Exploring Methods to Assess Scanner-Induced Domain Shifts in Medical Imaging

Bachelor Thesis

Bachelor of Science in Applied Computer Science

Kian Hinke

June 24, 2025

Supervisor:

1st: Prof. Dr. Christian Ledig
2nd: Sebastian Doerrich, M.Sc.

Chair of Explainable Machine Learning
Faculty of Information Systems and Applied Computer Sciences
Otto-Friedrich-University Bamberg

Abstract

Domain shift remains a major challenge in deploying machine learning (ML) models for medical imaging in real-world settings. Among its many other forms, scanner-induced domain shift—caused by differences in image acquisition across digitisation devices—can be particularly subtle yet impactful. This thesis investigates how such scanner-induced variability can be measured and interpreted using a same-anatomy, multi-scanner histopathology dataset comprising 44 samples scanned by five different devices.

Three complementary shift detection techniques are evaluated: BBSD, which analyses softmax output distributions of task-specific classifiers; MMD, which measures distances between feature vector distributions; and a multi-class variant of PAD*, which trains domain discriminators on both task-specific and task-agnostic features. These methods are applied across different representational levels using simple neural networks, a fine-tuned ResNet18, and off-the-shelf foundation models such as DINO and DINOv2.

The results suggest that task-specific methods like BBSD and PAD* are influenced by the behaviour of the underlying classifiers and are best interpreted as proxies for performance degradation on unseen domains. In contrast, task-agnostic methods such as MMD may detect shifts that are not always relevant to downstream performance. While more robust architectures like ResNet18 generalised well across scanner domains, foundation models showed limited ability to extract task-relevant features without fine-tuning—suggesting that their generalisability to histopathology remains an open question.

Overall, this work emphasizes that scanner-induced domain shift must be assessed with contextual understanding. The evaluated methods serve as lightweight, complementary tools for estimating and interpreting shift rather than providing absolute or objective measures.

Abstract

Domänenverschiebung ist nach wie vor eine große Herausforderung beim Einsatz von Modellen des maschinellen Lernens (ML) für die medizinische Bildgebung in realen Umgebungen. Unter den vielen Formen der Domänenverschiebung, kann Scanner-induzierte Domänenverschiebung—die durch Unterschiede in der Bilderfassung zwischen verschiedenen Digitalisierungsgeräten verursacht werden—besonders subtil, aber dennoch wirkungsvoll sein. In dieser Arbeit wird untersucht, wie eine solche Scanner-induzierte Variabilität gemessen und interpretiert werden kann, indem ein Histopathologie-Datensatz mit gleicher Anatomie und mehreren Scannern verwendet wird, der 44 Proben umfasst, die mit fünf verschiedenen Geräten gescannt wurden.

Es werden drei komplementäre Verfahren zur Erkennung von Verschiebungen bewertet: BBSD, das Softmax-Output-Verteilungen von aufgabenspezifischen Klassifikatoren analysiert; MMD, das Abstände zwischen Merkmalsvektorverteilungen misst; und eine Mehrklassenvariante von PAD*, die Domänendiskriminatoren sowohl auf aufgabenspezifische als auch aufgabenagnostische Merkmale trainiert. Diese Methoden werden auf verschiedenen Repräsentationsebenen unter Verwendung einfacher neuronaler Netze, eines fein abgestimmten ResNet18 und gebrauchsfertiger Grundmodelle wie DINO und DINOv2 angewandt.

Die Ergebnisse deuten darauf hin, dass aufgabenspezifische Methoden wie BBSD und PAD* durch das Verhalten der zugrundeliegenden Klassifikatoren beeinflusst werden und am besten als Indikatoren für eine Leistungsminderung auf unbekannten Domänen interpretiert werden können. Im Gegensatz dazu können aufgabenunabhängige Methoden wie MMD Verschiebungen erkennen, die für die nachgelagerte Leistung nicht immer relevant sind. Während robustere Architekturen wie ResNet18 gut über Scanner-Domänen hinweg generalisierten, zeigten Basismodelle nur begrenzte Fähigkeit, aufgabenrelevante Merkmale ohne Feintuning zu extrahieren - ein Hinweis darauf, dass ihre Generalisierbarkeit für die Histopathologie eine offene Frage bleibt.

Insgesamt unterstreicht diese Arbeit, dass scannerinduzierte Domänenverschiebungen mit kontextuellem Verständnis bewertet werden müssen. Die bewerteten Methoden dienen als leichtgewichtige, sich ergänzende Werkzeuge zur Abschätzung und Interpretation der Verschiebung, statt absolute oder objektive Maße zu liefern.

Acknowledgements

I want to thank my supervisor, Sebastian Doerrich, for the constant support throughout this challenging topic. His ideas, encouragement, and the resources he shared helped me navigate the process and made this thesis a meaningful learning journey.

The code with which the experiments were conducted, is provided on my personal GitHub and is accessible under the following link:

`https://github.com/KianHinke/scanner_domain_shift`

Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
ML	Machine Learning
FM	Foundation Model
NN	Neural Network
CNN	Convolutional Neural Network
MLP	Multilayer Perceptron
BBSD	Black Box Shift Detection
MMD	Maximum Mean Discrepancy
RKHS	Reproducing Kernel Hilbert Space
RBF	Radial Basis Function
PAD	Proxy A-Distance
CDF	Cumulative Distribution Function
PCA	Principal Component Analysis
CV	Coefficient of Variation
MAE	Mean Absolute Error
STD	Standard Deviation
FNR	False Negative Rate
FDR	False Discovery Rate
FPR	False Positive Rate
TPR	True Prositive Rate
WSI	Whole Slide Image
KS test	Kolmogorov-Smirnov test
DA	Domain Adaptation
DG	Domain generalisation

Notation

x	Input variable or feature vector
y	Target variable or label
$p(x), q(x)$	Probability distributions over input space
$p(y), q(y)$	Probability distributions over labels
$p(x, y), q(x, y)$	Joint probability distributions over inputs and labels
$p(x\|y), q(x\|y)$	Conditional distributions of inputs given label y
$p(y\|x), q(y\|x)$	Conditional distributions of labels given input x
$\mathbf{a}$	a vector
$\mathbf{a}_i$	Element i of vector $\mathbf{x}$
$\mathbf{A}$	Matrix notation for collection of vectors or data samples
$\mathbf{A^{-1}}$	Inverse of Matrix $\mathbf{A}$
$det(\mathbf{A})$	Determinant of $\mathbf{A}$
$\mathbf{A}_{i,:}$	Row i of matrix $\mathbf{X}$
$\mathbf{A}_{:,j}$	Column j of matrix $\mathbf{X}$
$f(x)$	Function, here often a classifier or feature extractor
G_f	Feature extractor part of a task model
G_y	Label predictor part of a task model
G_d^*	Domain discriminator (classifier trained on frozen features to detect domain)
H_0	Null hypothesis (e.g., no distribution shift)
H_1	Alternative hypothesis (distribution shift present)
$F_p(z), F_q(z)$	Empirical cumulative distribution functions (CDFs) of samples z
α	Significance level for statistical hypothesis testing
$\kappa(\cdot, \cdot)$	Kernel function measuring similarity between vectors
$\|\mathbf{z} - \tilde{\mathbf{z}}\|$	Euclidean distance between vectors $\mathbf{z}$ and $\tilde{\mathbf{z}}$
μ, σ	Mean and standard deviation of a data sample or distribution
$\mathbf{1}\{\cdot\}$	Indicator function, equals 1 if condition holds, else 0

1 Introduction

Over the last years, machine learning (ML) models have become increasingly effective in medical imaging tasks, but their performance can degrade unexpectedly when deployed in real-world settings that differ from the training environment (Guo et al., 2024; Pooch et al., 2020; Lafarge et al., 2017). A key reason for this drop is *domain shift*, a mismatch in the statistical properties of data between training and deployment environments. Such shifts can result from differences in imaging devices, acquisition protocols, or patient populations.

More specifically, *scanner-induced* domain shift occurs when the same tissue sample, scanned using different devices or settings, produces visually and statistically distinct image representations. This acquisition-induced variability can significantly impair the performance and reliability of ML models in clinical workflows. While there are many types of shift that affect the medical domain, scanner-induced domain shift may pose an underestimated issue (Aubreville et al., 2021). The challenge posed by scanner variability has been widely acknowledged in the literature (Khan et al., 2022; Madabhushi and Lee, 2016; Stacke et al., 2020; Aubreville et al., 2021) and performance degradation has been quantitatively assessed for different medical scanner modalities and reported to consistently deteriorate when affected by this type of shift (Guo et al., 2024; Wilm et al., 2023c). To mitigate domain shift, a lot of research has been done in the field of domain adaptation (DA) which aims to harmonize feature distributions between domains (Farahani et al., 2021). Alternatively, domain generalisation (DG) has introduced methods to train models that generalise well and remain robust to shifts (Zhou et al., 2022). However, relatively little attention has been paid to the underlying nature of the shifts and how they manifest and impact the data and downstream task models.

This work investigates scanner-induced domain shift using a same-anatomy, multi-scanner histopathology dataset. The focus is on evaluating how such shifts can be identified and interpreted in both the input- and latent space, and on how different methodological perspectives—such as task-specific and task-agnostic approaches—affect the detection process. These considerations are especially relevant for assessing whether visually subtle acquisition shifts translate into meaningful changes in model behaviour. By comparing existing methods across different representational levels and learning objectives, this study aims to provide a clearer understanding of how scanner-induced domain shift can be measured and interpreted in the context of medical imaging.

2 Related Work

This section provides an overview of selected research on domain shift detection that is particularly relevant to this paper. Rather than being a comprehensive survey, it highlights key methods that give insight to the methodological choices made here and illustrates how they complement each other or motivate further investigation.

Furthermore, the related work section sets the stage for the subsequent contribution section.

2.1 Black Box Shift Detection (BBSD)

BBSD was first introduced by Lipton et al. (2018) as part of a broader framework called Black Box Shift Estimation (BBSE), which is designed to detect and quantify label shift. The method involves using an existing, pre-trained label classifier and measuring the distance between the distributions of its outputs — either hard classification outputs (BBSE-hard) or softmax probabilities (BBSE-soft) — on source and target data. The authors showed that this approach can be effective even if the classifier is inaccurate, biased, or uncalibrated, as long as its confusion matrix is invertible. This makes BBSD a flexible and practical tool since it does not require retraining the classifier on source data and can be applied without needing labels in the target domain. Additionally, the authors found that BBSD offers more statistical power than kernel-based two-sample tests that operate directly on the input space. They also emphasised the advantage of using the classifier to reduce dimensionality before testing, which is especially helpful because two-sample tests struggle in high-dimensional settings (Ramdas et al., 2015). While the original focus of BBSD was on detecting label shift, the authors explicitly noted its broader applicability "to detect covariate shift, concept shift, and more general forms of nonstationarity" Lipton et al. (2018). In this Bachelor thesis, BBSD is applied in the context of a type of covariate shift, using the distributions of model outputs to detect changes in the input data distribution across different scanner domains.

Building on these theoretical foundations, Rabanser et al. (2019) conducted an extensive empirical evaluation of various shift detection methods, including BBSD, on standard computer vision datasets such as MNIST (LeCun et al., 2010) and CIFAR-10 (Krizhevsky, 2009). They found that BBSD "works surprisingly well under a broad set of shifts, even when the label shift assumption is not met" (Rabanser et al., 2019). Their experiments tested both hard and soft classifier outputs across multiple types of synthetic distribution shift, including covariate shift. Among the methods evaluated, BBSD using softmax outputs consistently performed the best for detecting shifts. Furthermore, they emphasised the practical value of this finding, noting that it enables a classifier that was trained for a downstream classification task to be repurposed as a shift detector after training, without significant additional effort. This makes BBSD a convenient and lightweight tool that looks promising with regard to application in real-world scenarios (Rabanser et al., 2019).

Recently, Roschewitz et al. (2024) introduced a framework for real-world shift identification, based on the comprehensive empirical validation by Rabanser et al. (2019). As part of this framework, they proposed a dual shift detector that combines BBSD and Maximum Mean Discrepancy (MMD) to leverage the strengths of output-based and feature-based detection methods, with the goal of achieving more robust and reliable results. This combined approach was applied to several prominent medical datasets across different imaging modalities, making it particularly relevant

for identifying distribution shifts in medical image analysis. In their evaluation, BBSD alone was found to be less reliable for detecting covariate shifts, including acquisition-related shifts. Instead, feature-based methods—such as MMD applied to encoder-derived representations—proved to be more effective and consistent. However, the effectiveness of these feature-based approaches was shown to depend heavily on the choice of encoder used to extract the lower-dimensional representations, highlighting the importance of selecting appropriate feature extractors in practice (Roschewitz et al., 2024).

While these three papers form a logical and developmental timeline of the BBSD method, there is another related method that should be mentioned in this context. Released just one year after the BBSD paper by Lipton et al. (2018), in 2019, Alberge et al. (2019) proposed a similar black-box shift detection approach focused on detecting covariate shift using a black-box setup. Both methods share the core principle of using the outputs of a pre-trained classifier to reduce the dimensionality of the data and avoid dependence on raw feature spaces, which are often high-dimensional and harder to compare statistically. However, instead of directly comparing the softmax outputs of the predictor, Alberge et al. compute the negative log-likelihood values (also called coding length) of the classifier's predictions and discretize them into bins. Finally, they employ a Pearson's Chi-squared test to compare the binned likelihood distributions, testing whether the model exhibits significantly different confidence behaviour under the new input distribution. This stands in contrast to BBSD, which compares the raw output distributions (softmax probabilities or hard predictions) using two-sample tests such as the Kolmogorov–Smirnov test or Maximum Mean Discrepancy (Alberge et al., 2019).

Although the approach of Alberge et al. (2019) offers a targeted formulation for detecting covariate shift, it was not used in this thesis. Instead, the BBSD method, as proposed by Lipton et al. (2018), was chosen due to its broader recognition and continued validation within the literature. Lipton et al.'s work was presented at the renowned International Conference on Machine Learning (ICML) and has since been cited extensively. It forms the basis for the subsequent studies by Rabanser et al. (2019) and Roschewitz et al. (2024), who have further refined the BBSD framework and confirmed its strong performance across various types of distribution shift, including covariate shift. This lineage of validation and adoption, along with the lightweight nature of the method and its continuous development, makes BBSD a suitable choice for the purposes of this thesis.

Further empirical evidence for the applicability of BBSD in medical imaging settings is provided in recent work by Kore et al. (2024), who studied data drift in chest radiograph classification. Unlike the earlier works by Lipton et al. (2018) and Rabanser et al. (2019), which primarily focused on synthetic or benchmark datasets, Kore et al. examined real-world temporal distribution shifts—specifically, the introduction of COVID-19-era chest X-rays—and tested the ability of BBSD to detect them. Using a fine-tuned TorchXRayVision model to predict 14 pathologies, they evaluated four drift detection strategies: tracking model performance, image-based detection (via a TorchXRay autoencoder), BBSD, and a hybrid image-and-output method.

In addition to the real-world drift, they evaluated the robustness and sensitivity of the employed strategies to synthetical drifts in different categories. Their findings confirmed that BBSD performs competitively, nearly matching the sensitivity of the combined method, especially under realistic distribution shifts. Notably, they treated drift detection as a statistical hypothesis test and reported only p-values, emphasizing detection over quantification. They also highlighted key limitations, such as the dependency of BBSD's sensitivity on sample size and patient-specific features. Crucially, they also discuss the poor performance of traditional performance metrics like AUC-ROC for detecting drift—although frequently utilised for this purpose in practice—and highlight the need for dedicated shift detection strategies (Kore et al., 2024).

These insights align with and extend earlier claims about BBSD's flexibility and practical utility, particularly in the medical imaging domain, while also underscoring some of the challenges when applying it to real-world clinical data.

2.2 PAD and PAD*

Originally introduced by Ben-David et al. (2006), the *Proxy A-Distance (PAD)* measures the divergence between domains by training a binary classifier to distinguish between samples from each domain. The misclassification rate of this domain discriminator is then used as a proxy for the distance between the source and target distributions in a given representation space. However, PAD is not sufficient on its own but should be interpreted alongside the performance of a model trained on the downstream classification task, particularly by considering the source error. The authors argue that when both the source error and PAD are low, the model is likely to generalise well to the target domain—provided that the labelling function is shared across domains (Ben-David et al., 2006).

Crucially, PAD only reflects how distinguishable the domains are within the representation space—it does not account for whether the source domain provides sufficient support in the regions of the input space that the target domain occupies. As a result, PAD can be misleadingly low if the domain discriminator relies on features that are irrelevant to the classification task. In such cases, even if source error and PAD are low, generalisation to the target may fail due to covariate shift—especially when the task-relevant features differ significantly across domains (Elsahar and Gallé, 2019).

To overcome this issue, Elsahar and Gallé (2019) proposed a refinement of this approach by introducing PAD*, a variation that computes domain divergence using task-specific representations. PAD* leverages the hidden representations of a model trained on the original classification task, making it more sensitive to differences that matter for the task and less influenced by irrelevant domain-specific features. This improves robustness and makes the metric more reflective of actual transfer performance, though it also introduces model dependence.

Building on this refined method, Aubreville et al. (2021) applied PAD* to mitotic histopathology figures from the MICCAI-MIDOG challenge 2021 data set which contains human breast cancer cases digitised by four scanners. Curiously, they found the PAD* results to be unrelated to the downstream performance on the mitosis object detection task. Notably, although the dataset contains an equal number of samples from four scanners, these samples do not necessarily originate from the same anatomical source. In other words, the same physical tissue sample was not digitised across all scanners.

This inconclusive relationship between PAD* scores and task performance highlights the need for further evaluation of the method and comparison to other approaches, particularly in settings with more controlled anatomical consistency across scanners.

2.3 Representation Shift

Stacke et al. (2020) proposed a representation-learning-based approach to quantify domain shift by analysing the internal representations learned by convolutional neural networks (CNNs), introducing a model-centric metric called *representation shift*. This method examines the CNN's internal feature space by statistically comparing the distributions of mean activation values from individual convolutional filters across source and target domains. Specifically, for each filter in a given convolutional layer, the mean activation is computed across all images in a dataset, and the resulting distributions are compared using discrepancy metrics such as the Wasserstein distance, Kullback-Leibler (KL) divergence, and the Kolmogorov-Smirnov statistic. This layer-wise analysis provides valuable insights into how domain shifts affect different stages of the feature hierarchy, revealing that discrepancies in earlier convolutional layers often correlate more strongly with performance degradation. A key advantage of this method is that it does not require labelled target data, although—in contrast to BBSD—it does require access to the model's internal activations. The authors evaluated their method across multiple CNN architectures and observed that the extent and nature of representation shift varied by model, emphasizing the model-specific nature of domain shift. The representation shift metric consistently showed a strong correlation with accuracy drops on unseen domains, particularly when measured in early layers, highlighting the importance of low-level feature consistency for generalisation. Furthermore, they applied their framework to histopathology data, the medical modality used in this thesis. By focusing on the medical imaging context, where prior work has shown that models behave differently than those trained on natural images (Raghu et al., 2019), they demonstrated the practical relevance of their approach. (Stacke et al., 2020)

2.4 Other work

Guo et al. (2024) conducted a broad and systematic investigation into scanner-induced domain shift using medical imaging data, making it the first large-scale

study to span multiple modalities and anatomical regions. Through extensive experiments, they show that model performance consistently drops when evaluated on data from scanners not seen during training, and that the severity of this shift varies across imaging modalities. Overall, the study highlights the need to account for scanner heterogeneity in medical imaging applications and emphasizes the practical challenges of deploying deep learning models in real-world clinical settings. Notably, their work focuses primarily on quantifying the impact of scanner variability through the performance of deep learning models, rather than attempting to analyse domain shift directly at the level of input distributions or latent representations, as explored in the following sections of this thesis.

Wilm et al. introduced the *Multi-Scanner Canine Cutaneous Squamous Cell Carcinoma Histopathology Dataset* used in this work (Wilm et al., 2023c). Using this dataset, they proposed a supervised pre-training approach to learn domain-invariant representations using "Barlow Triplets", an extension of another self-supervised approach called "Barlow Twins"(Zbontar et al., 2021). The Barlow Triplet method processes input tuples of corresponding image patches from multiple training scanners, thereby extending the original Barlow-Twin loss to ensure consistency across these diverse scanner views. While their method effectively harmonised representations across different scanners, it yielded only limited improvements on the downstream segmentation task. The authors suggest that scanner-specific characteristics may influence model performance in unexpected ways and emphasize that methods developed for natural images may not directly translate to the medical imaging domain (Wilm et al., 2023b).

3 Contribution

While numerous methods have been proposed for detecting domain shifts in medical imaging, they are often assessed only in terms of their ability to predict downstream performance degradation. However, scanner-induced domain shift remains comparatively understudied, particularly in terms of how it manifests and can be measured independently of task performance. Despite advances in domain adaptation and generalisation, relatively little work has focused on systematically analysing the shift itself across different stages of the machine learning pipeline.

This bachelor's thesis addresses this gap by systematically evaluating existing shift detection methods on a same-anatomy, multi-scanner histopathology dataset. The core objective is to investigate how scanner-induced variability—often visually subtle—can be detected and interpreted using a range of methodological perspectives.

To this end, several domain shift detection techniques are applied and compared, including BBSD (which leverages task-specific classifier outputs), MMD (which measures distances between feature distributions), and a multi-class extension of PAD* (which learns to discriminate domains based on input or latent features). These methods are evaluated in both the input space and latent space, using representa-

tions from a simple neural network, a fine-tuned ResNet18, and off-the-shelf foundation models such as DINO and DINOv2.

A key contribution of this work lies in comparing task-specific and task-agnostic approaches to shift detection, and in analysing how their behaviour is influenced by the underlying feature representations and model architectures. By applying these methods across models with different levels of complexity and training, the study aims to support a better understanding of how measurable shift relates to model behaviour, and how reliably different techniques can be used to assess acquisition-induced domain shift in practice.

4 Theoretical Background

This theoretical background section aims to introduce the key concepts, methods, and evaluation metrics that are essential for understanding the design, interpretation, and significance of the experiments presented in the subsequent sections.

4.1 Whole Slide Imaging in Histopathology

Microscopic Whole Slide Images (WSIs) are high-resolution digital scans of histopathology tissue sections, typically stained with hematoxylin and eosin (H&E) to highlight cellular structures and tissue morphology. Staining is a chemical process that adds contrast to biological tissues, making relevant features visible under a microscope (Feldman and Wolfe, 2014). The resolution of a WSI is often given in micrometers per pixel (e.g., 0.23µm/pixel), indicating the real-world size each pixel represents. Thus, lower values mean higher detail. Since WSIs are often extremely large, sometimes exceeding $100,000 \times 100,000$ pixels, they are typically divided into smaller, fixed-size patches for efficient machine learning processing (Afshari et al., 2023; Koohbanani et al., 2021). In this work, staining variation is not a shift factor, as each physical tissue sample was stained once and then digitised using multiple scanners—ensuring that any observed variation arises solely from differences in scanner hardware or configuration.

4.2 Relevant Shift Terms

Let $p(x, y)$ and $q(x, y)$ denote the joint distributions of input–label pairs in the source (training) and unseen target (testing) domains, respectively. From these, one can derive the marginal distributions $p(x)$, $q(x)$ (input distributions) and $p(y)$, $q(y)$ (label distributions), as well as the conditional distributions $p(x|y)$, $q(x|y)$ (how inputs manifest given labels) and $p(y|x)$, $q(y|x)$ (how labels are predicted from inputs). It is essential to distinguish between the conditional distributions $p(x \mid y)$ and $p(y \mid x)$. The distribution $p(x|y)$ describes how features typically appear for a given class—i.e. the "manifestation" of a target. In contrast, $p(y \mid x)$ represents

the model's predictive objective: the probability of a label given a particular input. Under label shift, $p(x \mid y)$ remains stable, meaning that features for each class look the same across domains, while the model may still face degraded performance because the prior probabilities of classes $p(y)$ have changed.

It should be noted that there are a lot of different shift terms circulating in the field and sometimes even the definitions of the same terms differ. For example, Farahani et al. (2021) introduce a definition for prior shift—also known as label shift/prior probability shift/target shift—that does not align with most other sources. Here, we adopt the most conceptually coherent definition, which is supported by most sources (Huyen, 2022; Moreno-Torres et al., 2012; Storkey et al., 2009; Lipton et al., 2018; Roschewitz et al., 2024).

4.2.1 Covariate Shift

In machine learning, *covariate shift* refers to a specific type of distributional shift where the input distribution changes between training and testing, but the conditional distribution of the output given the input remains the same. Formally, following the notation introduced above, covariate shift occurs when $p(x) \neq q(x)$ while the conditional distribution $p(y|x) = q(y|x)$ remains unchanged (Huyen, 2022; Farahani et al., 2021; Moreno-Torres et al., 2012). In the context of *scanner-induced shift*, this means that when a scanner changes the contrast of an image, it affects how the pixel values in the image are distributed. However, the way image inputs relate to their labels—like which patterns indicate a tumor—stays the same. Furthermore, note that when $p(x) \neq q(x)$ under the covariate shift assumption, the class-conditional distribution $p(x|y)$—i.e. how features manifest for a given class—may change.

4.2.2 Acquisition Shift and Scanner-Induced Shift

Acquisition shift is a subtype of *covariate shift* that encompasses all shift phenomena originating from data collection processes that lead to changes in the input distribution. This includes differences in sensor type, resolution, lighting conditions, imaging protocols, patient positioning, and—particularly in histopathology—staining protocols. Furthermore, *scanner-induced shift* is a more specific form of acquisition shift that occurs in medical imaging when the same tissue or structure is digitised using different scanners. These differences can introduce systematic variability in the resulting images—such as colour balance, contrast, or noise characteristics—without altering the underlying pathological features, thereby possibly affecting the model's generalisation across domains despite the task remaining unchanged. It should be noted that covariate shift can arise from different causes, such as sampling bias, which pertains to the selection of data rather than how it was collected, but also affects the marginal distribution of the inputs while the conditional distribution remains unchanged (Farahani et al., 2021; Moreno-Torres et al., 2012; Stacke et al., 2020; Roschewitz et al., 2024).

4.2.3 Label Shift

Label shift, also referred to as *prior shift* (Farahani et al., 2021) refers to a type of distributional shift where the label distribution differs between training and testing, while the conditional distribution of features given a label remains unchanged. Formally, label shift occurs when $p(y) \neq q(y)$ but $p(x \mid y) = q(x \mid y)$. This means that the way a class or target manifests in the input data stays the same, even though the prevalence of classes changes (Lipton et al., 2018).

One common cause of label shift is sampling bias. This occurs when the data used to train a model is collected in a way that overrepresents or underrepresents certain classes. For instance, a dataset may be artificially balanced to contain equal numbers of tumor and non-tumor cases, even if tumors are rare in reality. As a result, the model encounters a different class distribution during deployment than it saw during training, leading to a label shift.

4.3 Domain Adaptation and Domain Generalisation

Domain adaptation (DA) and *domain generalisation* (DG) are two strategies aimed at addressing performance drops caused by distribution shifts between training and testing data. In recent years, a lot of research has been dedicated to this field, because in real-world scenarios, models frequently encounter inputs that differ from data seen during training.

Firstly, **domain adaptation (DA)** aims to align feature distributions between a labelled source domain and an unlabelled or sparsely labelled target domain—for example, histopathology images acquired from different scanners. In medical imaging, where annotated data is often limited, DA techniques help models retain their performance across such domain gaps.

Shallow methods include instance-based approaches like Kernel Mean Matching (KMM), which reweight source samples under covariate shift, and feature-based approaches like Transfer Component Adaptation (TCA), which learn domain-invariant representations by minimizing distributional discrepancies in RKHS. Deep DA methods build on these ideas using neural networks, such as Deep Adaptation Networks (DAN), which align feature distributions via Multiple Kernel MMD, or adversarial strategies like Domain-Adversarial Neural Networks (DANN), which promote domain invariance through a gradient reversal layer that confuses a domain classifier during training (Farahani et al., 2021).

Domain generalisation (DG), on the other hand, aims to train models that perform robustly on unseen domains, without requiring any access to target domain data during training. Unlike domain adaptation, which allows for some form of target supervision or alignment, DG assumes only access to one or more related source domains and seeks to learn representations that generalise beyond them. This is particularly relevant in medical imaging, where deployment environments (e.g., unseen scanners or hospitals) may differ from those in the training set.

To address this, DG methods employ strategies such as domain alignment (e.g., matching distributions across source domains using MMD or adversarial training), meta-learning (simulating domain shift during training), data augmentation (generating diverse training examples to mimic unseen domains), and self-supervised learning (using pretext tasks to learn domain-agnostic features). These approaches aim to reduce reliance on scanner-specific features and encourage high-level, invariant representations that support robust generalisation without retraining (Zhou et al., 2022).

In the context of scanner—induced domain shift—where variations in medical image appearance arise from a wide range of causes—both approaches are highly relevant. However, they are mitigation strategies designed to improve model robustness after a domain shift has occurred and are therefore related but not directly subject of this work, which focuses on detecting, quantifying, and characterizing the scanner-induced shift itself.

4.4 Relevant Deep Learning Concepts and Architectures

Generally, deep learning models learn by optimizing a set of parameters to minimize a predefined loss function, which quantifies how far the model's predictions are from the true targets. During training, an input is passed through multiple layers of a network architecture in a process known as forward propagation, resulting in an output such as a class label or regression value. The loss is then computed, and the model uses backpropagation to calculate the gradients of the loss with respect to each parameter. These gradients indicate how the parameters should be adjusted to reduce the loss. An optimisation algorithm—e.g. stochastic gradient descent—updates the parameters incrementally. Through repeated iterations over the training data, the model improves its internal representations. The specific structure of a model's architecture significantly influences the types of representations it can learn and the range of tasks it can generalise to (Goodfellow et al., 2016; Schmidhuber, 2015).

4.4.1 Hidden Representations and Feature Extraction

While processing input data through multiple layers, deep learning models generate intermediate outputs at each stage of the network. These intermediate outputs are also referred to as *hidden representations* because they are not directly visible in the final prediction, but play an important role in how the model processes the data internally (Goodfellow et al., 2016). Typically, each layer transforms its input into increasingly abstract representations, allowing the model to capture low-level patterns—e.g., edges in images—in earlier layers and more complex, high-level features—e.g., parts of objects or even semantic categories—in deeper layers (Yosinski et al., 2014). The quality and structure of these hidden representations are essential to the performance and transferability of a model.

