] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context. May 2014.

[84] P Linardatos, V Papastefanopoulos, and S Kotsiantis. Explainable AI: A review of machine learning interpretability methods. *Entropy*, 23(1), 2020.

[85] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queueing Syst.*, 16(3):31–57, June 2018.

[86] G Litjens, T Kooi, B E Bejnordi, A A A Setio, F Ciompi, M Ghafoorian, J A W M van der Laak, and C I van Ginneken, Band Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[87] Y Liu, K Gadepalli, M Norouzi, GE Dahl, T Kohlberger, A Boyko, S Venugopalan, A Timofeev, PQ Nelson, GS Corrado, et al. Detecting cancer metastases on gigapixel pathology images. *arXiv preprint arXiv:1703.02442*, 2017.

[88] Yibing Liu, Haoliang Li, Yangyang Guo, Chenqi Kong, Jing Li, and Shiqi Wang. Rethinking attention-model explainability through faithfulness violation test. January 2022.

[89] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. November 2014.

[90] I Loshchilov and F Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*, 2017.

[91] M Y Lu, D F K Williamson, T Y Chen, R J Chen, M Barbieri, and F Mahmood. Data-efficient and weakly supervised computational pathology on whole-slide images. *Nature Biomedical Engineering*, 5(6):555–570, 2021.

[92] A Mahbod, G Schaefer, I Ellinger, R Ecker, Ö Smedby, and C Wang. A Two-Stage U-Net algorithm for segmentation of nuclei in H&E-Stained tissues. In *European Congress on Digital Pathology (ECDP)*, pages 75–82. Springer Verlag, 2019.

[93] A Mahendran and A Vedaldi. Salient deconvolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 120–135. Springer International Publishing, 2016.

[94] Vicky Makker, Helen MacKay, Isabelle Ray-Coquard, Douglas A Levine, Shannon N Westin, Daisuke Aoki, and Ana Oaknin. Endometrial cancer. *Nat Rev Dis Primers*, 7(1):88, December 2021.

[95] R Manikandan, A Kumar, and D Gupta. *Chapter 5 - Hybrid computational intelligence for healthcare and disease diagnosis*. Academic Press, 2020.

[96] Payden McBee, Fatima Zulqarnain, Sana Syed, and Donald E Brown. Image-Level uncertainty in Pseudo-Label selection for Semi-Supervised segmentation. *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, 2022:4740–4744, July 2022.

[97] T Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[98] B Mlecnik, M Van den Eynde, G Bindea, S E Church, A Vasaturo, T Fredriksen, L Lafontaine, N Haicheur, F Marliot, D Debetancourt, G Pairet, A Jouret-Mourin, J Gigot,

C Hubert, E Danse, C Dragean, J Carrasco, Y Humblet, V Valge-Archer, A Berger, F Pagès, J Machiels, and J Galon. Comprehensive intrametastatic immune quantification and major impact of immunoscore on survival. *Journal of the National Cancer Institute (JNCI)*, 110(1), 2018.

[99] Mahnaz Mohammadi, Jessica Cooper, Ognjen Arandelović, Christina Fell, David Morrison, Sheeba Syed, Prakash Konanahalli, Sarah Bell, Gareth Bryson, David J Harrison, and David Harris-Birtill. Weakly supervised learning and interpretability for endometrial whole slide image diagnosis. *Exp. Biol. Med.*, 247(22):2025–2037, November 2022.

[100] C Molnar. 10.1 learned features. `https://christophm.github.io/interpretable-ml-book/cnn-features.html`, 2021. Accessed: 2021-12-21.

[101] C Molnar. Chapter 5 Model-Agnostic methods. `https://christophm.github.io/interpretable-ml-book/agnostic.html`, 2021. Accessed: 2021-7-6.

[102] Christoph Molnar. 3.3 scope of interpretability. `https://christophm.github.io/interpretable-ml-book/scope-of-interpretability.html`, December 2022. Accessed: 2023-1-14.

[103] Christoph Molnar. 3.2 taxonomy of interpretability methods. `https://christophm.github.io/interpretable-ml-book/taxonomy-of-interpretability-methods.html`, July 2023. Accessed: 2023-8-8.