Building on this concept, *feature extraction* refers to the process of using the internal, hidden representations learned by a model—usually from its intermediate layers—as

compact summaries of the input data that capture important patterns or structures (Pan and Yang, 2010). Which information and patterns the model deems relevant is, of course, highly dependent on the learning goal of the model—i.e., how the model calculates its loss. In practice, feature extraction allows a model trained on one dataset or task to be reused for another by keeping its learned weights fixed and using its internal outputs as input for a new classifier or task-specific module (Yosinski et al., 2014). This is especially helpful when labelled data is limited, as it enables transfer learning—relying on general-purpose features learned during large-scale pre-training. In this work, these reduced representations are used to reduce the complexity of high-dimensional image data to enable the application of distance metrics for shift detection. Models like ResNets and Vision Transformers are commonly used for feature extraction, with different layers offering varying levels of abstraction depending on what the task requires (Dosovitskiy et al., 2020; Caron et al., 2021).

4.4.2 Multi-Layer Perceptrons (MLPs)

Multi-Layer Perceptrons (MLPs) are one of the most fundamental types of deep neural networks. They are made up entirely of fully connected layers, meaning that each neuron in one layer is connected to every neuron in the next. MLPs operate on fixed-size inputs that are represented as one-dimensional vectors, which means the input data must be flattened before being processed (Goodfellow et al., 2016). These networks learn by applying a series of linear transformations followed by non-linear activation functions—e.g. the ReLu function Nair and Hinton (2010)—allowing them to build increasingly abstract representations across multiple layers. In theory, this layered structure enables MLPs to approximate a wide variety of complex functions. However, they do not take advantage of any specific structure in the input data—such as the spatial layout in images or the sequential nature of time series. Because of this, they often need more training data and parameters to perform as well as architectures that include built-in assumptions about the data, such as convolutional or attention-based models (LeCun et al., 2015). Still, MLPs remain a key building block in deep learning and are especially useful in situations where inputs are already vectorised, or when model simplicity and interpretability are priorities.

4.4.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed to process grid-like data, such as images (LeCun et al., 2015). Unlike traditional multilayer perceptrons (MLPs), which rely on fully connected layers, CNNs use convolutional layers that apply learnable filters to local regions of the input. These filters function as feature detectors, capturing low-level patterns like edges and textures in the initial layers, and progressing to more abstract representations such as shapes and objects in deeper layers (Goodfellow et al., 2016; Schmidhuber, 2015). This hierarchical representation is key to CNNs' success in computer

vision tasks, as it allows them to effectively model the spatial relationships within data. Pooling layers are commonly employed to reduce the spatial dimensions of the feature maps, improving computational efficiency and helping the model recognize patterns regardless of their position in the image while keeping important details (LeCun et al., 2015; Goodfellow et al., 2016). The combination of convolutional and pooling layers enables CNNs to learn robust, multi-scale features from raw pixel data. As a result, CNNs have become foundational models in many vision-related tasks, including image classification, object detection, and semantic segmentation (He et al., 2016; Dosovitskiy et al., 2020).

4.4.4 Residual Networks (ResNets)

Originally introduced by He et al. (2016), *ResNets* extend the MLP idea by incorporating convolutional layers and introducing skip connections. This architecture enables the training of very deep networks by mitigating the vanishing gradient problem, which occurs when gradients—used to update the model's weights during training—become increasingly small as they are backpropagated through many layers, effectively preventing the earlier layers from learning. By using skip connections that bypass one or more layers, ResNets preserve the flow of gradients and help ensure that all layers receive meaningful updates. Like MLPs, ResNets learn hierarchical representations, but are better equipped for high-dimensional structured data such as images due to their spatial awareness in the convolutional operations.

4.4.5 Foundation Models and Self-Supervised Vision Transformers

In recent years, *foundation models* (FM) have become a popular and powerful approach across many areas of machine learning. These models are typically large neural networks trained on broad and diverse datasets using self-supervised or weakly supervised learning objectives (Bommasani et al., 2021). The idea is to learn general-purpose representations that can be easily adapted to different downstream tasks with minimal additional training. While this approach was initially applied to natural language processing, it has since made significant contributions to computer vision as well.

Among the most notable vision FMs are *DINO* and *DINOv2*. These models move away from traditional supervised learning toward self-supervised methods, where models learn directly from raw image data without the need for manual labels. Both are built on the *Vision Transformer* (ViT) architecture (Dosovitskiy et al., 2020), which divides an image into small patches and treats them like words in a sentence. It then uses a mechanism called self-attention to learn how different parts of the image relate to each other, allowing the model to understand both local details and global context. A special `[CLS]` token is added to the sequence and used to summarize the entire image, providing a compact representation that can be used for classification.

DINO and DINOv2 are trained with contrastive self-distillation, which encourages the model to produce consistent representations across different views of the same image (Caron et al., 2021; Oquab et al., 2023). While DINO is typically trained on ImageNet-1k (without labels), DINOv2 uses a much larger and more diverse dataset called LVD-142M—a curated collection of 142 million images from various sources. This broader dataset helps DINOv2 learn more robust and generalisable visual features (Oquab et al., 2023). These models are now widely used as pre-trained backbones in vision tasks and often outperform supervised models, especially in terms of generalisation and scalability (Parvaiz et al., 2023).

4.4.6 ImageNet-1k and LVD-142M

The ImageNet-1k dataset is a widely used benchmark dataset in computer vision, containing approximately 1.28 million labelled images across 1,000 object categories (classes). It is frequently used for evaluating and training supervised models due to its high-quality annotations and broad coverage of natural image classes (Deng et al., 2009).

In contrast, LVD-142M is a much larger and more diverse collection of about 142 million curated images, assembled by selecting samples from an uncurated pool of web images that were visually similar to those in datasets like ImageNet-1k, ImageNet-22k, Google Landmarks, and other fine-grained sources. Unlike ImageNet-1k, LVD-142M does not have specified classes and is designed for self-supervised learning. It was developed to support the pre-training of large-scale vision transformers such as DINOv2, with a focus on representational richness rather than explicit classification (Oquab et al., 2023).

4.4.7 Hyperparameters

In machine learning, hyperparameters influence how a model learns and can have a big impact on its final performance. Unlike parameters that are learned during training, hyperparameters are set beforehand and control the optimisation process.

Two key hyperparameters are the number of *epochs* and the *learning rate*. The number of *epochs* controls how many times the model iterates over the training data. With too few epochs, the model might not learn enough to effectively handle the task, while too many epochs may cause the model to overfit (Goodfellow et al., 2016). Overfitting means that the model is too overtuned to the data—including its noise—and will thus fail to generalise to unseen data. The *learning rate*, on the other hand, determines how big the parameter updates are during training. If it's too high, the model may never converge, while setting it too low can make training unnecessarily slow or make it get stuck in poor local minima because the weight updates in any direction are too small to cause improvements to the loss (Bengio, 2012).

Finding the right balance for these settings is a non-trivial task and often crucial to getting high-performing classifiers.

4.5 Black Box Shift Detection (BBSD)

BBSD is a method to identify and measure domain shift between datasets. This is accomplished by leveraging the outputs—either hard predictions or softmax probabilities—of an existing black-box classifier, which is trained on the source data, to detect shifts in the distribution of the target domain data. While originally designed for label shift, it has been shown to also reflect different types of shift, such as covariate shift, of which scanner-induced shift is a subtype. The underlying idea of this method is that shifts in the label distribution or even feature distribution will influence the behaviour of the black box classifier and thus be measurable in its outputs.

4.5.1 Key Theoretical Components and Assumptions

Label Shift Assumption: Formally, as introduced by Lipton et al. (2018), BBSD relies on the assumption of label shift, where the class-conditional distribution $p(x|y)$ remains invariant across domains, and only the marginal distribution of labels $p(y)$ changes. This assumption allows BBSD to focus on changes in the label distribution between the source and target domains without needing to account for changes in the feature distribution. Under this assumption, label shift can be measured without the need for labelled target data. However, Lipton et al. (2018) and Rabanser et al. (2019) have also applied BBSD on other types of shifts, including covariate shift, where the conditional distribution $p(y|x)$ remains the same across domains but the input distribution $p(x)$ changes. Furthermore, in real-world scenarios, multiple types of distribution shifts frequently affect the data simultaneously, and even slight changes in the input space can affect the classifier's outputs on the target data — particularly when using an imperfect classifier, which BBSD is specifically designed to accommodate.

Black Box Predictor: The method assumes access to a pre-trained label classifier $f : X \to Y$, which can be any machine learning model, such as deep neural networks, support vector machines, or decision trees, trained on the source data. The classifier acts as a "black box," meaning that the internal workings of the classifier do not need to be understood or accessible for BBSD to operate. The method focuses only on the classifier's output, i.e., the predicted labels or probabilities for the source and target datasets.

Invertible Confusion Matrix: A key requirement for BBSD's effectiveness is that the expected confusion matrix of the classifier must be invertible on the source data distribution. Invertibility $\mathbf{A^{-1}}$ of the confusion matrix implies that the classifier's outputs are linearly independent, which is the case if the determinant $det(\mathbf{A})$ of the matrix is $\neq 0$. Intuitively, the confusion matrix is typically invertible where the classifier predicts the true class more often when the input actually belongs to this class than when the input belongs to any other class (i.e. more true positives

than false positives) (Lipton et al., 2018). Conversely, if the confusion matrix is not invertible, it means that the predictor's behaviour for some classes is not sufficiently distinct from its behaviour for other classes. This would make it impossible to adequately determine the underlying shift in the label distribution from the observed changes in the predicted label distribution.

Shift Detection as Hypothesis Testing: BBSD detects distribution shifts by formulating the detection process as a hypothesis test on the classifier's output distribution. The null hypothesis H_0 assumes that there is no shift affecting the classifier's outputs, i.e., the output distributions are identical across domains,

$$p(f(x)) = q(f(x)).$$

The alternative hypothesis H_1 assumes that the distributions differ,

$$p(f(x)) \neq q(f(x)).$$

If a significant difference is detected between the classifier's output distributions on the source and target data, BBSD rejects the null hypothesis, indicating that a shift has occurred. The significance is determined by the p-value during the test and defined by the significance threshold α which is typically set to $\alpha = 0.05$, meaning that there is a 5% chance of incorrectly rejecting the null hypothesis when it is actually true—i.e., a 5% risk of a false positive (also called Type I error).

4.5.2 Variations: Hard and Soft Predictions

The detection of label shifts using BBSD involves comparing the classifier's outputs between the source and target domains using two-sample statistical tests. The method can be applied to both hard and soft predictions generated by the classifier.

1. Soft Predictions (BBSDs):

   When the classifier produces soft predictions (e.g., probabilities via softmax outputs), BBSD applies two-sample tests, such as the Kolmogorov-Smirnov (KS) test, to compare the predicted probabilities between the source and target domains per class. Since multiple classes are typically involved, a Bonferroni correction is used to control the error rate across multiple tests. If the corrected p-value is below a specified significance threshold, the null hypothesis of "no shift" is rejected. This approach has been shown to generally produce more reliable results and is used in the methodology of this work.

2. Hard Predictions (BBSDh):

   When the classifier produces hard class labels, BBSD uses statistical tests like Pearson's chi-squared test to compare the frequency distributions of predicted labels between the source and target datasets. However, Rabanser et al. (2019) found that BBSD on soft predictions (i.e. softmax outputs) was the best-performing overall method of dimensionality reduction for univariate testing.

BBSD on Covariate Shift

Although BBSD primarily focuses on label shift, Rabanser et al. (2019) found BBSD to be effective even for other types of shifts, where the label shift assumption is not met. It can be argued that in the case that the true label distributions of the target data were known to be equal, i.e. $p(y) = q(y)$, then shifts in the observed label distributions of the classifier's output, i.e. $p(f(x)) \neq q(f(x))$, should reflect shifts in the input distributions $p(x)$ and $q(x)$, as long as the conditional distributions $p(y|x) = q(y|x)$ remain invariant across domains. In such scenarios, the changes in the feature space may be reflected in the output distribution of the classifier.

However, Roschewitz et al. (2024) reported BBSD and other output-based methods in isolation to have limited effectiveness at detecting covariate shifts, including acquisition shifts. In these cases, BBSD alone may fail to detect shifts, and feature-based methods might be required for more comprehensive shift detection.

4.5.3 Kolmogorov-Smirnov (KS) Test

As early as 1933, Andrey Kolmogorov's foundational work derived the theoretical distribution of the maximum difference between empirical and theoretical cumulative distribution functions (CDFs). Nikolai Smirnov extended this work in 1939, developing the two-sample version of the test to compare whether two empirical distributions differ significantly. Today, computational implementations like the scipy.stats.ks_2samp test—based on the later work of Hodges Jr (1958)—make the test efficient and accessible.

Conceptually, the *Kolmogorov-Smirnov* (KS) two-sample test is a non-parametric test, meaning it assesses whether two independent samples originate from the same underlying distribution without making any assumptions about the shape (e.g., normal, exponential) of that distribution. Rabanser et al. (2019) define the test statistic as the maximum absolute difference Z between the *empirical cumulative distribution functions* (CDFs or ECDFs) of the two samples, given by:

$$Z = \sup_z |F_p(z) - F_q(z)|$$

where $F_p(z)$ denotes the CDF of the source and $F_q(z)$ denotes the CDF of the source and target distributions. It returns a value between 0 and 1: 0 if the distributions are identical and 1 if the distributions are completely non-overlapping.

Generally, the KS test doesn't require equal dataset sizes, but when one dataset is much smaller, the associated CDF will be less fine-grained and the measured maximum absolute distance between the CDFs will be less precise, as exemplarily visualised in Appendix Figure 33.

Application to BBSD In the context of BBSD, the KS test is applied as multiple univariate tests, meaning it is run independently for each class-specific softmax output. For example, to test for distributional shift in class 0, we compare the distribution of softmax probabilities assigned to class 0 for source domain data

(e.g., scanner A) against the same for target domain data (e.g., scanner B). The test statistic reflects the largest difference between these two CDFs. Under the null hypothesis that both samples come from the same distribution, the KS test provides a p-value indicating the probability of observing such a large difference just by chance. If this p-value falls below a pre-defined threshold, the null hypothesis is rejected, suggesting a statistically significant distribution shift.

Bonferroni Correction:

However, performing multiple tests—one per class—raises the risk of false positives due to the increased number of comparisons. To address this, a *Bonferroni correction* is applied to control the overall type I error rate (Bland and Altman, 1995). The corrected significance level is given by $\alpha_{\text{corrected}} = \alpha/m$, where α is the original significance level (e.g., 0.05) and m is the number of tests. For instance, if two independent KS tests are conducted (because there are two classes), then $\alpha_{\text{corrected}} = 0.05/2 = 0.025$. This means that each individual test must produce a p-value below 0.025 in order to be considered statistically significant at the overall 5% level. Alternatively, the original significance threshold can be retained and each p-value can be multiplied by the number of tests to obtain the corrected p-value (Rabanser et al., 2019; Roschewitz et al., 2024).

4.6 Maximum Mean Discrepancy (MMD) Permutation Test

The *Maximum Mean Discrepancy* (MMD) is another non-parametric test used to compare whether two samples—typically from a source and a target domain—are drawn from the same underlying distribution. It is a common distance metric and has been used extensively in DA and DG techniques (Yan et al., 2017; Long et al., 2013)—in DA, to align source and target domain distributions (e.g., in Kernel Mean Matching and Deep Adaptation Networks), and in DG, to align feature distributions across multiple source domains—with the common goal of learning domain-invariant representations by minimizing distributional discrepancy in a reproducing kernel Hilbert space (RKHS)—a high-dimensional space induced by a kernel function (Farahani et al., 2021; Zhou et al., 2022). Conceptually, MMD measures the distance between two probability distributions by comparing their mean embeddings in the RKHS. A mean embedding is a representation of an entire distribution as a single point in the RKHS, computed by averaging the mapped feature vectors of samples from that distribution. MMD quantifies the distance between these mean embeddings, and thus provides a non-parametric estimate of how different the two underlying distributions are (Rabanser et al., 2019; Zhou et al., 2022). In contrast to the Kolmogorov-Smirnov test, which is applied univariately to one dimension at a time, MMD is a multivariate test that compares the full joint distributions of multidimensional data, making it suitable for application on high-dimensional data and potentially allowing for the measurement of shifts that only manifest across combinations of features, although computational costs increase quickly with higher dimensionality.

An unbiased estimate of the squared Maximum Mean Discrepancy (MMD) between two distributions p and q can be computed using the following formula originally proposed by Gretton et al. (2012) and applied by Rabanser et al. (2019) and Roschewitz et al. (2024) in the context of shift detection:

$$\mathrm{MMD}^2 = \frac{1}{m^2 - m} \sum_{i=1}^{m} \sum_{j \neq i}^{m} \kappa(z_i, z_j) + \frac{1}{n^2 - n} \sum_{i=1}^{n} \sum_{j \neq i}^{n} \kappa(z'_i, z'_j) - \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \kappa(z_i, z'_j)$$

where:

- $\{z_i\}_{i=1}^{m} \sim p$ and $\{z'_j\}_{j=1}^{n} \sim q$ are samples drawn from distributions p and q, respectively.
- κ is a positive semi-definite kernel function that defines similarity in the embedding space (e.g. the Radial Basis Function).
- The first term measures average similarity within distribution p.
- The second term measures average similarity within distribution q.
- The third term measures similarity between distributions p and q.

Simply put, the MMD statistic captures how different two distributions are by comparing intra- and inter-distribution similarities. A larger MMD value indicates a greater dissimilarity between p and q.

A commonly chosen kernel κ for MMD is the Radial Basis Function (RBF) kernel, defined as:

$$\kappa(z, \tilde{z}) = \exp\left(-\frac{\|z - \tilde{z}\|^2}{\sigma}\right)$$

(Rabanser et al., 2019; Roschewitz et al., 2024)

where:

- z and $\tilde{z}$ are feature vectors representing two data points in the embedding space.
- $\|z - \tilde{z}\|^2$ is the squared Euclidean distance between the two feature vectors z and $\tilde{z}$, calculated as the sum of squared differences in each dimension of the vectors.
- σ (sigma) is a parameter that represents the width of the kernel. It controls the sensitivity of the kernel to the distance between the points. Smaller values of σ make the kernel sensitive to local distances, while larger values make it more global and less sensitive to individual differences. Rabanser et al. (2019)

derived σ as the median of the pairwise Euclidean distances between all data points in the feature space. This adaptive selection ensures that the kernel is appropriately scaled to the data distribution, making it robust to varying densities across different datasets.

- The term $\exp\left(-\frac{1}{\sigma}\|z - \tilde{z}\|^2\right)$ is the kernel function itself, which computes a similarity score between the vectors z and $\tilde{z}$. The closer the two vectors are, the higher the similarity score (close to 1), and as the distance between them increases, the score decreases exponentially, approaching 0.

Permutation Testing

To assess the significance of an observed test statistic like Maximum Mean Discrepancy (MMD), a permutation test can be used to compute a p-value. The null hypothesis assumes that the two samples, X and Y, are drawn from the same distribution—implying that any observed difference could be due to chance.

The process begins by computing the **observed MMD** between X and Y. If the null hypothesis holds, this value should be similar even after randomly reassigning the samples to the sets. The permutation test proceeds as follows:

1. Combine X and Y into a single dataset.
2. Randomly shuffle the combined data.
3. Split it into two new sets with the same sizes as X and Y.
4. Compute the MMD between these two subsets.
5. Repeat steps 2–4 many times (e.g., 1000 permutations) to generate a distribution of MMD values under the null hypothesis.

The **p-value** is calculated as the proportion of permutations in which the MMD is greater than or equal to the observed value. For example, if this occurs 25 times out of 1000 permutations, the p-value is:

$$\text{p-value} = \frac{25}{1000} = 0.025$$

A small p-value (e.g., below 0.05) suggests that the observed MMD is very unlikely under the null hypothesis, indicating a significant difference between the distributions. Conversely, a large p-value suggests that the observed discrepancy could actually have occurred by chance. This method provides a statistical way to determine whether a measured distance between datasets reflects a real distributional shift (Pesarin and Salmaso, 2010).

4.7 Proxy A-Distance*

In its original formulation, *Proxy-A-Distance* (PAD) evaluates the error of a binary domain classifier trained on raw input features. However, this approach can be overly sensitive to superficial domain differences that may not impact task performance. To address this, a modified version known as PAD* has been proposed. PAD* assesses the distinguishability of domains based on intermediate representations learned by a task-specific model, rather than raw inputs. Specifically, after training a task model composed of a feature extractor G_f and a label predictor G_y on source domain data, the feature extractor is frozen. A separate domain classifier G_d^* is then trained to discriminate between—i.e. distinguish—the intermediate features $G_f(x)$ from source and target examples. It will subsequently be referred to as *domain discriminator*. The PAD score is calculated based on the *generalisation error* $E(G_d^*(G_f(x)))$ of this domain classifier on held-out intermediate features of the original task classifier, using the formula:

$$\text{PAD}^* = 1 - 2E(G_d^*(G_f(x))).$$

A PAD* value of 0 corresponds to a random classifier (i.e., no domain discrepancy), while a value of 1 indicates perfect discrimination between domains. This makes PAD* a more targeted measure of domain shift, focusing on the task-relevant representations rather than superficial input differences.

To extend the PAD* to scenarios involving more than two domains, a multi-class version is proposed for this work, denoted as $\text{PAD}^*_{\text{multi}}$. This variant quantifies the overall domain shift across multiple scanner domains. Instead of a binary domain classifier, a multi-class discriminator is trained to predict the domain label (e.g., one of five scanners) based on the intermediate representations extracted by a frozen feature extractor G_f, which was previously trained on the primary classification task. The *mean absolute error* (MAE) of this domain classifier—which is essentially the aforementioned *generalisation error*—is computed as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{\hat{y}_i \neq y_i\}$$

where n denotes the number of samples, y_i the true domain label, and $\hat{y}_i$ the predicted domain label for sample i. The multi-class PAD* score is then calculated by normalizing this error against the expected error of a random classifier, $\text{MAE}_{\text{random}}$, using the formula:

$$\text{PAD}^*_{\text{multi}} = 1 - \frac{\text{MAE}}{\text{MAE}_{\text{random}}}$$

In the case of balanced domain classes, the random error is given by $\text{MAE}_{\text{random}} = 1 - \frac{1}{K}$, where K represents the number of domain classes (e.g., $K = 5$ results in $\text{MAE}_{\text{random}} = 0.8$). This normalisation ensures that $\text{PAD}^*_{\text{multi}} = 1$ indicates perfect discrimination between domains, while $\text{PAD}^*_{\text{multi}} = 0$ corresponds to performance

at chance level. This version of the method provides a straightforward metric for quantifying domain shift in a multi-class context.

4.8 Principal Component Analysis

Principal Component Analysis (PCA) is a frequently used statistical technique for dimensionality reduction of data by identifying its most essential features. This can be useful when working with high-dimensional data, such as hidden feature embeddings extracted from machine learning models (Roschewitz et al., 2024), to reduce computational overhead, filter out noise, or enable visualisation, e.g. by projecting the data into a 2D space.

It works by projecting several original variables into a smaller number of new, uncorrelated variables—called principal components—that retain as much of the original data's variance as possible. Essentially, PCA looks for linear combinations of the original variables that preserve the underlying variance structure. The principal components are ordered by the amount of variance they explain, with the first capturing the greatest variance, the second capturing the next greatest, and so on. Usually, most of the variance is expressed by the first few top components, but there is generally a trade-off between simplicity and information preservation. (Greenacre et al., 2022)

In the first step, PCA computes the covariance matrix of the data, which describes how much each pair of features varies together. This matrix captures relationships between features, including correlation. Then the so-called *eigenvectors* and *eigenvalues* of this covariance matrix are calculated. The eigenvectors represent directions in the feature space—i.e. new axes along which the data varies the most—while the eigenvalues indicate how much variance is captured in each direction. Finally, by selecting the top k eigenvectors (those with the largest eigenvalues), the data is projected into a lower-dimensional space that retains the most significant variance characteristics of the original dataset (Abdi and Williams, 2010; Ringnér, 2008; Greenacre et al., 2022)

4.9 Performance Metrics

4.9.1 F1 Score

In binary classification tasks, two fundamental metrics for evaluating model performance are *precision* and *recall*. These metrics are based on the concepts of true positives (TP), false positives (FP), and false negatives (FN). *Precision* measures the proportion of positive predictions that are correct and is defined as:

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

Recall, also referred to as sensitivity or the true positive rate (TPR), measures the proportion of actual positive instances that are correctly identified by the model. It is defined as:

$$\text{Recall (TPR)} = \frac{TP}{TP + FN}$$

While precision focuses on the correctness of positive predictions, recall emphasizes the model's ability to identify all positive cases.

The *F1 score*, also known as F-measure, combines both metrics into one value by computing their harmonic mean. It provides a balance between precision and recall and is particularly useful when the class distribution is imbalanced or when both false positives and false negatives carry significant consequences (Manning et al., 2008; Hossin and Sulaiman, 2015).

The F1 score is given by:

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

.

The value of the F1 score ranges from 0 to 1, with higher values indicating better overall performance in terms of both capturing relevant instances and avoiding false positives.

4.9.2 Reciever Operating Characteristic (ROC)

The *Receiver Operating Characteristic* (ROC) curve is a graphical representation of a binary classifier's performance across a range of threshold values. It plots the true positive rate (TPR), also known as recall or sensitivity, against the false positive rate (FPR), which is defined as the proportion of negative instances incorrectly classified as positive. The false positive rate is computed as:

$$\text{FPR} = \frac{FP}{FP + TN}$$

.

By varying the decision threshold applied to the model's output, which is continuous (i.e. softmax probabilities), different trade-offs between TPR and FPR can be visualised on the ROC curve.

The *Area Under the ROC Curve* (AUC-ROC) summarises this trade-off as a single value ranging from 0 to 1. A value of 1.0 indicates perfect discrimination between the positive and negative classes, while a value of 0.5 corresponds to random guessing. AUC-ROC is widely regarded as a robust and threshold-independent performance measure, particularly useful in situations with imbalanced class distributions. Intuitively, the AUC-ROC value represents the probability that, if you randomly select

one positive instance and one negative instance, the classifier will assign a higher probability score to the positive instance than to the negative instance. The theoretical foundations of ROC analysis and its application in diagnostic systems have been widely discussed in the literature (Metz, 1978; Hanley and McNeil, 1982; Fawcett, 2006).

4.9.3 False Negative Rate (FNR)

The *False Negative Rate* (FNR) represents the proportion of instances from a class that are incorrectly predicted as belonging to a different class. In the context of domain discrimination in PAD*, it reflects how often samples from a particular scanner domain are misclassified. The FNR is computed as:

$$\mathrm{FNR}_i = \frac{\mathrm{FN}_i}{\mathrm{TP}_i + \mathrm{FN}_i} = 1 - \mathrm{Recall}_i$$

where TP_i and FN_i represent the number of true positives and false negatives for class i, respectively (Powers, 2011). For a domain discriminator model, a high FNR indicates that samples from domain i are frequently confused with others, suggesting low distinguishability in the learned representation space.

4.9.4 False Discovery Rate (FDR)

The *False Discovery Rate* (FDR) measures the proportion of instances predicted to belong to a given class that actually originate from a different class. In other words, it captures how often predictions for domain i are incorrect. The FDR is defined as:

$$\mathrm{FDR}_i = \frac{\mathrm{FP}_i}{\mathrm{TP}_i + \mathrm{FP}_i} = 1 - \mathrm{Precision}_i$$

where TP_i and FP_i are the number of true positives and false positives for class i, respectively (Powers, 2011). Receiving a high FDR in the domain discrimination context suggests that many samples are falsely assigned to domain i, potentially indicating either class bias or feature overlap with other domains.

4.9.5 Pearson Correlation Coefficient

The *Pearson correlation coefficient* is a statistical measure of the linear relationship between two continuous variables (Pearson, 1896). Given two vectors of paired observations $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, the Pearson correlation r is computed as

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}},$$

where $\bar{x}$ and $\bar{y}$ are the means of x and y, respectively. The coefficient r ranges from -1 (perfect negative correlation) to $+1$ (perfect positive correlation), with a value of 0 indicating no correlation.

In the context of model evaluation, the Pearson correlation is particularly useful for detecting trends or patterns within results (Kuhn et al., 2013). For instance, in BBSD, it can be applied to KS distances across different model architectures to assess the consistency in how models capture distributional differences.

4.9.6 Coefficient of Variation

The *coefficient of variation* (CV) measures how much a set of values varies relative to its mean. It is defined as the ratio of the standard deviation σ (also called STD) to the mean μ:

$$\mathrm{CV} = \frac{\sigma}{\mu}$$

(Everitt, 1998)

Because it expresses variability relative to the average, the CV is especially useful for comparing variation across different scales or measurement units (Abdi, 2010). For example, a CV of 0.2 means the standard deviation is 20% of the mean, while a CV of 1.0 means the spread is as large as the average value itself.

In this work, the CV is used to summarize how much domain shift metrics—such as KS distances—fluctuate across scanner pairs. A high CV suggests that the results vary a lot depending on the scanner pair, while a low CV points to more consistent behaviour.

5 Methodology

This section provides details about the procedures and methods used in the experiments to ensure reproducibility, support critical assessment of the findings, and provide a foundation for future research. Each methodological step is fully described, including how the different domain shift detection approaches were applied. For additional clarity, visual representations of the experimental setups are provided in Appendix Figures 22 (input-space BBSD + MMD), 23 (latent-space MMD), and 24 (multi-class PAD*).

To improve comprehensibility given the complex experimental setup, each classifier model is denoted by the scanner identifier from which its training data originates, using the format `scanner_model` (e.g., `cs2_model`, `nz20_model`, `nz210_model`, `p1000_model`, `gt450_model`). Similarly, test sets are denoted as `scanner_test`, indicating that the data was acquired using the respective scanner. The general terms `scanner_model` and `scanner_test` refer to any model or test set associated with a

given scanner. Full, explicit descriptions are alternatively used where this is deemed beneficial for understanding.

Additionally, the terms *source-centric* and *target-centric* are used to describe the direction of evaluation. A source-centric view focuses on the results of a single `scanner_model` or reference domain across multiple test sets, while a target-centric view centers on a single `scanner_test` evaluated using multiple `scanner_models` or source domains.

5.1 Dataset and Preprocessing

5.1.1 Dataset

The dataset employed for all subsequent experiments is the *Multi-Scanner Canine Cutaneous Squamous Cell Carcinoma Histopathology Dataset* introduced by Wilm et al. (2023c), which is "a multi-scanner version of the SCC subset of the publicly available CATCH dataset" (Wilm et al., 2023a).