[104] Christoph Molnar. Chapter 5 interpretable models. `https://christophm.github.io/interpretable-ml-book/simple.html`, July 2023. Accessed: 2023-8-8.

[105] A Mordvintsev, C Olah, and M Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog (2015)*, 2015.

[106] W J Murdoch, C Singh, K Kumbier, R Abbasi-Asl, and B Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences (PNAS)*, 116(44):22071–22080, 2019.

[107] Kunal Nagpal, Davis Foote, Fraser Tan, Yun Liu, Po-Hsuan Cameron Chen, David F Steiner, Naren Manoj, Niels Olson, Jenny L Smith, Arash Mohtashamian, Brandon Peterson, Mahul B Amin, Andrew J Evans, Joan W Sweet, Carol Cheung, Theodorus van der Kwast, Ankur R Sangoi, Ming Zhou, Robert Allan, Peter A Humphrey, Jason D Hipp, Krishna Gadepalli, Greg S Corrado, Lily H Peng, Martin C Stumpe, and Craig H Mermel. Development and validation of a deep learning algorithm for gleason grading of prostate cancer from biopsy specimens. *JAMA Oncol*, 6(9):1372–1380, September 2020.

[108] Soojeong Nam, Yosep Chong, Chan Kwon Jung, Tae-Yeong Kwak, Ji Youl Lee, Jihwan Park, Mi Jung Rho, and Heounjeong Go. Introduction to digital pathology and computer-aided pathology. *J Pathol Transl Med*, 54(2):125–134, March 2020.

[109] Sajid Nazir, Diane M Dickson, and Muhammad Usman Akram. Survey of explainable artificial intelligence techniques for biomedical imaging with deep neural networks. *Comput. Biol. Med.*, 156:106668, April 2023.

[110] Birger Neuhaus, Thomas Schmid, and Jens Riedel. Zootaxa. In *Zootaxa*, volume 4322, pages 1–173. Magnolia Press, 2017.

[111] A Nguyen, A Dosovitskiy, J Yosinski, T Brox, and J Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 3395–3403. Curran Associates Inc., 2016.

[112] A Nguyen, J Yosinski, and J Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv*, 2016.

[113] W Nie, Y Zhang, and A Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *Proceedings of Machine Learning Research*, volume 80 of *Proceedings of Machine Learning Research*, pages 3809–3818. PMLR, 2018.

[114] S Nikolov, S Blackwell, A Zverovitch, R Mendes, M Livne, J De Fauw, Y Patel, C Meyer, H Askham, B Romera-Paredes, C Kelly, A Karthikesalingam, C Chu, D Carnell, C Boon, D D'Souza, S A Moinuddin, B Garie, Y McQuinlan, S Ireland, K Hampton, K Fuller, H Montgomery, G Rees, M Suleyman, T Back, C O Hughes, J R Ledsam, and O Ronneberger. Clinically applicable segmentation of head and neck anatomy for radiotherapy: Deep learning algorithm development and validation study. *Journal of Medical Internet Research (JMIR)*, 23(7):e26151, 2021.

[115] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, September 2021.

[116] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention U-Net: Learning where to look for the pancreas. *arXiv e-prints*, page arXiv:1804.03999, April 2018.

[117] C Olah, A Mordvintsev, and L Schubert. Feature visualization. *Distill*, 2(11), 2017.

[118] Kay R J Oskal, Martin Risdal, Emilius A M Janssen, Erling S Undersrud, and Thor O Gulsrud. A u-net based approach to epidermal tissue segmentation in whole slide histopathological images. *SN Applied Sciences*, 1(7):672, June 2019.

[119] F Otto and K C Tsou. A comparative study of DAPI, DIPI, and hoechst 33258 and 33342 as chromosomal DNA stains. *Stain Technology*, 60(1):7–11, 1985.

[120] Lakshmi Narayan Pandey, Rahul Vashisht, and Harish G Ramaswamy. On the interpretability of attention networks. In *14th Asian Conference on Machine Learning*. unknown, December 2022.