It comprises 220 Whole Slide Images (WSIs), derived from 44 canine histopathology samples, each digitised using five different microscopic whole slide scanners: Aperio ScanScope CS2 (Leica, Germany, 0.25 µm/pixel), NanoZoomer S210 (Hamamatsu, Japan, 0.22 µm/pixel), NanoZoomer 2.0-HT (Hamamatsu, Japan, 0.23 µm/pixel), Pannoramic 1000 (3DHISTECH, Hungary, 0.25 µm/pixel), and Aperio GT 450 (Leica, Germany, 0.26 µm/pixel) (Wilm et al., 2023a). For simplicity reasons, these scanners will be referred to as "cs2", "nz210", "nz20", "p1000" and "gt450" respectively. They represent the domains between which the scanner-induced shift is supposed to be measured. Figure 1 displays sample 01 as digitised by the five different scanners.

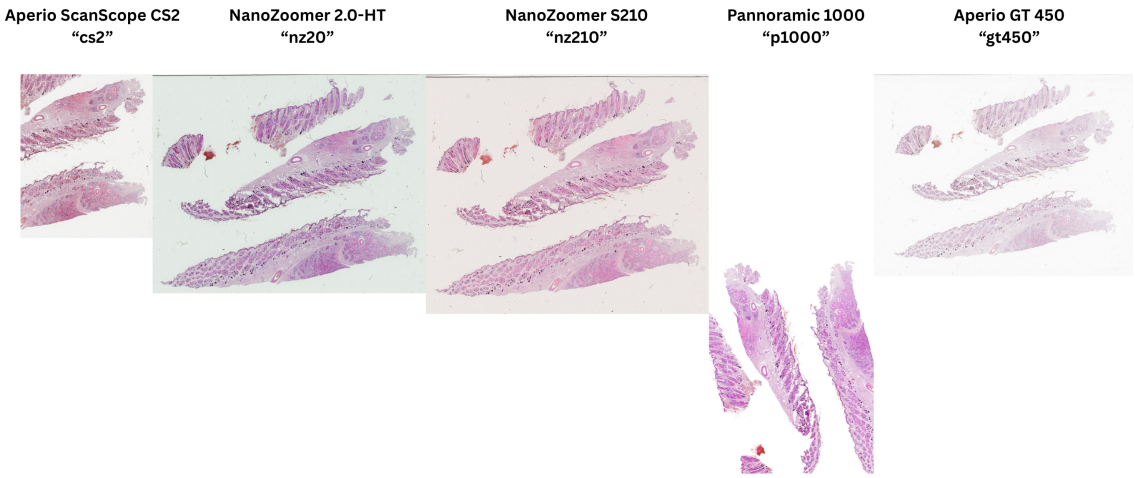

Figure 1: Tissue sample (ID 01) by all five scanners side-by-side. Slides were downscaled without cropping and size ratios between slides were preserved.

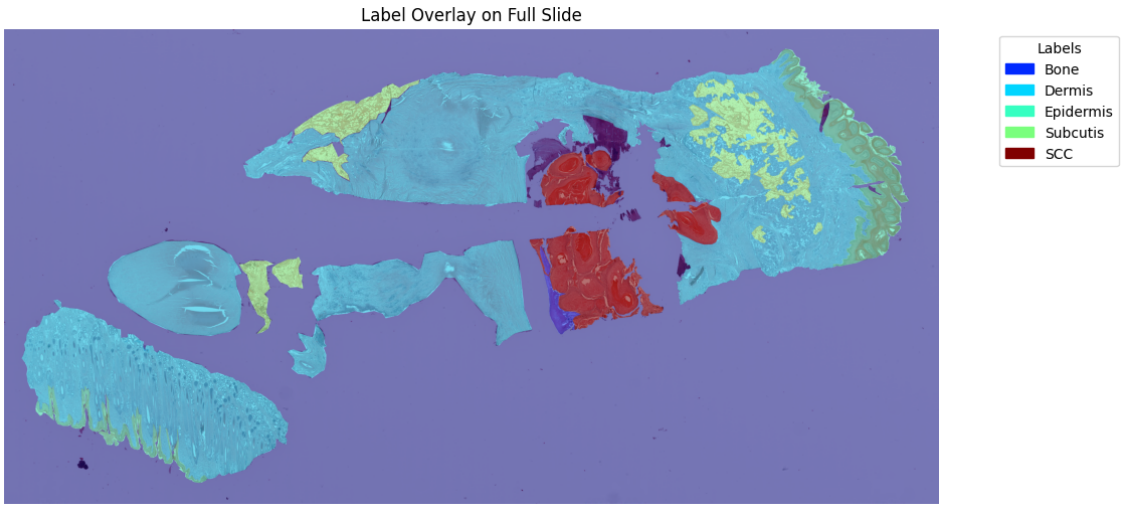

Figure 2: Label mask visualisation on example slide (ID 10 of cs2).

This dataset was selected because it isolates scanner-induced acquisition shifts while minimizing anatomical variability, making it well-suited for the subsequent experiments. However, it is important to note that in practice, multiple types of shifts that apply to different characteristics often occur at the same time Roschewitz et al. (2024) and same-anatomy samples are not the rule.

The WSIs were accompanied by segmentation polygons providing the coordinates of labelled regions, from which pixel-wise label masks were derived. These masks annotated each pixel with one of the following 14 classes: Background (0), Unassigned (-1), Bone (1), Cartilage (2), Dermis (3), Epidermis (4), Subcutis (5), Inflammation/Necrosis (6), Melanoma (7), Plasmacytoma (8), Mast Cell Tumor (9), Peripheral Nerve Sheath Tumor (10), Squamous Cell Carcinoma (11), Trichoblastoma (12), and Histiocytoma (13). Of these 15 different possible label values, labels 7 through 13 correspond to tumor tissue classes (positive class 1). The label mask is visualised in Figure 2.

5.1.2 Preprocessing

The dataset split used in the experiments was directly adopted from Wilm et al. (2023b) to ensure consistency with prior work and to avoid potential sampling bias due to lacking domain-specific medical expertise.

The WSIs vary significantly in spatial dimensions, with the smallest slide measuring $3,712 \times 2,789$ pixels and the largest reaching $12,288 \times 6,560$ pixels. With the exception of scanner "p1000", slide dimensions are inconsistent even intra-scanner. The slide with the largest absolute deviation from a square aspect ratio is *"scc_35_nz210.tif"*, which has a width of 9600 pixels, a height of 3872 pixels, and an aspect ratio of 2.48. In contrast, the image with an aspect ratio closest to 1 is *"scc_04_cs2.tif"*, measuring 5849×5850 pixels, nearly resulting in a perfect 1.00 ratio. To illustrate this further, Table 2 shows the average image dimensions per

Table 1: Assignment of slides to training, validation, and test splits.

Dataset Split	**Slide IDs**
Train	{01, 02, 04, 05, 06, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30, 31, 35, 36, 38, 39, 40, 41, 42, 43}
Validation	{03, 09, 25, 26, 37}
Test	{07, 08, 10, 11, 16, 32, 33, 34, 44}

scanner and Appendix Figure 34 visualises the width and height distributions of the slides per scanner as histograms.

Table 2: Average image dimensions (width + height) and average aspect ration (width/height) for each scanner.

Scanner	**Avg. Size**	**Avg. Ratio (width / height)**
cs2	11056.11	1.36
nz20	14108.36	1.48
nz210	14317.45	1.46
p1000	15341.00	0.41
gt450	12583.18	1.52

To accommodate these size differences while retaining relevant histological context, all WSIs were subdivided into patches of size 224×224 pixels while preserving the 3 RGB channel dimensions. Patches were extracted by calculating a patch grid, visualised in Figure 3. Minimal patch overlap was dynamically calculated based on height and width to ensure full slide coverage. The resulting patches of size $224 \times 224 \times 3$ are directly compatible with the ResNet18, as well as the DINO-base and DINOv2-base models that were chosen for the experiments.

For the binary classification task, all patches that contained any tumor-associated labels (classes 7 through 13) were labelled as class 1 (tumor). Non-tumor tissue types (classes 1 through 6) were assigned to class 0 (non-tumor). Patches labelled as Background (0) or Unassigned (-1) were excluded entirely from both the training and evaluation phases, in order to avoid model overfitting on information-empty regions. No additional preprocessing was applied beyond patch extraction and label filtering. Preprocessing of the patches was intentionally kept minimal to preserve the naturally occurring shifts present in the original images.

Table 3 shows the class distribution of the patches for each split subset per scanner.

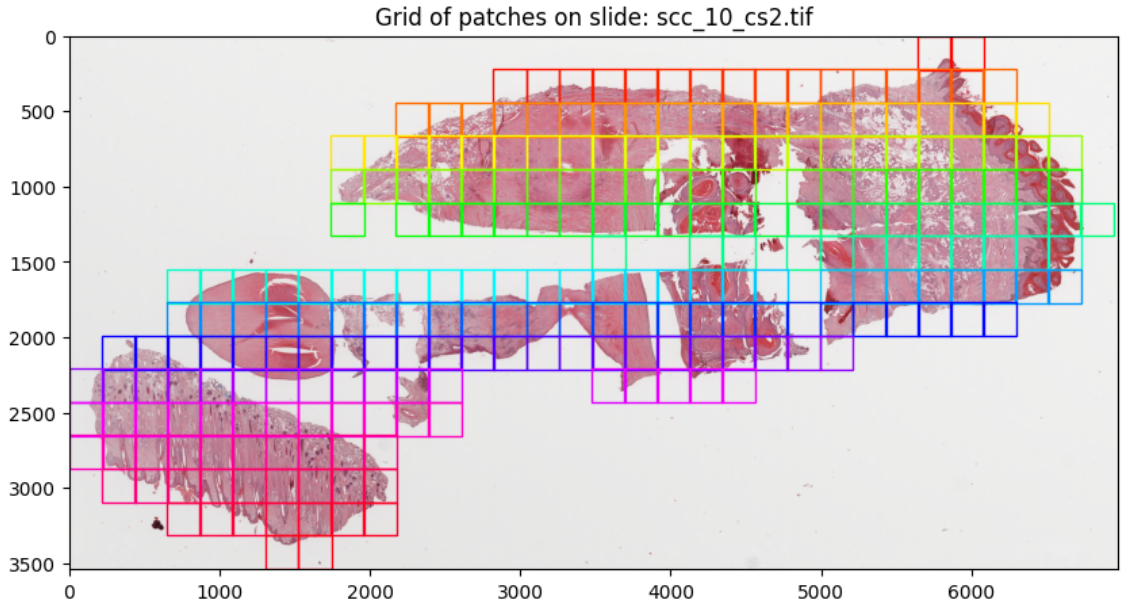

Figure 3: Patch grid visualisation on example slide (ID 10 of scanner cs2).

Table 3: Number of patches per data split and scanner. The first row of each subset shows the total number of patches; the second row shows the percentage of class 1 (tumor) samples.

Subset	**cs2**	**nz20**	**nz210**	**p1000**	**gt450**
Train	11,623	13,966	14,773	10,824	10,735
Class 1 (%)	38.41	37.52	37.22	40.46	38.29
Validation	1,845	2,253	2,355	1,831	1,710
Class 1 (%)	42.66	43.45	42.63	42.49	43.74
Test	2,807	3,367	3,519	2,592	2,627
Class 1 (%)	40.04	38.73	38.87	40.08	39.70

5.2 BBSD for Input Space

To evaluate domain shifts in input space, a Kolmogorov–Smirnov (KS) test was applied to softmax outputs of neural network classifiers trained on the image patches of the histopathology slides as part of the BBSD method. The following subsections provide a detailed description of the methodology used. Additionally, Appendix Figure 22 visualises the BBSD setup for the input-space-based shift detection.

5.2.1 Architecture of the TwoLayerNN

In order to allow for a more educated evaluation of the results and to enable comparability of the input- and latent-space models, a separate model was trained for each scanner's data, although the BBSD method explicitly doesn't demand the training

of a dedicated predictor on the source domain. For this purpose, a simple two-layer neural network (multi-layer perceptron, MLP) was trained as a classifier on the tumor task for each scanner separately using only that scanner's training data. This TwoLayerNN model is a simple feed-forward neural network that begins with an input layer that takes the 224*224 image with three channels (RGB) as a flattened vector. The first fully connected layer projects this input to a hidden representation of size 512, followed by a ReLU activation function to introduce non-linearity. The transformed features are then passed through a second fully connected layer, which maps the hidden representation to the final output layer of size two. The model outputs two raw scores (logits) from this final layer—one for each class, i.e. tumor tissue and normal tissue. This simple architecture was chosen not only to keep computational overhead low, but also to minimize the influence of the generalisation capabilities of more sophisticated models, which could obscure the shift we aim to measure. After all, the BBSD method is explicitly designed to work with imperfect classifiers.

5.2.2 Classifier Training and Hyperparameters

For each of the five scanners, a separate model `scanner_model` was trained on that scanner's training data for the original tumor classification task. This was done for the TwoLayerNN and a pre-trained ResNet18—resulting in a total of 10 models. The ResNet18 serves primarily as a more complex reference model alongside the TwoLayerNN. It may later be used to assess the impact of classifier complexity and resulting performance differences.

The training process for the TwoLayerNN models follows a simple supervised learning pipeline. The loss function used is "CrossEntropyLoss", suitable for multi-class classification, and the "Adam" optimizer, which adapts the learning rate of each parameter for efficient training (Goodfellow et al., 2016; Kingma and Ba, 2014). Training was conducted for a fixed 12 epochs using a learning rate of 0.0001 across experiments to support comparability and reproducibility. Note that the values of the hyperparameters were determined empirically and are not optimised for peak performance, as the main goal was not to develop a highly accurate classifier, but to provide a consistent basis for shift detection.

The ResNet18 model was initialised with weights pre-trained on the ImageNet-1k natural image dataset and fine-tuned using the same training pipeline for just 8 epochs. This number of epochs was chosen empirically, as it indicated convergence.

In each epoch, the inputs undergo a forward pass through the network to produce predictions. These predictions are compared to the ground truth using the loss function, and the gradients are computed via backpropagation. The optimizer then updates the model's parameters based on these gradients. Throughout this phase, the number of correct predictions is tracked to monitor training accuracy.

Then, to conclude each epoch, the model switches to evaluation mode, where no weights are updated. The validation data is passed through the model, and the predictions are compared with the true labels to calculate validation accuracy as well as

the F1 score. Because the TwoLayerNN model failed to learn in some runs—possibly because of a bad weight initialisation that led to vanishing gradients—this training process is repeated on three individual model instances, and the one with the highest F1 score on the last epoch's validation phase is finally saved as the designated classifier model for BBSD.

The training pipeline thus leaves us with 5 separate classifier/predictor model instances per architecture, each trained on the data of one of the five scanners on the same task and using the same hyperparameters.

Class Balancing:

Due to the class imbalance in the patch distributions of each subset (see Table 3, the dataloader was modified to approximately balance the classes using a weighted random sampler without replacement. For each subset, this sampler generates a new dataset per epoch, containing a roughly equal number of unique samples from each class. Class weights are computed as the inverse of their frequency, increasing the sampling probability of underrepresented classes. The sampler draws samples from both classes until the original, unbalanced dataset size is reached, ensuring approximately balanced classes. Although sampling is done with replacement—allowing the same sample to appear more than once—it is repeated independently each epoch. This approach balances class distribution while maintaining dataset size, trading off some data diversity for class-bias mitigation. For training, the inputs are then fed to the network in batches of size 64 with 4 workers for parallelised processing under moderate memory load.

5.2.3 BBSD

After training, each model `scanner_model` was used to compute softmax outputs for the test data `scanner_test` from all scanners—including the scanner it was trained on (source domain data). The softmax values were computed separately using the raw logits produced by the model's final layer. Notably, the TwoLayerNN and the ResNet18 themselves do not include a softmax activation at the end, as the used training loss function (CrossEntropyLoss) internally applies softmax during optimisation. This ensures numerical stability during training and allows for explicit control over softmax computation during evaluation.

Subsequently, to quantify distributional differences, the two-sample KS test was applied in a pairwise manner. For each pair of scanners, the test compared the distribution of softmax outputs from the model's source domain test set against the outputs from each of the five other test sets. One of these test sets is the source domain test set itself, which should return zero distance, while four of the test sets are from unseen (target domain) scanners. Since each model was trained on only one scanner's data, scanner pairs were treated as directional: for example, softmax outputs for the test set of scanner A `scannerA_test` and test set of scanner B `scannerB_test` obtained from a model trained on scanner A `scannerA_model` are distinct from those obtained when extracting softmax outputs for the same test sets

using a model that was trained on scanner B `scannerB_model`. Consequently, the test was applied to each scanner pair in both directions.

To ensure robustness, the KS test was bootstrapped with 100 iterations per scanner pair. In each iteration, softmax outputs were resampled to generate comparable subsets. This was necessary because the number of softmax outputs varied across scanners, reflecting differences in the number of patches per slide and therefore in the size of each test set. Repeated resampling mitigated the impact of these disparities and enabled stable statistical comparisons across scanner domains. For each class in the binary classification task, the resulting KS distance (ranging from 0 to 1) and the associated p-value were computed for every bootstrap iteration, and the final reported values were taken as the median across all iterations.

5.3 BBSD for Latent Space

In addition to the input-space shift analysis, a similar procedure was conducted for the latent space to investigate whether scanner-induced domain shifts persist after representation learning through foundation models. Feature vectors were extracted from each image patch using pre-trained DINO-base and DINOv2-base models without any fine-tuning. These representations were then used to train instances of the aforementioned two-layer neural network separately for each scanner. Appendix Figure 23 visualises the latent-space BBSD and MMD experiments.

5.3.1 Feature Extraction

Hidden representations of all patches were extracted using a DINO (ViT-Base/16, 224) and a DINOv2 (ViT-Base/16, 224) model instance, with the former being pre-trained on ImageNet-1k and the latter on the larger LVD-142M dataset. They serve as prominent and well-established examples of Vision Transformer-based foundation models and are publicly accessible. The models were not fine-tuned to the downstream task in order to assess their adaptability to the medical domain and minimize computational overhead—thus maintaining the lightweight, flexible nature of the BBSD approach.

Importantly, both models expect inputs of size $224 \times 224 \times 3$, allowing the patches defined by the patch grid to be directly fed into the foundation models without resizing or flattening. From each model, only the global CLS token was used as the patch representation for the downstream classification task. Both models produce feature vectors of size 768, which were subsequently used to train the latent-space classifiers. Matching output dimensionality was deemed essential to ensure the comparability of the experimental results.

5.3.2 Training and Hyperparameters

To ensure comparability with the input-space analysis and to isolate the effect of the feature space alone, the exact same neural network architecture and training configuration were used: The TwoLayerNN featured a hidden layer with 512 neurons, was trained for 12 epochs, and used a learning rate of 0.0001. As before, "CrossEntropyLoss" was used as a loss function and "Adam" as the optimizer. The only difference in this setting is that the inputs are no longer raw 224×224×3 image patches, but 768-dimensional feature vectors extracted by the foundation models. Accordingly, the input size of the two-layer neural network is adjusted to 768. **Since the rest of the training procedure remains identical to the input-space pipeline, it is not elaborated further to avoid redundancy**. This consistency in model design ensures that any observed differences in performance can be attributed to the input representation rather than changes in architecture or optimisation.

For clarity and simplicity in terminology, the TwoLayerNNs trained on DINO and DINOv2 features are referred to as using the DINO and DINOv2 architectures, respectively. While this naming is technically imprecise—since the classification heads are TwoLayerNNs—the architectural labels refer to the backbone responsible for feature extraction, which plays the primary role in determining the model's representational power. Thus, we distinguish between TwoLayerNN and ResNet18 for input space, and DINO and DINOv2 for latent space.

As in the previous experiment, a weighted random sampler with replacement was used to approximately balance the classes in each subset while preserving the original size of each subset. Class weights were computed as the inverse of class frequency, and a new balanced set of samples was drawn for each epoch. Batches of size 64 were processed using 4 parallel workers.

5.3.3 BBSD

Analogously to the input-space experiment, after training, softmax outputs were computed for the latent features corresponding to the test data from all scanners using the previously trained classifier. As before, the two-sample Kolmogorov–Smirnov test was applied pairwise to compare the distributions of source domain and target domain softmax outputs. The test was bootstrapped with 100 iterations to account for variations in test set sizes across scanners due to differences in the number of patches per slide. For each class in the binary classification task, the KS test returned a p-value and a KS distance metric ranging from 0 to 1, both of which were aggregated using the median across bootstraps to produce stable, comparable shift estimates in latent space.

5.4 MMD for Latent Space

To further quantify distributional differences across scanners, the Maximum Mean Discrepancy (MMD) was applied directly to the previously extracted feature vec-

tors from the pre-trained DINO and DINOv2 foundation models, as visualised in the latent-space experiment visualisation in Appendix Figure 23. This follows the "Duo" methodology proposed by Roschewitz et al. (2024), who demonstrated the value of combining output-based and feature-based shift detection methods. The implementation of the MMD permutation test used here was largely adapted from their work. The entire procedure was performed separately for features derived from DINO and DINOv2, allowing for a comparison of their sensitivity to domain shifts. Prior to MMD computation, Principal Component Analysis (PCA) was applied to the combined feature vectors of each scanner pair, reducing dimensionality to 32 components in order to retain the most relevant information while simplifying the comparison.

The MMD statistic was computed using a kernel-based approach with a Radial Basis Function (RBF) kernel. The kernel bandwidth (σ) was selected adaptively based on the median of all pairwise Euclidean distances between samples, as proposed by Rabanser et al. (2019), which adapts the kernel's sensitivity to the scale of the data. The MMD statistic itself is derived from the difference between intra-distribution similarities (within each scanner) and inter-distribution similarities (between scanners), providing a quantitative measure of how dissimilar the two distributions are in the latent space.

To assess the statistical significance of the observed MMD values, permutation testing was employed. Samples from both scanner distributions were pooled and randomly shuffled, and the MMD statistic was recalculated for each of 1000 permutations to simulate the null hypothesis that the distributions are identical. The p-value was then calculated as the proportion of permuted MMD values that exceeded the observed value.

MMD was computed for each scanner pair. In contrast to the KS test, the order of scanner pairs is irrelevant, and reversing it will return the same value. This is because MMD calculates the distance between distributions in the feature space without involving any trained models, making the test symmetric with respect to the input distributions. By comparison, the KS test was asymmetric, since the first scanner name in each pair represented the source domain test data, and the second scanner represented any of the five test sets, of which four are target domain data. As a result, reversing the order of scanners in the KS test will likely lead to different results due to differences between model instances trained on different domains.

5.5 PAD* (multi-class)

PAD* was chosen as an additional metric to enable a more differentiated and meaningful evaluation of the potential domain shift. While PAD* was originally designed to measure the shift between two domains, an adapted multi-class variant ($\text{PAD*}_{\text{multi}}$) is proposed here. In this setting, the domain discriminator G_d^* is trained to distinguish among all five scanner domains, providing a single aggregated measure that quantifies the overall distributional shift across domains. This experimental setup is visualised in Appendix Figure 24.

5.5.1 Training of the Feature Extractor

A dedicated ResNet18 model, pre-trained on ImageNet-1k, was fine-tuned on the tumor classification task to obtain task-specific hidden representations for use in the PAD* method. The model was trained for 8 epochs using a learning rate of 0.0001. These values were selected empirically and provided stable convergence in practice. To prevent scanner-specific biases from influencing the reduced representations, the training data from *all* five scanners was combined into a single dataloader. After training, the classification head was removed from the network by discarding the final fully connected layer. The resulting model was then used as a feature extractor to generate 512-dimensional hidden representations for each sample patch.

In addition to this dedicated fine-tuned ResNet18, two more off-the-shelf models were used as feature extractors as reference. Namely, a ResNet18, pre-trained on ImageNet-1k, without fine-tuning to observe the potential impact of task-specific features versus general image features on the downstream domain discriminator performance and subsequently PAD* results. The third model is a DINO model, also pre-trained on ImageNet-1k and chosen to inspect potential differences in the ability to extract domain-independent features between DINO and ResNet18 off-the-shelf.

Analogous to the class balancing approach used for training the task classifiers in BBSD, a random weighted sampler with replacement was employed to balance the class distribution within each training, validation, and test subset to mitigate class bias.

5.5.2 Training the Domain Discriminator

The fine-tuned and the off-the-shelf pre-trained ResNet18 models returned 512-dimensional feature vectors, which were then used to train a simple two-layer neural network (TwoLayerNN). It is the same architecture previously used in the BBSD experiments. However, the size of the hidden layer was reduced to 256 neurons to better match the lower input dimensionality. Thus, the first fully connected layer projects the 512-dimensional input to a hidden representation of size 256, followed by a ReLU activation function for non-linearity. The transformed features are then passed through a second fully connected layer, which maps the hidden representation to an output layer of size *5*. This final layer produces five raw scores (logits), one for each class—corresponding to the five scanners.

Unlike in the earlier setup, this TwoLayerNN was not trained to classify tumors. Instead, it was trained for domain discrimination. For this purpose, the original class labels were replaced with scanner identifiers. This allowed the model to learn to distinguish samples based on the scanner used for digitisation, relying on features relevant to the original classification task. This means that instead of the original tumor/non-tumor medical classification task with two classes, the model is now tasked to tell apart the different scanner domains, i.e. 5 classes.

Since the DINO model produces 768-dimensional feature vectors (as opposed to 512), the input size of its TwoLayerNN classification head was adjusted accordingly. Aside from this change, the training procedure remained identical.

As a result, three TwoLayerNN multi-class PAD* domain discriminators were trained: one using the pre-trained and fine-tuned ResNet18 features, one using the pre-trained ResNet18 features, and one using the pre-trained DINO features.

Balancing Class Distributions

As previously mentioned, scanners differ in slide sizes and thus the total number of extracted patches per scanner varies. Note that for the domain discriminator task, the class labels are the scanners from which the data originates. In a recent paper, Kunte et al. (2023) reported the reliability of the domain discriminator in PAD to be significantly diminished by class imbalance. To avoid such potential class bias, each group of subsets (training, validation, and test) was separately balanced by randomly *downsampling* all scanner-specific subsets to match the size of the smallest corresponding subset. In other words, all training subsets were sampled to the size of the smallest training subset among the scanners. The same was done independently for the validation and test subsets, yielding 10735 patches per training set, 1710 patches per validation set, and 2592 patches per test set.

This balancing is necessary to ensure that each scanner contributes the same amount of samples to the training so that the model learns the features of each scanner equally and doesn't overfit the domain with the most samples. Generally, the test and validation subsets do not influence the training and thus would not necessarily have to be balanced. Especially test sets are typically left unbalanced to retain real-world class prevalence. Since the scanner discrimination is an artificial task and balancing makes simple accuracy values and confusion matrices more interpretable, balancing was also applied to the test set in this case.

5.6 Evaluation Framework

To enable a consistent and interpretable presentation of results, the conventions used throughout the results section are introduced in the following.

Most metrics are computed for each combination of a trained model and an independent test set, resulting in matrices where each entry corresponds to a specific source domain and target domain pair.

For the original classification task, the distinction between models trained on a given scanner's data (`scanner_model`) and the scanners' test sets (`scanner_test`) is explicitly maintained. In all relevant result matrices, `scanner_model` is consistently placed on the y-axis and `scanner_test` on the x-axis. Each axis is clearly labelled to reflect this arrangement.

For the BBSD-based evaluation using the Kolmogorov–Smirnov (KS) test, a slightly different interpretation is required. Here, the y-axis refers not directly to the `scanner_model`, but to the softmax outputs corresponding to the model's source

domain test set—i.e., the in-domain data used as the reference distribution in the KS test. The x-axis, by contrast, represents the softmax outputs from the same model evaluated on each target domain (`scanner_test`). While the softmax values are derived from a model trained on the scanner indicated on the y-axis, the axis label in this context denotes the domain of the test set from which the softmax outputs were extracted.

5.6.1 Cross-Scanner Meta-Evaluation

In addition to the matrix-level results, a **cross-scanner meta-evaluation** is performed to reveal broader patterns in scanner behaviour. This analysis is structured along two complementary axes: the **source-centric view**, which considers each scanner in the role of the source domain (represented by matrix rows), and the **target-centric view**, which considers each scanner as the target domain (represented by matrix columns). In classifier performance evaluations, the source domain corresponds to the origin of the trained model (`scanner_model`), while in BBSD-based shift detection it refers to the softmax outputs from a model's in-domain test set. Conversely, the target domain (`scanner_test`) refers to the scanner providing the test set or comparison data.

To quantify trends under each of these perspectives, several summary statistics are computed across the rows and columns of the result matrices. These include the **mean values** to reflect overall performance or shift sensitivity, as well as the **coefficient of variation (CV)** to assess variability across scanners. Finally, to evaluate the influence of different model architectures, the **Pearson correlation** is calculated between the per-scanner results across architectures for each source domain (`scanner_model`).

5.6.2 Classifier Performance Evaluation

As described in the corresponding methodology subsections, a separate model was trained on each of the five scanners' training data for the original tumor classification task, serving as a foundation for BBSD-based shift detection.

In the input space, this involved training both a TwoLayerNN and a ResNet18 architecture per scanner, yielding a total of 10 models trained directly on raw image patches.

In the latent space, an equivalent setup was applied: TwoLayerNNs were trained on feature vectors extracted from a pre-trained DINO model and a pre-trained DINOv2 model, respectively, resulting in another 10 models. These latent-feature-based models are referred to as DINO and DINOv2, denoting the architecture responsible for feature extraction. Each of these 20 scanner-specific models is then evaluated on the test sets of all five scanners to naively detect scanner-induced class biases and confirm that the models are adequate for subsequent domain shift measurements.

To ensure the suitability of the original task classifiers—later used to extract softmax outputs for BBSD—the invertibility of the confusion matrix is verified on the source domain test data. Subsequently, the performance of the classifiers is further evaluated using accuracy, F1 score, the AUC-ROC and the class distributions of the classifier's predictions on the balanced binary test sets of each scanner. The F1 scores are used as the primary measure for adequate classifier performance, and their scores are compared across and within architectures.

As part of the cross-scanner meta-evaluation, source-centric and target-centric means and coefficients of variation (CVs) are computed based on the F1 scores. For each `scanner_model`, the source-centric mean summarises its performance across all five `scanner_test` sets, including its in-domain test set. Conversely, the target-centric mean captures how a given test set performs across models trained on each of the five scanners. To complement these averages, the corresponding CVs quantify the variability of the F1 scores that constitute each mean—indicating how consistent a model's performance is across test sets (source-centric) or how consistently a test set is handled by different models (target-centric). Additionally, scanner-wise Pearson correlation coefficients are calculated to assess architectural agreement. Specifically, for each scanner, the source-centric F1 scores of its corresponding models across different architectures are compared. For instance, the F1 scores of the `cs2` model when implemented as a ResNet are correlated with those of the `cs2` model when implemented as a TwoLayerNN. This pairwise analysis quantifies the extent to which different architectures yield similar cross-scanner performance patterns when trained on the same source domain.

5.6.3 BBSD Results Evaluation

For each of the 20 original task classifiers described previously, softmax outputs are extracted by evaluating the model on all five scanner-specific test sets. These outputs are then used to compute the Kolmogorov–Smirnov (KS) distance between 0 and 1 along with a p-value that expresses significance, following the BBSD methodology. Specifically, for a given classifier, the KS test quantifies the distance between the softmax distribution on its in-domain reference test set and that on each of the four out-of-domain target test sets. Notably, the Kolmogorov–Smirnov (KS) test, being a univariate test, yields a separate KS distance and p-value for each class. To control for multiple testing, the two p-values are Bonferroni-corrected by multiplying each by two—the number of tests performed. A KS test is considered significant if at least one of the corrected p-values falls below the significance threshold of 0.05. If both p-values are below the threshold, the smaller p-value and its corresponding KS distance are retained for the final result. These resulting distance values are then organised into matrices, where each row corresponds to a source domain (reference test set) and each column corresponds to a target domain, facilitating the dual source- and target-centric view, respectively.

The cross-scanner meta-evaluation for the KS results follows the same conceptual framework as the classifier performance evaluation, but with slight differences in

how the metrics are computed. In particular, both the mean and the coefficient of variation (CV) of the KS distances are computed from a reduced set of test set pairs: only those involving one of the four out-of-domain test sets are included, while the same-scanner (in-domain) test set is excluded. This applies to both the source-centric and target-centric perspectives. The exclusion is necessary because, for example, when using the `cs2` softmax outputs from the `cs2_model` as the reference, comparing them to the `cs2` test set outputs will trivially yield a KS distance of zero. Including these values would skew the results, artificially lowering the mean and distorting the CV. Thus, they are omitted to better capture meaningful cross-domain differences. The computation of these aggregate metrics is visualised on the example of mean BBSD values in Appendix Figure 32. It shows which values from the original result matrices correspond to which mean value and how the diagonal values (same-scanner values) are omitted from the mean. This same exclusion of diagonal entries is also applied when computing scanner-wise Pearson correlation coefficients across different architectures, in order to avoid trivial correlations that would bias the results.

For the class-wise p-values returned alongside the KS distance, a standard significance threshold of 0.05 is applied. Under this threshold, a measured distance is considered statistically significant unless the corresponding p-value exceeds 0.05, in which case the null hypothesis of equal distributions cannot be rejected. It is important to note that the p-values have already been Bonferroni-corrected prior to evaluation, so no further adjustment to the significance threshold is required.

5.6.4 MMD Results Evaluation

As described earlier, the Maximum Mean Discrepancy (MMD) values are computed between the hidden representations of pairs of scanner domains. These values are symmetric, meaning the order of the scanner pair does not affect the result—comparing scanner A to scanner B yields the same value as comparing B to A. Accordingly, the results are visualised as symmetric heatmaps.

Since MMD does not have a fixed upper bound, the maximum value of the heatmap colour scale is determined empirically. This ensures that the full range of observed values is accommodated while maintaining visual clarity and contrast. In general, MMD values should be interpreted in a relative sense and absolute values carry less meaning than their comparative differences across scanner pairs or model architectures.

MMD serves as a complementary metric for identifying domain shift patterns across scanner domains and model architectures. The values obtained from DINO and DINOv2 feature representations should be directly comparable, as both models produce feature vectors of identical size (768), and the same kernel with the same parameters is applied to both.

To support a more comprehensive analysis, mean MMD values, coefficients of variation (CV), and Pearson correlation coefficients are also computed. Unlike the

source- and target-centric meta-analysis performed for classifier and BBSD results, these statistics are all calculated along a single axis. This is because MMD values are solely defined by domain pairings, and flipping axes yields the same result due to symmetry.

To ensure the statistical significance of the observed distance values, all p-values from the permutation test with 1000 iterations are recorded to ensure that they lie below the significance threshold set at 0.05. To avoid plain zero values in the results, the index of the instances where the permutation MMD is higher than the observed MMD is set to a minimum of 1. Thus, all p-values should be ≥ 0.001.

5.6.5 PAD* (multi-class)

First, the feature extractor, trained on the original tumor task using all five scanners' data, is evaluated on a combined validation set of all scanners. The F1 score should be at least adequate for the feature extraction to be effective.

In the next step, the scanner discriminators, trained to differentiate between the five scanner domains, are evaluated using the extracted features of all five corresponding test sets, each downsampled to match the size of the smallest set to ensure balance. The primary evaluation metric is the normalised multi-class PAD* ($\text{PAD*}_{\text{multi}}$) score, which yields a value of 0 for random guessing and 1 for perfect discrimination. Additionally, overall accuracy and mean absolute error across all predictions are computed.

To complement the aggregated $\text{PAD*}_{\text{multi}}$ score and recover some of the granularity lost in moving from a binary to a multi-class setup, additional class-wise metrics are computed. Specifically, the F1 score, the false negative rate (FNR) and false discovery rate (FDR) are reported for each domain class. The FNR captures how often samples from a given domain are misclassified as belonging to another, while the FDR indicates how often predictions for a domain class are incorrect. These metrics enable a deeper analysis of which domains are more easily confused and which dominate the learned representation space, providing valuable insight into the asymmetries and structure of the domain shift that are not visible in the single scalar $\text{PAD*}_{\text{multi}}$ value.

6 Results

The following section presents the results of all conducted experiments. Details about the methodological evaluation framework are presented in the methodology section.

Analogous to the methodology section, in the following results, the terms DINO and DINOv2 refer to TwoLayerNN classifiers trained on features extracted by DINO and DINOv2 backbones, respectively. While the classification heads are identical,

the backbone architecture—responsible for feature extraction—is the key differentiator. Accordingly, TwoLayerNN and ResNet18 denote input-space (raw input image-trained) models, while DINO and DINOv2 refer to models that have been trained on DINO and DINOv2 features (latent-space representations).

Furthermore, to support more effective evaluation and identification of shift patterns, results for input- and latent-space models are typically presented side by side throughout the section.

6.1 Classifier Performance and Predictions

For the TwoLayerNN classifiers, all models returned an invertible confusion matrix on the balanced test set of their own data (source domain data). For the data of other scanners (target domain data), most of the confusion matrices were also invertible, except for:

- gt450 data tested on the `nz20` model — confusion matrix $\mathbf{C}$ not invertible: $\det(\mathbf{C}) = 0$
- `cs2` data tested on the `p1000` model — confusion matrix $\mathbf{C}$ not invertible: $\det(\mathbf{C}) = 0$
- gt450 data tested on the `p1000` model — confusion matrix $\mathbf{C}$ not invertible: $\det(\mathbf{C}) = 0$

To further investigate the behaviour of the responsible classifiers that returned these degenerate confusion matrices, the predicted class distributions are presented in Appendix Table 7 along with the naive accuracy scores and the F1 scores.

As the primary performance results, Table 6 shows the F1 scores of all five scanner-specific classifiers on all five scanners' test sets. To boost interpretability and highlight patterns and similarities, a side-by-side global view of heatmap matrices is shown in Figure 4.

The F1 scores show that from a source-centric perspective (per `scanner_model`), models trained on `nz20` and `p1000` data (`nz20_model` and `p1000_model`) consistently achieve lower performance scores than models trained on the other three scanners across all architectures. The `p1000_model`, in particular, is the only one to produce scores below 0.5 with the ResNet18 architecture when evaluated on `cs2`, `nz210`, and `gt450` domains. In contrast, `nz20_model` and `p1000_model` achieve relatively higher scores when evaluated on each other's data. Models trained on the remaining source domains generally perform better on `nz20` and `p1000` test data than the `nz20` and `p1000` models perform on theirs.

The AUC-ROC of all models on their respective source domain test data was also determined and plotted as a heatmap for visualisation in Figure 5. Note that ROC curves of the ResNet18 models, evaluated on their respective test sets, show significantly higher performance across all thresholds than those of the TwoLayerNN.

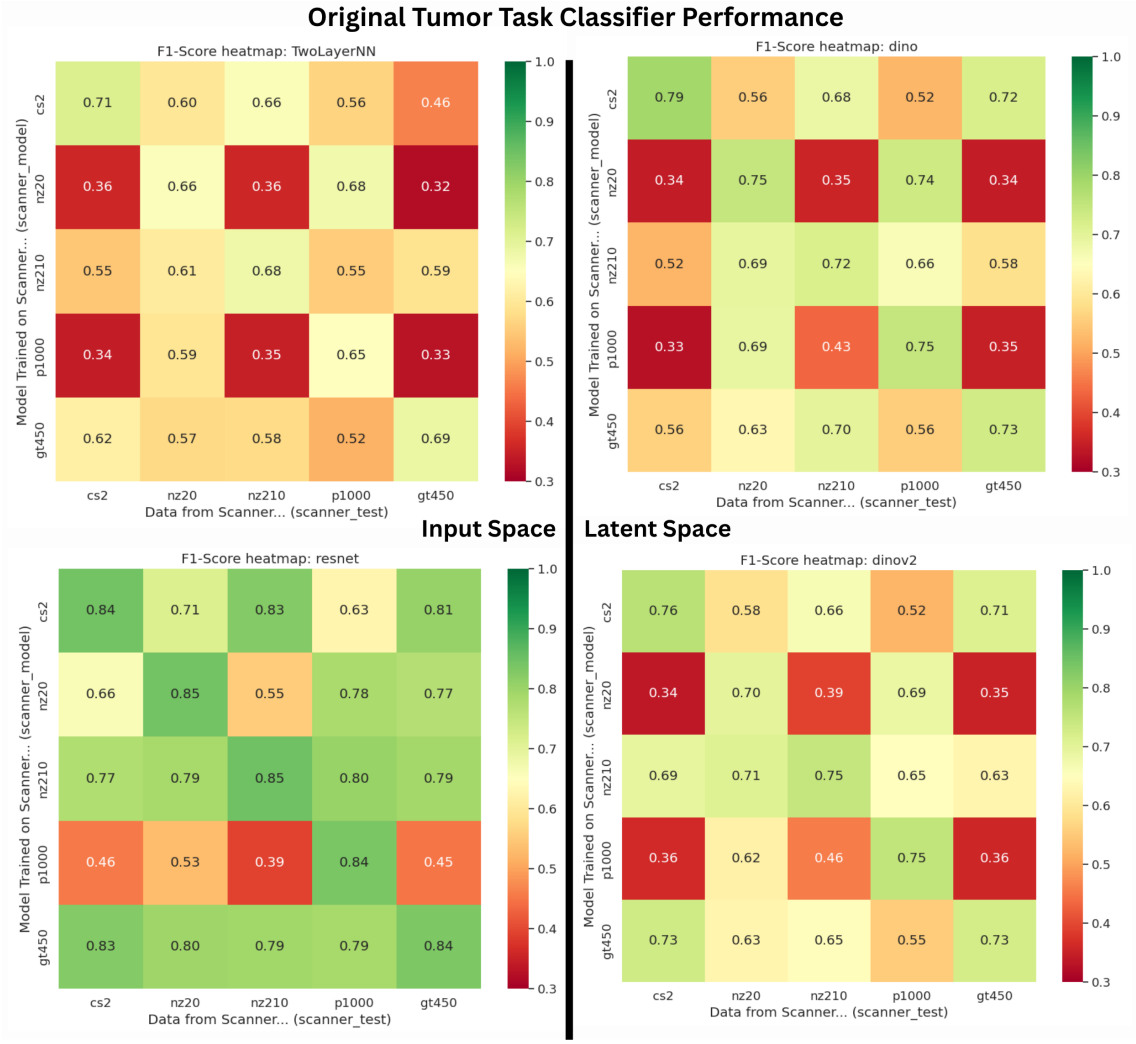

Figure 4: Overview of classifier F1-score heatmaps of input-space (left) and latent-space (right) classifiers.

In some cases, the ROC curves of a TwoLayerNN model tested on test data from all scanners exhibited unnaturally linear segments, as shown by the example in Appendix Figure 26—warranting further investigation.

Mean F1 Scores. For the source-centric analysis, the left heatmap in Figure 6 presents the mean F1 score per training scanner across all test sets per architecture, while the right heatmap shows the target-centric mean F1 score per test scanner across classifiers.

The left heatmap matrix in Figure 6 illustrates that models based on the ResNet18 architecture achieve the highest mean F1 scores across all test sets, ranging from **0.53** to **0.81**. In contrast, TwoLayerNN models utilizing DINO and DINOv2 as feature extractors yield comparable performance, with F1 scores ranging from **0.49** to **0.66**, though these scores are notably lower than those of the ResNet18-based

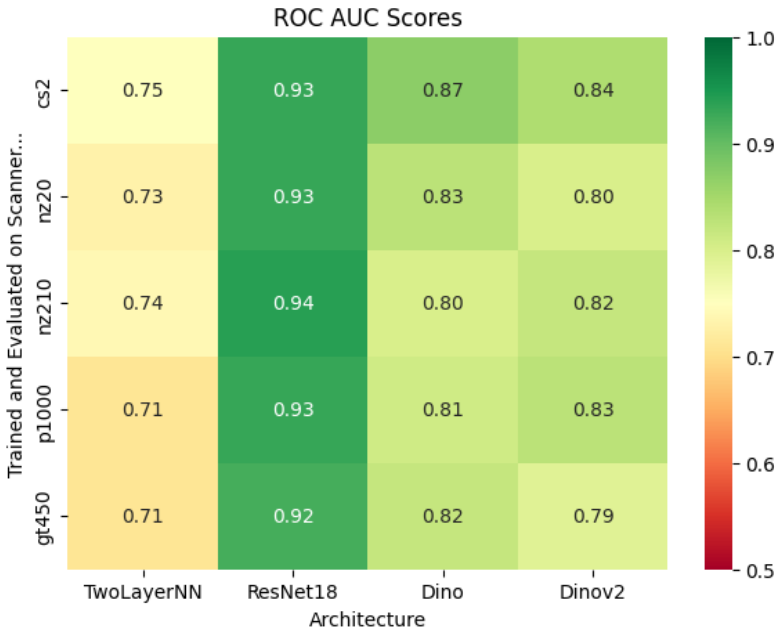

Figure 5: AUC-ROC of models evaluated on their source domain test data

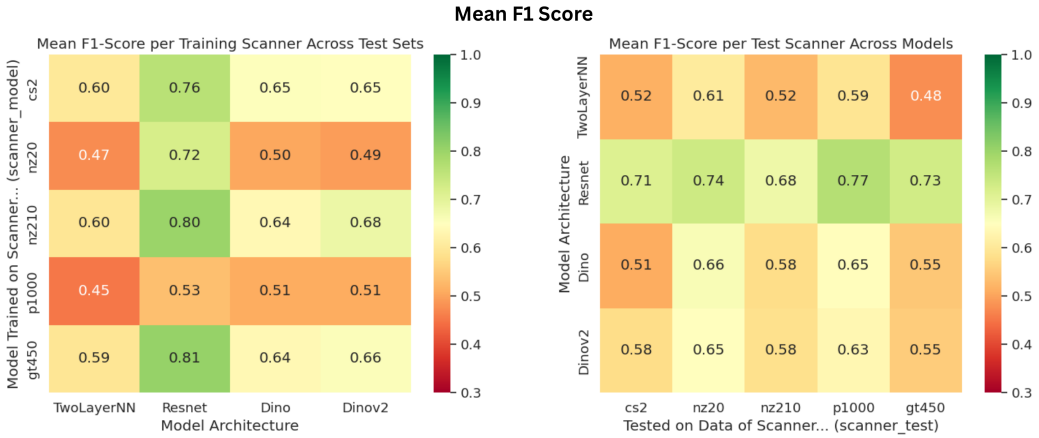

Figure 6: Mean F1-score heatmap matrices: source-centric (left) and target-centric (right).

models. The TwoLayerNN trained on the raw image data consistently records the lowest mean F1 scores, with values ranging from **0.45** to **0.60** across the different `scanner_models`. Furthermore, on the source-domain level, the mean F1 scores also show a clear pattern of inferior performance of the `nz20` and `p1000` source domain models when evaluated on all test datasets.

The right heatmap in Figure 6, on the other hand, presents the target-centric mean F1 scores per `scanner_test` set within each model architecture. The `nz20_test` and `p1000_test` sets consistently achieve higher scores than the remaining three test sets when evaluated on all models. Additionally, for the latent-space models DINO and DINOv2, each `scanner_test` set yields comparable performance scores ranging from

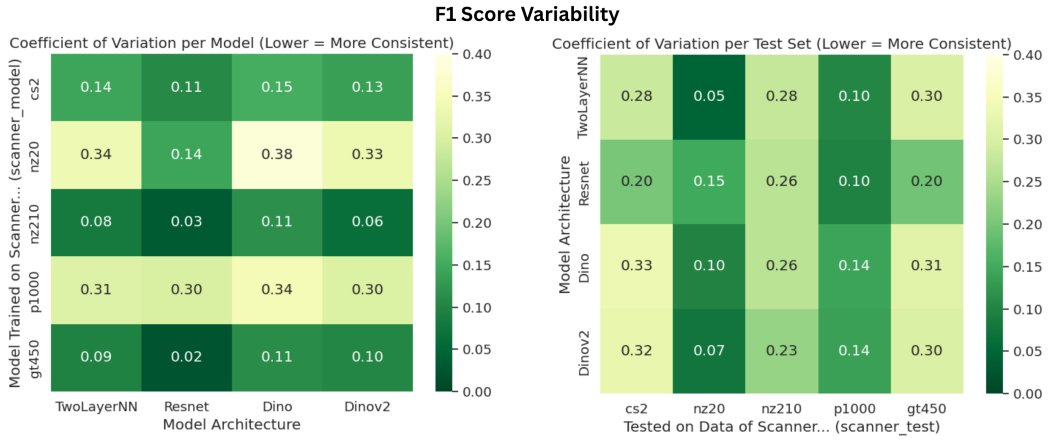

Figure 7: Coefficient of Variation heatmap matrices of F1 scores: source-centric (left) and target-centric (right).

0.51 to **0.66**. The input-space TwoLayerNN generally shows lower F1 scores than the other architectures, with values ranging from **0.48** to **0.61**. Meanwhile, with values ranging from **0.68** to **0.77**, ResNet18 consistently achieves the highest scores with low variability across all `scanner_test` sets. From a domain-level view, the target-centric results show higher F1 mean values for `cs2` and `p1000` test data than for the other domain data.

CV of F1 Scores. The dual approach of analysing the F1 scores from a source-centric and target-centric view is also used to measure **variability** in terms of the coefficient of variation across F1 scores per `scanner_model` and per `scanner_test` for each architecture in the left and right heatmap in Figure 7, respectively. For example, cell $\mathbf{A}_{1,1}$ in the left heatmap matrix $\mathbf{A}$ of the figure represents the coefficient of variation of the F1 scores of all five scanners' test sets on the TwoLayerNN model trained on scanner `cs2` training data (`cs2_model`). On the other hand, cell $\mathbf{B}_{1,1}$ in the right heatmap matrix $\mathbf{B}$ represents the coefficient of variation of the F1 scores of all five TwoLayerNN models on the `cs2` test set (`cs2_test`). The former captures the extent to which the F1 scores of a given `scanner_model` vary across different `scanner_test` sets, while the latter shows how much the F1 scores of a specific `scanner_test` vary across different `scanner_model`s. This analysis is valuable because the preceding figures, which present mean F1 scores, mask this variability by averaging the values.

In the source-centric view, illustrated by the left heatmap matrix in Figure 7, models trained on `nz20` and `p1000` exhibit substantially higher variance in F1 scores compared to models trained on the other three scanners. Although ResNet18-based models consistently show slightly lower variance than same-scanner models built with other architectures, overall variance tends to be higher across `scanner_model`s than across architectures.

In contrast, the target-centric view, represented by the right heatmap matrix of the same figure, reveals lower variance in F1 scores for test sets `nz20_test` and `p1000_test` when evaluated across different `scanner_model` configurations within each architecture. Notably, in this target-centric view, ResNet18 models do not consistently yield the lowest variance scores.

Pearson Correlation of F1 Scores The Pearson correlation coefficients between the F1 scores of a scanner's classifier across different model architectures exhibit significant variability. Additionally, the values appear to be influenced by the respective scanner's training data used to train each model, further contributing to the observed differences. Although the observed patterns are described in detail below, the heatmap visualisations for all five domains can be found in Appendix Figure 25 due to space constraints.

- **cs2:** Models trained on scanner `cs2` exhibit high pairwise Pearson correlation coefficients for F1 scores—mostly close to 1—except for the TwoLayerNN, which shows substantially lower correlations to the other architectures, with values around 0.3.
- **nz20 and p1000:** For scanners `nz20` and `p1000`, the F1 scores of the TwoLayerNN, DINO, and DINOv2 models are highly correlated (near 1), while correlations involving the ResNet18 architecture are notably lower.
- **nz210:** Models trained on scanner `nz210` show no consistent correlation pattern. The highest observed correlation is 0.84 between TwoLayerNN and ResNet18, while the lowest correlations involve DINOv2 with values of 0.53 when paired with ResNet18 and 0.5 when paired with DINO.
- **gt450:** Lastly, models from scanner `gt450` demonstrate generally high F1 score correlations across architectures, except for DINO, whose scores are less aligned with those of the other models.

6.2 BBSD Results

Recall that for each classifier, softmax values for all five test sets were computed from the classifier's raw logit outputs. In each pairwise comparison, the test set corresponding to the scanner used to train the classifier was designated as the source domain reference (test set A), while the other (test set B) was selected from any of the five available scanner test sets—including the reference domain set itself as well as the four target domain sets. Each pairwise KS test yields two distance values and two p-values, corresponding to the two output classes in the binary classification task. The p-values are Bonferroni-corrected and the lowest of the two along with its corresponding KS distance value is selected for the final results. Further analysis revealed that the KS distances for the two classes are very similar (See Appendix 29).

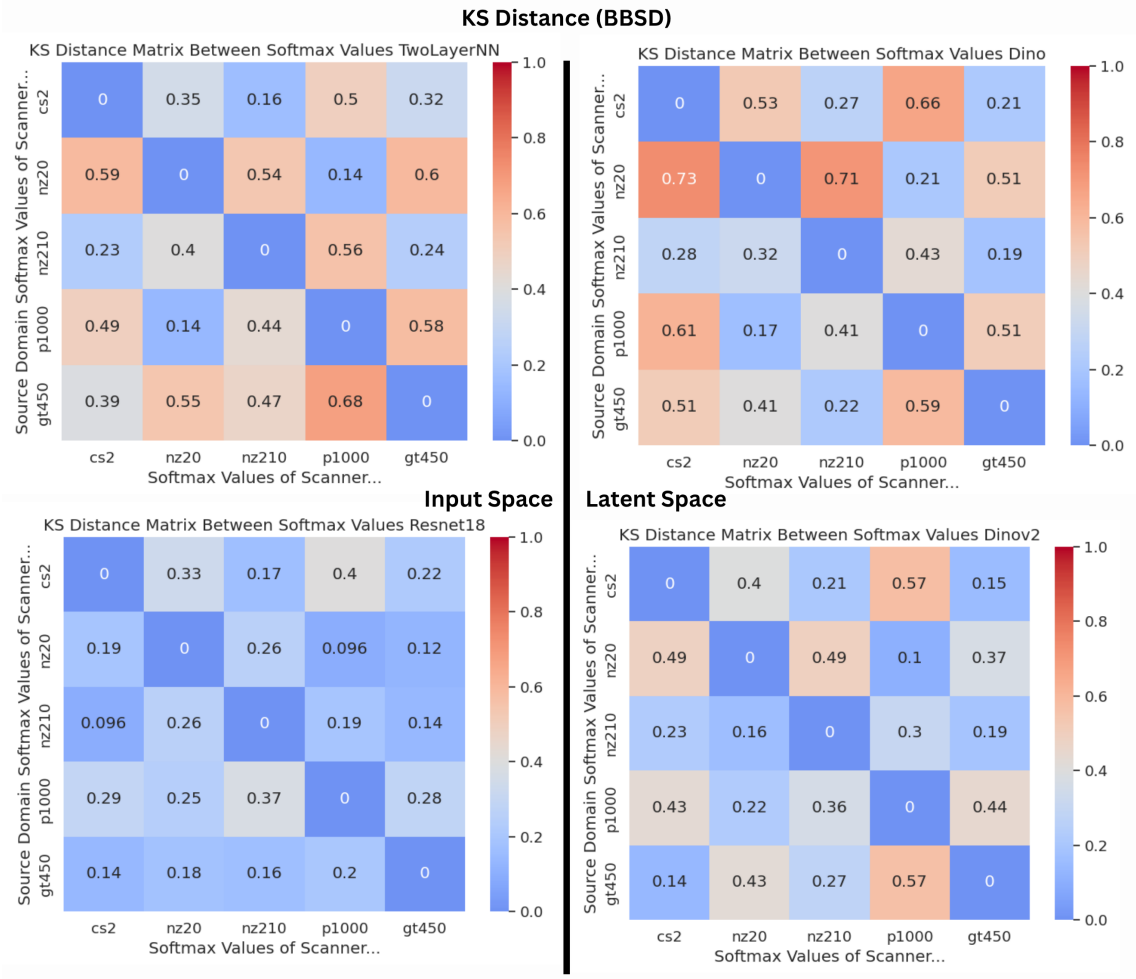

Figure 8: Overview of classifier KS-test-distance heatmap matrices of input-space (left) and latent-space (right) classifiers.

BBSD Values. The KS distances reported by the BBSD method are shown in Table 8. To boost comprehensibility, Figure 8 additionally presents the KS distances as side-by-side heatmap matrices and gives a global overview of the BBSD results, highlighting patterns across architectures.

When examining the KS distance scores across architectures side by side, a distinct pattern emerges: using `nz20` as the reference set results in elevated scores for all target sets except `p1000`. The inverse also holds—when `p1000` is used as the reference, the KS distances for `nz20` are mostly lower compared to those for the other target domains. Domains `cs2` and `nz210` seem to form a similar pair, with low distance values across all architectures.

Another interesting observation can be made when switching reference and target domains — e.g. in the result matrix of the TwoLayerNN (top-left heatmap). While `nz20` yields high KS distance values for both `nz210` and `cs2`, the inverse pairing exhibits noticeably lower values.

Overall, models based on ResNet18 and those trained on DINOv2 features yield lower KS distance scores compared to models trained in input space using a TwoLayerNN and those trained on DINO features. As shown in Table 8, among all ResNet18-based models, none of the pairwise comparisons exceed a KS distance of 0.4. Across all architectures and model comparisons, a total of 18 KS distance values exceed 0.5. Of these, only two originate from models trained on DINOv2 features, indicating that this architecture is rarely associated with particularly high domain shifts.

Mean BBSD Values. This pattern is consistent with the source-centric mean KS distances shown in the left heatmap matrix in Figure 9, where models based on ResNet18 and DINOv2 exhibit consistently lower mean scores across source scanners than those based on the TwoLayerNN or DINO. The same trend holds for the target-centric mean distances shown in the right matrix, reinforcing the observed performance differences across architectures.

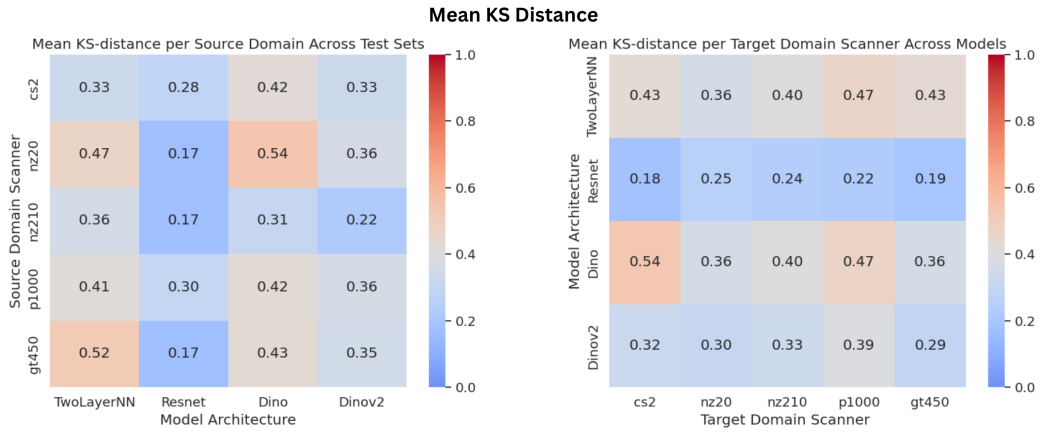

Figure 9: Mean KS-test-distance heatmap matrices: source-centric (left) and target-centric (right).

CV of BBSD Values. The coefficients of variation (CV) for the KS distance values reveal less distinct patterns compared to the mean values. In the source-centric view, presented on the left in Figure 10, DINOv2 stands out with a CV of 0.51 for the `cs2` source domain—this is the highest overall value across all scanners and architectures. DINOv2 also yields the highest CV values for the `nz20` and `gt450` source domains, with scores of 0.43 and 0.46, respectively. In contrast, the CVs for `nz210` and `p1000` as source domains are both 0.24, indicating substantially lower variation.

DINO shows generally moderate variability, with CV scores ranging between 0.28 and 0.44. While these values do not reflect particularly low variation, they are more consistent than those observed with DINOv2.

Regarding the input-space architectures, the CV for the `gt450` source domain of the TwoLayerNN is 0.21, which is notably lower than the other scores for this

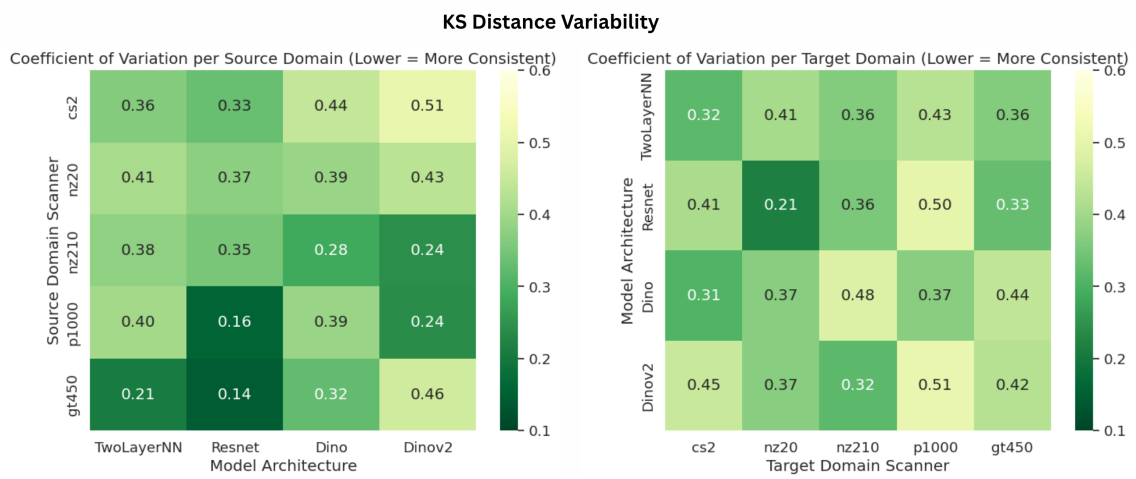

Figure 10: Coefficient of Variation heatmap matrices of BBSD KS-tests: source-centric (left) and target-centric (right).

architecture. For ResNet18, the `p1000` and `gt450` source domains yield the two lowest CVs overall, with values of 0.16 and 0.14, respectively.