[121] E R Parra, N Uraoka, M Jiang, P Cook, D Gibbons, M Forget, C Bernatchez, C Haymaker, I I Wistuba, and J Rodriguez-Canales. Validation of multiplex immunofluorescence panels using multispectral microscopy for immune-profiling of formalin-fixed and paraffin-embedded human tumor tissues. *Scientific Reports*, 7(1):13380, 2017.

[122] Prathyush S Parvatharaju, Ramesh Doddaiah, Thomas Hartvigsen, and Elke A Rundensteiner. Learning saliency maps to explain deep time series classifiers. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, pages 1406–1415, New York, NY, USA, October 2021. Association for Computing Machinery.

[123] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly, 2017.

[124] V Petsiuk, A Das, and K Saenko. Rise: Randomized input sampling for explanation of black-box models. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVC, 2018.

[125] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models, Sep 2018.

[126] S S Raab. The Cost-Effectiveness of immunohistochemistry. *Archives of Pathology & Laboratory Medicine*, 124(8):1185–1191, 2000.

[127] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? August 2021.

[128] M T Ribeiro, S Singh, and C Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144. ACM, 2016.

[129] O Ronneberger, P Fischer, and T Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer International Publishing, 2015.

[130] Jerome rony. Deep weakly-supervised learning methods for classification and localization in histology images: a survey. `https://www.groundai.com/project/deep-weakly-supervised-learning-methods-for-classification-and-localization-1`, September 2019. Accessed: 2019-12-2.

[131] C Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[132] Stuart Jonathan Russell, Peter Norvig, and Ernest Davis. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

[133] Rabia Saleem, Bo Yuan, Fatih Kurugollu, Ashiq Anjum, and Lu Liu. Explaining deep neural networks: A survey on the global interpretation methods. *Neurocomputing*, 513:165–180, November 2022.

[134] Divya Saxena and Jiannong Cao. Generative adversarial networks (GANs): Challenges, solutions, and future directions. *ACM Computing Surveys*, 54(3):1–42, May 2021.

[135] U Schmidt, M Weigert, C Broaddus, and G Myers. Cell detection with Star-Convex polygons. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 265–273. Springer International Publishing, 2018.

[136] J Schneider, C Mesk, and M Vlachos. Deceptive AI explanations: Creation and detection. In *Proceedings of the 14th International Conference on Agents and Artificial Intelli-*

gence, ICAART 2022, Volume 2, Online Streaming, February 3-5, 2022, pages 44–55. SCITEPRESS, 2022.

[137] C Schorr, P Goodarzi, F Chen, and T Dahmen. Neuroscope: An explainable AI toolbox for semantic segmentation and image classification of convolutional neural nets. *NATO Adv. Sci. Inst. Ser. E Appl. Sci.*, 11(5):2199, 2021.

[138] R R Selvaraju, M Cogswell, A Das, R Vedantam, D Parikh, and D Batra. Grad-CAM: Visual explanations from deep networks via Gradient-Based localization. In *IEEE International Conference on Computer Vision (ICCV)*, pages 618–626. IEEE, 2017.

[139] Reshma Shakya, Tam Hong Nguyen, Nigel Waterhouse, and Rajiv Khanna. Immune contexture analysis in immuno-oncology: applications and challenges of multiplex fluorescent immunohistochemistry. *Clin Transl Immunology*, 9(10):e1183, October 2020.

[140] Yash Sharma, Lubaina Ehsan, Sana Syed, and Donald E Brown. HistoTransfer: Understanding transfer learning for histopathology. June 2021.

[141] Rui Shi, Tianxing Li, Liguo Zhang, and Yasushi Yamaguchi. Visualization comparison of vision transformers and convolutional neural networks. *IEEE Trans. Multimedia*, pages 1–13, 2023.

[142] Juliensimon Julien Simon. Large language models: A new moore's law? `https://huggingface.co/blog/large-language-models`. Accessed: 2023-8-8.

[143] K Simonyan, A Vedaldi, and A Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations (ICLR) Workshop*. ICLR, 2014.