When examining the results row-wise—i.e., from the perspective of each source domain—some interesting differences emerge. For `p1000`, the CVs for TwoLayerNN and DINO are similar, as are those for ResNet18 and DINOv2, though the differences between the two pairs are substantial. In the case of the `gt450` source domain, the contrast between input-space and latent-space architectures is even more pronounced, with input-space models showing considerably lower variability.

Switching to the target-centric perspective, illustrated by the right heatmap matrix in Figure 10, few values stand out, and no consistent or pronounced pattern can be observed across architectures. The `p1000` target domain is notable for producing the two highest overall CV scores—0.51 and 0.50—indicating substantial variation in KS distance values across source domains for this target domain. On the opposite end, the `nz20` target domain yields the lowest CV score of 0.21, which is associated with the ResNet18 architecture.

In comparison to the source-centric view, the target-centric results are characterised by a more uniform distribution of CV values across architectures and scanner domains. The relatively narrow range of CV values in the target-centric view makes it more difficult to isolate specific trends or attribute variation to particular architectures or domains.

Pearson Correlation of BBSD Values. Analogous to the approach used in the classifier performance evaluation, Pearson correlation heatmaps are generated to analyse the consistency of KS distance patterns across different architectures. The results reveal substantial variation in correlation depending on the source domain scanner and the compared architectures. Due to space constraints, the correlation heatmaps can be found in the Appendix Figure 27.

- **cs2:** For the `cs2` scanner source domain, KS distances exhibit strong correlation across architectures. The value between DINO and DINOv2 is nearly perfect. ResNet18 also shows high correlation with all other models, with values around 0.95. TwoLayerNN shows slightly lower but still strong correlations with both DINO and DINOv2, each approximately 0.8.

- **nz20:** Similarly, for the `nz20` source domain, DINO and DINOv2 again display an almost perfect correlation. When paired with either TwoLayerNN or ResNet18, both DINO-based models maintain high correlation values ranging from 0.83 to 0.92. The only particularly lower correlation is observed between ResNet18 and TwoLayerNN, with a value of 0.58.

- **nz210:** Correlation results for the `nz210` scanner domain differ from the more consistent patterns observed in the other domains. The strongest correlation is found between the TwoLayerNN and DINO architectures, with a value of 0.90. The pairing of TwoLayerNN and ResNet18 yields a moderately high correlation of 0.67, while the correlation between TwoLayerNN and DINOv2 is lower at 0.58. The correlation between DINO and DINOv2 is 0.70, which also falls below the levels seen in most other domains. Notably, the ResNet18 and DINOv2 pairing stands out with a low negative correlation of –0.21. In total, the `nz210` domain exhibits more variability and less consistent inter-architecture agreement compared to the other scanner domains.

- **p1000:** For the `p1000` source domain, DINO and DINOv2 once again show a near-perfect correlation. The TwoLayerNN also aligns closely with DINOv2, yielding a similarly high score. The correlation between the TwoLayerNN and DINO is slightly lower at 0.91 but remains strong. In contrast, ResNet18 exhibits significantly weaker correlations with all other architectures, with values ranging only between 0.30 and 0.41.

- **gt450:** In the case of the `gt450` source domain, DINO exhibits an unexpectedly low correlation with all other architectures, with values ranging from 0.24 to 0.42. This stands in contrast to the other source domains, where DINO generally aligned more closely with the remaining models—making this divergence a notable observation in the overall results. Meanwhile, the remaining three architectures—TwoLayerNN, ResNet18, and DINOv2—correlate very strongly with one another, indicating a high degree of alignment in their KS distance values.

Summary: Overall, DINO and DINOv2 KS distance results correlate almost perfectly for scanner domains `cs2`, `nz20`, and `p1000`. For `gt450`, this near-perfect correlation happens in input space, between the TwoLayerNN and the Resnet18, while for `nz210` no such high correlation values are achieved. Furthermore, ResNet18 exhibits relatively lower correlation to other architectures of domains `nz20`, `nz210`, and `p1000`. Lastly, the low correlations between distance values from DINO and other architectures represent an unexpected divergence from the rest of the results.

P-Values of BBSD. The Bonferroni-corrected p-values take the value of 2 along the diagonal of each architecture's heatmap matrix, reflecting comparisons of a domain with itself. Aside from these, all corrected p-values fall far below the significance threshold of 0.05. The highest non-diagonal p-value—1.6×10^{-10}—is observed for the KS test between `nz20` as the reference domain and `p1000` as the target domain, using the ResNet18 architecture.

In summary, despite some KS distances being relatively small, all observed values are reported as statistically significant. An overview of the p-values can be found in Appendix Figure 28.

6.3 MMD Results

The Maximum Mean Discrepancy (MMD) is applied to the latent-space representations extracted from an off-the-shelf foundation model. It yields a distance value and a p-value for each pair of scanner domains, quantifying distributional differences in the corresponding feature spaces. Since MMD operates symmetrically on feature distributions, there is no designated source or target domain—each pair is treated with equal status. As a result, the corresponding heatmap matrices are symmetric and thus only evaluated along one axis. The two matrices are presented in Figure 11 for the two feature extractors used in this experiment: DINO and DINOv2.

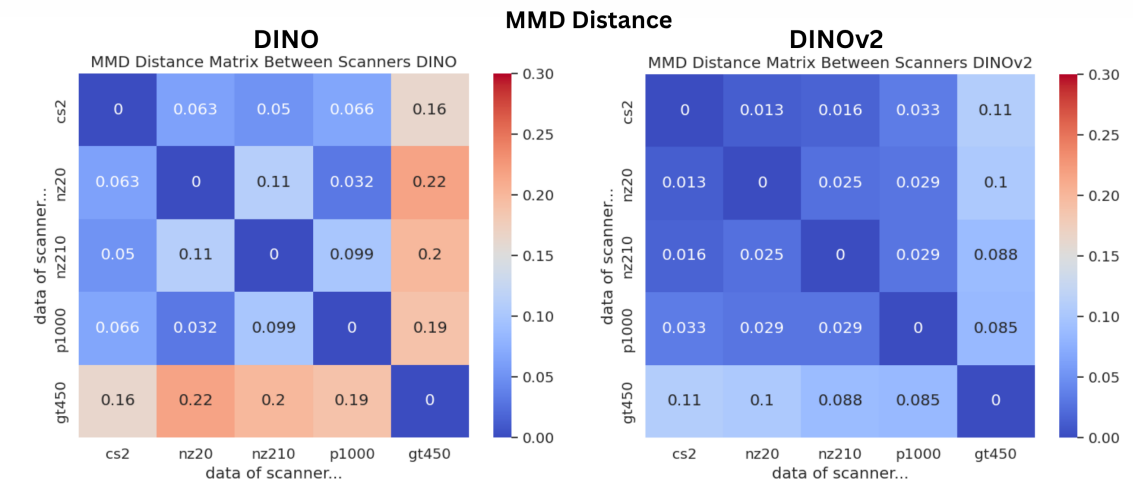

Figure 11: MMD-distance heatmap matrices for DINO and DINOv2.

MMD Values. As expected, the MMD between a domain and itself is always zero, as reflected by the main diagonal of the heatmaps. Beyond the diagonal, a notable difference in value ranges can be observed between the two architectures. DINOv2 yields comparatively low MMD values, ranging from 0.013 to 0.11, while DINO produces substantially higher distances, spanning from 0.032 up to 0.22. Particularly striking are the pairings involving the `gt450` scanner domain, which consistently result in the highest MMD values across both architectures—although

with different magnitudes. A pattern that is present in the DINO results but not as pronounced in DINOv2 is the slight elevation in MMD values for the scanner pairings (nz210, nz20) and (nz210, p1000) compared to the surrounding values.

Mean MMD Values. The mean MMD values, as shown on the left in Figure 12, support the initial observation of differing magnitudes between the two architectures: on average, DINO values are approximately twice as large as the corresponding DINOv2 values for each scanner domain. Additionally, for both architectures, the mean MMD values associated with the gt450 domain are nearly twice as high as those of the other scanner domains within the same architecture.

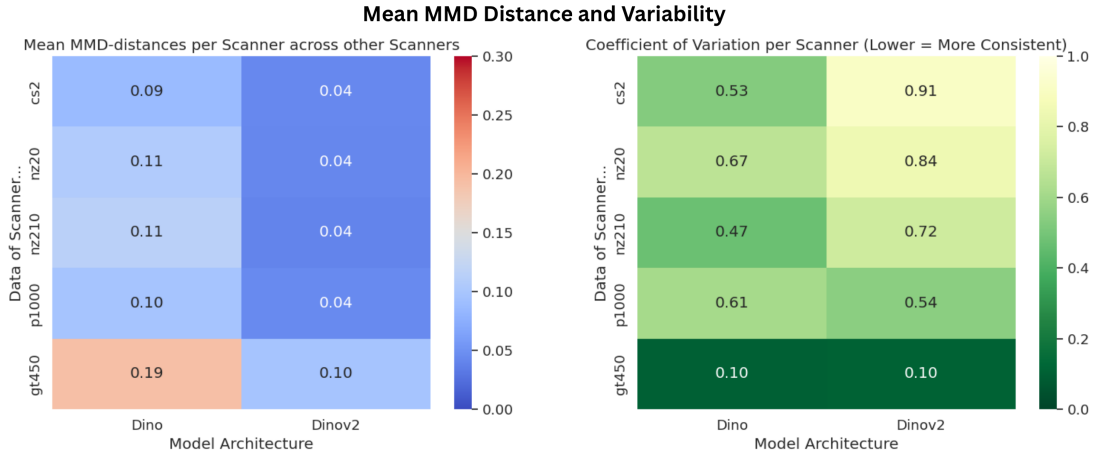

Figure 12: Mean MMD-distance (left) and mean CV-value (right) heatmap matrix.

CV of MMD Values. Regarding the coefficients of variation per domain—also presented in Figure 12—, the results are mixed and show no consistent pattern. For four of the five domains—cs2, nz20, nz210, and p1000—variation is generally high across both architectures, with values ranging from 0.47 to 0.91. The lowest of these is 0.47 for nz210 on DINO, while the highest is 0.91 for cs2 on DINOv2. In contrast, the domain gt450 stands out, where both DINO and DINOv2 yield a particularly low variation of just 0.10.

Pearson Correlation of MMD Values. In addition, Pearson correlation scores per scanner domain show consistently high values for four out of the five domains (see Appendix Figure 30). For nz20 and p1000, the correlations are strong, with values of 0.90 and 0.91, respectively. Even higher correlations are observed for cs2 and nz210, with values of 0.98 and 0.96, indicating very strong agreement between architectures for these domains. In contrast to this, the correlation for gt450 is notably different between the two models—showing a clear negative value of -0.29.

P-Values of MMD. Importantly, all p-values obtained from the permutation test with 1000 iterations yield the minimum possible value of 0.001, indicating the statistical significance of all observed MMD distance values. Since there are no deviations from this minimum value across any of the domain pairs or architectures, no further analysis of the p-values was conducted. For the sake of completeness, an overview of the p-values can be found in Appendix Figure 31.

6.4 multi-class PAD* Results

The fine-tuned ResNet18 that was used to extract task-specific features for the subsequent training of the domain discriminator achieves an F1 score of 0.88 in the last epoch on the held-out validation set, indicating adequate classification ability.

For the domain discriminator, Figure 13 displays the false negative rate (FNR) for each domain, grouped by discriminator model. Note that while all discriminators share the same TwoLayerNN architecture, their names reflect the backbone model used to extract the feature vectors on which they were trained.

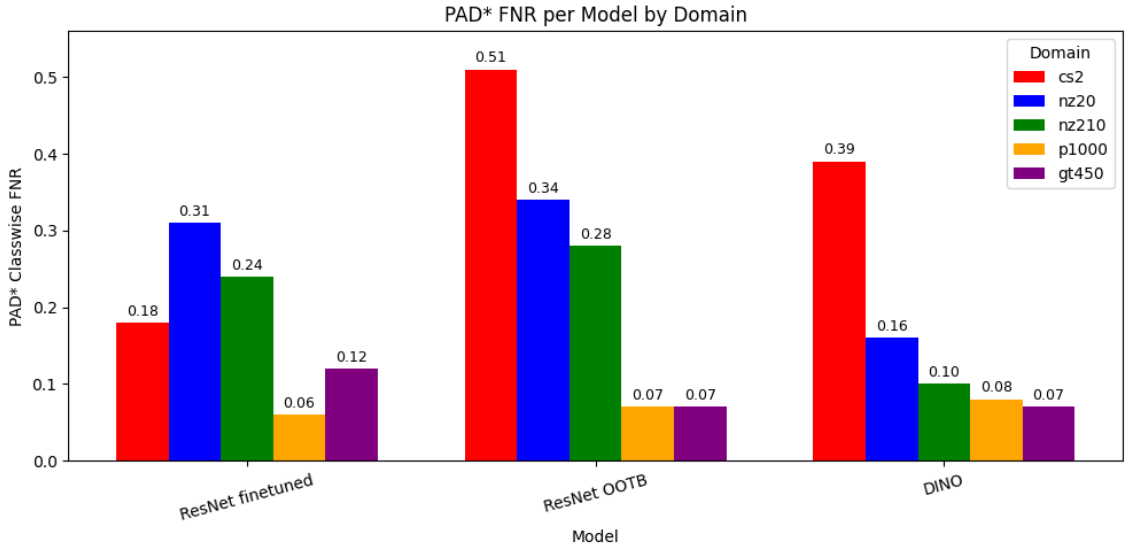

Figure 13: PAD*$_{\text{multi}}$ FNR per discriminator model by domain. Higher means less distinguishable (lower relative shift).

It illustrates how the models rank the domains in terms of distinguishability, where higher FNR implies that a given domain is less distinguishable from the other domains and thus exhibits lower relative shift. Several observations can be made: The off-the-shelf ResNet18- and DINO-based models yield the highest value for domain `cs2` and then rank domains `nz20`, `nz210`, `p1000`, and `gt450` descendingly. In contrast, the task-specific ResNet18 TwoLayerNN ranks `cs2` only in third place with a relatively low FNR. Additionally, the model shows a slightly elevated FNR for the domain `gt450` in comparison to the other models.

Table 4: Class-wise (domain-wise) metrics for each domain discriminator model.

Model	Domain	FNR	FDR	F1 Score
$\text{ResNet18}_{\text{fine-tuned}}$	cs2	0.1806	0.1729	0.8233
	nz20	0.3067	0.2187	0.7347
	nz210	0.2373	0.2152	0.7736
	p1000	0.0621	0.1556	0.8887
	gt450	0.1227	0.1559	0.8604
$\text{ResNet18}_{\text{pre-trained}}$	cs2	0.5089	0.1352	0.6265
	nz20	0.3399	0.2469	0.7035
	nz210	0.2840	0.3348	0.6897
	p1000	0.0691	0.1853	0.8689
	gt450	0.0714	0.3051	0.7949
$\text{DINO}_{\text{pre-trained}}$	cs2	0.3924	0.0848	0.7304
	nz20	0.1620	0.0368	0.8962
	nz210	0.1003	0.2374	0.8255
	p1000	0.0772	0.0768	0.9230
	gt450	0.0721	0.2789	0.8115

Complementarily, Figure 14 reorganises the same FNR values by grouping them by domain. This allows the performance of the different discriminator models on each domain to be compared directly.

It can be seen that for three domains—namely `cs2`, `nz20`, and `nz210`—the off-the-shelf ResNet18-feature based model shows the highest FNR. Domains `nz20` and `nz210` show similar FNRs for both the task-specific fine-tuned ResNet18 as well as the off-the-shelf ResNet18. Notably, the `p1000` domain consistently achieves very low FNR between 0.6 and 0.8 across all models, as does `gt450` with the exception of a slightly elevated FNR for the fine-tuned ResNet18.

To enable a deeper analysis of model behaviour, Table 4 shows the class-wise FNR, FDR and F1 scores for each domain and for each discriminator, denoted by the backbone feature-extractor architecture.

Finally, the metrics in Table 5 summarize the overall performance and cross-domain variability of the domain discriminator models. Notably, the task-specific ResNet18 outperforms its off-the-shelf counterpart across all reported metrics: it achieves higher F1 and $\text{PAD*}_{\text{multi}}$ scores, along with a lower mean absolute error (MAE) and lower FNR standard deviation (FNR STD). While the DINO-based model slightly

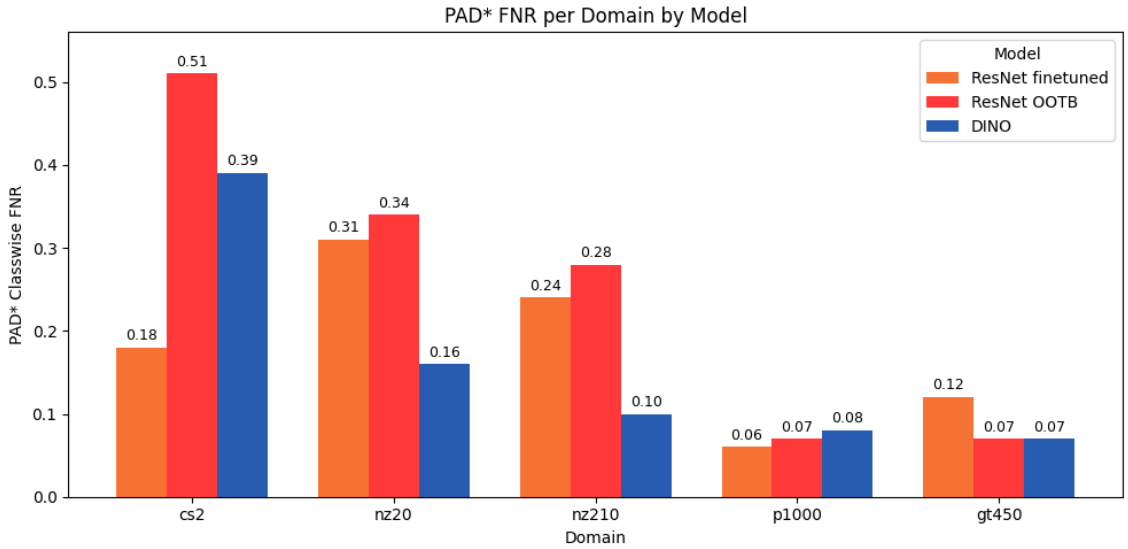

Figure 14: PAD*$_{\text{multi}}$ FNR per domain by discriminator model. Higher means less distinguishable (lower relative shift)

Table 5: Aggregate performance metrics for each domain discriminator model, denoted by the feature extraction backbone. F1 Score represents the mean F1 score across all classes. MAE denotes the mean absolute error. FNR STD is the standard deviation across the domain-wise FNR values. PAD*$_{\text{multi}}$ is the aggregate PAD* metric score.

Model	**F1 Score**	**MAE**	**FNR STD**	**PAD*$_{\text{multi}}$**
ResNet18$_{\text{fine-tuned}}$	0.8161	0.1819	0.098	0.7727
ResNet18$_{\text{pre-trained}}$	0.7367	0.2546	0.188	0.6817
DINO$_{\text{pre-trained}}$	0.8373	0.1608	0.133	0.7990

surpasses even the task-specific ResNet in F1 and PAD*$_{\text{multi}}$ and MAE, its elevated FNR STD indicates greater performance variability across domains—caused by the `cs2` domain FNR visualised in Figure 13.

6.5 PCA Data Visualisation

For further analysis, PCA was applied to reduced representations from three different stages: (1) the hidden representations of the input-space TwoLayerNN (i.e., after the first linear + ReLU layer), (2) the feature vectors extracted from the pre-trained DINO and DINOv2 models, and (3) the hidden representations of the latent-space TwoLayerNNs trained on those features. The resulting 2D visualisations aim to reveal how the data is structured within each representational space and whether domain-specific patterns or separations become visible. To enable meaningful visual

comparison, all PCA plots use a consistent scale for the same representational level, allowing not just the shape but also the compactness of cluster formations to be directly contrasted across architectures within a level.

These visualisations help to qualitatively assess the separability of domains and the potential impact of shifts. For practical reasons, only PCA plots that exhibit interpretable clustering or visible structure are shown, as including all visualisations—especially those that do not reveal any clear patterns—would be redundant and would obscure rather than support the interpretation. Firstly, reduced represen-

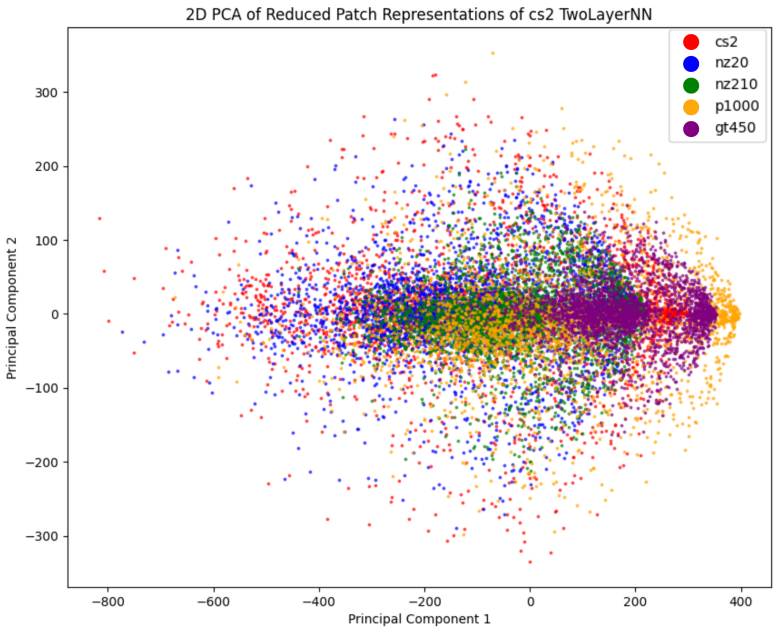

Figure 15: 2D PCA visualisation of reduced representations of the input-space TwoLayerNN trained on cs2 data.

tations from the hidden layer of the TwoLayerNN were reduced to 2D using PCA. Figure 15 shows the resulting distribution for the `cs2` domain as a representative example. The corresponding visualisations for the other four domains are omitted, as they exhibit a very similar overall structure. On the right side of the figure, the domains `gt450`, `cs2`, and `p1000` form relatively distinct clusters, standing out from the remaining data points, which are more loosely distributed and tend to overlap.

For the latent space, PCA was applied to both, directly to the hidden representations output by the DINO and DINOv2 models, as well as to the hidden representations of the downstream task classifiers that were trained using these features. As shown in Figure 16, the `gt450` domain forms a visibly tighter cluster than the other domains for both DINO and DINOv2. In the DINOv2 representation, the domain clusters also show more overlap overall, but `gt450` remains the most compact.

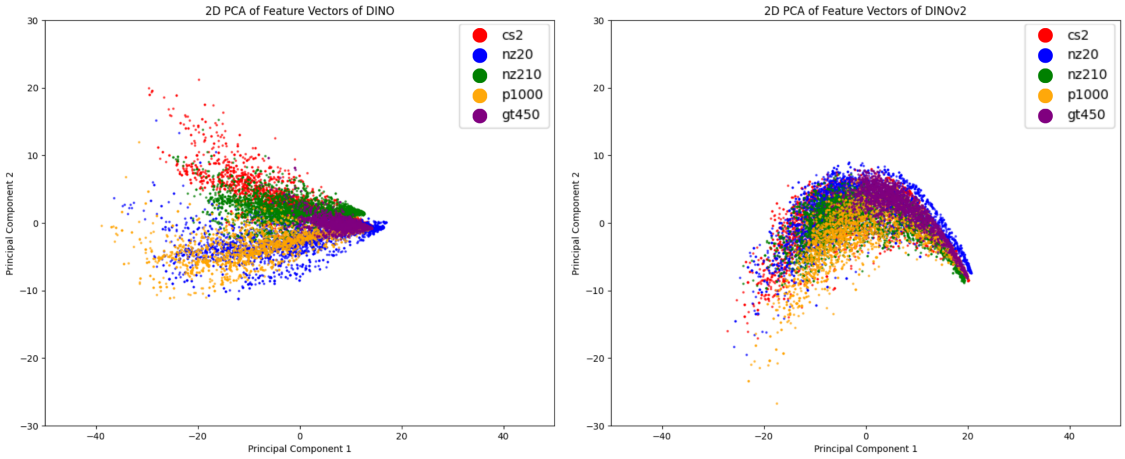

Figure 16: 2D PCA visualisations of feature vectors extracted using DINO (left) and DINOv2 (right).

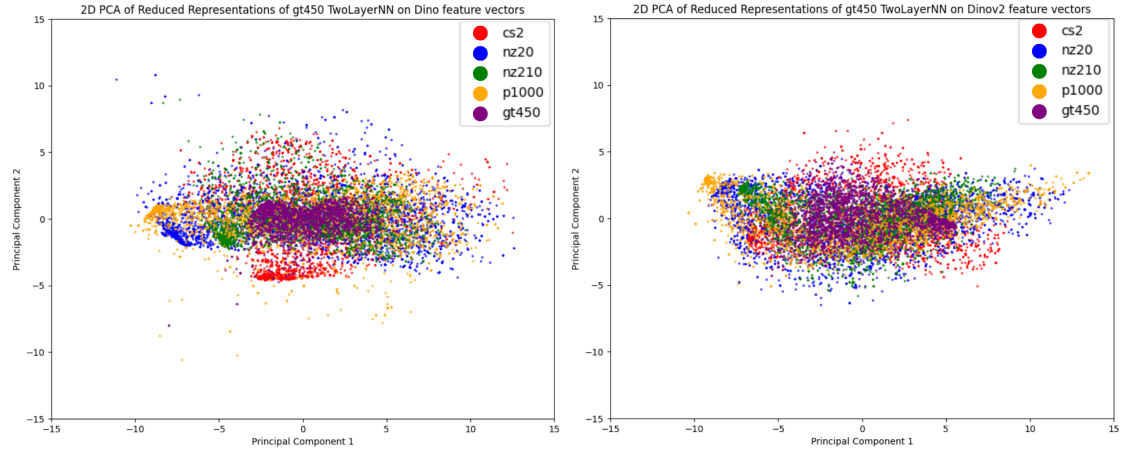

Figure 17: 2D PCA visualisations of reduced representations of the latent-space TwoLayerNNs trained on DINO (left) and DINOv2 (right) features extracted from gt450 data.

Finally, the hidden layer representations of the latent classifiers trained on DINO and DINOv2 features were reduced using PCA to examine their internal structure. As shown in Figure 17, for DINO-based classifiers, some instances from `cs2`, `nz20`, and `p1000` appear as distinct clusters separated from a central overlapping region. In contrast, such domain-specific clustering is less apparent in the DINOv2-based representations.

6.6 Intra-Architectural Metric Alignment

As a visual aid, four figures are provided to enable a side-by-side comparison of the domain rankings of the applied methods for each architecture.

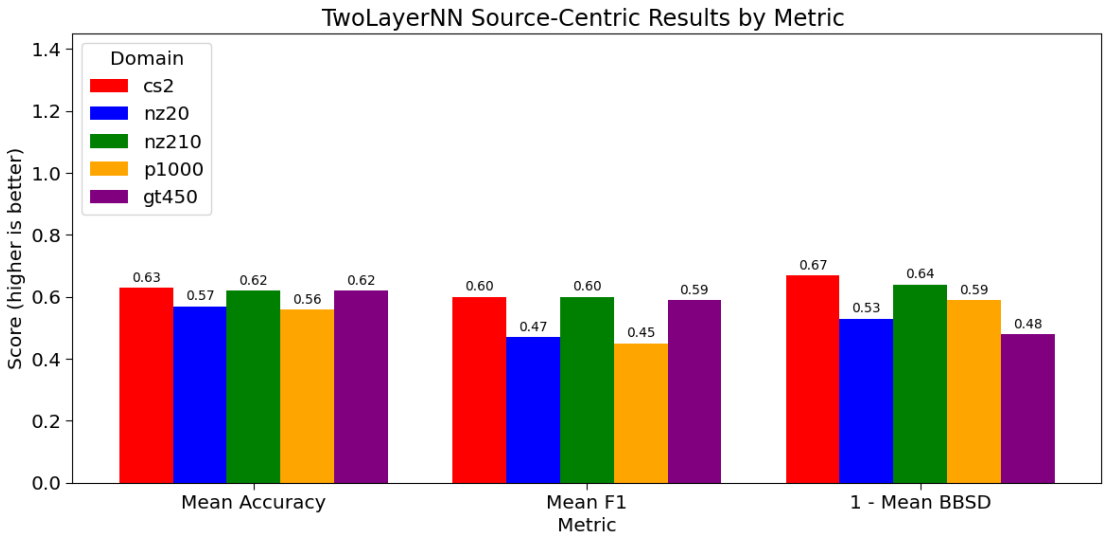

Figure 18: TwoLayerNN (input space) source-centric aggregate results for each domain, grouped by metric.

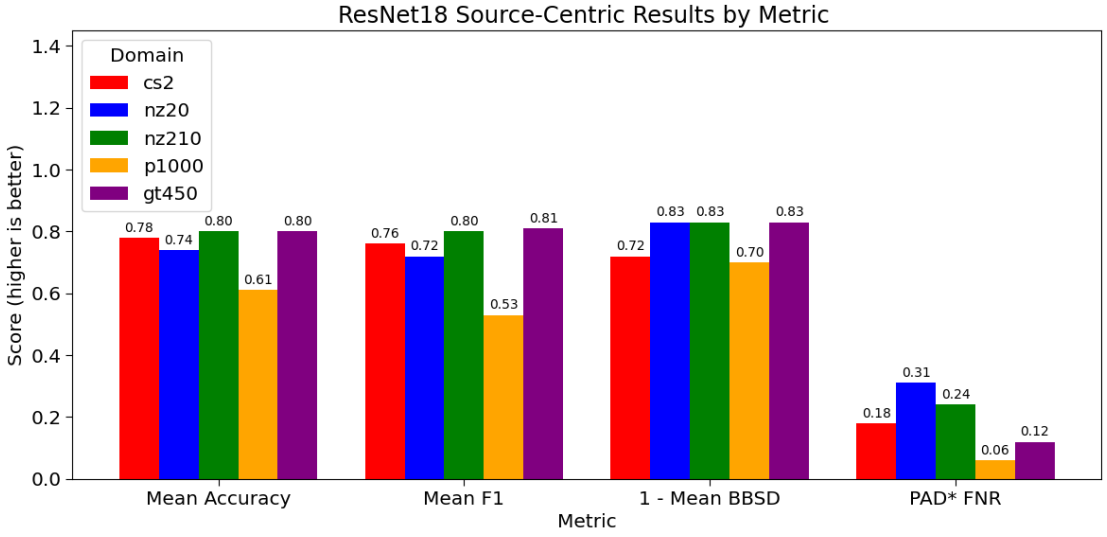

Figure 19: ResNet18 (input space) source-centric aggregate results for each domain, grouped by metric.

A separate plot for each architecture compares the scores of all domains across the evaluated metrics (See Figures TwoLayerNN: 18, ResNet18: 19, DINO: 20, DINOv2: 21. Each plot includes only the metrics available for the respective architecture.

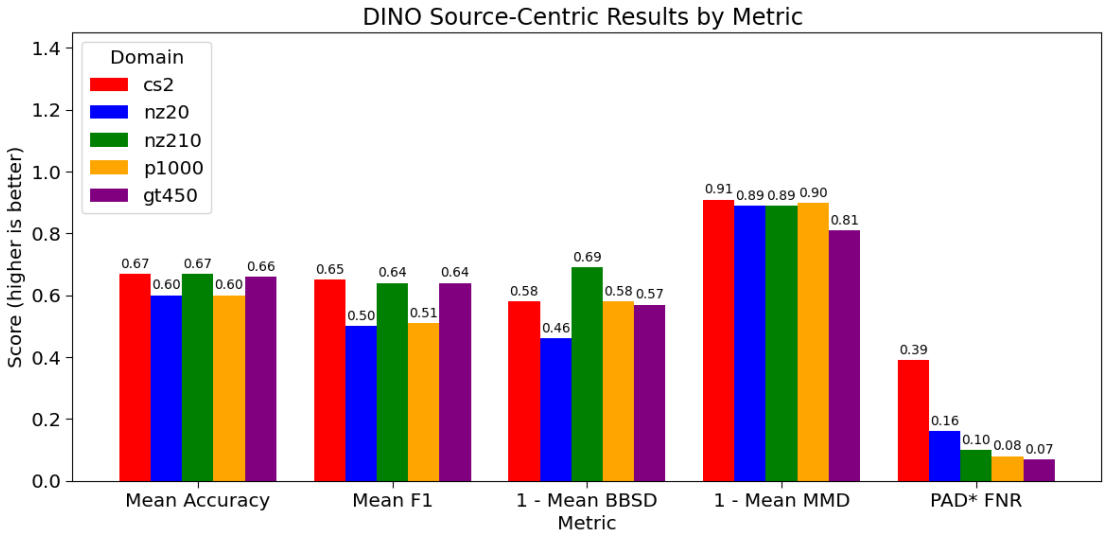

Figure 20: DINO (latent space) source-centric aggregate results for each domain, grouped by metric.

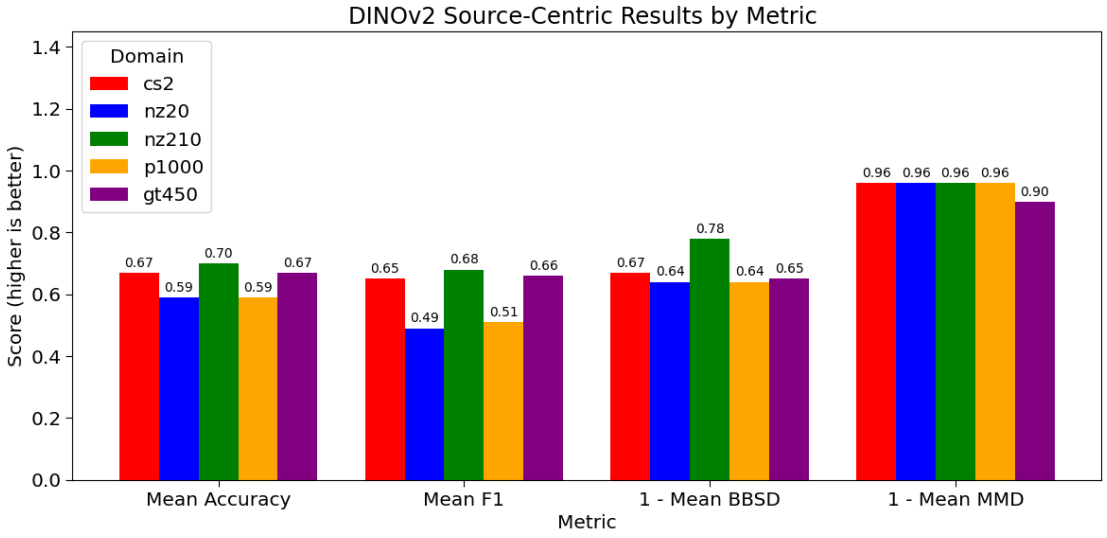

Figure 21: DINOv2 (latent space) source-centric aggregate results for each domain, grouped by metric.

While the absolute scores are not directly comparable across metrics, these visualisations help identify the relative ranking of domains and assess whether consistent patterns emerge across metrics within the same architecture.

This analysis is exploratory and should be interpreted with caution, as the metrics differ in meaning and the role of the architecture varies between them. For example, in the case of DINO and DINOv2, the $1 - MeanBBSD$ values reflect domain similarity based on KS distances derived from classifier outputs, whereas MMD scores are computed directly on the feature representations without using a

classifier. Similarly, for the ResNet18 PAD* FNR, the architecture refers to the task-specific feature extractor that determines the result, rather than the TwoLayerNN domain discriminator used in the PAD* computation.

7 Discussion

This section analyses the most prominent patterns observed in the results, aiming to interpret their significance and implications. It begins by examining the behaviour of the underlying classifiers, outlining their limitations and potential factors influencing their performance. Next, a domain-level analysis explores how the different shift assessment methods align or diverge in detecting scanner-induced shifts. Based on these findings, broader, dataset-independent insights and hypotheses are proposed. The discussion then addresses dataset-specific limitations and challenges that may have influenced the outcomes. Finally, the section closes with an outlook on future directions to overcome current limitations and advance the development of more robust and interpretable methods for assessing scanner-induced domain shifts.

7.1 Model Behaviour

The following model behaviour subsection is divided into two parts: (1) a discussion of the performance behaviour of input-space classifiers, (2) an analysis of latent-space classifiers using features from foundation models. They offer insight into how the classifiers and feature extractors may have influenced the results, and what this implies for interpreting the observed domain shifts.

7.1.1 Classifier Behaviour in Input Space

When analysing the performance results of the original tumor task classifiers, trained to serve as softmax output extractors for BBSD, the focus does not lie on discovering the best model for tumor classification. Rather, the performance metrics were mainly employed to observe the behaviour of the classifiers on various scanner domains before applying BBSD. Together with other metrics, they can boost interpretability by highlighting domain- or architecture-related patterns.

First and foremost, it is not surprising that the advanced ResNet18 architecture model achieves much higher performance scores than the TwoLayerNN classifier across all performance metrics. ResNet18 is architecturally better suited for processing raw $224 \times 224 \times 3$ image data than a simple two-layer linear network because it leverages convolutional layers to exploit spatial hierarchies and local patterns inherent in image structures. In contrast, the TwoLayerNN flattens the image into a 1D vector, thereby losing spatial relationships and treating every pixel independently. This limits its ability to learn meaningful visual features, especially in complex image domains like histopathology. Additionally, ResNet18 benefits from residual

connections, more layers, and more parameters, enabling it to model more intricate patterns and generalise more effectively across domains.

However, in the context of BBSD-based shift detection, the TwoLayerNN was intentionally used to avoid exactly these strong generalisation capabilities. Since BBSD relies on comparing softmax output distributions between in-domain and out-of-domain samples, a highly robust classifier may produce overly uniform or over-calibrated outputs across domains—potentially masking the domain shifts that are being measured. A simpler, less powerful classifier was deemed more likely to reflect distributional changes in the input data. To ensure these TwoLayerNN classifiers are still functionally adequate, the invertibility of their confusion matrices was verified on their respective source domain test sets. This confirms that the models are performing better than random guessing and are capable of distinguishing between classes at a minimal level necessary for BBSD application. This was only the case on the source domain though, as the confusion matrices of models trained on problematic domains yielded non-invertible confusion matrices, often completely overpredicting one class as seen in Appendix Table 7.

Another indication that the TwoLayerNN's decision boundary may be overly simplistic and overconfident appears in the form of the linear segments observed in some of the ROC curves (See Appendix Figure 26). These segments suggest that the model assigns extreme softmax probabilities—close to 0 or 1—for most samples, such that varying the decision threshold has little effect on classification outcomes over wide intervals. This behaviour was verified by inspecting the distribution of softmax outputs, which showed a strong skew towards the extremes. This indicates a lack of meaningful uncertainty. As such, these patterns in the ROC curves provide additional insight into how the TwoLayerNN tends to oversimplify the complex task of tumor classification from raw input images, failing to account for fine-grained distinctions, which leads to failure on more challenging or shifted data.

It should be noted that during the experiments, an alternative architecture, denoted as ThreeLayerNN, was also evaluated. This variant introduced an additional fully connected layer with a ReLU activation, aiming to support the compression from the high-dimensional $224 \times 224 \times 3$ input space by introducing an intermediate representation. The hypothesis was that a more gradual dimensionality reduction might help the network capture more informative features early on. However, this modification did not lead to a measurable improvement in performance, suggesting that the added capacity or depth did not meaningfully enhance feature extraction in this setting.

7.1.2 Foundation Model Features and Latent-Space Behaviour

With regard to the DINO- and DINOv2-feature trained latent-space classifiers, it is not overtly surprising that the ResNet18 outperforms them, as the architecture of the classifier itself is really a simple TwoLayerNN that uses the task-agnostic features extracted by pre-trained but not fine-tuned DINO and DINOv2 models.

It essentially functions like a classifier head, mapping the latent features to final predictions.

It is noteworthy, however, that these latent-space TwoLayerNNs consistently outperform their input-space counterparts for each source domain and on average also on target domains, although the latent-space models have to rely on task-agnostic features from the FMs. Multiple factors could be at play here: as mentioned before, the TwoLayerNN is likely too simple to effectively process the large image data. Even though the features extracted by the foundation models are task-agnostic, the provided dimensionality reduction presumably puts the latent-feature trained models at an advantage.

Another reason for improved performance could lie in the ability of foundation models to extract robust general image features, filtering out most of the noise, while the input-space TwoLayerNN may be more sensitive to noisy image data. However, if this alone characterised the main difference between the input-space and latent-space models, one would expect a more stable performance of the feature-trained classifiers across target domain test sets, as partially observed for the ResNet18 model. This is not really the case for the DINO- and DINOv2-based classifiers, which show similar sensitivity to out-of-domain data as the input-space TwoLayerNN. The same can be observed for BBSD where the latent-space models detect similar shifts as the task-specific TwoLayerNN, showing that those shifts persist in the extracted features. While it is undoubted that foundation models excel at generalising and extracting robust features, their lack of success in this setting could indicate a limitation. Specifically, it may suggest that the features they extract are not sufficiently task-relevant and are affected by domain shift, meaning the shift propagates into the latent space and negatively impacts downstream performance. A possible explanation for this is that these FMs were pre-trained on natural image datasets and therefore may struggle to generalise on medical images, such as microscopic histopathology slides, which differ substantially in texture and structure of the task-relevant features. It is also possible that the limited performance is partly due to the simplicity of the classification head. The TwoLayerNN may lack the capacity to effectively utilize the foundation model features, especially under domain shift where more expressive decision boundaries could be beneficial.

Curiously, while DINOv2 achieves lower shift magnitudes in MMD compared to DINO, this is not necessarily reflected as improved downstream classification performance. One possible explanation for the similarity in performance is that for classification, only the global [CLS] token was used as the image representation. This choice aligns with standard practice in transformer-based models, where the [CLS] token is designed to capture global semantic context suitable for classification tasks. However, although both DINO and DINOv2 produce [CLS] and patch tokens, DINOv2 introduces architectural and training improvements that better preserve spatial structure and semantic detail across the patch tokens. These advantages are particularly beneficial in tasks involving fine-grained spatial features. By relying solely on the global [CLS] token—which aggregates image information into a single vector—the experimental setup may have underutilised DINOv2's improved

spatial representations, which are better preserved in its patch tokens. This design choice could partially explain why DINOv2 did not outperform DINO in downstream classification, despite architectural advantages. The observed difference in MMD magnitudes between the two architectures may be due to the tighter feature clustering exhibited by DINOv2, as reflected in the PCA analysis (see Figure 16).

These findings highlight an important limitation of generic vision foundation models: even when self-supervised, they may not provide optimal representations for specialised domains like medical imaging. Their ability to generalise across natural image tasks does not necessarily extend to complex biomedical data, where relevant features are often subtle and domain-specific. As such, relying solely on off-the-shelf foundation models may lead to suboptimal performance in medical applications. This underscores the need for either fine-tuning these models on domain-relevant data or developing foundation models trained directly on large-scale medical image datasets.

7.2 Empirical Patterns and Domain-Level Observations

Classifier generalisation and Source-Domain Sensitivity of BBSD. Classifiers trained on domains such as `nz20` and `p1000` exhibited poor generalisation to most other domains, as reflected in low F1 scores, and in some instances, even non-invertible confusion matrices when tested on the other three domains. In contrast, models trained on other domains achieve more robust performance on these two domains, indicating that the source domain influences the generalisation ability. Additionally, these two domains still performed relatively well on each other, suggesting a certain degree of mutual compatibility. For ResNet18, only the `p1000` model stood out as generalising especially poorly to other domains. BBSD (the KS test) reflects similar patterns: models trained on `nz20` or `p1000` exhibit low KS distances when evaluated on each other, but significantly higher distances when tested on `cs2`, `nz210`, or `gt450`. This indicates that `nz20` and `p1000` may share similar feature characteristics and thus form a distinct pair.

Notably, the BBSD matrices themselves lack strict symmetry,—distances from a given source domain to others are not always equal in the reverse direction, suggesting that not only performance but also shift magnitude is influenced by the choice of source domain. This asymmetry was present even for classifiers like the ResNet18, which otherwise exhibited strong generalisation ability. This observation is also reflected in the CV results, which showed more uniform variability for the target-centric view than for the source-centric view. This observation suggests that in this context the choice of source domain has a more pronounced influence on the variability of KS distances than the choice of target domain or rather that some domains provide features that generalise better to other domains.

Divergent Shift Patterns in MMD. MMD and BBSD do not always agree on which domain pairs exhibit the largest shifts. For example, the BBSD classifier

using DINO features reports high KS distances for cs2 paired with nz20 and p1000 in both directions, yet MMD assigns low distances to those same pairs. Conversely, MMD consistently flags gt450 as a strong outlier: both DINO and DINOv2 feature spaces exhibit significantly elevated MMD distances for gt450 compared to any other domain. In contrast, BBSD only occasionally reports high KS distances involving gt450—and primarily when gt450 serves as the reference domain. For instance, neither DINO- nor DINOv2-based BBSD assigns a high KS distance to the gt450–nz210 pair, despite MMD indicating a substantial shift between these two domains. The PCA visualisations in Figure 16 indicate that gt450 data points form a more compact cluster with less overlap to those of the other domains in both DINO and DINOv2, which could explain the overall pronounced shift observed in MMD for this scanner domain.

Finally, the strong coupling between nz20 and p1000 previously observed in both BBSD and classifier performance is not consistently reflected in the MMD results: under DINO, comparisons between cs2 and both nz20 and p1000 yield low MMD scores. Furthermore, under DINOv2, the shift between nz20 and p1000 is not ranked particularly low relative to other domain pairs, despite BBSD consistently indicating minimal shift between them.

Multi-class PAD*: Agreement and Architectural Influence. When interpreting the performance of domain discriminators, it is important to note that a higher false negative rate (FNR) for a given domain indicates that its features are less distinguishable from those of other domains—suggesting a lower degree of domain shift. $\text{PAD*}_{\text{multi}}$ consistently yields low FNR for p1000, implying that this domain is easily separable from the rest and thus exhibits high shift, which aligns with the patterns observed in BBSD. Conversely, the high FNR for cs2 in both domain-agnostic models corresponds to low shift, matching the low values reported by the MMD method, which is also domain-agnostic in nature. When comparing the domain-wise F1 scores with the $\text{PAD*}_{\text{multi}}$ class-wise results, notable discrepancies emerge. While both p1000 and nz20 exhibit similarly poor F1 performance, the $\text{PAD*}_{\text{multi}}$ metrics indicate a high degree of shift for p1000 but significantly lower shift for nz20. This may reflect similar inconsistencies between PAD* and downstream performance observed by Aubreville et al. (2021).

Issue of Domain Interdependencies:

The ability of the domain discriminator to distinguish between domains depends heavily on the encoder used—that is, the classifier responsible for extracting task-specific representations. If these representations encode domain-specific features, PAD* will detect a stronger domain shift. Conversely, if the representations are more invariant to domain characteristics, the discriminator will struggle to separate domains, resulting in a lower measured shift. Ideally, task-relevant features should not overlap with domain-specific artifacts—such as scanner-induced variations or acquisition noise—but in practice, this separation is unlikely to hold. The $\text{PAD*}_{\text{multi}}$ extension presented in this work exhibits a more severe limitation in its results: For the fine-tuned ResNet18 domain discriminator, most error values align well with

the BBSD results—except for nz20, which shows a surprisingly high classification error. This would suggest low domain shift for nz20, directly contradicting the strong shift patterns observed in BBSD. However, examining the false negative rate (FNR) and false discovery rate (FDR) offers a potential explanation: while FNR and FDR are generally consistent across domains, nz20 stands out with a notably higher FNR than FDR, whereas p1000 shows the opposite trend. This suggests the model may have developed a bias toward p1000, leading it to misclassify some nz20 instances as p1000, thereby inflating nz20's FDR. This discrepancy reveals domain interdependencies as a possible vulnerability of the multi-class version of the PAD* metric, which may skew the result and obscure true shift patterns. Although the class-wise metrics used in the experiments retain much of their interpretability despite these limitations, training separate discriminators for each domain pair, as done in the original PAD*, is likely more robust and yields results that allow more straightforward interpretation.

DINO and DINOv2: Shift Discrepancy. For MMD, DINO and DINOv2 surprisingly exhibit some level of disagreement with regard to the overall magnitude of the MMD values as well as the domain pairs that exhibit shift. For BBSD, a notable exception is the gt450–cs2 pair, where the DINO-based model reports a pronounced KS distance, while the DINOv2-based model does not. This discrepancy is likely reflected in the 2D PCA visualisations, where the gt450 model's representations differ notably between the two models, particularly in the clustering patterns of the cs2 data points 17. Despite being tested using the same kernel and the models generating features of same dimensionality, this phenomenon, along with the generally lower MMD scores observed for DINOv2, may be explained by differences in feature geometry, i.e., the spatial arrangement of feature vectors. This interpretation is supported by Figure 16, where DINOv2 shows greater domain overlap than DINO, potentially leading to smaller kernel distances and thus lower MMD values.

Distortion of Correlation. Although not domain-related, it is important to note some limitations concerning the correlation values that were calculated throughout the experiments to measure the agreement of architectures. When computing Pearson correlations between DINO and DINOv2 MMD values, gt450's extreme outlier distances inflate the correlation coefficient, masking the fact that their relative rankings often diverge. As a result, gt450's large MMD values both drive up the coefficient of variation for the other domains and dominate the Pearson metric, suggesting that the architectures show a high level of agreement.

Alternative correlation measures like Spearman's rho might better capture agreement between the extractors. However, since the correlations throughout the results are computed over very small vectors (size 4 or 5), the resulting values have generally limited interpretability.

It is important to note that these findings reflect the behaviour of specific model architectures under the conditions tested. Caution is advised when generalising to

other architectures or pre-training strategies, especially those optimised for different domains or using different objectives.

7.3 General Insights and Hypotheses

Robust Models May Obscure Shifts. Substantial differences were observed in the magnitude of detected shift between the simple TwoLayerNN and the more advanced ResNet18 classifiers in input space. The ResNet18 consistently yields significantly lower KS distances and demonstrates both higher and more stable F1 scores across target domains, suggesting stronger generalisation. However, this robustness appears to limit its ability to reveal distributional shifts. In the experiment results, some of the shifts that are clearly reflected in the performance of the TwoLayerNN are not mirrored in the ResNet18's performance. This suggests that the magnitude of detected shift may not reflect objective shift severity, but rather the vulnerability of a given classifier to that shift.

While Kore et al. (2024) argued that aggregate performance metrics such as AUC-ROC are generally unreliable for shift detection, the results in this context reflect this mainly for more advanced and complex architectures like ResNet18. The simpler TwoLayerNN models, in contrast, exhibited closer alignment between performance degradation and BBSD-detected shifts. In view of the emergence of highly performant and robust architectures in practice, these findings underscore that traditional performance metrics alone may be insufficient for detecting or quantifying domain shifts.

BBSD Asymmetry Highlights Source Domain Influence BBSD exhibited directional asymmetry. Measured shifts between domain pairs varied depending on which domain the classifier was trained on. In other words, the measured shift from the reference domain A to the target domain B was not always equal to the shift from reference domain B to the target domain A. While most domain pairs exhibit roughly similar values in both directions, several show substantial discrepancies, indicating that the choice of source domain can significantly influence the measured shift. This asymmetry was observed consistently across all architectures, including those with strong generalisation such as ResNet18, suggesting that it is an inherent property of the BBSD method and not necessarily a result of poor classifier performance.

Conceptually, one would expect domain shifts to be symmetric, as the shift between two domains should not depend on direction. Since BBSD relies on a trained classifier, the results indicate that both the model architecture and the choice of source domain can significantly affect the resulting KS distances. The observed asymmetries suggest that classifiers trained on certain source domains may generalise better, potentially due to those domains exhibiting more domain-invariant features or containing less noise, subsequently impacting the measured shift.

Regarding architectural dependency, ResNet18 reflects some of the shift patterns identified by other classifiers but also shows notable discrepancies. These differences

are evident not only in the magnitude of detected shifts but also in which domains are identified as exhibiting a shift.

If this behaviour generalises beyond the studied domains, it has important practical implications. In real-world settings where the source domain is fixed and labelled target data is limited, such directionality could impair reliable shift detection or quantification. Additionally, the observed disagreements between architectures raise questions about the consistency of BBSD results with regard to its black-box design. However, this limitation mainly applies when BBSD is used as an objective shift quantification tool. When used to estimate how a shift might affect a specific pre-existing classifier, the alignment of BBSD results with the classifier's performance suggests that BBSD is effective for this application use case. This flexibility could be valuable in practice, where shift detection is often intended for assessing risks to a given deployed model.

Task Awareness Is Crucial for Interpreting Shift Magnitude. MMD did not always reflect the same shift patterns identified by BBSD or classifier-based performance. This discrepancy likely stems from a fundamental distinction: MMD is symmetric and task-agnostic, comparing global differences in feature distributions without considering how these differences influence model decisions. In contrast, BBSD is task-aware and surfaces shifts that affect a model's predictive behaviour—such as changes in class probability distributions or decision boundaries—which MMD might overlook. Consequently, BBSD is more likely to highlight shifts that are relevant to downstream performance, while MMD may capture fundamental, but potentially task-irrelevant distributional differences.

This distinction is further supported by the $\text{PAD}^*_{\text{multi}}$ results, which explicitly compare how task-specific and task-agnostic domain discriminators rank the difficulty of each domain. Interestingly, the task-agnostic models show strong agreement in their rankings despite using different feature extractors (DINO vs. ResNet18), suggesting they capture consistent structural differences in the data. In contrast, the fine-tuned ResNet18—although architecturally identical to the pre-trained version—produces a markedly different domain ranking, underscoring how task-specific fine-tuning changes the feature space to suit the class characteristics that matter for classification.

Overall Implications: Taken together, these findings suggest that shifts identified by task-agnostic methods like MMD are not necessarily aligned with shifts that impact task performance. Therefore, relying solely on task-agnostic measures may lead to overestimating the practical effect of domain shifts. This reinforces the importance of selecting appropriate feature encoders for MMD and highlights the complementary value of combining task-aware and task-agnostic methods, as advocated by Roschewitz et al. (2024). It is also important to note that while some methods reported relatively severe shifts for certain domains, these results should not be interpreted as absolute measures of shift severity, since the magnitude of the

reported values was shown to depend strongly on the architecture used to extract the features. Moreover, in the absence of ground truth for real-world (non-synthetic) domain shifts, it remains unclear how a "true" shift is quantified. Beyond basic visual inspection of the input images, there are no clear criteria for how different shifts should manifest in the outputs of shift assessment methods.

7.4 Dataset-Specific Challenges and Limitations

The chosen "Multi-Scanner Canine Cutaneous Squamous Cell Carcinoma Histopathology Dataset" is well-suited for investigating scanner-induced data shift, as it comprises 44 anatomical samples, each digitised using five different scanners. This setup ensures that anatomical content remains identical across scans, allowing scanner-specific differences to be isolated more reliably. However, several practical limitations should be noted that may, to some extent, account for the observed variability in model behaviour across different source domains:

Unaligned training set sizes. The varying slide size, pixel ratio, and sample placement within each slide pose significant challenges. The preprocessing method divides each whole slide image into 224×224 pixel patches to ensure complete coverage and enable their direct use in training. As a result, the number of patches per slide varies, leading to differences in split set size across scanners 3.

Thus, while class distributions were balanced before training the BBSD classifiers to mitigate class bias, the total training set sizes differed between scanners. Since each classifier is trained solely on one scanner's data, this does not directly affect the class distribution of model predictions. However, varying training set sizes may influence the classifier's behaviour and its ability to detect shifts. If this has a significant impact remains speculative because, given the simplicity of the TwoLayerNN and the fact that all datasets exceed 10,000 samples (see table 3), moderate size differences are unlikely to have a major effect.

Imprecision in Labelling. A further issue lies in how patches are labelled: any patch containing even a small portion of tumor tissue is labelled as a tumor patch. This rough labelling approach may introduce considerable label noise, as patches are labelled positive even if they contain only a minimal amount of tumor tissue. Such weakly informative or ambiguous samples can negatively impact a model's ability to learn meaningful, class-specific features, potentially leading it to overfit irrelevant visual patterns. This can degrade both performance and generalisation ability. However, it's important to note that optimal classification performance was not the primary objective of these experiments. Crucially, the classifiers still produced invertible confusion matrices, fulfilling the necessary condition for applying BBSD.

Sampler-induced duplicates. Another concern is the behaviour of the weighted random sampler. Since the percentage of tumor-patches out of all patches is roughly 40% for each unbalanced subset across scanners (see table 3), the sampler draws samples with replacement from both classes to restore the original unbalanced dataset size, likely resulting in duplicate samples. For a simple architecture like the TwoLayerNN, this duplication may cause overfitting to repeated samples or lead to unstable training dynamics. As mentioned in the methodology section, some training runs failed to learn effectively from the outset and showed no improvement over additional epochs. However, the classifier's limited performance may also be attributed to the simplicity of the architecture relative to the task complexity, rather than sampling issues alone. Furthermore, balancing the classes in this way still proved advantageous compared to training on the unbalanced data, as the simple TwoLayerNN architecture showed a strong tendency to overpredict the majority class, likely because this was an easier or perhaps the only viable strategy for minimizing overall error, given the difficulty of the task and the model's limited ability to reliably learn and interpret tumor-specific features, as discussed previously.

Rotation as a shift that most models seem sensitive to. The tissue samples from the `p1000` scanner appear to be consistently rotated by 90 degrees in the original dataset. This rotation could pose a challenge for the classifier, particularly if it lacks rotation-invariant representations and could also explain why the `p1000` domain consistently showed high shift values across methods. Rotation is often used as a data augmentation method, introducing variation that models can learn from. In the context of the conducted experiments, it is likely that the rotation altered spatial structure in a way that set it apart from the other scanner domains. When such variation is consistently associated with a particular domain, it may act as a form of scanner-specific shift.

Overall, the data handling decisions were influenced by practical constraints such as complexity and time. The methodology prioritised a balanced and representative sampling of each domain, while maintaining compatibility with the chosen model architectures. Optimizing classification performance was not the primary objective. Rather, the focus was on creating a consistent base for evaluating domain shifts.

7.5 Future Work

While this study explored several domain shift quantification methods, the focus was on comparing the behaviour of the different methods and implications on practical application remain limited. Future work should investigate the effectiveness of foundation models when fine-tuned on large-scale medical imaging datasets to assess whether task-specific adaptation improves their robustness to scanner-induced shifts. Additionally, evaluating models specifically developed for medical imaging could offer deeper insights into their sensitivity to domain shifts, especially in comparison to general-purpose architectures. Until recently, access to such specialised models was limited. However, this landscape is changing, with initiatives like the

Health AI Developer Foundations (HAI-DEF) suite, that now provide pre-trained, open-weight foundation models for various medical imaging domains—including radiology, dermatology, and pathology—as well as other healthcare modalities (Kiraly et al., 2024).

Another important direction involves developing more explainable and interpretable shift detection methods. Given BBSD's sensitivity to the choice of underlying classifier and its limitations as a black-box approach, incorporating explainable AI techniques—such as the representation shift method proposed by Stacke et al. (2020)—may help clarify the sources of domain shifts and guide appropriate mitigation strategies. This approach may prove more insightful than evaluating synthetic shifts, as there is a vast number of different shift sources and previous work has shown them to have varying impact (Kore et al., 2024).

Although the patch-grid approach enabled simple and lightweight slide preprocessing, it also introduced some variability that may have affected the experiments in unintended ways. To address these histopathology-specific data-preprocessing challenges in future work, it will be important to adopt stricter labelling criteria—such as requiring that the proportion of class-relevant content in a patch exceeds a defined threshold before assigning a label. Even more impactful may be adjustments to the preprocessing pipeline itself, particularly how patches are extracted. Ensuring that anatomically corresponding regions are captured across all scanners—e.g., through segmentation—would improve label quality, better isolate scanner-induced shifts, and subsequently enable fine-grained measurement of domain shift at the level of individual slides or patches. This would significantly enhance the precision and reliability of shift detection and analysis.