[144] Sushant Singh and Ausif Mahmood. The NLP cookbook: Modern recipes for transformer based deep learning architectures. *IEEE Access*, 9:68675–68702, 2021.

[145] Gaurav Sohaliya and Kapil Sharma. Semantic segmentation using generative adversarial networks with a feature reconstruction loss. In *2021 Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–7. IEEE, August 2021.

[146] Nasim Souly, Concetto Spampinato, and Mubarak Shah. Semi supervised semantic segmentation using generative adversarial network. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, October 2017.

[147] J T Springenberg, A Dosovitskiy, T Brox, and M A Riedmiller. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations Workshop Proceedings*. ICLR, 2015.

[148] Chetan L Srinidhi, Seung Wook Kim, Fu-Der Chen, and Anne L Martel. Self-supervised driven consistency training for annotation efficient histopathology image analysis. *Med. Image Anal.*, 75:102256, January 2022.

[149] F Sultana, A Sufian, and P Dutta. Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, 201-202:106062, 2020.

[150] W C C Tan, S N Nerurkar, H Y Cai, H H M Ng, D Wu, Y T F Wee, J C T Lim, J Yeong, and T K H Lim. Overview of multiplex immunohistochemistry/immunofluorescence techniques in the era of cancer immunotherapy. *Cancer Communications*, 40(4):135–153, 2020.

[151] J M Taube, G Akturk, M Angelo, E L Engle, S Gnjatic, S Greenbaum, N F Greenwald, C V Hedvat, T J Hollmann, J Juco, E R Parra, M C Rebelatto, D L Rimm, J Rodriguez-Canales, K A Schalper, E C Stack, C S Ferreira, K Korski, A Lako, S J Rodig, E Schenck, K E Steele, M J Surace, M T Tetzlaff, K von Loga, I I Wistuba, C B Bifulco, and Society for Immunotherapy of Cancer (SITC) Pathology Task Force. The society for immunotherapy of cancer statement on best practices for multiplex immunohistochemistry (IHC) and

immunofluorescence (IF) staining and validation. *The Journal for ImmunoTherapy of Cancer (JITC)*, 8(1), 2020.

[152] Y Tay, M Dehghani, S Abnar, H W Chung, W Fedus, J Rao, S Narang, V Q Tran, D Yogatama, and D Metzler. Scaling laws vs model architectures: How does inductive bias influence scaling? *arXiv*, 2022.

[153] The Royal College of Pathologists. The pathology workforce. `https://www.rcpath.org/discover-pathology/public-affairs/the-pathology-workforce.html`. Accessed: 2023-8-8.

[154] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE Trans Neural Netw Learn Syst*, 32(11):4793–4813, November 2021.

[155] S Trebeschi, J J M van Griethuysen, D M J Lambregts, M J Lahaye, C Parmar, F C H Bakers, N H G M Peters, R G H Beets-Tan, and H J W L Aerts. Deep learning for Fully-Automated localization and segmentation of rectal cancer on multiparametric MR. *Scientific Reports*, 7(1):5301, 2017.

[156] Mathias Uhlen, Cheng Zhang, Sunjae Lee, Evelina Sjöstedt, Linn Fagerberg, Gholamreza Bidkhori, Rui Benfeitas, Muhammad Arif, Zhengtao Liu, Fredrik Edfors, Kemal Sanli, Kalle von Feilitzen, Per Oksvold, Emma Lundberg, Sophia Hober, Peter Nilsson, Johanna Mattsson, Jochen M Schwenk, Hans Brunnström, Bengt Glimelius, Tobias Sjöblom, Per-Henrik Edqvist, Dijana Djureinovic, Patrick Micke, Cecilia Lindskog, Adil Mardinoglu, and Fredrik Ponten. A pathology atlas of the human cancer transcriptome. *Science*, 357(6352), August 2017.

[157] Unknown. Introduction to digital image analysis in whole-slide imaging: a white paper from the digital pathology association. *ScienceDirect*, 2022.