8 Conclusion

This work explored methods to assess scanner-induced domain shifts using a dataset of 44 histopathology samples, each digitised by five different whole slide scanners to isolate acquisition-related variability. The main objective was to investigate how such scanner-induced variability—which may be visually subtle—can be detected and interpreted across both input and latent feature spaces, using a range of methodological perspectives.

Three complementary domain shift detection techniques were evaluated: BBSD, which compares softmax output distributions of task-specific classifiers; MMD, which measures distances between feature vector distributions; and a multi-class variant of PAD*, which trains domain discriminators on both task-specific and task-agnostic features. These methods were applied to classifier outputs and feature representations from simple neural networks, a fine-tuned ResNet18, and off-the-shelf foundation models such as DINO and DINOv2.

The results indicate that task-specific approaches like BBSD and PAD* are heavily influenced by the behaviour of the underlying classifiers. BBSD showed directional

asymmetry in measured shifts, varying depending on the domain used for training the classifier, suggesting these methods are better interpreted as proxies for estimating performance degradation on unlabelled target data rather than objective quantifiers of domain shift. In contrast, task-agnostic methods such as MMD highlighted differences that may not always have practical relevance for downstream tasks, emphasizing that shift detection is not inherently objective and must be interpreted in the context of specific use cases.

From a model perspective, more advanced architectures such as ResNet18 exhibited robust generalisation across scanner domains, with limited performance degradation despite measurable shifts. This finding highlights a critical distinction: a detectable domain shift does not necessarily imply compromised model performance, particularly for well-trained, robust models. Off-the-shelf foundation models, however, showed limited effectiveness in extracting task-relevant features without fine-tuning, indicating that their assumed generalisability may not yet fully extend to specialised histopathology tasks.

It is important to note that while this study deliberately isolated scanner-induced domain shift, real-world shifts often involve multiple interacting factors such as biological variability, staining differences, and annotation inconsistencies. Additionally, some aspects of the preprocessing pipeline—such as patch extraction, labelling criteria and dataset balancing—may have influenced the manifestation or detectability of shift and should be considered when interpreting results. Moreover, none of the evaluated methods provide a fully objective or direct quantification of how much downstream performance will degrade due to shift. As such, the practical impact of scanner-induced domain shift—and thus the relevance of detecting it—varies considerably depending on the model, task, and context.

In practice, the development of machine learning models for medical imaging is advancing, and ongoing progress in domain generalisation and domain adaptation provides researchers with increasingly powerful tools for mitigating distributional shifts. However, the effectiveness of these methods often depends on how well they align with the specific type of shift encountered. In light of these findings, the methods studied here should be understood as lightweight estimators that capture complementary aspects of domain shift, rather than definitive quantification tools. Future work integrating explainable AI techniques may improve both the accuracy and interpretability of shift detection, enabling better identification of the underlying sources of shift and more informed model development.

Overall, these results underscore that understanding and assessing scanner-induced domain shift requires a nuanced approach that balances methodological thoroughness with context-aware interpretation—validating the focus of the thesis on exploring and evaluating shift detection methods beyond absolute quantification and towards practical insights on model behaviour and reliability.

A Appendix

A.1 Experiment Visualisations

A.2 Classifier Performance

A.3 BBSD

A.4 MMD

A.5 Other

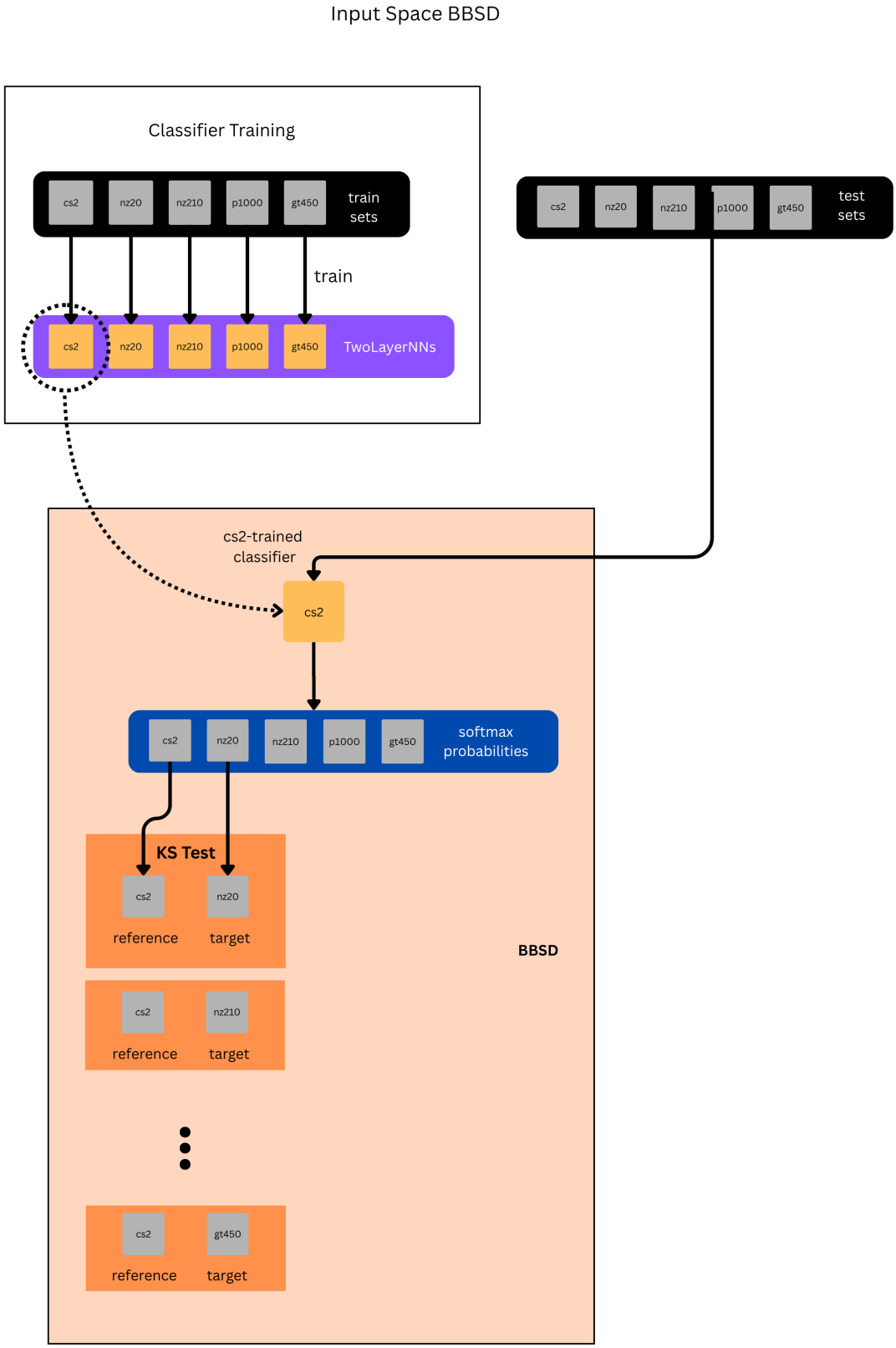

Figure 22: Visualisation of the input-space BBSD experiment. For simplicity, only the version with TwoLayerNN as the task classifier is visualised here.

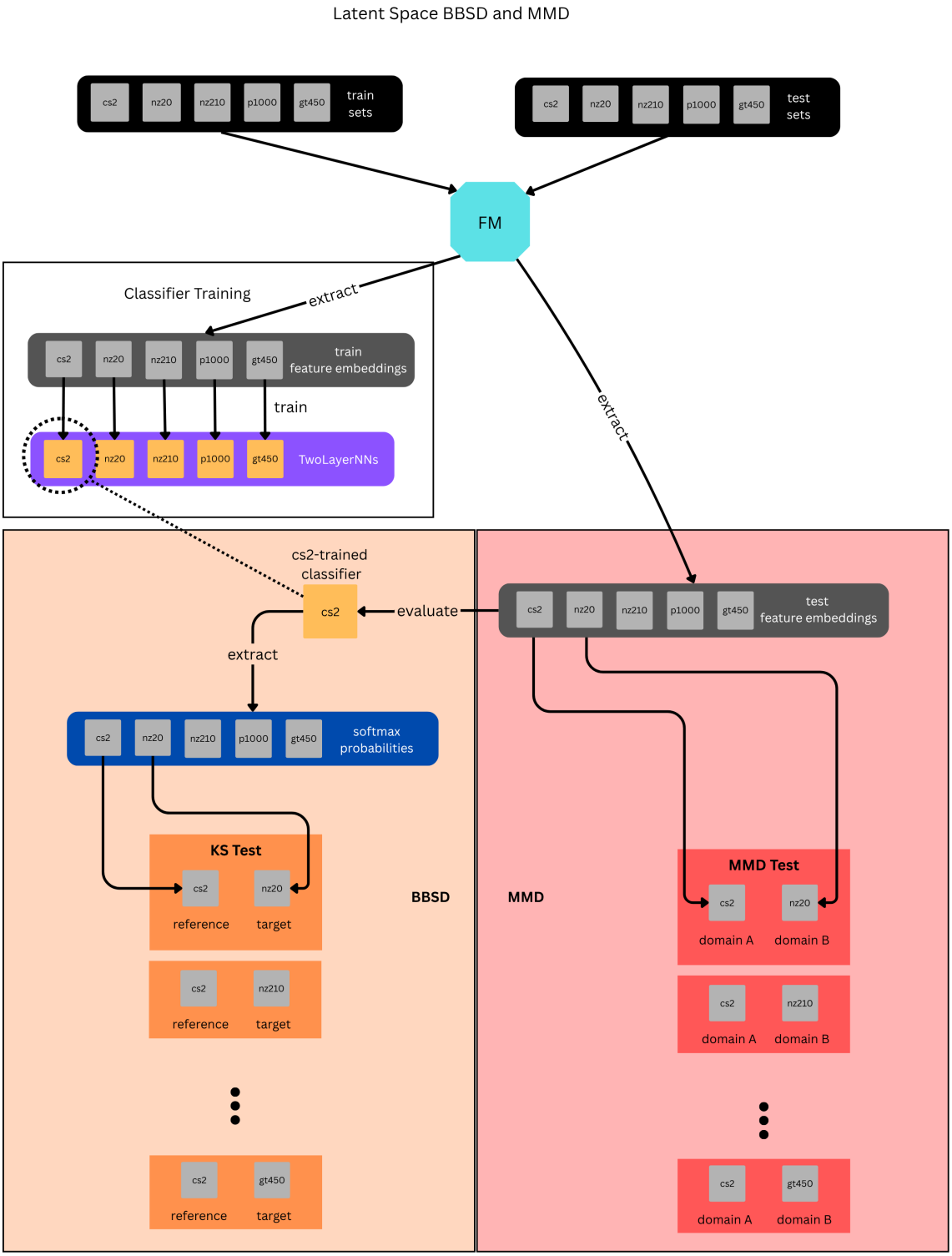

Figure 23: Visualisation of the latent-space BBSD and MMD experiments. FM denotes either one of the two foundation models used: DINO or DINOv2.

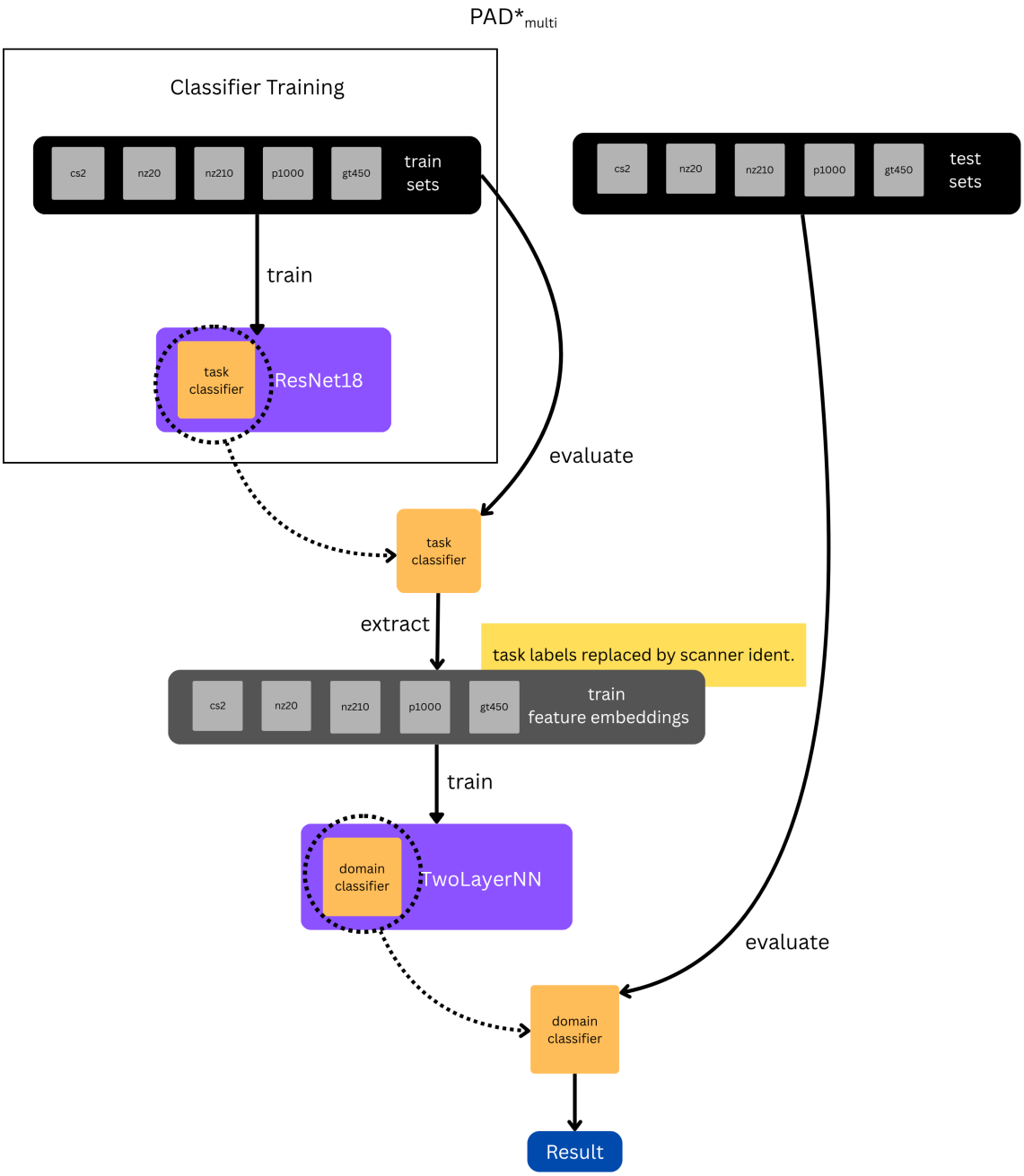

Figure 24: Visualisation of the $PAD*_{multi}$ experiment. For simplicity reasons, only one of the three versions is visualised here: ResNet18 as task-specific feature extractor.

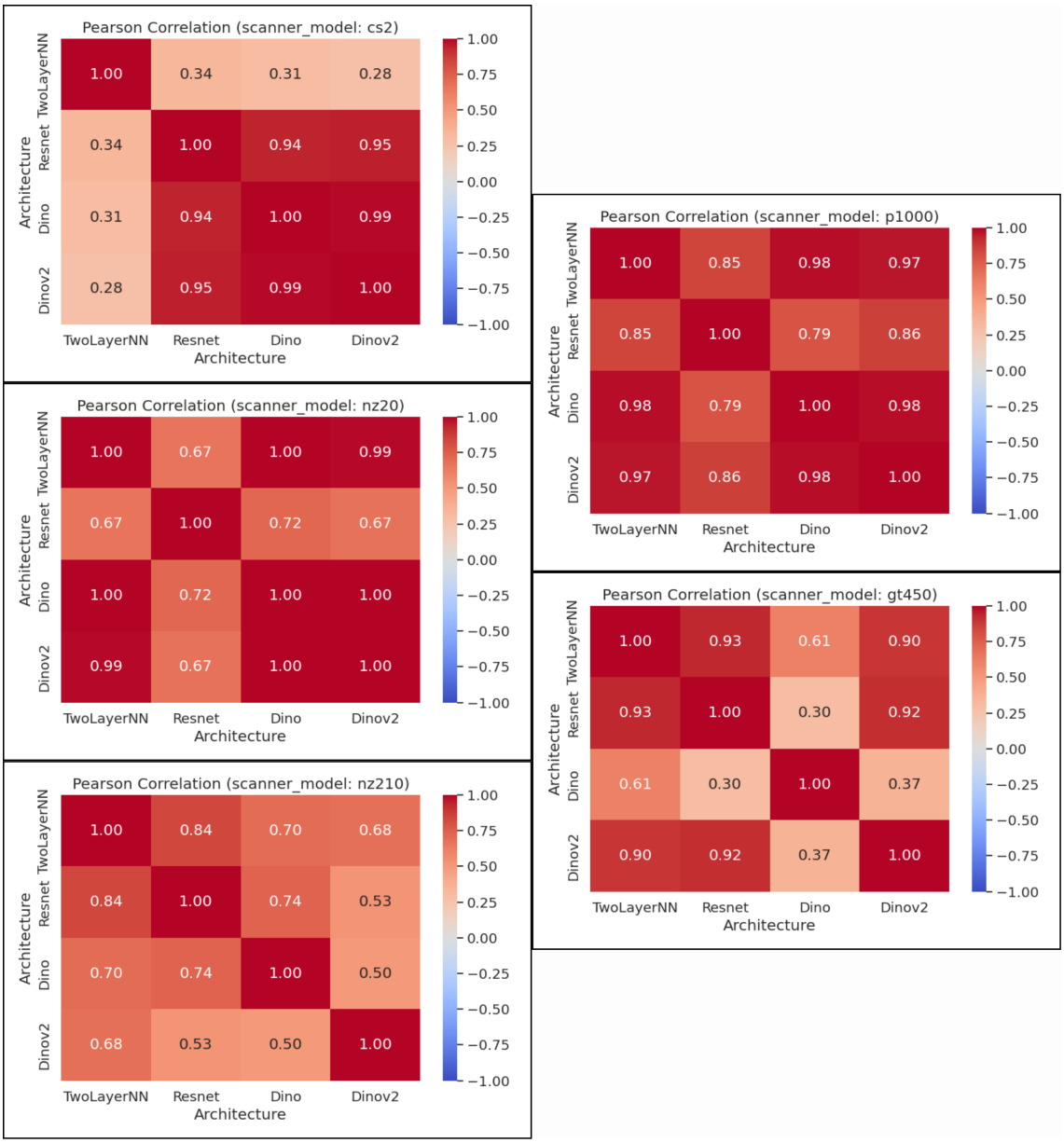

Figure 25: Pearson Correlation values between architecture pairs for F1 scores per classifier model.

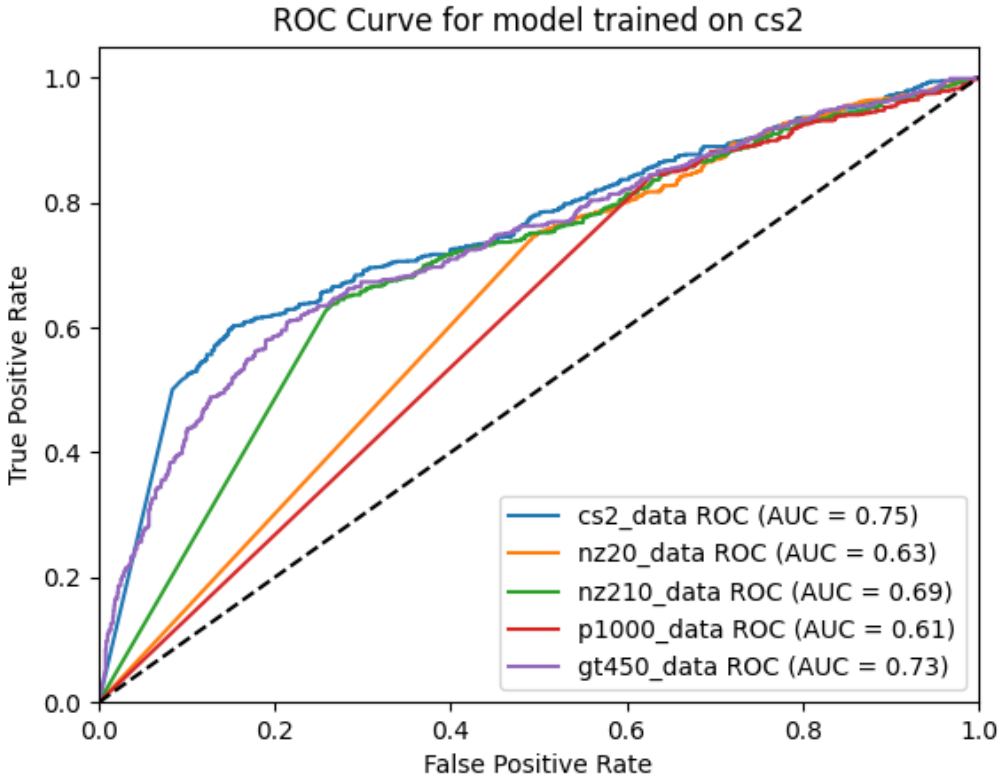

Figure 26: ROC curve of cs2-trained classifier evaluated on each scanner's test set. Linear segments are clearly visible for some target domains.

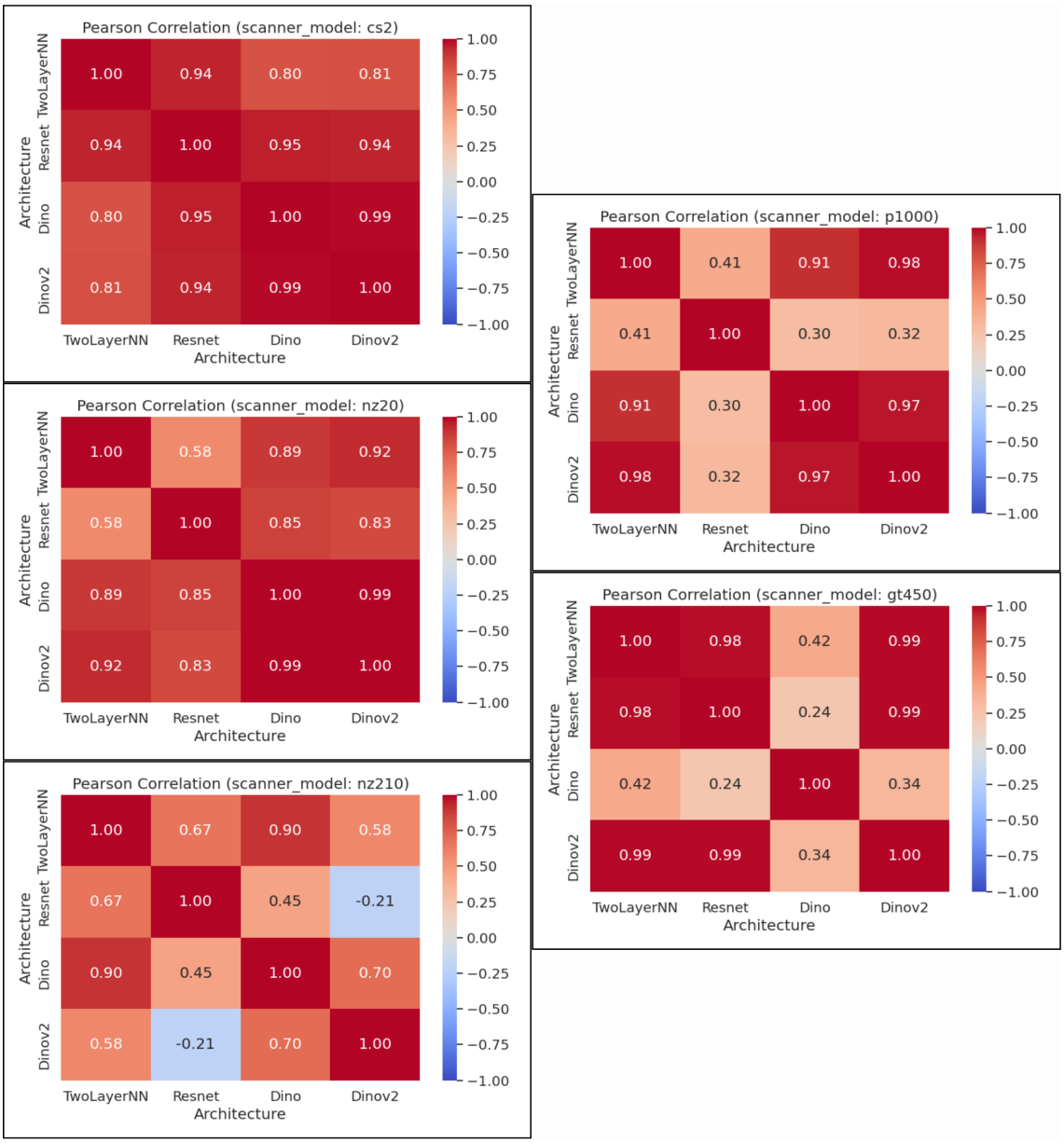

Figure 27: Pearson Correlation values between architecture pairs for KS-test values per classifier model.

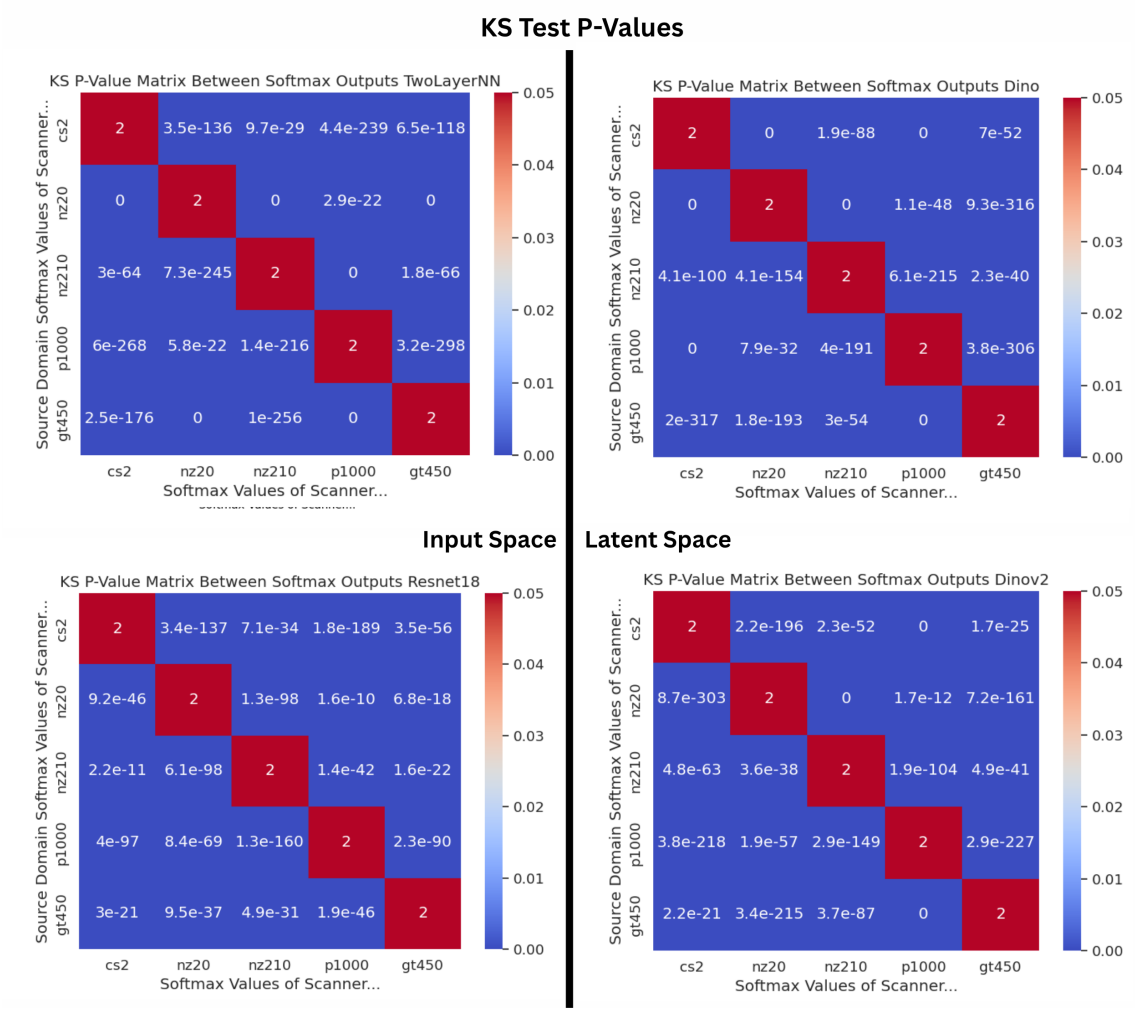

Figure 28: BBSD KS-test p-value heatmap matrices for input-space classifiers (left) and latent-space classifiers (right). Maximum of the y-scale (0.05) denotes the used significance threshold.

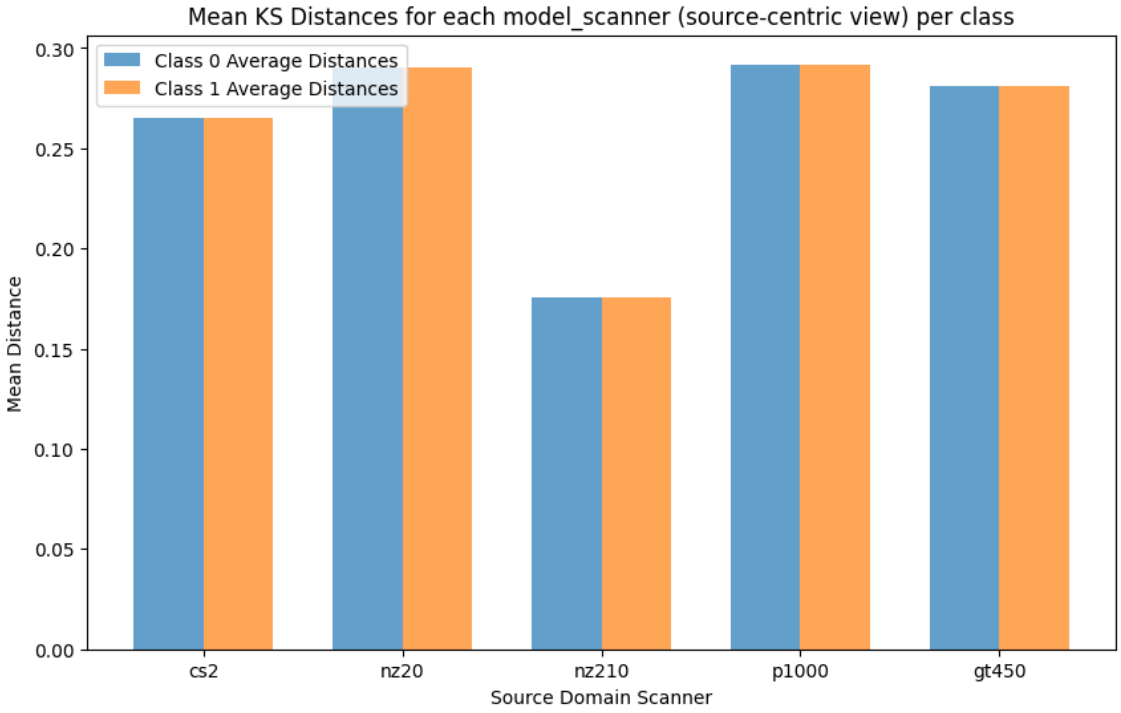

Figure 29: Class-wise mean KS distances for each source domain

Table 6: F1 scores of models trained on one scanner (Model) and tested on another (Test), across different model architectures (input and latent separated by the vertical line). Same-scanner (source domain) F1 scores are highlighted in yellow. F1 scores < 0.5 are coloured red.