[158] M Van den Eynde, B Mlecnik, G Bindea, T Fredriksen, S E Church, L Lafontaine, N Haicheur, F Marliot, M Angelova, A Vasaturo, D Bruni, A Jouret-Mourin, P Baldin,

N Huyghe, K Haustermans, A Debucquoy, E Van Cutsem, J Gigot, C Hubert, A Kartheuser, C Remue, D Léonard, V Valge-Archer, F Pagès, J Machiels, and J Galon. The link between the multiverse of immune microenvironments in metastases and the survival of colorectal cancer patients. *Cancer Cell*, 34(6):1012–1026.e3, 2018.

[159] Jeroen van der Laak, Geert Litjens, and Francesco Ciompi. Deep learning in histopathology: the path to the clinic. *Nat. Med.*, 27(5):775–784, May 2021.

[160] Amélie Viratham Pulsawatdi, Stephanie G Craig, Victoria Bingham, Kris McCombe, Matthew P Humphries, Seedevi Senevirathne, Susan D Richman, Phil Quirke, Leticia Campo, Enric Domingo, Timothy S Maughan, Jacqueline A James, and Manuel Salto-Tellez. A robust multiplex immunofluorescence and digital pathology workflow for the characterisation of the tumour immune microenvironment. *Mol. Oncol.*, 14(10):2384–2402, October 2020.

[161] D Wang, A Khosla, R Gargeya, H Irshad, and AH Beck. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*, 2016.

[162] Juanyan Wang and Mustafa Bilgic. Iit-ml.

[163] Lianjie Wang, Hui Geng, Yujie Liu, Lei Liu, Yanhua Chen, Fanchen Wu, Zhiyi Liu, Shiliang Ling, Yan Wang, and Lihong Zhou. Hot and cold tumors: Immunological features and the therapeutic strategies. *MedComm (2020)*, 4(5):e343, October 2023.

[164] Roni Wilentzik Müller and Irit Gat-Viks. Exploring neural networks and related visualization techniques in gene expression data. *Front. Genet.*, 11:402, May 2020.

[165] Eleanor Williams, Josh Moore, Simon W Li, Gabriella Rustici, Aleksandra Tarkowska, Anatole Chessel, Simone Leo, Bálint Antal, Richard K Ferguson, Ugis Sarkans, Alvis Brazma, Rafael E Carazo Salas, and Jason R Swedlow. The image data resource: A bioimage data integration and publication platform. *Nat. Methods*, 14(8):775–781, August 2017.

[166] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017.

[167] Yutong Xie, Bing Yang, Qingbiao Guan, Jianpeng Zhang, Qi Wu, and Yong Xia. Attention mechanisms in medical image segmentation: A survey. May 2023.

[168] F Xing and L Yang. Robust Nucleus/Cell detection and segmentation in digital pathology and microscopy images: A comprehensive review. *IEEE Reviews in Biomedical Engineering*, 9:234–263, 2016.

[169] Jun Xu, Chao Zhou, Bing Lang, and Qingshan Liu. Deep learning for histopathological image analysis: Towards computerized diagnosis on cancers. In Le Lu, Yefeng Zheng, Gustavo Carneiro, and Lin Yang, editors, *Deep Learning and Convolutional Neural Networks for Medical Image Computing: Precision Medicine, High Performance and Large-Scale Datasets*, Advances in computer vision and pattern recognition, pages 73–95. Springer International Publishing, Cham, 2017.

[170] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, November 2020.

[171] M D Zeiler and R Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer International Publishing, 2014.

[172] J Zhang, S A Bargal, Z Lin, J Brandt, X Shen, and S Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

[173] J Zhang, S A Bargal, Z Lin, Jo Brandt, X Shen, and S Sclaroff. Top-Down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

[174] Ling Zhang, Le Lu, Isabella Nogues, Ronald M Summers, Shaoxiong Liu, and Jianhua Yao. DeepPap: Deep convolutional networks for cervical cell classification. *IEEE J Biomed Health Inform*, 21(6):1633–1643, November 2017.

[175] Y Zhou, S Booth, M T Ribeiro, and J Shah. Do feature attribution methods correctly attribute features? *arXiv*, 2021.