F1 Scores

Model	Test	TwoLayerNN	ResNet18	DINO	DINOv2
cs2	cs2	0.71	0.84	0.79	0.76
cs2	nz20	0.61	0.71	0.56	0.58
cs2	nz210	0.66	0.83	0.68	0.66
cs2	p1000	0.56	0.63	0.52	0.52
cs2	gt450	0.47	0.81	0.72	0.71
nz20	cs2	0.33	0.66	0.34	0.34
nz20	nz20	0.65	0.85	0.75	0.70
nz20	nz210	0.35	0.55	0.35	0.39
nz20	p1000	0.68	0.78	0.74	0.69
nz20	gt450	0.35	0.77	0.34	0.35
nz210	cs2	0.58	0.77	0.52	0.69
nz210	nz20	0.58	0.79	0.69	0.71
nz210	nz210	0.68	0.85	0.72	0.75
nz210	p1000	0.55	0.80	0.66	0.65
nz210	gt450	0.58	0.79	0.58	0.63
p1000	cs2	0.34	0.46	0.33	0.36
p1000	nz20	0.61	0.53	0.69	0.62
p1000	nz210	0.34	0.39	0.43	0.46
p1000	p1000	0.66	0.84	0.75	0.75
p1000	gt450	0.36	0.45	0.35	0.36
gt450	cs2	0.60	0.83	0.56	0.73
gt450	nz20	0.56	0.80	0.63	0.63
gt450	nz210	0.59	0.79	0.70	0.65
gt450	p1000	0.52	0.79	0.56	0.55
gt450	gt450	0.66	0.84	0.73	0.73

Table 7: Predicted class distributions in percent (%), Accuracy, and F1 scores of TwoLayerNN models trained on data from one scanner (e.g. `cs2_model`) and evaluated on test data from all five scanners (cs2, nz20, nz210, p1000, gt450).

Train	**Test**	**Class 0 / 1 (%)**	**Accuracy (%)**	**F1 Score**
cs2	cs2	58.18 / 41.82	70.25	0.714
cs2	nz20	33.17 / 66.83	61.18	0.597
cs2	nz210	45.64 / 54.36	66.07	0.659
cs2	p1000	24.96 / 75.04	59.76	0.564
cs2	gt450	92.31 / 7.69	56.19	0.463
nz20	cs2	99.93 / 0.07	52.01	0.355
nz20	nz20	68.52 / 31.48	66.44	0.657
nz20	nz210	99.69 / 0.31	49.96	0.357
nz20	p1000	56.87 / 43.13	68.60	0.675
nz20	gt450	100.00 / 0.00	49.52	0.319
nz210	cs2	84.97 / 15.03	60.81	0.545
nz210	nz20	30.95 / 69.05	59.43	0.606
nz210	nz210	60.22 / 39.78	70.50	0.683
nz210	p1000	20.87 / 79.13	57.72	0.552
nz210	gt450	82.34 / 17.66	61.90	0.589
p1000	cs2	99.82 / 0.18	49.52	0.343
p1000	nz20	76.12 / 23.88	65.58	0.594
p1000	nz210	99.35 / 0.65	49.99	0.348
p1000	p1000	61.27 / 38.73	65.70	0.651
p1000	gt450	100.00 / 0.00	49.56	0.328
gt450	cs2	31.07 / 68.93	61.81	0.617
gt450	nz20	25.93 / 74.07	59.22	0.572
gt450	nz210	27.93 / 72.07	60.81	0.575
gt450	p1000	18.06 / 81.94	58.33	0.518
gt450	gt450	55.39 / 44.61	68.67	0.686

Table 8: KS distance scores between softmax distributions for each model architecture across scanner pairs. Values > 0.5 are highlighted in red.

Reference	Target	TwoLayerNN	ResNet18	DINO	DINOv2
cs2	cs2	0.000	0.000	0.000	0.000
cs2	nz20	0.352	0.332	0.531	0.396
cs2	nz210	0.157	0.166	0.268	0.206
cs2	p1000	0.496	0.404	0.662	0.574
cs2	gt450	0.319	0.221	0.212	0.149
nz20	cs2	0.593	0.193	0.733	0.488
nz20	nz20	0.000	0.000	0.000	0.000
nz20	nz210	0.538	0.258	0.710	0.489
nz20	p1000	0.140	0.096	0.207	0.105
nz20	gt450	0.596	0.125	0.514	0.371
nz210	cs2	0.228	0.096	0.284	0.226
nz210	nz20	0.403	0.257	0.322	0.161
nz210	nz210	0.000	0.000	0.000	0.000
nz210	p1000	0.558	0.194	0.430	0.302
nz210	gt450	0.240	0.140	0.187	0.189
p1000	cs2	0.494	0.291	0.614	0.433
p1000	nz20	0.139	0.246	0.167	0.225
p1000	nz210	0.436	0.373	0.406	0.360
p1000	p1000	0.000	0.000	0.000	0.000
p1000	gt450	0.580	0.281	0.509	0.442
gt450	cs2	0.388	0.136	0.515	0.136
gt450	nz20	0.547	0.179	0.406	0.427
gt450	nz210	0.465	0.164	0.217	0.274
gt450	p1000	0.678	0.202	0.590	0.569
gt450	gt450	0.000	0.000	0.000	0.000

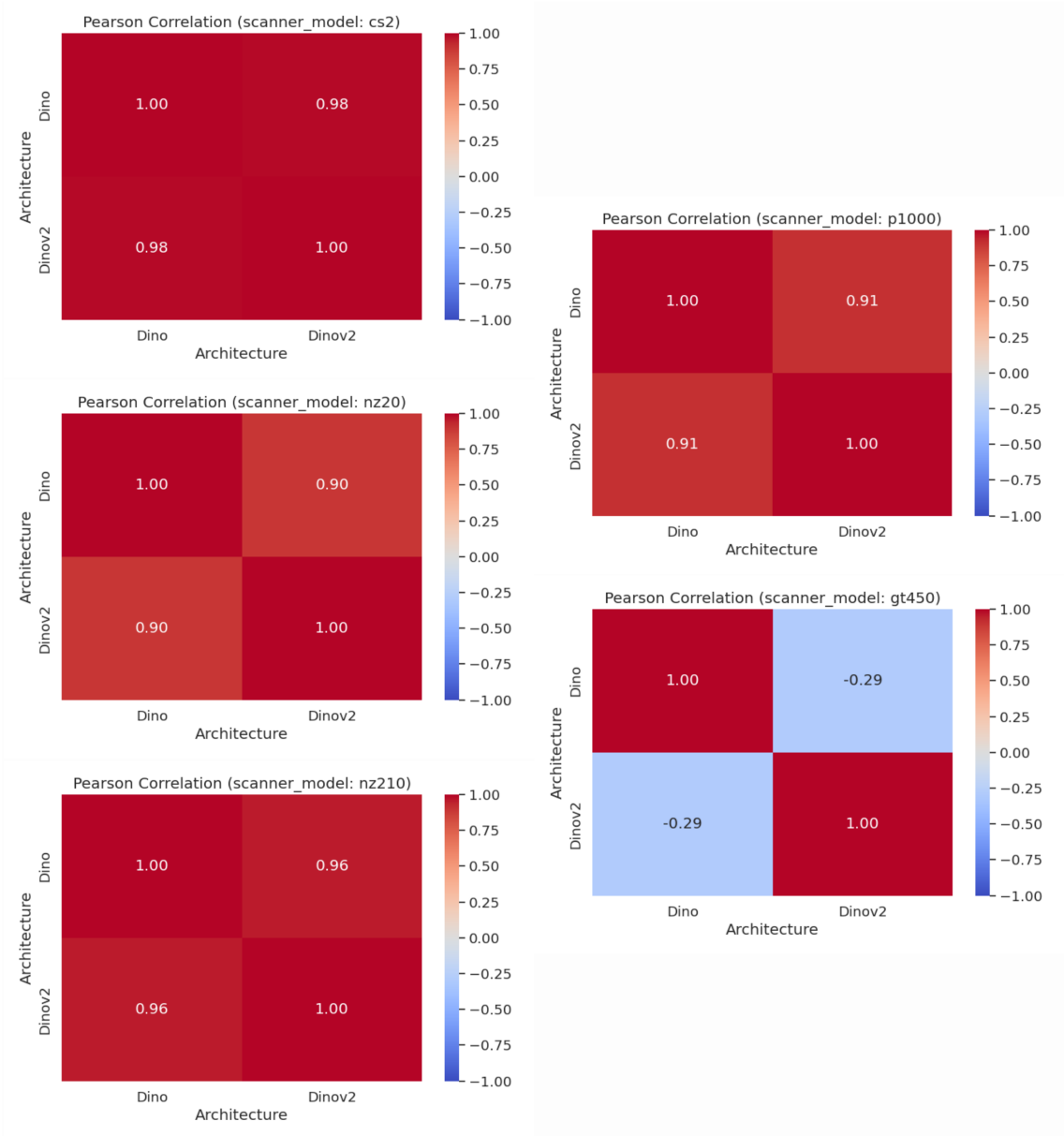

Figure 30: Pearson Correlation values between architecture pairs for MMD distances per scanner domain.

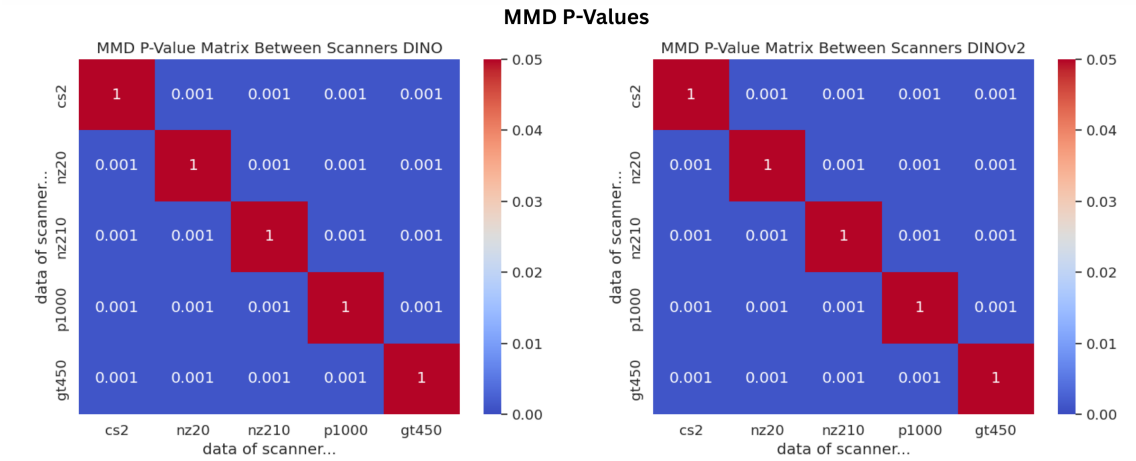

Figure 31: MMD p-value heatmap matrix per feature extractor. Maximum of the y-axis (0.05) denotes the used significance threshold.

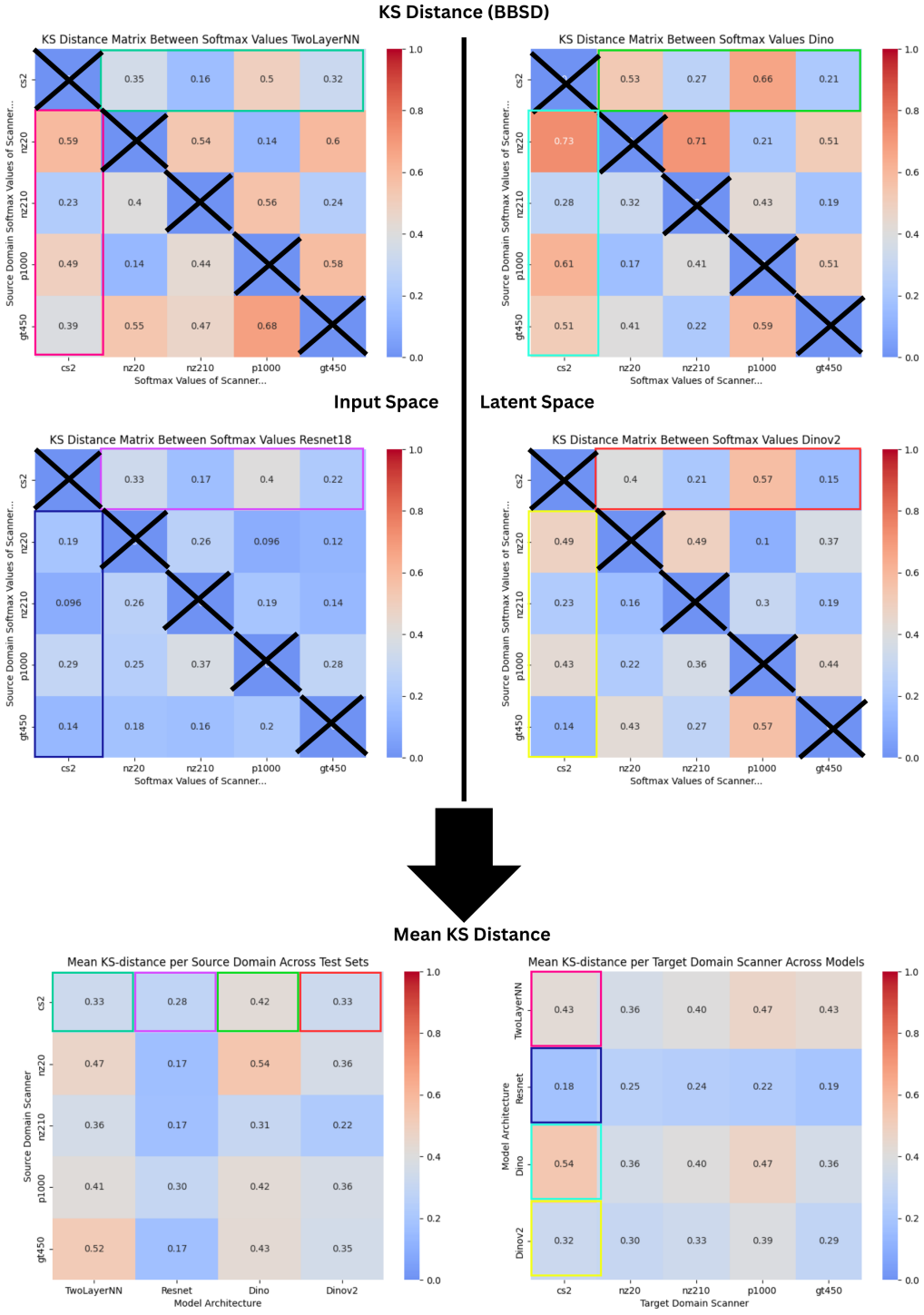

Figure 32: Visualisation of how the source- and target-centric mean BBSD values are calculated, excluding the diagonal values.

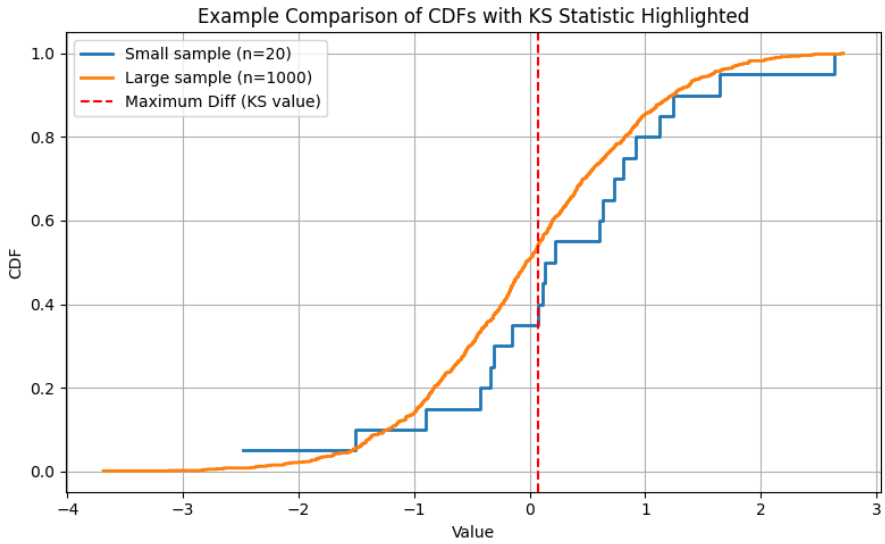

Figure 33: Exemplary visualisation of the comparison of empirical CDFs of two datasets with largely different number of samples.

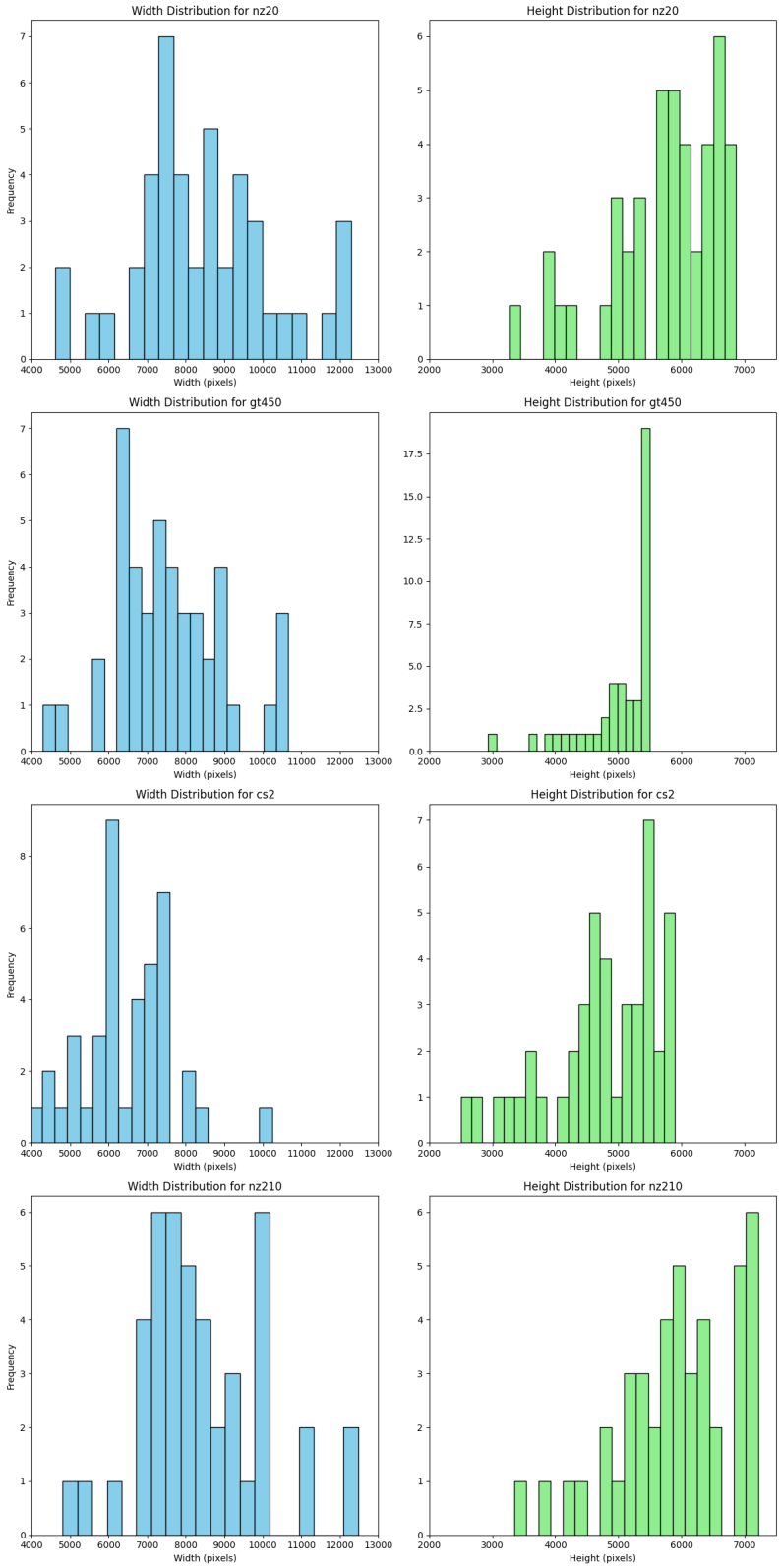

Figure 34: Height and width distributions of slides for scanners cs2, nz20, nz210 and gt450. Scanner p1000 was excluded because the aspect ratio of its slides is uniform.

Bibliography

Hervé Abdi. Coefficient of variation. *Encyclopedia of research design*, 1(5):169–171, 2010.

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

M. Afshari, S. Yasir, G.L. Keeney, R.E. Jimenez, J.J. Garcia, and H.R. Tizhoosh. Single patch super-resolution of histopathology whole slide images: a comparative study. *Journal of Medical Imaging (Bellingham)*, 10(1):017501, Jan 2023. doi: 10.1117/1.JMI.10.1.017501. Epub 2023 Jan 31.

Florence Alberge, Clément Feutry, Pierre Duhamel, and Pablo Piantanida. Detecting covariate shift with black box predictors. In *2019 26th international conference on telecommunications (ICT)*, pages 324–329. IEEE, 2019.

Marc Aubreville, Christof Bertram, Mitko Veta, Robert Klopfleisch, Nikolas Stathonikos, Katharina Breininger, Natalie ter Hoeve, Francesco Ciompi, and Andreas Maier. Quantifying the scanner-induced domain gap in mitosis detection, 2021. URL `https://arxiv.org/abs/2103.16515`.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade: Second edition*, pages 437–478. Springer, 2012.

J Martin Bland and Douglas G Altman. Multiple significance tests: the bonferroni method. *Bmj*, 310(6973):170, 1995.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Hady Elsahar and Matthias Gallé. To annotate or not? predicting performance drop under domain shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2163–2173, 2019.

Brian Everitt. The cambridge dictionary of statistics. In *The Cambridge dictionary of statistics*, pages 360–360. Cambridge University Press, 1998.

Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8): 861–874, 2006.

A.T. Feldman and D. Wolfe. Tissue processing and hematoxylin and eosin staining. In *Methods in Molecular Biology*, volume 1180, pages 31–43. Humana Press, 2014. doi: 10.1007/978-1-4939-1050-2_3.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Michael Greenacre, Patrick JF Groenen, Trevor Hastie, Alfonso Iodice d'Enza, Angelos Markos, and Elena Tuzhilina. Principal component analysis. *Nature Reviews Methods Primers*, 2(1):100, 2022.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Brian Guo, Darui Lu, Gregory Szumel, Rongze Gui, Tingyu Wang, Nicholas Konz, and Maciej A Mazurowski. The impact of scanner domain shift on deep learning performance in medical imaging: an experimental study. *arXiv preprint arXiv:2409.04368*, 2024.

James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Joseph Lawson Hodges Jr. The significance probability of the smirnov two-sample test. *Arkiv för matematik*, 3(5):469–486, 1958.

Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.

Chip Huyen. *Designing machine learning systems.* " O'Reilly Media, Inc.", 2022.

Amjad Khan, Andrew Janowczyk, Felix Müller, Annika Blank, Huu Giao Nguyen, Christian Abbet, Linda Studer, Alessandro Lugli, Heather Dawson, Jean-Philippe Thiran, et al. Impact of scanner variability on lymph node segmentation in computational pathology. *Journal of pathology informatics*, 13:100127, 2022.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Atilla P. Kiraly, Sebastien Baur, Kenneth Philbrick, Fereshteh Mahvar, Liron Yatziv, Tiffany Chen, Bram Sterling, Nick George, Fayaz Jamil, Jing Tang, Kai Bailey, Faruk Ahmed, Akshay Goel, Abbi Ward, Lin Yang, Andrew Sellergren, Yossi Matias, Avinatan Hassidim, Shravya Shetty, Daniel Golden, Shekoofeh Azizi, David F. Steiner, Yun Liu, Tim Thelin, Rory Pilgrim, and Can Kirmizibayrak. Health ai developer foundations, 2024. URL `https://arxiv.org/abs/2411.15128`.

Navid Alemi Koohbanani, Balagopal Unnikrishnan, Syed Ali Khurram, Pavitra Krishnaswamy, and Nasir Rajpoot. Self-path: Self-supervision for classification of pathology images with limited annotations. *IEEE Transactions on Medical Imaging*, 40(10):2845–2856, 2021.

Ali Kore, Elyar Abbasi Bavil, Vallijah Subasri, Moustafa Abdalla, Benjamin Fine, Elham Dolatabadi, and Mohamed Abdalla. Empirical data drift detection experiments on real-world medical imaging data. *Nature Communications*, 15 (1):1887, 2024. ISSN 2041-1723. doi: 10.1038/s41467-024-46142-w. URL `https://doi.org/10.1038/s41467-024-46142-w`.

Alex Krizhevsky. Cifar-10, 2009. URL `https://archive.ics.uci.edu/dataset/691/cifar+10`. DOI: 10.24432/C5889J.

Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

Deepti Kunte, Bram Cornelis, Claudio Colangeli, and Konstantinos Gryllias. Investigating the usage of proxy-a-distance as a measure of dataset shift detection and quantification in an automotive booming noise classification setting. In *Surveillance, Vibrations, Shock and Noise*, 2023.

Maxime W Lafarge, Josien PW Pluim, Koen AJ Eppenhof, Pim Moeskops, and Mitko Veta. Domain-adversarial neural networks to address the appearance variability of histopathology images. In *Deep Learning in Medical Image Analysis and*

Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3, pages 83–91. Springer, 2017.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207, 2013.

Anant Madabhushi and George Lee. Image analysis and machine learning in digital pathology: Challenges and opportunities. *Medical image analysis*, 33:170–175, 2016.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

C. E. Metz. Basic principles of roc analysis. *Seminars in Nuclear Medicine*, 8(4): 283–298, October 1978. ISSN 0001-2998. doi: 10.1016/s0001-2998(78)80014-2.

Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

Arshi Parvaiz, Muhammad Anwaar Khalid, Rukhsana Zafar, Huma Ameer, Muhammad Ali, and Muhammad Moazam Fraz. Vision transformers in medical computer vision—a contemplative retrospection. *Engineering Applications of Artificial Intelligence*, 122:106126, 2023.

Karl Pearson. Mathematical contributions to the theory of evolution.—iii. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 187:253–318, 1896. doi: 10.1098/rsta.1896.0007.

Fortunato Pesarin and Luigi Salmaso. The permutation testing approach: a review. *Statistica*, 70(4):481–509, 2010.

Eduardo HP Pooch, Pedro Ballester, and Rodrigo C Barros. Can we trust deep learning based diagnosis? the impact of domain shift in chest radiograph classification. In *Thoracic Image Analysis: Second International Workshop, TIA 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 8, 2020, Proceedings 2*, pages 74–83. Springer, 2020.

David Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1): 37–63, 2011.

Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019.

Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. *Advances in neural information processing systems*, 32, 2019.

Aaditya Ramdas, Sashank Reddi, Barnabas Poczos, Aarti Singh, and Larry Wasserman. On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29, 03 2015. doi: 10.1609/aaai.v29i1.9692.

Markus Ringnér. What is principal component analysis? *Nature biotechnology*, 26 (3):303–304, 2008.

Mélanie Roschewitz, Raghav Mehta, Charles Jones, and Ben Glocker. Automatic dataset shift identification to support root cause analysis of ai performance drift. *arXiv preprint arXiv:2411.07940*, 2024.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Karin Stacke, Gabriel Eilertsen, Jonas Unger, and Claes Lundström. Measuring domain shift for deep learning in histopathology. *IEEE journal of biomedical and health informatics*, 25(2):325–336, 2020.

Amos Storkey et al. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, 30(3-28):6, 2009.

Frauke Wilm, Marco Fragoso, Christof A. Bertram, Nikolas Stathonikos, Robert Klopfleisch, Andreas Maier, Katharina Breininger, and Marc Aubreville. Multi-scanner canine cutaneous squamous cell carcinoma histopathology dataset, January 2023a. URL `https://doi.org/10.5281/zenodo.7418555`.

Frauke Wilm, Marco Fragoso, Christof A Bertram, Nikolas Stathonikos, Mathias Öttl, Jingna Qiu, Robert Klopfleisch, Andreas Maier, Marc Aubreville, and Katharina Breininger. Mind the gap: Scanner-induced domain shifts pose challenges for representation learning in histopathology. In *2023 IEEE 20th international symposium on biomedical imaging (ISBI)*, pages 1–5. IEEE, 2023b.

Frauke Wilm, Marco Fragoso, Christof A. Bertram, Nikolas Stathonikos, Mathias Öttl, Jingna Qiu, Robert Klopfleisch, Andreas Maier, Katharina Breininger, and Marc Aubreville. Multi-scanner canine cutaneous squamous cell carcinoma histopathology dataset. In Thomas M. Deserno, Heinz Handels, Andreas Maier, Klaus Maier-Hein, Christoph Palm, and Thomas Tolxdorff, editors, *Bildverarbeitung für die Medizin 2023*, pages 206–211, Wiesbaden, 2023c. Springer Fachmedien Wiesbaden. ISBN 978-3-658-41657-7.

Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2272–2281, 2017.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021.

Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):4396–4415, 2022.

Declaration of Authorship

Ich erkläre hiermit gemäß §9 Abs. 12 APO, dass ich die vorstehende Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass die digitale Fassung der gedruckten Ausfertigung der Abschlussarbeit ausnahmslos in Inhalt und Wortlaut entspricht und zur Kenntnis genommen wurde, dass diese digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Place, Date	Signature